# Hiding a specific MFA method per relying party

## Important

This is provided as a sample script. Please thoroughly review the documentation, review source code and edit it to your specifications, and validate using different user browser scenarios. Maintenance and support for this customization in the production environment is your responsibility.

This was validated at a high level using Windows Server 2016 AD FS.

Background

AD FS does not natively provide for specifying second factor authentication (MFA) based on the relying party. In the migration scenarios, it may be desired to allow certain MFA providers for one relying party and other MFA providers for another relying party. This approach uses AD FS capability to customize the AD FS web theme to include an onload.js script that detects the relying party being accessed and hides undesired MFA methods for that relying party. Additionally, this script allows for prototyping/testing of hiding MFA method for a pre-defined list of users.

**IMPORTANT: user IDs are delivered in the script to the end user browser. If exposing usernames is a concern (and it is), this particular option/method should never be used in the production environment, and RP only restrictions should be used.**

**Note:** user ID array should contain on-premises UPNs of Azure AD users.

In this example, there are two MFA methods configured in AD FS:

* Azure MFA
* Certificate Authentication

The purpose of this sample is to hide Azure MFA from being visible when the user accesses a specific relying party and/or the user UPN is in the pre-defined list. In this case, the relying party has a resource URI of **urn:alexpav:adfs-app**

## Files

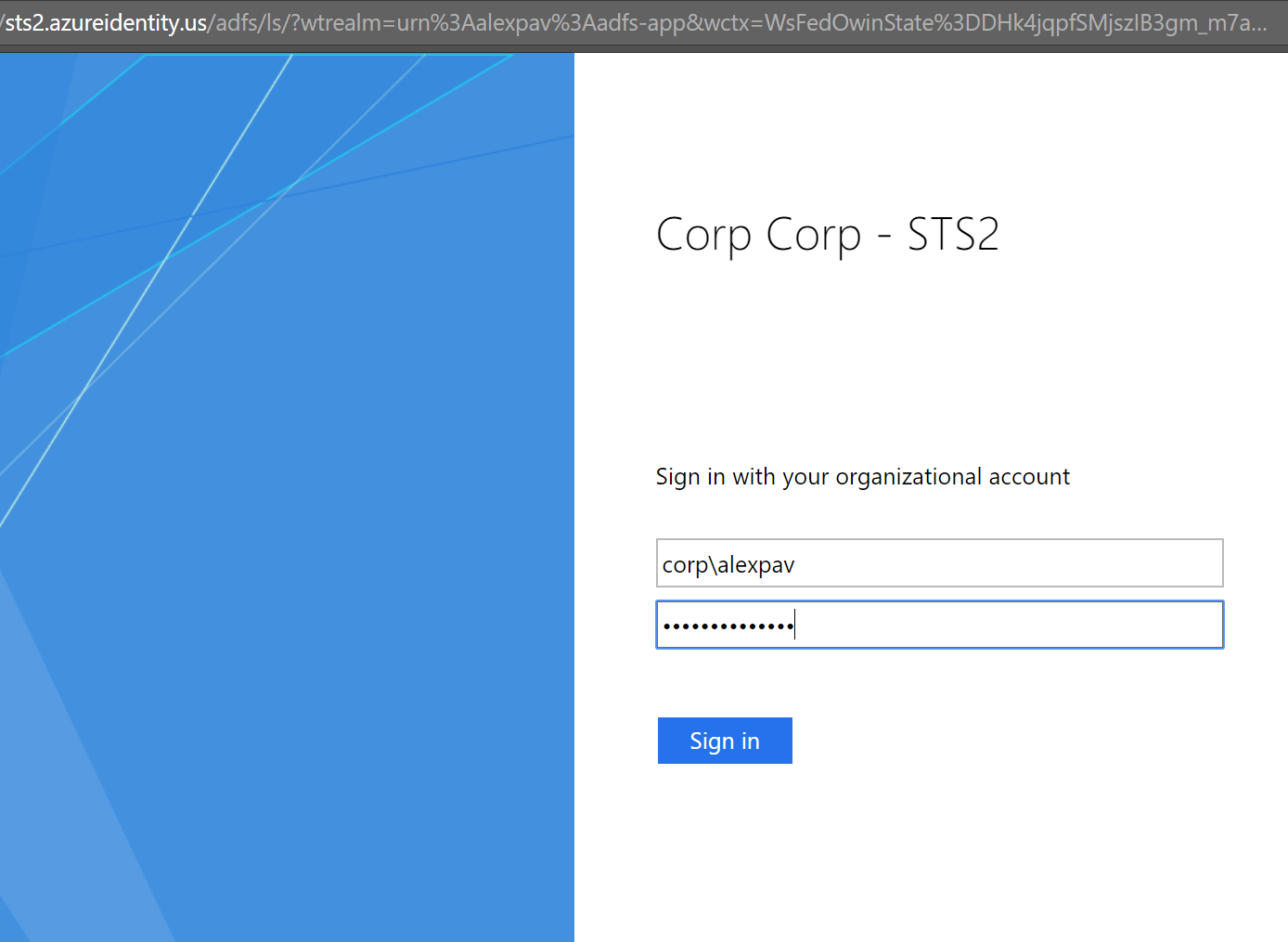
**onload.js** – to be included in the customized AD FS web theme

