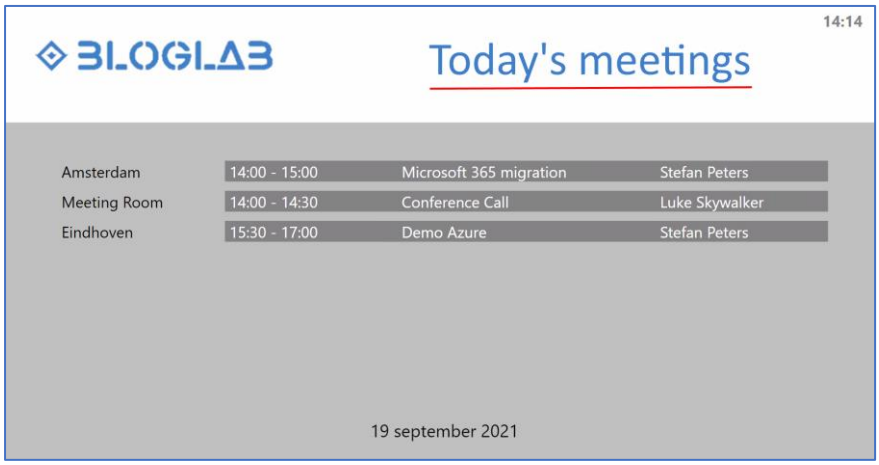
PS RoomSigning

2022.02.05

Contents

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.



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

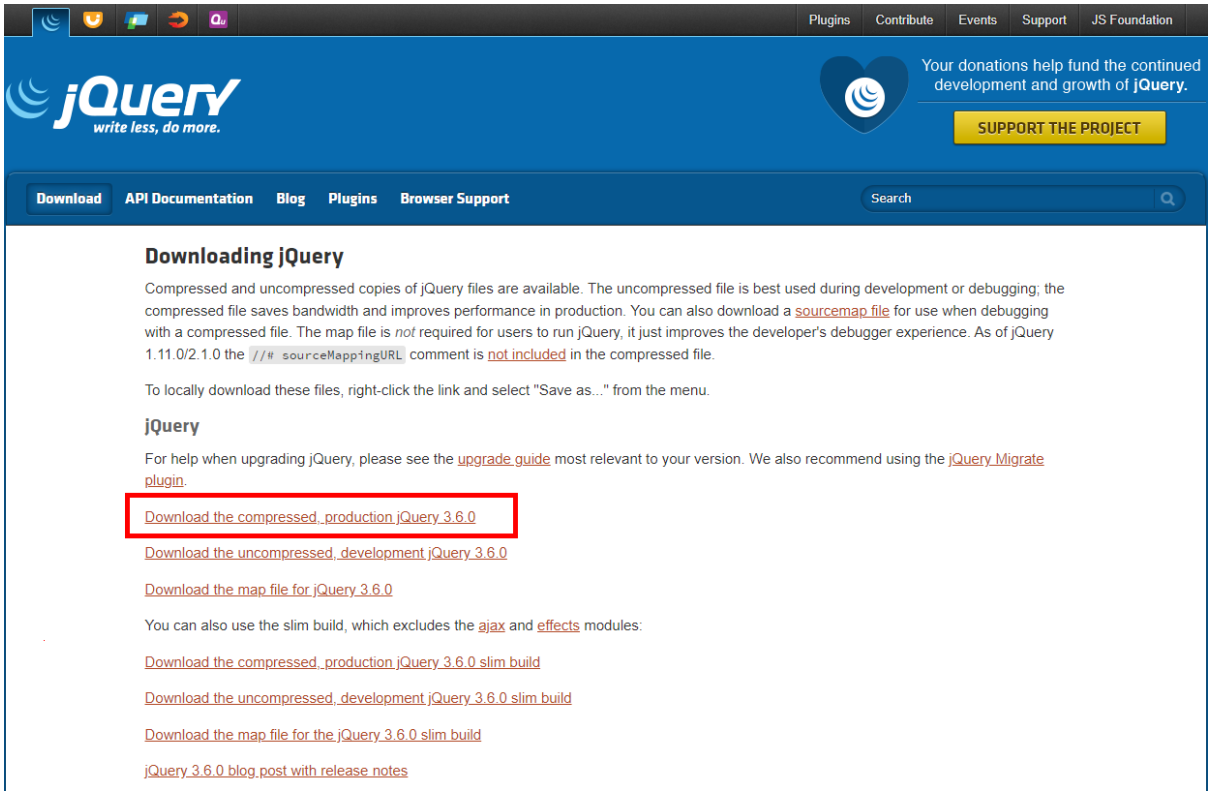
After you downloaded and extracted the PSRoomSigning2022.zip file you need to prepare the files.

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/



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:

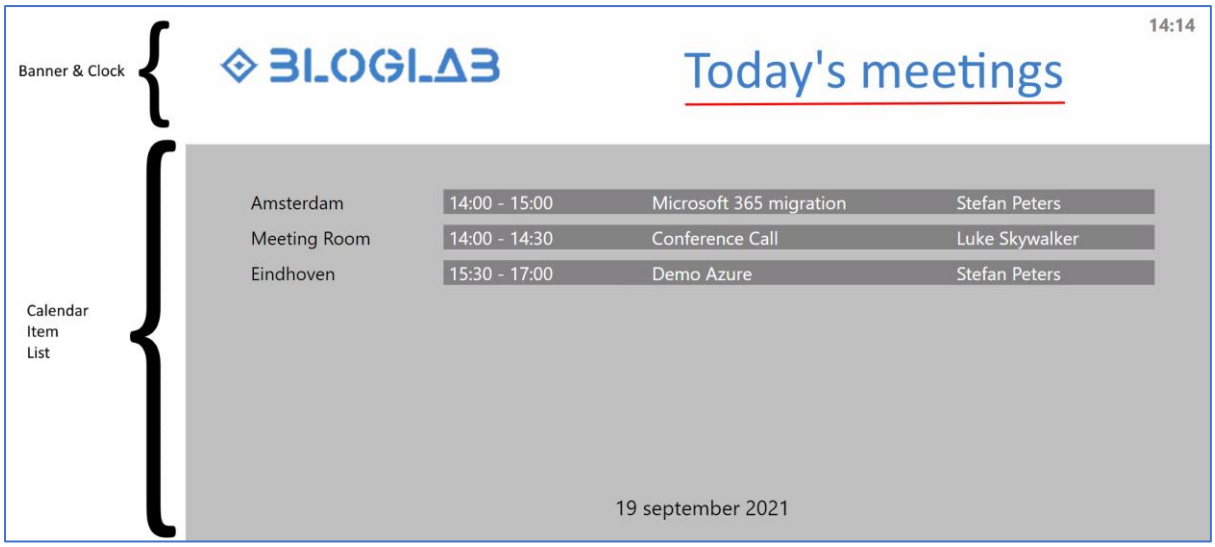
Line Nr	Value	Purpose
5	psroomsigning_bg.png	Background picture, save you picture with this name, or change the name in this htm file.
7	jquery-3.6.0.js	jQuery file, align name with downloaded file.
16-27	// comment	Uncomment the preferred clock location, align with lines 50,51 and 52.
35, 41	roomsigning.html	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 precoded clocks, right, center and left, with or without seconds. Only uncomment one!
    //
    document.getElementById('clock').innerHTML = h + ':' + m + '  ';              // Clock align right, no seconds.
    // document.getElementById('clock').innerHTML = h + ':' + m + '.' + s + '  ';  // 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 = '  ' + h + ':' + m ;          // Clock align left, no seconds.
    // document.getElementById('clock').innerHTML = '  ' + 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:

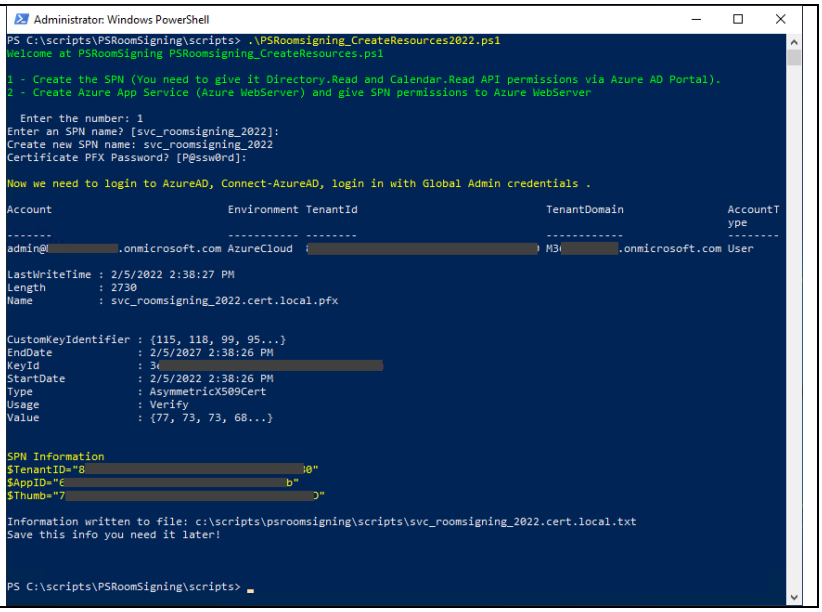
- Install-Module Az
- Install-Module AzureAD

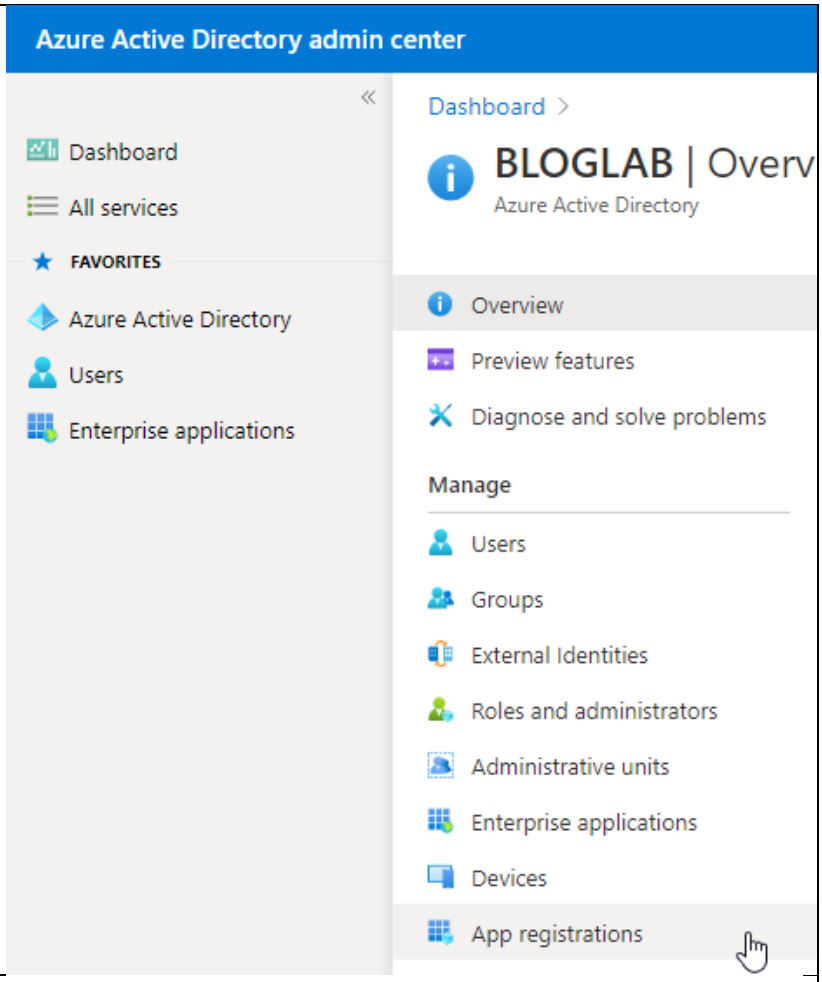
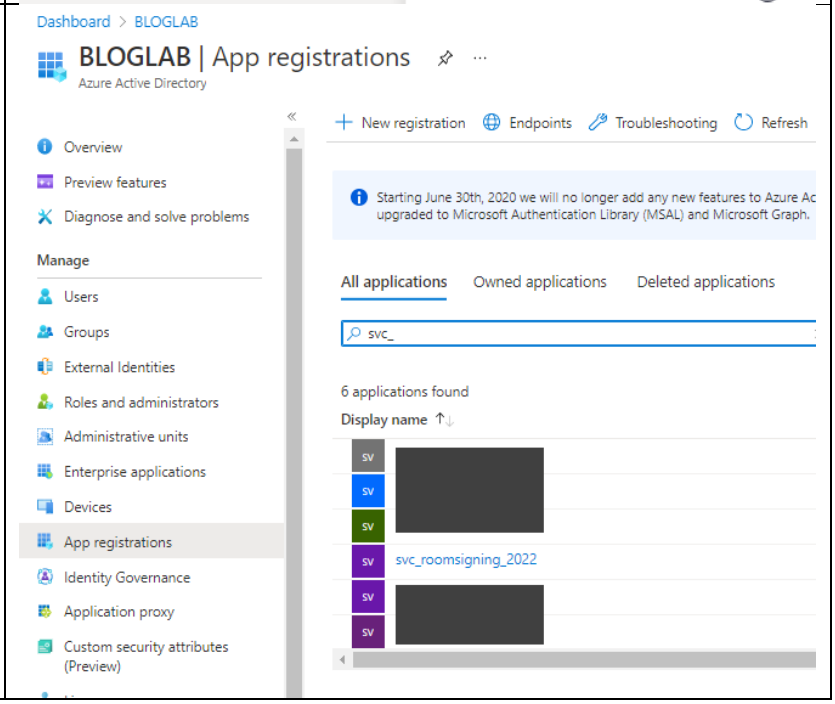
Tip: If you are hosting the script on Windows, create the SPN on the target server.

