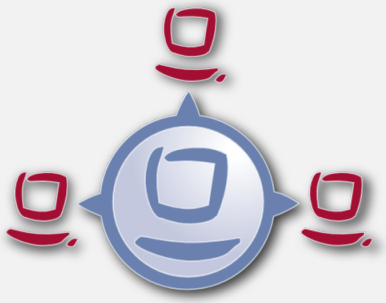

opsi-winst / opsi-script reference card (4.12.3.x)

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

Stand: March 14, 2021

Contents

For Windows [W], Linux [L] and MacOS [M]

Chapter 1

Copyright

Chapter 2

Global text constants

2.1 System directories

2.1.1 System directories [W]:

%ProgramFilesDir%: *c:\program files*

%ProgramFiles32Dir%: *c:\Program Files (x86)* //since 4.10.8

%ProgramFiles64Dir%: *c:\program files* //since 4.10.8

%ProgramFilesSysnativeDir% : *c:\program files* //since 4.10.8

%Systemroot% : *c:\windows*

%System% : *c:\windows\system32*

%Systemdrive% : *c:*

%ProfileDir% :
NT5: *c:\Documents and Settings*
NT6: *C:\users*

2.2 Common (AllUsers) directories [W]:

%AllUsersProfileDir% or %CommonProfileDir% :
NT5: *c:\Documents and Settings\All Users*
NT6: *C:\Users\Public*

%CommonStartMenuPath% or %CommonStartmenuDir% :
NT5: *c:\Documents and Settings\All Users\Startmenu*
NT6: *C:\ProgramData\Microsoft\Windows\Start Menu*

%CommonAppdataDir% :
NT5: *c:\Documents and Settings\All Users\Application Data*
NT6: *C:\ProgramData*

%CommonDesktopDir%
NT5: *c:\Documents and Settings\All Users\Desktop*
NT6: *C:\Users\Public\Desktop*

%CommonStartupDir%
NT5: *c:\Documents and Settings\All Users\Autostart*
NT6: *C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp*

%CommonProgramsDir%

2.3 Default User directories [W]:

%DefaultUserProfileDir% //since 4.11.1.1

2.4 Current user directories [W]:

User is the logged in user or given by /usercontext.

%AppdataDir% or %CurrentAppdataDir% : //since 4.10.8.13
NT5: *c:\Documents and Settings\%USERNAME%\Application Data* NT6: *c:\users\%USERNAME%\Appdata\Roaming*

%CurrentStartmenuDir% //since 4.10.8.13

%CurrentDesktopDir% //since 4.10.8.13

%CurrentStartupDir% //since 4.10.8.13

%CurrentProgramsDir% //since 4.10.8.13

%CurrentSendToDir% //since 4.10.8.13

%CurrentProfileDir% //since 4.11.2.1

2.5 /AllNtUserProfiles directory constants [W]:

%UserProfileDir%
or
%CurrentProfileDir% // since 4.11.2.1
NT5: *c:\Documents and Settings\%USERNAME%*
NT6: *c:\users\%USERNAME%*

2.6 opsi-winst Path and Directories [W/L/M]:

%ScriptPath% or %ScriptDir%

%ScriptDrive%

%OpsiscriptDir% (since 4.12.3.6), %WinstDir%

%OpsiscriptVersion% (since 4.12.3.6), %WinstVersion% (since 4.10.8.3)

%Logfile%

%opsiScriptHelperPath% %ProgramFiles32Dir%\opsi.org\opsiScriptHelper\lib // since 4.11.3.2

%opsiTmpDir% : c:\opsi.org\tmp // since 4.11.4.3

%opsiLogDir% : c:\opsi.org\log // since 4.11.4.3

%opsidata% : c:\opsi.org\data // since 4.12.0.12

%opsiapplog% : c:\opsi.org\applog // since 4.12.0.12

2.7 Network informations [W/L/M]:

`%Host%` : value of environment variable HOST.

`%PCName%`: value of environment variable PCNAME, or if absent of COMPUTERNAME.

`%Username%` : Name of actual user.

`%IPName%` : The dns name of the pc. Usually identical with the netbios name and therefore with `%PCName%` besides that the netbios names uses to be uppercase.

`%IPAddress%` : may be the IP-Address of the machine. Use funktion `GetMyIpByTarget()` instead.

2.8 Service Data [W/L/M]

`%HostID%` : FQDN of the client

`%opsiserviceURL%`

`%opsiServer%`

`%opsiDepotId%` //since 4.11.4

`%opsiserviceUser%` FQDN used for the connection to the opsi-config-server

`%opsiservicePassword%`

`%installingProdName%`: productid //since 4.10.8

`%installingProdVersion%`: product version //since 4.10.8

`%installingProduct%` : productid (deprecated)

2.9 Functions to handle constants [W/L/M]

`replaceOpsiConstants(`<string list>`)` : `stringlist` //since 4.12.3.6 [W/L/M]

`replaceOpsiConstants(`<string>`)` : `string` //since 4.12.3.6 [W/L/M]

Chapter 3

In Primary Sections

3.1 Kinds of Primary Sections [W/L/M]:

`[Initial]`

`[Actions]`

`[sub`<identifier>`]`

`sub` <file name>

`[ProfileActions]` [W]

3.2 Winst control [W/L/M]:

`encoding=`<encoding> // (default is system encoding) since 4.11.4.2

`LogLevel` (deprecated)

`SetLogLevel` = <number> or `SetLogLevel` = <string> // (default=6)

```
SetLogLevel = 7
SetLogLevel = "7"
```

`ExitOnError` = <boolean value> // (default=false)

`ScriptErrorMessages` = <boolean value> // (default=true)

`FatalOnSyntaxError` = <boolean value> // (default=true) since 4.11.3.2

`FatalOnRuntimeError` = <boolean value> // (default=false) since 4.11.3.2

`AutoActivityDisplay` = <boolean value> // (default=false); if true shows a marquee (endless) progressbar while winbatch/dosbatch sections are . //since 4.11.4.7

`forceLogInAppenMode` = <boolean value> // (default=false); if true log will be send in append mode . //since 4.12.3.6

`Message` <string> or `Message` = <const string>

`ShowMessageFile` <string>

`ShowBitMap` [<file name>] [<sub title>]

`comment` <string> or `comment` = <const string>

`LogError` <string> or `LogError` = <const string>

`LogWarning` <string> or `LogWarning` = <const string>

`includelog` <file name> <tail size> //since 4.11.2.1 [W/L/M]

`includelog` <file name> <tail size> [<encoding>] //since 4.11.4.1 [W/L/M]

```
includelog "%Scriptpath%\test-files\10lines.txt" "5"
```

`SetConfidential` <secret string> //since 4.11.3.5 [W/L/M]

`asConfidential(` <secret string expression> `)` //since 4.12.0.16 [W/L/M]

`Pause` <string> or `Pause` = <const string>

`Stop` <string> or `stop` = <const string>

`include_insert` <file name> // since 4.11.3

`include_append` <file name> // since 4.11.3

`NormalizeWinst` // (set normal window state) since 4.11.3

`IconizeWinst` // (set minimized window state)

`MaximizeWinst` // (set maximized window state) // since 4.11.5.1

`RestoreWinst` // (restore last window state)

`SetSkinDirectory` <path to skin.ini> // since 4.11.3.5

3.3 Variables [W/L/M]:

3.3.1 Strings

`DefVar` <variable name>

`Set` <variable name> `=` <value>

3.3.2 Stringlists

`DefstringList` <variable name>

3.4 Functions

3.4.1 String functions

3.4.1.1 Important

`GetOS` // *Linux* or *Windows_NT* [W/L/M]

`getLinuxDistroType` // *debian* or *redhat* or *suse* (see `getLinuxVersionMap`) [L]

`GetMsVersionInfo` //Windows Version Information [W]

`GetSystemType` //OS Architecture ("64 Bit System" or "x86 System") [W/L/M]

`getRegistryValue(`<keystr>, <varstr> [, <access str>]`)` : string //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

`GetRegistrystringvalue ("`[key] var`")` [W]

`GetRegistryStringValue32 ("`[key] var`")` //since 4.10.8 [W]

`GetRegistryStringValue64 ("[key] var")` //since 4.10.8 [W]

`GetRegistryStringValueSysNative ("[key] var")` //since 4.10.8 [W]

`GetValueFromInifile` (file, section, key, default value) [W/L/M]

```
GetValueFromInifile("myfile","mysec","mykey","")
```

`GetProductProperty` (<PropertyName>, <DefaultValue>) [W/L/M]

