Microsoft

Azure AD B2C Custom Policies

Deep Dive on Custom Policy Schema

Microsoft Corporation
Published: September 2018
Version: 0.9 (DRAFT)

Author: Philippe Beraud (Microsoft France)
Contributors/Reviewers: Marcelo di Iorio (Microsoft Spain), Kim Cameron, Brandon Murdoch, Ronny Bjones, Jose Rojas (Microsoft Corporation)

For the latest information on Azure Active Directory, please see
http://azure.microsoft.com/en-us/services/active-directory/

Abstract: Azure AD, the Identity Management as a Service (IDaaS) cloud multi-tenant service with proven ability to handle billions of authentications per day, extends its capabilities to manage consumer identities with a new service for Business-to-Consumer (B2C): Azure AD B2C.

Azure AD B2C is "IDaaS for Customers and Citizens" designed with Azure AD privacy, security, availability, and scalability for customer/citizen identity and access management (CIAM). It's a comprehensive, cloud-based, 100% policy driven solution where declarative policies encode the identity behaviors and experiences as well as the relationships of trust and authority inside a Trust Framework (TF).

Whilst the built-in policies in Azure AD B2C leverage a dedicated TF tailored by Microsoft, i.e. the "Microsoft Basic Trust Framework" in which you can set up for your configuration these predefined policies, the custom policies give you full control, and thus allows you to author and create your own Trust Framework through declarative policies. They thus provide you with all the requirements of an Identity "Hub".

This document is intended for IT professionals, system architects, and developers who are interested in understanding the advanced capabilities Azure AD B2C provides, and more particularly in this context how to author their own Trust Framework (custom policies).

Table of Content

Notice

This document covers the custom policies[1] now available for evaluation under public preview for all Azure Active Directory B2C (Azure AD B2C) customers. Custom policies are designed primarily for advanced identity pros/developers who need to address the most complex identity scenarios.

This feature set indeed requires developers to configure the Identity Experience Framework (mostly) directly via XML file editing. This method of configuration is powerful but more complex. Advanced identity pros/developers using the Identity Experience Framework should plan to invest some time completing walk-throughs and reading the online reference documentation beyond this series of documents.

For most scenarios, we recommend that you use the built-in policies[2] of Azure AD 2BC. Built-in policies are easier to set up for your configuration. You can use built-in and custom policies in the same Azure AD B2C tenant.

As of this writing, custom policies are in public preview and may be substantially modified before GA. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[3].

This document will be updated to reflect the changes introduced at GA time for custom policies.

This document reflects current views and assumptions be 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.

---

[1] AZURE ACTIVE DIRECTORY B2C: CUSTOM POLICIES: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-overview-custom

[2] AZURE ACTIVE DIRECTORY B2C: BUILT-IN POLICIES: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-policies

[3] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

Introduction

Azure AD B2C is a cloud identity service for your consumer-facing web and mobile applications. Azure AD B2C is designed to solve the identity management challenges that have emerged, as economic and competitive pressures drive commercial enterprises, educational institutions, and government agencies to shift their service delivery channels from face-to-face engagements to online web and mobile applications.

Based on standardized protocols, Azure AD B2C is "IDaaS for Customers and Citizens" designed with Azure AD privacy, security, availability, and scalability for customer/citizen identity and access management (CIAM). The "secret sauce" of Azure AD B2C to achieve the above objectives resides in the 100% policy driven Identity Experience Framework that consume fit to purpose declarative policies.

Many of the most frequently used identity use cases can be addresses using the B2C extension in the Azure portal as the developer control surface. However, there some advanced features only available by writing custom user journeys which must be configured directly into policy XML files and uploaded to the B2C tenant. Access to this incremental feature set is available via the custom policies in Azure AD B2C as per currently available public preview.

> **Note** For a basic level of proficiency with the policy configuration available directly in the B2C Admin portal, see the introduction video BUSINESS-TO-CONSUMER IDENTITY MANAGEMENT WITH AZURE ACTIVE DIRECTORY B2C[4] where all the relevant B2C Admin **portal** settings are.

Azure AD B2C indeed allows you to author and create your own (Trust Framework) custom policies from scratch if you want to for your user journeys. You're completely in control of the set of policies and no longer restricted to the configuration of predefined built-in policies within the "Microsoft Basic Trust Framework" for the policies that are defined form the portal UI. The intention is to create a flexible model for a custom policy definition to support a wide variety in the operational complexity and governance rigor of the policy that is to be enforced for your more advanced user journeys.

The custom policy XML files can define and configure:

- The document reference(s) defining the federated identity ecosystem of the community that relates to them.
- The (predefined) operational "runtime" rules, a.k.a. the user journeys that automate and/or control the exchange and usage of the claims.
- The identity and attribute providers, a.k.a. the *claims* providers, and the technical profiles they support along with the (out-of-band) LOA/LOP accreditation that relates to these providers.
- The integration with attribute verifiers, a.k.a. the *claims* providers.
- The relying parties in the community (by inference).
- The metadata for establishing network communications between participants. These metadata along with the technical profiles will be used in the course of a transaction to plumb "on the wire" interoperability between the relying party and other community participants.
- The protocol conversion if any (SAML 2.0, WS-Fed, WS-Trust, OAuth2, and OpenID Connect).

---

[4] BUSINESS-TO-CONSUMER IDENTITY MANAGEMENT WITH AZURE ACTIVE DIRECTORY B2C: https://channel9.msdn.com/Events/Build/2016/P423

- The authentication requirements.
- The multifactor orchestration if any.
- A shared schema for all the claims available and mappings to participants of a community of interest.
- All the *claims* transformations - along with the possible data minimization in this context - to sustain the exchange and usage of the claims.
- The blinding and encryption.
- The claims storage.
- Etc.

These policies constitute the machine-readable portion of the TF construct in Azure AD B2C with all the operational details (claims providers' metadata and technical profiles, claims schema definition, claims transformation functions, and user journeys, etc.) filled in (for a specific community of interest) to facilitate operational orchestration and automation by the Azure AD B2C Identity Experience Framework. Policies should be considered as a domain-specific language (DSL), i.e. "a computer language specialized to a particular application domain"[5] with inheritance, "if" statements, polymorphism.

They are assumed to be "living documents" in Azure AD B2C since there is more than a chance that their contents will change over time with respect to the active participants declared in the policies, and also potentially in some situations to the terms and conditions for being a participant (in a community of interested).

To offer such a flexibility and appropriately deal with the complexity this implies, the retained approach in Azure AD B2C unsurprisingly consists in leveraging the XML capabilities along with the advanced tooling available in this space for defining and consuming from a computing standpoint and thus for processing purposes a specific policy.

Objectives of this document

A specific policy will be defined in a policy XML file. As such, a policy XML file allow to define all the policy details along with the metadata to enforce on the transactions that will take place between the relying parties that subscribe to the policy and the claims providers.

Available as part of the "Starter Pack", a well-defined XML schema allows (see section § *Appendix A. Trust Framework Policy XML schema and tools*) allows to model such a flexible policy XML file to fully author a specific policy. Any policy XML file MUST respect this XML schema definition.

> **Note** For information on the "Starter Pack" and how to get started with, see the second of this series of document.

This sixth document aims at helping you achieve a better understanding of the Trust Framework (TF) policies, the XML schema that sustains them, and thus enable you to modify the policy XML file(s) (based on the core templates the "Starter Pack") to achieve certain business goals and your advanced identity use cases that come along with.

[5] Domain-specific language: https://en.wikipedia.org/wiki/Domain-specific_language

It is intended as a reference document for describing how a TF policy is logically structured in accordance to the above XML schema.

Non-objectives of this document

This series of documents is not intended as an overview document for the Azure AD offerings but rather focusses on this Azure AD B2C identity service, and more specifically on the custom policies as per currently available public preview.

> **Note** For more information, see the article Getting started with Azure AD[6]. As well as the whitepapers Active Directory from the on-premises to the cloud[7] and An overview of Azure AD[8] as part of the same series of documents.

Organization of this document

To cover the aforementioned objectives, this document of the series is organized in the following eight sections:

- Declaring a custom policy.
- Understanding the technical components of a custom policy.
- Specifying a base policy.
- Specifying the building blocks.
- Specifying the claims providers.
- Specifying the token issuers.
- Specifying the user journeys.
- Specifying the relying party.

These sections provide the information details necessary to successfully build your own policies based on the already available features as per the currently available public preview.

About the audience

This document is intended for IT professionals, system architects, and developers who are interested in understanding how to author their own policies.

[6] Getting started with Azure AD: https://docs.microsoft.com/en-us/azure/active-directory/get-started-azure-ad
[7] Active Directory from the on-premises to the cloud: https://aka.ms/aadpapers
[8] An overview of Azure AD: https://aka.ms/aadpapers

Declaring a custom policy

Specifying the root element of policy

A specific custom policy is defined within the top level *TrustFrameworkPolicy* XML element of a policy XML file.

The *TrustFrameworkPolicy* XML element contains the following attributes:

Attribute	Required	Description
PolicySchemaVersion	True	Determine the schema version published by Microsoft using which this policy is to be executed. As of this writing, this must be "*0.3.0.0*" Type: String
TenantId	True	Specify the unique identifier of the B2C tenant to which this policy belongs. Type: String
TenantObjectId	False	Specify the unique identifier of the object ID of the Azure tenant. Type: String
PolicyId	True	Specify the unique identifier for the policy. This identifier must be prefixed by "B2C_1A_" Type: String
PublicPolicyUri	True	Specify the URI for the policy which is an appropriate name of the policy outside of the Azure AD B2C system. Type: String
DeploymentMode	False	Indicate the mode when uploading the policy XML file in a B2C tenant. Type: String Value: one of the following values as per *DeploymentModeType* enumeration in the custom policy XML schema: • **Development**. • **Production**. (default) • **Debugging**.
UserJourneyRecorderEndpoint	False	Specify the Url of a service able to receive http posts documenting user journey progress. Type: String Value: Url in the format http://{host}?stream={guid} (where the braces are omitted)

The following XML snippet illustrates how to specify this top level *TrustFrameworkPolicy* XML element:

```
<?xml version="1.0" encoding="utf-8"?>

<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           TenantId="litware369b2c.onmicrosoft.com"
           PolicyId="B2C_1A_SignUp"
           PublicPolicyUri="http://litware369b2c.onmicrosoft.com">
    …
</TrustFrameworkPolicy>
```

This top-level XML element allows in turn to specify the policy itself with possibly the participant contact information, the document references, as well as of course the technical components that constitute the policy.

These topics are covered hereafter in a dedicated section.

Specifying the participant contact information of a custom policy

Participants in a community regulated by a custom policy are defined in terms of the roles they play. These are typically the operational roles for conducting online transactions: identity provider, attribute provider, attribute verifier, and federation operator.

These are the governance roles which are typically performed through offline processes: trust framework provider (TFP) that authors the trust framework policy, assessor, auditor, dispute resolution arbitrator, and special assessor.

These approved participants should be identified in the policy, along with the role(s) they perform and their contact information. Explanations of the meanings of these roles and the qualifications for performing them are not published in the policy. This information is available in the documentation referenced by the policy (see next section below).

To specify the list of contacts who can be communicated with for notifications and issues regarding the policy, a *Contacts* XML element should be declared under the top-level XML element of the policy XML file. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
Contact	0:n	Provide a participant contact information.

The *Contact* XML element contains in turn the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify this particular contact.

Along with the following XML elements:

XML element	Occurrences	Description
DisplayName	1:1	Specify the name of the contact for a given participant in the community of interest. Type: String
TelephoneNumber	1:1	Specify the telephone number of the contact for a given participant in the community of interest. Type: String
Email	1:1	Specify the email address of the contact for a given participant in the community of interest. Type: String
Role	1:1	Specify the role of the contact for a given participant in the community of interest. Type: String

Each *DocumentReference* XML element contains a required *Id* attribute. This attribute enables to specify a machine understandable identifier that is used to uniquely identify this particular Contact for a given participant in the community of interest.

The following XML snippet illustrates how to specify participant contact information:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …

  <!-- A list of all the contacts who can be communicated with for notifications and issues regarding this policy -->
  <Contacts>
    <Contact id="1">
      <DisplayName>Litware369 Customer Support</Name>
      <TelephoneNumber>+14258828080</TelephoneNumber>
      <Email>johndoe@litware369.com</Email>
      <Role>Administrator</Role>
    </Contact>
  </Contacts>

  …
</TrustFrameworkPolicy>
```

Specifying the document references of a custom policy

The custom policy may be defined through various documents that need to be reviewed and understood by human beings. Certain documents, such as terms of use or privacy policy, may be made available to the relying parties or even the users before they sign up to the use one of the services provided by Azure AD B2C.

The relying parties (RPs) may use these documents to determine whether the policy is appropriate for the purposes it intends to use it for. The users may view these documents to look at the parameters within which RPs and the policy will operate and determine whether they want to participate or not. These documents

constitute the human readable portion of the policy construct in Azure AD B2C. The policy XML file supports link(s) to that documentation if any. The URL(s) enable hosting the policy's documentation anywhere. Moreover, the policy's documentation could be in any suitable format that can meet the UI formatting and localization requirements.

To provide a list of URLs to the policy's documentation, a *DocumentReferences* XML element should be added under the top-level XML element of the policy XML file. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
DocumentReference	0:n	Provide a reference to a document for the policy.

Each *DocumentReference* XML element contains in turn the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify this particular document reference as part of the policy's documentation if any.

This element also contains the following XML element:

XML element	Occurrences	Description
DocumentReference	0:n	Provide a reference to a content part of the documentation that relates to the policy.

The *DocumentReference* XML element contains in turn the following XML elements:

XML element	Occurrences	Description
DisplayName	1:1	Specify the display name of the document that is part of the policy's documentation.
Url	1:1	Specify the Url where the document is located. Due to formatting and localization concerns Azure AD B2C has to deal with, this enables representing the actual content in a more suitable medium, such as an HTML page or a document. Type: String.

The following XML snippet illustrates how to provide document reference(s) to the Trust Framework policy:

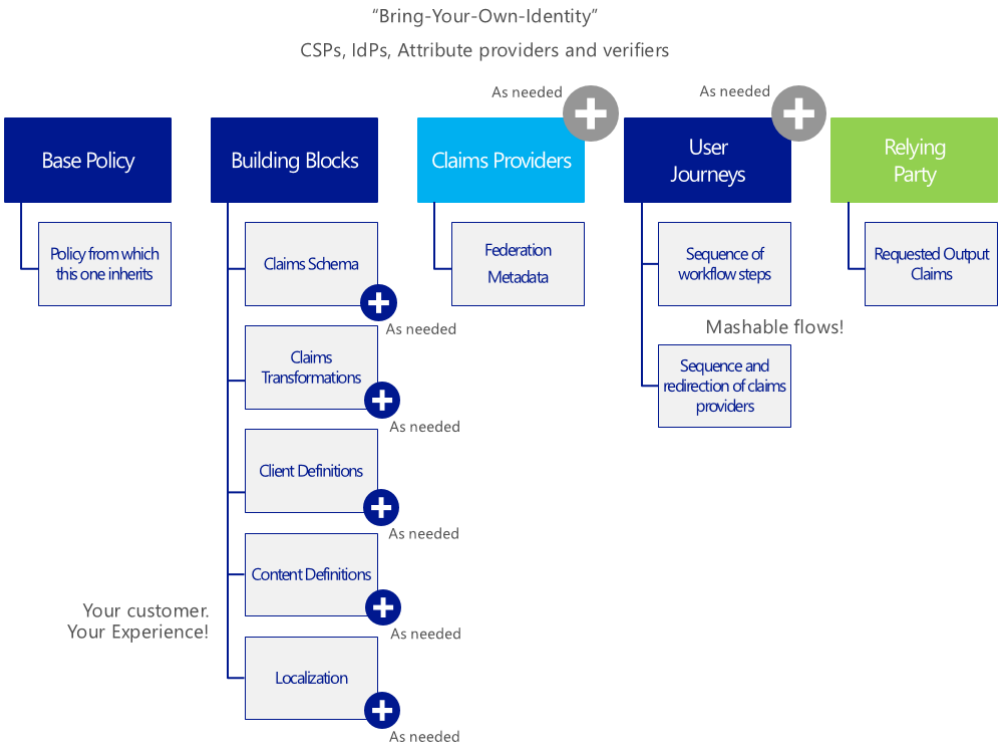
```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …

  <!-- Documents that are disclosed to the users. -->
  <DocumentReferences>
    <DocumentReference id="TermOfUser">
      <Name>Terms of Use</Name>
      <Url>http://www.contoso269.com/policy/b2Ctou.html</Url>
    </DocumentReference>
    <DocumentReference id="PrivacyTerms">
      <Name>Privacy Terms</Name>
      <Url>http://www.contoso269.com/policy/b2cprivacyterms.html</Url>
    </DocumentReference>
  </DocumentReferences>

  …
</TrustFrameworkPolicy>
```

Before digging into more details regarding the policy, let's discuss for the moment the main technical components that serve to define such a policy.

Understanding the technical components of a custom policy

As further depicted in this section, and beyond specific metadata that defines contacts' information along with document references for the policy as covered so far, a custom policy XML file truly constitutes the machine-readable portion of a policy with all the operational details:

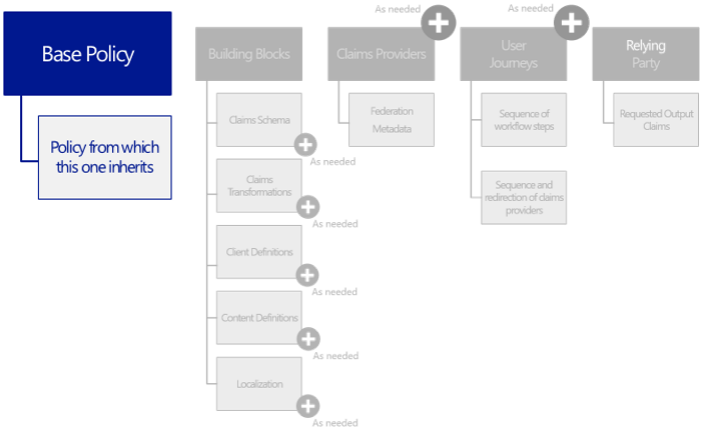
- The base policy the considered policy inherits from. Remember that a TF in Azure AD B2C is a set of policies.
- The building blocks with the claims definitions, the supported claims transformation functions, and other definitions.
- The claims providers with their supported technical profiles and related metadata that may be used in the various user journeys.
- The user journeys through which a user is taken to retrieve the claims that are to be presented to the relying party.
- And the relying party.



For example, all claims must correspond to one of the above claim schema definitions and all claims providers that participate in various user journeys must appear in the custom policy XML file. The user journeys bind the claims to the claims provider whenever a claims provider is called.

The following sections depict how to the above operational details.

Specifying a base policy



If the considered policy derives from a base policy in accordance to the supported inheritance model (see below), a *BasePolicy* XML element must then be declared under the above top-level *TrustFrameworkPolicy* XML element of the policy XML file.

The inheritance model as implemented by Azure AD B2C is as follows:

- The parent policy and child policy are of the same schema and flexibility.
- Child policy can inherit the parent policy functionality and '*extend it*'.
- Adding a new element effectively extends the parent policy.
- There is NO limit on the number of levels.

And, in terms of override:

- The child policy can override 'allowed' aspects of the parent policy.
- Same element in the child policy and parent policy implies, child policy is overriding that element of the parent policy.

Policies and counterpart policy XML files are designed to enable defining such policies, that various applications could use and thereby be compliant.

Consequently, this *BasePolicy* XML element is a reference to the base policy from which this policy is derived, or if you prefer under which the current policy is operational. The reference can be across B2C tenants.

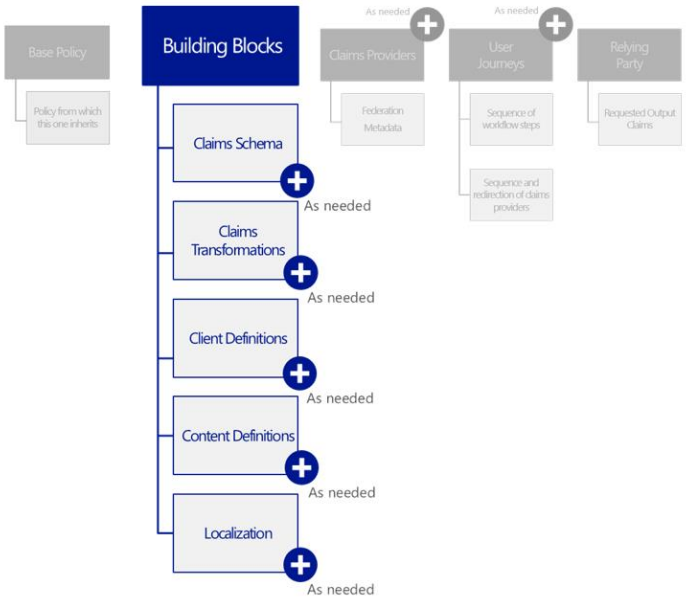
This element is optional. This element contains the following XML elements:

XML element	Occurrences	Description
TenantId	1:1	Specify the identifier of the B2C tenant that published the base policy. The base policy is looked up inside the B2C tenant specified here. Type: String
PolicyId	1:1	Specify the identifier of the base policy. The policy is looked up using this identifier within the B2C tenant specified by the above XML element. Type: String

The following XML snippet illustrates how to specify a base policy from which this policy is derived:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …

  <BasePolicy>
    <TenantId>litware369b2c.onmicrosoft.com</TenantId>
    <PolicyId>B2C_1A_base-v2</PolicyId>
  </BasePolicy>

  …
</TrustFrameworkPolicy>
```

Specifying the building blocks



User journeys may rely on various definitions. To provide all the definitions that are used by the user journeys, a *BuildingBlocks* XML element should be added under the top-level *TrustFrameworkPolicy* XML element of the policy XML file. The element consists of the claims schema, the claims transformations, the client definitions, the content definitions, and the localization support that are used elsewhere in the policy XML file.

This element is optional.

This element contains the following XML elements that define specific sections:

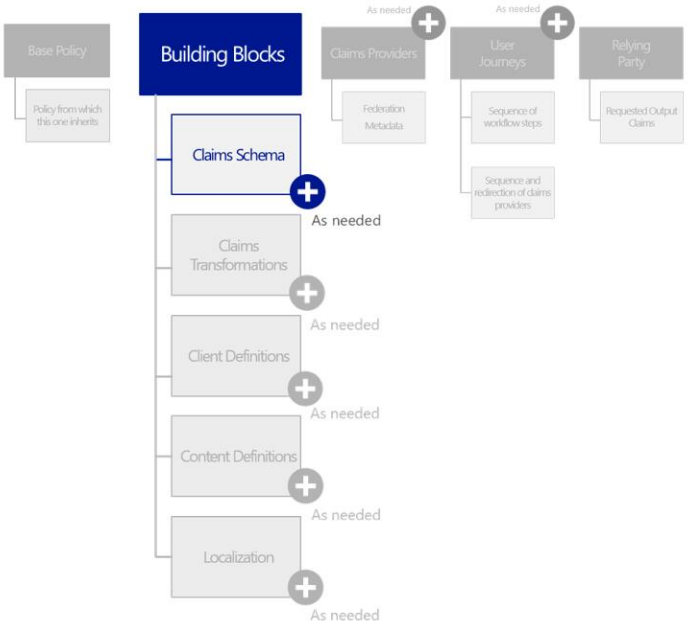
XML element	Occurrences	Description
ClaimsSchema	0:1	Provide all the claim types that can be referenced as part of the policy.
Predicates	0:1	Define basic string validations to check for claim type(s), each of them returning true or false.
InputValidations	0:1	Define input validation rules that can be used to validate claim type(s).
ClaimsTransformations	0:1	Contain a list of claims transformations that can be used in the user journeys as part of the policy.
ClientDefinitions	0:1	Specify various properties specific to the end-user device for which the policy is being executed.
ContentDefinitions	0:1	Contain URLs to external content, for example, URLs to pages used in claims providers as part of the policy.
Localization	0:1	Define the supported cultures and contains strings and collections in those cultures

The following XML snippet illustrates how to declare this *BuildingBlocks* XML element:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
          PolicySchemaVersion="0.3.0.0"
          …>
  …

  <BuildingBlocks>
    <ClaimsSchema>
      …
    </ClaimsSchema>
    <Predicates>
      …
    </Predicates>
    <InputValidations>
      …
    </InputValidations>
    <ClaimsTransformations>
      …
    </ClaiClaimsTransformations>
    <ClientDefinitions>
      …
    </ClientDefinitions>
    <ContentDefinitions>
      …
    </ContentDefinitions>
    <Localization>
      …
    </Localization>
  </BuildingBlocks>