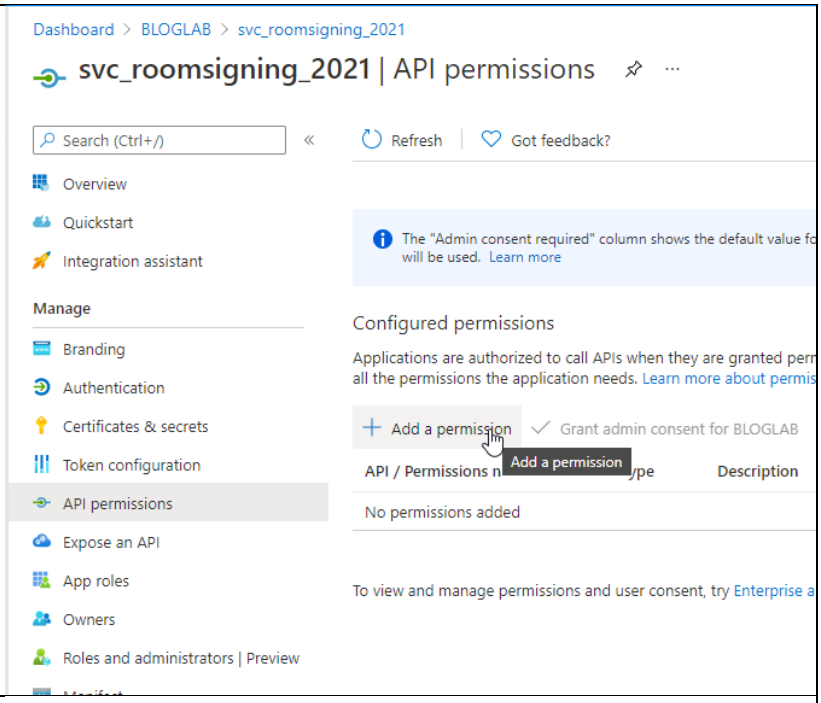
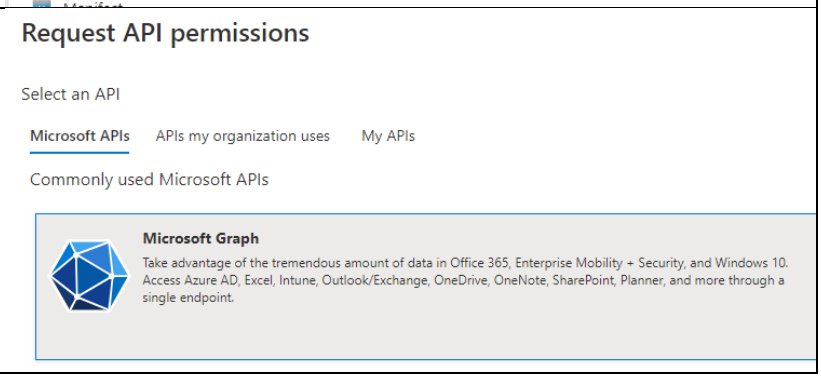
Create a service account that will login to the server and can run the scheduled task on this server.

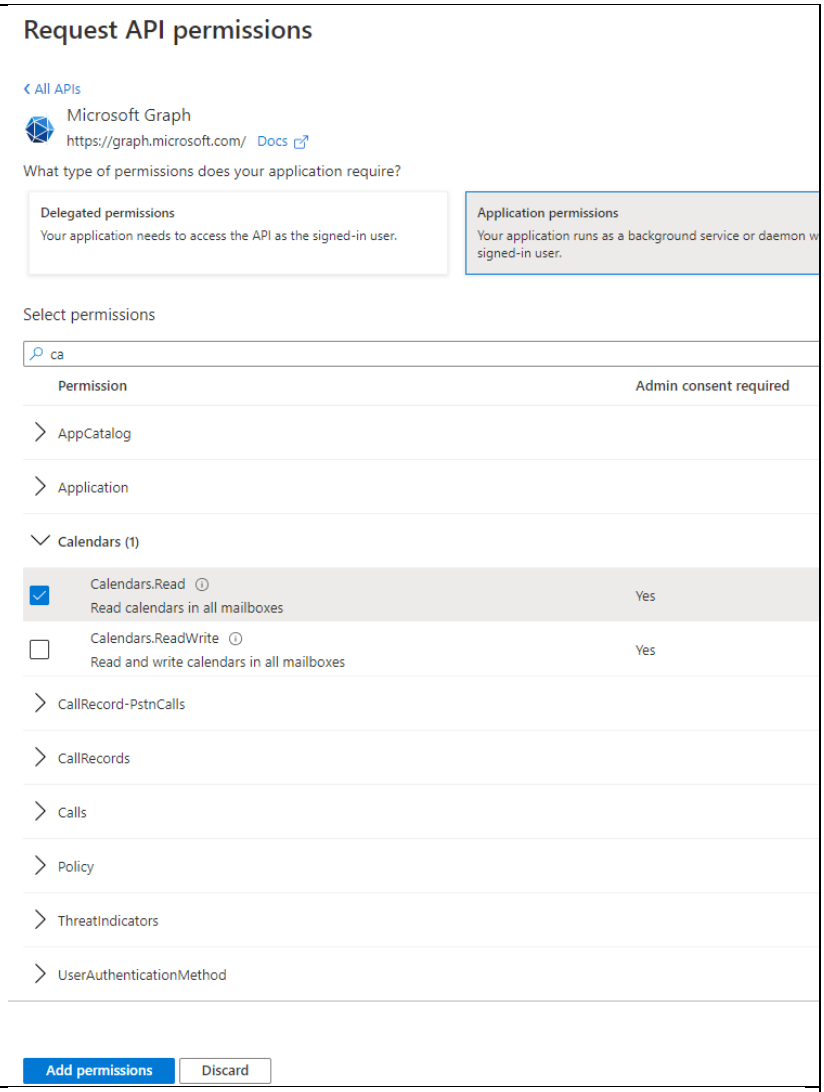
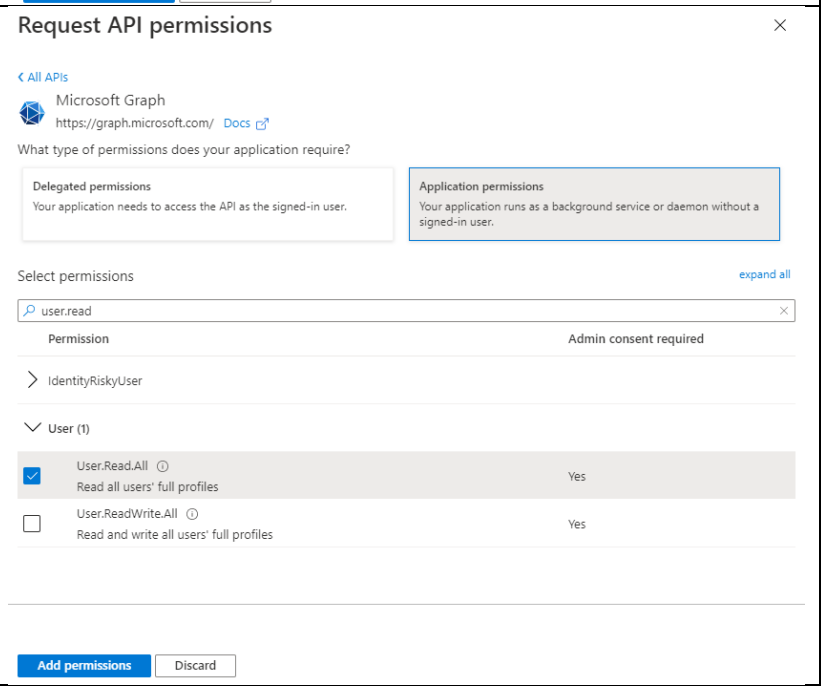
Login to the server with the service account that will run the Scheduled Tasks, the script will save the certificate in the user context. Setup folder structure as above in the example and table. Open an elevated PowerShell and browse to the c:\scripts\PSRoomSigning\scripts folder.

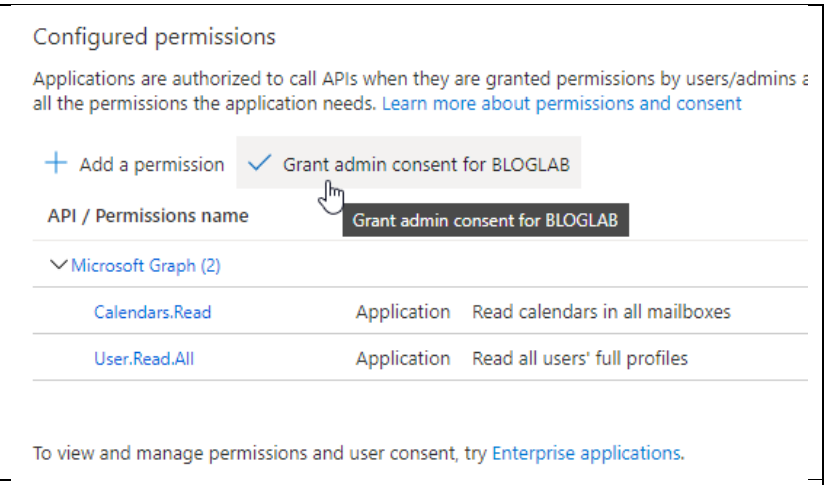
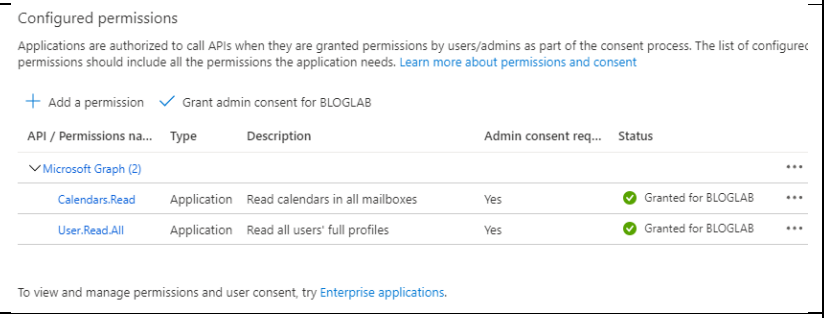
Execute the “PSRoomSigning_CreateResources022.ps1” script and follow setup 1.

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.	

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.	
Click on Microsoft Graph.	

Click Application permissions type in ‘ca’ in the search bar and select Calendar.Read Click Add permissions on the bottom.	
Click Application permissions type in ‘user.read’ in the search bar and select User.Read Click Add permissions on the bottom.	

Click on “Grant Admin consent for.”	
Click Yes.	
Both permissions are granted.	
The SPN is now setup for the PSRoomSigning script.	

