

PS RoomSigning

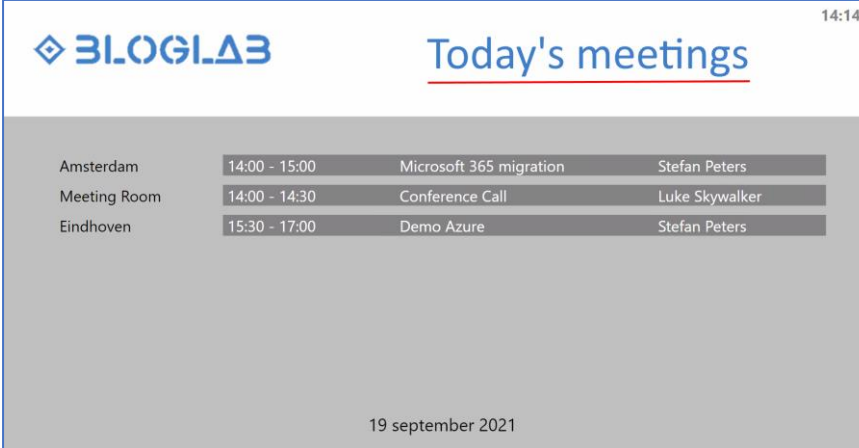
2022.02.05

Contents

Introduction	2
How does it work?	2
Revision History	3
System Requirements	3
For the PowerShell script	3
For the webserver	3
Preparation	4
Download the jquery.js file from the source	4
Prepare the HTML files	5
Prepare the background	6
Create the Service Principal Name (SPN).....	7
Install PowerShell Modules.....	7
Prepare for hosting the webserver on Azure.....	11
Prepare PSroomSigning2022.ps1 script.....	15
Installation	17
Install on Azure RM.....	17
Test Azure App Service.....	22
Azure Scheduler	23
PUBLIC Visibility	26
Install on Windows Server	28
Configure the scheduled task	29
Configure your display devices	33
Windows 10 Client configuration	33

Introduction

PSRoomSigning is a HTML and PowerShell based solution for RoomSigning, the script can run on AzureRM or Windows operating system. This new version needs a webserver compatible with jQuery. PSRoomSigning will show the meetings of your meeting rooms in your environment. If you have an information display in your company restaurant or foyer you can display this information. Only meetings of today are displayed and meetings in the past are no longer on the screen.

The screenshot shows a web application interface. At the top left is the 'BLOGLAB' logo. At the top right is the time '14:14'. In the center, the title 'Today's meetings' is underlined. Below the title is a table with meeting information. The table has four columns: Location, Time, Meeting Name, and Organizer. The data rows are: Amsterdam (14:00 - 15:00, Microsoft 365 migration, Stefan Peters), Meeting Room (14:00 - 14:30, Conference Call, Luke Skywalker), and Eindhoven (15:30 - 17:00, Demo Azure, Stefan Peters). At the bottom center, the date '19 september 2021' is displayed.

14:14			
BLOGLAB Today's meetings			
Amsterdam	14:00 - 15:00	Microsoft 365 migration	Stefan Peters
Meeting Room	14:00 - 14:30	Conference Call	Luke Skywalker
Eindhoven	15:30 - 17:00	Demo Azure	Stefan Peters
19 september 2021			

Example of RoomSigning

How does it work?

The PowerShell script is designed to run as a scheduled task, or scheduled Azure Automation Runbook. The script needs a Service Principal Name in Azure AD with 'Calendar.Read' and 'User.Read' permissions. And when your target is Azure Web App Service the SPN needs Contributor permissions on the App Service.

When the script runs it will read the configured Exchange Online Room mailboxes and will render an HTML file in a configured target location. This HTML file is the visual representation of the Exchange Rooms. The HTML location needs to be a location where your display device can access it. This can be a webserver running local or in Azure.

This new version has a new method for refreshing. The old method was not reliable in the long run. After some time, the client would display "Server error 500". Most likely the HTML was being rewritten and exactly on that moment the refresh occurred.

Revision History

2022 – Second release

- Reinvented the solution with extra jQuery script and splitting the HTML into two files.
- Add jQuery script for refreshing the page, making it more robust and stable.
- Requires webserver, doesn't work local anymore.
- Fixed the PSRoomsigning_CreateResources.ps1 script now it works.

2021 – First release

System Requirements

For the PowerShell script

- Azure RM, Automation Account with PowerShell RunBook.
- Windows 10/11 or Windows Server any supported version, tested on Windows Server 2019.

For the webserver

- Microsoft IIS on Windows Server
- Azure App Service and App Service Plan
- jQuery compatible webserver

Preparation

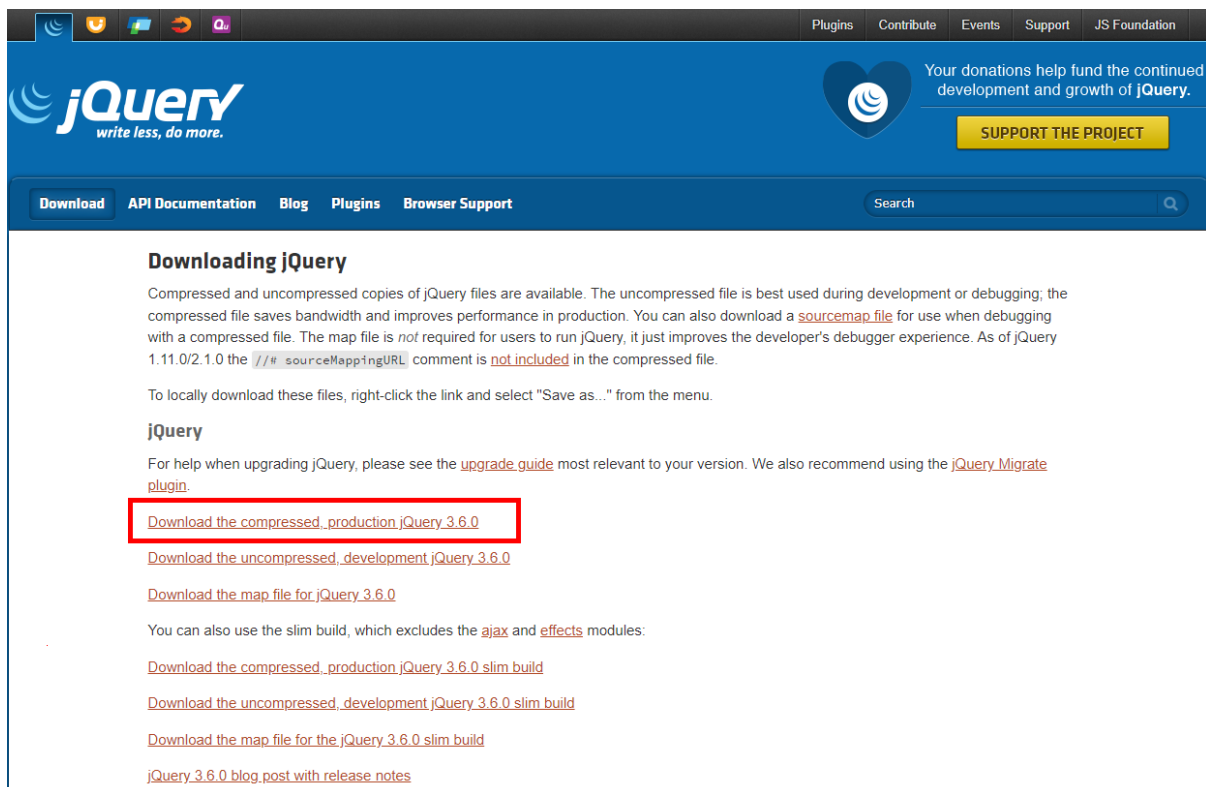
After you downloaded and extracted the PSRoomSigning2022.zip file you need to prepare the files.

Filename	Purpose
.\webserver\default.htm	Static HTML file that is opened by the client.
.\webserver\psroomsigning_bg.png	Static background picture.
.\webserver\jquery-3.6.0.js (*1)	jQuery file required for the new page refreshing technic.
.\scripts\psroomsigning2022.ps1	The PowerShell script that generates the 'roomsigning.html' file.
.\scripts\PSRoomsigning_CreateResources2022.ps1	Script to create some Azure resources.
PSRoomSigning_Manual.pdf	This manual.

*1 file not in ZIP, download it from the source

Download the jquery.js file from the source

Goto website: <https://jquery.com/download/>



The screenshot shows the jQuery website's download page. The header includes the jQuery logo and navigation links like 'Download', 'API Documentation', 'Blog', 'Plugins', and 'Browser Support'. The main content area is titled 'Downloading jQuery' and provides information about compressed and uncompressed files. A red box highlights the link 'Download the compressed, production jQuery 3.6.0'. Below this, there are links for the development version, map file, and slim build versions. The footer mentions the jQuery 3.6.0 blog post with release notes.

Download the compressed production jQuery 3.6.0 version (or later if new is available). Save the file and name it like this: "jquery-3.6.0.js" (Align filename with actual version)

Prepare the HTML files

The default.htm file can be customized to suit your needs.

Editable values:

Line Nr	Value	Purpose
5	psroomsigning_bg.png	Background picture, save you picture with this name, or change the name in this htm file.
7	jquery-3.6.0.js	jQuery file, align name with downloaded file.
16-27	// comment	Uncomment the preferred clock location, align with lines 50,51 and 52.
35, 41	roomsigning.html	HTML file created by the PowerShell script, align name.
42	75000	Reload timeout for reloading the HTML created by the PowerShell script. 75000 milliseconds = 75 seconds.
50,51,52	<h1 style> <!-- comment --->	Change the <h1 style> according to your wishes. You can customize color, font, size etc. Uncomment one of the 3 examples, aligned with lines 16-27.

*1: line numbers from original file in ZIP.

```
<!-- Read the manual for more information about configuration in this file. --->
<html>
<head>
  <style>
    body { background-image: url('psroomsigning_bg.png'); }
  </style>
  <script src="jquery-3.6.0.js"></script>
  <script>
    function startTime() {
      const today = new Date();
      let h = today.getHours();
      let m = today.getMinutes();
      let s = today.getSeconds();
      m = checkTime(m);
      s = checkTime(s);

      //
      // 6 different pre-coded clocks, right, center and left, with or without seconds. Only uncomment one!
      //
      document.getElementById('clock').innerHTML = h + ':' + m + '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'; // Clock align right, no seconds.
      // document.getElementById('clock').innerHTML = h + ':' + m + '.' + s + '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'; // Clock align right, with seconds.

      // document.getElementById('clock').innerHTML = h + ':' + m; // Clock align center, no seconds.
      // document.getElementById('clock').innerHTML = h + ':' + m + '.' + s; // Clock align center, with seconds.

      // document.getElementById('clock').innerHTML = '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;' + h + ':' + m; // Clock align left, no seconds.
      // document.getElementById('clock').innerHTML = '&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;' + h + ':' + m + '.' + s; // Clock align left, with seconds.
      //
      setTimeout(startTime, 1000); }
    function checkTime(i) {
      if (i < 10) {i = "0" + i}; // add zero in front of numbers < 10
      return i;
    }
  </script>

  <script>
    $(function(){ $("#includedContent").load("roomsigning.html", {'f' + (Math.random()*1000000)}); });
  </script>

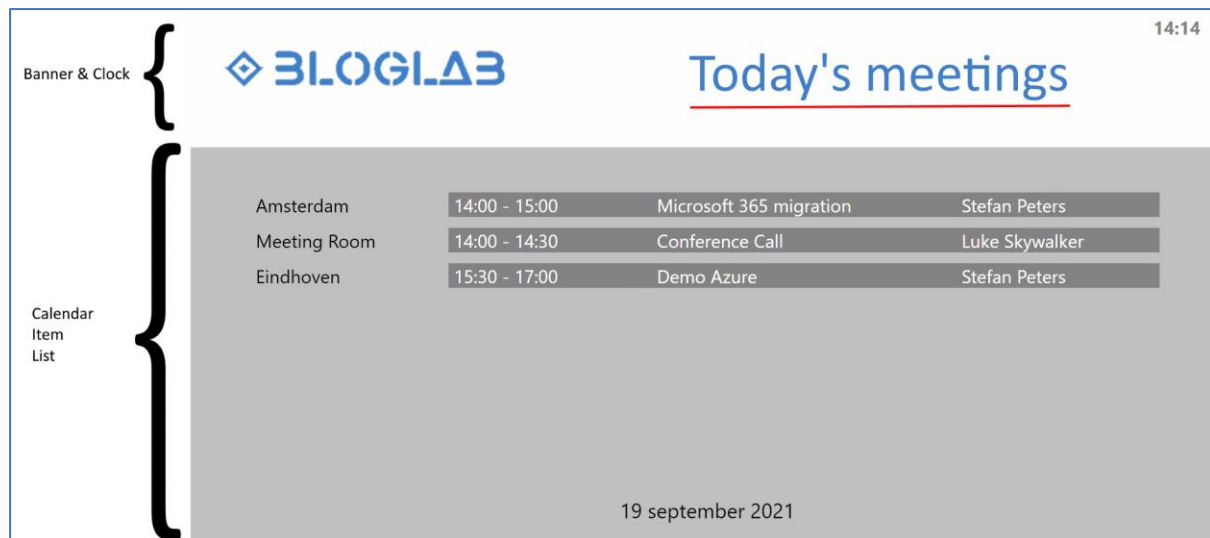
  <script>
    setInterval(function()
    {
      $(function){ $("#includedContent").load("roomsigning.html", {'f' + (Math.random()*1000000)}); });
    }, 75000);
  </script>
</head>

<body onload="startTime()">
  <!--
    Three locations for the clock, right, center and left. Uncomment only one. You can change the font, font-size and color to your liking.
  -->
  <h1 style="font-family: Segoe UI;font-size:32px;color:848284;" align="right" id="clock"></h1>
  <!-- <h1 style="font-family: Segoe UI;font-size:32px;color:848284;" align="center" id="clock"></h1> --->
  <!-- <h1 style="font-family: Segoe UI;font-size:32px;color:848284;" align="left" id="clock"></h1> ---->
  <div id="includedContent"></div>

</body>
</html>
```