  …
</TrustFrameworkPolicy>
```

The next sections depict each type of definitions: claims (along with input validations and related predicates if any), claims transformations, client definitions, content definitions, and localization.

Specifying the claims listing



The purpose of the claims listing (a.k.a. claims schema) in the policy aims at defining all the claim types that that are dealt with in the policy XML file, and thus that can be referenced from other XML elements of the policy XML file to. These are referenced (see *ClaimTypeReferenceId* XML attribute) from other places in the policy XML file such as technical profile's input/output claim's as well as claims transformations.

To specify the list of supported claims for the policy, a *ClaimsSchema* XML element must be declared under the *BuildingBlocks* section of the policy XML file. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
ClaimType	1:n	Provide a valid single claim type definition that can be used within the community of interested, and thus that can be referenced from other XML elements of the policy XML file.

The *ClaimType* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify this particular claim type and reference it from other XML elements in the policy XML file.
StatementType	False	Specify the type of statement the claim type represents. This type may be used by claims transformations and may thus participate in comparison or arithmetic operations. Associating an appropriate type ensures that these operations are performed correctly by the transforms

	Value: one of the following values as per *StatementType* enumeration in the custom policy XML schema; • **Attribute**. Default value if not specified. • **Authentication**. • **Subject**.

and the following XML elements.

XML element	Occurrences	Description
DisplayName	0:1	Specify the human understandable name of the claim type that is displayed to the users on various screens. Type: String
DataType	0:1	Specify the type of data stored in the claim type. This type may be used by claims transformations and may thus participate in comparison or arithmetic operations. Associating an appropriate type ensures that these operations are performed correctly by the transformations. Value: one the following supported data types as per *DataType* enumeration in the custom policy XML schema. These types are a subset of the types specified by W3C XML Schema documentation[9]: • **boolean**. Any Boolean value ("true","false",1, or 0). • **date.** Any valid date value, which consists of top-open intervals of exactly one day in length on the timelines of dateTime. • **dateTime**. Any valid date and/or time value. • **duration**. Any valid duration of time value. • **int**. Any integer value. • **string**. Any alphanumeric string value. • **stringCollection**. Any collection of alphanumeric string value. Type: String (enumeration)
DefaultPartnerClaimTypes	0:n	Enable to specify the partner claim types to use for a specified protocol if a partner claim type is not provided in a claim mapping. Type: complex type
Mask	0:1	Specify an optional string of masking characters that can be applied to the claim when displaying the claim for example the phone number 324-232-4343 masked as XXX-XXX-4343 Type: String
AdminHelpText	0:1	Specify a description of the claim type that can be helpful for the administrators to understand the purpose and/or usage of the claim type. Due to formatting and localization concerns Azure AD B2C has to deal with, this is a link to the actual content. This enables representing the actual content in a more suitable medium, such as an HTML page or a document Type: String
UserHelpText	0:1	Specify a description of the claim type that can be helpful for the end users to understand the purpose and/or usage of the claim type.

[9] XML SCHEMA PART 2: DATATYPES SECOND EDITION W3C RECOMMENDATION 28 OCTOBER 2004: http://www.w3.org/TR/xmlschema-2/

		Due to formatting and localization concerns Azure AD B2C has to deal with, this is a link to the actual content. This enables representing the actual content in a more suitable medium, such as an HTML page or a document Type: String
UserInputType	0:1	Specify the type of input control that should be available to the end user when manually entering claim data for this claim type. Type: String (enumeration) Value: one of the following supported type of input controls as per *UserInputType* enumeration in the custom policy XML schema: • **TextBox**. • **DateTimeDropdown.** • **RadioSingleSelectduration**. • **DropdownSingleSelect**. • **CheckboxMultiSelect**. • **Password**. • **Readonly**. • **Button**.
Restriction	0:1	Provide the value restrictions for this claim, such as a regular expression (Regex) or a list of acceptable values. Type: complex type
InputValidationReference	0:1	Provide the input validation rule for this claim. Type: complex type

The *DefaultPartnerClaimTypes* XML may contain the following XML element:

XML element	Occurrences	Description
Protocol	0:n	Specify a technical profile that is allowed to be used against a claims provider selection

Each *Protocol* XML element of the above *DefaultPartnerClaimTypes* section contains in turn the following attributes:

Attribute	Required	Description
Name	True	Specify the name of a valid protocol supported by Azure AD B2C. This name is one of the following list as per *ProtocolName* enumeration in the custom policy XML schema: • **None**. • **OAuth1**. OAuth 1.0 protocol standard as per IETF specification. • **OAuth2**. OAuth 2.0 protocol standard as per IETF specification. • **SAML2**. SAML 2.0 protocol standard as per OASIS specification. • **OpenIdConnect**. OpenID Connect 1.0 protocol standard as per OpenID foundation specification. • **WsFed**. WS-Federation protocol standard as per OASIS specification. This protocol is NOT part of the features included in the public preview. For more information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[10]. • **WsTrust**. WS-Trust protocol standard as per OASIS specification. This protocol is NOT part of the features included in the public preview. For more information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[11]. • **Proprietary**. References for the above specifications are provided late in this document.
PartnerClaimType	True	Specify the claim type to use in this case

The *Mask* XML element in the above table contains the following attributes:

Attribute	Required	Description
Type	True	Specify the type of the claim mask. This attribute takes one of the following values as per *MaskTypeTYPE* enumeration in the custom policy XML schema: • **Simple**. A simple text mask that is applied to the leading portion of a string claim. • **Regex**. A regular expression that can be applied to the string claim as whole. If the latter value is specified, an optional eponym attribute must also be filled with the regular expression to use (see below).
Regex	False	If *Type* is set to **Regex**, specify the regular expression to use

[10] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

[11] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

The optional *Restriction* XML element of the above *ClaimType* section contains in turn the following attribute:

Attribute	Required	Description
MergeBehavior	False	Specify how the enumeration values will be merged together with any *ClaimType* present in a parent policy with the same identifier. This attribute takes one of the following values as per *MergeBehavior* enumeration in the custom policy XML schema: • **Append**. Specify that the collection of data present should be appended to the end of the collection specified in the parent policy. • **Prepend**. Specify that the collection of data present should be added before the collection specified in the parent policy. • **ReplaceAll**. Specify that the collection of data specified in the parent policy should be ignored, using instead the data specified in the current policy.

Along with the following XML elements:

XML element	Occurrences	Description
Enumeration	1:n	Define an available option for the user to select for a claim in the UI, such as a value in a dropdown.
Pattern	1:1	Specify the regular expression to use.

See subsection § *Specifying a restriction for the claim* hereafter for more information.

Finally, the optional *InputValidationReference* XML element of the above *ClaimType* section contains in turn the following attribute:

Attribute	Required	Description
Id	True	Specify the input validation rule to observe in order to be valid.

See subsections § *Specifying a restriction for the claim*

As covered above, claim type can be check against a specific restriction rule. For the purpose, the optional *Restriction* XML element of the above *ClaimType* section contains occurrence(s) of the *Enumeration* and *Pattern* XML elements.

The *Enumeration* XML element contains in turn one of the following attributes:

Attribute	Required	Description
Text	True	Specify the user-friendly display string that should be shown to the user in the UI for this option. Value: String
Value	True	Specify the claim value associated with selecting this option. Value: String
SelectByDefault	False	Indicating whether or not this option should be selected by default in the UI.

	Value: Boolean

The *Pattern* XML element contains in turn one of the following attributes:

Attribute	Required	Description
RegularExpression	True	Specify the regular expression that claims of this type must match to be valid.
HelpText	False	Describe the pattern/regular expression for this claim to the user.

The following XML snippet illustrates how to specify claim types based on the above definitions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimSchemas>
      …
      <ClaimType Id="email">
          <DisplayName>Email address</DisplayName>
          <DataType>string</DataType>
          <DefaultPartnerClaimTypes>
            <Protocol Name="OpenIdConnect" PartnerClaimType="email" />
          </DefaultPartnerClaimTypes>
          <AdminHelpText/>
          <UserHelpText>Email address that can be used to contact you.</UserHelpText>
          <UserInputType>TextBox</UserInputType>

          <Restriction>
            <Pattern RegularExpression="^[a-zA-Z0-9.!#$%&amp;&#8217;*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$"
                     HelpText="Please enter a valid email address" />
          </Restriction>

      </ClaimType>
      …
    </ClaimSchemas>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying an input validation for the claim and *Specifying the related predicates for the input validation* hereafter for more information.

The following XML snippet illustrates how to specify claim types based on the above definitions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimSchemas>
      …

      <ClaimType Id="phoneNumber">
          <DisplayName>Office Phone</DisplayName>
          <DataType>string</DataType>
          <Mask Type="Simple">XXX-XXX-</Mask>
          <AdminHelpText>User's telephone number</AdminHelpText>
          <UserHelpText>Your telephone number</UserHelpText>
      </ClaimType>

      …
    </ClaimSchemas>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying a restriction for the claim

As covered above, claim type can be check against a specific restriction rule. For the purpose, the optional *Restriction* XML element of the above *ClaimType* section contains occurrence(s) of the *Enumeration* and *Pattern* XML elements.

The *Enumeration* XML element contains in turn one of the following attributes:

Attribute	Required	Description
Text	True	Specify the user-friendly display string that should be shown to the user in the UI for this option. Value: String
Value	True	Specify the claim value associated with selecting this option. Value: String
SelectByDefault	False	Indicating whether or not this option should be selected by default in the UI. Value: Boolean

The *Pattern* XML element contains in turn one of the following attributes:

Attribute	Required	Description
RegularExpression	True	Specify the regular expression that claims of this type must match to be valid.
HelpText	False	Describe the pattern/regular expression for this claim to the user.

The following XML snippet illustrates how to specify claim types based on the above definitions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimSchemas>
      …
      <ClaimType Id="email">
          <DisplayName>Email address</DisplayName>
          <DataType>string</DataType>
          <DefaultPartnerClaimTypes>
            <Protocol Name="OpenIdConnect" PartnerClaimType="email" />
          </DefaultPartnerClaimTypes>
          <AdminHelpText/>
          <UserHelpText>Email address that can be used to contact you.</UserHelpText>
          <UserInputType>TextBox</UserInputType>

          <Restriction>
            <Pattern RegularExpression="^[a-zA-Z0-9.!#$%&amp;&#8217;*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$"
                     HelpText="Please enter a valid email address" />
          </Restriction>

      </ClaimType>
      …
    </ClaimSchemas>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying an input validation for the claim

As covered above, claim type can be check against input validation rule. The *InputValidations* section of the *BuildingBlocks* XML element allows to specify such rules. For that purpose, it contains the following XML element:

XML element	Occurrences	Description
InputValidation	1:n	Define a basic input validation.

Each *InputValidation* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular input validation rule and reference it from the *ClaimType* XML elements in the policy XML file.

and the following XML element:

XML element	Occurrences	Description
PredicateReferences	1:n	Allow to perform boolean aggregations against a series of predicate (similar to **and** and **or**). Each *PredicateReferences* must be true for the input validation rule to succeed.

The *PredicateReferences* section of the *InputValidation* XML element contains the following attributes:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular parameter and reference it from other XML elements in the policy XML file.
MatchAtLeast	True	Specify how many predicate reference checks must return true to consider the input validation successful. Type: Integer
HelpText	False	Override the error message defined in the *Predicate* XML elements that it references in the series of *PredicateReference*.

And the following XML element:

XML element	Occurrences	Description
PredicateReference	1:n	Specify a predicate to use for the input validation.

Each *PredicateReference* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify the predicate to use as part of the input validation.

The following XML snippet illustrates a typical *InputValidation* element.

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimSchemas>
      …
    </ClaimSchemas>
    …
    <InputValidations>
      …
      <InputValidation Id="PasswordValidation">
        <PredicateReferences Id="LengthGroup" MatchAtLeast="1">
          <PredicateReference Id="Length" />
        </PredicateReferences>
        <PredicateReferences Id="3of4" MatchAtLeast="3" HelpText="You must have at least 3 of the following character
                                                                  classes:">
          <PredicateReference Id="Lowercase" />
          <PredicateReference Id="Uppercase" />
          <PredicateReference Id="Number" />
          <PredicateReference Id="Symbol" />
        </PredicateReferences>
      </InputValidation>
      …
    </InputValidations>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

This XML element is in turn referenced in a *ClaimType* element to define which input validation rule is used to validate that claim.

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimSchemas>
      …
      <ClaimType Id="newPassword">
        <Restriction>
          <Pattern RegularExpression="^.*$" HelpText="" />
        </Restriction>

        <InputValidationReference Id="PasswordValidation" />

      </ClaimType>
      <ClaimType Id="reenterPassword">
        <Restriction>
          <Pattern RegularExpression="^.*$" HelpText="" />
        </Restriction>

        <InputValidationReference Id="PasswordValidation" />

      </ClaimType>
      …
    </ClaimSchemas>
    …
    <InputValidations>
      …
    </InputValidations>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying the related predicates for the input validation

Predicates allow to define a basic string validation to check against a claim type and that returns true or false.

The *Predicates* section of the *BuildingBlocks* XML element contains the following XML element:

XML element	Occurrences	Description
Predicate	1:n	Define a basic string validation.

Each *Predicate* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular predicate and reference it from the *InputValidation* XML elements in the policy XML file.
Method	True	Specify the method type. Value: one of the following types: • **IsLengthRange**. A simple value range for the length of the string claim as whole. • **MatchesRegex.** A regular expression that can be applied to the string claim as whole.
HelpText	False	Provide an error message for end users if the check fails. This string can be localized using the language customization feature[12].

and the following XML element:

XML element	Occurrences	Description
Parameters	1	Provide the parameters for the method type of the string validation.

The *Parameters* section of the *Predicates* XML element contains the following XML element:

XML element	Occurrences	Description
Parameter	1:n	Define a parameter for the string validation.

Each *Parameter* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular parameter and reference it from other XML elements in the policy XML file.

Both the number and value of the *Parameter* element(s) depends on the method type specified.

The following XML snippet illustrates a predicate with the **MatchesRegex** method, which is used to match a regular expression. In this snippet, it matches string that contains numbers.

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
          PolicySchemaVersion="0.3.0.0"
          …>
  …
  <BuildingBlocks>
    <ClaimsSchema>
      …
    </ClaimsSchema>
    <Predicates>
      …
```

[12] LANGUAGE CUSTOMIZATION IN AZURE ACTIVE DIRECTORY B2C: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-language-customization

```
      <Predicate Id="PIN" Method="MatchesRegex" HelpText="The password must be a pin.">
        <Parameters>
          <Parameter Id="RegularExpression">^[0-9]+$</Parameter>
        </Parameters>
      </Predicate>
```

```
      …
    </Predicates>
    <InputValidations>
      …
    </InputValidations>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

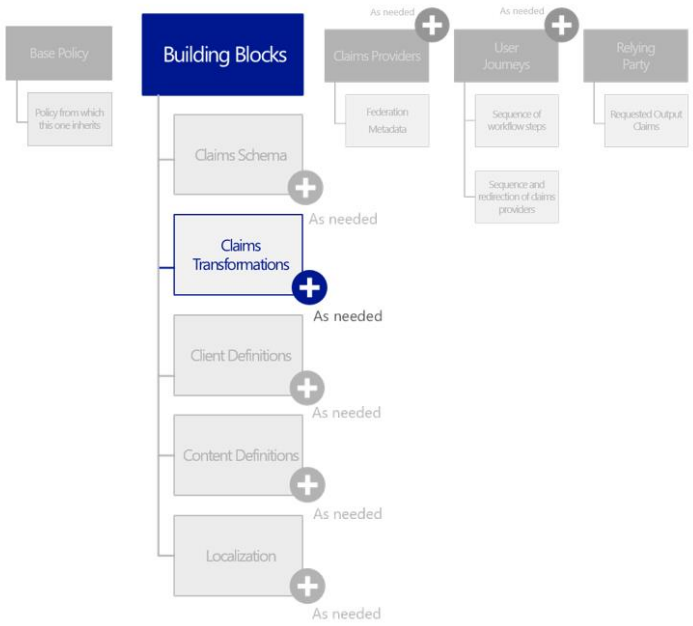
Next let's consider another XML snippet for a predicate with the **IsLengthRange** method. This method takes a minimum and maximum string length.

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
    <ClaimsSchema>
      …
    </ClaimsSchema>
    <Predicates>
      …
```

```
      <Predicate Id="Length" Method="IsLengthRange" HelpText="The password must be between 8 and 16 characters.">
        <Parameters>
          <Parameter Id="Minimum">8</Parameter>
          <Parameter Id="Maximum">16</Parameter>
        </Parameters>
      </Predicate>
```

```
      …
    </Predicates>
    <InputValidations>
      …
    </InputValidations>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying the claims transformation



As described above, the policy claims listing is fixed (at any given moment) in terms of the claims that can be provided to claims providers or relying parties as part of a user journey.

Individual claims providers may not issue claims using the same type or naming conventions. In such cases, the policy XML file must specify the claims transformation functions that will be performed to map issued claims to the schema. This mapping information needs to be recorded to ensure that Azure AD B2C performs the necessary transformations.

Furthermore, we expect that some policy may return derived attributes to comply with some privacy policies, such as age is in a certain category (e.g. above a given number of years) rather than divulging the user's actual birthdate. In these cases, the claims listing must advertise the derived attribute types, and the claims transformation functions must describe the mapping from the raw data delivered from the original issuer.

Some Trust Framework policies may also include attribute value substitution policies (such as mapping specific domain name suffixes to generic ones). The claims transformation functions must include the value that will result from such mappings.

Considering the above descriptions and needs, a claims transformation function essentially converts a given claim into another one. In other words, it can be used to modify existing *ClaimsSchema* claims or generate new ones.

To include the list of claims transformation functions that can be used in the user journeys, a *ClaimsTransformations* XML element must be declared under the *BuildingBlocks* section of the policy XML file. If a claims transformation function is not included in the policy, then it cannot be used in any user journey within that policy. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
ClaimsTransformation	1:n	Specify a supported claims transformation function that can be used within the community of interested. The claims transformation takes a set of claims, process them, and output another set of claims.

The *ClaimsTransformation* XML elements contain the following attributes:

Attribute	Required	Description
id	True	Specify a machine understandable identifier that is used to uniquely identify this particular claim transformation and reference it from other XML elements in the policy XML file.
TransformationMethod	True	Specify the suitable transform method as its name suggest. It's a machine understandable identifier to reference a published transform method to be used to define a claims transformation. Value: one of the supported transform methods such as: • **AddItemToStringCollection**. • **AddParameterToStringCollection**. • **AssertStringClaimsAreEqual**. • **ChangeCase**. • Etc. See below table for a complete list of the available transform methods.

And in turn the following XML elements:

XML element	Occurrences	Description
InputClaims	0:1	Specify an optional list of the claim types that are taken as input to the claims transformation. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section.
InputParameters	0:1	Specify an optional list of the parameters that are provided as input to the claims transformation. Each of these elements contains a value that is passed verbatim to the transformation.
OutputClaims	0:1	Specify an optional list of the claim types that are taken as output to the claims transformation. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section.

The above *InputClaims* XML elements contain the following XML element:

XML element	Occurrences	Description
InputClaim	0:n	Specify an expected claim type.

Each *InputClaim* XML element contains the following attributes:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file.

TransformationClaimType	True	Is set to "*InputClaim*".

Likewise, the *InputParameters* XML elements contain the following XML element:

XML element	Occurrences	Description
InputParameter	0:n	Specify an input parameter that are provided as input to the considered claims transformation.

Each *InputParameter* XML elements contain the following attributes:

Attribute	Required	Description
id	True	Specify an identifier that is a reference to a parameter of the considered transformation method as specified by the *TransformationMethod* attribute of the above *ClaimsTransformation* XML element.
DataType	True	Specify the type of data of the parameter, such as String, Boolean, Int or DateTime as per *DataType* enumeration in the custom policy XML schema. This type is used to perform arithmetic operations correctly.
Value	True	Specify a value that is passed verbatim to the transformation.

Eventually, the *OutputClaims* XML elements contain the following XML element:

XML element	Occurrences	Description
OutputClaim	0:n	Specify an expected claim type.

Each *OutputClaim* XML element contains the following attributes:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file.
TransformationClaimType	True	Is set to *"OutputClaim*".

As of this writing, the following table lists the supported transform methods along with their input/output claims and input parameters.

Transform method	Description	Type	InputClaim	InputParameter	OutputClaim
AddItemToStringCollection	Add the provided string claim to a that contains collection of strings	Claims	item (String) collection (StringCollection)		collection (StringCollection)
AddParameterToStringCollection	Add the provided string parameter to a claim that contains collection of strings	Claims	collection (StringCollection)	item	collection (StringCollection)
AssertBooleanClaimIsEqualToValue		Assert			
AssertDateTimeIsGreaterThan		Assert			
AssertStringClaimsAreEqual	Compare two claims, and throws an exception if they are not equal according to the specified comparison	Assert	inputClaim1 (String) inputClaim2 (String)	stringComparison [ordinal, ordinalIgnoreCase]	
ChangeCase	Change the case of the provided string claim to the one specified	Claims	inputClaim1 (String)	toCase [lower, upper]	outputClaim (String)
ClaimToClaimEquality		Claims			
CompareClaimToValue	Compare claim to the provided parameter, and returns a claim with true indicating that the values match, false otherwise	Claims	inputClaim1 (String)	operator [equal, not equal] compareTo ignoreCase (true, false)	outputClaim (Boolean)
CompareClaims	Compare two claims and returns a claim with true indicating that the claims match, false otherwise	Claims	inputClaim1 (String) inputClaim2 (String)	operator [equal, not equal] ignoreCase (true, false)	outputClaim (Boolean)
CreateAlternativeSecurityId	Create a JSON representation of the user's *alternativeSecurityId* property that can be used in calls to Graph API. This string is consumed by the Azure AD claims provider when *PartnerClaimType* is alternativeSecurityId	Claims	key (String) identityProvider (String)		alternativeSecurityId
CreateStringClaim	Create a string claim from the provided parameter in the policy	Claims		value (String)	createdClaim (String)
CreateRandomString	Create a random string using the random number generator used. If the random number generator is of type "integer", optionally a seed parameter and a maximum number may be	Claims		randomGeneratorType [guid, integer]	outputClaim (String)

	provided. An optional string format parameter allows the output to be formatted using it, and an optional base64 parameter specifies whether the output is base64 encoded			seed (Int32) maximumNumber (Int32) stringFormat (String) base64 [true, false]	
ConvertNumberToStringClaim		Claims			
DoesClaimExist		Claims			
Equality		Claims			
FormatStringClaim	Format a given claim according to the provided format string. This transformation uses the C# String.Format method. Please see its documentation for more details	Claims	inputClaim (String)	stringFormat (String)	outputClaim (String)
FormatStringMultipleClaims					
GetAgeGroupAndConsentProvided		Claims			
GetClaimFromJson	Given a JSON string of key value pairs, extract the specified claim	Claims	inputJson (String)	claimToExtract (String)	extractedClaim (String)
GetCurrentDateTime		Claims			
GetEmailFromJsonTransformation		Claims			
GetMappedValueFromLocalizedCollection		Claims			
GetSingleItemFromStringCollection	Get the first item from the provided string collection.	Claims	collection (StringCollection) - optional		extractedItem (String)
GetSingleValueFromJsonArray	Get the first item from the provided JSON string.	Claims	inputJsonClaim (String)		extractedClaim (String)
Hash	Hash the provided plain text using the salt and a secret whose identifier is provided as a parameter	Claims	plaintext (String) salt (String)	randomizerSecret (String)	hash (String)
IsTermsOfUseConsentRequired		Claims			
NullClaim	Replace the value of a claim with null	Claims	claim_to_null (String)		claim_to_null (String)
SetClaimsIfStringsMatch		Claims			

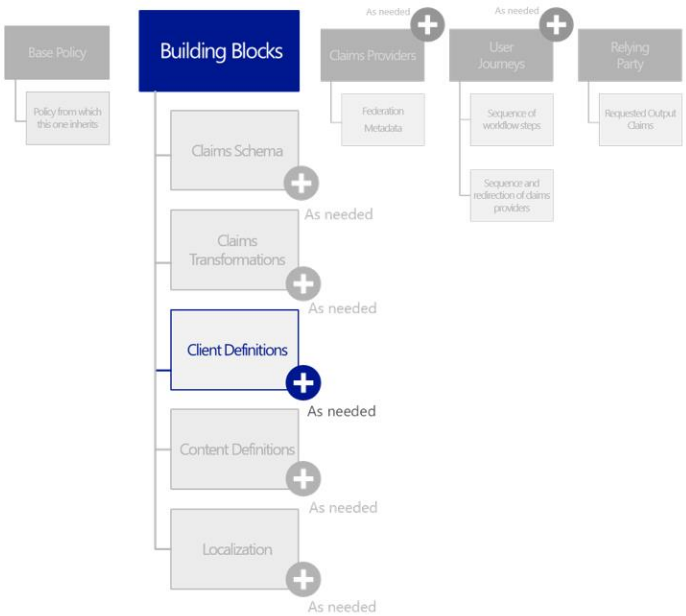
Considering the above, the following XML snippet illustrates how to define a list of applicable claims transformation functions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
          PolicySchemaVersion="0.3.0.0"
          …>
  …
  <BuildingBlocks>
    …
```

```
    <ClaimsTransformations>
      <ClaimsTransformation Id="CreateDisplayNameFromFirstNameAndLastName"
                            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>
      …
    </ClaimsTransformations>
```

```
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying the client definitions



Client definitions specify various properties specific to the end-user device for which the policy is being executed. **These definitions should be considered deprecated.**

To include the list of client definitions that can be used in the user journeys, a *ClientDefinitions* XML element must be declared under the *BuildingBlocks* section of the policy XML file. This element is optional.

It contains the following XML element:

XML element	Occurrences	Description
ClientDefinition	0:n	Define a client definition.

Each *ClientDefinition* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular client definition and reference it from other XML elements in the policy XML file.

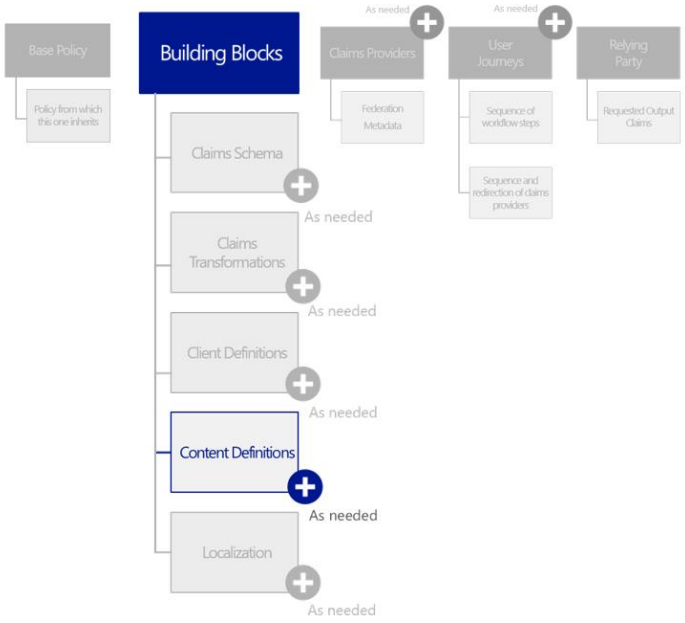
Along with the following XML element:

XML element	Occurrences	Description
ClientUIFilterFlags	0:1	Specify a comma-separated list of filter flags for the client UI. Type: String (enumeration: LineMarkers, MetaRefresh)

The following XML snippet illustrates how to define a list of client definitions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
          PolicySchemaVersion="0.3.0.0"
          …>
  …
  <BuildingBlocks>
    …

    <ClientDefinitions>
      <ClientDefinition Id="DefaultWeb">
        <ClientUIFilterFlags>LineMarkers, MetaRefresh</ClientUIFilterFlags>
      </ClientDefinition>
      …
    </ClientDefinitions>

    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Specifying the content definitions



Content definitions define the HTML and page template that are used for a given UI step, for example the claims providers' selection.

A parent policy can define the default look and feel via the URI to the HTML5 (it refers its CSS). A child policy can modify the look and feel by overriding the LoadUri for that HTML5 (see below).

As such, content definitions contain URLs to external content, for example, URLs to pages used in claims providers' selection.

Related external content is defined by crafting HTML5/CSS files (*.cshtml* files) as appropriate.

> **Note** For security reasons, the use of JavaScript is currently blocked for customization.

Azure AD B2C allows running code in your consumer's browser and uses the modern and standard approach Cross-Origin Resource Sharing (CORS)[13] to load content from a specific URL that you specify in a policy to point to the above HTML5/CSS files. You can specify different URLs for different pages thanks to content definitions.

> **Note** For more information, see the article AZURE ACTIVE DIRECTORY B2C: HOW TO CUSTOMIZE THE AZURE AD B2C USER INTERFACE (UI)[14].

To include the list of content definitions that can be used in the user journeys, a *ContentDefinitions* XML element must be declared under the *BuildingBlocks* section of the policy XML file. This element is optional.

[13] CROSS-ORIGIN RESOURCE SHARING W3C RECOMMENDATION 16 JANUARY 2014: http://www.w3.org/TR/cors/
[14] AZURE ACTIVE DIRECTORY B2C: HOW TO CUSTOMIZE THE AZURE AD B2C USER INTERFACE (UI): https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-ui-customization

It contains the following XML element:

XML element	Occurrences	Description
ContentDefinition	0:n	Define a content definition.

Each *ContentDefinition* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular content definition and reference it from other XML elements in the policy XML file. Value: one of the following supported type: • **api.error**. Error page • **api.idpselections.** Identity provider selection page • **api.idpselections.signup**. Identity provider selection for sign-up • **api.localaccountpasswordreset**. Forgot password page • **api.localaccountsignin**. Local account sign-in page • **api.localaccountsignup**. Local account sign-up page • **api.phonefactor**. Multi-factor authentication page • **api.selfasserted**. Social account sign-up page • **api.selfasserted.profileupdate**. Profile update page • **api.signuporsignin**. Unified sign-up or sign-in page • **api.signuporsigninwithkmsi**. Unified sign-up or sign-in page with 'Keep me signed in' (KMSI)

Along with the following XML elements:

XML element	Occurrences	Description
LoadUri	0:1	Specify the relative URL of the CSHTML page (i.e. an ASP.NET Razor Web page) or an HTML5/CSS page for the content definition. Type: String
RecoveryUri	0:1	Specify the relative URL of the HTML page for displaying an error relating to the content definition. Type: String
DataUri	0:1	Specify the relative URL of a XML file that provides user experience to invoke for the considered step. Type: String
Metadata	0:1	Specify the metadata utilized by the content definition. Type: collection of *Item* of key/value pairs.
LocalizedResourcesReferences	0:1	Specify what language resources to look for each language code.

The following metadata item keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
DisplayName	True	Specify the display name for the page in which the content definition will be rendered. Type: String

The *LocalizedResourcesReferences* XML element contains the following XML attribute:

Attribute	Required	Description
MergeBehavior	False	Specify the merge behavior with the content definition. Value: one of the following values as per *MergeBehavior* enumeration in the custom policy XML schema: • **Append**. Specify that the collection of data present should be appended to the end of the collection specified in the parent policy. • **Prepend**. Specify that the collection of data present should be added before the collection specified in the parent policy. • **ReplaceAll**. Specify that the collection of data specified in the parent policy should be ignored, using instead the data specified in the current policy.

As well as the following XML element:

XML element	Occurrences	Description
LocalizedResourcesReference	1:n	Specify what language resources to look for a specific language code.

Each *LocalizedResourcesReference* XML element contains in turn the following XML attributes:

Attribute	Required	Description
Language	True	Specify a culture for displaying content and that conforms to languages using language tag as per RFC 5646 TAGS FOR IDENTIFYING LANGUAGES[15]. Value: one of the following 36 supported language codes: • **bn**. Bangla • **cs**. Czech • **da**. Czech • **de**. German • **el**. Greek • **en**. English • **fi**. Finnish • **fr**. French • **gu**. Gujarati • **hi**. Hindi • **hr**. Croatian • **hu**. Hungarian

[15] RFC 5646 TAGS FOR IDENTIFYING LANGUAGES: http://tools.ietf.org/html/rfc5646

		• **it**. Italian • **ja**. Japanese • **kn**. Kannada • **ko**. Korean • **ml**. Malayalam • **mr**. Marathi • **ms**. Malay • **nb**. Norwegian Bokmal • **nl**. Dutch • **pa**. Punjabi • **pl**. Polish • **pt-br**. Portuguese - Brazil • **pt-pt**. Portuguese - Portugal • **ro**. Romanian • **ru**. Russian • **sk**. Slovak • **sv**. Swedish • **ta**. Tamil • **te**. Telegu • **th**. Thai • **tr**. Turkish • **zh-hans**. Chinese - Simplified • **zh-hant**. Chinese - Traditional
Url	False	Specify a search engine URL, for example "https://bing.com"
LocalizedResourcesReferenceId	True	Specify a machine understandable identifier that is used to uniquely identify a particular collection of localized resources, and reference it from the *Localization* XML element in the policy XML file. See next section § *Specifying the supported locales*. For more information, see article LANGUAGE CUSTOMIZATION IN CUSTOM POLICIES[16].

The following XML snippet illustrates how to define a list of content definitions:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
          PolicySchemaVersion="0.3.0.0"
          …>
  …
  <BuildingBlocks>
    …
```