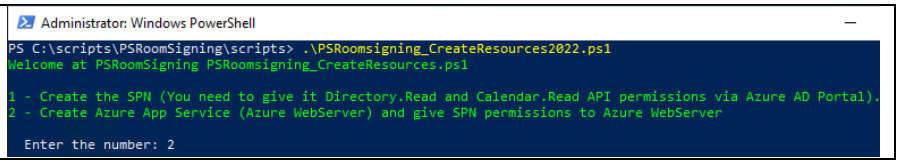
Prepare for hosting the webserver on Azure

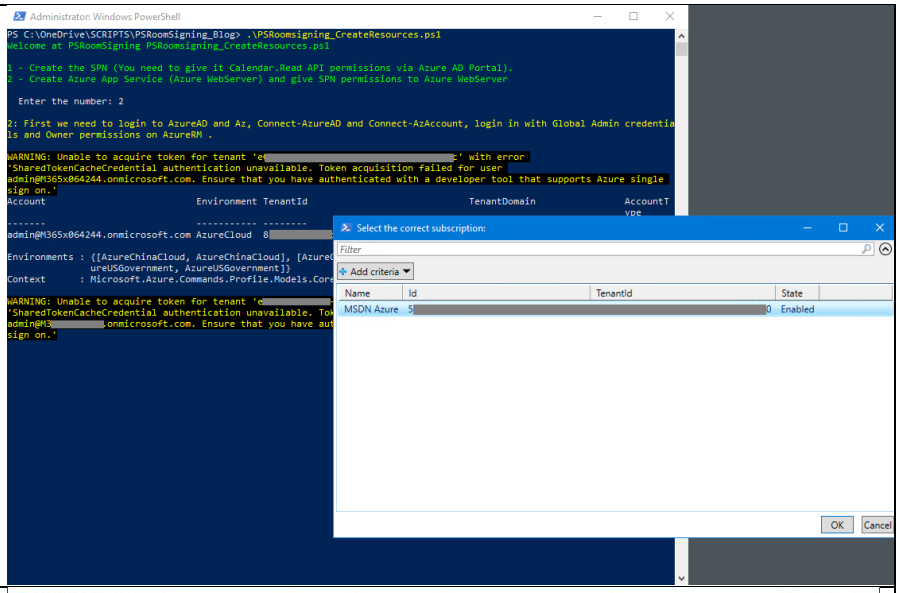
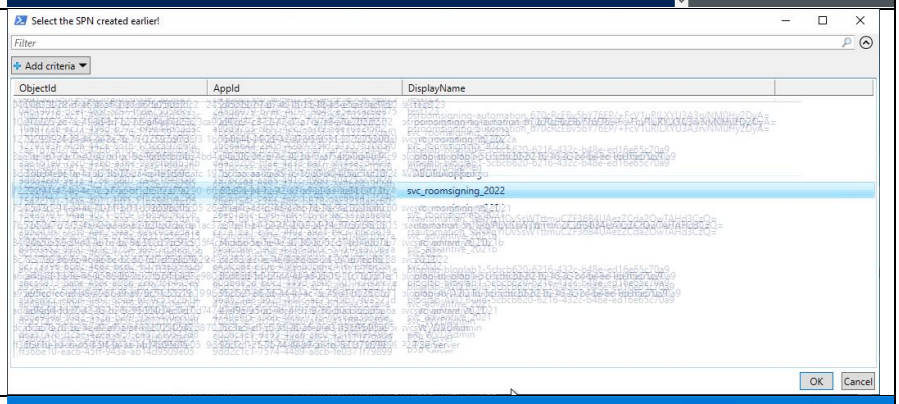
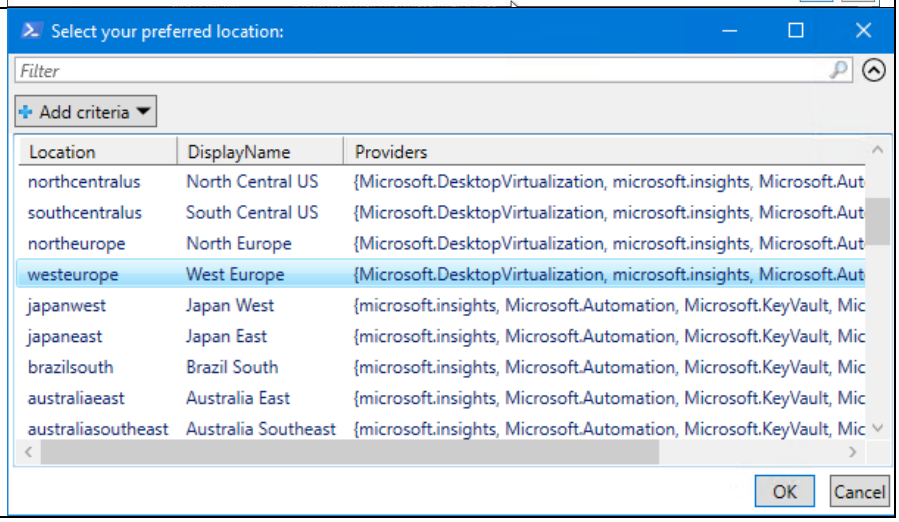
You need a Windows computer to configure these parts, so first create the SPN as described in “Create the Service Principal Name (SPN)” in the Preparations chapter.

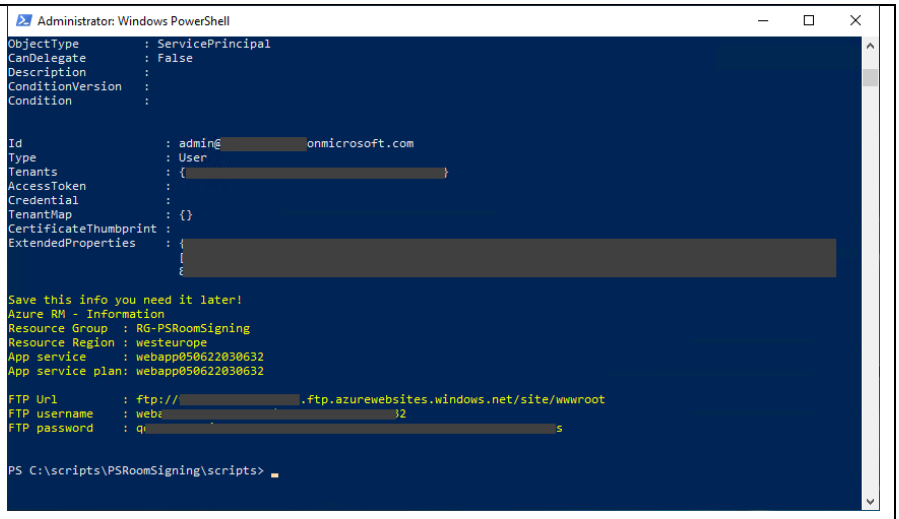
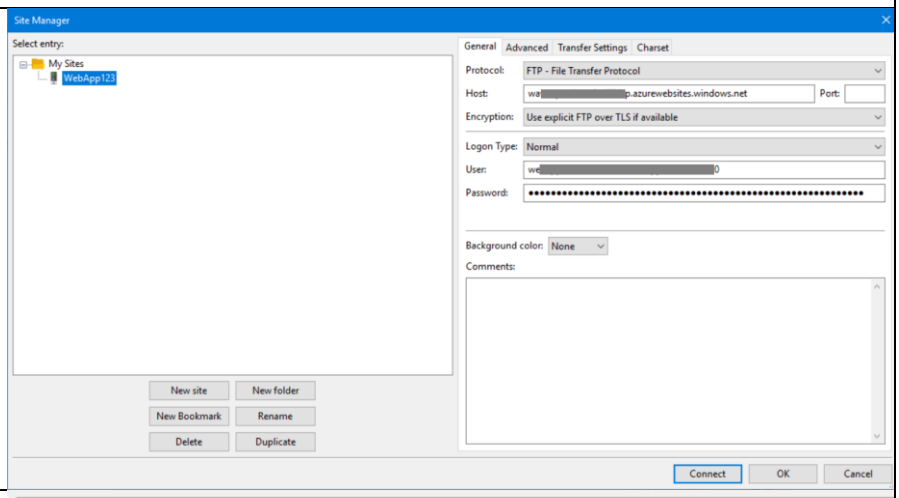
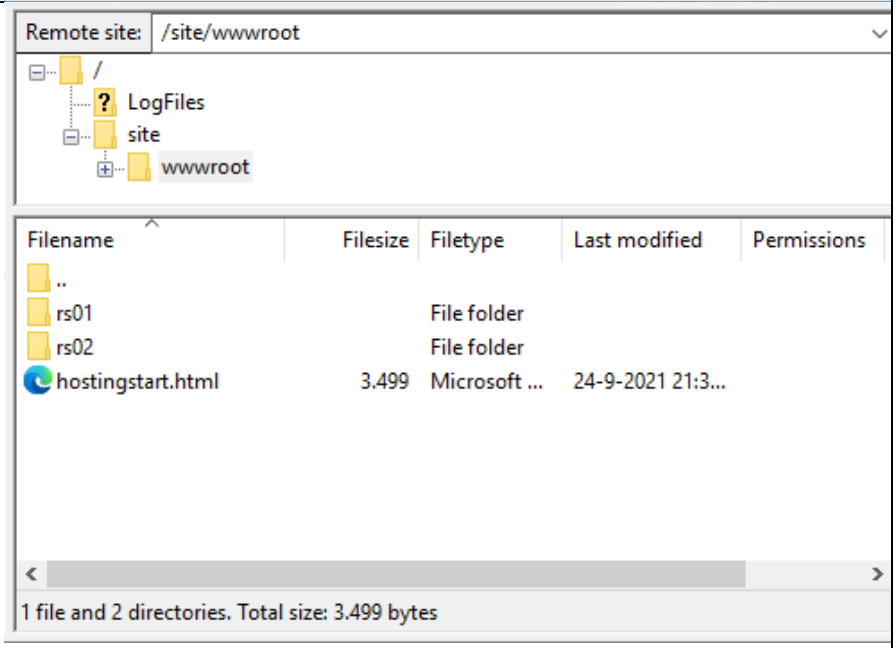
On the Windows computer where you created the SPN run the PSRoomSigning_CreateResources2022.ps1 script again.

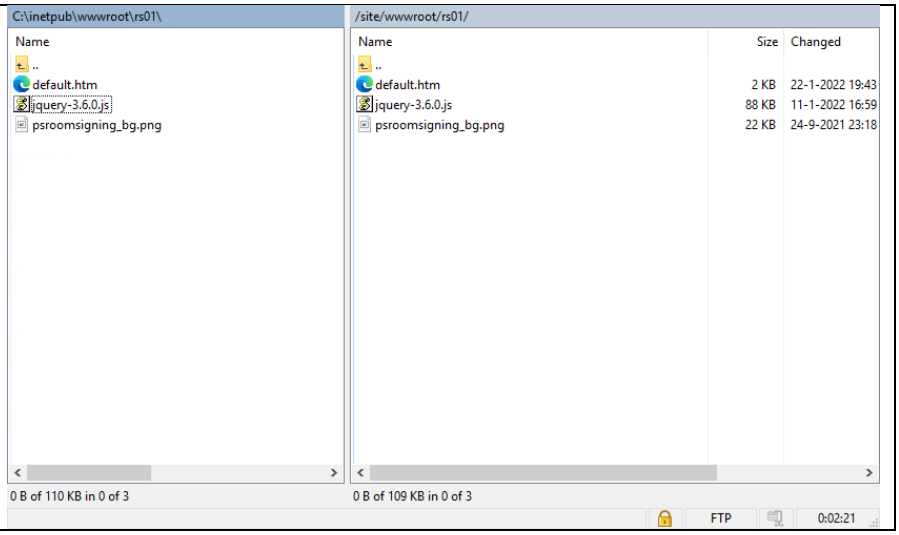
This script will create the webserver components in Azure Resource Manager and give the SPN the required permissions to save files on the webserver.

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

Run the script again. Choose option 2.	

Select the correct Azure Subscription	
Select the SPN created earlier.	
Select the Azure location.	

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.	
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]
`$SaveOnAzureAppService=$false` False – save on local disk for Windows Server True – save on Azure Web Service *Required Setting*

`$AzureAppServiceRelativePath  = "rs01"`
`$AzureAppServiceName          = "webapp2021"`
`$AzureAppServiceResourceGroup = "rg-psroomsigning"`
 	
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”
 	
Required Setting for Azure Web Server target.

`$WindowsBasedLocation = "C:\inetpub\wwwroot\rs01\roomsigning.html"` Path to the local wwwroot folder, including the filename. *Required Setting for Windows Server target*

`$TenantID="XXXXXXXX-YYYY-ZZZZ-XXXX-XXXXYYYYZZZZ"`
`$AppID="11111111-2222-3333-4444-555566667777"`
`$Thumb="1234567890ABCDEF1234567890ABCDEF12345678"`
 	
Replace these variables with the info after running option 1 from this script: “PSRoomsigning_CreateResources2022.ps1”
 	
Required Setting

`$rooms="meetingroom1@domain.com","meetingroom2@domain.com"` Enter one or more meeting room email addresses in this variable. These rooms will be read for meeting items. *Required Setting*

`$Itembackgroundcolor="848284" # Item level background color default: 848284`
`$RoomNameColor="000000"       # Room name color default: 000000`
`$ItemColor="FFFFFF"           # Item color default: FFFFFF`
`$DateStampColor="000000"      # DateStamp color default: 000000`
 	
Variables for configuring colors.