Prepare the background

The ZIP download contains an example background for the RoomSigning screen. You can customize this to your own needs. Use any PNG editor to change the logo's, the colors etc. You can even have multiple backgrounds on different sites, so each instance has its own background.



The default filename of the background picture is: psroomsigning_bg.png this file must be placed on the webserver in the same folder as default.htm and jquery*.js file.

Create the Service Principal Name (SPN)

For any deployment you need a Service Principal Name (SPN) in Azure AD to authenticate the Microsoft.Graph script.

If you are installing the script on a Windows Server, you can execute these preparations on the target Windows Server.

Install PowerShell Modules

Install the PowerShell modules “Az” and “AzureAD” on a recent and supported version of Windows. If you want to publish the result on a webserver install IIS or another webserver.

PowerShell Commands to install the modules:

- Install-Module Az
- Install-Module AzureAD

Tip: If you are hosting the script on Windows, create the SPN on the target server.

Create a service account that will login to the server and can run the scheduled task on this server.

Login to the server with the service account that will run the Scheduled Tasks, the script will save the certificate in the user context. Setup folder structure as above in the example and table. Open an elevated PowerShell and browse to the c:\scripts\PSRoomSigning\scripts folder.

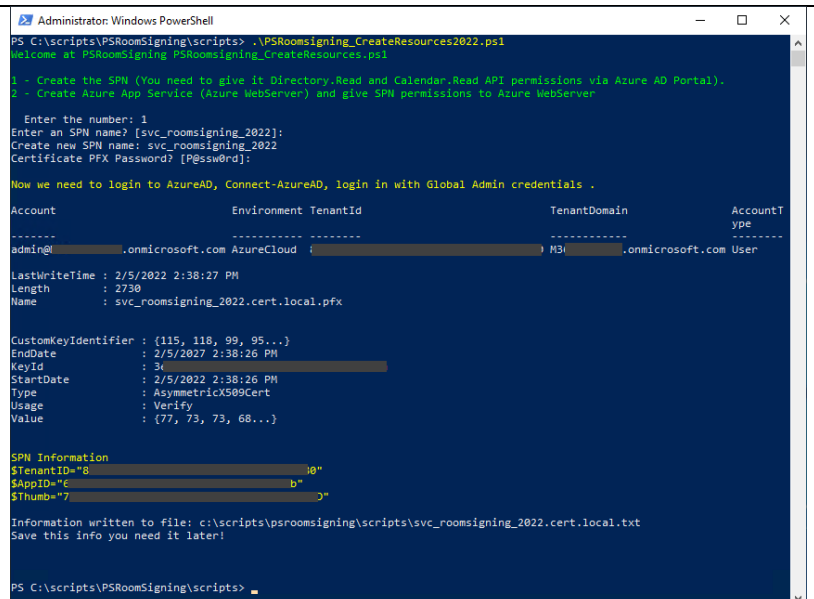
Execute the “PSRoomSigning_CreateResources022.ps1” script and follow setup 1.

First, we need to create the SPN, choose option 1.

Change the SPN name if you like and enter a secret for the PFX file.

You will be prompted to login with a Global Admin account.

Save the yellow output: SPN Information
You need to paste these 3 variables into your script.



```
PS C:\scripts\PSRoomSigning\scripts> .\PSRoomSigning_CreateResources022.ps1
Welcome at PSRoomSigning PSRoomSigning_CreateResources.ps1

1 - Create the SPN (You need to give it Directory.Read and Calendar.Read API permissions via Azure AD Portal).
2 - Create Azure App Service (Azure WebServer) and give SPN permissions to Azure WebServer

Enter the number: 1
Enter an SPN name? [svc_roomsigning_2022]:
Create new SPN name: svc_roomsigning_2022
Certificate PFX Password? [P@ssw0rd]:

Now we need to login to AzureAD, Connect-AzureAD, login in with Global Admin credentials .

Account Environment TenantId TenantDomain AccountType
-----
admin@.onmicrosoft.com AzureCloud HSI.onmicrosoft.com User

LastWriteTime : 2/5/2022 2:38:27 PM
Length : 2730
Name : svc_roomsigning_2022.cert.local.pfx

CustomKeyIdentifier : {115, 118, 99, 95...}
EndDate : 2/5/2022 2:38:26 PM
KeyId : 34
StartDate : 2/5/2022 2:38:26 PM
Type : AsymmetricX509Cert
Usage : Verify
Value : {77, 73, 73, 68...}

SPN Information
$TenantID="8"
$AppID="e"
$Thumb="7"

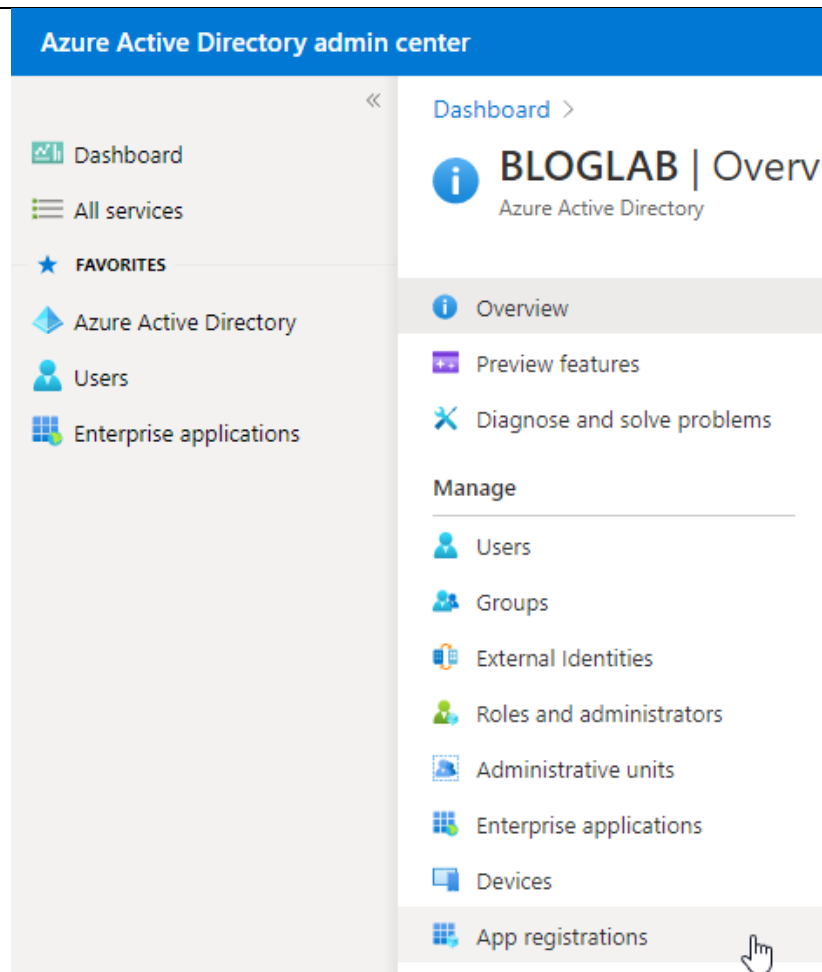
Information written to file: c:\scripts\psroomsigning\scripts\svc_roomsigning_2022.cert.local.txt
Save this info you need it later!

PS C:\scripts\PSRoomSigning\scripts>
```

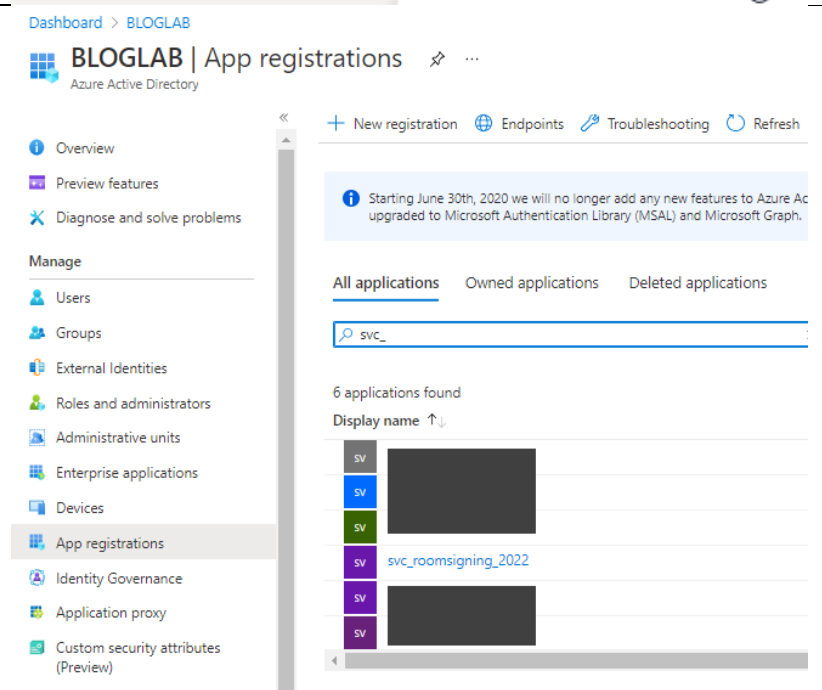
Login to Azure AD portal as Global Admin.

<https://aad.portal.azure.com>

Click Azure Active Directory and then “App Registrations”



Find the SPN created earlier and click on it.



Click on API permissions and click
Add a permission.

Dashboard > BLOGLAB > svc_roomsigning_2021

svc_roomsigning_2021 | API permissions

Search (Ctrl+/) Refresh Got feedback?

- Overview
- Quickstart
- Integration assistant

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration
- API permissions**
- Expose an API
- App roles
- Owners
- Roles and administrators | Preview

The "Admin consent required" column shows the default value for all the permissions the application needs. [Learn more](#)

Configured permissions

Applications are authorized to call APIs when they are granted permissions. All the permissions the application needs. [Learn more about permissions](#)

[+ Add a permission](#) ☒ Grant admin consent for BLOGLAB

API / Permissions name	Add a permission	Type	Description
No permissions added			

To view and manage permissions and user consent, try [Enterprise app](#)

Click on Microsoft Graph.