`GetConfidentialProductProperty` (<PropertyName>, <DefaultValue>) //since 4.11.5.2 [W/L/M]

`trim`(<string>) [W/L/M]

`lower`(<string>) [W/L/M]

`upper`(<string>) [W/L/M]

`unquote`(<string>,<quote-string>) //since 4.11.2.1 [W/L/M]

`unquote2`(<string>,<quote-string>) //since 4.11.5.2 [W/L/M]

`stringReplace`(<string>, <oldPattern>, <newPattern>) //since 4.11.3 [W/L/M]

`strLength`(<string>) //since 4.11.3 [W/L/M]

`strPos`(<string>, <sub string>) //since 4.11.3 [W/L/M]

`strPart`(<string>, <start pos>, <number of chars>) //since 4.11.3 [W/L/M]

`getValue`(<key string>, <hash string list>) [W/L/M]

`getValueBySeparator`(<key string>,<separator string>,<hash string list>) //since 4.11.2.1 [W/L/M]

`getValueFromFile`(<key string>, <file name>) //since 4.11.4.4 [W/L/M]

`getValueFromFileBySeparator`(<key string>,<separator string>,<file name>) //since 4.11.4.4 [W/L/M]

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

3.4.1.2 Special: License Management

`DemandLicenseKey(` poolId [, productId [,windowsSoftwareId]])

```
set $mykey$ = DemandLicenseKey ("", "office2007")
```

`FreeLicense (`poolId [, productId [,windowsSoftwareId]])`

```
set $result$ = FreeLicense("", "office2007")
```

3.4.1.3 Special: Usercontext / loginscripts [W]:

`GetUserSID`(<Windows Username>)

`GetLoggedInUser` //since 4.11.1.2

`GetUsercontext` //since 4.11.1.2

`GetScriptMode` possible values *Machine*,*Login* //since 4.11.2.1

`saveVersionToProfile` - save productversion-packageversion to local profile //since 4.11.2.1

`readVersionFromProfile` : string - read productversion-packageversion from local profile //since 4.11.2.1

`scriptWasExecutedBefore` : boolean - is true if saved and running productversion-packageversion are identical //since 4.11.2.1

3.4.1.4 Other

`GetHostsName (`<hostaddress>`)` [W/L/M]

`GetHostsAddr (`<hostname>`)` [W/L/M]

`ExtractFilePath (`<path>`)` [W/L/M]

`calculate(`<arithmetic string expression>`)` // since 4.11.3.5 : knows: `+-*/()` [W/L/M]

`DecStrToHexStr (` <decstring>, <hexlength>`)` [W/L/M]

`HexStrToDecStr (`<hexstring>`)` [W/L/M]

`base64EncodeStr(`<string>`)` [W/L/M]

`base64DecodeStr(`<string>`)` [W/L/M]

`convert2Jsonstr(`<string>`)` //since 4.10.8.3

`RandomStr` [W/L/M]

`RandomStrWithParameters` [W/L/M]

`RandomIntStr(`<number str>`) : string` [W/L/M]

`CompareDotSeparatedStrings(`<string1>, <string2>`) : string` [W/L/M]

`CompareDotSeparatedNumbers(`<string1>, <string2>`) : string` [W/L/M]

`EnvVar (`<environment variable>`)` [W/L/M]

`ParamStr` [W/L/M]

`getDiffTimeSec` (Time in seconds since last marktime) //since 4.11.3 [W/L/M]

`SidToName(`<well known sid>`)` //since 4.11.3: gives localized name of the sid [W]

`GetMyIpByTarget(`<target ip addr>`) : string` //since 4.11.3.2 /4.11.6 [W/L/M]

`GetIpByName(`<ip addr / ip name>`)` //since 4.11.3.2 [W/L/M]

`reencodestr(`<str>, <from>, <to>`)` //since 4.11.4.2 [W/L/M]

`strLoadTextFile (` <filename> `)` //since 4.11.4.6 [W/L/M]

`strLoadTextFileWithEncoding (` <filename> , <encoding>`)` //since 4.11.4.6 [W/L/M]

`GetShortWinPathName(<longpath string>)` //since 4.11.5.2 [W]

`stringinput(`< message str>,< boolstr confidential>`) : string` //since 4.12.1.2 [W/L/M]

`which(`<command in path>`) : string` (command with path) //since 4.12.3.6 [W/L/M]

`replaceOpsiConstants(`<string>`) : string` //since 4.12.3.6 [W/L/M]

3.4.1.5 Deprecated

`GetNtVersion` Deprecated - please use `GetMsVersionInfo` [W]

`IniVar (`<key>`)` : (deprecated; use GetProductProperty) [W]

`SubstringBefore (`<string1>, <string2>`)` (deprecated; use `splitString` / `takestring`) [W/L/M]

3.4.2 String list functions

3.4.2.1 Important

splitString (<string1>, <string2>) [W/L/M]

```
set $list1$ = splitString ("\\server\share\dir","\")
```

splitStringOnWhiteSpace (<string>) [W/L/M]

loadTextFile (<file name>) [W/L/M]

loadUnicodeTextFile (<file name>) [W]

loadTextFileWithEncoding(<file name> , <encoding>) //since 4.11.5 [W/L/M]

composeString (<string list>, <Link>) [W/L/M]

takeString (<index>, <list>) [W/L/M]

setStringInListAtIndex(<newstring>,<list>,<indexstr>) : stringlist //since 4.11.6 [W/L/M]

takeFirstStringContaining(<list>,<search string>) [W/L/M]

getOutStreamFromSection (<dos section name>) [W/L/M]

```
set $list$= getOutStreamFromSection ('DosInAnIcon_try')
```

shellCall (<command string>) : stringlist (output) //since 4.11.4.2 [W/L/M]

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

getReturnListFromSection (<xml section name>) [W/L/M]

getListContaining(<list>,<search string>) [W/L/M]

getListContainingList(<list1>,<list2>) //since 4.11.3.7 [W/L/M]

count (<list>) [W/L/M]

emptylist (<list>) //since 4.11.3.7 [W/L/M]

for %<identifier>% in <list> do <one statement | sub section> [W/L/M]

```
for %s% in $list1$ do sub_test_string
```

GetProcessList //since 4.11.1.2; gives list of exename;pid;dom/user [W/L/M]

getProductPropertyList(<propname>,<default value>) //since 4.11.3 [W/L/M]

getRegistryKeyList32(<regkey>) //since 4.11.3 [W]

getRegistryKeyList64(<regkey>) //since 4.11.3 [W]

getRegistryKeyListSysnative(<regkey>) //since 4.11.3 [W]

getRegistryVarList32(<regkey>) //since 4.11.3 [W]

getRegistryVarList64(<regkey>) //since 4.11.3 [W]

getRegistryVarListSysnative(<regkey>) //since 4.11.3 [W]

getProfilesDirList //since 4.11.3.2 [W/L/M]

listFiles (<Path>, <Searchmask> , <SearchSubDirectories>, <[Redirection]>) : stringlist //since 4.12.3 [W/L/M]

```
Set $Filelist$ = listFiles("C:\windows\system32","*.Devices.*.dll","False","64bit")
```

replaceOpsiConstants(<string list>) : stringlist //since 4.12.3.6 [W/L/M]

3.4.2.2 Infomaps

`GetLocaleInfoMap` [W]

`GetMSVersionMap` [W]

`getLinuxVersionMap` //since 4.11.4 [L]
keys are (example):

```
Distributor ID=Ubuntu
Description=Ubuntu 12.04.2 LTS
Release=12.04
Codename=precise
kernel name=Linux
node name=detlefvm05
kernel release=3.2.0-40-generic-pae
kernel version=#64-Ubuntu SMP Mon Mar 25 21:44:41 UTC 2013
machine=i686
processor=athlon
hardware platform=i386
operating system=GNU/Linux
```

`getFileInfoMap(` <file name> `)` [W]

`getProductMap` // since 4.11.2.4 [W/L/M]
keys are: id, name, description, advice, productversion, packageversion, priority, installationstate, lastactionrequest, lastactionresult, installedversion, installedpackage, installedmodificationtime,actionrequest

`getRegistryVarMap32(`<regkey>`)` //since 4.11.3 [W]

`getRegistryVarMap64(`<regkey>`)` //since 4.11.3 [W]

`getRegistryVarMapSysnative(`<regkey>`)` //since 4.11.3 [W]

`getHWBiosInfoMap` //since 4.11.4 [W/L/M]

`editmap(`< strlist>`) : stringlist` //since 4.12.1.2 [W/L/M]