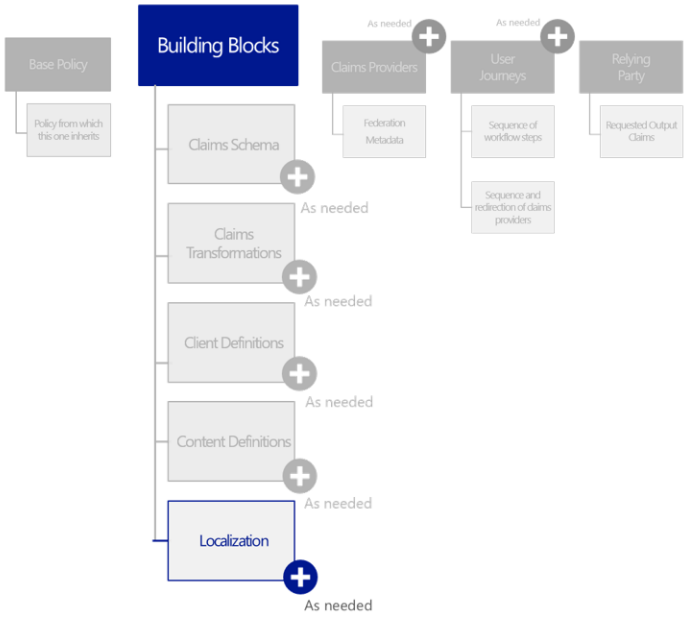
```
    <ContentDefinitions>
      <ContentDefinition Id="api.error">
        <LoadUri>~/tenant/default/exception.cshtml</LoadUri>
        <RecoveryUri>~/common/default_page_error.html</RecoveryUri>
        <DataUri>~/elements/exception/globalexception.xml</DataUri>
        <Metadata>
          <Item Key="DisplayName">Error page</Item>
        </Metadata>
      </ContentDefinition>
      <ContentDefinition Id="api.idpselections">
        <LoadUri>~/tenant/default/idpSelector.cshtml</LoadUri>
        <RecoveryUri>~/common/default_page_error.html</RecoveryUri>
        <DataUri>~/elements/homerealm/homerealm.xml</DataUri>
        <Metadata>
```

[16] LANGUAGE CUSTOMIZATION IN CUSTOM POLICIES: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-language-customization-custom

```
        <Item Key="DisplayName">Idp selection page</Item>
        <Item Key="language.intro">Sign in</Item>
      </Metadata>
    </ContentDefinition>
    <ContentDefinition Id="api.idpselections.signup">
      <LoadUri>~/tenant/default/idpSelector.cshtml</LoadUri>
      <RecoveryUri>~/common/default_page_error.html</RecoveryUri>
      <DataUri>~/elements/homerealm/homerealm.xml</DataUri>
      <Metadata>
        <Item Key="DisplayName">Idp selection page</Item>
        <Item Key="language.intro">Sign up</Item>
      </Metadata>
    </ContentDefinition>
    …
  <ContentDefinitions>
```

```
  …
 <BuildingBlocks>
 …
</TrustFrameworkPolicy>
```

Specifying the supported locales



Localization allow you to support multiple locales, i.e. languages, in the policy for the user journeys (see section § *Specifying the user journeys* later in this document).

As you already can infer from the above sections, policies control all elements of customer-facing content including text (strings).

The localization support in policies allows you to:

1. Setup the explicit list of the supported language in a policy and pick a default language.
2. Provide language-specific strings and collections.

Note For more information, see article LANGUAGE CUSTOMIZATION IN CUSTOM POLICIES[17].

For that purpose, a *Localization* XML element must be declared under the *BuildingBlocks* section of the policy XML file. This element is optional.

It contains the following attribute:

Attribute	Required	Description
Enabled	True	Specify if the localization is enabled on or not. Type: boolean Value: true or false.

Along with the following XML elements:

XML element	Occurrences	Description
SupportedLanguages	0:1	Define all the cultures that are supported by this policy
LocalizedResources	0:1	Contain all the translated strings for a specific language.

The *SupportedLanguages* XML element contains the contains the following attribute:

Attribute	Required	Description
DefaultLanguage	True	Specify the default language for the policy. Value: String that represents one of the 36 supported languages for displaying content (and that conforms to language tag as per RFC 5646 TAGS FOR IDENTIFYING LANGUAGES). the default language that the user will see in user journeys if any other supported culture is specified. Value: one of the following 36 supported language codes: • **bn**. Bangla • **cs**. Czech • **da**. Czech • **de**. German • **el**. Greek • **en**. English • **fi**. Finnish • **fr**. French • **gu**. Gujarati • **hi**. Hindi • **hr**. Croatian • **hu**. Hungarian • **it**. Italian • **ja**. Japanese • **kn**. Kannada • **ko**. Korean • **ml**. Malayalam

[17] LANGUAGE CUSTOMIZATION IN CUSTOM POLICIES: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-language-customization-custom

		• **mr**. Marathi • **ms**. Malay • **nb**. Norwegian Bokmal • **nl**. Dutch • **pa**. Punjabi • **pl**. Polish • **pt-br**. Portuguese - Brazil • **pt-pt**. Portuguese - Portugal • **ro**. Romanian • **ru**. Russian • **sk**. Slovak • **sv**. Swedish • **ta**. Tamil • **te**. Telegu • **th**. Thai • **tr**. Turkish • **zh-hans**. Chinese – Simplified • **zh-hant**. Chinese - Traditional
MergeBehavior	False	Specify the merge behavior. Value: one of the following values as per *MergeBehavior* enumeration in the custom policy XML schema: • **Append**. Specify that the collection of data present should be appended to the end of the collection specified in the parent policy. • **Prepend**. Specify that the collection of data present should be added before the collection specified in the parent policy. • **ReplaceAll**. Specify that the collection of data specified in the parent policy should be ignored, using instead the data specified in the current policy.

Along with the following XML elements:

XML element	Occurrences	Description
SupportedLanguage	0:n	Specify one supported language for the policy. Value: String that represents one of the 36 supported languages for displaying content (and that conforms to language tag as per RFC 5646 TAGS FOR IDENTIFYING LANGUAGES). See above *DefaultLanguage* attribute of the SupportedLanguages XML element.

If ui-locales isn't specified in the query string and the user's browser asks for one language, supported languages are shown to the user.

The *LocalizedResources* XML element contains the following attribute:

Attribute	Required	Description
Culture	True	Specify, if the *Id* attribute is not set, one supported language for the localized resources. Value: String that represents one of the 36 supported languages for displaying content (and that conforms to language tag as per RFC 5646 TAGS FOR IDENTIFYING LANGUAGES). See above *DefaultLanguage* attribute of the SupportedLanguages XML element.
Id	True	Specify, if the *Culture* attribute is not set, a machine understandable identifier that is used to uniquely identify a particular collection of localized resources and reference it from the *LocalizedResourcesReference* XML elements in the policy XML file. See section § *Specifying the content definitions*.

Along with the following XML elements:

XML element	Occurrences	Description
LocalizedCollections	0:1	Allow defining entire collections in various languages. A collection can have different number of items, and different strings for various languages. Examples of collections include the enumerations that appear in claim types, e.g. country/region list, and are shown to the user in a drop-down list.
LocalizedStrings	0:1	Define all the strings, except those that appear in collections, in various cultures.

The following XML snippet illustrates how to define the localization support for both English (default) and French:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
   …
   <BuildingBlocks>
      …
      <ContentDefinitions>
         …
         <ContentDefinition Id="api.signuporsignin">
            <LoadUri>~/tenant/default/unified.cshtml</LoadUri>
            <RecoveryUri>~/common/default_page_error.html</RecoveryUri>
            <DataUri>urn:com:microsoft:aad:b2c:elements:unifiedssp:1.0.0</DataUri>
            <Metadata>
               <Item Key="DisplayName">Signin and Signup</Item>
            </Metadata>

            <LocalizedResourcesReferences>
               <LocalizedResourcesReference Language="en" LocalizedResourcesReferenceId="api.signuporsignin.en" />
               <LocalizedResourcesReference Language="fr" LocalizedResourcesReferenceId="api.signuporsignin.fr" />
            </LocalizedResourcesReferences>

         </ContentDefinition>
         …
      <ContentDefinitions>
      …
```

```
    <Localization Enabled="true">
      <SupportedLanguages DefaultLanguage="en">
        <SupportedLanguage>en</SupportedLanguage>
        <SupportedLanguage>fr</SupportedLanguage>
      </SupportedLanguages>
      <LocalizedResources Culture="en" Id="api.signuporsignin.en" >
        …
      </LocalizedResources>
      <LocalizedResources Culture="fr" Id="api.signuporsignin.fr">
        …
      </LocalizedResources>
    </Localization>
```

```
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

As illustrated above, creating localized resources for content definitions is a two steps process:

1. Specifying in the resource ID for the desired languages as per above *LocalizedResourcesReference* XML elements. See previous section § *Specifying the content definitions*.
2. Creating *LocalizedResources* XML elements with corresponding IDs in the building blocks.

The target content definition is in this sample the unified Sign-Up or Sign-In (SUSI) page.

Now that we have depicted the overall structure for localization support, let's consider in more details the inner XML elements.

Each *LocalizedCollection* XML element under the *LocalizedCollections* XML element contains the following attributes:

Attribute	Required	Description
ElementType	True	Define the type of the element in the policy XML file. Type: String (enumeration) Value: **ClaimType**.
ElementId	True	Contain a reference to a claim type already defined in the *ClaimsSchema* section. Value: String
TargetCollection	True	Specify as its name indicates the target collection. Value: String
Override	False	Specify if the value of the element is to override. Type: Boolean Value: true or false

Along with the following XML elements:

XML element	Occurrences	Description
Item	0:n	Define an available option for the user to select for a claim in the UI, such as a value in a dropdown.

In turn, the *Item* XML element contains the following attributes:

Attribute	Required	Description
Text	True	Specify the user-friendly display string that should be shown to the user in the UI for this option. Value: String
Value	True	Specify the claim value associated with selecting this option. This is the what is returned in the claim when this option is selected. Value: String
SelectByDefault	False	Indicate whether or not this option should be selected by default in the UI. Value: Boolean

This XML element allows providing language specific claims listing, as an illustration for the *country* claim type:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
     …
     <Localization Enabled="true">
        <SupportedLanguages DefaultLanguage="en">
           …
        </SupportedLanguages>
        <LocalizedResources Culture="en">
           <LocalizedCollection ElementType="ClaimType" ElementId="country" TargetCollection="Restriction">
              …
              <Item Text="France" Value="France" />
              <Item Text="French Guiana" Value="French Guiana" />
              <Item Text="French Polynesia" Value="French Polynesia" />
              <Item Text="French Southern Territories" Value="French Southern Territories" />
              <Item Text="U.S. Outlying Islands" Value="U.S. Outlying Islands" />
              <Item Text="U.S. Virgin Islands" Value="U.S. Virgin Islands" />
              <Item Text="United Kingdom" Value="United Kingdom" />
              <Item Text="United States" Value="United States" />
              …
           </LocalizedCollection>
           …
           <LocalizedStrings>
           …
           </LocalizedStrings>
        </LocalizedResources>
        …
     </Localization>
     …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

Similarly, the *LocalizedStrings* XML element contains the following XML elements:

XML element	Occurrences	Description
LocalizedString	0:n	Specify one localized string as its name indicates.

In turn, the *LocalizedString* XML element contains the following attributes:

Attribute	Required	Description
ElementType	True	Define the type of the element in the policy XML file. Type: String (enumeration) Value: one the following supported data types: • **ClaimsProvider**. Labels for your identity providers (Facebook, Google, Azure AD, etc.) • **ClaimType**. Labels for your attributes and their corresponding help text, or field validation errors. • **UxElement**. Other string elements on the page that are there by default, such as buttons, links, or text. • **ErrorMessage**. Form validation error messages.
ElementId	True	If *ElementType* is set to **ClaimType**, contains a reference to a claim type already defined in the *ClaimsSchema* section.
StringId	True	If *ElementType* is set to: • **ClaimsProvider**, contains the *TargetClaimsExchangeId* attribute of a *ClaimsProviderSelections* XML element in the policy. • **ClaimType**, specify the attribute of a particular claim type. Value: one the following attributes as per *ClaimType* XML element definition in the custom policy XML schema: • **DisplayName**. • **AdminHelpText**. • **UserHelpText**. • **UxElement**, specify the name of the particular labels. • **ErrorMessage**, specify the key of the particular error message in the metadata of a *TechnicalProfile* XML element. Value: one of the followings: • **UserMessageIfClaimsPrincipalDoesNotExist**. • **UserMessageIfInvalidPassword**. • **UserMessageIfOldPasswordUsed**.
Override	False	Specify if the value of the element is to override. Type: Boolean Value: true or false

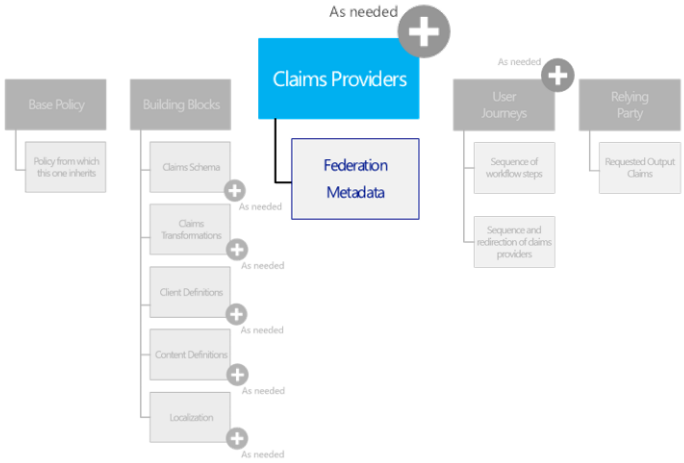
The value of this XML element corresponds the actual localized string.

This XML element allows providing language specific claims listing, as an illustration for the *password* claim type:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <BuildingBlocks>
     …
     <Localization Enabled="true">
        <SupportedLanguages DefaultLanguage="en">
           …
        </SupportedLanguages>
        <LocalizedResources Culture="en">
           …
        </LocalizedResources>
        <LocalizedStrings>

          <LocalizedString ElementType="ClaimType" ElementId="password" StringId="DisplayName">
            Password
          </LocalizedString>
          <LocalizedString ElementType="ClaimType" ElementId="password" StringId="AdminHelpText">
            Enter password
          </LocalizedString>
          <LocalizedString ElementType="ClaimType" ElementId="password" StringId="UserHelpText">
            Enter password
          </LocalizedString>

           …
       </LocalizedStrings>
     </LocalizedResources>
     …
    </Localization>
    …
  <BuildingBlocks>
  …
</TrustFrameworkPolicy>
```

In the above illustration, "*Password*" and "*Enter password*" are the actual string that can be set in accordance to the specified culture for the *password* claim type for respectively the *DisplayName* attribute value and the *AdminHelpText* and *UserHelpText* attributes' value.

Specifying the claims providers



Identity providers, attribute providers, attribute verifiers, directory provider, MFA provider, self-asserted attribute provider, etc. are all modelled as claims providers.

To include a list of claims providers along with their technical profiles that may be used in the various user journeys, a *ClaimsProviders* XML element must be declared under the above top-level *TrustFrameworkPolicy* XML element of the policy XML file. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
ClaimsProvider	1:n	Declare an accredited claims provider that can be leveraged within the community of interested in the various user journeys.

The *ClaimsProvider* XML elements represents a claims provider, along with its supported technical profiles. Such an element contains the following XML elements:

XML element	Occurrences	Description
Domain	0:1	Specify The human understandable domain name for the claim provider. Type: String
DisplayName	0:1	Specify the human understandable name of the claims provider that can be displayed to the users. Type: String
TechnicalProfiles	1:1	Specify a list of technical profiles supported by the claim provider. Every claims provider must have one or more technical profiles which determines the end points and the protocols needed to communicate with that claims provider. In fact, in Azure AD B2C, it is the technical profile that is referenced elsewhere for communication with a particular claims provider. A claims provider can have multiple technical profiles for various reasons. For example, multiple technical profiles may be defined because the considered claims provider supports multiple protocols, various

endpoints with different capabilities, or releases different claims at different assurance levels. It may be acceptable to release sensitive claims in one user journey, but not in another one.

See next section(s).

The following XML snippet illustrates how to define a claim provider:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …

  <ClaimsProviders>
      <ClaimsProvider>
        <DisplayName>Some Claims Provider</DisplayName>
        <TechnicalProfiles>
          <TechnicalProfile …>
            …
          </TechnicalProfile>
          …
        </TechnicalProfiles>
      </ClaimsProvider>
      …
  </ClaimsProviders>

  …
</TrustFrameworkPolicy>
```

The subsections describe how to define a valid technical profile via the *TechnicalProfile* XML element that specifically represents a given technical profile supported by a claims provider.

Specifying technical profile(s) for a given claims provider

As per the aforementioned OIX model, a technical profile consists in a set of constraints on the use of a specific technology for the exchange of digital identity information to ensure interoperability and maintain compliance with required LOA (and/or LOP).

A policy XML file uses technical profiles for two purposes:

1. Technical details are published as connection metadata to ensure on-the-wire interoperability between participants in the community
2. Technical profiles must be tagged with unique names that are used to “label” the interface(s) for which claims providers have been certified for conformance to LOA requirements.

Azure AD B2C mediates connections between relying parties and claims providers. As part of the Trust Framework policy, technical profiles provide a contract for Azure AD B2C to contact claims providers. A claims provider can have multiple technical profiles, each technical profile defining a specific contract for federating with that specific provider.

For that purpose, it notably consists of:

1. Protocol – SAML 2.0, WS-Federation/WS-Trust, OpenID Connect (OIDC), OAuth 2.0, etc.

   (The WS-Federation/WS-Trust protocols are NOT part of the features included in the public preview. For more information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[18].)

2. Metadata
3. Cryptographic references – i.e. reference to a secret in the B2C tenant secret store.
4. Input/persited/output claims:
   a. Mentions the list of the claims provider can expect in request/persist/respond with.
   b. And their mapping from the *PartnerClaimType* to the claim's used in the policy XML file.
5. Input/output claims transformations:
   a. Reference to the *ClaimsTransformation*'s defined in the policy XML file.
   b. Input transforms are run prior to sending the request to the provider and output transforms are run on receiving the response from the provider.

In addition, a technical profile is usually certified for a LOA and thus one claims provider may have multiple technical profiles for different LOAs. As described later in this document for the relying party, technical profiles similarly provide a contract for relying parties to contact Azure AD B2C services.

The *TechnicalProfiles* section of the above *ClaimsProviders* XML element contains the following XML element:

XML element	Occurrences	Description
TechnicalProfile	0:n	Define a technical profile supported by the claim provider.

Each *TechnicalProfile* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular technical profile and reference it from other XML elements in the policy XML file, for example *OrchestrationSteps* and *InputTokenSources*.

Along with the following XML elements:

XML element	Occurrences	Description
Domain	0:1	Specify the he human understandable domain name for the technical profile. Type: String
DisplayName	0:1	Specify the human understandable name of the technical profile that can be displayed to the users. Type: String

[18] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

Description	0:1	Specify a human understandable description of the technical profile that can be displayed to the users. Type: String
Protocol	0:1	Specify the protocol used for the federation.
InputTokenFormat	0:1	Specify the format of the input token. Type: String (enumeration) Value: one of the types as per *TokenFormat* enumeration in the custom policy XML schema: • **JSON**. • **JWT**. JSON Web Token as per IETF specification. • **SAML11**. SAML 1.1 security token as per OASIS specification. • **SAML2**. SAML 2.0 security token as per OASIS specification. • **CpimUnsigned**.
OutputTokenFormat	0:1	Specify the format of the output token. Type: String (enumeration) Value: one of the types as per *TokenFormat* enumeration in the custom policy XML schema. See above.
AssuranceLevelOfOutputClaims	0:1	List the assurance level of the claims that are retrieved from the technical profile. Type: String
RequiredAssuranceLevelsOfInputClaims	0:1	Lists the assurance levels that a claim must have for it to be used as an input claim to the technical profile. Type: String
SubjectAuthenticationRequirements	0:1	Specify the requirements regarding the conscious and active participation of the subject in authentication.
Metadata	0:1	Specify the metadata utilized by the protocol for communicating with the endpoint in the course of a transaction to plumb "on the wire" interoperability between Azure AD B2C and other community participants. Type: collection of *Item* of key/value pairs.
CryptographicKeys	0:1	Specify a list of cryptographic keys used in this technical profile. For more information, see section § *Managing your key containers for Trust Framework (policies)* in the third document of this series.
Suppressions	0:1	Specify a list of suppressions supported by the protocol. Type: String
PreferredBinding	0:1	If the protocol supports multiple bindings, represent binding preferred by the protocol, for example HTTP POST or HTTP GET in the case of SAML 2.0 Type: String
InputTokenSources	0:1	Represent the list of technical profiles of the claims providers from which the original tokens are to be sent. Azure AD B2C can indeed send the original token from one claims provider to another claims provider.

InputClaimsTransformations	0:1	Specify an optional list of references to claims transformations that should be executed before any claims are sent to the claims provider or the relying party. Each of these elements contains reference to a *ClaimsTransformation* already defined in the *ClaimsTransformations* section.
InputClaims	0:1	Specify an optional list of the claim types that are taken as input in the technical profile. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section.
PersistedClaims	0:1	Specify an optional list of the claim types that are persisted by the claims provider that relates to the technical profile. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section.
OutputClaims	0:1	Specify an optional list of the claim types that are taken as output in the technical profile. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section.
OutputClaimsTransformations	0:1	Specify an optional list of references to claims transformations that should be executed after claims are received from the claims provider. Each of these elements contains reference to a *ClaimsTransformation* already defined in the *ClaimsTransformations* section.
ValidationTechnicalProfiles	0:n	Specify a list of other technical profiles that the technical profile uses for validation purposes.
SubjectNamingInfo	0:1	Control the production of the subject name in tokens (e.g. SAML) where subject name is specified separately from claims.
SubjectAuthenticationRequirements	0:1	Specify the requirements regarding the conscious and active participation of the subject in authentication
Extensions	0:1	Allow any xml from any namespace outside of the document namespaces to be included in the element. This represents an extension point for elements as its name indicates.
IncludeClaimsFromTechnicalProfile	0:1	Indicate a machine understandable identifier that is used to uniquely identify a different technical profile. All input and output claims from referenced technical profile will be added to this technical profile. Referenced technical profile must be defined in the same policy XML file. Type: String
IncludeTechnicalProfile	0:1	Indicate a machine understandable identifier that is used to uniquely identify a different technical profile. All data from referenced technical profile will be added to this technical profile. Referenced technical profile must exist in the same policy XML file.
UseTechnicalProfileForSessionManagement	0:1	Indicate a machine understandable identifier to uniquely identify a different technical profile to be used for session management.
EnableForUserJourney	0:1	Specify if the technical user profile is enable for a user journey. if the technical profile should be used within a user journey, this includes *ClaimProviderSelections*. If this value is set to **true**, it will disable the selection. Value: one of the following values as per *EnabledForUserJourneysValues* enumeration in the custom policy XML schema: • **true**. Execute the technical profile.

	• **false**. Always skip the technical profile. • **OnClaimsExistence**. Only execute the technical profile if the claim specified in the technical profile's metadata is present in the user journey storage. • **Always**. (default) Execute the technical profile. • **Never**. Always skip the technical profile.

The *Protocol* XML element in the above table contains the following attributes:

Attribute	Required	Description
Name	True	Specify the name of a valid protocol supported by Azure AD B2C that is used as part of the technical profile. Type: String (enumeration) Value: one of the following types as per *ProtocolName* enumeration in the custom policy XML schema: • **None**. • **OAuth1**. OAuth 1.0 protocol standard as per IETF specification. • **OAuth2**. OAuth 2.0 protocol standard as per IETF specification. • **SAML2**. SAML 2.0 protocol standard as per OASIS specification. • **OpenIdConnect**. OpenID Connect 1.0 protocol standard as per OpenID foundation specification. • **WsFed**. WS-Federation (WS-Fed) 1.2 protocol standard as per OASIS specification. • **WsTrust**. WS-Trust 1.3 protocol standard as per OASIS specification. • **Proprietary**. For a RESTful based provider.
Handler	False	Specify the fully-qualified name of the assembly that will be used by Azure AD B2C to determine the protocol handler if the protocol name is set to "Proprietary". **It is invalid to provide this attribute with any other protocol name.** Type: String

The *RequiredAssuranceLevelsOfInputClaims* XML element in the above table contains the following XML element:

XML element	Occurrences	Description
RequiredAssuranceLevelOfInputClaims	0:n	List an assurance level that a claim must have in order for it to be used as an input claim to the technical profile.

The *SubjectAuthenticationRequirements* XML element in the above table contains the following XML attributes:

Attribute	Required	Description
TimeToLive	True	Specify The maximum number of minutes cached credentials can be used following an active authentication by the subject. Type: Integer

ResetExpiryWhenTokenIssued	False	If **true** then whenever a token is issued (even using a cached credential), set the expiry time to the current time plus the TimeToLive value. Default is **false**. Type: Boolean

The *CryptographicKeys* XML element in the above table contains the following XML element:

XML element	Occurrences	Description
Key	0:n	Define a cryptographic key used in this technical profile.