Request API permissions

Select an API

Microsoft APIs APIs my organization uses My APIs

Commonly used Microsoft APIs



Microsoft Graph

Take advantage of the tremendous amount of data in Office 365, Enterprise Mobility + Security, and Windows 10. Access Azure AD, Excel, Intune, Outlook/Exchange, OneDrive, OneNote, SharePoint, Planner, and more through a single endpoint.

Click Application permissions type in 'ca' in the search bar and select Calendar.Read

Click Add permissions on the bottom.

Request API permissions

[← All APIs](#)



Microsoft Graph

<https://graph.microsoft.com/> [Docs](#)

What type of permissions does your application require?

Delegated permissions

Your application needs to access the API as the signed-in user.

Application permissions

Your application runs as a background service or daemon without a signed-in user.

Select permissions

ca

Permission

Admin consent required

> AppCatalog

> Application

✓ Calendars (1)

☒ Calendars.Read ⓘ

Read calendars in all mailboxes

Yes

☐ Calendars.ReadWrite ⓘ

Read and write calendars in all mailboxes

Yes

> CallRecord-PstnCalls

> CallRecords

> Calls

> Policy

> ThreatIndicators

> UserAuthenticationMethod

Add permissions

Discard

Click Application permissions type in 'user.read' in the search bar and select User.Read

Click Add permissions on the bottom.

Request API permissions

×

[← All APIs](#)



Microsoft Graph

<https://graph.microsoft.com/> [Docs](#)

What type of permissions does your application require?

Delegated permissions

Your application needs to access the API as the signed-in user.

Application permissions

Your application runs as a background service or daemon without a signed-in user.

Select permissions

[expand all](#)

user.read

Permission

Admin consent required

> IdentityRiskyUser

✓ User (1)

☒ User.Read.All ⓘ

Read all users' full profiles

Yes

☐ User.ReadWrite.All ⓘ

Read and write all users' full profiles

Yes

Add permissions

Discard

Click on "Grant Admin consent for."	<div>Configured permissions</div> <div>Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent</div> <div><div><div>+</div>Add a permission</div><div><div>✓</div>Grant admin consent for BLOGLAB</div></div> <div><div>API / Permissions name</div><div>Grant admin consent for BLOGLAB</div></div> <div><div>▼</div>Microsoft Graph (2)</div> <div><div>Calendars.Read</div><div>Application</div><div>Read calendars in all mailboxes</div></div> <div><div>User.Read.All</div><div>Application</div><div>Read all users' full profiles</div></div> <div>To view and manage permissions and user consent, try Enterprise applications.</div>
Click Yes.	<div>Grant admin consent confirmation.</div> <div>Do you want to grant consent for the requested permissions for all accounts in BLOGLAB? This will update any existing admin consent records this application already has to match what is requested.</div> <div><div>Yes</div><div>No</div></div>
Both permissions are granted.	<div>Configured permissions</div> <div>Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent</div> <div><div><div>+</div>Add a permission</div><div><div>✓</div>Grant admin consent for BLOGLAB</div></div> <div><div>API / Permissions na...</div><div>Type</div><div>Description</div><div>Admin consent req...</div><div>Status</div></div> <div><div>▼</div>Microsoft Graph (2)</div> <div><div>Calendars.Read</div><div>Application</div><div>Read calendars in all mailboxes</div><div>Yes</div><div>✔ Granted for BLOGLAB</div><div>...</div></div> <div><div>User.Read.All</div><div>Application</div><div>Read all users' full profiles</div><div>Yes</div><div>✔ Granted for BLOGLAB</div><div>...</div></div> <div>To view and manage permissions and user consent, try Enterprise applications.</div>
The SPN is now setup for the PSRoomSigning script.	

Prepare for hosting the webserver on Azure

You need a Windows computer to configure these parts, so first create the SPN as described in "Create the Service Principal Name (SPN)" in the Preparations chapter.

On the Windows computer where you created the SPN run the PSRoomSigning_CreateResources2022.ps1 script again.

This script will create the webserver components in Azure Resource Manager and give the SPN the required permissions to save files on the webserver.

If you are running the scripts on Windows Server only you can skip this part.

Run the script again. Choose option 2.	<pre> Administrator: Windows PowerShell PS C:\scripts\PSRoomSigning\scripts> .\PSRoomSigning_CreateResources2022.ps1 Welcome at PSRoomSigning PSRoomSigning_CreateResources.ps1 1 - Create the SPN (You need to give it Directory.Read and Calendar.Read API permissions via Azure AD Portal). 2 - Create Azure App Service (Azure WebServer) and give SPN permissions to Azure WebServer Enter the number: 2 </pre>
---	---

Select the correct Azure Subscription

The screenshot shows a Windows PowerShell terminal window with the following content:

```
PS C:\OneDrive\SCRIPTS\PSRoomSigning_Blog> .\PSRoomSigning_CreateResources.ps1
Welcome at: PSRoomSigning PSRoomSigning_CreateResources.ps1

1 - Create the SPM (You need to give it Calendar.Read API permissions via Azure AD Portal).
2 - Create Azure App Service (Azure WebServer) and give SPM permissions to Azure WebServer

Enter the number: 2

2: First we need to login to AzureAD and Az, Connect-AzureAD and Connect-AzAccount, login in with Global Admin credentials and Owner permissions on AzureRM .

WARNING: Unable to acquire token for tenant 'e[REDACTED]' with error
'SharedTokenCacheCredential authentication unavailable. Token acquisition failed for user
'admin@365064244.onmicrosoft.com. Ensure that you have authenticated with a developer tool that supports Azure single
sign on.'
```

A dialog box titled "Select the correct subscription:" is overlaid on the terminal. It contains a "Filter" input field, an "Add criteria" button, and a table with the following data:

Name	Id	TenantId	State
MSDN Azure	5[REDACTED]	0	Enabled

The dialog box also has "OK" and "Cancel" buttons at the bottom right.

Select the SPN created earlier.

[illegible]

Select the Azure location.

Select your preferred location:

Filter

+ Add criteria ▼

Location	DisplayName	Providers
northcentralus	North Central US	{Microsoft.DesktopVirtualization, microsoft.insights, Microsoft.Aut
southcentralus	South Central US	{Microsoft.DesktopVirtualization, microsoft.insights, Microsoft.Aut
northeurope	North Europe	{Microsoft.DesktopVirtualization, microsoft.insights, Microsoft.Aut
westeurope	West Europe	{Microsoft.DesktopVirtualization, microsoft.insights, Microsoft.Aut
japanwest	Japan West	{microsoft.insights, Microsoft.Automation, Microsoft.KeyVault, Mic
japaneast	Japan East	{microsoft.insights, Microsoft.Automation, Microsoft.KeyVault, Mic
brazilsouth	Brazil South	{microsoft.insights, Microsoft.Automation, Microsoft.KeyVault, Mic
australiaeast	Australia East	{microsoft.insights, Microsoft.Automation, Microsoft.KeyVault, Mic
australiasoutheast	Australia Southeast	{microsoft.insights, Microsoft.Automation, Microsoft.KeyVault, Mic

< >

OK Cancel

Enter resource group name and the WebApp prefix and let the script create the resources.

AzureRM info and FTP credentials are displayed, copy them for later reference. This info is also saved in a TXT file named after the resource group in the script folder.

```

Administrator: Windows PowerShell

ObjectType      : ServicePrincipal
CanDelegate     : False
Description     : 
ConditionVersion : 
Condition       : 

Id              : admin@onmicrosoft.com
Type            : User
Tenants         : { }
AccessToken     : 
Credential      : 
TenantId        : { }
CertificateThumbprint : 
ExtendedProperties : { }

Save this info you need it later!
Azure RM - Information
Resource Group : RG-PSRoomSigning
Resource Region : westeurope
App service    : webapp050622030632
App service plan: webapp050622030632

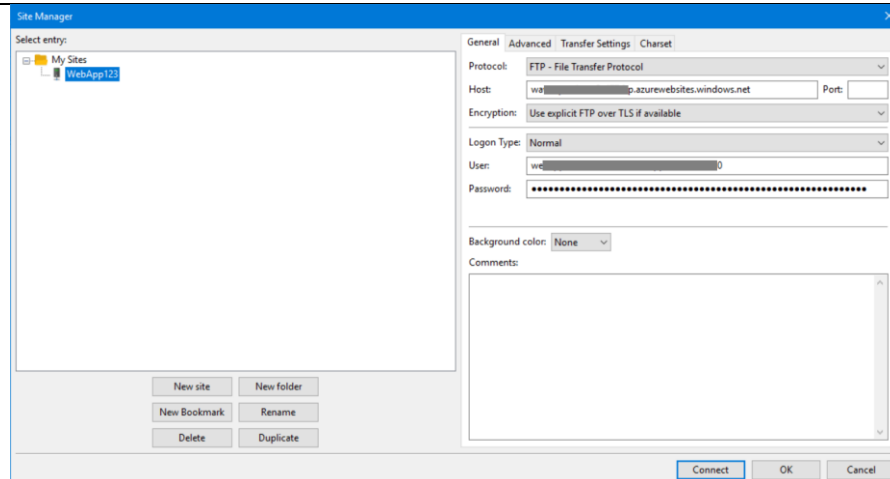
FTP Url         : ftp://.ftp.azurewebsites.windows.net/site/wwwroot
FTP username    : webapp050622030632
FTP password    : q*****s

PS C:\scripts\PSRoomSigning\scripts>

```

On the Windows Computer install any FTP client and enter the FTP credentials.

In the screenshot you see File Zilla.

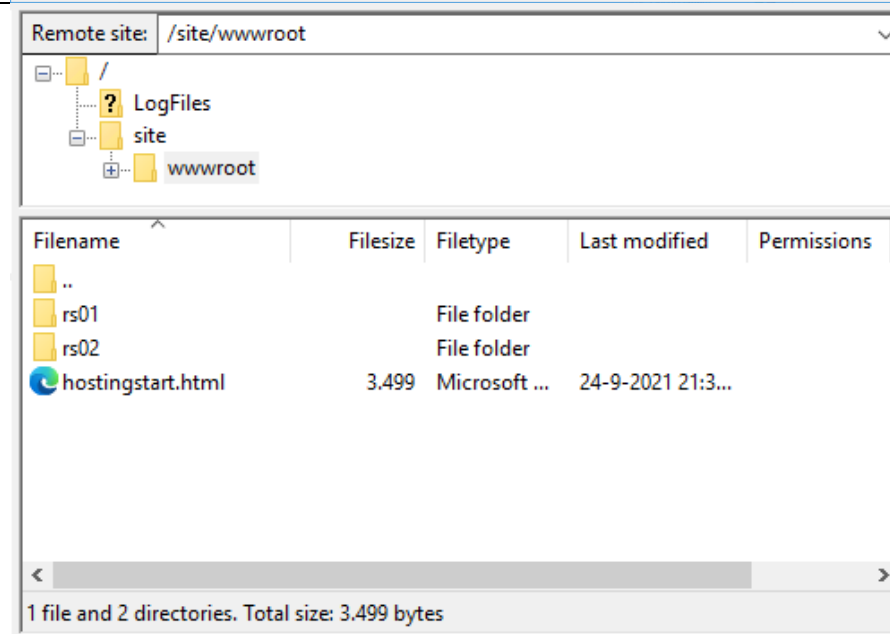


On the FTP folder, browse to /site/wwwroot and create one or more subfolders for the different PSRoomSigning instances.