3.4.2.3 Other

`createStringList (`<string0>, <string1> ,... `)` [W/L/M]

```
set $list1$ = createStringList ('a','b')
```

`reverse (`<list>`)` [W/L/M]

`getSectionNames(`<ini-file>`)` [W/L/M]

`retrieveSection (`<section name>`)` [W/L/M]

`getSubList (`<start index> : <end index>, <list>`)` [W/L/M]

`getSubListByMatch (`<search string>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getSubListByMatch (`<search list>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getSubListByContaining (` <search string>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getSubListByContaining (`<search list>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getSubListByKey (`<search string>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getSubListByKey (`<search list>, <target list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`getKeyList (`<list>`)` :stringlist //since 4.12.0.14 [W/L/M]

`addtolist(`<list>,<string>`)` //since 4.10.8 [W/L/M]

addListToList(<dest list>,<src list>) //since 4.10.8 [W/L/M]

reencodestrlist(<list>, <from>, <to>) //since 4.11.4.2 [W/L/M]

removeFromListByContaining(<search string>, <target list>) : stringlist //since 4.11.5.1 [W/L/M]

removeFromListByContaining(<search list>, <target list>) : stringlist //since 4.11.5.1 [W/L/M]

removeFromListByMatch(<searchstring>,<target list>) : stringlist //since 4.11.6 [W/L/M]

3.4.3 Boolean operators and functions

<string1> = <string2> [W/L/M]

<bool1> AND <bool2> [W/L/M]

<bool1> OR <bool2> [W/L/M]

NOT(<bool3>) [W/L/M]

FileExists (<file name>) [W/L/M]

FileExists32 (<file name>) [W]

FileExists64 (<file name>) [W]

FileExistsSysNative (<file name>) [W]

FileOrFolderExists (<file or folder path> [,<access str>]) : boolean [W/L/M] *sysnative* is the default for <access str>. Otherwise, it can be *32bit*, *64bit* or *sysnative*

LineExistsIn (<string>, <file name>) [W/L/M]

LineBeginning_ExistsIn (<string>, <file name>) [W/L/M]

LineContaining_ExistsIn(<string>, <file name>) //since 4.11.4.10: true: if a in <file name> contains <string> [W/L/M]

XMLAddNamespace(<XMLfilename>, <XMLelementname>, <XMLnamespace>) [W]

XMLRemoveNamespace(<XMLfilename>, <XMLelementname>, <XMLnamespace>) [W]

HasMinimumSpace (<drive letter>, <capacity>) [W]

Example:

```
if not (HasMinimumSpace ("%SYSTEMDRIVE%", "500 MB"))
   LogError "Required free space of 500 MB not available on %SYSTEMDRIVE%"
   isFatalError
endif
```

opsiLicenseManagementEnabled [W/L/M]

runningAsAdmin //since 4.11.1.1 [W/L/M]

isLoginScript //since 4.11.2.1 [W]

contains(<str>, <substr>) : bool //since 4.11.3: true if <substr> in <str> [W/L/M]

isNumber(<str>) //since 4.11.3: true if <str> represents an integer [W/L/M]

runningOnUefi //since 4.11.4.3: true: if the running OS was booted in UEFI mode [W]

runningInPE //since 4.12.0.13: true: if the running OS is a Windows PE [W/L/M]

isDriveReady(<drive letter>) //since 4.11.4.4: true: if the drive can be accessed [W]

runningWithGui : bool` //since 4.12.3.6: true: if the running OS has a GUI (at Win+Mac always true)[M/L/W]

saveTextFile(<list>, < filename>) //since 4.11.4.4: true: if list is succesfully written to file [W/L/M]

`saveTextFileWithEncoding(<list>, < filename>, <encoding>) : bool` //since 4.11.6.4: true: if list is succesfully written to file [W/L/M]

`saveUnicodeTextFile(<list>, < filename>, <encoding>) : bool` //since 4.12.4.13: true: if list is succesfully written to unicode file [W/L/M]

`CompareDotSeparatedNumbers(<str1>,<relation str>,<str2>)` //since 4.11.5.2: [W/L/M]

`CompareDotSeparatedStrings(<str1>,<relation str>,<str2>)` //since 4.11.5.2: [W/L/M]

`RegKeyExists(<regkey>[,<access str>]) : bool` //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

`RegVarExists(<regkey>, <var str> [,<access str>]) : bool` //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

`isPingReachable(<host>) : boolean` //since 4.12.3.6 [W/L/M]

`isValidFQDN(<domain name>) : boolean` //since 4.12.4.4 [W/L/M]

3.4.4 Misc functions

`Killtask` <process name> [W/L/M]

`requiredOpsiscriptVersion` <relation operator> <version> //since 4.12.3.6 [W/L/M]

```
requiredOpsiscriptVersion >= "4.12.3.6"
```

`requiredWinstVersion` <relation operator> <version> [W/L/M]
outdatet - use `requiredOpsiscriptVersion`

`UpdateEnvironment` //since 4.11.5 [W]:
Subsequent calls of `winbatch` with the parameter `/RunElevated` will see the changed Environment (NT6 only).

3.4.5 Flow control

if - else - endif [W/L/M]

Syntax:

`if` <condition>
;statement(s)
[`else`
;statement(s)]
`endif`

Example:

```
Set $NTVer$ = GetMsVersionInfo
if ( $NTVer$ >= "6" )
     sub_install_win7
else
  if ( $NTVer$ = "5.1" )
    sub_install_winXP
  else
    stop "not a supported OS-Version"
  endif
endif
```

for - to - do Statement //since 4.11.5 [W/L/M]

`for` %<temporary string variable>% `=` <start string> `to` <end string> `do` <one statement>

Example:

```
for %s% = "1" to "5" do sub_iteration_test
```

Switch / Case Statement //since 4.11.5 [W/L/M]

Syntax:

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

Example:

```
set $ConstTest$ = "5"
Switch $ConstTest$
        Case "1"
                set $CompValue$ = "1"
        EndCase
        Case "2"
                set $CompValue$ = "2"
        EndCase
        DefaultCase
                set $CompValue$ = "notexisting"
        EndCase
EndSwitch
```

`isFatalError` [W/L/M]

`isFatalError` <string> //since 4.11.3.2 [W/L/M]

`isSuccess` //since 4.11.3.7 [W/L/M]

`isSuspended` //since 4.11.4.1 [W/L/M]

`noUpdateScript` //since 4.11.3.7 [W/L/M]

`ExitWindows /Reboot` [W/L/M]

`ExitWindows /ImmediateReboot` [W/L/M]

`ExitWindows /ImmediateLogout` [W]

`ExitWindows /ShutdownWanted` [W]

`ExitWindows /RebootWanted` (deprecated, acts like /Reboot) [W]

`sleepSeconds` <Integer> or <string> : noresult [W/L/M]

`ChangeDirectory` <directory> //since 4.11.2.6 [W/L/M]

Chapter 4

Secondary Sections

4.1 Winbatch [W/L/M]

Function: execute programs via operating system API

`[WinBatch`<identifier>`]`

Modifier:

`/LetThemGo`

`/WaitForProcessEnding` "<program.exe>"

`/TimeOutSeconds` <seconds>

`/WaitForWindowAppearing` <window title> (*does not work with 64 Bit programs*) [W]

`/WaitForWindowVanish` <window title> (*does not work with 64 Bit programs*) [W]

`/RunElevated` // since 4.11.3: only at >= NT6 ; no network access [W]

`/RunAsLoggedOnUser` // since 4.11.3.5 ; works only inside *userLoginScripts* [W]

`/32Bit` //since 4.11.3.5 [W]

`/64Bit` //since 4.11.3.5 [W]

`/SysNative` //since 4.11.3.5 [W]

4.2 DosBatch and DosInAnIcon (ShellBatch and ShellInAnIcon) [W/L/M]

Function: Execute section via cmd.exe [W] or bash [L/M]

`[DosBatch`<identifier>`]` <optional parameters> <winst <modifier>>

`[DosInAnIcon`<identifier>`]` <optional parameters> <winst <modifier>>

`[ShellBatch`<identifier>`]` <optional parameters> <winst <modifier>>

`[ShellInAnIcon`<identifier>`]` <optional parameters> <winst <modifier>>

Modifier: //since 4.11.1.1

`/32Bit` [W]

`/64Bit` [W]

`/SysNative` [W]

`/showoutput` [W/L/M] // since 4.11.4.7

`/WaitForProcessEnding` "<program.exe>" // since 4.12.4 [W/L/M]

`/TimeOutSeconds` <seconds> // since 4.12.4 [W/L/M]

