Trust Framework Policy

Hands-on Lab and Interactive Discussion

Released:

Terms of Use

Microsoft Confidential - For Internal Use Only ***-*** Do Not Distribute Outside of Microsoft

© 2016 Microsoft Corporation. All rights reserved.

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Microsoft Copyright Permissions** at [**http://www.microsoft.com/permission**](http://www.microsoft.com/permission)

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The Microsoft company name and Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

**This document reflects current views and assumptions as of the date of development and is subject to change.  Actual and future results and trends may differ materially from any forward-looking statements.  Microsoft assumes no responsibility for errors or omissions in the materials.**

**THIS DOCUMENT IS FOR INFORMATIONAL AND TRAINING PURPOSES ONLY AND IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.**

|  |  |
| --- | --- |
| About the Author | |
| **Author**: | **Omer Iqbal** Principal Software Engineer Identity Engineering Group |

“About the Authors” is an important item as it establishes the credibility of the text by explaining how much experience the author team has in the field. Add as many “About the Authors” as needed from the Quick Parts Gallery. Change “Author” to a different title, such as “Subject Matter Expert” if that is more appropriate. Include SMEs who assisted with the development of the course. Add this to every Module.

Table of Contents

[Lab Environment 1](#_Toc451274645)

[Lab Overview 2](#_Toc451274646)

[Exercise 1: Setup the Trust Framework policy and Relying Party policies 3](#_Toc451274647)

[Set up Notepad++ with XML Tools (Optional) 3](#_Toc451274648)

[Set up XML Notepad (Optional) 4](#_Toc451274649)

[Setup cryptographic keys 5](#_Toc451274650)

[Exercise 1: Create a new User Journey to sign in a user without MFA 6](#_Toc451274651)

[Exercise 2: Collect a new attribute from user during sign up and send in the token 7](#_Toc451274652)

[Exercise 3: Persist a claim in the directory, and read and send in token from directory 8](#_Toc451274653)

[Exercise 4: Apply a claims transformation to create a new claim 10](#_Toc451274654)

[Appendix 11](#_Toc451274655)

# Lab and Discussion Overview

Several paragraphs – or longer – describing an overview of the lab including a description of the lab and why certain topics are covered. This is also called the ‘LAB ABSTRACT’ that will be used for hand-off to conferences as part of the content hand-off process

During this lab, you will run several exercises that will help you achieve a better understanding of the Trust Framework Policy, and enable you to modify the policy to achieve certain business goals.

This is an interactive format open to discussions, so feel free to ask question as we move along.

#### Learning Objectives

After completing the exercises in this lab, you will be able to:

* Understand the relationship between different elements of the policy, such as User Journeys, Technical Profiles, Claim Types, and Transformations.
* Be able to create XML policies for various authentication scenarios, and upload them in the portal for use by applications.
* Create a new user journey to create a different user experience.
* Add a new claim type:
  + Collect values from the user during sign up.
  + Send those values in the token to the application.
  + Persist those values in the directory, and read them back to send in the token.
* Transform a claim to create a new claim to meet certain business requirements, such as for privacy.

# Exercise 1: Setup the Trust Framework policy and Relying Party policies

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

In this exercise you will setup the environment for editing, and executing XML policies.

After completing this exercise, you will understand the XML schema for trust framework policies, setup cryptographic keys in your tenant, upload the policies in your tenant, and run them using the B2C admin portal.

The following two tasks have steps to set up XML tools. You can also use any text editor, such as Notepad, but these tools make XML editing easier.

