(https://twitter.com/FutureProcessin

(https://plus.google.com/1109395185

(https://www.youtube.com/user/Futu

(https://www.linkedin.com/company/
processing-sp--

(http://www.goldenline.pl/firma/futu
processin

(http://www.slideshare.net/FuturePr

04.03.2015

Introduction to Claims-Based Authentication and Authorization in .NET

Show menu

TECHNOLOGIES PROCESSES, STANDARDS AND QUALITY ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/) CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)

Type

TOMASZ KRAWCZYK (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/AUTHOR/TOMASZ-KRAWCZYK/)

MAIN PAGE (https://www.future-processing.pl/blog)

THE ARTICLE DESCRIBES THE TERM OF CLAIMS-BASED AUTHENTICATION AND ITS PRINCIPLES OF OPERATION ON .NET PLATFORM AND BENEFITS OF USING THIS KIND OF APPROACH. THIS ARTICLE IS THE FIRST ONE OF THE SERIES. IN THE NEXT ONE THE IMPLEMENTATION PROCESSES AND MORE SCENARIOS WILL BE PRESENTED.

SHARE

COMMENT

CATEGORIES

BIG DATA (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CATEGORY/BIG-DATA/)

AUTHENTICATION AND AUTHORIZATION

Authentication and authorization are two closely related concepts, which are used to build security mechanism in systems and applications. Information security is the practice of protecting information from unauthorized

processing.pl/blog/category/big-data/econib/)

- Machine learning (https://www.future-processing.pl/blog/category/big-data/machine-learning/)

PROCESSES, STANDARDS AND QUALITY (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CATEGORY/PROCESSES-STANDARDS-AND-QUALITY/)

- Code Quality Assurance (https://www.future-processing.pl/blog/category/processes-standards-and-quality/static-code-analysis/)
- Development methodologies (https://www.future-processing.pl/blog/category/processes-standards-and-quality/development-methodologies/)
- Performance (https://www.future-processing.pl/blog/category/processes-standards-and-quality/performance/)
- Security (https://www.future-processing.pl/blog/category/processes-standards-and-quality/security-...)
- Software Craftsmanship

(https://twitter.com/FutureProcessin
(https://plus.google.com/1109395185
(https://www.youtube.com/user/Futu
(https://www.linkedin.com/company processing-sp--
(http://www.goldenline.pl/firma/futu processin
(http://www.slideshare.net/FuturePr

Show menu

TECHNOLOGIES PROCESSES, STANDARDS AND QUALITY ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/) CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)

Type

access, use or even modification. Confidentiality, integrity and availability (known as CIA triad) is a model designed to guide policies of information security within an organization. In this article we are focusing on confidentiality, which refers to restriction of access to sensitive information that can be achieved by authentication and authorization.

Authentication allows to verify someone's identity in the system (e.g. **Are you User X in the system Y?**). The following method, based on user and password, is still one of the most common solutions. However, nowadays there are other methods in use, especially those based on PKI (Public Key Infrastructure) or biometrics.

Generally, all authentication methods work in the same scenario – user sends some data to application (or system) which allows to identify him (e.g. username and password, certificate, finger print) and application verifies if the "user identity" exists in its id store (data base, trusted certificate store).

Authorization verifies "**what you are authorized to do (describes user permissions)**" based on user identity, which was verified in the authentication process. Many applications use roles to describe user rights (e.g. user in Admin role is allowed to create others users, user in Reporter role has right to print reports) – after positive authentication process for user identity roles are assigned. In the figure below a typical authentication and authorization process has been presented.

Figure 1. Typical authentication and authorization process.