`$maxcalendaritems=9   # Maximum number of items on the screen.`
`$maxsubjectlength=53  # Truncates the subject on maximum chars`
`$skiplines=3          # Add empty lines from top of screen to align with background picture.`
 	
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)
```

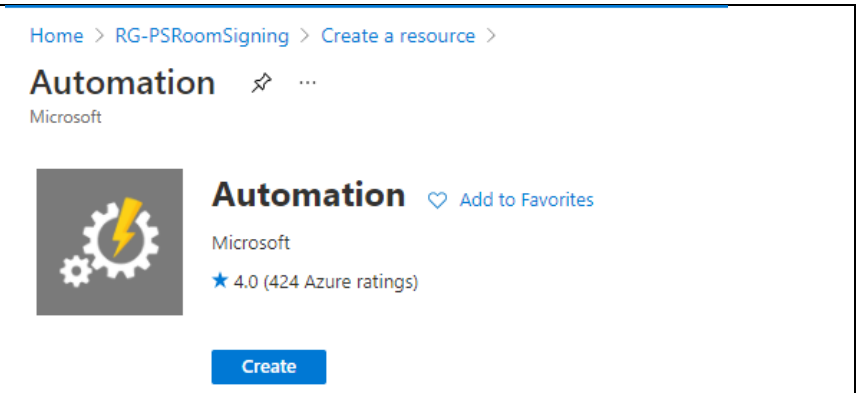
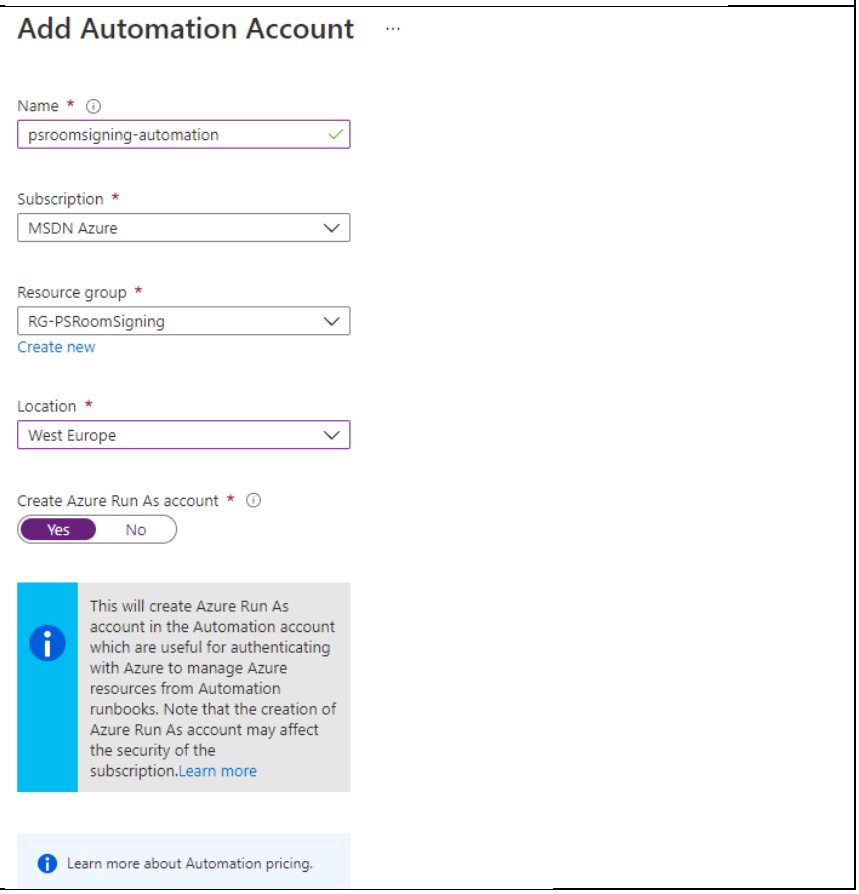
If you are not from The Netherlands you need to reconfigure these values, in the comments the PowerShell commands are listed how to find your values to put in the variables.

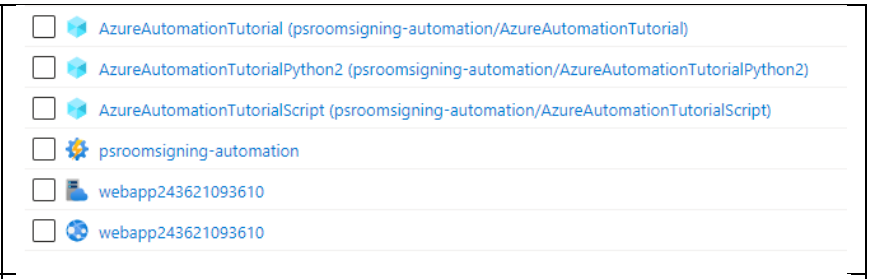
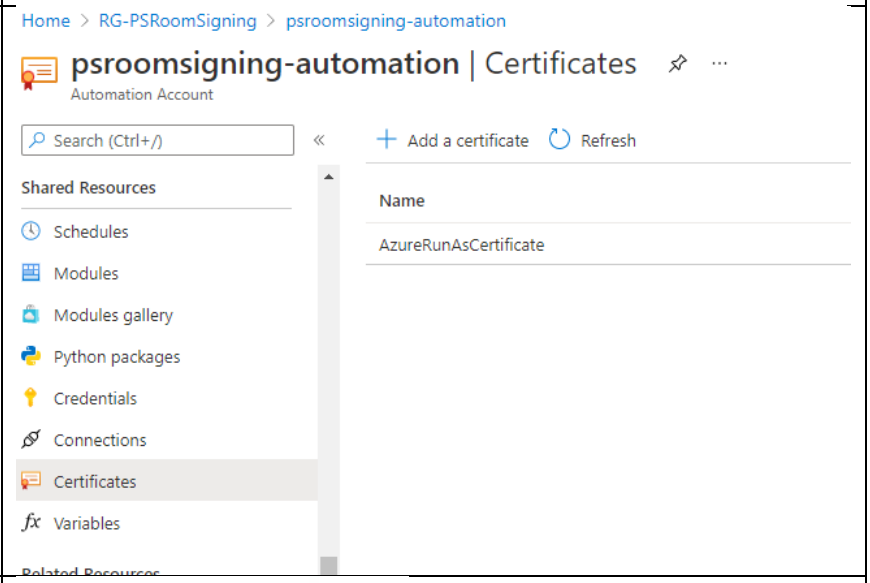
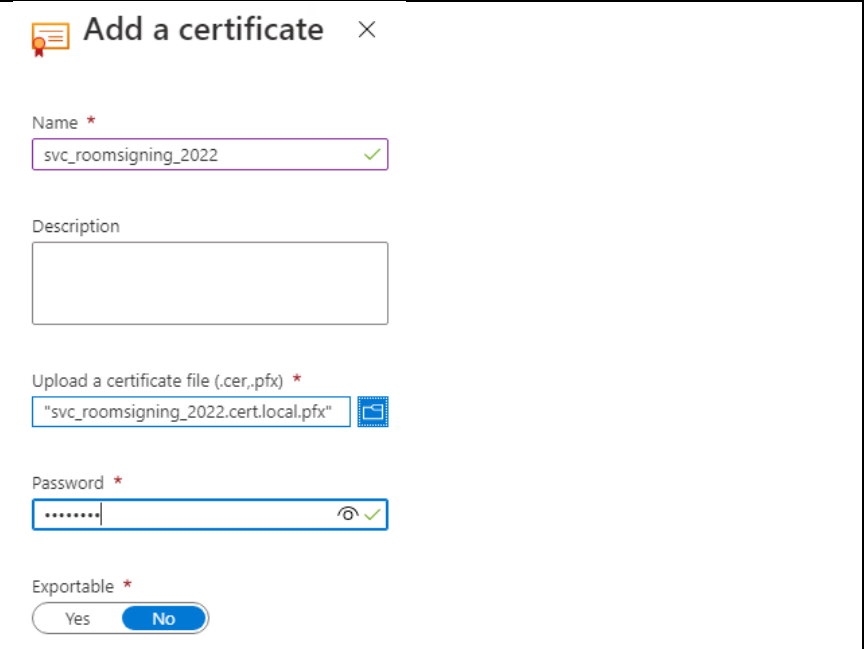
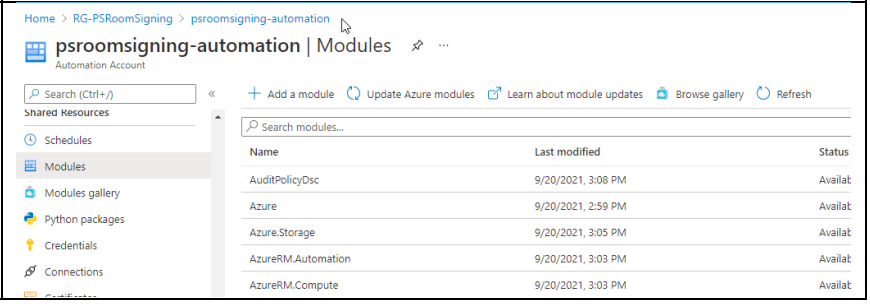
All other PowerShell commands do not need to be changed and should work as is.

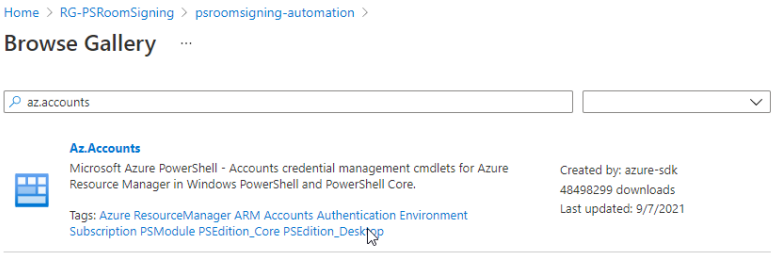
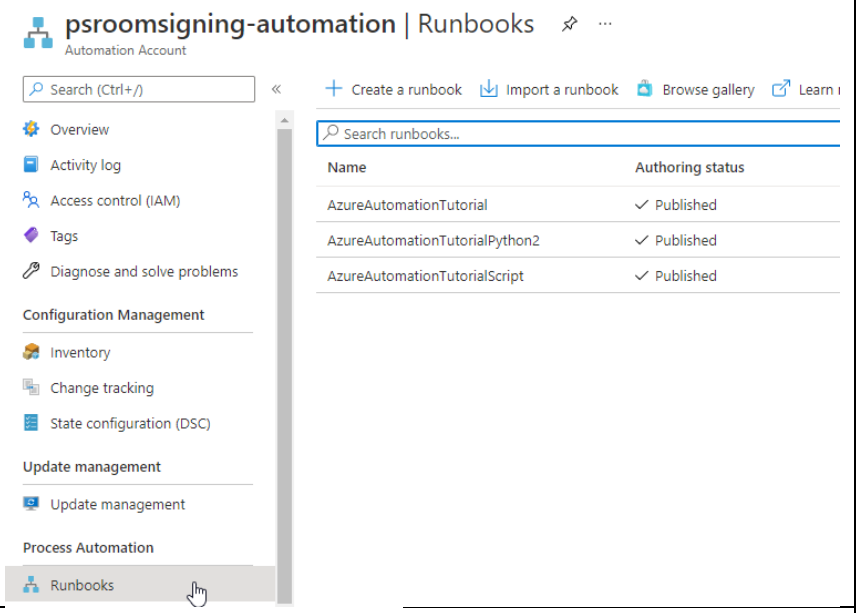
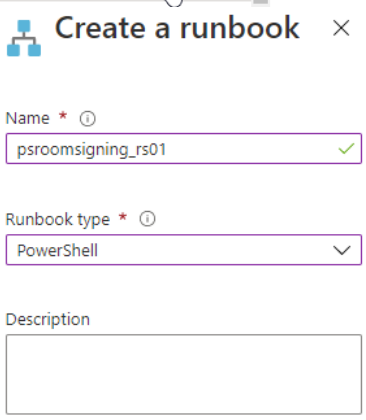
Installation

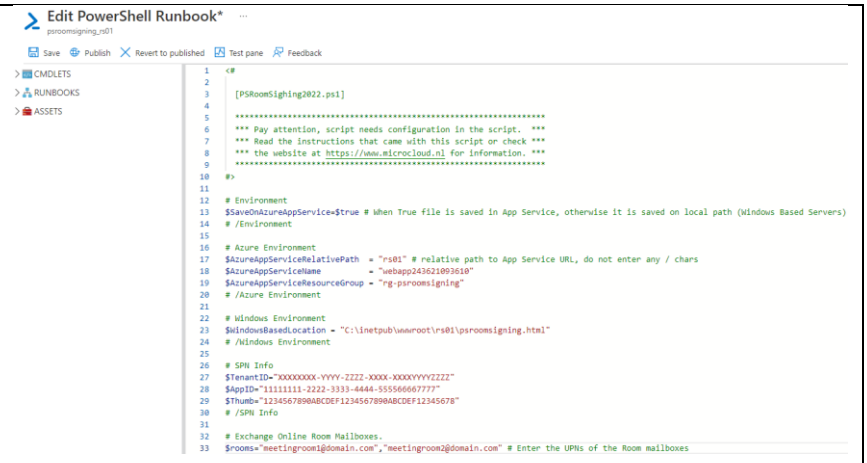
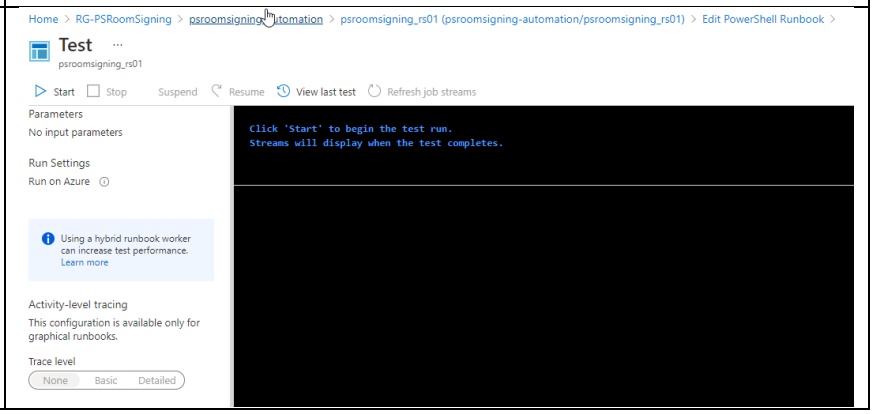
Install on Azure RM

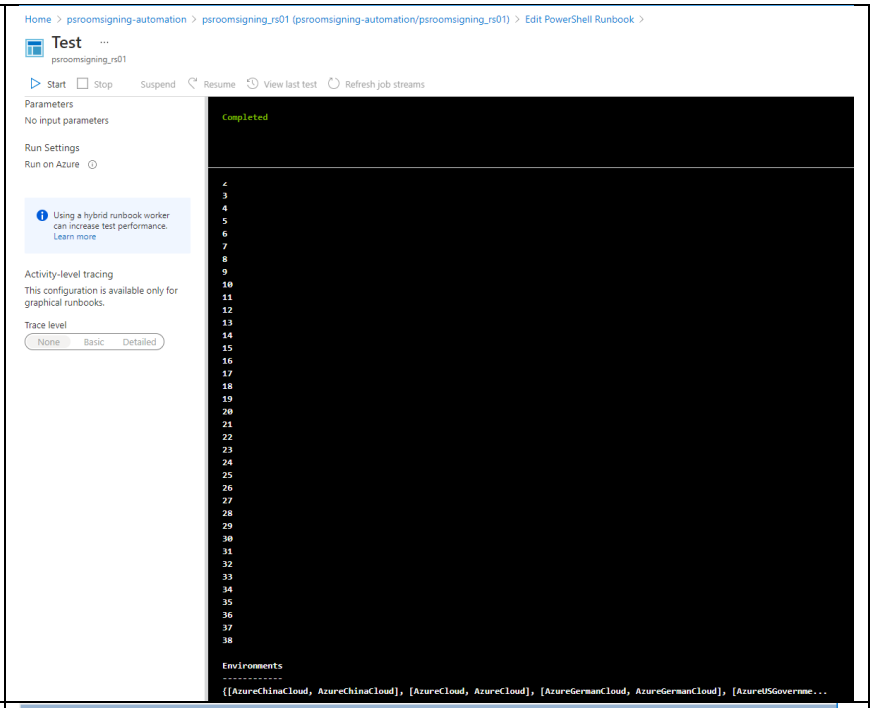
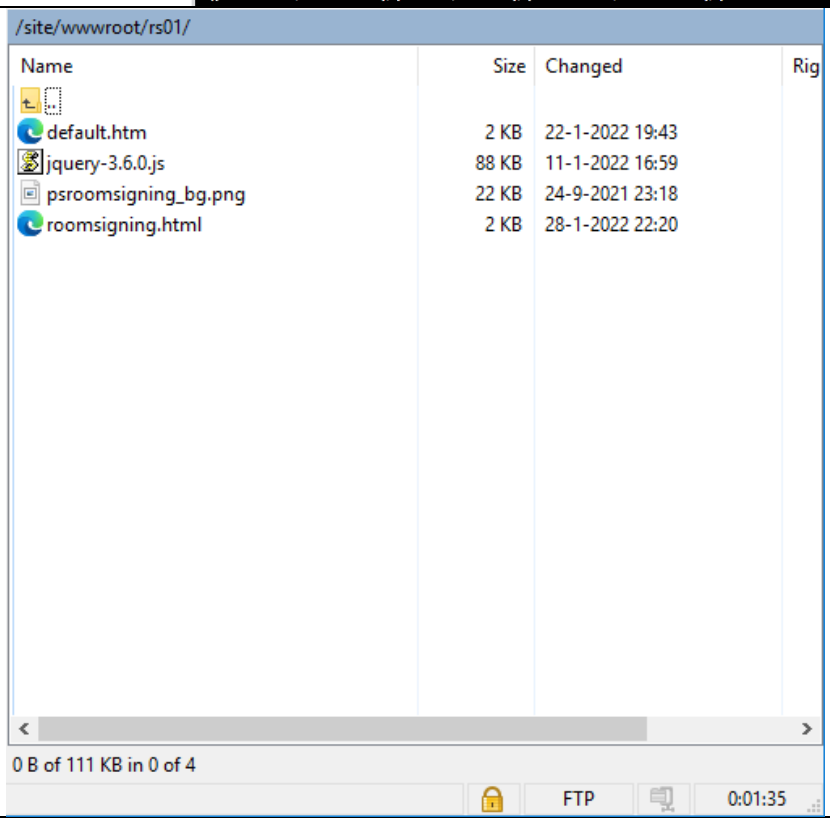
In this section we will create and configure Azure Automation PowerShell RunBooks. Login to Azure Portal with enough permissions to create new resources. I will create the new resources in the Resource Group created earlier with the script. RG-PSRoomSigning.

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

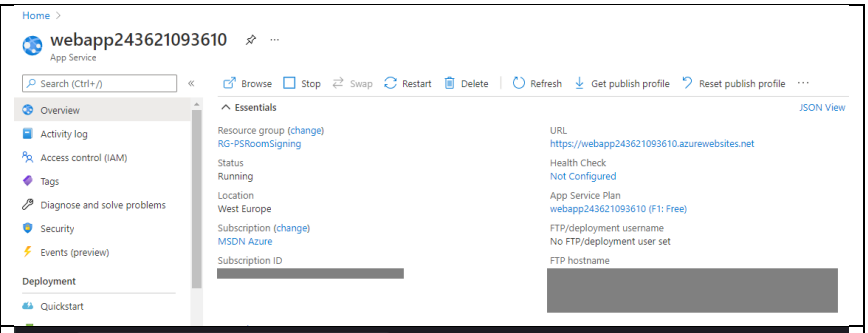