In the screenshot you see:

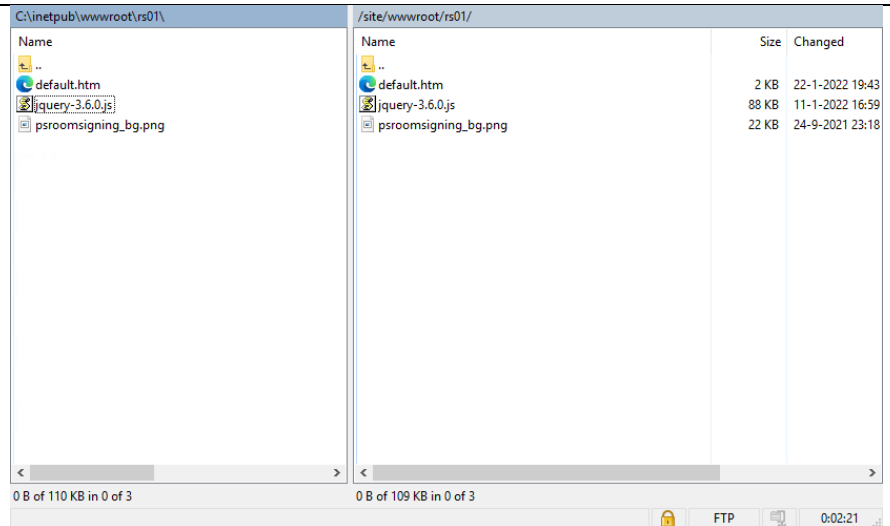
.\rs01
.\rs02

You can delete the hostingstart.html it is not required.



Use the FTP client to upload these files to the to the corresponding folders.

default.htm
jquery-3.6.0.js
psroomsigning_bg.png



Prepare PSRoomSigning2022.ps1 script

In the table below are all configurable items from the “PSRoomSigning2022.ps1” script.

For running on Windows Server, the first setting (`$SaveOnAzureAppService`) needs to be turned to `$false`.

You can run multiple instances of this script, for different floors, buildings or whatever suits your needs. You need to create a folder structure (Windows) or multiple Runbooks (Azure) to accommodate for multiple instances. Each copy of the script needs to be configured separately.

[PSRoomSigning2022.ps1]
<code>\$SaveOnAzureAppService=\$false</code> False – save on local disk for Windows Server True – save on Azure Web Service <i>Required Setting</i>
<code>\$AzureAppServiceRelativePath = "rs01"</code> <code>\$AzureAppServiceName = "webapp2021"</code> <code>\$AzureAppServiceResourceGroup = "rg-psroomsigning"</code> Parameters to find the Azure Web App Service and relative path for multiple instances. Values are listed when you choose options 2 from the script: “PSRoomsigning_CreateResources.ps1” <i>Required Setting for Azure Web Server target.</i>
<code>\$WindowsBasedLocation = "C:\inetpub\wwwroot\rs01\roomsigning.html"</code> Path to the local wwwroot folder, including the filename. <i>Required Setting for Windows Server target</i>
<code>\$TenantID="XXXXXXXX-YYYY-ZZZZ-XXXX-XXXXXXXXXXXX"</code> <code>\$AppID="11111111-2222-3333-4444-555566667777"</code> <code>\$Thumb="1234567890ABCDEF1234567890ABCDEF12345678"</code> Replace these variables with the info after running option 1 from this script: “PSRoomsigning_CreateResources2022.ps1” <i>Required Setting</i>
<code>\$rooms="meetingroom1@domain.com","meetingroom2@domain.com"</code> Enter one or more meeting room email addresses in this variable. These rooms will be read for meeting items. <i>Required Setting</i>
<code>\$ItembackgroundColor="848284" # Item level background color default: 848284</code> <code>\$RoomNameColor="000000" # Room name color default: 000000</code> <code>\$ItemColor="FFFFFF" # Item color default: FFFFFFFF</code> <code>\$DateStampColor="000000" # DateStamp color default: 000000</code> Variables for configuring colors.
<code>\$maxcalendaritems=9 # Maximum number of items on the screen.</code> <code>\$maxsubjectlength=53 # Truncates the subject on maximum chars</code> <code>\$skiplines=3 # Add empty lines from top of screen to align with background picture.</code> Variables speak for themselves just play with the values if default doesn’t look good.

```
$datestamp=$true # Add the date on the bottom of the screen.  
$ShowSnapshotTimeStamp=$false # Adds a timestamp when the snapshot/html file was  
created, designed for troubleshooting but can be used in production also.
```

Choose true/false only for these two.

```
$timezoneID="W. Europe Standard Time" # Use this command to list alle available  
timezones: get-timezone -ListAvailable copy the ID in this line.  
$CultureID ="nl-NL" # Use this command to list all available  
cultures, put the info from the NAME column in this variable:  
[System.Globalization.CultureInfo]::GetCultures([System.Globalization.CultureType  
s]::AllCultures)
```

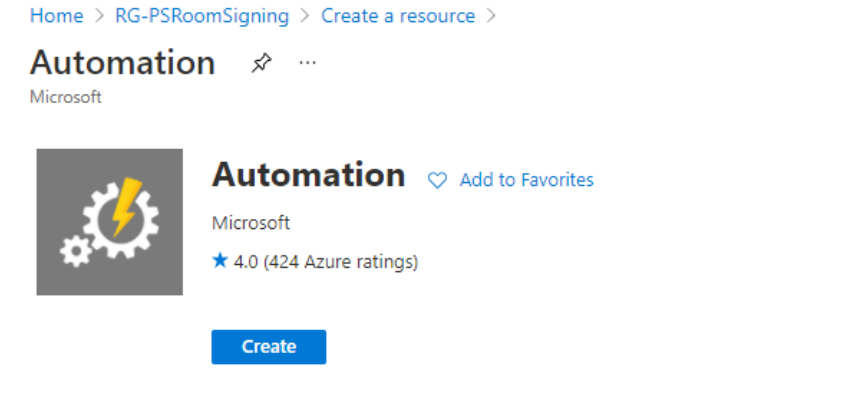
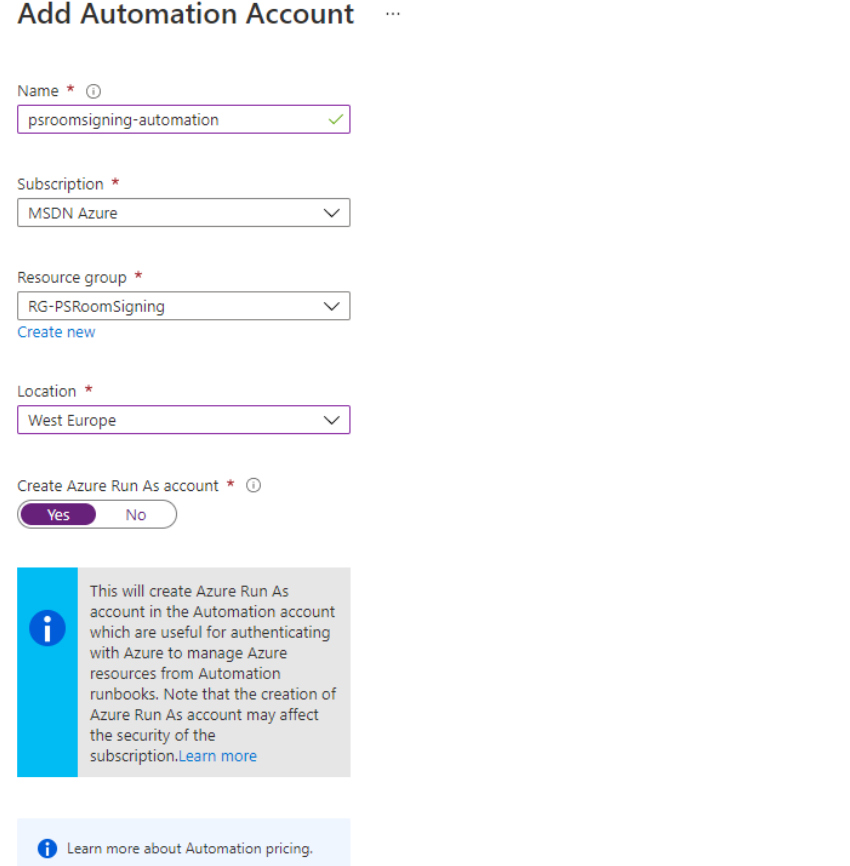
If you are not from The Netherlands you need to reconfigure these values, in the comments the PowerShell commands are listed how to find your values to put in the variables.

All other PowerShell commands do not need to be changed and should work as is.

Installation

Install on Azure RM

In this section we will create and configure Azure Automation PowerShell RunBooks. Login to Azure Portal with enough permissions to create new resources. I will create the new resources in the Resource Group created earlier with the script. RG-PSRoomSigning.

<p>From the resource group click Create, search for Automation, and click Create.</p>	
<p>Enter the required information.</p> <p>Create Azure Run As Account: YES</p> <p>Click Create at the bottom.</p>	

<p>Your resource group contents will look something like this.</p>	<div><div><div><div><div></div><div>AzureAutomationTutorial (psroomsigning-automation/AzureAutomationTutorial)</div></div><div><div></div><div>AzureAutomationTutorialPython2 (psroomsigning-automation/AzureAutomationTutorialPython2)</div></div><div><div></div><div>AzureAutomationTutorialScript (psroomsigning-automation/AzureAutomationTutorialScript)</div></div><div><div></div><div>psroomsigning-automation</div></div><div><div></div><div>webapp243621093610</div></div><div><div></div><div>webapp243621093610</div></div></div></div></div>
<p>Click on psroomsigning-automation.</p> <p>On the Shared Resources section click Certificates.</p> <p>(You need to scroll down for this)</p> <p>Click on Add a certificate.</p>	<div><div><div>Home > RG-PSRoomSigning > psroomsigning-automation</div><div><div>psroomsigning-automation Certificates</div><div>Automation Account</div></div><div><div><div>Search (Ctrl+ /)</div><div><</div><div>+ Add a certificate</div><div>🔄 Refresh</div></div><div><div>Shared Resources</div><div><div><div>Schedules</div><div>Modules</div><div>Modules gallery</div><div>Python packages</div><div>Credentials</div><div>Connections</div><div>Certificates</div><div>Variables</div></div></div></div><div><div>Name</div><div>AzureRunAsCertificate</div></div></div></div></div>
<p>Browse to the location where you ran the PSRoomSignin_CreateResources.ps1 script, in that folder you should find a PFX file that contains the authentication certificate for the SPN.</p> <p>Enter the password of the PFX and select NO for exportable.</p> <p>Note keep this certificate in a safe place.</p> <p>On the bottom click create.</p>	<div><div><div>Add a certificate</div><div>×</div></div><div><div>Name *</div><div>svc_roomsigning_2022</div></div><div><div>Description</div><div></div></div><div><div>Upload a certificate file (.cer,.pfx) *</div><div>svc_roomsigning_2022.cert.local.pfx</div></div><div><div>Password *</div><div>.....</div></div><div><div>Exportable *</div><div>Yes</div><div>No</div></div></div>
<p>On the Shared Resources section click Modules.</p> <p>Click Browse gallery.</p>	<div><div><div>Home > RG-PSRoomSigning > psroomsigning-automation</div><div><div>psroomsigning-automation Modules</div><div>Automation Account</div></div><div><div><div>Search (Ctrl+ /)</div><div><</div><div>+ Add a module</div><div>🔄 Update Azure modules</div><div>📖 Learn about module updates</div><div>🔍 Browse gallery</div><div>🔄 Refresh</div></div><div><div>Shared Resources</div><div><div><div>Schedules</div><div>Modules</div><div>Modules gallery</div><div>Python packages</div><div>Credentials</div><div>Connections</div></div></div></div><div><div>Search modules...</div><div><div><div>Name</div><div>Last modified</div><div>Status</div></div><div><div>AuditPolicyDsc</div><div>9/20/2021, 3:08 PM</div><div>Availat</div></div><div><div>Azure</div><div>9/20/2021, 2:59 PM</div><div>Availat</div></div><div><div>Azure.Storage</div><div>9/20/2021, 3:05 PM</div><div>Availat</div></div><div><div>AzureRM.Automation</div><div>9/20/2021, 3:03 PM</div><div>Availat</div></div><div><div>AzureRM.Compute</div><div>9/20/2021, 3:03 PM</div><div>Availat</div></div></div></div></div></div></div>

Search for and import these modules:


- Az.accounts
- Az.websites
- Microsoft.Graph.Authentication
- Microsoft.Graph.Calendar
- Microsoft.Graph.Users

If asked choose runtime version 5.1.