`/RunElevated` // since 4.12.4: only at >= NT6 ; no network access [W]

The modifiers has to be seperated by *winst* from the parameters.

```
DosInAnIcon_do_64bit_stuff winst /64Bit
```

Commands: see manual

4.3 ExecWith [W/L/M]

Function: Execute section via any interpreter

`[ExecWith`<identifier>`]` <path to interpreter>

Modifier:

`/LetThemGo`

`/EscapeStrings`

`/32Bit` //since 4.11.3.5 [W]

`/64Bit` //since 4.11.3.5 [W]

`/SysNative` //since 4.11.3.5 [W]

The modifiers has to be seperated by *winst* from the parameters. The following example call the 64Bit version of the powershell.exe.

```
ExecWith_do_64bit_stuff "%System%\WindowsPowerShell\v1.0\powershell.exe" winst /64Bit
```

Commands: see manual

4.4 Files [W/L/M]

Function: File Operations

`[Files`<identifier>`]`

Modifier [W]:

`/AllNTUserProfiles`

`/AllNTUserSendTo` [W]

`/32Bit` //since 4.10.8 [W]

`/64Bit` //since 4.10.8 [W]

`/SysNative` //since 4.10.8 [W]

Commands:

`checkTargetPath =` <destination directory> [W/L/M]

`copy` [Options] <source file(s)> <destination directory> [W/L/M]

some options:

`-s` recursive [W/L/M]

`-V` version control against targetdir [W]

`-v` version control against targetdir, %systemroot% and %system% **(do not use it)** [W]

`-c` continue without reboot even if it is needed [W]

`-d` date check [W]

`-u` update [W]

`-x` extract [W]

`-w` weak (do not overwrite protected files) [W]

`-n` no overwrite [W]

`-r` copy read only attribute [W]

`-h` follow symlinks [L] //since 4.11.6.14

`delete` [Options] <path[/mask]] // [W/L/M]

some options: `-s` recursive `-f` force

Example (**do not forget the trailing Backslash**):
```
delete -sf c:\delete_this_dir\
```

`del` [Options] <path[/mask]] //since 4.11.2.1 [W/L/M]

Works like `delete` but on
```
del -s -f c:\not-exists
```
if `c:\not-exists` not exists it do not search complete `c:\` for `not-exits`

Example (**you may forget the trailing Backslash**):
```
del -sf c:\delete_this_dir
```

`chmod` <mask> <path> //since 4.11.4.1 [L]

`hardlink` <existing file> <new file> // since 4.11.5 [W/L/M]

`symlink` <existing file> <new file> // since 4.11.5 [W/L/M]
At Windows `symlink` is only available at NT6 and up.

`rename` <old filename> <new filename> // since 4.11.5 [W/L/M]

`move` <old filename> <new filename> // since 4.11.5 [W/L/M]

`zipfile` <source dir> <zip file> // since 4.12.1 [W/L/M]

`unzipfile` <zip file> <target dir> // since 4.12.1 [W/L/M]

4.5 Registry [W]

Function: edit Registry

Standard method call:
`[Registry`<identifier>`]`
works with the specified section.

Alternative method call:
`Registry loadUnicodeTextFile(`<.reg file>`) /regedit`
import the specified <.reg file>.

Alternative method call (deprecated):
`Registry loadUnicodeTextFile(`<.addreg file>`) /addreg`
import the specified <.addreg file>.

Modifier:

`/AllNTUserDats`

/32Bit //since 4.10.8

/64Bit //since 4.10.8

/SysNative //since 4.10.8

Commands:

OpenKey <Key>

```
openkey [HKLM\Software\opsi.org]
```

Set <varname> = <registry type>:<value>

Add <varname> = <registry type>:<value>

Examples for registry types:

```
set "var1" = "my string"
set "var2" = REG_SZ:"my string"
set "var3" = REG_EXPAND_SZ:"%ProgramFiles%"
set "var4" = REG_DWORD:123      (Decimal)
set "var5" = REG_DWORD:0x7b     (Hexadecimal)
set "var6" = REG_BINARY:00 01 02 0F 10
set "var7" = REG_MULTI_SZ:"A|BC|de"
```

Supp <varname> <list char> <supplement>

```
supp "Path" ; "C:\utils; %JAVABIN%"
```

GetMultiSZFromFile <varname> <file name>

SaveValueToFile <varname> <file name>

DeleteVar <varname>

DeleteKey <registry key> (does since 4.11.2.1 also work with /AllNTUserDats)

4.6 Patches [W/L/M]

Function: edit Ini-files

[Patches<identifier>] <file name>

Modifier:

/AllNTUserProfiles //since 4.11.3 [W]

Commands:

add [<section name>] <variable1> = <value1>

set [<section name>]<variable1> = <value1>

addnew [<section name>]<variable1> = <value1>

change [<section name>]<variable1> = <value1>

del [<section name>] <variable1> = <value1>

del [<section name>] <variable1>

delsec [<section name>]

replace <variable1>=<value1> <variable2>=<value2>

4.7 PatchTextFile [W/L/M]

Function: edit text files

`[PatchTextFile<identifier>]` <file name>

Modifier:

`/AllNTUserProfiles` //since 4.11.3.4 [W]

Commands:

`Set_Mozilla_Pref ("<preference type>", "<preference key>", "<preference value>")`
preference type takes any value.
Some examples for preference types: `pref`, `user_pref`, `lock_pref` or `lockPref`.

`AddStringListElement_To_Mozilla_Pref ("<preference type>", "<preference key>", "<add value>")`

`Set_Netscape_User_Pref ("<key>", "<value>")` (*deprecated*)

`AddstringListElement_To_Netscape_User_Pref` (*deprecated*)

`FindLine` <search string>

`FindLine_StartingWith` <search string>

`FindLine_Containing` <search string>

`GoToTop`

`AdvanceLine` [<number of lines>]

`GoToBottom`

`DeleteTheLine`

`AddLine_` <line> or `Add_Line_` <line>

`InsertLine` <line> or `Insert_Line_` <line>

`AppendLine` <line> or `Append_Line` <line>

`Append_File` <file name>

`Subtract_File` <file name>

`SaveToFile` <file name>

`Sorted`

`setKeyValueSeparator` <separator char> //since 4.11.4.4 [W/L/M]

`setValueByKey` <keystr> <valuestr> //since 4.11.4.4 [W/L/M]

4.8 LinkFolder [W/L/M]

Function: Startmenue + Desktop Icons

`[LinkFolder<identifier>]`

Commands:

`set_basefolder` <system folder>

`set_subfolder` <folder path> (at Linux set always "")

```
set_link
  name:             <link name>
  target:           <path and name of the program>
  parameters:       [command line arguments]
  working_dir:      [working directory]
  icon_file:        [path and name of icon file, default=target]
  icon_index:       [number of icon in icon file, default=0] [W]
  shortcut:         [keyboard shortcut for calling the target] [W]
  link_categories: [list of categories] [L]
end_link
```

`delete_element` <link name>

`delete_subfolder` <folder path> [W]

The predefined virtual system folders which can be used are at Windows:
`desktop, sendto, startmenu, startup, programs, desktopdirectory,`
`common_startmenu, common_programs, common_startup, common_desktopdirectory`
and at Linux:
`common_programs,common_startup,desktop, startup`

Predefined `link_categories` for Linux:
`AudioVideo, Audio, Video, Development, Education, Game, Graphics, Network, Office, Settings, System, Utility`

Examples

```
set_basefolder common_desktopdirectory
set_subfolder ""
set_link
  name: opsi-winst
  target: "%ProgramFiles32Dir%\opsi.org\opsi-client-agent\opsi-winst\winst32.exe"
end_link
```

```
[LinkFolder_configed_lin]
set_basefolder common_programs
set_subfolder ""

set_link
  name: opsi-configed-Local
  target: java
  parameters: $parameter$
  icon_file: "$InstallDir$/opsi.png"
  link_categories: System;Utility;
end_link
```

The predefined virtual system folders:
`desktop, sendto, startmenu, startup, programs, desktopdirectory`
are pointing to the folders of the user that the script is running. If you use it in a userLoginScript with the opsi *User Profile Management* extension these virtual folders point to the folder of the user that just had logged in.

shortcut defaults to empty. // since 4.11.6.7
`shortcut` may be a combination of [*shift,alt,ctrl*] (not case sensitiv) divided by ' , '-,+ an a *Key* or a *Virtual Key Code*.
The *Key* is a letter (*A* - *Z*) or a numeral (*0* - *9*). All other Keys must be given by there *Virtual Key Code* identifier. To get these identifier (as well as the allowed combinations) just use the following helper program:
http://download.uib.de/opsi4.0/helper/showkeys.exe

4.9 OpsiServiceCall [W/L/M]

Function: opsi-Service access

`[OpsiServiceCall<identifier>]`

Commands: see manual

4.10 PatchHosts [W/L/M]

Function: hosts-files bearbeiten

`[PatchHosts<identifier>]`

Commands:

`setaddr` <hostname> <IPaddress>

`setname` <IPaddress> <hostname>

`setalias` <hostname> <alias>

`setalias` <IPadresse> <alias>

`delalias` <hostname> <alias>

`delalias` <IPaddress> <alias>

`delhost` <hostname>

`delhost` <ipadresse>

`setComment` <ident> <comment>

4.11 XML2 Sections (Experimental) [W/L/M]

Function: edit XML files
since 4.12.1.0

`[XML2<identifier>]`

Commands:

- `strictMode =` (true/false) ; Default: false
- `openNode` <xml2 path>
  Open the given Path as actual node. If the path does not exist it will be created
- `SetAttribute` <attr name> <attr value>
  At the actual node set <attr value> as value of <attr name>. If <attr name> not exists, it will be created.
- `AddAttribute` <attr name> <attr value>
  If at the actual node the attribute <attr name> not exists, it will be created with <attr value> as value. If <attr name> still exists, nothing will be changed.
- `DeleteAttribute` <attr name>
  If at the actual node the attribute <attr name> exists, it will be deleted.
- `addNewNode` <node name>
  Create at the actual node a new sub node <node name> and make this new node to the actual node.
- `setNodeText` <string>

- `DeleteNode` <xml2 path>
- `gotoParentNode`
  Make the parent node to the actual node.

Some notes to the command parameters:

- <xml2 path> `strictMode =false`:
  A line of xml node names with optional one attribute seprated by `` // ' + Example: `node_level-1_number-1 // node_level-2_B color="green"`
- <xml2 path> `strictMode =true`:
  A line of xml node names with all existing attributes seprated by `` // ' + Example: `node_level-1_number-1 // node_level-2_B color="green" count="65"`

see also : Section 5.24

4.12 XMLPatch [W]

Function: edit XML files
Depricated: please use xml2 sections: Section 4.11
and xml2 functions: Section 5.24

`[XMLPatch`<identifier>`]`

Commands: see manual

4.13 ExecPython [W/L/M]

Function: Execute section via python interpreter

`[ExecPython`<identifier>`]`

Commands: see manual

4.14 LdapSearch [W]

Function: read from LDAP

`[LdapSearch`<identifier>`]`

Commands: see manual

Chapter 5

By Topic

5.1 Compare related functions [W/L/M]

`CompareDotSeparatedStrings(`<string1>, <string2>`)` : `string` [W/L/M]

`CompareDotSeparatedStrings(`<str1>,<relation str>,<str2>`)` : `bool` //since 4.11.5.2: [W/L/M]

`CompareDotSeparatedNumbers(`<string1>, <string2>`)` : `string` [W/L/M]

`CompareDotSeparatedNumbers(`<str1>,<relation str>,<str2>`)` : `bool` //since 4.11.5.2: [W/L/M]

`boolToString(`<boolean expression>`)` : bool string (true/false) // since 4.12.0.0 [W/L/M]

`stringToBool(`<string expression: true/false>`)` : boolean // since 4.12.0.0 [W/L/M]

5.2 Crypt / Hash related functions [W/L/M]

`DecStrToHexStr (` <decstring>, <hexlength>`)` : `string` [W/L/M]

`HexStrToDecStr (`<hexstring>`)` : `string` [W/L/M]

`base64EncodeStr(`<string>`)` : `string` [W/L/M]

`base64DecodeStr(`<string>`)` : `string` [W/L/M]

`RandomStr` : `string` [W/L/M]

`RandomIntStr(`<number str>`)` : `string` [W/L/M]

`encryptStringBlow(`<keystring>,<datastring>`)` : `string` [W/L/M]

`decryptStringBlow(`<keystring>,<datastring>`)` : `string` [W/L/M]

`md5sumFromFile(`<path to file>`)` : `string` [W/L/M]

5.3 Defined Functions and Libraries [W/L/M]

since 4.12.0.0

Definition

```
DefFunc <func name>([calltype parameter ptype][,[calltype parameter ptype]]) : ftype
<function body>
endfunc
```

Where:

- `DefFunc` is the keyword used to start defining a local function..
- *<func name>* is the freely choosen name of the function.
- *calltype* is the call type of the parameter [`val` | `ref`]. `val`=*Call by Value*, `ref`=*Call by Reference*. Default: `val`
- *parameter* is the free selected name of the call parameter which is available as a local variable within the function under the aforementioned name.
- *ptype* is the type of data of the parameter wether `string` or `stringlist`.
- *ftype* is the type of data of the function wether `string` ,`stringlist` or `void`. `void` declares that no result is returned.
- *<function body>*: is the body of the function which opsi-script syntax must suffice.
- `endfunc` is the keyword used to end defining a local function..

`importLib` <string expr> ; import library // since 4.12.0.0
<string expr> : <file name>[.<file extension>][::<function name>]
If no *.<file extension>* is given `.opsiscript` is used as default.
If no *::<function name>* is given, all function from the given file will be imported.

<file name> is:

- A complete path to an existing file. [W/L/M]
- A existing file in `%ScriptPath%` [W/L/M]
- A file in `%opsiScriptHelperPath%\lib` [W]
  Is equivalent to: *%ProgramFiles32Dir%\opsi.org\opsiScriptHelper\lib*
- A existing file in `%ScriptPath%/../lib` [W/L/M]
- A existing file in `%WinstDir%\lib` [W] or `/usr/share/opsi-script/lib` [L]

The tests for the location of the <file name> are done in the order above. *opsi-script* uses the first file it finds that has a matching name.

5.4 Encoding related functions [W/L/M]

`encoding=`<encoding> // (default is system encoding) since 4.11.4.2

`GetLocaleInfoMap : stringlist` [W]

`reencodestr(`<str>, <from>, <to>`) : string` //since 4.11.4.2 [W/L/M]

`reencodestrlist(`<list>, <from>, <to>`) : stringlist` //since 4.11.4.2 [W/L/M]

`loadUnicodeTextFile (`<file name>`) : stringlist` [W]

`loadTextFileWithEncoding(` <file name> `,` <encoding>`) : stringlist` //since 4.11.5 [W/L/M]

`strLoadTextFileWithEncoding (` <filename> , <encoding>`) : string` //since 4.11.4.6 [W/L/M]

`saveTextFileWithEncoding(`<list>, < filename>`,` <encoding>`) : bool` //since 4.11.6.4: true: if list is succesfully written to file [W/L/M]

`includelog` <file name> <tail size> [<encoding>] ` : noresult`` //since 4.11.4.1 [W/L/M]

5.5 Error / Warning related functions [W/L/M]

`ExitOnError` = <boolean value> // (default=false)

`ScriptErrorMessages` = <boolean value> // (default=true)

`FatalOnSyntaxError` = <boolean value> // (default=true) since 4.11.3.2

`FatalOnRuntimeError` = <boolean value> // (default=false) since 4.11.3.2

`LogError` <string> or `LogError` = <const string>

`LogWarning` <string> or `LogWarning` = <const string>

`isFatalError` [W/L/M]

`isFatalError` <string> //since 4.11.3.2 [W/L/M]

`markErrorNumber`

`errorsOccurredSinceMark` <relation> <integer> : boolean

```
markErrorNumber
comment "log error and thereby increase the error counter"
if errorsOccurredSinceMark > 0
        comment "There was an error ..."
endif
```

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

`shellCall (`<command string>`) : string (exitcode)` //since 4.11.6.1 [W/L/M]

`processCall(`<string>`) : string (exitcode)` //since 4.11.6.1 [W/L/M]

`getLastServiceErrorClass` : string

`getLastServiceErrorMessage` : string

5.6 File related functions [W/L/M]

`strLoadTextFile (`<file name>`) : string` [W/L/M]

`strLoadTextFileWithEncoding (` <filename> , <encoding>`) : string` //since 4.11.4.6 [W/L/M]

`loadTextFile (`<file name>`) : stringlist` [W/L/M]

`loadUnicodeTextFile (`<file name>`) : stringlist` [W]

`loadTextFileWithEncoding(` <file name> `,` <encoding>`) : stringlist` //since 4.11.5 [W/L/M]