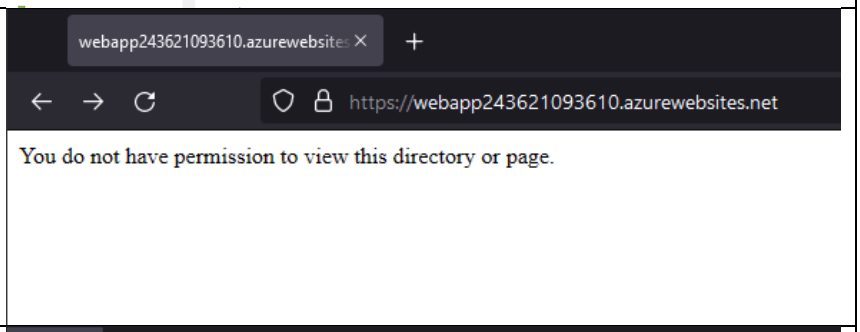
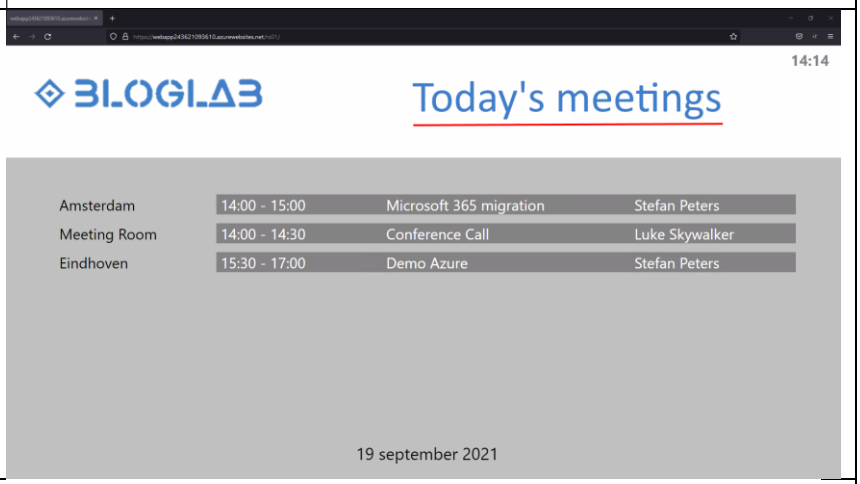
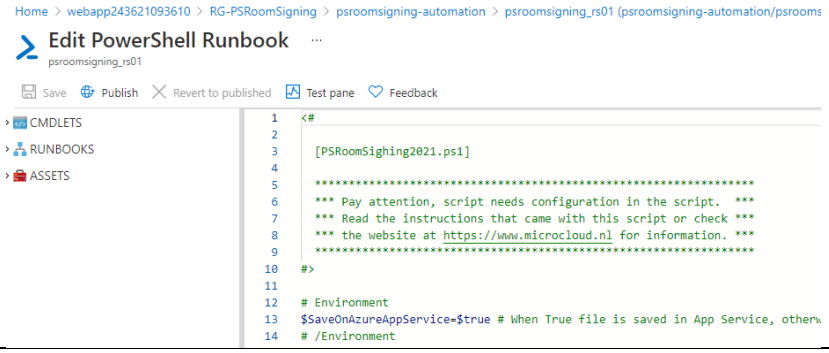
Your resource group contents will look something like this.	
Click on psroomsigning-automation. On the Shared Resources section click Certificates. (You need to scroll down for this) Click on Add a certificate.	
Browse to the location where you ran the PSRoomSignin_CreateResources.ps1 script, in that folder you should find a PFX file that contains the authentication certificate for the SPN. Enter the password of the PFX and select NO for exportable. Note keep this certificate in a safe place. On the bottom click create.	
On the Shared Resources section click Modules. Click Browse gallery.	

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.	 If you import them in this order, you should not have trouble with dependencies. If a dependency is already importing but still askes for it, the import is not finished yet.
Scroll back up and click on Runbooks. Click on Create a runbook	
Enter a name and select the runbook type: PowerShell On the bottom click Create. If asked choose runtime version 5.1.	

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.	

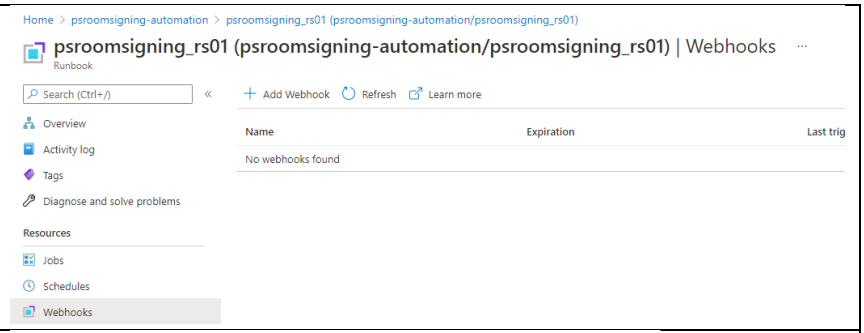
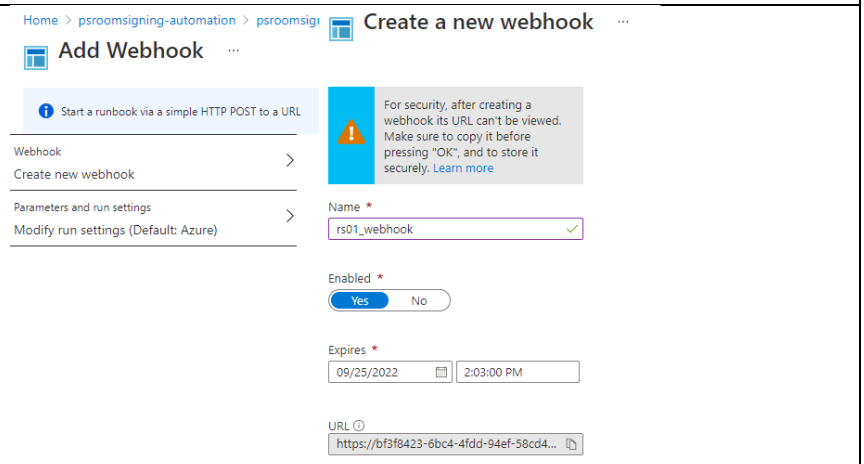
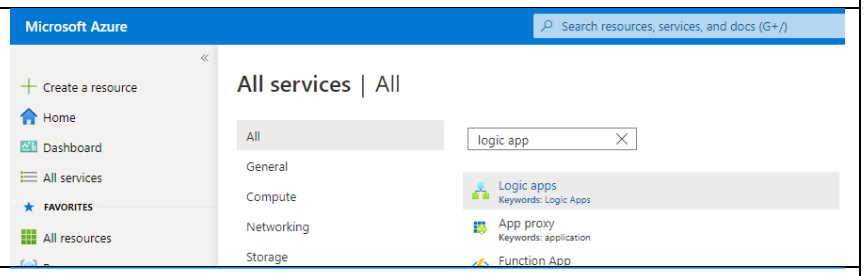
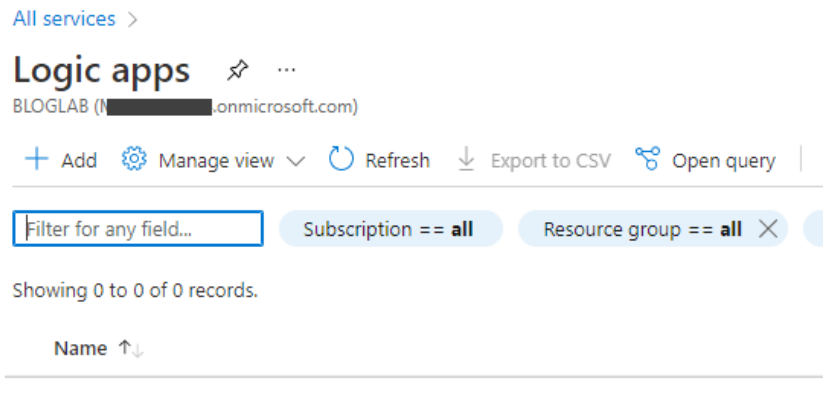