Home > RG-PSRoomSigning > psroomsigning-automation >

Browse Gallery

Search: az.accounts

**Az.Accounts**
Microsoft Azure PowerShell - Accounts credential management cmdlets for Azure Resource Manager in Windows PowerShell and PowerShell Core.
Tags: Azure ResourceManager ARM Accounts Authentication Environment Subscription PSModule PSEdition_Core PSEdition_Desktop

Created by: azure-sdk
48498299 downloads
Last updated: 9/7/2021

If you import them in this order, you should not have trouble with dependencies. If a dependency is already importing but still asks for it, the import is not finished yet.

Scroll back up and click on Runbooks.

Click on Create a runbook

psroomsigning-automation | Runbooks

Automation Account

Search (Ctrl+/)

+ Create a runbook | Import a runbook | Browse gallery | Learn more

Search runbooks...

Name	Authoring status
AzureAutomationTutorial	✓ Published
AzureAutomationTutorialPython2	✓ Published
AzureAutomationTutorialScript	✓ Published

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Configuration Management

Inventory

Change tracking

State configuration (DSC)

Update management

Update management

Process Automation

Runbooks

Enter a name and select the runbook type: PowerShell

On the bottom click Create.

If asked choose runtime version 5.1.

Create a runbook

Name * ⓘ

psroomsigning_rs01 ✓

Runbook type * ⓘ

PowerShell ▾

Description

Copy/Paste the PSRoomSigning2022.ps1 file.

In the Edit PowerShell Runbook change the parameters you want to change.

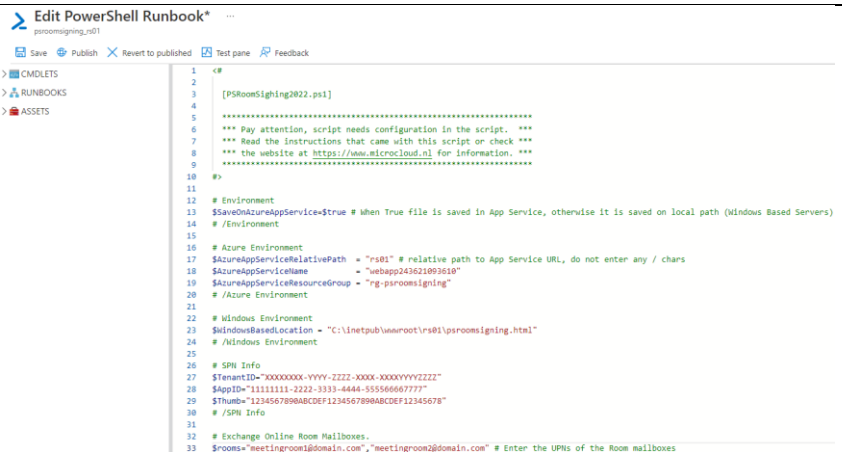
Check the “Configuration in PowerShell script” section in the Preparations chapter.

Make sure you enter the correct SPN info, and the Azure Environment resources.

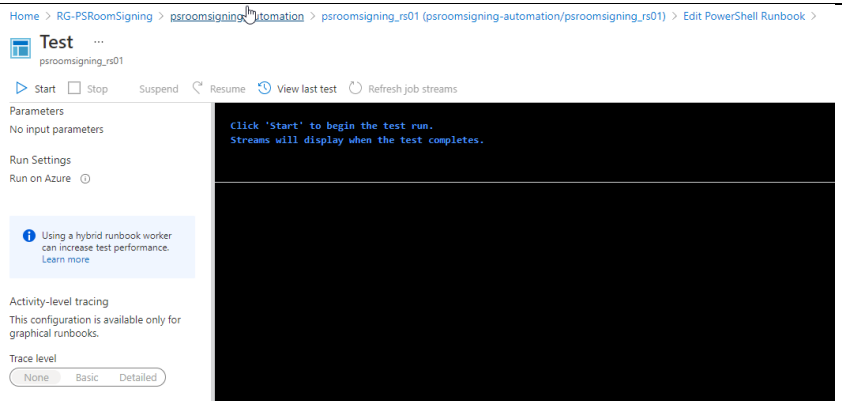
Click Save

Click the Test pane to test the script.

In the test pane, click Start.



```
1 <#
2
3 [PSRoomSigning2022.ps1]
4
5 *****
6 *** Pay attention, script needs configuration in the script. ***
7 *** Read the instructions that came with this script or check ***
8 *** the website at https://www.microcloud.nl for information. ***
9 *****
10 #>
11
12 # Environment
13 $SaveOnAzureAppService=$true # When True file is saved in App Service, otherwise it is saved on local path (Windows Based Servers)
14 # /Environment
15
16 # Azure Environment
17 $AzureAppServiceRelativePath = "rs01" # relative path to App Service URL, do not enter any / chars
18 $AzureAppServiceName = "webapp243621093610"
19 $AzureAppServiceResourceGroup = "rg-psroomsigning"
20 # /Azure Environment
21
22 # Windows Environment
23 $WindowsBasedLocation = "C:\inetpub\wwwroot\rs01\psroomsigning.html"
24 # /Windows Environment
25
26 # SPN Info
27 $TenantID="XXXXXXX-YYYY-ZZZZ-XXXX-XXXXYYYYZZZZ"
28 $AppID="11111111-2222-3333-4444-555566677777"
29 $Thumb="1234567890ABCDEF1234567890ABCDEF12345678"
30 # /SPN Info
31
32 # Exchange Online Room Mailboxes.
33 $rooms="meetingroom@domain.com","meetingroom2@domain.com" # Enter the URIs of the Room mailboxes
```



The result should look something like this.

If you get errors, check the SPN info in the script, check the certificate and de API permissions of the SPN. Also a wrong resource groupname, or webapp name can cause errors.

Also check if all required PowerShell modules are imported.

Home > psroomsigning-automation > psroomsigning_rs01 (psroomsigning-automation/psroomsigning_rs01) > Edit PowerShell Runbook >

Test ...
psroomsigning_rs01

Start Stop Suspend Resume View last test Refresh job streams

Parameters
No input parameters

Run Settings
Run on Azure

Using a hybrid runbook worker can increase test performance. [Learn more](#)

Activity-level tracing
This configuration is available only for graphical runbooks.

Trace level
None Basic Detailed

Completed

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

Environments

([AzureChinaCloud, AzureChinaCloud], [AzureCloud, AzureCloud], [AzureGermanCloud, AzureGermanCloud], [AzureUSGovernme ...

After the test you should see the roomsigning.html in the wwwroot/rs01 folder.

/site/wwwroot/rs01/

Name	Size	Changed	Rig
default.htm	2 KB	22-1-2022 19:43	
jquery-3.6.0.js	88 KB	11-1-2022 16:59	
psroomsigning_bg.png	22 KB	24-9-2021 23:18	
roomsigning.html	2 KB	28-1-2022 22:20	

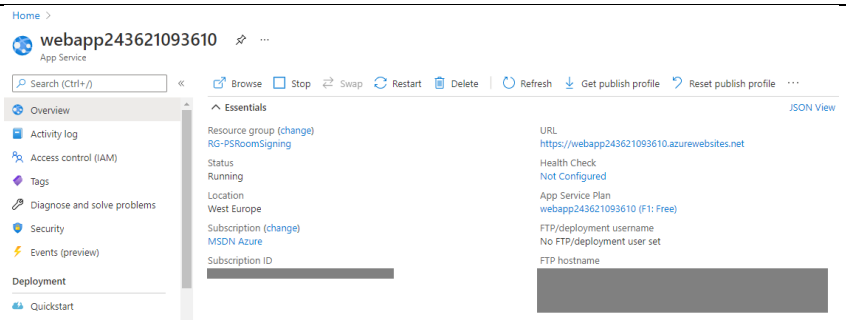
0 B of 111 KB in 0 of 4

FTP 0:01:35

Test Azure App Service

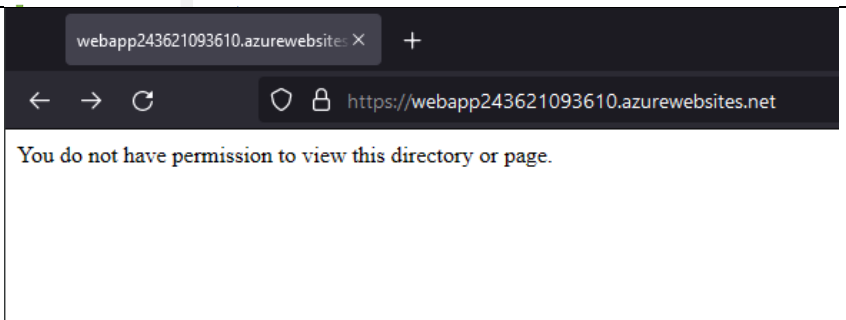
In Azure portal, browse to you App Service object.

You can copy out the URL from this page or click Browse to open the website in your browser.

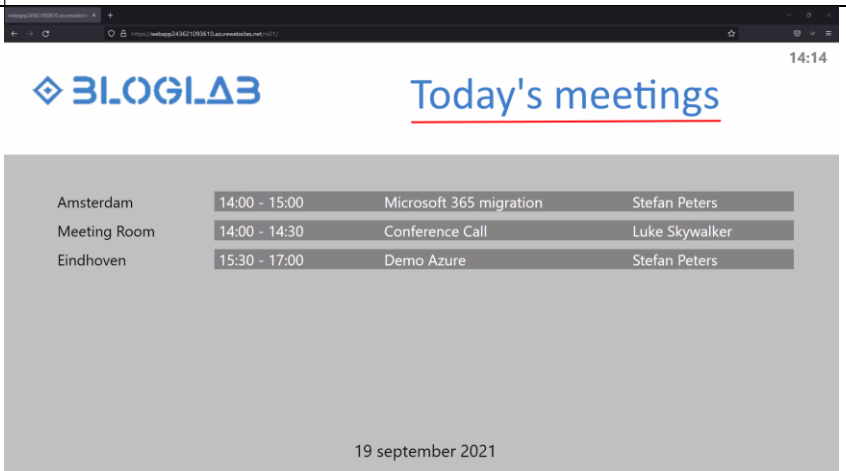


If you have deleted the hostingstart.html file you will get this error.

Add the /rs01 to the url and check again!

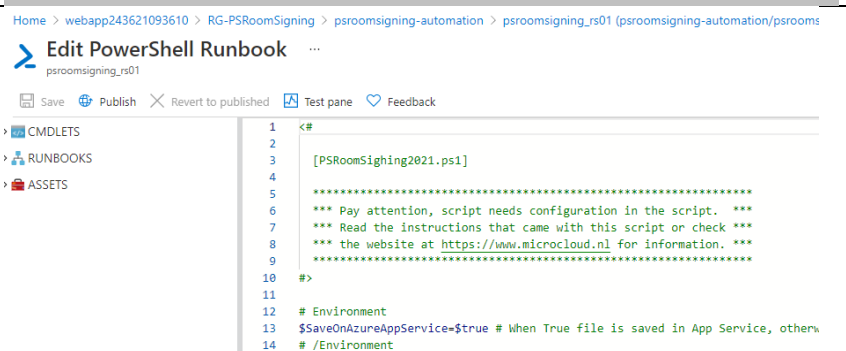


When you add the /rs01 the page should be possible.