`FileExists (`<file name>`) : bool` [W/L/M]

`FileExists32 (`<file name>`) : bool` [W]

`FileExists64 (`<file name>`) : bool` [W]

`FileExistsSysNative (`<file name>`) : bool` [W]

`FileOrFolderExists` (<file or folder path> [,<access str>]) : boolean [W/L/M] *sysnative* is the default for <access str>. Otherwise, it can be *32bit*, *64bit* or *sysnative*

`listFiles (`<Path>, <Searchmask> , <SearchSubDirectories>, <[Redirection]>`) : stringlist` //since 4.12.3 [W/L/M]

```
Set $Filelist$ = listFiles("C:\windows\system32","*.Devices.*.dll","False","64bit")
```

`DirectoryExists` (<path>, [<access str>]) : `bool` [W/L/M] //since 4.12.1.0
<access str> = one of 32bit, 64bit, sysnative ; default sysnative ; ignored at non windows

`LineExistsIn` (<string>, <file name>) : `bool` [W/L/M]

`LineBeginning_ExistsIn` (<string>, <file name>) : `bool` [W/L/M]

`LineContaining_ExistsIn`(<string>, <file name>) : `bool` //since 4.11.4.10 [W/L/M]
true: if a in <file name> contains <string>

`saveTextFile`(<list>, < filename>) : `bool` //since 4.11.4.4 [W/L/M]
true: if list is succesfully written to file

`saveTextFileWithEncoding`(<list>, < filename>, <encoding>) : `bool` //since 4.11.6.4 [W/L/M]
true: if list is succesfully written to file

`getFileInfoMap`(<file name>) : `stringlist` [W]

`getFileInfoMap32`(<file name>) : `stringlist` //since 4.11.6.6 [W]

`getFileInfoMap64`(<file name>) : `stringlist` //since 4.11.6.6 [W]

`getFileInfoMapSysnative`(<file name>) : `stringlist` //since 4.11.6.6 [W]

`ExtractFilePath` (<path>) : `string` [W/L/M]

`ExtractFileExtension` (<path>) : `string` [W/L/M] //since 4.12.1

`ExtractFileName` (<path>) : `string` [W/L/M] //since 4.12.1

see also: Section 4.4

see also: Section 4.7

5.7 Ini file related functions [W/L/M]

`GetValueFromInifile` (file, section, key, default value) : `string` [W/L/M]

```
GetValueFromInifile("myfile","mysec","mykey","")
```

`getSectionNames`(<ini-file>) : `stringlist` [W/L/M]

`GetSectionFromInifile`(<ini-file>) : `stringlist` [W/L/M]

`getValue`(<key string>, <hash string list>) : `string` [W/L/M]

`getValueBySeparator`(<key string>,<separator string>,<hash string list>) : `string` //since 4.11.2.1 [W/L/M]

`getValueFromFile`(<key string>, <file name>) : `string` //since 4.11.4.4 [W/L/M]

`getValueFromFileBySeparator`(<key string>,<separator string>,<file name>) : `string` //since 4.11.4.4 [W/L/M]

see also: Section 4.6

5.8 Interaction [W/L/M]

`Pause` <string> or `Pause` = <const string>

`Stop` <string> or `stop` = <const string>

`setActionProgress <string>` : noresult //since 4.11.3 [W/L/M]

`Message` <string> or `Message` = <const string>

`ShowMessageFile` <string>

`ShowBitMap` [<file name>] [<sub title>]

`stringinput`(< message str>,< boolstr confidential>) : `string` //since 4.12.1.2 [W/L/M]

`editmap`(< strlist>) : `stringlist` //since 4.12.1.2 [W/L/M]

5.9 License Management related functions [W/L/M]

`DemandLicenseKey( poolId [, productId [,windowsSoftwareId]] ) : string`

```
set $mykey$ = DemandLicenseKey ("", "office2007")
```

`FreeLicense (`poolId [, productId [,windowsSoftwareId]]) : string`

```
set $result$ = FreeLicense("", "office2007")
```

`getLastServiceErrorClass` : string

`getLastServiceErrorMessage` : string

`opsiLicenseManagementEnabled` : bool

5.10 Linux specific functions [W/L/M]

`GetOS : string` // *Linux* or *Windows_NT* or *macos* [W/L/M]

`getLinuxDistroType : string` // *debian* or *redhat* or *suse* (see `getLinuxVersionMap`) [L]

`getLinuxVersionMap : stringlist` //since 4.11.4 [L]

`chmod` in Files sections [L/M]

`cleanupPackageSystem : void` //since 4.13.4 [L]

`installupdates : string` //since 4.13.4 [L]

`debinstall($packagelist$ : stringlist) : string` //since 4.13.4 [L]

`redinstall($packagelist$ : stringlist) : string` //since 4.13.4 [L]

`suseinstall($packagelist$ : stringlist) : string` //since 4.13.4 [L]

`ucsinstall($packagelist$ : stringlist) : string` //since 4.13.4 [L]

5.11 Logging related functions [W/L/M]

`SetLogLevel` = <number> or `SetLogLevel` = <string> // (default=6)

```
SetLogLevel = 7
SetLogLevel = "7"
```

`Message` <string> or `Message` = <const string>

`comment` <string> or `comment` = <const string>

`LogError` <string> or `LogError` = <const string>

`LogWarning` <string> or `LogWarning` = <const string>

`includelog` <file name> <tail size> //since 4.11.2.1 [W/L/M]

`includelog` <file name> <tail size> [<encoding>] //since 4.11.4.1 [W/L/M]

```
includelog "%Scriptpath%\test-files\10lines.txt" "5"
```

`SetConfidential` <secret string> //since 4.11.3.5 [W/L/M]

`asConfidential(` <secret string expression> `)` //since 4.12.0.16 [W/L/M]

`forceLogInAppenMode =` <boolean value> // (default=false); if true log will be send in append mode . //since 4.12.3.6

opsi-configs

`opsi-script.global.debug_prog` : boolean ; if false: only Warnings and Errors from program logging; default: false

`opsi-script.global.debug_lib` : boolean ; if false: only Warnings and Errors from library logging; default: false

`opsi-script.global.default_loglevel` : intstr ; set the default log level; default: *6*

`opsi-script.global.force_min_loglevel` : intstr ; set the minimal loglevel; default: *0*

`opsi-script.global.ScriptErrorMessages` : boolean ; overwrites the opsi-script internal default; default: false

`opsi-script.global.AutoActivityDisplay` : boolean ; overwrites the opsi-script internal default; default: true

5.12 Network related functions [W/L/M]

`GetHostsName (`<hostaddress>`) : string` [W/L/M]

`GetHostsAddr (`<hostname>`) : string` [W/L/M]

`GetMyIpByTarget(`<target ip addr>`) : string` //since 4.11.3.2 /4.11.6 [W/L/M]

`GetIpByName(`<ip addr / ip name>`) : string` //since 4.11.3.2 [W/L/M]

`isValidIP4 (<ip4adr>) : boolean` //since 4.12.1

`isValidIP4Network (<ip4adr>, <netmask>) : boolean` //since 4.12.1

`isValidIP4Host (<ip4adr>, <netmask>) : boolean` //since 4.12.1

`getIP4NetworkByAdrAndMask (<ip4adr>, <netmask>) : string` //since 4.12.1

`getDefaultNetmaskByIP4adr (<ip4adr>) : string` //since 4.12.1

`parseUrl(<url string>) : stringlist` //since 4.12.1

`createUrl(<urlcomponents list>) : string` //since 4.12.1

`isPingReachable(<host>) : boolean` //since 4.12.3.6

`isValidFQDN(<domain name>) : boolean` //since 4.12.4.4 [W/L/M]

5.13 Number related functions [W/L/M]

`isNumber(`<str>`) : bool` //since 4.11.3: true if <str> represents an integer [W/L/M]

`CompareDotSeparatedNumbers(`<str1>,<relation str>,<str2>`) : bool` //since 4.11.5.2: [W/L/M]

`CompareDotSeparatedNumbers(`<string1>, <string2>`) : string` [W/L/M]

`calculate(`<arithmetic string expression>`) : string (number)` // since 4.11.3.5 [W/L/M]
knows: `+-*/()`

`DecStrToHexStr (` <decstring>, <hexlength>`) : string` [W/L/M]

`HexStrToDecStr (`<hexstring>`) : string` [W/L/M]

`RandomIntStr(`<number str>`) : string` [W/L/M]

5.14 Operating System related functions [W/L/M]

`GetOS : string` // *Linux* or *Windows_NT* or *macos* [W/L/M]