(https://www.future-processing.pl/blog/category/processes-standards-and-quality/software-craftsmanship/)

– Software security (https://www.future-processing.pl/blog/category/processes-standards-and-quality/security/)

– Test Automation (https://www.future-processing.pl/blog/category/processes-standards-and-quality/testing-automation/)

– Test management (https://www.future-processing.pl/blog/category/processes-standards-and-quality/testing-management/)

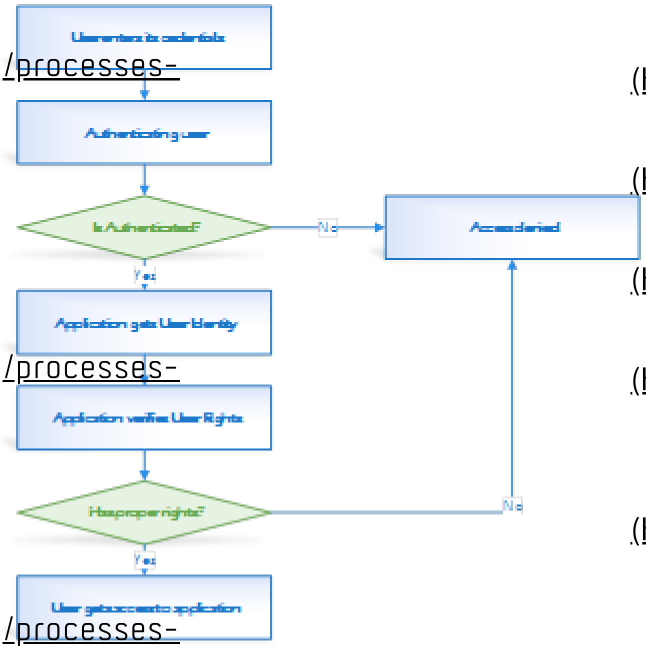
SPACE (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CATEGORY/SPACE/)

TECHNOLOGIES (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CATEGORY/TECHNOLOGIES/)

– .NET (https://www.future-processing.pl/blog/category/technologies/net-technologies/)

– Blockchain (https://www.future-processing.pl/blog/category/technologies/blockchain/)

– Cloud (https://www.future-processing.pl/blog/category/technologies/cloud/)


User enters its credentials
Authenticating user
Is Authenticated?
No
Access denied
Yes
Application gets User Identity
Application verifies User Rights
Has proper rights?
No
Yes
User gets access to application

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Typical-authentication-and-authorization-process.png)

A “classical authentication” approach means that every application has its own authentication mechanism that leads to many different problems. There are lots of ways to build identity support into an application, and each communication framework treats identity differently, with different object or storage models. One of the most important problems of this approach is dispersion of the user identity. Each application stores its own copy of user data, which is used to build user identity. Very often user data (identities) are duplicated in multiple applications or systems, generating problem of data integrity. This kind of approach to authentication does not support Single Sign On (SSO) mechanism. Nowadays, mechanism called SSO is becoming more and more popular due to possibility for allowing authentication in many applications using one application of user identity. User logs once (authenticates and authorizes) and gains access to all systems without being prompted to log in again. Using Active Directory (and authentication based on AD) partially solved problem with SSO. Despite its advantages AD approach cannot be used in every situation.

- Databases (https://www.future-processing.pl/blog/category/technologies/databases/)
- Java (https://www.future-processing.pl/blog/category/technologies/java/)
- JavaScript (https://www.future-processing.pl/blog/category/technologies/java-script/)
- Mobile (https://www.future-processing.pl/blog/category/technologies/mobile-technologies/)
- Others (https://www.future-processing.pl/blog/category/technologies/others/)
- PHP (https://www.future-processing.pl/blog/category/technologies/php/)
- Web Front-End (https://www.future-processing.pl/blog/category/technologies/web-frontend/)

Show menu

TECHNOLOGIES PROCESSES ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/) CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/) STANDARDS AND QUALITY

Type

(https://twitter.com/FutureProcessin

(https://plus.google.com/1109395185

(https://www.youtube.com/user/Futu

(https://www.linkedin.com/company processing-sp--

(http://www.goldenline.pl/firma/futu processin

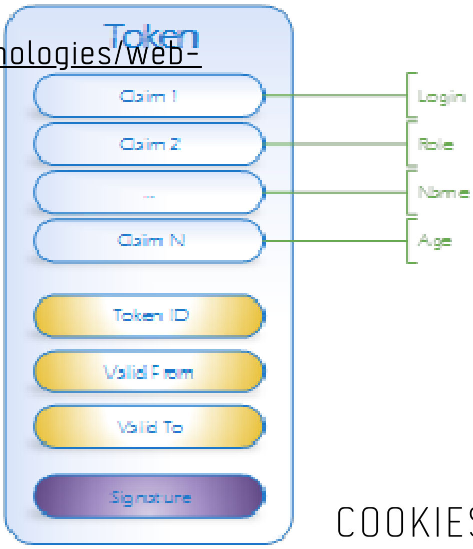
(http://www.slideshare.net/FuturePr

CLAIMS-BASED IDENTITY

One step toward to solve those problems is to stop building custom identity plumbing and user account databases into every new application. Claims-based identity is a straightforward idea, founded on a limited number of concepts: claims, tokens, identity providers, and a few more. This approach helps us to reduce the excess of identity management by providing a central point for user and role administration (providers support for Single Sign On mechanism).

Claims can contain information about the user, roles or permissions, useful to build flexible authorization model. Token contains one or more claims and every claim contains some specific information. The token is digitally signed by token issuer when it's created, so that it can be verified at the receiver end. Token can also contain additional information e.g. expiry date or id.

Figure 2. Security Token Structure.



(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-2-Security-Token-Structure.png)

User identity is stored and verified in one central place called Identity Provider. Identity Provider gives information (list of claims) about a user that are utilized in by application in authentication and authorization process.

Identity provider consists of Security Token Service and Identity Store. Identity Store contains information about

Technical Blog (https://www.future-processing.pl/blog)

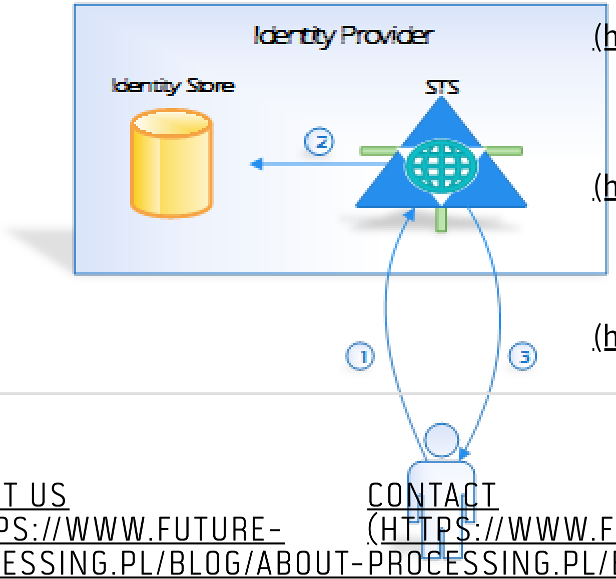
Future Processing (http://www.future-processing.com)

(https://www.facebook.com/FutureP

user identity – it can be based on database store or AD. Security Token Service (STS) is responsible for handling user requests and creating tokens. To verify and create user identity STS acquires information stored in Identity Store.

(https://twitter.com/FutureProcessin
(https://plus.google.com/110939518
(https://www.youtube.com/user/Futu
(https://www.linkedin.com/company
processing-sp--
(http://www.goldenline.pl/firma/futu
processin
(http://www.slideshare.net/FuturePr

Figure 3. Identity Provider.