## Set up Notepad++ with XML Tools (Optional)

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Setup Notepad++ | 1. Go to <https://notepad-plus-plus.org/download> and install the latest version. |
| Install XML Tools plugin | 1. Open Notepad++. 2. Click **Plugins | Plugin Manager | Show Plugin Manager**. 3. From the list, select **XML Tools** and click **Install**. 4. After the installation completes, you should be able to click **Plugins** menu option in Notepad++ and see a new option **XML Tools** appear. |
| Validate a policy using the provided XSD | 1. Click **File | Open** and open **lamnahealth.onmicrosoft.com\_B2C\_1A\_base.xml** policy. 2. Click **Plugins | XML Tools | Validate Now**. 3. Put the path to the **TrustFrameworkPolicy\_0.3.0.0.xsd** in the text box and click **OK**. 4. A message box should show stating “XML is valid.” 5. Add the following line at line 10 (right after that **TrustFrameworkPolicy** node:   <Foo></Foo>   1. Validate XML again by clicking **Plugins | XML Tools | Validate Now**. 2. An error message similar to the following should show:   ERROR: Element '{http://schemas.microsoft.com/online/cpim/schemas/2013/06}**Foo**': This element is not expected. Expected is one of ( {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**BasePolicy**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**Contacts**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**DocumentReferences**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**BuildingBlocks**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**ClaimsProviders**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**UserJourneys**, {http://schemas.microsoft.com/online/cpim/schemas/2013/06}**RelyingParty** ). |

## Set up XML Notepad (Optional)

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Setup Notepad++ | 1. Go to <http://xmlnotepad.codeplex.com/releases> and click on the link for the latest release. 2. From the link, download XML Notepad. |
| Open and view a policy | 1. Open XML Notepad, click **File | Open** and open **lamnahealth.onmicrosoft.com\_b2c\_1a\_base.xml** policy. 2. The left hand side **Tree View** shows the policy elements in a hierarchy. You can expand any of those elements to go deeper into the policy. |
| Associate the policy with XSD | 1. Click **View| Schemas**, which will open a dialog box. 2. Click **File | Add Schemas**, put the path to **TrustFrameworkPolicy\_0.3.0.0.xsd** in the text box and click **Open**. 3. This will add the schema to the XML Schemas list. Click **OK**. 4. To test whether the XSD is linked correctly, expand **BuildingBlocks** and right click **ClaimsSchema**. 5. In the context menu, click **Element | After**. It will show a list of possible elements that can appear from the XSD. 6. You can select any one of them, which will result in a validation error at the bottom of the screen. This is because all of the elements have already been added and duplicates are not allowed. 7. Delete the newly created node, or click **File | Reload** to discard the changes. |

## Setup cryptographic keys

### IDP secrets – no setup required

A policy typically requires a number of cryptographic keys to be setup. These are required by various providers. For example, federation with Facebook requires a client secret for Facebook to trust AAD on behalf of the application.

The policy that you are provided with already references the same Facebook secret that was created through the portal. Other IDP secrets can be handled similarly.

### Setup token related keys

The policy references two additional secrets that need to be setup:

* TokenSigningKeyContainer – used for signing tokens.
* TokenEncryptionKeyContainer – encrypts certain tokens, e.g. the refresh token, that can be decrypted by Azure AD later.
  + Refresh token, auth. code, etc. are sent by Azure AD to the client and encrypted to maintain confidentiality and integrity. When Azure AD reads the token back, it is decrypted using the same key container.

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Create required keys for the policies | 1. Go to <https://github.com/beejones/B2CDemoTools> 2. Click **Download ZIP** and decompress it. 3. All DLLs need to be unblocked ☹ Right click on the individual DLLs, click **Properties | Unblock | OK**. 4. After that, open PowerShell, go that folder, and type the following:   import-module .\ExploreAdmin.dll   1. The enter the following commands to create the keys:   New-CpimKeyContainer lamnahealth.onmicrosoft.com TokenSigningKeyContainer TokenSigningKeyContainer rsa 2048 0 0  New-CpimKeyContainer lamnahealth.onmicrosoft.com TokenEncryptionKeyContainer TokenEncryptionKeyContainer rsa 2048 0 0 |

## Upload policies and run user journeys

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Modify the tenantId in the policies | 1. Open **lamnahealth.onmicrosoft.com\_B2C\_1A\_base.xml** policy. 2. In the root **TrustFrameworkPolicy** element, update the **TenantId** attribute.    * At this point, you may also want to rename the file to reflect the tenant, but the file name is not used by the B2C system when the policies are uploaded. 3. All other policies (e.g. **lamnahealth.onmicrosoft.com\_signup.xml**) contain the tenant id of the policy as well as the tenant of the base policy. Both need to be updated.    * The tenant id and policy id of the base policy are referenced in **BasePolicy** element. Modify the **TenantId** element in **BasePolicy** element to point to the correct element. |
| Upload the policies | 1. Go to <https://portal.azure.com>, and sign in using an admin account of that tenant, e.g. [admin@tenant.onmicrosoft.com](mailto:admin@tenant.onmicrosoft.com). 2. Click **Browse | Azure AD B2C**. 3. Once the B2C blade opens, click **All Policies**. 4. Click on the **Upload Policy** button at the top of the blade, and press the folder icon to select the **\*\_base.xml** policy. The base policy must be uploaded first as the other policies depend on this. 5. When the policy is uploaded, the name is prepended with **B2C\_1A\_**. This is done to differentiate between the policies created by the portal and the handcrafted ones. 6. After the base policy is uploaded, other policies can be uploaded in any order. |
| Run the policies | 1. In the **All Policies** blade, click on a policy, such as **B2C\_1A\_signin**, to open the policy blade. 2. Select an application and click **Run now** to test the policy. |

# Exercise 1: Create a new User Journey to sign in a user without MFA

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

#### Scenario

In this exercise you will modify the existing sign in user journey to create another sign in user journey without MFA.

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Copy an existing user journey to create a new one | 1. Open the **\*\_base.xml** policy and locate the **SignIn** user journey. 2. Copy the XML between **<UserJourney>** and **</UserJourney>** elements, and paste it as an independent user journey. 3. Rename the **Id** attribute on the copied **UserJourney** element to **SignIn\_NoMFA** or something similar. |
| Create a new policy to execute the new user journey | 1. Copy the **\*\_signin.xml** policy to **\*\_signin\_nomfa.xml** or something similar. 2. Open the **\*\_signin\_nomfa.xml** and change the **PolicyId** attribute on **TrustFrameworkPolicy** element to **signin\_nomfa**. 3. Change the **ReferenceId** attribute of the **DefaultUserJourney** element to **SignIn\_NoMFA**. 4. Save and upload the policy using Azure Portal. 5. Select the newly uploaded policy from the **AllPolicies** blade after it has been uploaded and click the **Run Now** button. 6. This should allow an existing user to sign in without phone verification. |

# Exercise 2: Collect a new attribute from the user during sign up and send in the token

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

#### Scenario

In this exercise you will use a newly created attribute in the B2C admin portal in the sign up policy to collect it from the user.

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Add a claim type for the new attribute in the claims schema section. | 1. Open the **\*\_base.xml** policy and go to the **ClaimsSchema** section. 2. Copy the **givenName** claim type that is at the end of the ClaimsSchema section. It has all the important attributes required of a new claim type. 3. In the copied ClaimType, update the **Id** to a unique value so it can be referenced from other places in the policy. 4. Update the elements to provide appropriate values. Note that **DisplayName** and **UserHelpText** are shown to the user. 5. Update **DefaultPartnerClaimTypes** to reflect the *default* claim types that you expect to go in the token to the application that use various protocols. |
| Add the attribute in the self-asserted provider to collect from the user in local account creation | 1. In the **\*\_base.xml** policy and go to the **SignUp** user journey. 2. In step 2, locate the technical profile name for the claims exchange **SignUpWithLogonEmailExchange**. It should be **LocalAccountSignUpWithLogonEmail**. 3. Search that technical profile in the XML policy. In the OutputClaims, add the newly created claim. 4. Essentially an OutputClaim in the SelfAssertedAttributeProvider indicates that this claim needs to be sent back by the provider and thus will be sourced from the user. |
| Send the attribute to the application in the token | 1. Open **\*\_signup.xml** policy and add a new **OutputClaim** with the **ClaimTypeReferenceId** set to the **Id** of the new **ClaimType** you added in the **\*\_base.xml** policy. 2. Save, upload and run the policy using Azure Portal to test the new attribute being collected from the user during local account creation, and sent in the token. |

# Exercise 3: Persist an attribute in the directory, and enable user to edit it

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

#### Scenario

In this exercise you will persist a claim collected from the user in the directory during sign up, and then send it in the token when the user signs in.

After completing this exercise, you will understand:

* The role of an application in schema extensions.
* How B2C creates and uses an application to extend the schema.
* How Azure AD technical profiles persist claims in the directory, and read them from the directory.
* Interaction between self-asserted provider and AAD provider.

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Create a new attribute in the B2C admin portal | 1. Go to <http://portal.azure.com> and sign in as the tenant admin. 2. Click **Browse | User attributes | Add**. 3. Provide the name (e.g. gamerTag) and description for the attribute and click **Create**. 4. Close the **User attributes** blade. |
| Use the attribute in a sign-up policy to create the attribute in directory (current workaround) | 1. Click **Sign-up policies** to open the policies blade. 2. Select an existing sign-up policy, click **Edit** and then **Sign-up attributes**. 3. Select this newly create attribute and click **OK**. 4. This attribute will now be created in the directory when the policy is saved. |
| Find the name of the extension property from Graph API  *All extension properties are created by an application and AAD B2C creates an application in every tenant called b2c-extensions-app for this purpose* | 1. Find the b2c-extensions-app application:    * EITHER type **b2c get-b2c-application** to invoke b2c app and get the extension application OR    * Sign in to <https://graphexplorer.cloudapp.net> using tenant admin and issue a GET query with https://graph.windows.net/<*tenant>* /applications/?$filter=displayName+eq+'b2c-extensions-app' 2. Copy the **appId** and **objectId** from the application JSON from the response. 3. Get a list of all extension properties on this application:    * EITHER type **b2c get-extension-attribute *<objectId>*** OR    * Issue a GET with https://graph.windows.net/*<tenant>*/applications/<*objectId*>/extensionProperties 4. Both of these will give a list of extension properties in JSON. 5. Find the extension property you just created and copy its **name**. An example is extension\_9a485fd9c0054e229e6f74ed8b19f9b9\_GamerTag from my tenant. |
| Configure policy with extensions application  *All AAD technical profiles include AAD-Common, which allows adding all configuration in one place* | 1. In the **\*\_base.xml** policy, locate the **AAD-Common** technical profile. 2. Add a **Metadata** section with two items, one containing the **objectId** of the extensions application and the other with its **appId**:   <Metadata>  <Item Key="ApplicationObjectId">*objectId*</Item>  <Item Key="ClientId">*appId*</Item>  </Metadata>   1. These settings B2C locate the application and use it during claims persistence. |
| Update the AAD technical profile to persist the new claim for local account sign up  *ValidationTechnicalProfiles are used for validating user data, which includes persisting them in the directory* | 1. If you recall, the **LocalAccountSignUpWithLogonEmail** self-asserted profile was used during local account creation. It references the **AAD-UserWriteUsingLogonEmail** profile. 2. This is marked as **ValidationTechnicalProfile** which means that this profile is called to validate the user provided data before user journey moves forward. As part of this validation, the data is persisted in the directory as well. 3. Add a **PersistedClaim** in the **AAD-UserWriteUsingLogonEmail** with the **ClaimTypeReferenceId** as the **Id** of the **ClaimType** created in the **ClaimsSchema** section in previous lab and **PartnerClaimType** as the **name** of the extension property, e.g.:   <PersistedClaim  ClaimTypeReferenceId="gamerTag"  PartnerClaimType="extension\_9a5fd05229\_GamerTag"/> |
| Update the AAD technical profile used during sign in to read the claim and send in token | 1. The **SignIn** user journey uses **AAD-UserReadUsingObjectId** to read user’s claims for local account sign in (in orchestration step 4). 2. In the **AAD-UserReadUsingObjectId** add an **OutputClaim** with ClaimTypeReferenceId and PartnerClaimType similar to the one added before:   <OutputClaim  ClaimTypeReferenceId="gamerTag"  PartnerClaimType="extension\_9a5fd05229\_GamerTag"/>   1. Now run the SignUp policy to create a new user which should persist this new claim, and then run the SignIn policy to verify that the claim is in the token. |
| Show the claim in profile edit so user can update it | 1. The **EditProfile** user journey also uses **AAD-UserReadUsingObjectId** to read the claims after signing in the user. That profile is already updated to read the new claim. 2. It uses the **SelfAsserted-ProfileUpdate** profile for displaying the claims to the user, and then updating them in the directory via **ValidationTechnicalProfile**. 3. Add the new claim type as the **InputClaim** and **OutputClaim** of the **SelfAsserted-ProfileUpdate**. PartnerClaimType is not needed:   <InputClaim ClaimTypeReferenceId="gamerTag"/>   1. The InputClaim ensures that the value read from the directory is sent to the SelfAsserted provider so it can display it to the user. OutputClaim gets that value back so it can be sent in the token. 2. Update the **ValidationTechnicalProfile** (**AAD-UserWriteProfileUsingObjectId**) and add this claim as the **PersistedClaim**. This will ensure that the updated claim is written to the directory. 3. Execute the EditProfile policy to confirm that the claim has been added to the directory. |

# Exercise 4: Apply a claims transformation to create a new claim

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

#### Scenario

In this exercise you will create a new claims transformation to create a displayName from givenName and surname, and send it in the token.

After completing this exercise, you will understand:

* How claims transformations are declared in the policy, the role of InputClaims, OutputClaims and InputParameters.
* The relationship between claims transformation and claims providers.

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Create the claims transformation | 1. In the **\*\_base.xml** policy, go to the **ClaimsTransformations** section. 2. Add a new **ClaimsTransformation** which uses a predefined **TransformationMethod**, provides the **InputClaims** and **InputParameters**, and returns **OutputClaims**.   <ClaimsTransformation Id="CreateDisplayNameFromGivenNameAndSurname" TransformationMethod="FormatStringMultipleClaims">  <InputClaims>  <InputClaim ClaimTypeReferenceId="givenName" TransformationClaimType="inputClaim1" />  <InputClaim ClaimTypeReferenceId="surname" TransformationClaimType="inputClaim2" />  </InputClaims>  <InputParameters>  <InputParameter Id="stringFormat" DataType="string" Value="{0} {1}" />  </InputParameters>  <OutputClaims>  <OutputClaim ClaimTypeReferenceId="displayName" TransformationClaimType="outputClaim" />  </OutputClaims>  </ClaimsTransformation>   1. This is just the declaration of the transformation, the **Id** is what the transformation will be referenced using. |
| Reference the claims transformation from a claims provider | 1. In the technical profile **LocalAccountSignUpWithLogonEmail**, add the following **OutputClaimsTransformation**:   <OutputClaimsTransformations>  <OutputClaimsTransformation ReferenceId="CreateDisplayNameFromGivenNameAndSurname"/>  </OutputClaimsTransformations>   1. Now when the technical profile will be invoked and claims are returned from that technical profile, this claims transformation will be applied to create (or overwrite) the displayName claim. 2. Execute the **\*\_signup.xml** policy again, displayName should appear in the token. |