The result should look something like this. *If you get errors, check the SPN info in the script, check the certificate and de API permissions of the SPN. Also a wrong resource groupname, or webapp name can cause errors.* *Also check if all required PowerShell modules are imported.*	
After the test you should see the roomsigning.html in the wwwroot/rs01 folder.	

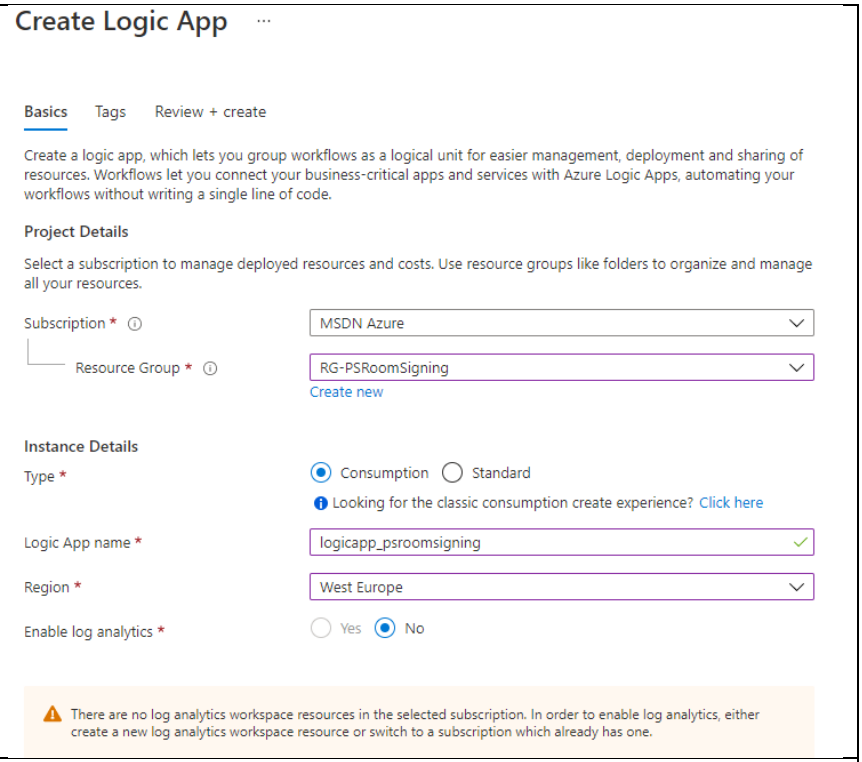
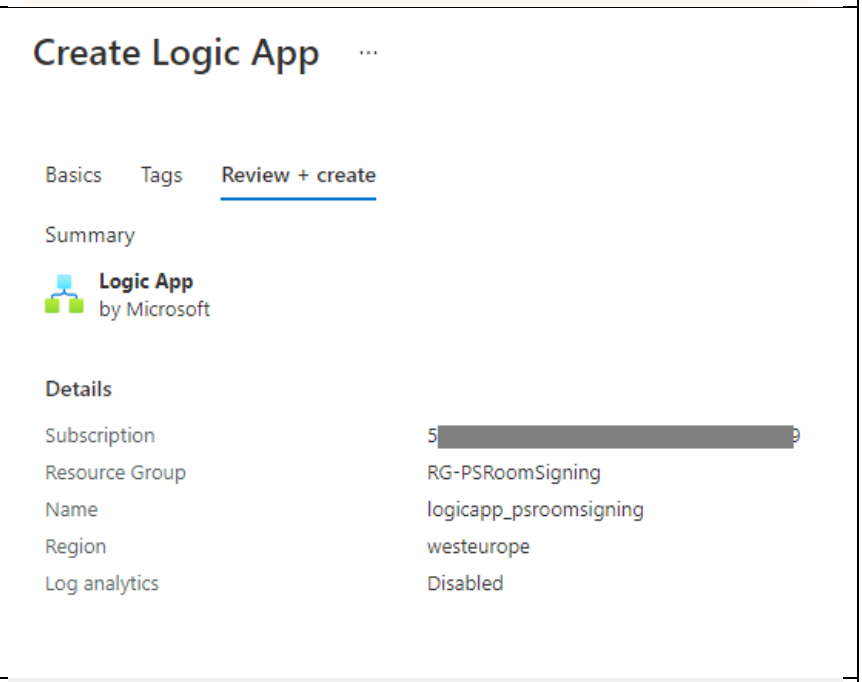
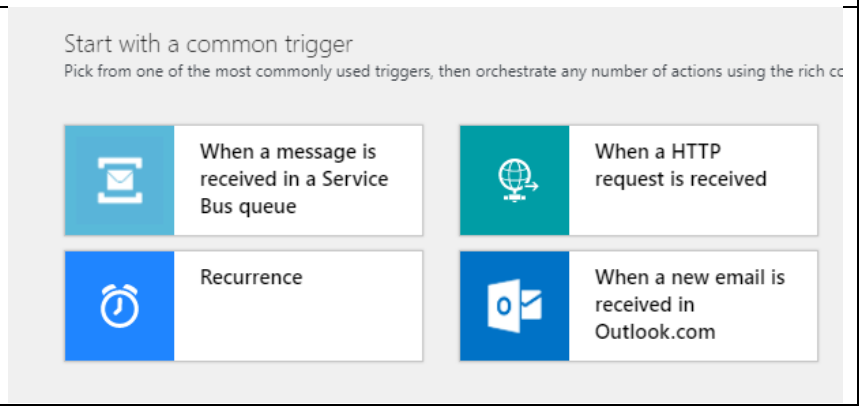
Test Azure App Service

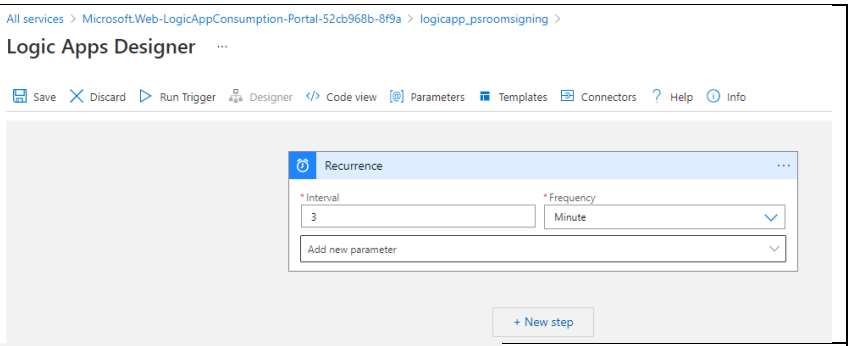
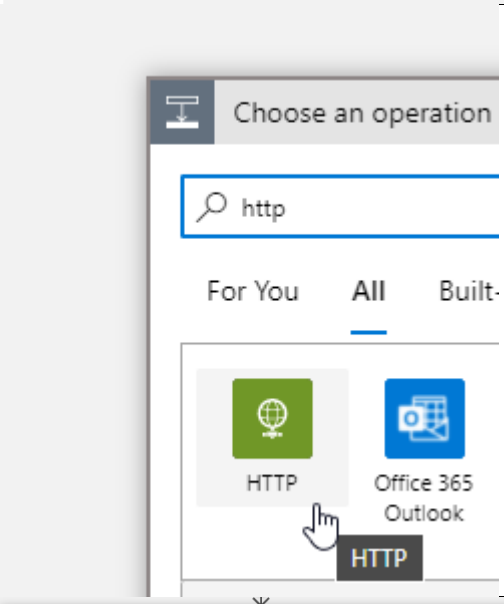
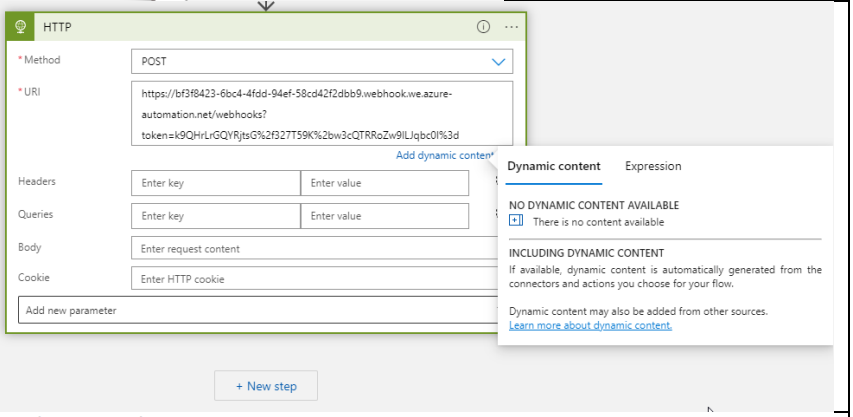
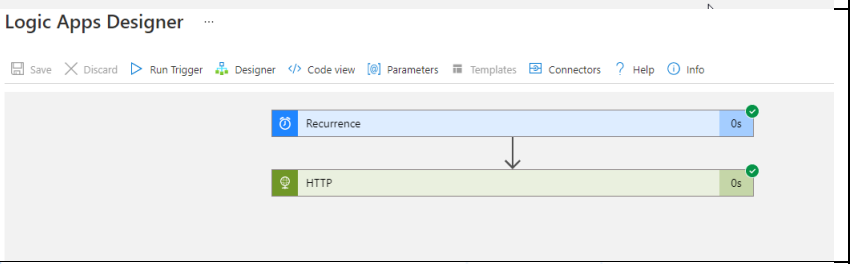
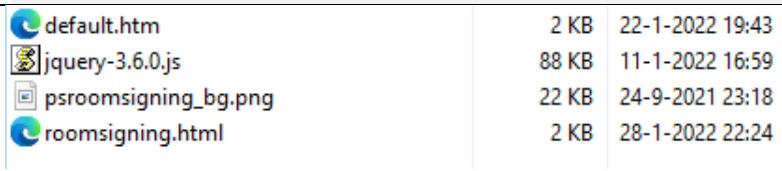
In Azure portal, browse to you App Service object. You can copy out the URL from this page or click Browse to open the website in your browser.	
If you have deleted the hostingstart.html file you will get this error. Add the /rs01 to the url and check again!	
When you add the /rs01 the page should be 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.	

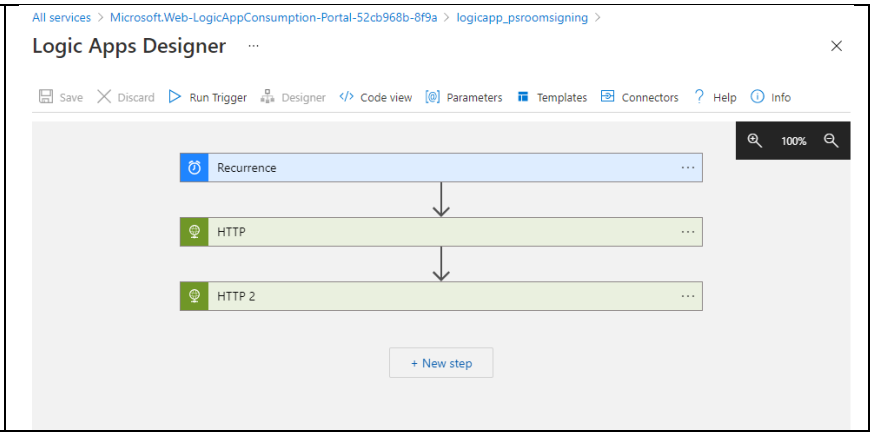
Azure Schedular

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.

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

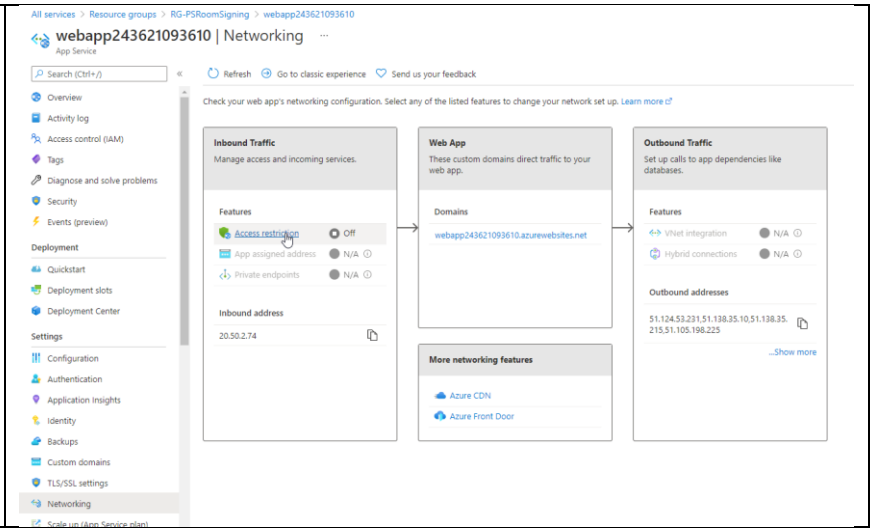
Select the resource group of your RoomSigning components. RG-PSRoomSigning Type: Consumption Name the logic app and select your region. Next.	
Click Review + Create on the bottom of the page.	
When object is created click on Go to resource. Find the trigger “Recurrence” Click on it.	