`GetMsVersionInfo : string` //Windows Version Information [W]

`GetMSVersionMap : stringlist` [W]

`getLinuxDistroType : string` // *debian* or *redhat* or *suse* (see `getLinuxVersionMap`) [L]

`getLinuxVersionMap : stringlist` //since 4.11.4 [L]

`getMacosVersionInfo : string` //macOS Version Information //since 4.12.1.0 [M]

`getMacosVersionMap : stringlist` //macOS Version map //since 4.12.1.0 [M]

`GetSystemType : string` //OS Architecture ("64 Bit System" or "x86 System") [W/L/M]

`getListFromWMI(`<wmi namespace str>,<wmi class str>,<property list>,<condition str>`) : stringlist` //since 4.12.1.0 [W]

`EnvVar (`<environment variable>`) : string` [W/L/M]

`getProfilesDirList : stringlist` //since 4.11.3.2 [W/L/M]

`listFiles (`<Path>, <Searchmask> , <SearchSubDirectories>, <[Redirection]>`) : stringlist` //since 4.12.3 [W/L/M]

```
Set $Filelist$ = listFiles("C:\windows\system32","*.Devices.*.dll","False","64bit")
```

`which(`<command in path>`) : string` (command with path) //since 4.12.3.6 [W/L/M]

`runningAsAdmin : bool` //since 4.11.1.1 [W/L/M]

`runningOnUefi : bool`` //since 4.11.4.3: true: if the running OS was booted in UEFI mode [W/L/M]

`runningInPE : bool`` //since 4.12.0.13: true: if the running OS is a Windows PE [W]

`isDriveReady(`<drive letter>`) : bool`` //since 4.11.4.4: true: if the drive can be accessed [W]

`runningWithGui : bool`` //since 4.12.3.6: true: if the running OS with GUI (at Win+Mac always true)[M/L/W]

5.15 opsiservicecall and json Related functions [W/L/M]

`jsonIsValid(`<jsonstr>`) : boolean` //since 4.11.6: [W/L/M]

`jsonIsArray(`<jsonstr>`) : boolean` //since 4.11.6: [W/L/M]

`jsonIsObject(`<jsonstr>`) : boolean` //since 4.11.6: [W/L/M]

`jsonAsObjectHasKey(`<jsonstr>,<keystr>`) : boolean` //since 4.11.6: [W/L/M]

`jsonAsArrayCountElements(`<jsonstr>`) : intstr` //since 4.11.6: [W/L/M]

`jsonAsObjectCountElements(`<jsonstr>`) : intstr` //since 4.11.6: [W/L/M]

`jsonAsArrayGetElementByIndex(`<jsonstr>, <indexstr>`) : jsonstring` //since 4.11.6: [W/L/M]

`jsonAsObjectGetValueByKey(`<jsonstr>, <keystr>`) : valuestring` //since 4.11.6: [W/L/M]

`jsonAsObjectSetValueByKey(`<jsonstr>, <keystr>,<valuestring>`) : jsonstring` //since 4.11.6: [W/L/M]

`jsonAsObjectSetStringtypeValueByKey(`<jsonstr>, <keystr>,<valuestring>`) : jsonstring` //since 4.11.6: [W/L/M]

`jsonAsObjectDeleteByKey(`<jsonstr>, <keystr>`) : jsonstring` //since 4.11.6.4: [W/L/M]

`jsonAsArrayPutObjectByIndex(`<jsonstr>, <indexstr>, <objectstr>`) : jsonstring` //since 4.11.6: [W/L/M]

`jsonAsArrayDeleteObjectByIndex(<jsonstr>, <indexstr>) : jsonstring` //since 4.11.6.4: [W/L/M]

`jsonAsArrayToStringList(<jsonstr>) : stringlist` //since 4.11.6: [W/L/M]

`jsonAsObjectGetKeyList(<jsonstr>) : stringlist` //since 4.11.6: [W/L/M]

`jsonStringListToJsonArray(<strlist>) : jsonstr` //since 4.11.6: [W/L/M]

`convert2Jsonstr(<string>)` //since 4.10.8.3

see also: OpsiServiceCall Section 4.9

5.16 opsi related functions [W/L/M]

`getProductMap : stringlist` // since 4.11.2.4 [W/L/M]
keys are: id, name, description, advice, productversion, packageversion, priority, installationstate, lastactionrequest, lastactionresult, installedversion, installedpackage, installedmodificationtime,actionrequest

`getProductPropertyList(<propname>,<default value>) : stringlist` //since 4.11.3 [W/L/M]

`GetProductProperty (<PropertyName>, <DefaultValue> ) : string` [W/L/M]

`GetConfidentialProductProperty ( <PropertyName>, <DefaultValue>) : string` //since 4.11.5.2 [W/L/M]

`setActionProgress <string>` : noresult //since 4.11.3 [W/L/M]

`retrieveSection (<section name>) : stringlist` [W/L/M]

`replaceOpsiConstants(<string list>) : stringlist` //since 4.12.3.6 [W/L/M]

`replaceOpsiConstants(<string>) : string` //since 4.12.3.6 [W/L/M]

5.17 Process and Script Related functions

`Killtask` <process name> ` : noresult` [W/L/M]

`ChangeDirectory` <directory> ` : noresult` //since 4.11.2.6 [W/L/M]

`GetProcessList : stringlist` //since 4.11.1.2; gives list of exename;pid;dom/user [W/L/M]

`processIsRunning(<process name>) : boolean` //since 4.11.6.1 [W/L/M]

`shellCall (<command string>) : stringlist (output)` //since 4.11.4.2 [W/L/M]

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

`shellCall (<command string>) : noresult` //since 4.11.6.1 [W/L/M]

`shellCall (<command string>) : string (exitcode)` //since 4.11.6.1 [W/L/M]

`powershellcall (<commandstr> [,<access str>=`*sysnative* `[,<policy bool str>=`*true*`]]) : stringlist (output)` //since 4.12.0.16 [W]

`powershellcall (<commandstr> [,<access str>=`*sysnative* `[,<policy bool str>=`*true*`]]) : noresult` //since 4.12.0.16 [W]

`powershellcall (<commandstr> [,<access str>=`*sysnative* `[,<policy bool str>=`*true*`]]) : string (exitcode)` //since 4.12.0.16 [W]

`getOutStreamFromSection (<dos section name>) : stringlist (output)` [W/L/M]

```
set $list$= getOutStreamFromSection ('DosInAnIcon_try')
```

`processCall(<string>) : string (exitcode)` //since 4.11.6.1 [W/L/M]

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

`includelog` <file name> <tail size> : `noresult` //since 4.11.2.1 [W/L/M]

`includelog` <file name> <tail size> [<encoding>] : `noresult` //since 4.11.4.1 [W/L/M]

`waitForPackageLock(<seconds timeout string>,<bool should we kill>) : bool` //since 4.11.6.1 [L]

`which(<command in path>) : string` (command with path) //since 4.12.3.6 [W/L/M]

see also: ExecWith sections Section 4.3

see also: ShellBatch sections Section 4.2

see also: Winbatch sections Section 4.1

5.18 Regular expression related functions [W/L/M]:

`isRegexMatch(<string>, <pattern>) : boolean` //since 4.12.1

`getSubListByContainingRegex(<pattern>, <target list>) : stringlist` //since 4.12.1

`getSubListByContainingRegex(<pattern list>, <target list>) : stringlist` //since 4.12.1

`getRegexMatchList(<pattern>, <target list>) : stringlist` //since 4.12.1

`getRegexMatchList(<pattern list>, <target list>) : stringlist` //since 4.12.1

`removeFromListByContainingRegex(<pattern>, <target list>) : stringlist` //since 4.12.1

`removeFromListByContainingRegex(<pattern list>, <target list>) : stringlist` //since 4.12.1

`stringReplaceRegex(<string>, <pattern>, <replacement string>) : string` //since 4.12.1

`stringReplaceRegexInList(<target list>, <pattern>, <replacement string>) : stringlist` //since 4.12.1

5.19 Registry related functions [W]

`getRegistryValue(<keystr>, <varstr> [, <access str>])` : string //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

`GetRegistrystringvalue("[key] var") : string` [W]

`GetRegistryStringValue32 ("[key] var") : string` //since 4.10.8 [W]

`GetRegistryStringValue64 ("[key] var") : string` //since 4.10.8 [W]

`GetRegistryStringValueSysNative ("[key] var") : string` //since 4.10.8 [W]