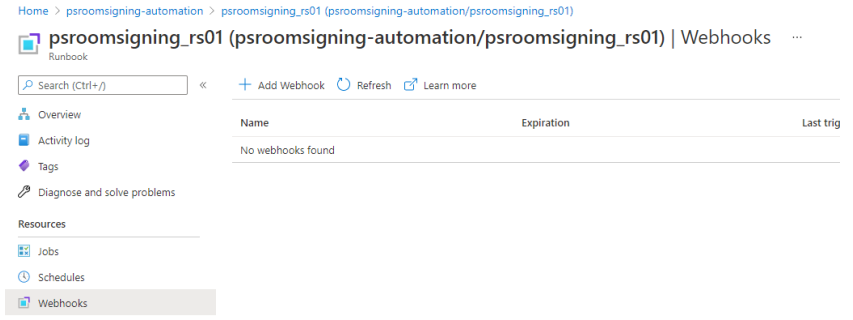
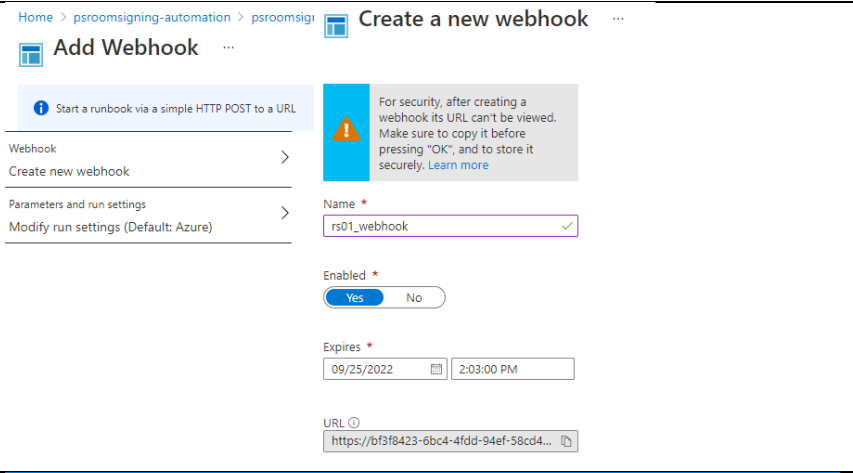
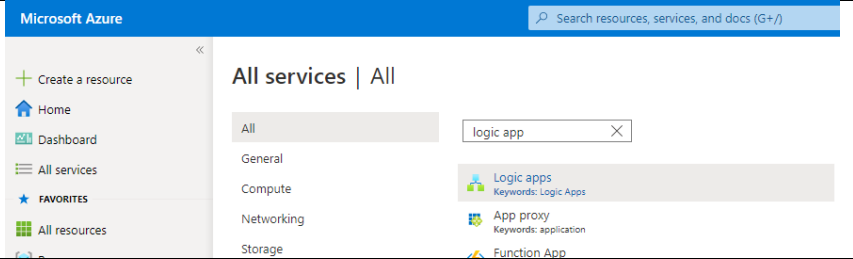
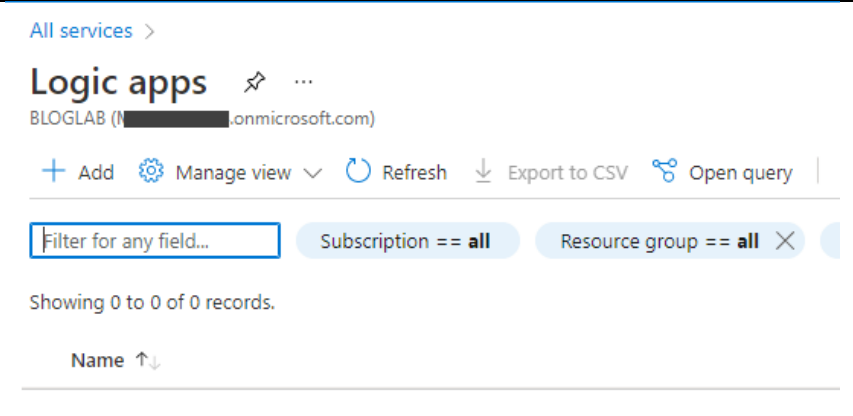
If the webpage is as expected you need to publish the script.






Go back to Edit PowerShell Runbook and click on Publish. Click yes to confirm.



Azure Scheduler

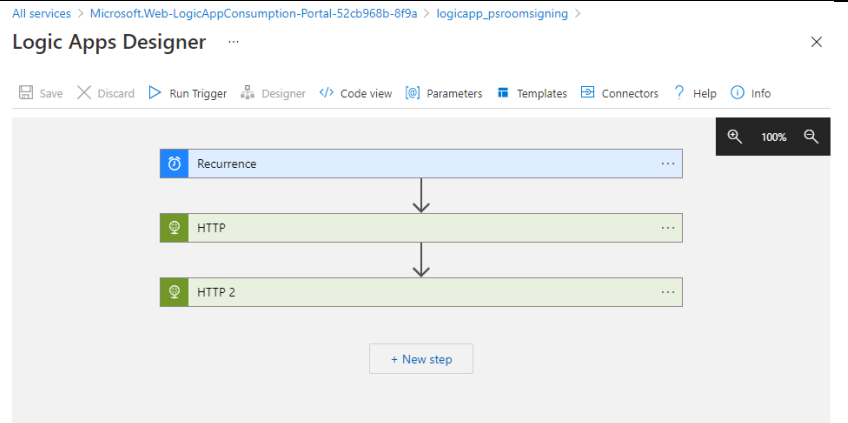
Although the Automation account allows scheduling the frequency is too long (minimum is 1 hour). I recommend running the script every 2-5 minutes. Please adjust this to your needs.

<p>In Azure Portal go to your psroomsigning_rs01 runbook, in the Resources section click Webhooks and click Add Webhook.</p>	
<p>Name the Webhook, enable it and choose an expire date. Note this date in your calendar for updating.</p> <p>Make sure you copy the URL and paste it somewhere safe; it will not be shown again.</p> <p>Click Create in the Add Webhook section.</p>	
<p>On the left menu click All Services and search for Logic Apps</p> <p>Click Logic apps.</p>	
<p>Click Add.</p>	

<p>Select the resource group of your RoomSigning components.</p> <p>RG-PSRoomSigning</p> <p>Type: Consumption</p> <p>Name the logic app and select your region.</p> <p>Next.</p>	<div><h2>Create Logic App</h2><div>BasicsTagsReview + create</div><p>Create a logic app, which lets you group workflows as a logical unit for easier management, deployment and sharing of resources. Workflows let you connect your business-critical apps and services with Azure Logic Apps, automating your workflows without writing a single line of code.</p><p>Project Details</p><p>Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.</p><div><div>Subscription * ⓘ</div><div>MSDN Azure</div></div><div><div>Resource Group * ⓘ</div><div>RG-PSRoomSigning</div><div>Create new</div></div><p>Instance Details</p><div><div>Type *</div><div><div><input checked="" type="radio"/> Consumption</div><div><input type="radio"/> Standard</div></div><div>Looking for the classic consumption create experience? Click here</div></div><div><div>Logic App name *</div><div>logicapp_psroomsigning</div></div><div><div>Region *</div><div>West Europe</div></div><div><div>Enable log analytics *</div><div><div><input type="radio"/> Yes</div><div><input checked="" type="radio"/> No</div></div></div></div> <div>⚠ There are no log analytics workspace resources in the selected subscription. In order to enable log analytics, either create a new log analytics workspace resource or switch to a subscription which already has one.</div>										
<p>Click Review + Create on the bottom of the page.</p>	<div><h2>Create Logic App</h2><div>BasicsTagsReview + create</div><p>Summary</p><div> Logic App by Microsoft</div><p>Details</p><table><tr><td>Subscription</td><td>5</td></tr><tr><td>Resource Group</td><td>RG-PSRoomSigning</td></tr><tr><td>Name</td><td>logicapp_psroomsigning</td></tr><tr><td>Region</td><td>westeurope</td></tr><tr><td>Log analytics</td><td>Disabled</td></tr></table></div>	Subscription	5	Resource Group	RG-PSRoomSigning	Name	logicapp_psroomsigning	Region	westeurope	Log analytics	Disabled
Subscription	5										
Resource Group	RG-PSRoomSigning										
Name	logicapp_psroomsigning										
Region	westeurope										
Log analytics	Disabled										
<p>When object is created click on Go to resource.</p> <p>Find the trigger “Recurrence”</p> <p>Click on it.</p>	<div><p>Start with a common trigger</p><p>Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich cc</p><div><div>When a message is received in a Service Bus queue</div><div>When a HTTP request is received</div><div>Recurrence</div><div>When a new email is received in Outlook.com</div></div></div>										

<p>Default interval is 3 minutes.</p> <p>Change to your requirements.</p> <p>Click on New step.</p>	<div><div>Logic Apps Designer</div><div><div>Save</div><div>Discard</div><div>Run Trigger</div><div>Designer</div><div>Code view</div><div>Parameters</div><div>Templates</div><div>Connectors</div><div>Help</div><div>Info</div></div><div><div>Recurrence</div><div><div>Interval</div><div>3</div><div>Frequency</div><div>Minute</div></div><div>Add new parameter</div></div><div>+ New step</div></div>		
<p>Search for HTTP and select the default green HTTP button.</p>	<div><div>Choose an operation</div><div><div>http</div></div><div><div>For You</div><div>All</div><div>Built-</div></div><div><div><div>HTTP</div></div><div><div>Office 365 Outlook</div></div></div><div>HTTP</div></div>		
<p>Select method: POST</p> <p>Paste the Webhook URL in the URI part.</p> <p>Click on Save.</p>	<div><div>HTTP</div><div><div>Method</div><div>POST</div></div><div><div>URI</div><div>https://bf3f8423-6bc4-4fdd-94ef-58cd42f2dbb9.webhook.office-automation.net/webhooks?token=k9QHRlrGQVRjtsG%2f327T59K%2bw3cQTRRoZw9lLJqbc0l%3d</div></div><div><div>Headers</div><div>Enter key</div><div>Enter value</div></div><div><div>Queries</div><div>Enter key</div><div>Enter value</div></div><div><div>Body</div><div>Enter request content</div></div><div><div>Cookie</div><div>Enter HTTP cookie</div></div><div>Add new parameter</div><div>+ New step</div></div> <div><div>Dynamic content</div><div>Expression</div><div>NO DYNAMIC CONTENT AVAILABLE</div><div>There is no content available</div><div>INCLUDING DYNAMIC CONTENT</div><div>If available, dynamic content is automatically generated from the connectors and actions you choose for your flow.</div><div>Dynamic content may also be added from other sources.</div><div>Learn more about dynamic content</div></div>		
<p>To test if it works click on Run Trigger.</p>	<div><div>Logic Apps Designer</div><div><div>Save</div><div>Discard</div><div>Run Trigger</div><div>Designer</div><div>Code view</div><div>Parameters</div><div>Templates</div><div>Connectors</div><div>Help</div><div>Info</div></div><div><div>Recurrence</div><div>0s</div></div><div><div>HTTP</div><div>0s</div></div></div>		
<p>Check the FTP folder if the timestamp of roomsigning.html has changed. If it did the trigger works!</p>	<div><div>default.htm</div><div>2 KB</div><div>22-1-2022 19:43</div></div> <div><div>jquery-3.6.0.js</div><div>88 KB</div><div>11-1-2022 16:59</div></div> <div><div>psroomsigning_bg.png</div><div>22 KB</div><div>24-9-2021 23:18</div></div> <div><div>roomsigning.html</div><div>2 KB</div><div>28-1-2022 22:24</div></div>		

If you host multiple RoomSigning instances as Runbook you can add more Webhook HTTP POSTs in the same logic app, they will run on the same interval.