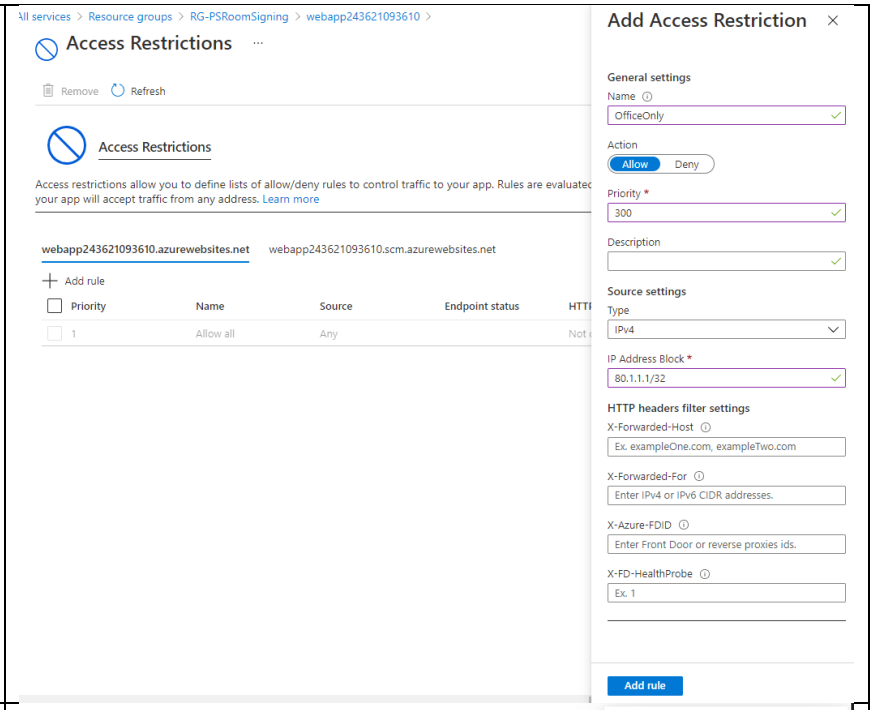
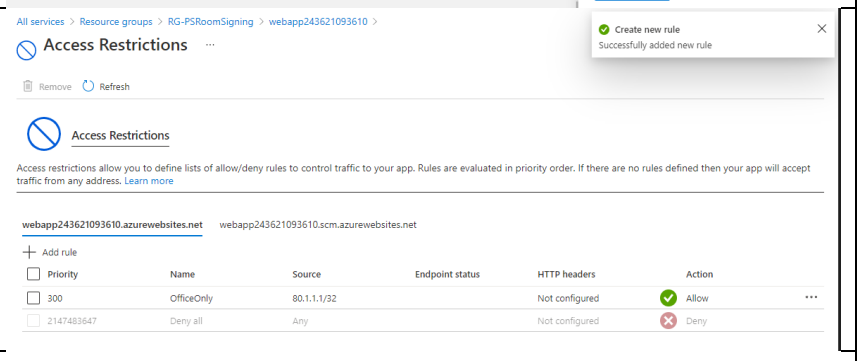
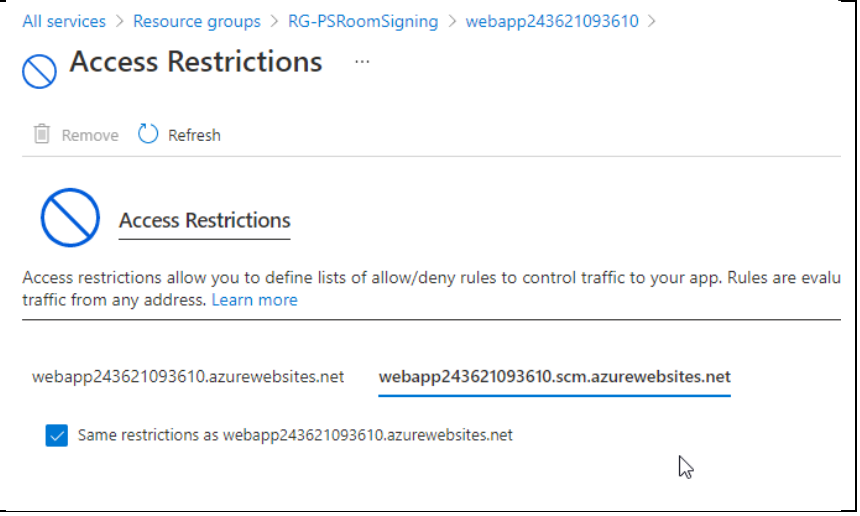
Default interval is 3 minutes. Change to your requirements. Click on New step.	
Search for HTTP and select the default green HTTP button.	
Select method: POST Paste the Webhook URL in the URI part. Click on Save.	
To test if it works click on Run Trigger.	
Check the FTP folder if the timestamp of roomsigning.html has changed. If it did the trigger works!	

If you host multple 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.	
After creating this rule an implicit Deny is added automatically.	
Click on the web243.scm.*** section and enable “same restrictions as web***”	

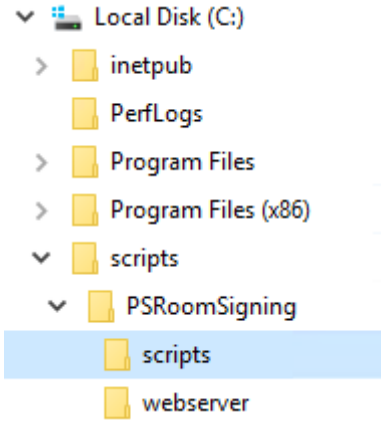
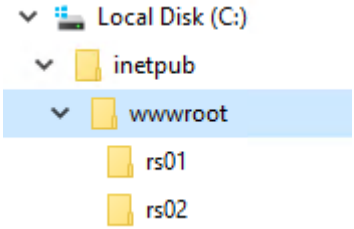
Install on Windows Server

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:

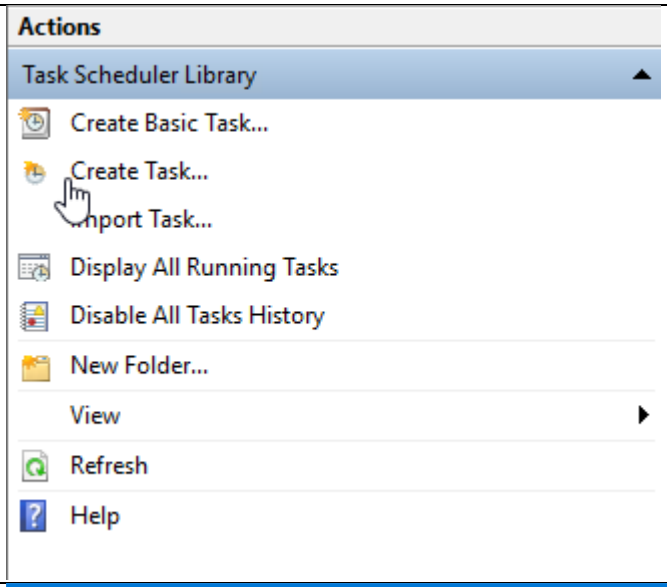
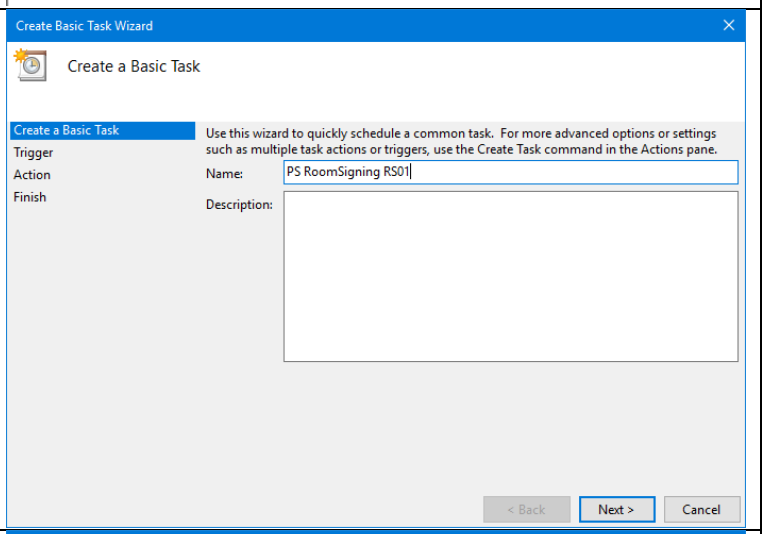
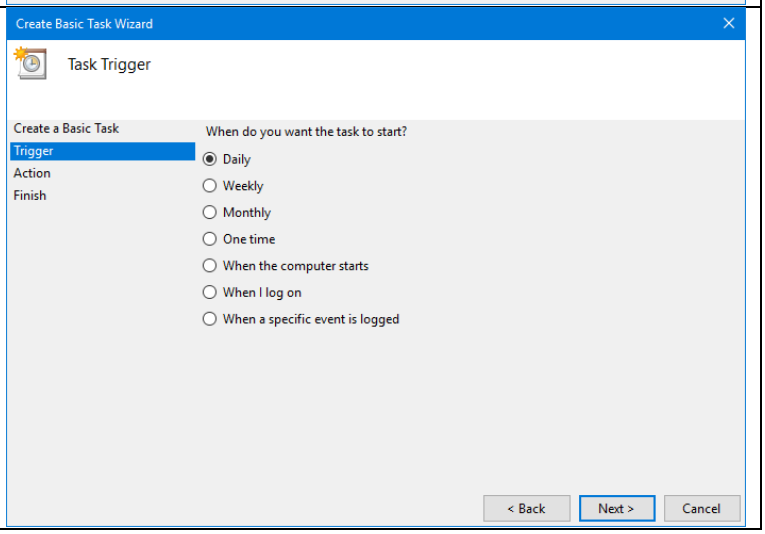
PowerShell Script	**WebServer**
	

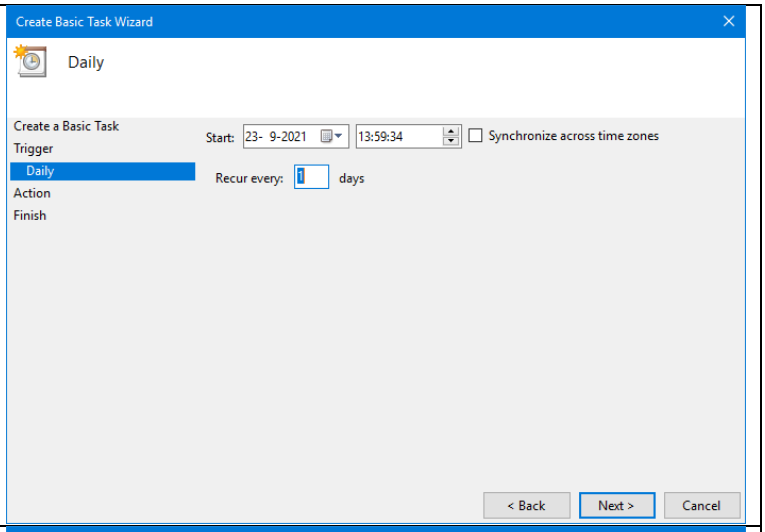
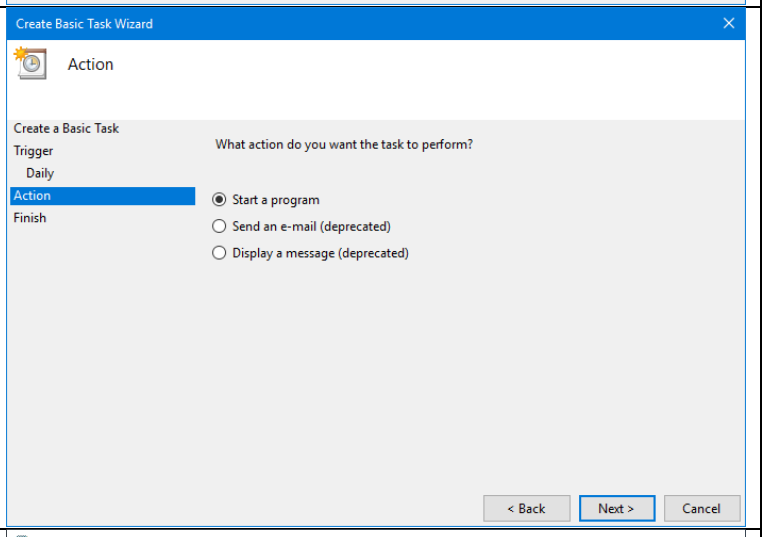
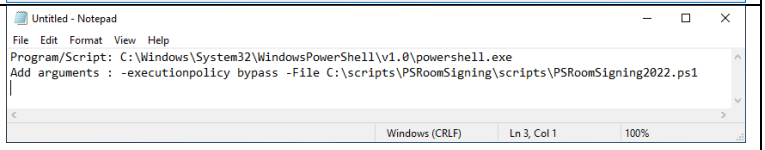
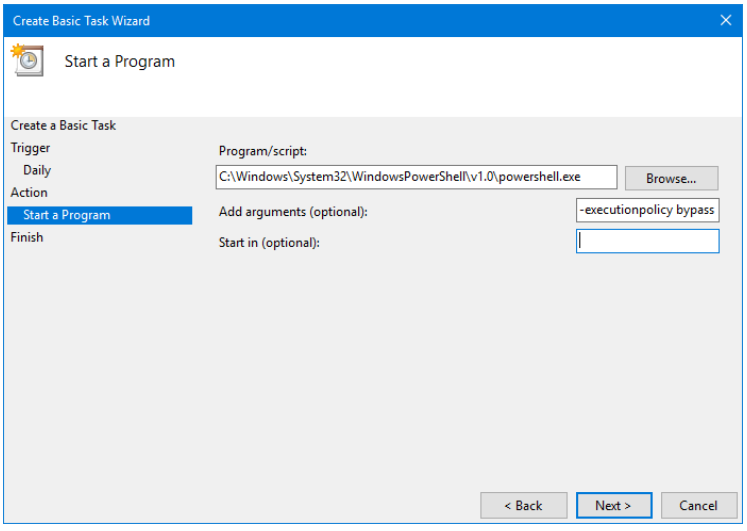
C:\scripts\PSRoomSigning		
	\scripts	
		PSRoomSigning2022.ps1
		PSRoomsigning_CreateResources.ps1

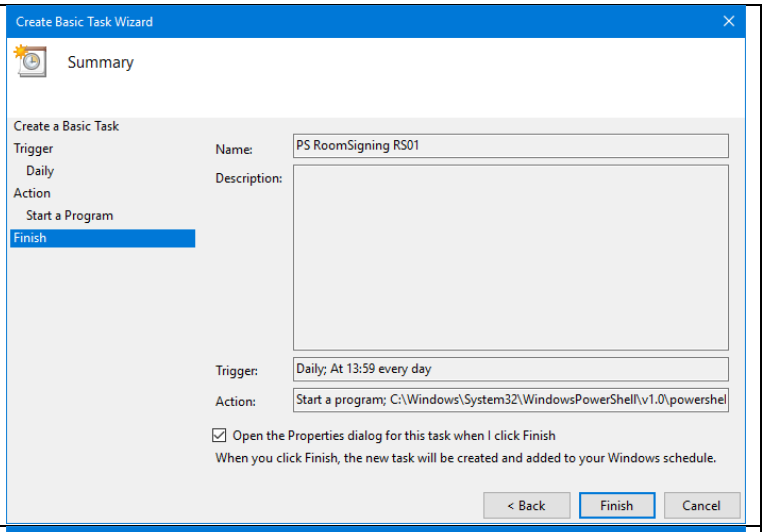
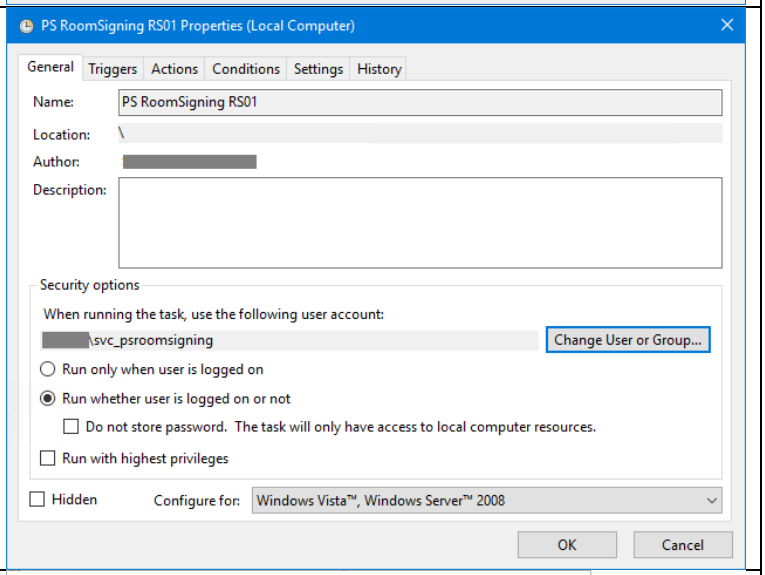
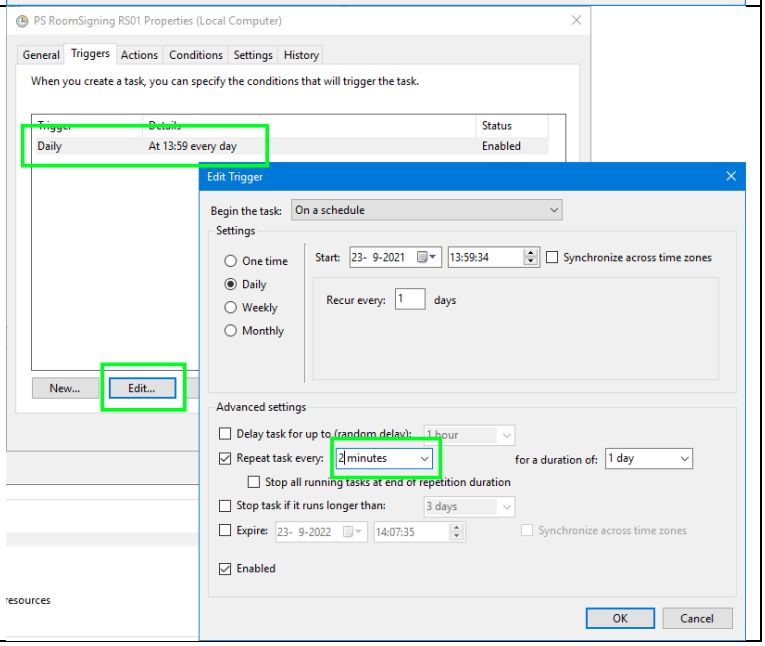
C:\inetpub\wwwroot		
	\rs01	
		default.htm
		psroomsigning_bg.png
		jquery-3.6.0.js
		roomsigning.html
Optional	**\rs02**	
		default.htm
		psroomsigning_bg.png
		jquery-3.6.0.js
		roomsigning.html

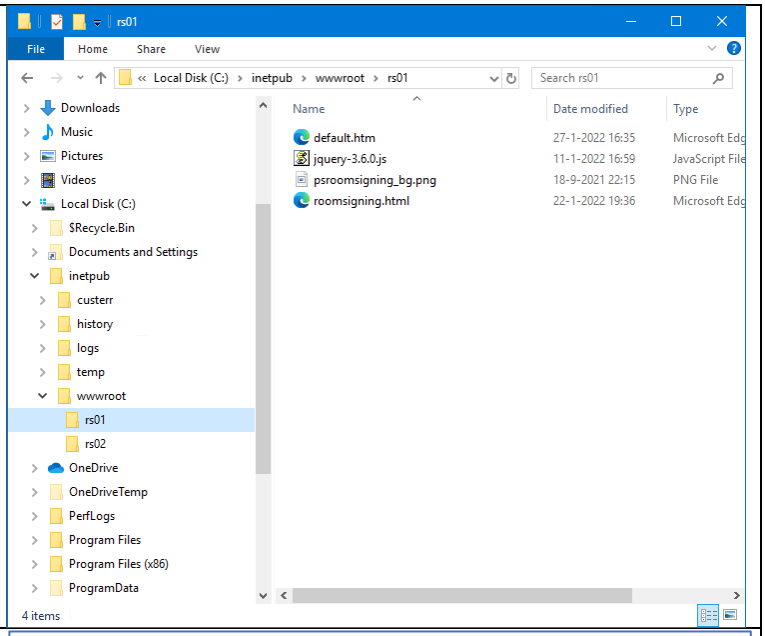
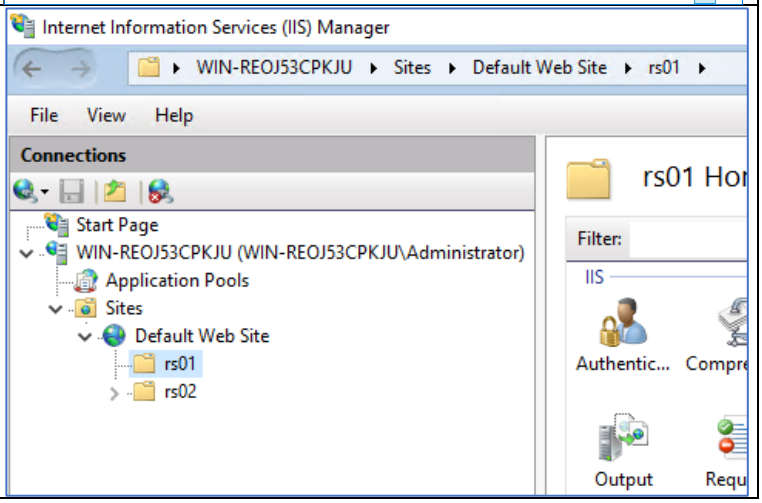
Copy the files you prepared earlier into the c:\inetpub\wwwroot\rs01 folder as shown in the above table. Other folder structures are also possible, this is an example. The *roomsigning.html* is not in the ZIP and will be created after running the PSRoomSigning2022.ps1 script for the first time.

Configure the scheduled task

Open Task Scheduler. Click “Create Basic Task…”	
Enter a name for the scheduled task. Next	
Choose Daily, Next	

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\WindowsPowerShell\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.	 

Enable “Open the Properties …” Finish.	
If you want to change the service account, click “Change User or Group.” Change the radio buttons to: “Run whether user is logged on or not” Highest privileges should not be necessary.	
Click on the Triggers TAB, select the Daily, click Edit… and enable “Repeat task every:” Change the selection to 5 minutes and if you want you can change it even lower to 2 minutes. 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.

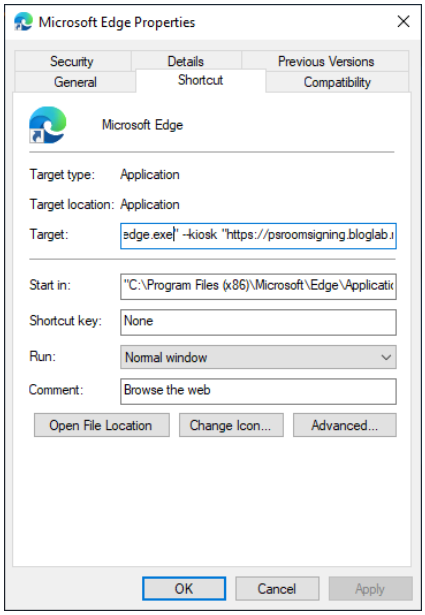
Windows 10 Client configuration

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.