`getRegistryKeyList32(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryKeyList64(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryKeyListSysnative(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarList32(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarList64(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarListSysnative(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarMap32(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarMap64(<regkey>) : stringlist` //since 4.11.3 [W]

`getRegistryVarMapSysnative(<regkey>) : stringlist` //since 4.11.3 [W]

`RegKeyExists(<regkey>[,<access str>]) : bool` //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

`RegVarExists(<regkey>, <var str> [,<access str>]) : bool` //since 4.12.0.16 [W]
<access str> = one of 32bit, 64bit, sysnative ; default sysnative

see also: Section 4.5

5.20 String handling functions [W/L/M]

`splitString (<string1>, <string2>) : stringlist` [W/L/M]

```
set $list1$ = splitString ("\\server\share\dir","\")
```

`splitStringOnWhiteSpace (<string>) : stringlist` [W/L/M]

`composeString (<string list>, <Link>) : string` [W/L/M]

`takeString (<index>, <list>) : string` [W/L/M]

`setStringInListAtIndex(<newstring>,<list>,<indexstr>) : stringlist` //since 4.11.6 [W/L/M]

`takeFirstStringContaining(<list>,<search string>) : string` [W/L/M]

`getIndexFromListByContaining(<list> : stringlist,<search string> : string`)` : <number> : string` //since 4.12.0.13 [W/L/M]

`contains(<str>, <substr>) : bool` //since 4.11.3: true if <substr> in <str> [W/L/M]

`isNumber(<str>) : bool` //since 4.11.3: true if <str> represents an integer [W/L/M]

`trim(<string>) : string` [W/L/M]

`lower(<string>) : string` [W/L/M]

`upper(<string>)` [W/L/M]

`unquote(<string>,<quote-string>) : string` //since 4.11.2.1 [W/L/M]

`unquote2(<string>,<quote-string>) : string` //since 4.11.5.2 [W/L/M]

`stringReplace(<string>, <oldPattern>, <newPattern>) : string` //since 4.11.3 [W/L/M]

`strLength(<string>) : string (number)` //since 4.11.3 [W/L/M]

`strPos(<string>, <sub string>) : string (numner)` //since 4.11.3 [W/L/M]

`strPart(<string>, <start pos>, <number of chars>) : string` //since 4.11.3 [W/L/M]

`getValue(<key string>, <hash string list> ) : string` [W/L/M]

`getValueBySeparator(<key string>,<separator string>,<hash string list> ) : string` //since 4.11.2.1 [W/L/M]

`getValueFromFile(<key string>, <file name>) : string` //since 4.11.4.4 [W/L/M]

`getValueFromFileBySeparator(<key string>,<separator string>,<file name>) : string` //since 4.11.4.4 [W/L/M]

`EscapeString: <sequence of characters> : string`// [W/L/M]

`stringReplaceRegex(<string>, <pattern>, <replacement string>) : string` //since 4.12.1 [W/L/M]

`stringinput(< message str>,< boolstr confidential>) : string` //since 4.12.1.2 [W/L/M]

5.21 Stringlist handling functions [W/L/M]

getListContaining(<list>,<search string>) : stringlist [W/L/M]

getListContainingList(<list1>,<list2>) : stringlist //since 4.11.3.7 [W/L/M]

getIndexFromListByContaining(<list> : stringlist,<search string> : string`)` : <number> : string //since 4.12.0.13 [W/L/M]

count (<list>) : string (number) [W/L/M]

emptylist (<list>) : stringlist //since 4.11.3.7 [W/L/M]

for %<identifier>% in <list> do <one statement | sub section> [W/L/M]

```
for %s% in $list1$ do sub_test_string
```

createStringList (<string0>, <string1> ,...) : stringlist [W/L/M]

```
set $list1$ = createStringList ('a','b')
```

reverse (<list>) : stringlist [W/L/M]

getSubList (<start index> : <end index>, <list>) : stringlist [W/L/M]

getSubListByMatch (<search string>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getSubListByMatch (<search list>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getSubListByContaining (<search string>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getSubListByContaining (<search list>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getSubListByKey (<search string>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getSubListByKey (<search list>, <target list>) :stringlist //since 4.12.0.14 [W/L/M]

getKeyList (<list>) :stringlist //since 4.12.0.14 [W/L/M]

addtolist(<list>,<string>) : stringlist //since 4.10.8 [W/L/M]

addListToList(<dest list>,<src list>) : stringlist //since 4.10.8 [W/L/M]

reencodestrlist(<list>, <from>, <to>) : stringlist //since 4.11.4.2 [W/L/M]

removeFromListByContaining(<search string>, <target list>) : stringlist //since 4.11.5.1 [W/L/M]

removeFromListByContaining(<search list>, <target list>) : stringlist //since 4.11.5.1 [W/L/M]

removeFromListByMatch(<searchstring>,<target list>) : stringlist //since 4.11.6 [W/L/M]

takeString (<index>, <list>) : string [W/L/M]

takeFirstStringContaining(<list>,<search string>) : string [W/L/M]

setStringInListAtIndex(<newstring>,<list>,<indexstr>) : stringlist //since 4.11.6 [W/L/M]

jsonAsArrayToStringList(<jsonstr>) : stringlist //since 4.11.6: [W/L/M]

jsonStringListToJsonArray(<strlist>) : jsonstr //since 4.11.6: [W/L/M]

jsonAsObjectGetKeyList(<jsonstr>) : stringlist //since 4.11.6: [W/L/M]

splitString(<string1>, <string2>) : stringlist [W/L/M]

```
set $list1$ = splitString ("\\server\share\dir","\")
```

`splitStringOnWhiteSpace (<string>) : stringlist` [W/L/M]

`composeString(<string list>, <Link>) : string` [W/L/M]

`getValue(<key string>, <hash string list> ) : string` [W/L/M]

`getValueBySeparator(<key string>,<separator string>,<hash string list> ) : string` //since 4.11.2.1 [W/L/M]

`getSubListByContainingRegex(<pattern>, <target list>) : stringlist` //since 4.12.1

`getSubListByContainingRegex(<pattern list>, <target list>) : stringlist` //since 4.12.1

`getRegexMatchList(<pattern>, <target list>) : stringlist` //since 4.12.1

`getRegexMatchList(<pattern list>, <target list>) : stringlist` //since 4.12.1

`removeFromListByContainingRegex(<pattern>, <target list>) : stringlist` //since 4.12.1

`removeFromListByContainingRegex(<pattern list>, <target list>) : stringlist` //since 4.12.1

`stringReplaceRegexInList(<target list>, <pattern>, <replacement string>) : stringlist` //since 4.12.1

`editmap(< strlist>) : stringlist` //since 4.12.1.2 [W/L/M]

`areListsEqual(< strlist1>, <strlist2>, <flag>) : boolean`

5.22 Time / Date related functions [W/L/M]

`sleepSeconds` <Integer> or <string> : noresult [W/L/M]
breaks the program execution for <string> seconds. <string> has to represent an Integer Value

`markTime` : noresult [W/L/M]
sets a time stamp for the current system time and logs it.

`getDiffTimeSec : string` (Time in seconds since last marktime) //since 4.11.3 [W/L/M]

`timeStampAsFloatStr : string` (Floating Number - format: *days.decimal days*) //since 4.11.6 [W/L/M]

5.23 Usercontext / loginscripts related functions [W]:

`GetUserSID(<Windows Username>) : string`

`GetLoggedInUser : string` //since 4.11.1.2

`GetUsercontext : string` //since 4.11.1.2

`GetScriptMode : string` possible values *Machine*,*Login* //since 4.11.2.1

`saveVersionToProfile : noresult` - save productversion-packageversion to local profile //since 4.11.2.1

`readVersionFromProfile : string` - read productversion-packageversion from local profile //since 4.11.2.1

`scriptWasExecutedBefore : boolean` - is true if saved and running productversion-packageversion are identical //since 4.11.2.1

5.24 XML related functions (XML2) (Experimental) [W/L/M]:

`getXml2DocumentFromFile(<path to xml file>) : xml2stringlist` //since 4.12.1

`getXml2Document(<stringlist wit xml>) : xml2stringlist` //since 4.12.1

`xml2GetFirstChildNodeByName(<xml2stringlist>, <node name str>) : xml2stringlist` //since 4.12.1.

`getXml2UniqueChildnodeByName(`<xml2stringlist>, <node name str>`) : xml2stringlist` //since 4.12.1.

`getXml2AttributeValueByKey(`<xml2stringlist>, <attr name str>`) : string` //since 4.12.1.

`getXml2Text(`<xml2stringlist>`) : string` //since 4.12.1.

see also : Section 4.11