Show menu
TECHNOLOGIES ▾
PROCESSES, STANDARDS AND QUALITY ▾
ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/)
CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)
Type

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-3-Identity-Provider.png)

The figure above illustrates the authentication process. The process is composed of 3 steps:

1. User (via application) sends authenticate request (with credentials) to STS

2. STS verifies user credentials, gets information about identity

3. STS creates and returns token

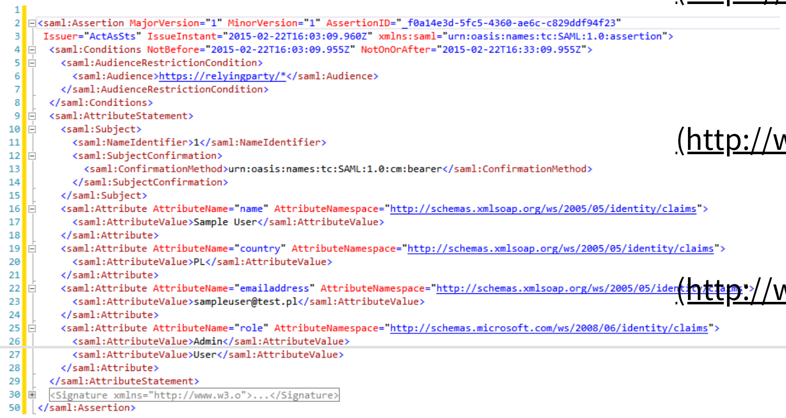
Communication between user (application) and STS is provided by using the standard protocol WS-Trust. WS-Trust is a WS-* specification and OASIS standard that provides extensions to WS-Security, specifically dealing with the issuing, renewing, and validating security tokens. The request is authenticated in some way, such as by providing a Kerberos ticket, a password from the user, certificates or something else. Typically, the request contains both, the name of the user for whom this token should be issued and a URI identifying the application the user wishes to access. During communication token is serialized using SAML- Security Assertion Markup

by
Technical Blog
(https://www.future-processing.pl/blog)
Future Processing
(http://www.future-processing.com)
(https://www.facebook.com/FutureP

Language (SAML) is an XML standard that helps to secure web domains to exchange user authentication and authorization data. Using SAML, an online service provider, can contact a separate online identity provider to authenticate users who are trying to access secure content.

(https://twitter.com/FutureProcessin
(https://plus.google.com/1109395185
(https://www.youtube.com/user/Futu
(https://www.linkedin.com/company processing-sp--
(http://www.goldenline.pl/firma/futu processin
(http://www.slideshare.net/FuturePro

Figure 4. Sample SAML Token.

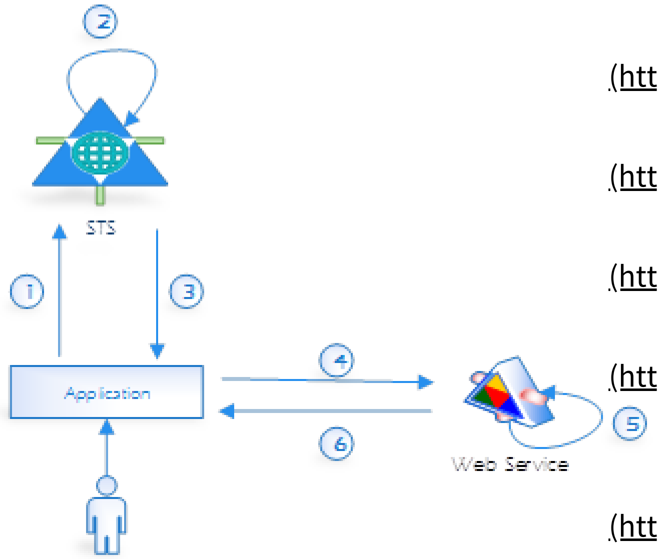


```
<saml:Assertion MajorVersion="1" MinorVersion="1" AssertionID="_f0a14e3d-5fc5-4360-ae6c-c829ddf94f23"
 Issuer="ActAsSts" IssueInstant="2015-02-22T16:03:09.960Z" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
  <saml:Conditions NotBefore="2015-02-22T16:03:09.955Z" NotOnOrAfter="2015-02-22T16:33:09.955Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>https://relyingparty/*</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier>1</saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="name" AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
      <saml:AttributeValue>Sample User</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="country" AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
      <saml:AttributeValue>PL</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="emailaddress" AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
      <saml:AttributeValue>sampleuser@test.pl</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="role" AttributeNamespace="http://schemas.microsoft.com/ws/2008/06/identity/claims">
      <saml:AttributeValue>Admin</saml:AttributeValue>
      <saml:AttributeValue>User</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
  <Signature xmlns="http://www.w3.o">...</Signature>
</saml:Assertion>
```