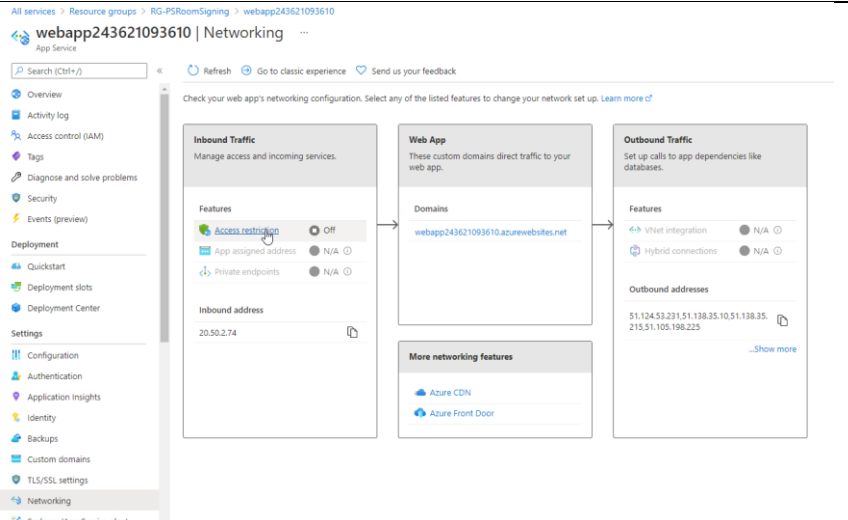
PUBLIC Visibility

If you followed the instructions, you have an Azure App Service that will host your HTML files for the PSRoomSigning. By default, the web service is open for the entire world to be viewed. You can change this by following the next steps.

In Azure Portal browse to your App Service.

In the Setting section click on "Networking"

Next click on Access restriction.



Click on Add rule

Name the rule, give it a priority number, and add your PUBLIC IP of your Office breakout internet.

Click Add rule.

Add as many PUBLIC IP rules as you need.

Access Restrictions

Access restrictions allow you to define lists of allow/deny rules to control traffic to your app. Rules are evaluated in priority order. If there are no rules defined then your app will accept traffic from any address. [Learn more](#)

webapp243621093610.azurewebsites.net webapp243621093610.scm.azurewebsites.net

+ Add rule

Priority	Name	Source	Endpoint status	HTTP headers	Action
1	Allow all	Any	Not configured		Allow

Add Access Restriction

General settings

Name

Action ☒ Allow ☐ Deny

Priority

Description

Source settings

Type

IP Address Block

HTTP headers filter settings

X-Forwarded-Host

X-Forwarded-For

X-Azure-FDID

X-FD-HealthProbe

Add rule

Create new rule
Successfully added new rule

After creating this rule an implicit Deny is added automatically.

Access Restrictions

Access restrictions allow you to define lists of allow/deny rules to control traffic to your app. Rules are evaluated in priority order. If there are no rules defined then your app will accept traffic from any address. [Learn more](#)

webapp243621093610.azurewebsites.net webapp243621093610.scm.azurewebsites.net

+ Add rule

Priority	Name	Source	Endpoint status	HTTP headers	Action
300	OfficeOnly	80.1.1.1/32	Not configured		Allow
2147483647	Deny all	Any	Not configured		Deny

Click on the web243.scm.*** section and enable "same restrictions as web***"

Access Restrictions

Access restrictions allow you to define lists of allow/deny rules to control traffic to your app. Rules are evaluated in priority order. If there are no rules defined then your app will accept traffic from any address. [Learn more](#)

webapp243621093610.azurewebsites.net webapp243621093610.scm.azurewebsites.net

☒ Same restrictions as webapp243621093610.azurewebsites.net

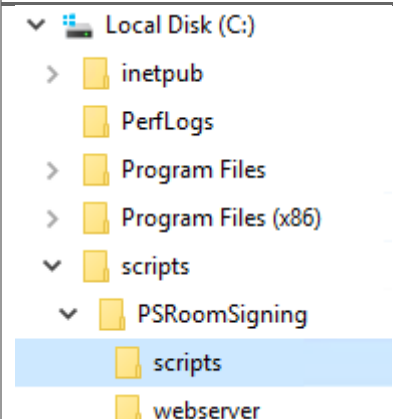
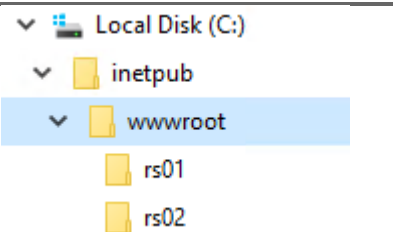
Install on Windows Server

On your Windows Server where you want to run the script, install the PowerShell modules “Az” and “Microsoft.Graph” on a recent and supported version of Windows. If you want to publish the result on a webserver install IIS or another webserver.

Commands to install the modules:

- Install-Module Az
- Install-Module Microsoft.Graph

Example folder structures for IIS webserver:

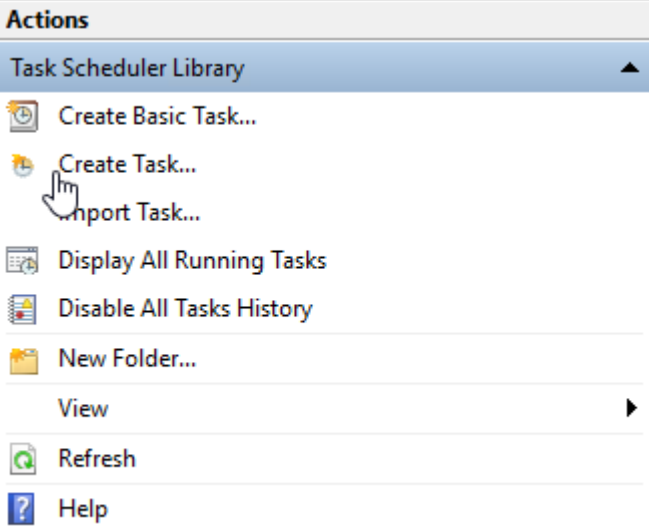
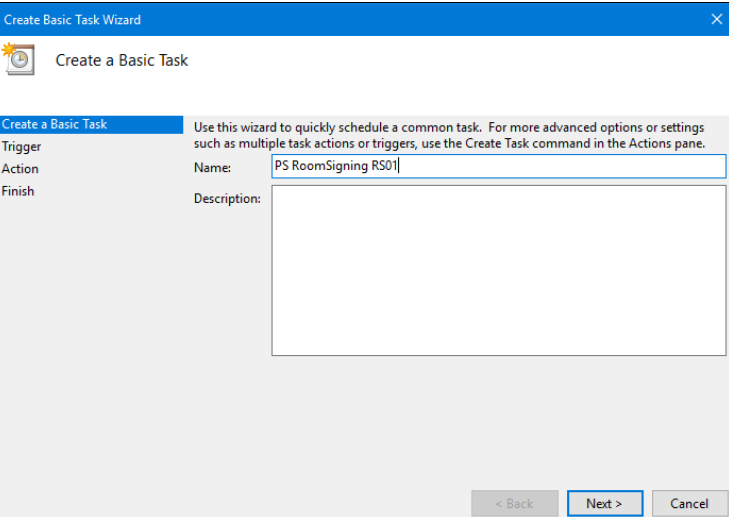
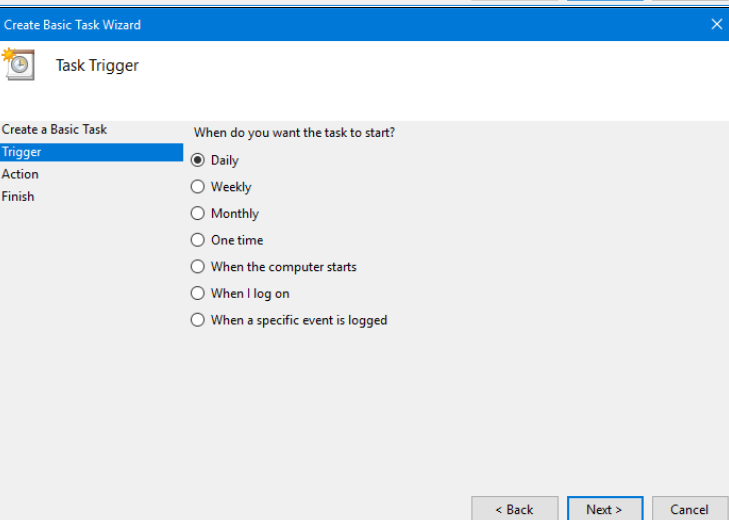
PowerShell Script	WebServer
 <p>Local Disk (C:)</p> <ul style="list-style-type: none"> inetpub PerfLogs Program Files Program Files (x86) scripts <ul style="list-style-type: none"> PSRoomSigning <ul style="list-style-type: none"> scripts webserver 	 <p>Local Disk (C:)</p> <ul style="list-style-type: none"> inetpub <ul style="list-style-type: none"> wwwroot <ul style="list-style-type: none"> rs01 rs02

C:\scripts\PSRoomSigning		
	\scripts	
		PSRoomSigning2022.ps1
		PSRoomSigning_CreateResources.ps1

C:\inetpub\wwwroot		
	\rs01	
		default.htm
		psroomsigning_bg.png
		jquery-3.6.0.js
		roomsigning.html
Optional	\rs02	
		default.htm
		psroomsigning_bg.png
		jquery-3.6.0.js
		roomsigning.html

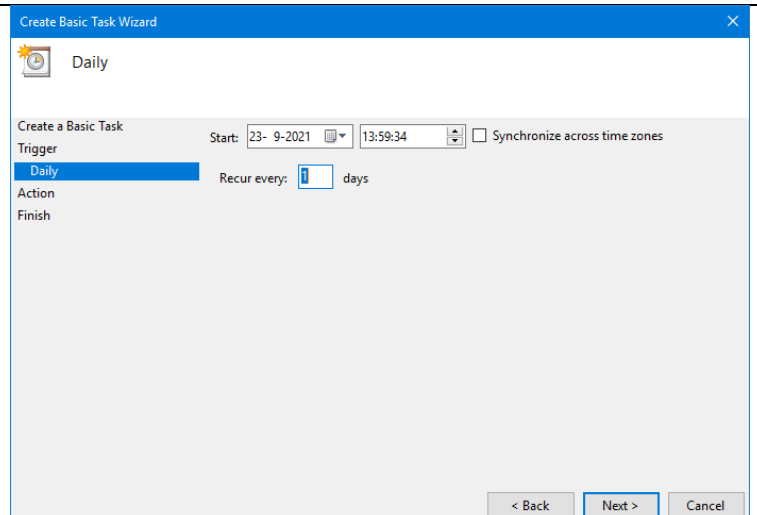
Copy the files you prepared earlier into the c:\inetpub\wwwroot\rs01 folder as shown in the above table. Other folder structures are also possible, this is an example. The *roomsigning.html* is not in the ZIP and will be created after running the PSRoomSigning2022.ps1 script for the first time.

Configure the scheduled task

<p>Open Task Scheduler.</p> <p>Click "Create Basic Task..."</p>	 <p>The screenshot shows the 'Task Scheduler Library' window. The 'Create Basic Task...' option is highlighted with a mouse cursor. Other visible options include 'Create Task...', 'Import Task...', 'Display All Running Tasks', 'Disable All Tasks History', 'New Folder...', 'View', 'Refresh', and 'Help'.</p>
<p>Enter a name for the scheduled task.</p> <p>Next</p>	 <p>The screenshot shows the 'Create Basic Task Wizard' dialog box, Step 1: 'Create a Basic Task'. The 'Name' field contains 'PS RoomSigning RSD1'. The 'Description' field is empty. The 'Next >' button is highlighted.</p>
<p>Choose Daily, Next</p>	 <p>The screenshot shows the 'Create Basic Task Wizard' dialog box, Step 2: 'Task Trigger'. The 'Trigger' tab is selected. The 'When do you want the task to start?' section has 'Daily' selected with a radio button. Other options include 'Weekly', 'Monthly', 'One time', 'When the computer starts', 'When I log on', and 'When a specific event is logged'. The 'Next >' button is highlighted.</p>

Choose any start time, could be just before business day, for 24/7 display start time doesn't matter.