Each *Key* XML element contains in turn the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular key (pair), and reference it from other XML elements in the policy XML file. Type: String
StorageReferenceId	True	Specify a machine understandable identifier that is used to uniquely identify a particular storage key container, and reference it from other XML elements in the policy XML file. For more information, see section § *Managing your key containers for Trust Framework (policies)* in third document of this series. Type: String

The *InputTokenSources* XML elements contain the following XML element:

XML element	Occurrences	Description
TechnicalProfile	1:n	Specify a source for that can be the input assertions for the current technical profile.

Each *TechnicalProfile* XML element contains the following attribute:

Attribute	Required	Description
Id	True	Specify a reference to a machine understandable identifier that is used to uniquely identify a particular technical profile in the policy XML file. Type: String

Each *InputClaim* XML element contains the following attribute:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String

The *InputClaimsTransformations* XML elements contain the following XML element:

XML element	Occurrences	Description
InputClaimsTransformation	0:n	Specify a claims transformation that should be executed before any claims are sent to the claims provider or the relying party. A claims transformation can indeed be used to modify existing *ClaimsSchema* claims or generate new ones.

Each *InputClaimsTransformation* XML element contains the following attribute:

Attribute	Required	Description
ReferenceId	True	Specify a reference to a *ClaimsTransformation* already defined in the *ClaimsTransformations* section in the policy XML file. Type: String

The *InputClaims* XML elements contain the following XML element:

XML element	Occurrences	Description
InputClaim	0:n	Specify an expected claim type.

Each *InputClaim* XML element contains the following attribute:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String

the *PersistedClaims* XML elements contain the following XML element:

XML element	Occurrences	Description
PersistedClaim	0:n	Specify an expected claim type. Claim mappings are used to determine the provider claim type before sending to the claims provider.

Each *PersistedClaim* XML element contains the following attributes:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String
DefaultValue	False	Specify a default value to create a claim if the claim indicated by *ClaimTypeReferenceId* does not exist so that the resulting claim can be used as an *InputClaim* by the technical profile. Type: String
PartnerClaimType	False	Identify the *ClaimType* of the external partner that the specified policy claim type maps to. If the *PartnerClaimType* attribute is not specified,

		then the specified policy claim type is mapped to the partner claim type of the same name. Type: String
OverwriteIfExists	False	Provides an optional property to the claims provider indicating whether the claim can be overwritten in the claims provider records if the claim provider supports overwriting. Type: true or false

Eventually, the *OutputClaims* XML elements contain the following XML element:

XML element	Occurrences	Description
OutputClaim	0:n	Specify an expected claim type.

Each *OutputClaim* XML element contains the following attributes:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String
DefaultValue	False	Specify a default value if not set. Type: String
PartnerClaimType	False	Specify the partner claim type. Type: String
Required	False	Specify this claim is required. Type: String

The *OutputClaimsTransformations* XML elements contain the following XML element:

XML element	Occurrences	Description
OutputClaimsTransformation	0:n	Specify a claims transformation that should be executed after claims are received from the claims provider. A claims transformation can indeed be used to modify existing *ClaimsSchema* claims or generate new ones.

Each *OutputClaimsTransformation* XML element contains the following attribute:

Attribute	Required	Description
ReferenceId	True	Specify a reference to a *ClaimsTransformation* already defined in the *ClaimsTransformations* section in the policy XML file. Type: String

The *ValidationTechnicalProfiles* XML element in the above table contains the following XML element:

XML element	Occurrences	Description
ValidationTechnicalProfile	1:n	Define a technical profile to be used for validating some or all of the output claims of the referencing technical profile. Therefore, all the input

	claims of the referenced technical profile must appear in the output claims of the referencing technical profile.

ValidationTechnicalProfile and *IncludeTechnicalProfile* XML elements contain in turn the following attributes:

Attribute	Required	Description
ReferenceId	True	Specify a reference to a machine understandable identifier that is used to uniquely identify a particular technical profile already defined in the policy XML file.

The *SubjectNamingInfo* XML element in the above table contains the following attributes:

Attribute	Required	Description
ClaimType	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String
NameQualifier	False	Type: String
SPNameQualifier	False	Type: String
Format	False	Type: String
SPProvidedID	False	Type: String

IncludeTechnicalProfile and *UseTechnicalProfileForSessionManagement* XML elements in the above table contain the following attribute:

Attribute	Required	Description
ReferenceId	True	Specify a reference to a machine understandable identifier that is used to uniquely identify a particular technical profile already defined in the policy XML file.

The following XML snippet illustrates how to define a technical profile for a claim provider:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <ClaimsProviders>
    <ClaimsProvider>
      <DisplayName>Some Claims Provider</DisplayName>
      <TechnicalProfiles>

        <TechnicalProfile …>
          …
        </TechnicalProfile>
        …

      </TechnicalProfiles>
    </ClaimsProvider>
    …
  </ClaimsProviders>
  …
</TrustFrameworkPolicy>
```

The next sections further detail how to specify adequate technical profiles depending on the nature of the claims provider. It more particularly describes the related metadata information to specify.

Specifying a technical profile for an Azure AD claims provider

This section outlines the specifics of a technical profile for interacting with an Azure AD provider.

A suitable technical profile definition for an Azure AD provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"Web.TPEngine.Providers.AzureActiveDirectoryProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"

The following metadata item key may or must be present in the *Metadata* XML element:

Item key	Required	Description
Operation	True	Determine the operation to be performed. Value: one of the following types: • **Read**. • **Write**. • **DeleteClaims**. • **DeleteClaimsPrincipal**.
DeleteAllPersistedClaims	False	Apply when *Operation* is set to **DeleteClaims**. Value: true or false. If true, delete all claims specified in the *PersistedClaims* XML element in the policy. If false, only delete those claims that are passed at run-time.
ClaimsPersistenceTenant	False	If the B2C tenant is different from the relying party, determine in which tenant is the user object created and claims stored. Value: one of the followings: • **{TRUSTFRAMEWORK}**. • **{RELYINGPARTY}**. Curly brackets are part of the value.
CreateClaimsPrincipalIfItDoesNotExist	False	Apply when *Operation* is set to **Write**. Value: true or false. If true, then a new user object is created if it does not exist. If false, then the user object is not created and the claims may not be written if the user object does not exist.
RaiseErrorIfClaimsPrincipalDoesNotExist	False	Enable to raise an error. Value: true or false. If true, raises an error if the user object does not exist in the directory.

UserMessageIfClaimsPrincipalDoesNotExist	False	If an error is to be raised (see previous key setting), specify the message to show to the user if user object does not exist. Value: String
RaiseErrorIfClaimsPrincipalAlreadyExists	False	Apply when *Operation* is set to **Write**. Value: true or false. If true, raises an error if the user object already exists.
UserMessageIfClaimsPrincipalAlreadyExists		If an error is to be raised (see previous key setting), specify the message to show to the user if user object already exists. Value: String
ApplicationObjectId	False	Specify the application object identifier for extension attributes. Value: ObjectId of an application in the tenant
ClientId	False	Specify the client id for accessing the tenant as third party. Value: ClientId of an application in the tenant
Key	False	Specify the key for accessing the tenant as third party. Value: Key of an application in the tenant
SchemaExtensionTenant		If the B2C tenant is different from the relying party, determine in which tenant are the extension attributes created. Value: one of the followings: • **{TRUSTFRAMEWORK}**. • **{RELYINGPARTY}**. Curly brackets are part of the value.
ConvertEmptyClaimsToNull	False	If an empty string is encountered as a claim value, convert it to null. Value: true or false (default: true)
ProvideDefaultValueForDisplayName	False	If an empty or null value is encountered for *displayName*, replace it with "unknown". Value: true or false (default: true)
DefaultClaimValueForDisplayName		Use the specified value instead of "unknown" for *displayName* if an empty or null value is encountered. (see previous key setting) Value: any valid *displayName* string

The *InputClaimsTransformations* XML element may contain a collection of *InputClaimsTransformation* that should be executed before any claims are sent to the Azure AD provider. It can thus generate new input claims.

The *InputClaims* XML element contains the claims bag as the input with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that Azure AD claim type.

The *PersistedClaims* XML element may contain the claims bag as all the values that should be persisted by Azure AD with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that Azure AD claim type.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that Azure AD claim type.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The following XML snippet illustrates a series of technical profiles (with includes) for an Azure AD provider:

```
<TechnicalProfile Id="AAD-Common">
   <DisplayName>Azure Active Directory</DisplayName>
   <Protocol Name="Proprietary"
             Handler="Web.TPEngine.Providers.AzureActiveDirectoryProvider, Web.TPEngine, Version=1.0.0.0,
                      Culture=neutral, PublicKeyToken=null" />
</TechnicalProfile>
<TechnicalProfile Id="AAD-WriteCommon">
   <Metadata>
     <Item Key="Operation">Write</Item>
     <Item Key="RaiseErrorIfClaimsPrincipalAlreadyExists">true</Item>
   </Metadata>
   <PersistedClaims>
      <PersistedClaim ClaimTypeReferenceId="displayName" DefaultValue="unknown" />
      <PersistedClaim ClaimTypeReferenceId="passwordPolicies" DefaultValue="DisablePasswordExpiration" />
      <PersistedClaim ClaimTypeReferenceId="Verified.strongAuthenticationPhoneNumber"
                      PartnerClaimType="strongAuthenticationPhoneNumber" />
      <PersistedClaim ClaimTypeReferenceId="strongAuthenticationEmailAddress" />
   </PersistedClaims>
   <OutputClaimsTransformations>
      <OutputClaimsTransformation ReferenceId="CreateEmailsFromOtherMailsAndSignInNamesInfo" />
      <OutputClaimsTransformation ReferenceId="AddStrongAuthenticationEmailToEmails" />
   </OutputClaimsTransformations>
   <IncludeTechnicalProfile ReferenceId="AAD-Common" />
</TechnicalProfile>
<TechnicalProfile Id="AAD-UserWriteUsingAlternativeSecurityId">
   <Metadata>
      <Item Key="UserMessageIfClaimsPrincipalAlreadyExists">You are already registered, please press the back button and
               sign in instead.</Item>
   </Metadata>
   <InputClaimsTransformations>
      <InputClaimsTransformation ReferenceId="CreateOtherMailsFromEmail" />
   </InputClaimsTransformations>
   <InputClaims>
      <InputClaim ClaimTypeReferenceId="AlternativeSecurityId" PartnerClaimType="alternativeSecurityId"
                  Required="true" />
   </InputClaims>
   <PersistedClaims>
      <PersistedClaim ClaimTypeReferenceId="alternativeSecurityId" />
      <PersistedClaim ClaimTypeReferenceId="userPrincipalName" />
      <PersistedClaim ClaimTypeReferenceId="otherMails" />
      <PersistedClaim ClaimTypeReferenceId="mailNickName" DefaultValue="unknown" />
   </PersistedClaims>
   <OutputClaims>
      <OutputClaim ClaimTypeReferenceId="objectId" />
      <OutputClaim ClaimTypeReferenceId="newUser" PartnerClaimType="newClaimsPrincipalCreated" />
      <OutputClaim ClaimTypeReferenceId="otherMails" />
   </OutputClaims>
   <IncludeTechnicalProfile ReferenceId="AAD-WriteCommon" />
</TechnicalProfile>
```

Implementation notes on the technical profile

Fulfilling the requirements to perform an operation

Unless the technical profile performs a **Read** operation, there must be exactly one *InputClaim* XML element in the claims bag. If the *Operation* is **Write**, then it must also be appearing in a *PersistedClaims* XML element (see below).

If the technical profile performs **Write** or **DeleteClaims** operations, then it must also have an *PersistedClaims* XML element. The Identity Experience Framework in Azure AD B2C passes all the values as-is to the underlying tenant. This gives maximum control at the policy level.

The following table lists the required claims in the *PersistedClaims* XML element for a technical profile that will performs a **Write** operation to create a user (i.e. the metadata key *CreateClaimsPrincipalIfItDoesNotExist* is set to true).

Type of account	*InputClaims*	Required claims in *PersistedClaims*
Social IDP-based	*alternativeSecurityId*	*alternativeSecurityId*, *userPrincipalName*, *displayName*, *mailNickname*
Name coexistence account	*logonIdentifier.emailAddress* or *logonIdentifier.username*	*displayName*, *mailNickname* and either *logonIdentifier.emailAddress* or *logonIdentifier.username* (whichever is in the *InputClaims*)
Regular account	*userPrincipalName*	*userPrincipalName*, *displayName*, *mailNickname*

Note See section User Entity[19] in the article ENTITY AND COMPLEX TYPE REFERENCE | GRAPH API REFERENCE for all the properties that can be provided when a user is created. The required fields listed on the article are trumped by the required fields in the above table. This is because the article only considers regular accounts since other types of accounts are internal only.

Note When a value is provided for certain properties, it is passed to the directory by Azure AD B2C even if it is null. This is because a policy could overwrite a property value to delete it.

This also means that you as the policy author have to know certain rules about acceptable claim values:

1. The required attributes from the above table must be provided in the *PersistedClaims*, in addition to any other claims that need to be stored.
2. *userPrincipalName* must be of the format user@tenant.onmicrosoft.com, i.e. @tenant domain name.
3. *displayName* is required and cannot be an empty string.

Processing of claim mapping and transformations

The table below shows how the transformations and mappings referenced in the technical profile are processed.

[19] ENTITY AND COMPLEX TYPE REFERENCE | GRAPH API REFERENCE: https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/entity-and-complex-type-reference#EntityreferenceUserEntity

Regardless of the provider, after any claim transformation is executed, the output claims from it are immediately stored in the User Journey storage. User Journeys are covered in section § *Specifying the user journeys*.

Thus, the result of input claims transformations ends up in User Journey storage from where it can be used in input claims, persisted claims, and even as the input claims of output claims transformations.

Order	Policy construct	Required claims in *PersistedClaims*
1	*InputClaimsTransformation*	• Input claims of every InputClaimsTransformation are picked up from the User Journey storage, and after execution, the output claims are put back in the User Journey storage. Thus output claims of an input claims transformation can be input claims of a subsequent input claims transformation as well.
2	*InputClaims*	• Claims are picked up from the User Journey storage – this allows all output claims of input claims transformations to be used as input claims as well (see above).
3	*PersistedClaims*	• Claims are picked up from the UserJourney storage – this allows all output claims of input claims transformations to be used as persisted claims as well. • If a claim has an empty string value, it is replaced with null (empty strings are not allowed). • If displayName property is provided wtih empty or null value, it is replaced with a default name. • Please see the above information relating to the metadata keys and their interpretation on controlling this behavior.
4	*OutputClaims*	• If Operation is **Read**: this is just the claims read. • If Operation is **Write**: an object is returned from Azure AD after a write, any claims that are returned in that object are returned. If a value of an input claim changed during write, it will come back with an updated value. The claims that are listed (and mapped) in the output claims collection are put in the UserJourney storage.
5	*OutputClaimsTransformations*	• Input claims of every output claims Transformation are picked up from the User Journey storage. Thus, output claims ot the technical profile (previous step) as well as output claims of the input claims transformations (step #1) can be input claims of an output claims transformation. • After execution, the output claims are put back in the User Journey storage. Thus, output claims of an output claims transformation can also be input claims of a subsequent output claims transformation.

Supporting custom attributes

Azure AD has a few predefined sets of attributes, such as *givenName*, *lastName*, *telephoneNumber*, *userPrincipalName*, etc. for a *User* object.

Note For more information, see section User Entity[20] in the article ENTITY AND COMPLEX TYPE REFERENCE | GRAPH API REFERENCE.

This said, Azure AD also allows the creation of additional custom attributes, a.k.a. *Extensions* properties (see third document of this series) and refers to them as *extension* attributes. They can be defined on various objects types, such as User, Application, Service Principal, and others. However, a (custom) policy in Azure AD B2C supports adding extension attributes only on users though.

Such a support implies to register extension attributes on an Application object.

Note For more information, see section § *Supporting custom attributes for Azure AD claims providers* in fourth document of this series.

Specifying a technical profile for a SAML 2.0 claims provider

The Identity Experience Engine in Azure AD B2C provides a support for the SAML 2.0. This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

SAML 2.0, as a token format and a protocol, is very popular in the public sector with government agencies as well as with enterprises and educational institutions.

SAML 2.0 is a suite of specifications and, as such, comprises a set of normative and non-normative documents. SAML 2.0 essentially defines XML-based **assertions** and **protocols**, **bindings**, and **profiles**.

A prior understanding or even better a knowledge of the underlying concepts is necessary to appropriately define a technical profile in this space.

If you're not familiar with all of these concepts, the critical aspects of SAML 2.0 are covered in detail in the following four normative documents:

1. ASSERTIONS AND PROTOCOLS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0[21] (SAMLCore), the core specification.
2. BINDINGS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0[22] (SAMLBind), which maps the SAML messages onto the standard messaging or communication protocols.
3. PROFILES FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0[23] (SAMLProf), the use cases or the "*How-to*" in regards to the use of SAML in various situations.
4. And CONFORMANCE REQUIREMENTS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0[24] (SAMLConform), the operational modes for the SAML 2.0 implementations.

---

[20] ENTITY AND COMPLEX TYPE REFERENCE | GRAPH API REFERENCE: https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/entity-and-complex-type-reference#EntityreferenceUserEntity

[21] ASSERTIONS AND PROTOCOLS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[22] BINDINGS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0: http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[23] PROFILES FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[24] CONFORMANCE REQUIREMENTS FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) V2.0: http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf

The term SAML Core, in relationship with the SAMLCore core specification, refers to the general syntax and semantics of SAML assertions (a.k.a. tokens) as well as the protocol used to request and transmit those assertions from one system entity to another. Most of the time, the SAML assertion you may have to consider will be the so-called "bearer" assertion, a short-lived bearer token (i.e. without a proof of possession). Such an assertion may include both an authentication statement and an attribute statement.

A SAML 2.0 protocol describes how certain SAML elements (including assertions) are packaged within SAML request and response elements and gives the processing rules that SAML entities like an IdP in our context must follow when producing or consuming these elements. For the most part, a SAML protocol is a simple request-response protocol. It is important to keep in mind that a SAML protocol always refers to what is transmitted, and not how (the latter is determined by the choice of binding). In the context of this paper, the most interesting SAML protocols are the Authentication Request Protocol, and the Artifact Resolution Protocol,

A SAML 2.0 binding determines how SAML requests and responses map onto standard messaging or communications protocols. In other words, it's a mapping of a SAML protocol message onto standard messaging formats and/or communications protocols. SAML 2.0 completely separates the binding concept from the underlying profile (see below).

The SAML 2.0 standard defines several bindings:

- HTTP Redirect (GET) binding,
- HTTP POST binding,
- HTTP Artifact binding,
- Etc.

A SAML 2.0 profile is a concrete manifestation of a defined use case or the "*How-to*" using a particular combination of assertions, protocols, and bindings, assertions. Indeed, it describes in detail how SAML 2.0 assertions, protocols, and bindings combine to support the considered use case. These

The SAML 2.0 standard defines several profiles:

- Web Browser SSO profile,
- Artifact Resolution profile,
- Enhanced Client or Proxy (ECP) profile,
- Etc.

The most important one is certainly the web Browser SSO profile since this is the primary SAML use case for web SSO and federation. One should also note that these profiles support various possible deployment models.

Based on the above short introduction on SAML 2.0, and the related concepts and normative documents, let's consider how to define a suitable technical profile.

The *Name* attribute of the *Protocol* XML element has to be set to **SAML2** as per *ProtocolName* enumeration in the custom policy XML schema.

The following metadata item keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
PartnerEntity	True	Allow to specify a CDATA section (ignored by the XML parser). This section contains the SAML 2.0 metadata of the SAML 2.0 claims provider.
IssuerUri	False	Specify the Issuer Uri in the SAML 2.0 assertion.

TreatUnsolicitedResponseAsRequest	False	Allow an unsolicited response to serve as an authentication request. Value: true or false. When set to true, this allows to send an assertion to an IDP by an IDP: IDP initiated SSO.
WantsSignedRequests	False	Indicate if you want signed request. Value: true or false. Set to false (i.e. turned that off) for testing when you don't have the production keys from the testing environment.
WantsSignedAssertions	False	Indicate if you want signed assertions. Value: true or false.
RequestsSigned	False	Value: true or false.
WantsSignedResponses	False	Indicate if you want signed response. Value: true or false.
ResponsesSigned	False	Value: true or false.
AssertionsEncrypted	False	Specify that the assertion must be encrypted before being sent. Value: true or false.

The *Suppressions* XML element - IN SPECIAL INTERRUPT CASES - provides the ability to unblock the situation, suppress certain things in the authentication request.

The following metadata item keys may be present in the *Suppressions* XML element:

Item key	Required	Description
AuthnRequest	False	Specify for the authentication request message a comma-separated list of optional attribute/element of the SAML message since some implementations don't support all the optional attribute/element as per protocol and above specifications, what is considered optional. Values in the list: Consent, Destination, Issuer, Extensions, AssertionConsumerServiceUrl, AssertionConsumerServiceIndex, AttributeConsumingServiceIndex, ProtocolBinding, IsPassive, ForceAuthn, ProviderName, Scoping, RequestedAuthenticationcontext, Conditions, NameIDPolicy, and Subject.
LogoutRequest	False	Specify for the logout request message a comma-separated list of optional attribute/element of the SAML message since some implementations don't support all the optional attribute/element as per protocol and above specifications, what is considered optional. Values in the list: Consent, Destination, Issuer, Extensions, NotOnOrAfter, Reason, and SessionIndex.

Note This is very unusual to suppress them.

Similarly, the following key may be present in the *CryptographicKeys* XML element:

Key	Required	Description
SamlAssertionSigning	True	Specify the X509 certificate (RSA key set) to use to sign SAML assertions.
SamlMessageSigning	True	Specify the X509 certificate (RSA key set) to use to sign SAML messages.

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element is empty or absent.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type. **This also enables to take the subject name of the SAML assertion instead of as an attribute**:

```
<OutputClaim ClaimTypeReferenceId="UserId" PartnerClaimType="assertionSubjectName" />
```

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to check or modify the output claims or generate new ones.

The following XML snippet illustrates a technical profile for a SAML 2.0 claims provider:

```
<TechnicalProfile Id="IdP-SAML2-Outbound">
  <DisplayName>Foo IdP</DisplayName>
  <Description>Login with your Foo IdP account</Description>
  <Protocol Name="SAML2" />
  <Metadata>
    <Item Key="WantsSignedAssertions">false</Item>
    <Item Key="TreatUnsolicitedResponseAsRequest">true</Item>
    <Item Key="ResponsesSigned">false</Item>
    <Item Key="IssuerUri">
       https://login.microsoftonline.com/te/litware369b2c.onmicrosoft.com/B2C_1A_TrustFrameworkBase
    </Item>
    <Item Key="PartnerEntity">
      <![CDATA[
        <md:EntityDescriptor entityID="https://web.dev1.foo.com/saml20"
                             xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
          <md:IDPSSODescriptor WantAuthnRequestsSigned="true"
                               protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <md:KeyDescriptor use="signing">
              <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                <X509Data>
                  <X509Certificate>MIIF1DCCA7ygAwIBAgIHC4Dr2wGTODMA…15lH8uDLHNp/ctEQ=</X509Certificate>
                </X509Data>
              </KeyInfo>
            </md:KeyDescriptor>
            <md:KeyDescriptor use="encryption">
              <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                <X509Data>
                   <X509Certificate>MIIF0jCCA7qgAwIBAgIHC4D4+…P4V2neBrOzAQWFo</X509Certificate>
                </X509Data>
              </KeyInfo>
              <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
            </md:KeyDescriptor>
            <md:ArtifactResolutionService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                                          Location="https://web.dev1.foo.com/saml20/soap"
                                          index="0"
                                          isDefault="true"/>
            <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
                                    Location="https://web.dev1.foo.com/saml20/slo"/>
            <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
                                    Location="https://web.dev1.foo.com/saml20/slo"/>
            <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                                    Location="https://web.dev1.foo.com/saml20/soap"/>
            <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</md:NameIDFormat>
            <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
            <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
            <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted</md:NameIDFormat>
            <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
            <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
                                    Location="https://web.dev1.foo.com/saml20/login"/>
            <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
                                    Location="https://web.dev1.foo.com/saml20/login"/>
          </md:IDPSSODescriptor>
          <md:Organization>
            <md:OrganizationName xml:lang="en">Foo Inc.</md:OrganizationName>
            <md:OrganizationDisplayName xml:lang="en">Foo Inc.</md:OrganizationDisplayName>
```

```
            <md:OrganizationURL xml:lang="en"/>
          </md:Organization>
          <md:ContactPerson contactType="technical">
            <md:Company>Foo Inc.</md:Company>
            <md:GivenName>John</md:GivenName>
            <md:SurName>Doe</md:SurName>
            <md:EmailAddress>john.doe@foo.com</md:EmailAddress>
            <md:TelephoneNumber>4254166431</md:TelephoneNumber>
          </md:ContactPerson>
        </md:EntityDescriptor>
      ]]>
    </Item>
  </Metadata>
  <CryptographicKeys>
     <Key Id="SamlAssertionSigning" StorageReferenceId="B2C_1A_SAMLSigningCert"/>
     <Key Id="SamlMessageSigning" StorageReferenceId="B2C_1A_SAMLSigningCert"/>
  </CryptographicKeys>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="AccessLevel" PartnerClaimType="accessLevel" />
    <OutputClaim ClaimTypeReferenceId="AddressCity" PartnerClaimType="city" />
    <OutputClaim ClaimTypeReferenceId="GivenName" PartnerClaimType="cn" />
    <OutputClaim ClaimTypeReferenceId="Email" PartnerClaimType="email" />
    <OutputClaim ClaimTypeReferenceId="Gender" PartnerClaimType="gender" />
    <OutputClaim ClaimTypeReferenceId="MiddleName" PartnerClaimType="middleName" />
    <OutputClaim ClaimTypeReferenceId="AddressZip" PartnerClaimType="postalCode" />
    <OutputClaim ClaimTypeReferenceId="SurName" PartnerClaimType="sn" />
    <OutputClaim ClaimTypeReferenceId="AddressState" PartnerClaimType="stateCode" />
    <OutputClaim ClaimTypeReferenceId="AddressLine2" PartnerClaimType="streetAddressLine2" />
    <OutputClaim ClaimTypeReferenceId="AddressLine1" PartnerClaimType="streetAddressLine1" />
    <OutputClaim ClaimTypeReferenceId="UserId" PartnerClaimType="assertionSubjectName" />
  </OutputClaims>
</TechnicalProfile>
```

Implementation notes on the technical profile

In doing a SAML outbound, the only required metadata item key is *PartnerEntity*.

It can be either:

- The URL of the partner's federation metadata, in which case the metadata is fetched and cached.

-or-

- The actual metadata contained within a CDATA section.

The Entity Id URI as per SAML specification (See "8.3.6 Entity Identifier" in the SAML 2.0 (SAMLCore) core specification) is usually a URL or other identifier given by the Service Provider (SP) that uniquely identifies it. With the Identity Experience Framework in Azure AD B2C, it will default to:

https://<*b2c_service_hostname*>/<your_*b2c_tenant*>/<*your_base_policy*>

For example:

https://login.microsoftonline.com/te/litware369b2c.onmicrosoft.com/B2C_1A_TrustFrameworkBase

Where:

- <*b2c_service_hostname*> is specified by *login.microsoftonline.com/te*.
- <*your_b2c_tenant*> is specified by *litware369b2c.onmicrosoft.com*.
- <your_*base_policy*> is the base XML policy the executed custom policy inherits from, for example *B2C_1A_TrustFrameworkBase*.

Note A Trust Framework base XML policy typically defines the core elements that Azure AD B2C basically leverages: core claims schema and transformations definitions, some key claims providers, and core user journeys. For more information, see section § *Leveraging the inheritance model* in third document of this series.

Generally speaking, SAML claim providers (a.k.a. Identity Provider or IdP in short) must normally be configured to return their assertions to specific addresses. The Identity Experience Framework in Azure AD B2C generates this endpoint and based on the custom policy being executed.

This endpoint will be:

https://<*b2c_service_hostname*>/<*your_b2c_tenant*>/<*your_custom_policy*>/SAMLP/SSO/ASSERTIONCONSUMER

Where <your_*custom_policy*> is the custom policy being executed, for example *B2C_1A_Signup_Signin*.

The partner SAML claim provider must post its response token to this endpoint's URL, for example:

https://login.microsoftonline.com/te/litware369b2c.onmicrosoft.com/B2C_1A_Signup_Signin/SAMLP/SSO/ASSERTIONCONSUMER

Additionally, *CryptographicKeys* XML elements may be required for Azure AD B2C to sign the request, and thus need to be configured separately in the **Policy Keys** section located under Identify Framework Experience user interface. See third document of this series.