**variables.js** – to be hosted outside of AD FS to easily edit variables without making changes to the web theme in AD FS. This example uses external.js hosted in an Azure storage account (blob)

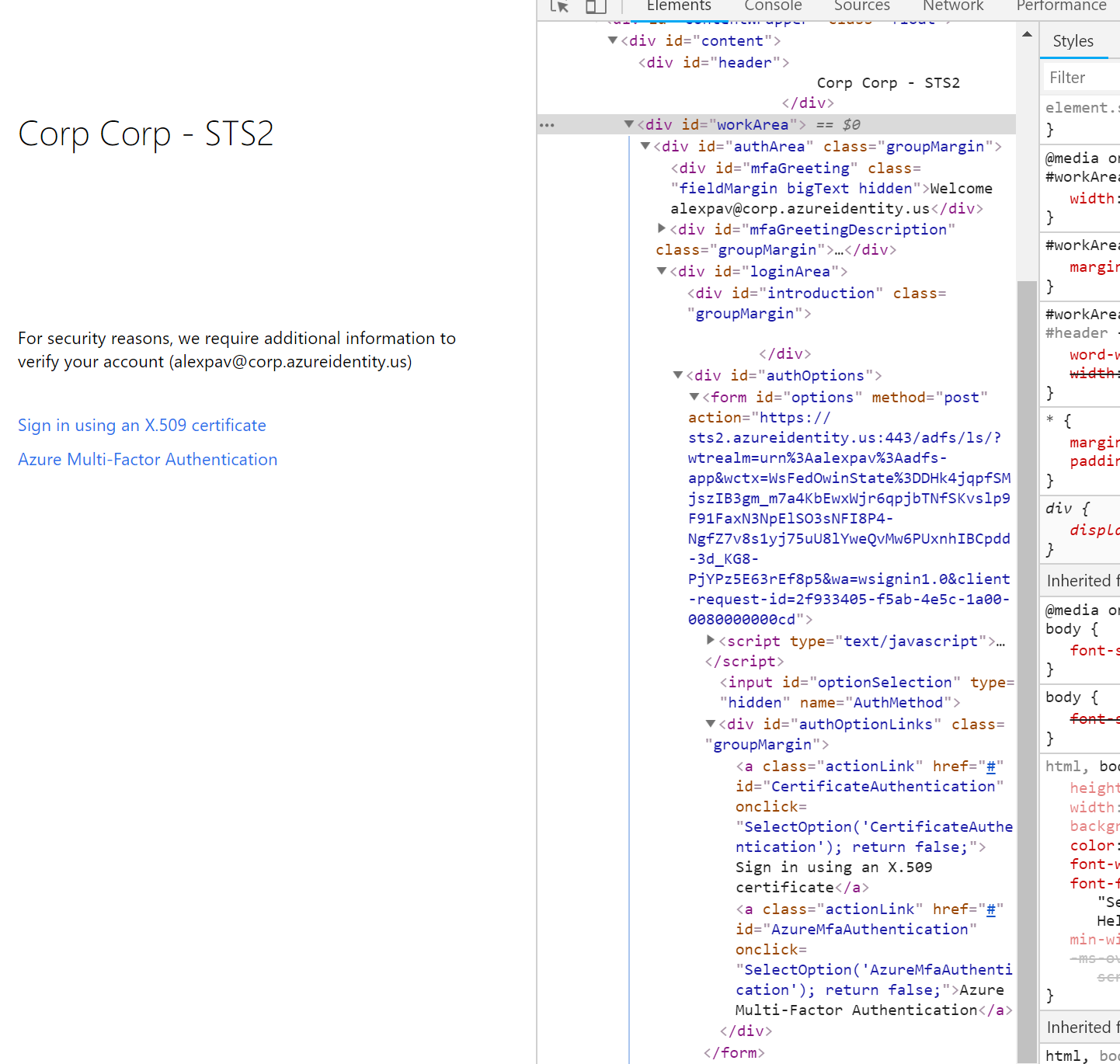
## Details

### Step 1 - Get the HTML element IDs for various MFA methods presented by AD FS to users:

1. Access the application that requires MFA from AD FS
2. Sign in to AD FS using your username and password:



1. On the MFA method selection page, access the browser developer tools **(F12).** This example uses Google Chrome as the browser.



1. On the **Elements** tab of the developer tools, navigate the document structure as follows: **body** -> **fullPage** -> **contentWrapper** -> **content** ->**workArea** -> **loginArea** -> **authOptions** -> **options** -> **authOptionLinks**
2. Review the links in the **authOptionLinks** section and get the element id of the link that needs to be hidden. In this case, we are hiding the link with id of **AzureMfaAuthentication**

### Step 2 - Export the AD FS theme

The purpose of customizing the theme is to edit the onload.js script to hide the undesired MFA method from the user for our specific relying party.