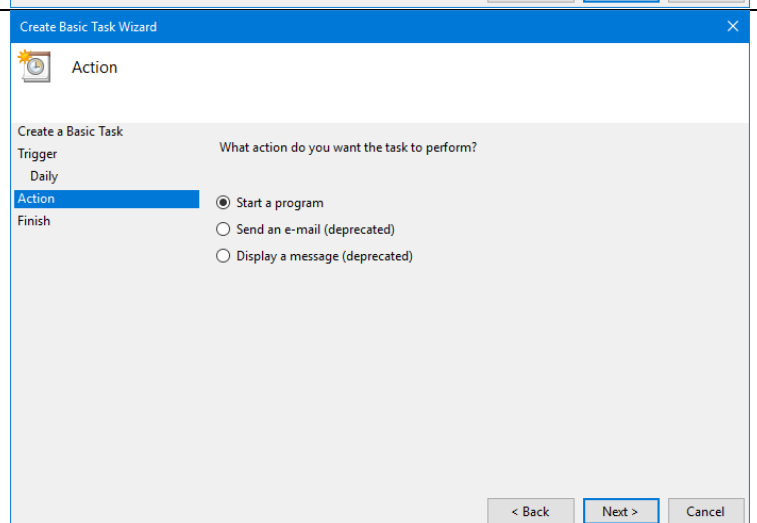
Next.



Action

Start a program

Next.



Browse to the powershell.exe in the folder:
`C:\Windows\System32\Windows PowerShell\v1.0\`

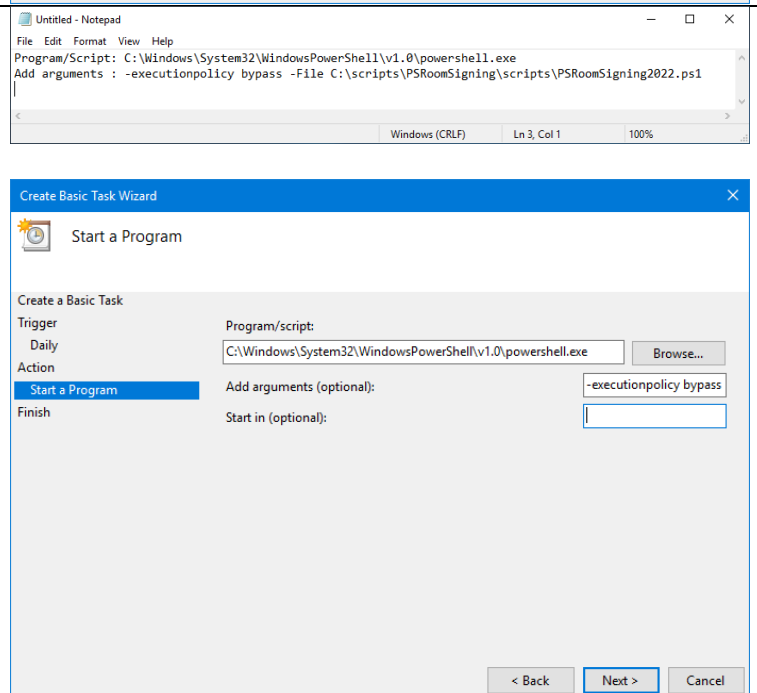
In the Add Arguments type these parameters

`-executionpolicy bypass -File`

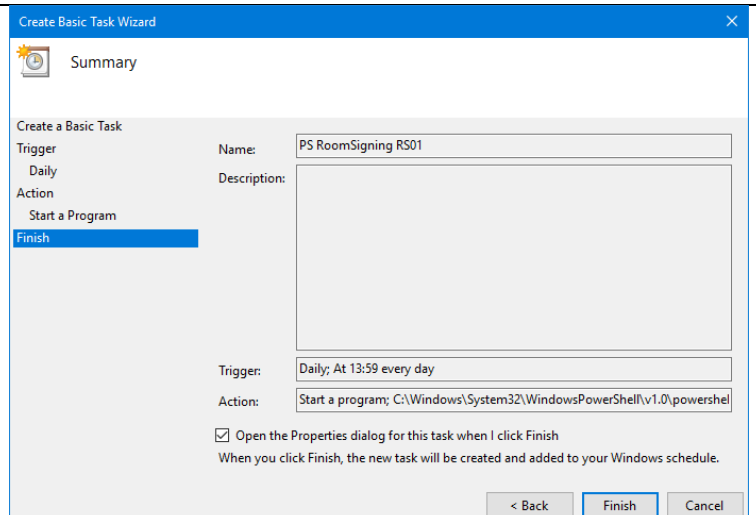
Followed by the full path to the script.

TIP: I usually prepare the paths and parameters in notepad see screenshot. That helps in getting the commands and arguments in first time right.

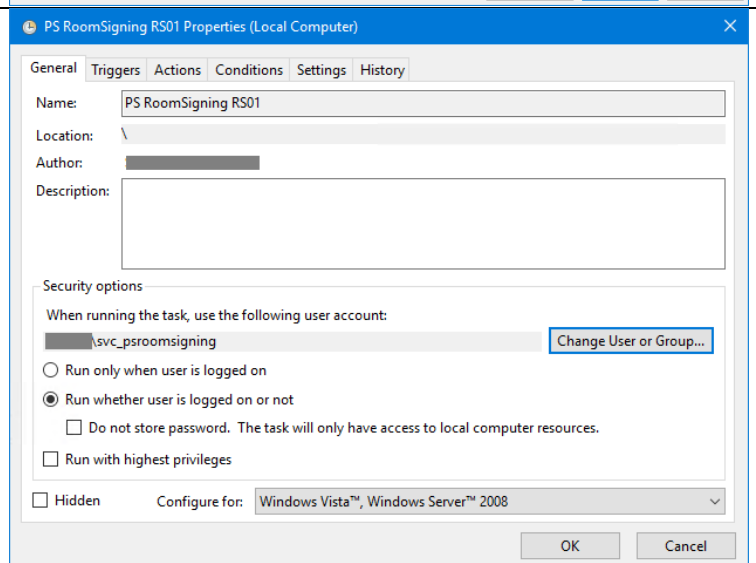
Next.



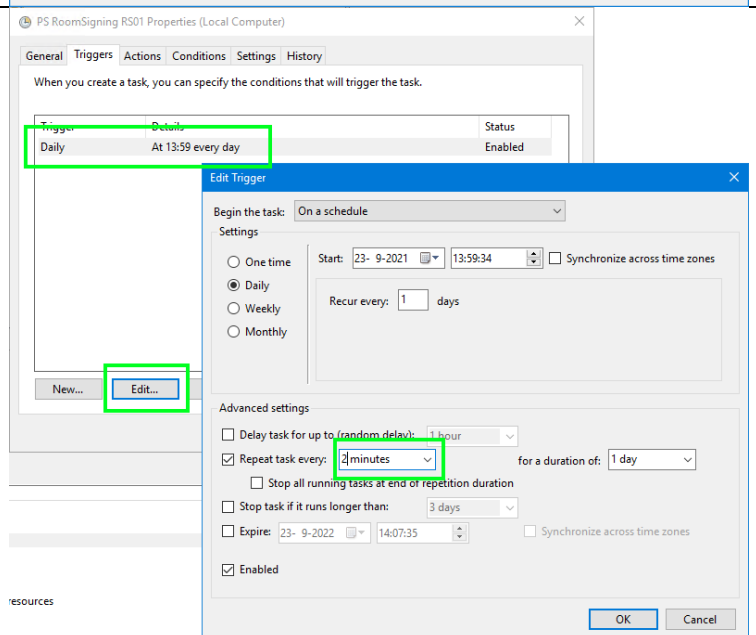
Finish.



Highest privileges should not be necessary.

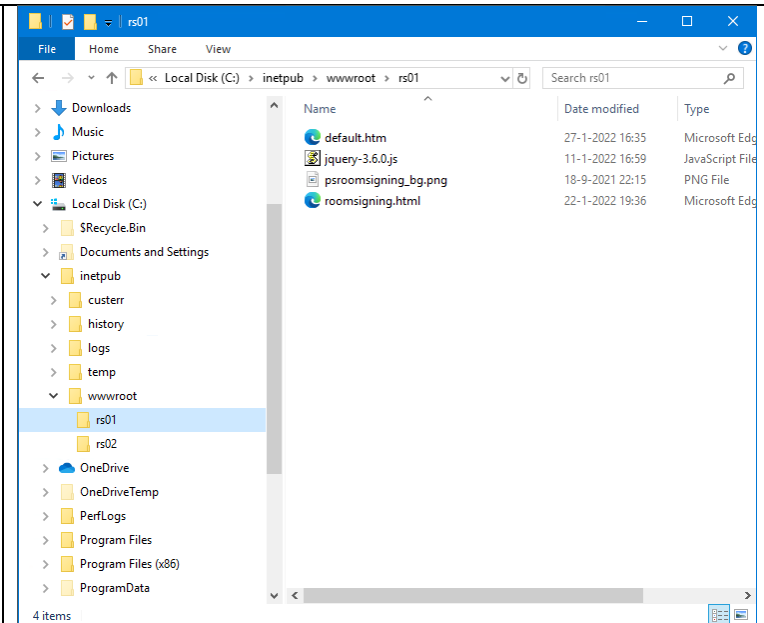


Keep clicking OK until all settings are saved. If prompted enter the service account password.



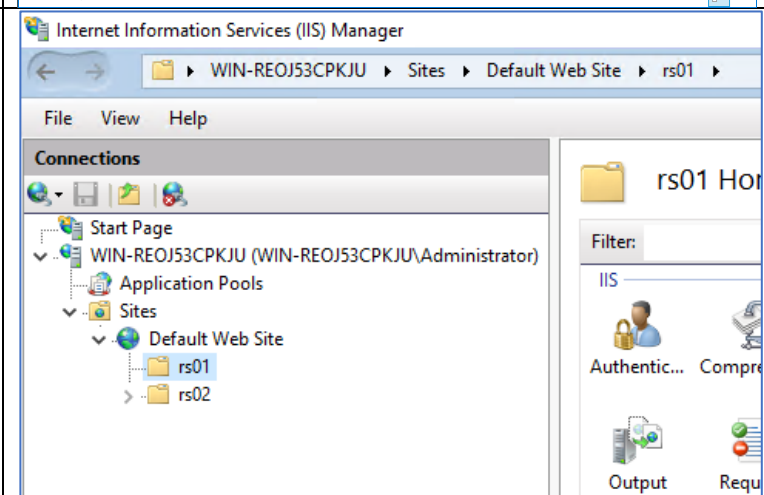
After at least one successful run the folder should look like this.

default.htm
jquery-3.6.0.js
psroomsigning_bg.png
roomsigning.html (generated by PowerShell script)



Your IIS Manager could look like this:

Make sure you comply to you company security policies for webserver like using a certificate for https.



Configure your display devices

You could use any device that can display this website like Linux, Android, or Windows.

Suggestions:

- Old tablet that connects via WiFi to the webserver.
- Old PC connected to a TV mounted on the lunchroom wall of your office.

You can also distribute the URL to your users or embed it on your intranet so your users can quickly check if any meeting room is busy or not.

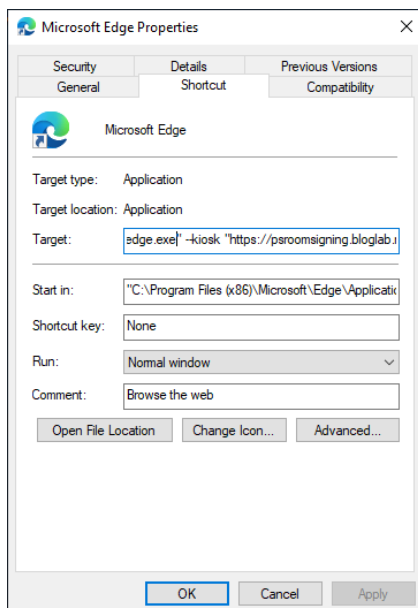
Windows 10 Client configuration

In the real world I have configured a Windows 10 client to autologin and autostart a webbrowser with a special parameter to show the PSRoomSigning page. My favorite browser for this is Microsoft Edge Chromium but any modern browser should work. For a Windows 10 client create a shortcut to the EXE file of Microft Edge Chromium or copy the default shortcut to your startup folder.

Change the target with this line:

```
"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --kiosk "https://psroomsigning.bloglab.nl/rs01" --edge-kiosk-type=fullscreen
```

The kiosk parameters will autostart the browser fullscreen and lock the user interface of the browser. It can only be stopped with ALT-F4 😊



Example of the properties of the shortcut.

End of document.