```
…
<CryptographicKeys>
   <!--This is the key you have created in Azure AD B2C -> Identify Framework Experience->Policy Keys section -->
   <Key Id="SamlAssertionSigning" StorageReferenceId="B2C_1A_SAMLSigningCert"/>
   <Key Id="SamlMessageSigning" StorageReferenceId="B2C_1A_SAMLSigningCert"/>
</CryptographicKeys>
…
```

Specifying a technical profile for a WS-Fed claims provider

As of this writing, this capability should be considered as under DEVELOPMENT and is NOT part of the features included in the public preview. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[25].

Azure AD B2C provides a support for the WS-Federation (WS-Fed) passive protocol. This can be used for instance to connect to an on-premises AD FS infrastructure.

This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

[25] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

Note For more information about and details that pertain to WS-Federation 1.2, see the WEB SERVICES FEDERATION LANGUAGE (WS-FEDERATION) VERSION 1.2 OASIS STANDARD[26].

The WS-Fed protocol provides a federation metadata document format. The general syntax and semantics of metadata are defined in the aforementioned specification.

The *Name* attribute of the *Protocol* XML element has to be set to **WsFed** as per *ProtocolName* enumeration in the custom policy XML schema.

The following metadata item keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
PartnerEntity	True	Allow to specify a link to the federation metadata XML file of the WS-Fed claims provider. Type: String
IssuerUri	False	Allow to override the *wtRealm* parameter of the WS-Fed request by the specified value. Type: String
Saml2AttributeEncodingInfo	False	Allow to specify a CDATA section (ignored by the XML parser). This section contains SAML attribute statement of the WS-Fed claims provider. Type: String
client_id	False	Should be considered as a "vestigal remanent of protocol signal": protocol gas Type: String

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element is empty or absent.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The *OutputTokenFormat* XML element enables to specify the SAML token format (**SAML11** vs. **SAML2**) as per *TokenFormat* enumeration in the custom policy XML schema. if omitted, **SAML11** is the default.

The *SubjectAuthenticationRequirements* XML element enables to specify the requirements if any regarding the conscious and active participation of the subject in authentication.

The *SubjectNamingInfo* XML element enables to control the production of the subject name in tokens (e.g. SAML) where subject name is specified separately from claims. This is the exact opposite of the *assertionSubjectName* described for the SAML 2.0 claims provider.

[26] WEB SERVICES FEDERATION LANGUAGE (WS-FEDERATION) VERSION 1.2 OASIS STANDARD: http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html

The following XML snippet illustrates a technical profile for a WS-Fed claims provider:

```
<TechnicalProfile Id="ADFS-WSFED-Outbound">
  <DisplayName>Foo ADFS Passive</DisplayName>
  <Description>Some suitable description for foo ADFS</Description>
  <Protocol Name="WsFed" />
  <Metadata>
   <Item Key="PartnerEntity">https://adfs.foo.com/FederationMetadata/2007-06/FederationMetadata.xml</Item>
   <Item Key="IssuerUri">https://te.identityblog.org/identityblog.onmicrosoft.com/MsolPassword</Item>
  </Metadata>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="ImmutableId" PartnerClaimType="ImmutableID" />
    <OutputClaim ClaimTypeReferenceId="UserId" PartnerClaimType="UPN" />
  </OutputClaims>
</TechnicalProfile>
```

The second XML snippet further illustrates how to specify the token format (**SAML11** vs. **SAML2**) as well as the format customization for the outbound claims:

```
<TechnicalProfile Id=" ADFS-WSFED">
   <DisplayName>Foo ADFS Passive</DisplayName>
   <Protocol Name="WsFed" />
   <OutputTokenFormat>SAML11</OutputTokenFormat>
   <SubjectAuthenticationRequirements TimeToLive="4" ResetExpiryWhenTokenIssued="false" />
   <Metadata>
     <Item Key="Saml2AttributeEncodingInfo">
       <![CDATA[
         <saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:Attribute FriendlyName="UserPrincipalName"
                             Name="IDPEmail"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
              <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="xs:string">
              </saml2:AttributeValue>
            </saml2:Attribute>
          </saml2:AttributeStatement>
       ]]>
    </Item>
    <Item Key="Saml11AttributeEncodingInfo">
      <![CDATA[
        <saml:AttributeStatement xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
          <saml:Attribute AttributeName="ImmutableID"
                          AttributeNamespace="http://schemas.microsoft.com/LiveID/Federation/2008/05">
            <saml:AttributeValue></saml:AttributeValue>
          </saml:Attribute>
         <saml:Attribute AttributeName="UPN" AttributeNamespace="http://schemas.xmlsoap.org/claims">
           <saml:AttributeValue></saml:AttributeValue>
         </saml:Attribute>
        </saml:AttributeStatement>
      ]]>
      </Item>
      <Item Key="PartnerEntity">https://www.identityblog.com/wp-content/uploads/2015/01/metadata.xml</Item>
      <Item Key="client_id">customClientId</Item>
  </Metadata>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="immutableId"  PartnerClaimType="ImmutableID" />
    <OutputClaim ClaimTypeReferenceId="userPrincipalName" PartnerClaimType="UPN" />
    <OutputClaim ClaimTypeReferenceId="AuthenticationContext" DefaultValue="urn:federation:authentication:windows" />
  </OutputClaims>
  <SubjectNamingInfo ClaimType="immutableId" />
</TechnicalProfile>
```

Implementation notes on the technical profile

In doing a WS-Fed outbound, the only required metadata item key is *PartnerEntity*.

It can be either:

- The URL of the partner's federation metadata, for example, with AD FS that would be *https://adfs.hostname.example/FederationMetadata/2007-06/FederationMetadata.xml* in which case the metadata is fetched and cached.

-or-

- The actual metadata contained within a CDATA section.

As per WS-Fed specification, the WS-Fed request includes a *wtRealm* parameter. If not specified in metadata, the Azure AD B2C Identity Experience Framework will default this to:

https://<*b2c_service_hostname*>/<your_*b2c_tenant*>/<*your_base_policy*>

For example:

https://login.microsoftonline.com/te/litware369b2c.onmicrosoft.com/B2C_1A_TrustFrameworkBase

Where:

- <*b2c_service_hostname*> is specified by *login.microsoftonline.com/te*.
- <*your_b2c_tenant*> is specified by *litware369b2c.onmicrosoft.com*.
- <your_*base_policy*> is the base XML policy the executed custom policy inherits from, for example *B2C_1A_TrustFrameworkBase*.

Note A Trust Framework base XML policy typically defines the core elements that Azure AD B2C basically leverages: core claims schema and transformations definitions, some key claims providers, and core user journeys. For more information, see section § *Leveraging the inheritance model* in third document of this series.

The *wtRealm* parameter of the WS-Fed request can be overridden by adding an *IssuerUri* metadata item key.

Note This name, which refers to the issuer of the request, will eventually be changed to be **WsRealm** Key.

Generally speaking, WS-Fed claim providers (a.k.a. Identity Provided or IdP in short) must normally be configured to return their assertions to specific addresses. Hence, as per WS-Fed specification, the WS-Fed request also includes a *wReply* parameter, which is the URL to which the response should be posted.

The Identity Experience Framework generates this endpoint and includes it in a *wReply* parameter in the WsFed request. This endpoint will be:

https://<*b2c_service_hostname*>/<*your_b2c_tenant*>/<*your_custom_policy*>/WSFED/SSO/ASSERTIONCONSUMER

Where <your_*custom_policy*> is the custom policy being executed, for example *B2C_1A_Signup_Signin*.

The partner WS-Fed claim provider must post its response token to this endpoint's URL, for example:

https://login.microsoftonline.com/te/litware369b2c.onmicrosoft.com/B2C_1A_Signup_Signin/WSFED/SSO/ASSERTIONCONSUMER

Additionally, no *CryptographicKeys* XML element is required because the WsFed request is not signed and the *PartnerEntity* metadata item key contains the certificate used by the partner to sign its assertions, eliminating the need for this be configured separately.

Specifying a technical profile for a WS-Trust claims provider

As of this writing, this capability should be considered as under DEVELOPMENT and is NOT part of the features included in the public preview. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[27].

Azure AD B2C provides a support for the WS-Trust active protocol. This can be used for instance to connect to an on-premises AD FS infrastructure.

This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

> **Note** For more information about and details that pertain to WS-Federation 1.3, see the WS-TRUST VERSION 1.3 OASIS STANDARD[28].

The *Name* attribute of the *Protocol* XML element has to be set to **WsTrust** as per *ProtocolName* enumeration in the custom policy XML schema.

The following metadata item keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
PartnerEntity	True	Allow to specify a link to the federation metadata XML file of the WS-Fed claims provider.
IssuerUri	True	Specify the Issuer Uri in the token.
ActiveEndpointUrl	True	Specify the URL of the active endpoint to be used with the WS-Trust claims provider. Such a provider may expose multiple endpoints to accommodate various situations or configurations.
ValidationType	False	Specify the type of validation to conduct regarding the existence of the user account. Value: AccountDoesNotExist (default) or AccountDoesNotExist.
ClaimsValidationUserMessage	False	Specify a message to display to the user as part of the validation process. It further refers to a validation technical profile, for use when employed a validation profile.

Similarly, the following key must be present in the *CryptographicKeys* XML element:

Key	Required	Description
SamlMessageSigning	True	Specify the X509 certificate (RSA key set) to use to sign SAML messages.

The *InputClaimsTransformations* XML element is absent.

---

[27] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

[28] WS-TRUST VERSION 1.3 OASIS STANDARD: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf

The *InputClaims* XML element contains the claims bag as the input with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The following XML snippet illustrates a technical profile for a WS-Trust claims provider:

```
<TechnicalProfile Id="ADFS-WSTRUST-Outbound">
  <DisplayName>Foo ADFS Active</DisplayName>
  <Description>Some suitable description for foo ADFS</Description>
  <Protocol Name="WsTrust" />
  <Metadata>
    <Item Key="PartnerEntity">https://adfs.foo.com/FederationMetadata/2007-06/FederationMetadata.xml</Item>
    <Item Key="IssuerUri">https://te.identityblog.org/identityblog.onmicrosoft.com/MsolPassword</Item>
    <Item Key="ActiveEndpointUrl">https://adfs.foo.com/adfs/services/trust/2005/usernamemixed</Item>
    <Item Key="ValidationType">AccountDoesNotExist</Item>
    <Item Key="ClaimsValidationUserMessage">
      That password is your foo password. Please choose a different password for use in the cloud.
    </Item>
   </Metadata>
   <CryptographicKeys>
    <Key Id="SamlMessageSigning" StorageReferenceId="B2CSigningCertificate" />
  </CryptographicKeys>
  <InputClaims>
    <InputClaim ClaimTypeReferenceId="UserId" PartnerClaimType="userPrincipalName" />
    <InputClaim ClaimTypeReferenceId="password" PartnerClaimType="test_password" />
  </InputClaims>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="UserId" PartnerClaimType="UPN" />
  </OutputClaims>
  <ValidationTechnicalProfiles>
    <ValidationTechnicalProfile ReferenceId="CheckPassword">
  <ValidationTechnicalProfiles>
</TechnicalProfile>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for an OAuth 1.0 claims provider

Azure AD B2C provides a support for the OAuth 1.0 protocol. This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

OAuth 1.0 is the very first generation of an authorization protocol for accessing information. Many organizations have already deprecated its usage in favor of OAuth 2.0. It's generally encouraged to migrate to OAuth 2.0 as soon as possible (see next section).

Note For more information about and details that pertain to OAuth 1.0, see the RFC 5849 THE OAUTH 1.0 PROTOCOL[29].

A suitable technical profile definition for an OAuth 1.0 claims provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **OAuth1** as per *ProtocolName* enumeration in the custom policy XML schema.

The following keys must be present in the *Metadata* XML element:

Key	Required	Description
client_id	False	Specify the identifier attributed by the OAuth 1.0 provider.
ProviderName	False	Specify the name of the OAuth1 provider.
request_token_endpoint	True	Indicate the URL of the request token endpoint as per RFC 5849.
authorization_endpoint	True	Indicate the URL of the authorization endpoint as per RFC 5849.
access_token_endpoint	True	Indicate the URL of the token endpoint as per RFC 5849. The callback The callback endpoint is: https://login.microsoftonline.com/<*your_b2c_tenant*>.onmicrosoft.com | <*your_b2c_tenant_Id*>/oauth1/auth

Similarly, the following key must be present in the *CryptographicKeys* XML element:

Key	Required	Description
client_secret	True	Specify the client secret attributed by the OAuth 1.0 provider. Define the key and algorithm.

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element is empty or absent.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

[29] RFC5849 THE OAUTH 1.0 PROTOCOL: http://tools.ietf.org/html/rfc5849

The following XML snippet illustrates a technical profile for Twitter:

```
<TechnicalProfile Id="Twitter-OAuth1">
  <DisplayName>Twitter</DisplayName>
  <Protocol Name="OAuth1" />
  <Metadata>
    <Item Key="ProviderName">twitter</Item>
    <Item Key="request_token_endpoint">https://api.twitter.com/oauth/request_token</Item>
    <Item Key="authorization_endpoint">https://api.twitter.com/oauth/authenticate</Item>
    <Item Key="access_token_endpoint">https://api.twitter.com/oauth/access_token</Item>
    <Item Key="client_id">CMoQQI355stTG3lumdgYRsf8a</Item>
  </Metadata>
  <CryptographicKeys>
    <Key Id="client_secret" StorageReferenceId="TwitterSecret" />
  </CryptographicKeys>
  <InputClaims />
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="userId" PartnerClaimType="user_id" />
    <OutputClaim ClaimTypeReferenceId="givenName" PartnerClaimType="given_name" />
    <OutputClaim ClaimTypeReferenceId="surname" PartnerClaimType="family_name" />
    <OutputClaim ClaimTypeReferenceId="displayName" PartnerClaimType="screen_name" />
    <OutputClaim ClaimTypeReferenceId="email" PartnerClaimType="email" />
    <OutputClaim ClaimTypeReferenceId="identityProvider" DefaultValue="twitter.com" />
    <OutputClaim ClaimTypeReferenceId="authenticationSource" DefaultValue="socialIdpAuthentication" />
  </OutputClaims>
  <OutputClaimsTransformations>
    <OutputClaimsTransformation ReferenceId="CreateUserPrincipalName" />
    <OutputClaimsTransformation ReferenceId="CreateAlternativeSecurityId" />
    <OutputClaimsTransformation ReferenceId="CreateSubjectClaimFromAlternativeSecurityId" />
  </OutputClaimsTransformations>
</TechnicalProfile>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for an OAuth 2.0 claims provider

Azure AD B2C provides a support for the OAuth 2.0 protocol. This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

OAuth 2.0 is more than gaining popularity in the Internet as an authorization protocol for accessing information. This is the primarily protocol for authorization and delegated authentication. Generally speaking, using OAuth 2.0, an application can gain access (with consent from the resource owner – which could be the end user or the administrator user) to impersonate the user, or users in his organization to access the resource.

> **Note** For more information about and details that pertain to OAuth 2.0, see the RFC 6749 The OAuth 2.0 Authorization Framework[30] (updated by the RFC 8252 OAuth 2.0 for Native Apps[31]) and RFC 6750 The OAuth 2.0 Authorization Framework: Bearer Token Usage[32].

A suitable technical profile definition for an OAuth 2.0 claims provider implies the followings.

---

[30] RFC 6749 The OAuth 2.0 Authorization Framework: http://tools.ietf.org/html/rfc6749

[31] RFC 8252 OAuth 2.0 for Native Apps: https://tools.ietf.org/html/rfc8252

[32] RFC 6750 The OAuth 2.0 Authorization Framework: Bearer Token Usage: http://tools.ietf.org/html/rfc6750

The *Name* attribute of the *Protocol* XML element has to be set to **OAuth2** as per *ProtocolName* enumeration in the custom policy XML schema.

The following keys must or may be present in the *Metadata* XML element:

Key	Required	Description
client_id	False	Specify the identifier attributed by the OAuth 2.0 provider.
ProviderName	False	Specify the name of the OAuth2 provider.
authorization_endpoint	True	Indicate the URL of the authorization endpoint as per RFC 6749.
AccessTokenEndpoint	True	Indicate the URL of the token endpoint as per RFC 6749.
ClaimsEndpoint	True	Indicate the URL of the user info endpoint as per RFC 6749.
redirect_uri	False	Indicate the redirection point URL as per RFC 6749.
response_types	False	Specify the response type as per RFC6749. Value: code or token
scope	False	Specify the scope of the access request as per RFC 6749
issuer	False	Specify the issuer of the access request as per RFC 6749
HttpBinding	False	Specify the expected HTTP binding. Value: **GET** or **POST**

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element is empty or absent.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The following XML snippet illustrates a technical profile for Google:

```
<TechnicalProfile Id="Google">
  <DisplayName>Google</DisplayName>
  <Protocol Name="OAuth2" />
  <Metadata>
    <Item Key="ProviderName">google</Item>
    <Item Key="authorization_endpoint">https://accounts.google.com/o/oauth2/auth</Item>
    <Item Key="AccessTokenEndpoint">https://accounts.google.com/o/oauth2/token</Item>
    <Item Key="ClaimsEndpoint">https://www.googleapis.com/oauth2/v1/userinfo</Item>
    <Item Key="scope">email</Item>
    <Item Key="HttpBinding">POST</Item>
  </Metadata>
  <InputClaims />
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="userId" PartnerClaimType="id" />
    <OutputClaim ClaimTypeReferenceId="email" PartnerClaimType="email" />
    <OutputClaim ClaimTypeReferenceId="givenName" PartnerClaimType="given_name" />
    <OutputClaim ClaimTypeReferenceId="surname" PartnerClaimType="family_name" />
    <OutputClaim ClaimTypeReferenceId="displayName" PartnerClaimType="name" />
    <OutputClaim ClaimTypeReferenceId="identityProvider" DefaultValue="google.com" />
    <OutputClaim ClaimTypeReferenceId="authenticationSource" DefaultValue="socialIdpAuthentication" />
  </OutputClaims>
  <OutputClaimsTransformations>
    <OutputClaimsTransformation ReferenceId="CreateUserPrincipalName" />
    <OutputClaimsTransformation ReferenceId="CreateAlternativeSecurityId" />
```

```
    <OutputClaimsTransformation ReferenceId="CreateSubjectClaimFromAlternativeSecurityId" />
  </OutputClaimsTransformations>
</TechnicalProfile>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for an OpenID Connect claims provider

Azure AD B2C provides a support for the OpenID Connect protocol. This section outlines the specifics of a technical profile for interacting with claims provider supporting this standardized protocol.

OpenID Connect 1.0 defines an identity layer on top of OAuth 2.0 and represents the state of the art in modern authentication protocols. It's a suite of lightweight specifications that provide a framework for identity interactions via REST like APIs. It is based on OAuth 2.0.

> **Note** For more information about and details that pertain to OpenID Connect, see the specification OpenID Connect Core 1.0[33].

> **Note** The OpenID Foundation has recently launched a certification program for OpenID Connect 1.0 implementations. For more information, see the article The OpenID Foundation Launches OpenID Connect Certification Program[34].
>
> Having an OpenID Connect certification program provides confidence that certified implementations will "just work" together. This represents another important step on the road to widely-available secure interoperable digital identity for all the devices and applications that people use. Microsoft is proud to be a key contributor to the development of OpenID Connect 1.0 and now of its certification program. Azure AD has successfully passed the certification and is certified[35] as an OpenID Connect 1.0 identity provider.

OpenID Connect 1.0 provides a JSON configuration document (*openid-configuration.json*) as per OpenID Connect Discovery[36] specification. This metadata information in JSON format provides configuration information such as the OAuth 2.0 endpoint locations, the signing key and issuer values to validate, etc. in analogy to what happens for WS-Federation and SAML 2.0 as described before.

With this in mind, a suitable technical profile definition for an OAuth 2.0 claims provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **OpenIdConnect** as per *ProtocolName* enumeration in the custom policy XML schema.

---

[33] OpenID Connect 1.0: https://msdn.microsoft.com/en-us/library/azure/dn645541.aspx
[34] The OpenID Foundation Launches OpenID Connect Certification Program: http://openid.net/2015/04/17/openid-connect-certification-program/
[35] OpenID Certification: http://openid.net/certification/
[36] OpenID Connect Discovery 1.0 incorporating errata set 1: http://openid.net/specs/openid-connect-discovery-1_0.html

The following keys must or may be present in the *Metadata* XML element:

Key	Required	Description
client_id	False	Specify the identifier attributed by the OpenID Connect provider.
IdTokenAudience	False	Specify the audience of the id token.
ProviderName	True	Specify the name of the OpenID Connect provider.
METADATA	False	Specify the JSON configuration document as per OpenID Connect Discovery specification.
authorization_endpoint	True	Indicate the URL of the authorization endpoint as per OpenID Connect Core 1.0 specification.
redirect_uri	False	Indicate the redirection point URL as per OpenID Connect Core 1.0 specification.
response_types	False	Specify the response type as per OpenID Connect Core 1.0 specification. Value: id_token, code or token
response_mode	False	Specify the response mode. Value: query
scope	False	Specify the scope of the access request as per OpenID Connect Core 1.0 specification.
issuer	False	Specify the issuer of the access request as per OpenID Connect Core 1.0 specification.
HttpBinding	False	Specify the expected HTTP binding. Value: **GET** or **POST**
LocalAccountProfile	False	Value: true or false

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element contains the claims bag as the input with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The following XML snippet illustrates a technical profile for an OpenID Connect claims provider:

```
<TechnicalProfile Id="LocalAccountSignInWithEvoSts">
  <DisplayName>Local Account SignIn</DisplayName>
  <Protocol Name="OpenIdConnect" />
  <Metadata>
    <Item Key="ProviderName">https://sts.windows.net/</Item>
    <Item Key="METADATA">https://login.microsoftonline.com/{tenant}/.well-known/openid-configuration</Item>
    <Item Key="authorization_endpoint">https://login.microsoftonline.com/{tenant}/oauth2/authorize</Item>
    <Item Key="response_types">id_token</Item>
    <Item Key="response_mode">query</Item>
    <Item Key="scope">email openid</Item>
    <Item Key="client_id">bb2a2e3a-c5e7-4f0a-88e0-8e01fd3fc1f4</Item>
    <Item Key="IdTokenAudience">bb2a2e3a-c5e7-4f0a-88e0-8e01fd3fc1f4</Item>
    <Item Key="UsePolicyInRedirectUri">false</Item>
    <Item Key="HttpBinding">POST</Item>
    <Item Key="LocalAccountProfile">true</Item>
  </Metadata>
  <InputClaims>
    <InputClaim ClaimTypeReferenceId="logonIdentifier" PartnerClaimType="login_hint" />
    <InputClaim ClaimTypeReferenceId="nux" PartnerClaimType="nux" DefaultValue="1" />
    <InputClaim ClaimTypeReferenceId="nca" PartnerClaimType="nca" DefaultValue="1" />
    <InputClaim ClaimTypeReferenceId="tenantId" PartnerClaimType="domain_hint"
                DefaultValue="{Policy:RelyingPartyTenantId}" />
    <InputClaim ClaimTypeReferenceId="prompt" PartnerClaimType="prompt" DefaultValue="{OIDC:prompt}" />
    <InputClaim ClaimTypeReferenceId="mkt" PartnerClaimType="mkt" DefaultValue="{Culture:RFC5646}" />
    <InputClaim ClaimTypeReferenceId="lc" PartnerClaimType="lc" DefaultValue="{Culture:LCID}" />
  </InputClaims>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="objectId" PartnerClaimType="oid" />
    <OutputClaim ClaimTypeReferenceId="tenantId" PartnerClaimType="tid" />
    <OutputClaim ClaimTypeReferenceId="givenName" PartnerClaimType="given_name" />
    <OutputClaim ClaimTypeReferenceId="surName" PartnerClaimType="family_name" />
    <OutputClaim ClaimTypeReferenceId="displayName" PartnerClaimType="name" />
    <OutputClaim ClaimTypeReferenceId="userPrincipalName" PartnerClaimType="upn" />
    <OutputClaim ClaimTypeReferenceId="authenticationSource" DefaultValue="evoStsAuthentication" />
  </OutputClaims>
  <OutputClaimsTransformations>
    <OutputClaimsTransformation ReferenceId="CreateSubjectClaimFromObjectID" />
  </OutputClaimsTransformations>
</TechnicalProfile>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for a self-asserted claims provider

A suitable technical profile definition for a self-asserted claims provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C, for example in the below illustrations:

"*Web.TPEngine.Providers.SelfAssertedAttributeProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following keys must present in the *Metadata* XML element:

Key	Required	Description
IpAddressClaimReferenceId	True	
ContentDefinitionReferenceId	True	
TokenLifeTimeInSeconds	True	
LocalAccountType	False	
LocalAccountProfile	False	

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element can be empty or absent or may contain the claims bag as the input with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

The following XML snippet illustrates a technical profile for a self-asserted claims provider:

```
<TechnicalProfile Id="SocialLoginSetPasswordHash">
  <DisplayName>Set Password Hash for Social Account</DisplayName>
  <Protocol Name="Proprietary"
            Handler="Web.TPEngine.Providers.SelfAssertedAttributeProvider, Web.TPEngine, Version=1.0.0.0,
                     Culture=neutral, PublicKeyToken=null"/>
  <Metadata>
    <Item Key="IpAddressClaimReferenceId">IpAddress</Item>
    <Item Key="ContentDefinitionReferenceId">api.localaccountsignup</Item>
    <Item Key="TokenLifeTimeInSeconds">3600</Item>
  </Metadata>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="CloudPassword" />
    <OutputClaim ClaimTypeReferenceId="password" Required="true"/>
    <OutputClaim ClaimTypeReferenceId="reenterPassword" Required="true"/>
  </OutputClaims>
  <ValidationTechnicalProfiles>
    <ValidationTechnicalProfile ReferenceId="ADFS-WSTRUST-Outbound"/>
    <ValidationTechnicalProfile ReferenceId="AAD-UserWriteHashedPassword"/>
  </ValidationTechnicalProfiles>
</TechnicalProfile>
```

This second XML code snippet further illustrates the use of the *IncludeClaimsFromTechnicalProfile* XML element where only the output claims are considered from the profile, and all the rest of the technical profile is ignored:

```
<TechnicalProfile Id="LocalAccountSignUpWithLogonName">
  <DisplayName>User ID signup</DisplayName>
  <Protocol Name="Proprietary"
            Handler="Web.TPEngine.Providers.SelfAssertedAttributeProvider, Web.TPEngine, Version=1.0.0.0,
                     Culture=neutral, PublicKeyToken=null" />
  <Metadata>
    <Item Key="IpAddressClaimReferenceId">IpAddress</Item>
    <Item Key="ContentDefinitionReferenceId">api.localaccountsignup</Item>
    <Item Key="TokenLifeTimeInSeconds">3600</Item>
```

```
    <Item Key="LocalAccountType">Username</Item>
    <Item Key="LocalAccountProfile">true</Item>
  </Metadata>
  <CryptographicKeys>
    <Key Id="issuer_secret" StorageReferenceId="JwtTokenSigningKeyContainer" />
  </CryptographicKeys>
  <InputClaims>
    <InputClaim ClaimTypeReferenceId="UserId" />
  </InputClaims>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="objectId" Required="true" />
    <OutputClaim ClaimTypeReferenceId="userId" Required="true" />
    <OutputClaim ClaimTypeReferenceId="password" Required="true" />
    <OutputClaim ClaimTypeReferenceId="reenterPassword" Required="true" />
    <OutputClaim ClaimTypeReferenceId="email" PartnerClaimType="Verified.Email" Required="true" />
    <OutputClaim ClaimTypeReferenceId="emails" />
    <OutputClaim ClaimTypeReferenceId="strongAuthenticationEmailAddress" />
    <OutputClaim ClaimTypeReferenceId="sub" Required="true" />
    <OutputClaim ClaimTypeReferenceId="executed-SelfAsserted-Input" DefaultValue="true" />
    <OutputClaim ClaimTypeReferenceId="newUser" />
    <OutputClaim ClaimTypeReferenceId="authenticationSource" />
    <OutputClaim ClaimTypeReferenceId="userPrincipalName" />
  </OutputClaims>
  <ValidationTechnicalProfiles>
    <ValidationTechnicalProfile ReferenceId="AAD-UserWriteUsingLogonName" />
  </ValidationTechnicalProfiles>
  <IncludeClaimsFromTechnicalProfile>SelfAsserted-Input</IncludeClaimsFromTechnicalProfile>
  <IncludeTechnicalProfile ReferenceId="LocalAccountContentOverrides" />
</TechnicalProfile>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for a RESTful claims provider

A suitable technical profile definition for a RESTful claims provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C, for example in the below illustrations:

"*Web.TPEngine.Providers.RestfulProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
userinfo_endpoint	True	Indicate the URL of the user info endpoint.
AuthenticationType	True	Specify the type of authentication being performed by the RESTful claims provider. Type: String Value: one of the following types: • **None**. • **Basic**. • **ClientCertificate**.