1. On the AD FS server, access Windows PowerShell
2. Run the following commands:

#create a working folder

mkdir "c:\adfsTheme"

#export the default theme to disk for modification

#IMPORTANT: this exports the default theme. If you have already customized a theme, you may need to export the custom theme instead

Export-AdfsWebTheme -Name default -DirectoryPath "C:\adfsTheme"

### Step 3 - Edit the variables.js script

1. Determine variable values.
   1. **hideMethod** is determined by examining the HTML content of the MFA page in AD FS (see step 1, item 4)
   2. **optionHide** has three choices. Select the desired configuration option for hiding (per RP, per user list, or per both RP and user list):

//var optionHide = 'HidePerRP' (per relying party),

//var optionHide = 'HidePerUser' (per defined array of users),

//var optionHide = 'HideForBoth' (both for RP and specific users)

* 1. **uriOfRP** is the URL-safe relying party identifier from the URL string on the MFA page in AD FS (https://sts2.azureidentity.us/adfs/ls/?wtrealm=**urn%3Aalexpav%3Aadfs-app**&wctx=WsFedOwinState%3Dhphd-8LEeFDUzmzFsRwCUzi1we8-jWdQS8mla1i1ghZBfsRO0mL2WB4DlJ5pQbc7nzLLEtzoPbO\_JRtYNs216RwwGnKAwm9d1OmZ-UavOPg5c5rknTebSG5HYOB7UzEh&wa=wsignin1.0)
  2. **pilotUsersArray** is the list of users in a JavaScript array format

1. Save the variables.js script and upload it onto an SSL protected externally accessible web server, such as an Azure Storage Account. Get the full URL of the variables.js file so that it can be referenced in onload.js

### Edit the onload.js script

1. Open **C:\adfsTheme\script\onload.js**
2. The fundamental of how the script works is that we use a loadExternal callback function that adds the DOM element with the external script. Details of why this has to be done are outlined on [this Stack Overflow forum post](https://stackoverflow.com/questions/950087/how-do-i-include-a-javascript-file-in-another-javascript-file). **IMPORTANT: Some of this may be browser type dependent. Please perform your testing and validation for multiple platforms.**
3. Add the following content after editing the script source to point to your URL location for the variables.js file:

var scriptSource = "https://azurestorageblob1.blob.core.windows.net/adfs/variables.js";

function loadExternal(url, callback) {

var head = document.getElementsByTagName('head')[0];

var script = document.createElement('script');

script.type = 'text/javascript';

script.async = false;

script.src = url;

script.onreadystatechange = callback;

script.onload = callback;

head.appendChild(script);

}

var hideFunction = function () {

//get the user ID from the page

var userId = document.getElementById('mfaGreeting').innerText.split(" ")[1];

//hiding for RP only

if (optionHide == 'HidePerRP') {

// Check whether the MFA provider to-be-hidden element is present on the page

var mfaElement = document.getElementById(hideMethod);

//get the current page URL. The relying party identifier is in the URL. Needs to be tested for various scenarios (IDP vs SP initiated, etc.)

var urlHref = window.location.href;

//Relying party name in URL safe format that we are detecting

if (mfaElement && (urlHref.indexOf(uriOfRP) !== -1)) //if the MFA element is present AND the current page URL contains the identifier

{

//hide the desired MFA method.

document.getElementById(hideMethod).style.display = 'none';

}

}

if (optionHide == 'HideForBoth') {

// Check whether the undesired element is present on the page

var mfaElement = document.getElementById(hideMethod);

//get the current page URL. The relying party identifier is in the URL. Needs to be tested for various scenarios (IDP vs SP initiated, etc.)

var urlHref = window.location.href;

//Relying party name in URL safe format that we are detecting

//if the MFA element is present AND the current page URL contains the identifier

//as well as if the user is in the user list

if (mfaElement && (urlHref.indexOf(uriOfRP) !== -1) && (pilotUsersArray.indexOf(userId !== -1))) {

//hide the desired MFA method.

document.getElementById(hideMethod).style.display = 'none';

}

}

if (optionHide == 'HidePerUser') {

// Check whether the undesired element is present on the page

var mfaElement = document.getElementById(hideMethod);

if (mfaElement && (pilotUsersArray.indexOf(userId) !== -1)) {

//hide the desired MFA method.

document.getElementById(hideMethod).style.display = 'none';

}

}

};

loadExternal(scriptSource, hideFunction);

### Add the modified onload.js script to the new theme and apply the theme

1. Open Windows PowerShell on the AD FS server
2. Create a new theme on the AD FS server:

#IMPORTANT: if the theme has been previously modified, the source of the theme

should be the name of the theme and not ‘default’

New-AdfsWebTheme –Name HideAzureMFA –SourceName default

1. Add the edited onload.js to the new theme:

Set-AdfsWebTheme -TargetName HideAzureMFA -AdditionalFileResource @{Uri=’/adfs/portal/script/onload.js’;path="c:\adfsTheme\script\onload.js"}

1. Apply the new theme with edited onload.js to AD FS:

Set-AdfsWebConfig -ActiveThemeName HideAzureMFA

### Testing

Access the web application and sign in to AD FS using the user name and password. After sign-in, when MFA is presented, the undesired MFA method is hidden:

