# Allgemeine Befehle

## Systeminformation anzeigen

### CMD

```
systeminfo
```

### GUI

```
msinfo32
```

(seit 2012 kein Systemverlauf)

## Touch in Powershell

```
(Get-ChildItem(".\Svcrunap.exe")).LastWriteTime = Get-Date
```

```
get-childitem | foreach { $_.LastWriteTime=Get-Date }
```

## Royal TS Prepare Server Powershell-Skript

prepare_server.ps1

```
# prepare_server.ps1
#
# Script is to be used at own risk
#
# PowerShell script that does the following steps:
#     1. Enable-PSRemoting for localhost
#     2. Enables the Firewall Rule "Windows Management Instrumentation
(WMI-In)"
#     3. Enables the Firewall Rule "Windows Management Instrumentation
(DCOM-In)"
#
# execution:
#     powershell.exe prepare_server.ps1
#
# tested on:
#     1. windows server 2012 r2
#     2. windows server 2012
#
# known issues:
#     problem: on windows server 2008 r2 the powershell commandlet
"Get-NetFirewallRule" is missing.
#     fix: use the following commandline to enable the firewall vie
```

```
netsh:
#         netsh advfirewall firewall set rule name="Windows Management
Instrumentation (WMI-In)" new enable=yes
#         netsh advfirewall firewall set rule name="Windows Management
Instrumentation (DCOM-In)" new enable=yes
#
#         but our tests showed that this is not needed anyways on this
operating system.
#
#
#    problem: File can not be loaded because running scripts is
disabled on this system.
#    fix: enabled runnning scripts via Set-ExecutionPolicy unrestricted
#
#

function Test-PSRemoting
{
    param(
        [Parameter(Mandatory = $true)]
        $computername
    )

    try
    {
        $errorActionPreference = "Stop"
        $result = Invoke-Command -ComputerName $computername { 1 }
    }
    catch
    {
        Write-Verbose $_
        return $false
    }

    if($result -ne 1)
    {
        Write-Verbose "Remoting to $computerName returned an unexpected
result."
        return $false
    }
    $true
}

function EnableFirewallRuleIfNeeded {
    param ([string]$ruleName)


    $rules = Get-NetFirewallRule -DisplayName $ruleName

    foreach($rule in $rules)
    {
```

```
        if($rule.Enabled.ToString() -eq "False")
        {
            $msg = "Rule '{0}/({1})' is disabled. Do you want to
enabled it? (y/n)" -f $rule.DisplayName, $rule.Name
            $en = Read-Host $msg
            if($en -eq "y")
            {

                Set-NetFirewallRule -Name $rule.Name -Enabled True
                $newrule = Get-NetFirewallRule -Name $rule.Name

                "Rule '{0}/({1})'.Enabled is '{2}' now" -f
$rule.DisplayName, $rule.Name, $newrule.Enabled
            }
        }
        else
        {
            $msg = "Rule '{0}/({1})' is enabled already. No changes
needed." -f $rule.DisplayName, $rule.Name
            Write-Host $msg
        }
    }

}



# PSRemoting
function EnablePSRemotingIfNeeded {
    $psremotingEnabled = Test-PsRemoting -computername localhost
    if($psremotingEnabled -eq $false)
    {
        $msg = "PSRemoting is not enabled on this machine. Do you want
to enable it? (y/n)"
        $en = Read-Host $msg
        if($en -eq "y")
        {

            $err = @()
            Enable-PSRemoting -ErrorAction SilentlyContinue -
ErrorVariable err


            if ($err -ne $null)
            {
                Write-Host "PSRemoting could not be enabled:"
                $err
            }
            else
            {
                Write-Host "PSRemoting is now enabled on this
```

```
computer."
            }
        }
        else
        {
            Write-Host "No changes were done to PSRemoting. Royal
Server needs it to be enabled to work."
        }
    }
    else
    {
        Write-Host "PSRemoting is already enabled on this computer."
    }
}


# Enable-PSRemoting
EnablePSRemotingIfNeeded

# firewall rules
EnableFirewallRuleIfNeeded "Windows Management Instrumentation (WMI-
In)"
EnableFirewallRuleIfNeeded "Windows Management Instrumentation (DCOM-
In)"
```