Username	False	Specify the username in the form of an email address as per RFC 822 if the type of authentication is set to **Basic**.
ClientCertificate	False	Specify the client certificate to use if the type of authentication is set to **ClientCertificate**.
SendClaimsIn	False	Specify how the input claims are sent to the RESTful claims provider. Type: String Value: one of the following types: • **Body** (default). Should be sent in the request body (JSON format). • **Form**. Should be sent in the request body (& separated key value format, same as query string but in the body). • **Header**. Should be sent in the request header. • **QueryString**, should be sent in the request query string.
ClaimsFormat	True	Specify the format for the output claims. Type: String Value: one of the above types (**Body**, **Form**, **Header**, and **QueryString**).
IgnoreServerCertificateErrors	True	Specify how to handle the certificate chains checking: Value: one of the following types: • **True**. Ignore certificate chains checking. • **False**. Do certificate chains checking.

Depending of the type of authentication, the *CryptographicKeys* XML element is absent or present. More specifically, if the type of authentication is set to **None**, the *CryptographicKeys* XML element is absent. It will be otherwise present for any other value.

Thus, if the type of authentication is set to **Basic**, the following keys must be present in the *CryptographicKeys* XML element:

Key	Required	Description
BasicAuthenticationUsername	True	Specify the username to use to authenticate.
BasicAuthenticationPassword	True	Specify the password to use to authenticate.

Similarly, if the type of authentication is set to **ClientCertificate**, the following key must be present in the *CryptographicKeys* XML element:

Key	Required	Description
ClientCertificate	True	Specify the X509 certificate (RSA key set) to use to authenticate. Mutual authentication will be potentially supported in future.

The *InputClaimsTransformations* XML element is absent.

The *InputClaims* XML element contains the claims bag as the input with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *PersistedClaims* XML element is absent.

The *OutputClaims* XML element contains the claims bag as the output with possible mapping information between a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file and that partner claim type, and/or a default value.

The *OutputClaimsTransformations* XML element may contain a collection of *OutputClaimsTransformation* to be used to modify the output claims or generate new ones.

Considering the above, the following XML snippet illustrates three technical profiles for the same RESTful claims provider, i.e.:

1. A first one (identified as "*RestfulProvider-NoAuth*") with no authentication required,
2. A second one (identified as "*RestfulProvider-BasicAuth*") with Basic authentication,
3. And eventually a third one (identified as "*RestfulProvider-BasicAuth*") with client-based X509 certificate authentication.

As illustrated, the type of authentication to perform if any (partially) guides the definition of the related profile as you can now imagine:

```
<TechnicalProfiles>
  <TechnicalProfile Id="RestfulProvider-NoAuth">
    <DisplayName>Restful Claims Provider</DisplayName>
    <Protocol Name="Proprietary" Handler="Web.TPEngine.Providers.RestfulProvider, Web.TPEngine, Version=1.0.0.0,
                                Culture=neutral, PublicKeyToken=null"/>
    <Metadata>
      <Item Key="userinfo_endpoint">https://reflector.trafficmanager.net/rest/restfulresponder.aspx</Item>
      <Item Key="AuthenticationType">None</Item>
      <Item Key="SendClaimsIn">Body</Item>
      <Item Key="ClaimsFormat">Body</Item>
      <Item Key="IgnoreServerCertificateErrors">true</Item>
   </Metadata>
   <InputClaims>
     <InputClaim ClaimTypeReferenceId="SendToProvider" DefaultValue="Hello World!" />
   </InputClaims>
   <OutputClaims>
     <OutputClaim ClaimTypeReferenceId="RestfulResponse" />
     </OutputClaims>
  </TechnicalProfile>

  <TechnicalProfile Id="RestfulProvider-BasicAuth">
  <DisplayName>Restful Claims Provider</DisplayName>
  <Protocol Name="Proprietary" Handler="Web.TPEngine.Providers.RestfulProvider, Web.TPEngine, Version=1.0.0.0,
                                      Culture=neutral, PublicKeyToken=null"/>
  <Metadata>
    <Item Key="userinfo_endpoint">https://reflector.trafficmanager.net/rest/basic/restfulbasicauthresponder.aspx</Item>
    <Item Key="AuthenticationType">Basic</Item>
    <!-- -- >
    <Item Key="Username">some@username</Item>
  </Metadata>
  <CryptographicKeys>
    <Key Id="BasicAuthenticationUsername" StorageReferenceId="SomeKeyName" />
    <Key Id="BasicAuthenticationPassword" StorageReferenceId="PasswordSecret" />
  </CryptographicKeys>
  <InputClaims>
    <InputClaim ClaimTypeReferenceId="SendToProvider" DefaultValue="Hello World!" />
  </InputClaims>
  <OutputClaims>
    <OutputClaim ClaimTypeReferenceId="RestfulResponse" />
  </OutputClaims>
  </TechnicalProfile>

  <TechnicalProfile Id="RestfulProvider-ClientCertificate">
  <DisplayName>Restful Claims Provider</DisplayName>
  <Protocol Name="Proprietary" Handler="Web.TPEngine.Providers.RestfulProvider, Web.TPEngine, Version=1.0.0.0,
                                      Culture=neutral, PublicKeyToken=null"/>
  <Metadata>
    <Item Key="userinfo_endpoint">
       https://reflector.trafficmanager.net/rest/clientcertificate/restfulclientcertificateresponder.aspx
    </Item>
    <Item Key="AuthenticationType">ClientCertificate</Item>
   </Metadata>
   <CryptographicKeys>
```

```
      <Key Id="ClientCertificate" StorageReferenceId="TestCertificate" />
    </CryptographicKeys>
    <InputClaims>
      <InputClaim ClaimTypeReferenceId="SendToProvider" DefaultValue="Hello World!" />
    </InputClaims>
    <OutputClaims>
      <OutputClaim ClaimTypeReferenceId="RestfulResponse" />
    </OutputClaims>
  </TechnicalProfile>
</TechnicalProfiles>
```

Implementation notes on the technical profile

None.

Specifying a technical profile for session management

Single Sign-On (SSO) session management allows a custom policy's author to control how Azure AD B2C continues to interact with a user after successful authentication i.e. showing or not a selection of identity providers to choose from, entering again local account details.

It can be thought of in two halves, the first dealing with interactions native to Azure AD B2C and the other interactions with external parties such as Facebook.

Azure AD B2C does not override or bypass SSO sessions that might be held by external parties. Rather the route through Azure AD B2C to get to the external party is "remembered" avoiding the need to re-prompt the user. But the ultimate SSO decision remains with the external party, and **this is important**.

As such, SSO session management uses the same semantics as any other technical profile. When an orchestration step is executed, the technical profile associated with the considered step is queried for a *UseTechnicalProfileForSessionManagement* reference (see section § *Specifying technical profile(s) for a given claims provider*). If one is specified, the referenced SSO session management provider per its identifier is then checked to see if the user is a session participant. If so, the SSO session provider is used to repopulate the session. Similarly, when the execution of an orchestration step is complete, the provider is used to store information in the session if an SSO session management provider has been specified.

A number of SSO session management provider classes have been defined, and that can be used for purpose.

Note For more information, see article AZURE AD B2C: SINGLE SIGN-ON (SSO) SESSION MANAGEMENT[37].

Specifying a technical profile for a Noop SSO session provider

This section outlines the specifics of a technical profile for the Noop SSO session provider.

As the name indicates, this provider does nothing. You can use this provider for suppressing SSO behavior for a specific technical profile.

A suitable technical profile definition for this provider implies the followings.

[37] AZURE AD B2C: SINGLE SIGN-ON (SSO) SESSION MANAGEMENT: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-sso-custom

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"*Web.TPEngine.SSO.NoopSSOSessionProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following XML snippet illustrates a typical technical profile for the Noop SSO session provider:

```
<TechnicalProfile Id="SM-AAD">
   <DisplayName>Session Mananagement Provider</DisplayName>
   <Protocol Name="Proprietary" Handler="Web.TPEngine.SSO.NoopSSOSessionProvider, Web.TPEngine, Version=1.0.0.0,
                                        Culture=neutral, PublicKeyToken=null" />
</TechnicalProfile>
```

Specifying a technical profile for a default SSO session provider

This section outlines the specifics of a technical profile for the default SSO session provider.

This provider can be used for storing claims in a session. A suitable technical profile definition for this provider implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"*Web.TPEngine.SSO.DefaultSSOSessionProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The *PersistedClaims* XML element may contain the claims bag as all the values that should be persisted in the session. When the provider is used to rehydrate the session, the persisted claims are added to the claims bag.

The *OutputClaims* XML element contains the claims bag as the output for the additional claims that can be used to indicate the session state, etc.

The following XML snippet illustrates a typical technical profile for the default SSO session provider:

```
<TechnicalProfile Id="SM-AAD">
   <DisplayName>Session Mananagement Provider</DisplayName>
   <Protocol Name="Proprietary" Handler="Web.TPEngine.SSO.DefaultSSOSessionProvider, Web.TPEngine, Version=1.0.0.0,
                                        Culture=neutral, PublicKeyToken=null" />
   <PersistedClaims>
      <PersistedClaim ClaimTypeReferenceId="objectId" />
      <PersistedClaim ClaimTypeReferenceId="newUser" />
      <PersistedClaim ClaimTypeReferenceId="executed-SelfAsserted-Input" />
   </PersistedClaims>
   <OutputClaims>
      <OutputClaim ClaimTypeReferenceId="objectIdFromSession" DefaultValue="true" />
   </OutputClaims>
</TechnicalProfile>
```

Specifying a technical profile for an external login SSO session provider

This section outlines the specifics of a technical profile for the external login SSO session provider.

This provider can be used for adding an external provider to session. This is used to suppress the "Choose account provider" screen.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"*Web.TPEngine.SSO.ExternalLoginSSOSessionProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following XML snippet illustrates a typical technical profile for the external login SSO session provider:

```
<TechnicalProfile Id="SM-SocialLogin">
   <DisplayName>Session Mananagement Provider</DisplayName>
   <Protocol Name="Proprietary" Handler="Web.TPEngine.SSO.ExternalLoginSSOSessionProvider, Web.TPEngine, Version=1.0.0.0,
                                        Culture=neutral, PublicKeyToken=null" />
</TechnicalProfile>
```

Specifying a technical profile for an SAML SSO session provider

This section outlines the specifics of a technical profile for the SAML SSO session provider.

This provider can be used this for managing both the B2C SAML session and any external SAML identity providers. SAML session logout requires the SessionIndex and NameID to complete.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"*Web.TPEngine.SSO.SamlSSOSessionProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following metadata item key may or must be present in the *Metadata* XML element:

Item key	Required	Description
IncludeSessionIndex	True	Indicates to the provider that the session index should be stored. Value: true or false.
RegisterServiceProviders	True	Indicates that the provider should register all SAML service providers that have been issued an assertion. Value: true or false.

When using the provider for storing a SAML identity provider session, the above parameters should both be false.

When using the provider for storing the B2C SAML session, the above parameters should be true or omitted as the defaults are true.

The following XML snippet illustrates a typical technical profile for the SAML SSO session provider:

```
<TechnicalProfile Id="SM-Reflector-SAML">
   <DisplayName>Session Management Provider</DisplayName>
   <Protocol Name="Proprietary" Handler="Web.TPEngine.SSO.SamlSSOSessionProvider, Web.TPEngine, Version=1.0.0.0,
                                        Culture=neutral, PublicKeyToken=null" />
   <Metadata>
      <Item Key="IncludeSessionIndex">false</Item>
      <Item Key="RegisterServiceProviders">false</Item>
```

```
   </Metadata>
</TechnicalProfile>
```

Implementation notes on the SSO management session technical profiles

SSO management session profiles are specified using the *UseTechnicalProfileForSessionManagement* element of a technical profile as already depicted (See section § *Specifying technical profile(s) for a given claims provider*):

<UseTechnicalProfileForSessionManagement ReferenceId="<*Id*>" />

Where *Id* is the identifier of the related SSO Management Session profile.

When an orchestration step is executed, the technical profile (i.e. the function) associated with the step is queried for a *UseTechnicalProfileForSessionManagement* reference. If one exists, the referenced profile is then checked to see if it is a session participant. If so the SSO session provider is used to repopulate the session.

When the execution of an orchestration step is complete, if an SSO session management provider has been specified, the provider is used to store information in the session.

The *UserJourneyBehaviors* element of the Relying Party (RP) policy can be used to control the session lifetime, scope and behavior (see section § *Specifying the relying party*).

```
…
<UserJourneyBehaviors>
   <SessionExpiryType>Absolute</SessionExpiryType>
   <SessionExpiryInSeconds>1200</SessionExpiryInSeconds>
</UserJourneyBehaviors>
…
```

The SSO session is currently stored in encrypted cookies.

Specifying technical profiles for Application Insights

As already illustrated in the fifth document of this series, the Identity Framework in Azure AD B2C works well with Azure Application Insights to gain insights on user behavior and troubleshoot your own policies in development or in production.

In addition to the above, this section outlines the specifics of technical profiles for the Application Insights provider.

This provider can be used for gaining insights on user behavior, measuring performance, and creating notifications from Application Insights.

The *Name* attribute of the *Protocol* XML element has to be set to **Proprietary** as per *ProtocolName* enumeration in the custom policy XML schema. Thus, the handler attribute must contain the fully-qualified name of the protocol handler assembly that will be used by Azure AD B2C in this case:

"*Web.TPEngine.Providers.UserJourneyContextProvider, Web.TPEngine, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null*"

The following metadata item key may or must be present in the *Metadata* XML element:

Item key	Required	Description
InstrumentationKey	True	Indicates the Application Insights instrumentation key which will be used for logging the events. Value: the instrumentation key.
DeveloperMode	True	Indicates whether developer mode is enabled. This controls how events are buffered. In a development environment with minimal event volume, enabling the developer mode results in events being sent immediately to Application Insights. Value: true or false.
DisableTelemetry	True	Indicates whether telemetry should be enabled or not.

The following XML snippet illustrates a typical technical profile for the Application Insights provider:

```
<TechnicalProfile Id="AzureInsights-Common">
   <DisplayName>Azure Appication Insights </DisplayName>
   <Protocol Name="Proprietary" Handler="Web.TPEngine.Providers.Insights.AzureApplicationInsightsProvider, Web.TPEngine,
                  Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />
      <Metadata>
         <Item Key="InstrumentationKey">xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</Item>
         <Item Key="DeveloperMode">false</Item>
         <Item Key="DisableTelemetry ">false</Item>
      </Metadata>
      <InputClaims>
        <!-- Properties of an event are added through the syntax {property:NAME}, where NAME is property being added to
             the event. DefaultValue can be either a static value or a value that's resolved by one of the supported
             DefaultClaimResolvers. -->
        <InputClaim ClaimTypeReferenceId="PolicyId" PartnerClaimType="{property:Policy}"
                    DefaultValue="{Policy:PolicyId}" />
        <InputClaim ClaimTypeReferenceId="CorrelationId" PartnerClaimType="{property:JourneyId}" />
        <InputClaim ClaimTypeReferenceId="Culture" PartnerClaimType="{property:Culture}"
                    DefaultValue="{Culture:RFC5646}" />
      </InputClaims>
   </TechnicalProfile>
</TechnicalProfiles>
```

Implementation notes on the technical profiles

This provider sends event data directly to Application Insights by using the instrumentation key provided. A technical profile uses this provider to define an event from B2C. The profile specifies the name of the event, the claims that will be recorded, and the instrumentation key. To post an event, the technical profile is then added as an orchestration step or as a validation technical profile in a custom user journey.

Application Insights can unify the events by using a correlation ID to record a user session. Application Insights makes the event and session available within seconds and presents many visualization, export, and analytical tools.

Note For more information, see article TRACK USER BEHAVIOR IN AZURE AD B2C JOURNEYS BY USING APPLICATION INSIGHTS[38].

[38] TRACK USER BEHAVIOR IN AZURE AD B2C JOURNEYS BY USING APPLICATION INSIGHTS: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-custom-guide-eventlogger-appins

Specifying the token issuers

The technical profile definition in the policy XML file described for a claim provider in the previous section also applies in the case of a token issuer that can be Azure AD B2C service itself or a remote token issuer that your organization hosts.

This section highlights the few differences if any and the specifics in the related technical profile definitions.

Using a remote token issuer service

As of this writing, this capability should be considered as under DEVELOPMENT and is NOT part of the features included in the public preview. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[39].

If you want to use a remote token issuer service in lieu of the ones provided by the Azure AD B2C service, you must add a *RequestOperation* key item in the metadata of the technical profile as follows:

```
<TechnicalProfiles>
  <TechnicalProfile …>
    <Metadata>
    <Item Key="RequestOperation">HttpRequest</Item>
    …
    <Metadata>
    …
  </TechnicalProfile>
  …
</TechnicalProfile>
```

If such a key is not specified, no remote token issuer service will be used in the technical profiles described in the next section.

If a remote token issuer service is specified, you also need to specify a cryptographic key with an identifier that is set to "client_secret" as follows.

```
<TechnicalProfile …>
  <Metadata>
    <Item Key=" RequestOperation">HttpRequest</Item>
    …
  <Metadata>
  …
  <CryptographicKeys>
    <Key Id="client_secret" StorageReferenceId="PairwiseClientSecret" />
    …
  </CryptographicKeys>
  …
</TechnicalProfile>
```

[39] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

The "PairwiseClientSecret" identifier in the above XML snippet is used to uniquely identify a particular storage key container. For more information, see section § *Managing your key containers for Trust Framework (policies)* in the third document of this series.

This key is used to authenticate the Identity Experience Engine in Azure AD B2C to the remote token issuer service.

Specifying a technical profile for a SAML 2.0 token issuer

As of this writing, this capability should be considered as under DEVELOPMENT and is NOT part of the features included in the public preview. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[40].

A suitable technical profile definition for a SAML 2.0 token issuer implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **None** as per *ProtocolName* enumeration in the custom policy XML schema - this should become default overtime but **None** will be deprecated - while the *OutputTokenFormat* XML element has to be set to **SAML2** (for SAML 2.0 assertions) as per *TokenFormat* enumeration in the custom policy XML schema.

The following keys must or may be present in the *Metadata* XML element:

Key	Required	Description
IssuerUri	False	Specify the issuer in the SAML assertion. If not specified, default to something rational. Type: String
TokenLifeTimeInSeconds	False	Specify the lifetime of the SAML assertion. Type: Integer
RequestOperation	False	Allow if specified to use a remote token issuer service in lieu of the one provided by the Azure AD B2C service. See section § *Using a remote token issuer service*. Type: String Value: Must be set to **HttpRequest**.
client_secret	False	Specify a cryptographic key to use if a remote token issuer service is being used. Define the key and algorithm.

Similarly, the following keys must or may be present in the *CryptographicKeys* XML element:

Key	Required	Description
SamlMessageSigning	True	Specify the X509 certificate (RSA key set) to use to sign SAML messages.
SamlAssertionSigning	True	Specify the X509 certificate (RSA key set) to use to sign SAML assertions.
SamlAssertionDecryption	True	Specify the X509 certificate (RSA key set) to use to decrypt SAML messages if you have to.

[40] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

Note All the above keys may reference the same X509 certificate.

Both the *InputClaims* and *OutputClaims* XML elements are empty or absent. Likewise, the *OutputClaimsTransformations* XML element is also absent here.

The following XML snippet illustrates how to define a technical profile for a SAML 2.0 token issuer:

```
<TechnicalProfile Id="Saml2AssertionIssuer">
  <DisplayName>Token Issuer</DisplayName>
  <Protocol Name="None" />
  <OutputTokenFormat>SAML2</OutputTokenFormat>
  <Metadata>
    <Item Key="IssuerUri">https://sts.windows.net/csdii.onmicrosoft.com/B2C_1A_casinitiated</Item>
    <Item Key="TokenLifeTimeInSeconds">600</Item>
   </Metadata>
   <CryptographicKeys>
     <Key Id="SamlMessageSigning" StorageReferenceId="B2CSigningCertificate" />
     <Key Id="SamlAssertionSigning" StorageReferenceId="B2CSigningCertificate" />
    <Key Id="SamlAssertionDecryption" StorageReferenceId="B2CSigningCertificate" />
  </CryptographicKeys>
  <InputClaims />
  <OutputClaims />
</TechnicalProfile>
```

Specifying a technical profile for a WS-Fed token issuer

As of this writing, this capability should be considered as under DEVELOPMENT and is NOT part of the features included in the public preview. For information, see RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW[41].

A suitable technical profile definition for a WS-Fed token issuer implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **None** as per *ProtocolName* enumeration in the custom policy XML schema - this should become default overtime, but **None** will be deprecated - while the *OutputTokenFormat* XML element has to be set to **SAML11** (for SAML 1.1 assertions) or **SAML2** (for SAML 2.0 assertions) as per *TokenFormat* enumeration in the custom policy XML schema.

The following keys must or may be present in the *Metadata* XML element:

Item key	Required	Description
IssuerUri	False	Specify the issuer in the SAML assertion. If not specified, default to something rational. Type: String
RequestOperation	False	Allow if specified to use a remote token issuer service in lieu of the one provided by the Azure AD B2C service. See section § *Using a remote token issuer service*. Type: String Value: Must be set to **HttpRequest**.
client_secret	False	Specify a cryptographic key to use if a remote token issuer service is being used. Define the key and algorithm.

[41] RELEASE NOTES FOR AZURE ACTIVE DIRECTORY B2C CUSTOM POLICY PUBLIC PREVIEW: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-developer-notes-custom

Similarly, the following keys must be present in the *CryptographicKeys* XML element:

Key	Required	Description
SamlAssertionSigning	True	Specify the X509 certificate used to signed the SAML assertion.
SamlMessageSigning	True	Specify the X509 certificate used to signed the SAML message.

Both the *InputClaims* and *OutputClaims* XML elements are empty or absent. Likewise, the *OutputClaimsTransformations* XML element is also absent here.

The following XML snippet illustrates how to define a technical profile for a WS-Fed token issuer:

```
<TechnicalProfile Id="WsFedIssuer">
  <DisplayName>Token Issuer</DisplayName>
  <Protocol Name="None"/>
  <OutputTokenFormat>SAML11</OutputTokenFormat>
  <Metadata>
    <Item Key="IssuerUri">[B2C_ADFS_ISSUER_URI]</Item>
  </Metadata>
  <CryptographicKeys>
    <Key Id="SamlAssertionSigning" StorageReferenceId="B2CSigningCertificate" />
    <Key Id="SamlMessageSigning" StorageReferenceId="B2CSigningCertificate" />
  </CryptographicKeys>
  <InputClaims />
  <OutputClaims />
</TechnicalProfile>
```

Specifying a technical profile for a JWT token issuer

The JSON Web Token (JWT) format is a compact token format that is especially apt for REST-based development. JWT use is growing, and products supporting the format are increasingly common in the industry.

> **Note** For more information about and details that pertain to the JWT format, see the RFC 7519 JSON WEB TOKEN (JWT)[42] (updated by the RFC 7797 JSON WEB SIGNATURE (JWS) UNENCODED PAYLOAD OPTION[43]).

A suitable technical profile definition for a JWT token issuer implies the followings.

The *Name* attribute of the *Protocol* XML element has to be set to **None** as per *ProtocolName* enumeration in the custom policy XML schema - this should become default overtime, but **None** will be deprecated - while the *OutputTokenFormat* XML element has to be set to **JWT** as per *TokenFormat* enumeration in the custom policy XML schema.

[42] JSON WEB TOKEN (JWT): http://tools.ietf.org/html/rfc7519
[43] RFC 7797 JSON WEB SIGNATURE (JWS) UNENCODED PAYLOAD OPTION: https://tools.ietf.org/html/rfc7797

The following keys may be present in the *Metadata* XML element that will be absent if not set:

> **Note** For more information, see article AZURE ACTIVE DIRECTORY B2C: MANAGE SSO AND TOKEN CUSTOMIZATION WITH CUSTOM POLICIES[44].

Item key	Required	Description
RequestOperation	False	Allow if specified to use a remote token issuer service in lieu of the one provided by the Azure AD B2C service. See section § *Using a remote token issuer service*. Type: String Value: Must be set to **HttpRequest**.
client_secret	False	Specify a cryptographic key to use if a remote token issuer service is being used. Define the key and algorithm.
token_lifetime_secs	False	Specify the access token lifetime in seconds. Type: Integer Default value: 3600 seconds (60 minutes)
id_token_lifetime_secs	False	Specify the refresh token lifetime in seconds. Type: Integer Default value: 1209600 seconds (14 days)
refresh_token_lifetime_secs	False	Specify the refresh token sliding window lifetime in seconds. If you don't want to enforce a sliding window lifetime, use the key below instead. Type: Integer Default value: 7776000 (90 days).
allow_infinite_rolling_refresh_token	False	Allow to not enforce a sliding window lifetime. Type: Boolean Default value: false.
IssuanceClaimPattern	False	Allow to change the Issuer (iss) claim inside the JWT token Type: String Value: one of the following types: • **AuthorityAndTenantGuid**. • **AuthorityWithTfp**.
AuthenticationContextReferenceClaimPattern	False	Allow to set a claim representing the policy: TFP (trust framework policy). This is the recommended approach. If the key is omitted, the value of the claim is ACR (authentication context reference). Type: String Value: None.

[44] AZURE ACTIVE DIRECTORY B2C: MANAGE SSO AND TOKEN CUSTOMIZATION WITH CUSTOM POLICIES: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-manage-sso-and-token-configuration

AuthenticationContextReferenceClaimPattern	*False*	*Allow to set a claim representing the policy: TFP (trust framework policy). This is the recommended approach. If the key is omitted, the value of the claim is ACR (authentication context reference).* *Type: String* *Value: None.*

In addition to the above, the Subject (sub) claim is set by default to ObjectID. If you would like to switch this to Not Supported, replace the XML element <OutputClaim ClaimTypeReferenceId="objectId" PartnerClaimType="sub" /> with this one <OutputClaim ClaimTypeReferenceId="sub" /> in the relying party (RP) policy.

The following key must be present in the *CryptographicKeys* XML element:

Key	Required	Description
issuer_secret	True	Specify the issuer secret, i.e. a default key for the JWT token issuer. Define the algorithm that will be used for signing the JWT tokens. If refer to a symmetric key (referenced as "oct" in the RFC 7517 JSON Web Key (JWK)[45] standard, which means binary secret), the signature will be "HS256". Conversely, if refer to a RSA key (referenced as "RSA" in the JWK standard), the signature will be "RS256". "oct" and "RSA" are currently the only key types supported. If multiple keys are specified in the key container, the last key will be used as the signature key. This mechanism may be extended in the future with validity parameters in the key container.

Both the *InputClaims* and *OutputClaims* XML elements are empty or absent. Likewise, the *OutputClaimsTransformations* XML element is also absent here.

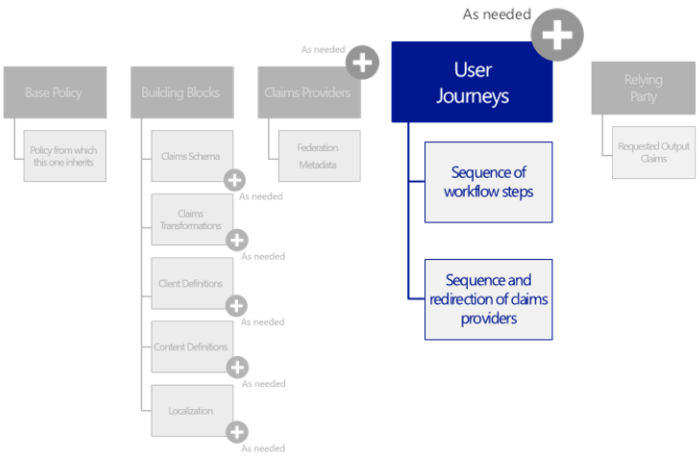
The following XML snippet illustrates how to define a technical profile for a JWT token issuer:

```
<TechnicalProfile Id="JwtIssuer">
  <DisplayName>JWT Issuer</DisplayName>
  <Protocol Name="None" />
  <OutputTokenFormat>JWT</OutputTokenFormat>
  <Metadata>
     <Item Key="token_lifetime_secs">3600</Item>
     <Item Key="id_token_lifetime_secs">3600</Item>
     <Item Key="refresh_token_lifetime_secs">1209600</Item>
     <Item Key="rolling_refresh_token_lifetime_secs">7776000</Item>
     <Item Key="IssuanceClaimPattern">AuthorityAndTenantGuid</Item>
     <Item Key="AuthenticationContextReferenceClaimPattern">None</Item>
  </Metadata>
  <CryptographicKeys>
    <Key Id="issuer_secret" StorageReferenceId="PairwiseClientSecret" />
  </CryptographicKeys>
</TechnicalProfile>
```

Now that you have some background in the definition of the various possible type of claims provider, let's consider the definition of the user journeys as part of the policy XML file.

[45] RFC 7517 JSON WEB KEY (JWK): https://tools.ietf.org/html/rfc7517

Specifying the user journeys



User journeys specify explicit paths through which a policy allows a relying party application to obtain the desired claims for a user. The user is taken through these paths to retrieve the claims that are to be presented to the relying party. In other words, user journeys define the business logic of what an end user goes through as the Azure AD B2C Identity Experience Framework processes the request.

These user journeys should essentially be considered as templates available to satisfy the core need of the various replying parties of the community of interest. As such, they greatly facilitate the definition the relying party part of a policy (See next section).

A policy can define multiple user journeys. Each user journey is a sequence of orchestration steps (see below). A policy needs to define the user journey it enforces by picking one of them.

To define the user journeys supported by the policy, a *UserJourneys* XML element must be declared under the top-level XML element of the policy XML file. This element is optional.

This element contains the following XML element:

XML element	Occurrences	Description
UserJourney	1:n	Declare a user journey that defines all the constructs necessary for a complete user flow.

Each *UserJourney* XML elements contain in turn the following attribute:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular user journey and reference it from other XML elements in the policy XML file.

And the following XML elements:

XML element	Occurrences	Description
AssuranceLevel	0:1	Specify a measurement of identity assurance (LOA) when the claims are presented to the relying party at the conclusion of the orchestration steps contained in the user journey. Note: LOP is a privacy metric that is not yet widely adopted. In the absence of such a metric, the policy doesn't include any specific information of privacy conformance level. A reference to the privacy policy may be simply specified in the policy documentation. Type: String
PreserveOriginalAssertion	0:1	Specify if the original assertion should be preserved. Claims are presented to the relying party Application in a token generated by Azure AD B2C. However, a user journey may state, using a true or a false for this element, that the original assertion which was returned from the claims provider(s) must also be preserved so that if needed, it can be looked at by relying party for auditing or diagnostic purposes. Type: Boolean
OrchestrationSteps	0:1	Specify an orchestration sequence that must be followed through for a successful transaction, i.e. a complete user flow. Thus, every user journey consists of an ordered list of orchestration steps that are executed in sequence. If any step fails, the transaction fails. See later in this document.
ClientDefinition	0:1	Reference settings definition section that determines the client behavior.
CryptographicKeys	0:1	Specify a list of cryptographic keys used in this user journey.

These elements may be declared for any given user journey.

The *ClientDefinition* XML element in the above table contains the following XML attribute:

Attribute	Required	Description
ReferenceId	True	Reference the identifier of the policy to use.

The *CryptographicKeys* XML element in the above table contains the following XML elements:

XML element	Occurrences	Description
Key	1:n	Represent a cryptographic key that are used within the policy. Since this is a sensitive secret, the actual cryptographic key is stored outside of the policy and would generally reside in a system deemed secure for cryptographic storage. For more information, see section § *Managing your key containers for Trust Framework (policies)* in the third document of this series.

The subsections describe how to define a user journey via the *UserJourney* XML element – along with the other XML sub-elements - that specifically represents a given user journey supported by the considered policy.

Providing property information for a given user journey

As illustrated above, the information property for a given technical policy are conveyed by some of the mandatory nested XML element of the *UserJourney* XML element. More specifically, to provide such an information, the *Id*, *AssuranceLevel*, and *PreserveOriginalAssertion* XML elements should be filled in.

The following XML snippet illustrates the property information for a given user journey:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>

    <UserJourney Id="B2CSignUp_2FA">
      <AssuranceLevel>FAL-2</AssuranceLevel>
      <PreserveOriginalAssertion>true</PreserveOriginalAssertion>
      …
    </UserJourney>

    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Specifying the orchestration steps for a given user journey

To support the given type of transaction, a user journey is further represented as an orchestration sequence that must be followed through for a successful transaction. Thus, every user journey consistent of an ordered list of *orchestration steps* that are executed in sequence. If any step fails, the transaction fails.

These orchestration steps reference both the building blocks and the claims providers allowed in the policy XML file. Any orchestration step that is responsible to show/render end-user user experience also has a reference to the corresponding content definition identifier (see section § *Specifying the content definitions* earlier in this document).

To specify the ordered list of orchestration steps, an *OrchestrationSteps* XML element must be declared as part of the policy. This element is mandatory.

This element contains the following XML element:

XML element	Occurrences	Description
OrchestrationStep	1:n	Define a given ordered orchestration step in the overall technical policy choreography to sustain.

A user journey contains at least an orchestration step.

Each orchestration step is modelled by an *OrchestrationStep* XML element. This element contains the following attributes:

Attribute	Required	Description
Order	Yes	Specify the order in list of orchestration steps. Type: Number
Type	*Yes*	Specify the type of the orchestration step. Type: String (enumeration) Value: one of the following types as per *OrchestrationStepType* enumeration in the custom policy XML schema: • **ConsentScreen**. Indicate that the orchestration step presents text to the user to which the user must consent. • **ClaimsProviderSelection**. Indicate that the orchestration step presents various claims providers to the user for the user to select one. • **CombinedSignInAndSignUp**. Indicate that the orchestration step presents a combined social provider sign-in and local account sign-up page. • **ClaimsExchange**. Indicate that the orchestration step exchanges claims with a claims provider. • **ReviewScreen**. Indicates that the orchestration step presents a review screen for the user to review the claims which the user must accept. • **SendClaims**. Indicate that the orchestration step sends the claims to the relying party with a token issued by a claims issuer. • **UserDialog**. Indicates that the orchestration step presents a user dialog to the user for the capturing of information. • **Noop**. Indicate that the orchestration step does nothing and is included to cope with errors in layering.
ContentDefinitionReferenceId	*No*	Specify a valid content identifier for a reference to an external content that the orchestration step can display to the user. Due to formatting and localization concerns Azure AD B2C has to deal with, this enables representing the actual content in a more suitable medium, such as an HTML page or a document. Type: String
CpimIssuerTechnicalProfileReferenceId	*No*	Used on *SendClaims* steps to define the technical profile identifier of the claims provider that will issue the token for the relying party. If absent, no relying party token will be created. Type: String

The above five types of orchestration steps may occur multiple times in the ordered list.

Each *OrchestrationStep* XML element also contains the following XML elements depending on its type:

XML element	Occurrences	Description
Preconditions	0:n	Define a list of preconditions that must be satisfied for the orchestration step to execute.
ClaimsProviderSelections	0:n	Define a list of claims provider selection options for the orchestration step.
ClaimsExchanges	0:n	Define a list of claims exchanges for the orchestration step.

The *Preconditions* XML element contains the following XML element:

XML element	Occurrences	Description
Precondition	0:n	Depending on the technical profile being used, either redirect the user's client corresponding to the claims provider selection that the user may have selected, or make a server call to exchange claims.

Each *Precondition* XML element contains the following attributes:

Attribute	Required	Description
Type	Yes	Specify the type of check or query to perform for this precondition to perform. Type: String Value: one of the following types as per *PreconditionType* enumeration in the custom policy XML schema: • **ClaimsExist**. Specify that the actions should be performed if the specified claims exist in the user's current claim set. • **ClaimEquals**. Specify that the actions should be performed if the specified claim exists and its values is equal to the specified value.
ExecuteActionsIf	Yes	Specify if the actions in this precondition should be performed if the test is true or false. Value: Boolean

And the following XML elements:

XML element	Occurrences	Description
Value	1:n	Specify the data that is used by the check. For example, if the Type of this check is **ClaimsExist**, this field will specify a *ClaimTypeReferenceId* to query for.
Action	1:n	Specify the action that should be taken if the precondition check within an orchestration step is true. Value: one of the following types as per *PreconditionActionType* enumeration in the custom policy XML schema:

- **SkipThisOrchestrationStep**. Specify that the associated OrchestrationStep should not be executed.

The following XML code snippet illustrates the use of the *Preconditions* XML elements:

```
</Preconditions>
  <Precondition Type="ClaimEquals" ExecuteActionsIf="true">
    <Value>authenticationSource</Value>
    <Value>socialIdpAuthentication</Value>
    <Action>SkipThisOrchestrationStep</Action>
  </Precondition>
</Preconditions>
```

The *ClaimsProviderSelections* XML element contains the following XML element:

XML element	Occurrences	Description
ClaimsProviderSelection	0:n	Shows options for the selection between various claims providers in a given step.

Each *ClaimsProviderSelection* XML element contains the following attribute:

Attribute	Required	Description
TargetClaimsExchangeId	False	Specify a machine understandable identifier that is used to uniquely identify this particular claims exchange step, and reference it in a given step. Type: String
ValidationClaimsExchangeId	False	Specify a machine understandable identifier that is used to uniquely identify this particular validation claims exchange step, and reference it in a given step. Type: String

The *ClaimsExchanges* XML element contains the following attribute:

Attribute	Required	Description
UserIdentity	False	Value: Boolean (default: false)

Along with the following XML element:

XML element	Occurrences	Description
ClaimsExchange	0:n	Depending on the technical profile being used, either redirect the user's client corresponding to the *ClaimsProviderSelection* that the user may have selected, or makes a server call to exchange claims.

Each *ClaimsExchange* XML element contains in turn the following attributes:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify this particular claims exchange step, and reference it from a claims provider selection step in the policy XML file.
TechnicalProfileReferenceId	True	Specify the unique identifier of the technical profile which is used for claims exchange.

The usage of the above XML elements is illustrated in the next sections.

The following XML snippet illustrates how to specify multiple orchestration steps (three in this case):

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>

         <OrchestrationStep Order="1" Type="ClaimsProviderSelection">
           …
         </OrchestrationStep>
         <OrchestrationStep Order="2" Type="ClaimsExchange">
           …
         </OrchestrationStep>
         <OrchestrationStep Order="3" Type="ClaimsExchange">
           …
         </OrchestrationStep>
         <OrchestrationStep Order="4" Type="SendClaims" … />

       <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

The following sections provides additional information on these XML elements along with a XML code snippet to illustrate how to create such elements.

Displaying a consent screen

To display a consent screen to user as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *ConsentScreen* should be added to the list of the orchestration steps of the journey.

This element must contain the *ContentDefinitionReferenceId* attribute that refer to the content definition to use.

The following XML snippet illustrates how to display a consent screen:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="B2CSignUp_2FA">
       …
       <OrchestrationSteps>
         <OrchestrationStep Order="1" Type="ConsentScreen" ContentDefinitionReferenceId="api.consent">
         …
      <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Selecting a claims provider

This is a classic *Home Realm Discovery* (HRD)/*Where Are You From* (WAYF) screen. It is required when multiple identity providers are supported for a given user journey.

To give an option to the user to select one of the accredited claims providers to get claims from as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *ClaimsProviderSelection* should be added to the list of the orchestration steps of the journey.

This element must contain the *ContentDefinitionReferenceId* attribute that refer to the content definition to use. This element may also contain a *ClaimsProviderSelections* XML element.

The following XML snippet illustrates how to select a claims provider:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>
         <OrchestrationStep Order="1" Type="ClaimsProviderSelection" ContentDefinitionReferenceId="api.idpselections">
           <ClaimsProviderSelections>
             <ClaimsProviderSelection TargetClaimsExchangeId="SignUpWithLogonEmailExchange" />
             <ClaimsProviderSelection TargetClaimsExchangeId="GoogleExchange" />
           </ClaimsProviderSelections>
         <OrchestrationStep>
         …
      <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Displaying a combined social provider sign-in and local account sign-up page

To display a combined social provider sign-in and local account sign-up page to user as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *CombinedSignInAndSignUp* should be added to the list of the orchestration steps of the journey. This element may contain a nested *ClaimsExchanges* XML element. This element is covered in the next section.

The following XML snippet illustrates how to display a combined social provider sign-in and local account sign-up page to user:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>

         <OrchestrationStep Order="1" Type="CombinedSignInAndSignUp" ContentDefinitionReferenceId="api.signinandsignup">
           <ClaimsExchanges>
             <ClaimsExchange Id="LogonDiscovery" TechnicalProfileReferenceId="SelfAsserted-LogonIdentifierDiscovery" />
           </ClaimsExchanges>
         </OrchestrationStep>

         …
      <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Exchanging claims with a claims provider

To exchange claims with a claims provider as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *ClaimsExchange* should be added to the list of the orchestration steps of the journey.

The type *ClaimsExchange* encompasses the following orchestration steps of the journey:

1. **Federate with an identity provider**. Either there is a single identity provider to use or an *ClaimsProviderSelections* orchestration step gives the end-user selection. So this step is modelled as a *ClaimsExchange* orchestration step.
2. **Read/Write to a directory provider**. Directory provider such as Azure AD is also modelled as a claims provider. So, this step is also modelled as a *ClaimsExchange* orchestration step.
3. **Read/Write to an attribute provider/verifier**. Attribute providers/verifiers are also modelled as a claims provider. So, this step is modelled as a *ClaimsExchange* orchestration step.
4. **Collect user asserted attributes**. Collecting user entered information in the flows is very common, e.g. collecting application specific information during a sign-up user journey. In such a context, self-asserted attributes' claims provider is also modelled as a *ClaimsExchange* orchestration step. This step shows a UI.

5. **Step up with multifactor authentication (MFA)**. Modelled a PhoneFactor-proxy provider as a claims provider, so this is a *ClaimsExchange* orchestration step as well. In order to call the phone or allow phone input during registration, this step also shows a UI.

This *OrchestrationStep* XML element of type *ClaimsExchange* may contain a *Preconditions* XML element and must contain a *ClaimsExchanges* XML element.

The following XML snippet illustrates how to exchange claims with a claims provider:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>
         …

         <OrchestrationStep Order="2" Type="ClaimsExchange">
           <ClaimsExchanges>
             <ClaimsExchange Id="SignUpWithLogonEmailExchange"
                             TechnicalProfileReferenceId="LocalAccountSignUpWithLogonEmail" />
             <ClaimsExchange Id="GoogleExchange" TechnicalProfileReferenceId="Google-OAuth2" />
           </ClaimsExchanges>
         </OrchestrationStep>

         …
      </OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Displaying a review screen

To display a review screen to user as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *ReviewScreen* should be added to the list of the orchestration steps of the journey.

The following XML snippet illustrates how to display a review screen:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="B2CSignUp_2FA">
       …
       <OrchestrationSteps>

         <OrchestrationStep Order="1" Type="ReviewScreen">
           …
         </OrchestrationStep>

         …
      </OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
```

```
  …
</TrustFrameworkPolicy>
```

Sending claims with a token issued by a claims issuer

To send claims with a token issued by a claims issuer as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *SendClaims* should be added to the list of the orchestration steps of the journey. This element must contain the *CpimIssuerTechnicalProfileReferenceId* attribute that refer to the claims issuer to use.

The XML element generally constitute the last step of a given user journey.

The following XML snippet illustrates how to exchange claims with a claims provider:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>
         …
         <OrchestrationStep Order="8" Type="SendClaims" CpimIssuerTechnicalProfileReferenceId="JwtIssuer" />
       <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Displaying a dialog

To display a user dialog to collect information as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *UserDialog* should be added to the list of the orchestration steps of the journey.

The following XML snippet illustrates how to display a review screen:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>
         <OrchestrationStep Order="1" Type="UserDialog">
           …
         </OrchestrationStep>
         …
       <OrchestrationSteps>
    </UserJourney>
    …
```

```
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

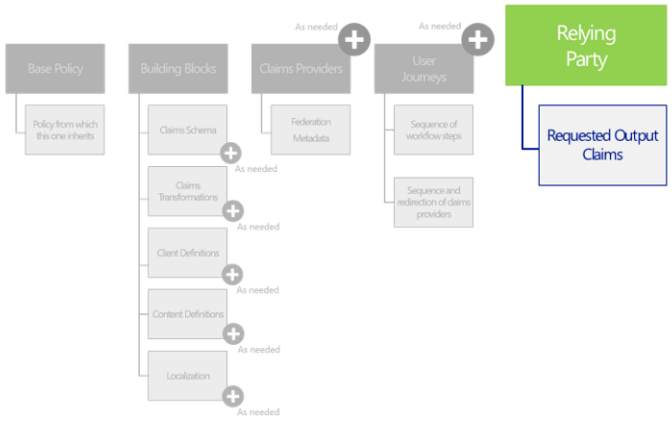
Coping with errors

To cope with errors as part of the explicit path definition of a given user journey, an *OrchestrationStep* XML element of type *Noop* should be added to the list of the orchestration steps of the journey.

The following XML snippet illustrates how to display a review screen:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
           PolicySchemaVersion="0.3.0.0"
           …>
  …
  <UserJourneys>
    <UserJourney Id="SomeUserJourney">
       …
       <OrchestrationSteps>

         <OrchestrationStep Order="1" Type="Noop">
           …
         </OrchestrationStep>

         …
      <OrchestrationSteps>
    </UserJourney>
    …
  </UserJourneys>
  …
</TrustFrameworkPolicy>
```

Specifying the relying party



The relying party information chooses the user journey to enforce for the current request. It also chooses the list of claims the relying party application would like to get as part of the issued token.

Multiple applications can use a given policy. They will all receive the same token with claims and the end user will go through the same user journey.

Conversely, a single application can use multiple policies. This allows the application to achieve functionality such as basic sign-in, step-up, sign-up, etc.

To specify the relying party information, a *Relying Party* XML element must be declared must be declared under the top-level XML element of the policy XML file. This element is optional.

This element contains the following XML elements:

XML element	Occurrences	Description
DefaultUserJourney	0:1	Define the default user journey for the relying party application.
UserJourneyBehaviors	0:1	Control the scope of various user journey behaviors.
TechnicalProfile	0:1	Define a technical profile supported by the relying party application. The technical profile provides in this context a contract for the relying party application to contact Azure AD B2C.

These above *DefaultUserJourney* and *TechnicalProfile* elements must be declared for any given *RelyingParty* XML element.

The *DefaultUserJourney* XML element contains in turn the following attribute:

Attribute	Required	Description
ReferenceId	True	Specify a machine understandable identifier that is used to uniquely reference a particular user journey in the policy XML file.

The *UserJourneyBehaviors* XML element contains the following XML elements:

XML element	Occurrences	Description
JourneyInsights	0:1	Define the Application Insights configuration. For more information, see APPLICATION INSIGHTS DOCUMENTATION[46].
SingleSignOn	0:1	Control the scope of the single sign-on (SSO) behavior of a user journey.
SessionExpiryType	0:1	Control the whether the session is rolling or absolute. Value: one of the following values as per *SessionExpiryTypeTYPE* enumeration in the custom policy XML schema for the names of the valid values the single sign-on session type: • **Rolling**. • **Absolute**.
SessionExpiryInSeconds	0:1	Control the time of the session expiry in seconds. Value: Integer
ContentDefinitionParameters	0:1	Specify the list of key value pairs to be appended to the content definition load Uri.

The *JourneyInsights* XML element contains in turn the following attribute:

Attribute	Required	Description
InstrumentationKey	True	Specify your instrumentation key for Application Insights. Type: String Value: your instrumentation key
DeveloperMode	True	Specify if the developer mode is active for the policy. Setting it to true is good for development but constrained at high volumes because it tells Application Insights to expedite the telemetry through the processing pipeline Type: Boolean
ClientEnabled	True	Define if client-side scripts are sent to Application Insights for tracking page view and client-side errors. Type: Boolean
ServerEnabled	True	Define if the existing UserJourneyRecorder JSON are sent as a custom event to Application Insights for tracking page view and client-side errors. Type: Boolean

[46] APPLICATION INSIGHTS DOCUMENTATION: https://docs.microsoft.com/en-us/azure/application-insights/

TelemetryVersion	True	Specify the version for the telemetry Type: string Value: 1.0.0

The following XML snippet illustrates how to define a **campaignId** query string parameter name:

```
<UserJourneyBehaviors>
   <JourneyInsights InstrumentationKey="you_instrumentation_key"
                    DeveloperMode="true"
                    ClientEnabled="false"
                    ServerEnabled="true"
                    TelemetryVersion="1.0.0" />
</UserJourneyBehaviors>
```

The *SingleSignOn* XML element contains in turn the following attributes:

Attribute	Required	Description
Scope	True	Define the scope of the single sign-on (SSO) behavior and thus govern the user's sign in experience across multiple apps and policies in the tenant. Type: String Value: one of the following values as per *UserJourneyBehaviorScopeType* enumeration in the custom policy XML schema: • **Tenant**. Indicates that the behavior is applied for all policies in the tenant. For example, a user being put through two policy journeys for a given tenant will not be prompted for identity provider selection. • **Application**. Indicate that the behavior is applied for all policies for the application making the request. For example, a user being put through two policy journeys for a given application will not be prompted for identity provider selection. • **Policy**. Indicate that the behavior only applies to a policy. For example, a user being put through two policy journeys for a given trust framework will be prompted for identity provider selection when switching between policies. • **Disabled**. Indicate that the behavior is suppressed. For example, in the case of SSO, no session is maintained for the user and the user will always be prompted for identity provider selection.
KeepAliveInDays	False	Specify the maximum time period for the keep me signed in (KMSI) and before. Type: Integer Default value: 14 (days).

The following XML snippet illustrates how to change your session behavior and single sign-on (SSO) configuration:

```
<UserJourneyBehaviors>
   <SingleSignOn Scope="Application" KeepAliveInDays="14" />
   <SessionExpiryType>Absolute</SessionExpiryType>
   <SessionExpiryInSeconds>86400</SessionExpiryInSeconds>
</UserJourneyBehaviors>
```

The *ContentDefinitionParameters* XML element contains the following XML elements:

XML element	Occurrences	Description
Parameter	0:n	Define a query string parameter to be used in a request to the Identity Experience Framework in Azure AD 2BC to run a custom policy. Type: String

The *Parameter* XML element contains in turn the following attribute:

Attribute	Required	Description
Name	True	Specify an identifier for the parameter.

The value of *Parameter* XML element is any query string parameter name as part of a request. It should be specified as follow:

{OAUTH-KV:<your_query_string_paramater_name>}

Where <your_query_string_paramater_name>is the name of the query string parameter.

The following XML snippet illustrates how to define a **campaignId** query string parameter name:

```
<UserJourneyBehaviors>
   <ContentDefinitionParameters>
      <Parameter Name="campaignId">{OAUTH-KV:campaignId}</Parameter>
   </ContentDefinitionParameters>
</UserJourneyBehaviors>
```

The *TechnicalProfile* XML element basically follows the structure outlined before for a technical profile. Thus, it contains in turn the following attributes:

Attribute	Required	Description
Id	True	Specify a machine understandable identifier that is used to uniquely identify a particular technical profile in the policy XML file.

And the following XML elements:

XML element	Occurrences	Description
DisplayName	0:1	Specify the human understandable name of the technical profile that can be displayed to the users. Type: String
Description	0:1	Specify a human understandable description of the technical profile that can be displayed to the users. Type: String
Protocol	0:1	Specify the protocol used for the federation.
Metadata	1:1	Specify the metadata utilized by the protocol for communicating with the endpoint in the course of a transaction to plumb "on the wire" interoperability between the relying party and other community participants. Type: collection of *Item* of key/value pairs.

OutputClaims	0:1	Specify an optional list of the claim types that are taken as output in the technical profile. Each of these elements contains reference to a *ClaimType* already defined in the *ClaimsSchema* section or in a policy from which this policy XML file inherits.
OutputTokenFormat	0:1	Specify the format of the output token. Type: String (enumeration) Value: one of the following types as per *TokenFormat* enumeration in the custom policy XML schema. See above.
SubjectAuthenticationRequirements	0:1	Specify the requirements regarding the conscious and active participation of the subject in authentication
SubjectNamingInfo	0:1	Control the production of the subject name in tokens (e.g. SAML) where subject name is specified separately from claims.

The *Protocol* XML element in the above table contains the following attributes:

Attribute	Required	Description
Name	True	Specify the name of a valid protocol supported by Azure AD B2C that is used as part of the technical profile. Type: String (enumeration) Value: one of the following types as per *ProtocolName* enumeration in the custom policy XML schema: • **OAuth1**. OAuth 1.0 protocol standard as per IETF specification. • **OAuth2**. OAuth 2.0 protocol standard as per IETF specification. • **SAML2**. SAML 2.0 protocol standard as per OASIS specification. • **OpenIdConnect**. OpenID Connect 1.0 protocol standard as per OpenID foundation specification. • **WsFed**. WS-Federation (WS-Fed) 1.2 protocol standard as per OASIS specification. • **WsTrust**. WS-Trust 1.3 protocol standard as per OASIS specification.

As already introduced, the *OutputClaims* XML elements contain the following XML elements:

XML element	Occurrences	Description
OutputClaim	0:n	Specify the name of an expected claim type in the supported list for the policy to which the relying party subscribes. This claim serves as an output for the technical profile.

Each *OutputClaim* XML element contains the following attributes:

Attribute	Required	Description
ClaimTypeReferenceId	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String
DefaultValue	False	Specify a default value if not set. Type: String

PartnerClaimType	False	Specify the partner claim type. Type: String
Required	False	Specify this claim is required. Type: String

The *SubjectAuthenticationRequirements* XML element in the above table contains the following attributes:

Attribute	Required	Description
TimeToLive	True	Specify the maximum number of minutes cached credentials can be used following an active authentication by the subject. Type: Integer
ResetExpiryWhenTokenIssued	Optional	Specify how the expiry time is set. Type: Boolean. Default is False. If True then whenever a token is issued (even using a cached credential) the expiry time is set to the current time plus the *TimeToLive*.

The *SubjectNamingInfo* XML element in the above table contains the following attributes:

Attribute	Required	Description
ClaimType	True	Specify a reference to a *ClaimType* already defined in the *ClaimsSchema* section in the policy XML file. Type: String
NameQualifier	False	Type: String
SPNameQualifier	False	Type: String
Format	False	Type: String
SPProvidedID	False	Type: String

Considering the above explanation, the following XML snippet illustrates how to define a relying party:

```
<?xml version="1.0" encoding="utf-8"?>
<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                      xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"
                      PublicPolicyUri="http://example.com"
                      PolicySchemaVersion="0.3.0.0"
                      TenantId="litware369b2c.onmicrosoft.com"
                      PolicyId="B2C_1A_MsolActive">
  <BasePolicy>
    <TenantId>litware369b2c.onmicrosoft.com</TenantId>
    <PolicyId>B2C_1A_base-v2</PolicyId>
  </BasePolicy>
```

```
  <RelyingParty>
    <DefaultUserJourney ReferenceId="ActiveRST"/>
    <TechnicalProfile Id="PolicyProfile">
      <DisplayName>WsFedProfile</DisplayName>
      <Protocol Name="WsFed" />
      <OutputTokenFormat>SAML11</OutputTokenFormat>
      <SubjectAuthenticationRequirements TimeToLive="4" ResetExpiryWhenTokenIssued="false" />
      <Metadata>
        <Item Key="Saml2AttributeEncodingInfo">
         <![CDATA[
           <saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
             <saml2:Attribute FriendlyName="UserPrincipalName"
```

```
                            Name="IDPEmail"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
                </saml2:AttributeValue>
              </saml2:Attribute>
            </saml2:AttributeStatement>
          ]]>
        </Item>
        <Item Key="Saml11AttributeEncodingInfo">
          <![CDATA[
          <saml:AttributeStatement xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
            <saml:Attribute AttributeName="ImmutableID"
                            AttributeNamespace="http://schemas.microsoft.com/LiveID/Federation/2008/05">
              <saml:AttributeValue></saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute AttributeName="UPN" AttributeNamespace="http://schemas.xmlsoap.org/claims">
              <saml:AttributeValue></saml:AttributeValue>
            </saml:Attribute>
          </saml:AttributeStatement>
          ]]>
        </Item>
        <Item Key="PartnerEntity">https://www.litware369b2c.com/wp-content/uploads/2015/01/metadata.xml</Item>
        <Item Key="client_id">customClientId</Item>
      </Metadata>
      <OutputClaims>
        <OutputClaim ClaimTypeReferenceId="immutableId"  PartnerClaimType="ImmutableID" />
        <OutputClaim ClaimTypeReferenceId="userPrincipalName" PartnerClaimType="UPN" />
        <OutputClaim ClaimTypeReferenceId="AuthenticationContext" DefaultValue="urn:federation:authentication:windows" />
      </OutputClaims>
      <SubjectNamingInfo ClaimType="immutableId" />
    </TechnicalProfile>
  </RelyingParty>

</TrustFrameworkPolicy>
```

This concludes this sixth and last document of this series.

Appendix A. Trust Framework Policy XML schema and tools

Trust Framework Policy XML Schema

The Trust Framework policy XML schema is provided as part of the "Starter Pack" of Azure AD B2C". **The second part of the series of documents depicts how to proceed to get started with the "Starter Pack" of Azure AD B2C.**

The Trust Framework policy XML schema file *TrustFrameworkPolicy_0.3.0.0.xsd* is located under the root **Starter-Pack** folder.

If you've followed the instructions of the second part, the "Starter Pack" is located under the *C:\Code\active-directory-b2c-custom-policy-starterpack* folder.

Generating .NET Classes for serialization/deserialization purposes

The XML Schema Definition (*Xsd.exe*) tool[47] notably enables to generate .NET common language runtime classes from XSD, XDR, and XML files. This tool is used here against the above Trust Framework policy XML schema file *TrustFrameworkPolicy_0.3.0.0.xsd* to generate NET Classes for serialization/deserialization purposes.

> **Note** The XML Schema Definition tool is part of the .NET Framework tools that make it easier for you to create, deploy, and manage applications and components that target the .NET Framework. The .NET Framework tools are automatically installed with Visual Studio. The second part of the series of documents invites you to install Visual Studio 2017 (i.e. Microsoft Visual Studio Community 2017[48]) as per the prerequisites for the "Starter Pack".

To generate the .NET classes for serialization/deserialization purposes based on the aforementioned schema, proceed with the following steps:

1. Open a Developer Command Prompt for Visual Studio command prompt.

[47] XML SCHEMA DEFINITION TOOL (XSD.EXE): https://docs.microsoft.com/en-us/dotnet/standard/serialization/serialization-tools
[48] Visual Studio Community 2017: https://www.visualstudio.com/free-developer-offers/

Note The Developer Command Prompt for Visual Studio automatically sets the environment variables that enable you to easily use .NET Framework tools. The Developer Command Prompt is installed with full or community editions of Visual Studio. It is not installed with the Express versions of Visual Studio. For more information, see article DEVELOPER COMMAND PROMPT FOR VISUAL STUDIO[49].

2. From the command prompt, type the following command to run the XML Schema Definition tool:

```
xsd.exe /n:Microsoft.Cpim.Common.Persistence.Autogenerated /classes TrustFrameworkPolicy_0.3.0.0.xsd
```

Once the command completes, a file *TrustFrameworkPolicy_0.3.0.0.cs* is generated.

Generating .NET Core Classes for serialization/deserialization purposes

To generate the .NET Core classes for serialization/deserialization purposes based on the aforementioned schema, proceed as per previous section to generate a file *TrustFrameworkPolicy_0.3.0.0.cs*.

The generated class can be compiled and used in .Net Core.

Installing XML tooling for editing the custom policy XML files

This optional section suggests some XML tooling to leverage for editing the policy XML files, and more particularly Visual Studio Code (with the Azure AD B2C extension) and Notepad++ (with XML Tools).

However, nothing prevents you to also use any text editor of your choice, such as Notepad, but these tools make custom policy XML files editing easier.

Using Visual Studio Code with the Azure AD B2C extension

The Azure AD B2C extension for Visual Studio Code lets you quickly navigate through a custom policy, and create new XML elements, such as technical profiles, claim definition, etc. as discussed throughout this document. As of this writing, this extension is in Beta.

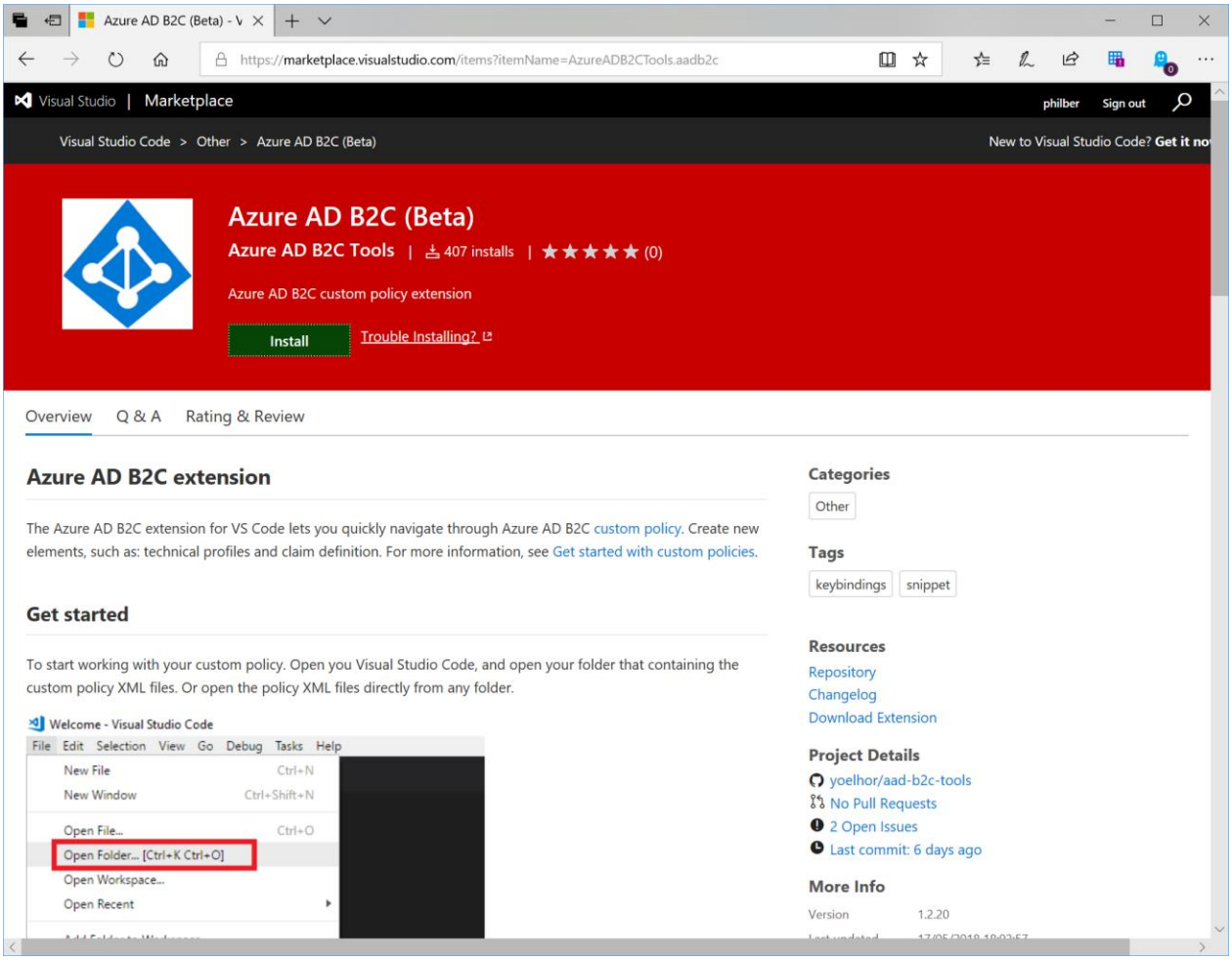
[49] DEVELOPER COMMAND PROMPT FOR VISUAL STUDIO: https://docs.microsoft.com/en-us/dotnet/framework/tools/developer-command-prompt-for-vs

Important note The extension is developed and managed by the open-source community in GitHub. The extension is not part of Azure AD B2C and it's not supported under any Microsoft standard support program or service. The extension is provided AS IS without warranty of any kind. For any issue, visit the GitHub repository[50].

To set up the extension for Visual Studio Code, proceed with the following steps:

1. Open a browsing session and navigate to :

   https://marketplace.visualstudio.com/items?itemName=AzureADB2CTools.aadb2c.



Note The second part of the series of documents invites you to install Visual Studio Code as per the prerequisites for the "Starter Pack".

2. Click **Install** and proceed with the setup the Azure AD B2C extension for VS Code as instructed.

Once the installation of the extension completes, to start working with your custom policy, proceed with the following steps:

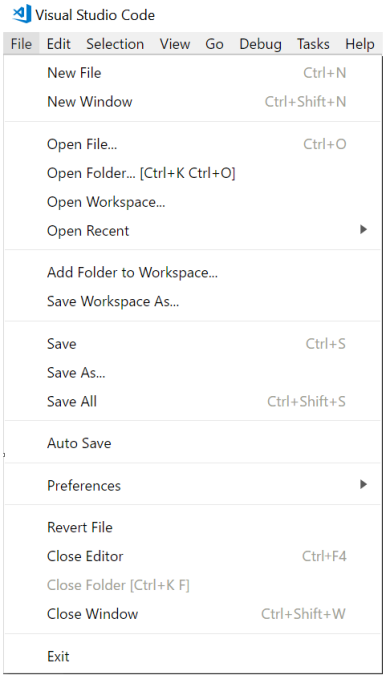
1. Open Visual Studio Code.

[50] yoelhor/aad-b2c-vs-code-extension on GitHub: https://github.com/yoelhor/aad-b2c-vs-code-extension

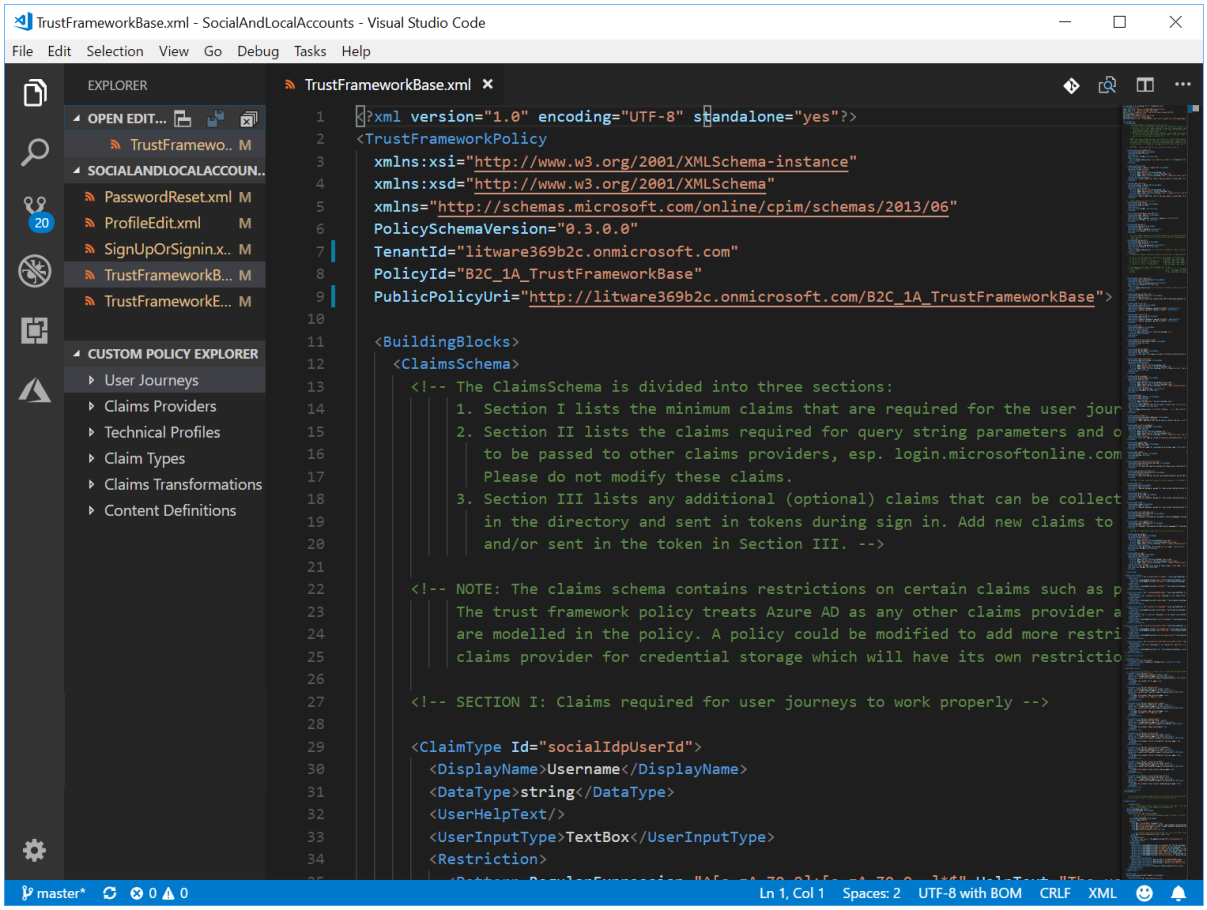
2. Open the folder that contains the custom policy XML files, for instance the *SocialAndLocalAccounts* folder for the SocialAndLocalAccounts scenario/core template provided by the "Starter Pack". See the second document of this series.

   - or-

   Open the policy XML files directly from any folder. Select for instance the *TrustFrameworkBase.xml* custom base policy provided by the "Starter Pack" for the SocialAndLocalAccounts scenario/core template.



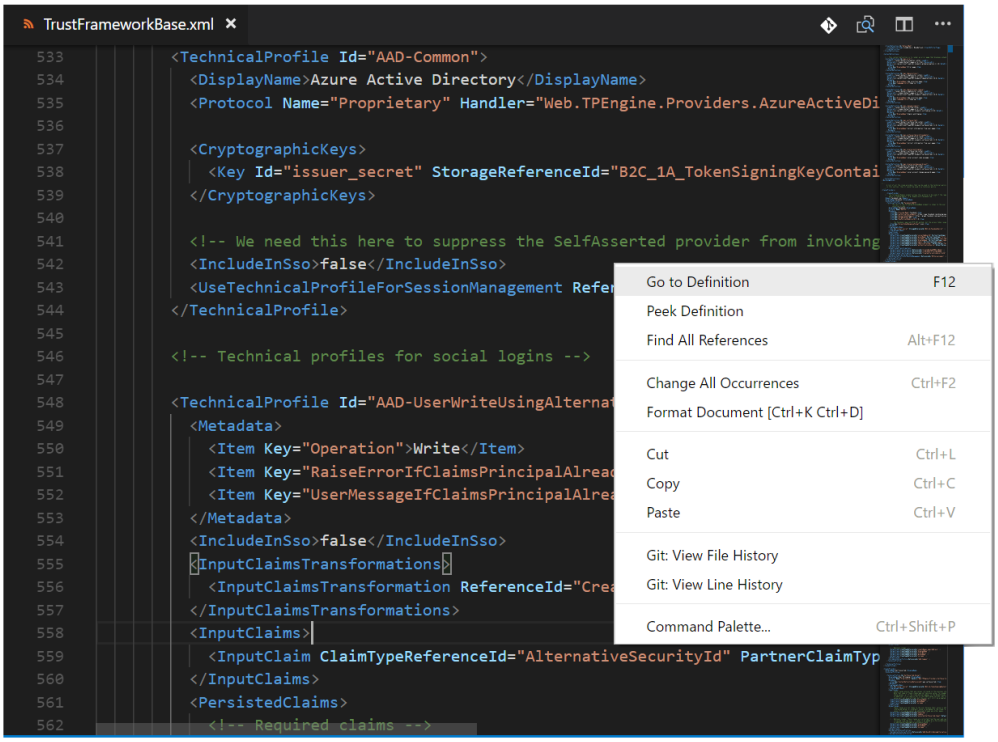
3. Regardless of the above options, and to illustrate the Azure AD B2C custom policy features, open the *TrustFrameworkBase.xml* custom base policy provided by the "Starter Pack" for the SocialAndLocalAccounts scenario/core template.



4. From **Custom policy explorer**, click on the XML element type of your choice, for instance Technical Profiles, and select the element you want to open, for instance AAD-Common.

Note	The custom policy explorer shows elements from selected file only.

5. To go any XML element definition, right-click and select **Go to definition** or **Peak definition**.



Note Go to definition navigates you to the source element in the selected file only.

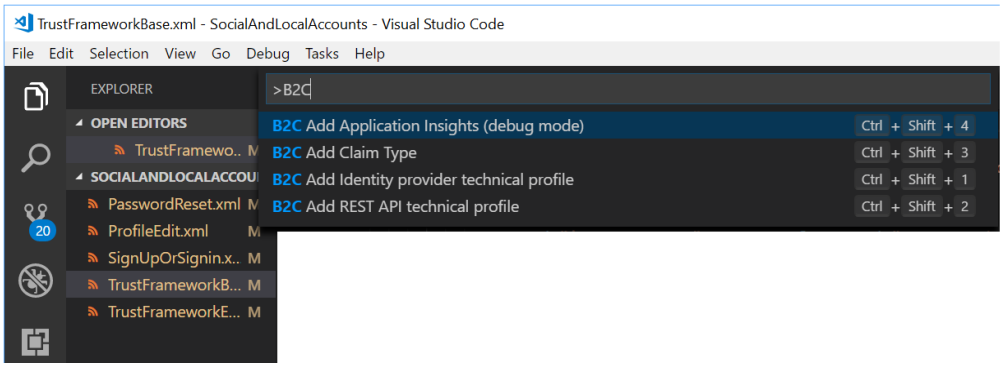
6. To search for references in your **Open folder** XML files or any XML file you open with Visual Studio Code, select **Find all references**.

The Azure AD B2C extension for VS Code also enables you to add following XML elements to your custom policy XML file:

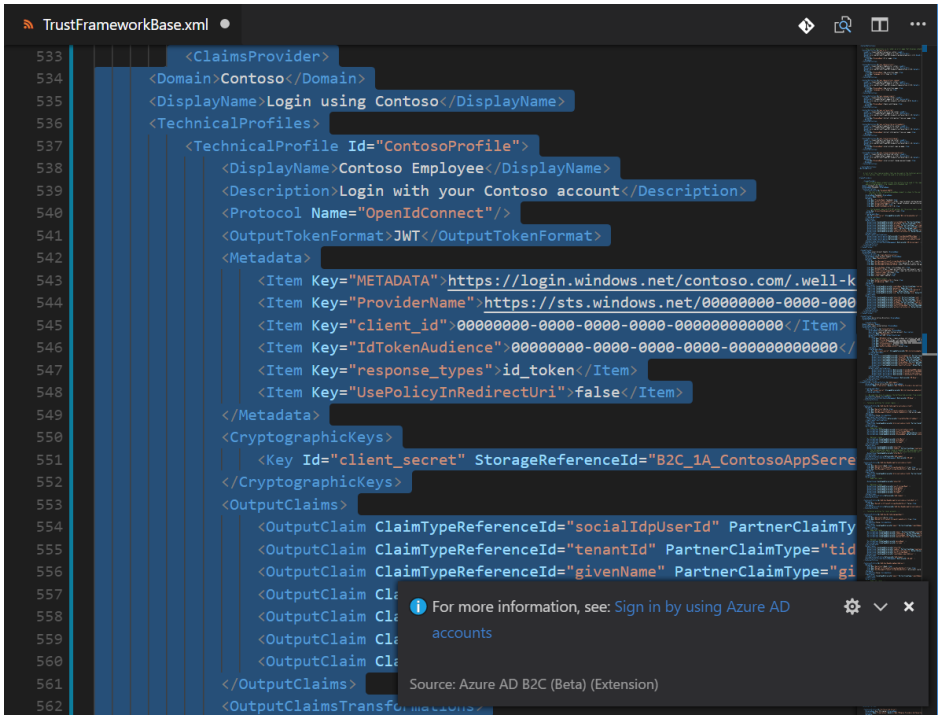
- Application Insights (debug mode).
- Claim Type.
- Identity provider technical profile.
- REST API technical profile.

To add such an XML element, proceed with the following steps:

1. Type CTRL+SHIFT+P to open the command palette.
2. Type "B2C".



3. Select for instance **Add Identity provider technical profile**. A list of supported technical profiles is displayed.
4. Select Azure AD in the provided list. After you run the command, the Azure AD B2C extension for VS Code shows you information message with a link to relevant article in the available documentation online.



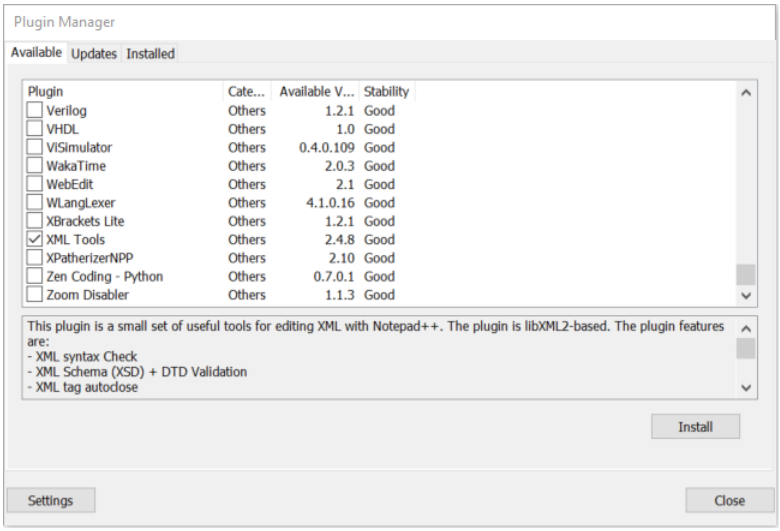
This completes our "quick tour" of the extension.

Using Notepad++ with XML Tools

Compared to the above specific extension for Visual Studio Code, the XML tools for Notepad++ "only" provide XML features.

To set up Notepad++ with XML Tools, proceed with the following steps:

1. Open a browsing session and navigate to https://notepad-plus-plus.org/download.
2. Click **Download** to install the latest version (7.5.6 as of this writing).
3. Confirm the download of the file *npp.7.5.6.Installer.exe* and execute it.
4. After the installation completes, run Notepad++.
5. Click **Plugins | Plugin Manager | Show Plugin Manager**.
6. From the list, select **XML Tools**.

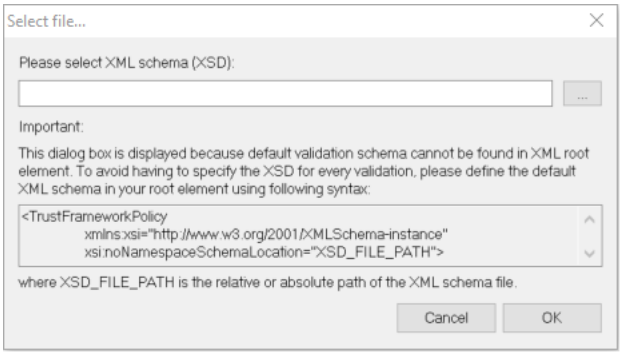


7. Click **Install**. The download and the installation of the selected plugin starts.
8. Restart Notepad++ to complete the installation steps.

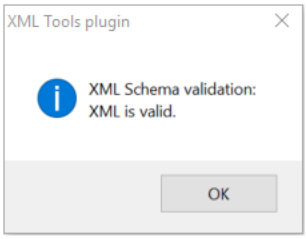
After the installation of the plugin completes, you should be able to click **Plugins** menu option in Notepad++ and see a new option **XML Tools** appear.

To validate a policy using the provided XML schema file with the custom policies of the core templates of the "Starter Pack", proceed with the following steps:

1. Click **File | Open** and open for instance the *TrustFrameworkBase.xml* custom base policy provided by the "Starter Pack" for the SocialAndLocalAccounts scenario/core template
2. Click **Plugins | XML Tools | Validate Now**. A **Select file...** dialog pops up.



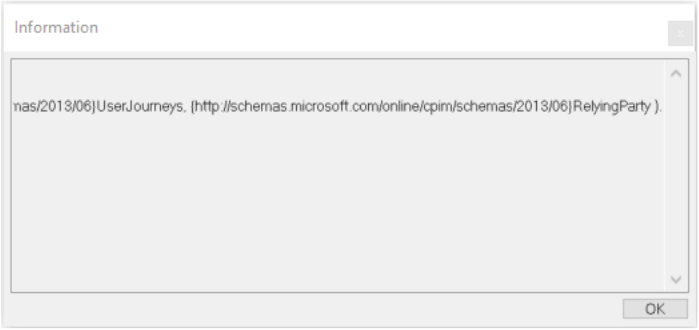
3. Specify the path to the XML schema file *TrustFrameworkPolicy_0.3.0.0.xsd* under the *Starter-Pack* folder in the text box, and then click **OK**.
4. A message box should show stating "XML Schema validation: XML is valid."



5. Add the following line at line 10 (right after that **TrustFrameworkPolicy** node:

```
<Foo></Foo>
```

6. Validate XML again by clicking **Plugins | XML Tools | Validate Now**.
7. An error message similar to the following should show:




```
Validation of current file using XML schema:


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 ).
```