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-4-Sample-SAML-Token.png)

The claims-based identity mechanism can be used to build authentication and authorization process in application. Based on a token issued by STS, an application can verify whether user is authenticated as well as define user rights. Received token can be also used to authenticate in other application and services (SSO support). In the figure below a communication flow between STS and Web Service via client application has been presented.

Figure 5. Web Service with STS.



(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-5-Web-Service-with-STS.png)

1. The client initializes and sends (via application) authentication request to the STS. The request contains client's credentials required to authenticate the client.

2. The STS validates the client's credentials.

3. The STS issues a security token to the client. If the client's credentials are successfully validated, the STS issues a security token (such as a SAML token). The token contains claims which represent user identity.

4. The client (via application) sends a request message to the service. The request message contains the received token.

5. The service validates the security token and processes the request. To validate token connection between service and STS is not necessary – issuer validation is based on PKI (this mechanism will be described in detail in the next article)

6. (Optional) The service initializes and sends a response message to the client.

ACTIVE VS PASSIVE STS

The claims-based authentication and authorization mechanism is generally used in web application. Communication between web application and STS can be made in two modes : passive and active. Which kind of communication will be used depends on STS type. STS can be categorized into two basic groups: Passive and Active.

Active STS is a special type of web service, which is based on WS-Trust protocol. Passive STS is generally a special kind of web site, where user is redirected during authentication process. User authentication credentials are given on STS login form. After authentication process user is redirected to source application.

In the next series of articles a practical aspect of both (passive and active) approaches will be presented.

(https://twitter.com/FutureProcessin

(https://plus.google.com/1109395185

(https://www.youtube.com/user/Futu

(https://www.linkedin.com/company processing-sp--

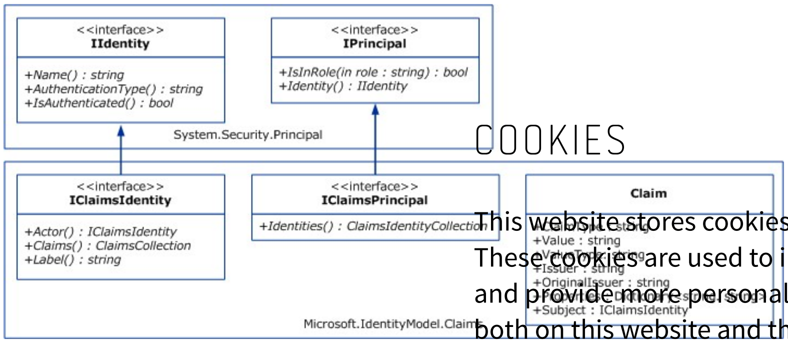
(http://www.goldenline.pl/firma/futu processin

(http://www.slideshare.net/FuturePro

WIF

The claims-based identity has been evolving within the Microsoft .NET Framework during the last few years. WIF (Windows Identity Foundation) was designed to unify and simplify the claims-based identity approach. WIF (current version 4.5) is a set of .NET Framework classes for implementing claims-based identity. In ASP.NET and Windows Communication Foundation (WCF) technologies IIdentity and IPrincipal interfaces are used to work with the user's identity information. WIF extends these interfaces, as illustrated in the following diagram:

Figure 6. WIF basic classes.

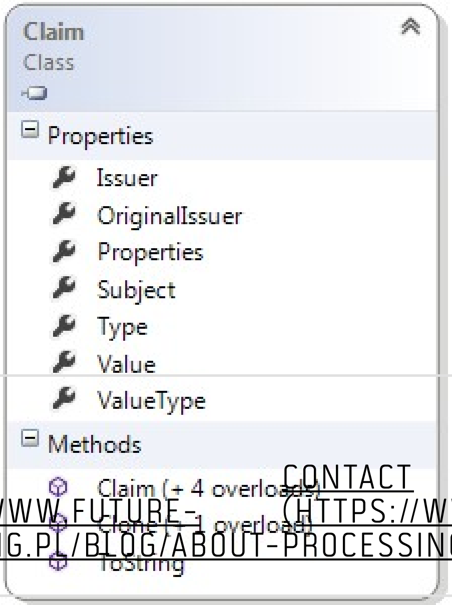

<<interface>>
IIdentity
+Name() : string
+AuthenticationType() : string
+IsAuthenticated() : bool
<<interface>>
IPrincipal
+IsInRole(in role : string) : bool
+Identity() : IIdentity
System.Security.Principal
<<interface>>
IClaimsIdentity
+Actor() : IClaimsIdentity
+Claims() : ClaimsCollection
+Label() : string
<<interface>>
IClaimsPrincipal
+Identities() : ClaimsIdentityCollection
Claim
+ClaimType : string
+Value : string
+ValueType : string
+Issuer : string
+OriginalIssuer : string
+Properties : Dictionary<string, string>
+Subject : IClaimsIdentity
Microsoft.IdentityModel.Claims

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-6.-WIF-basic-classes..jpg)

Identity interface allows to verify whether authentication process has finished successfully (IsAuthenticated () method). ICliamsIdentity extends this interface to access to claims collection. Windows Identity Foundation represents a claim within the Claim class.

(https://twitter.com/FutureProcessin
(https://plus.google.com/110939518
(https://www.youtube.com/user/Futu
(https://www.linkedin.com/company processing-sp--
(http://www.goldenline.pl/firma/futu processin
(http://www.slideshare.net/FuturePr

Figure 7. Claim Class.



Show menu
TECHNOLOGIES ▾
PROCESSES, STANDARDS AND QUALITY ▾
ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/)
CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)
Type

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/figure-7-Claim-Class.jpg)

Claim.ClaimType is a string that tells you what the claim means.
E.g. (typically a URI):
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth.
Many predefined claim types have been defined in WIF – class ClaimTypes contains all predefined claims types.
This claim type list can be also extend – it is possible to create our claim type definition and use it in our solution.
The type of the claim allows to read its value from Claim.Value.
Authorization mechanism can be built using IsInRole method on the IPrincipal interface. Checking access based on roles is called Role-Based Access Control (RBAC).
Special type of claim called Role (http://schemas.microsoft.com/ws/2008/06/identity/claims/role) allows to define user roles. The IsInRole method verifies,

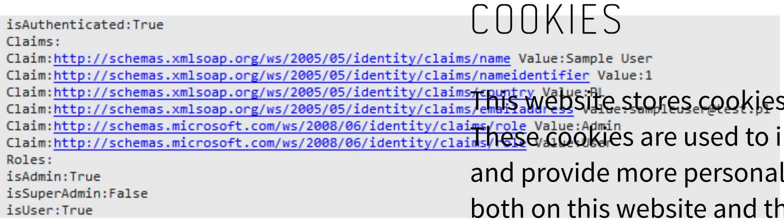
based on this claim, whether user belongs to the specified role.

Sample code:

```
//create Claim Identity
            var userIdentity = new ClaimsIdentity("Ident
            userIdentity.Label = "Identity";
            userIdentity.AddClaim(new Claim(ClaimTypes.Nar
            userIdentity.AddClaim(new Claim(ClaimTypes.Na
            userIdentity.AddClaim(new Claim(ClaimTypes.Cou
            userIdentity.AddClaim(new Claim(ClaimTypes.Ema
            userIdentity.AddClaim(new Claim(ClaimTypes.Rol
            userIdentity.AddClaim(new Claim(ClaimTypes.Ro
            var identity = new ClaimsPrincipal(userIdenti

            //isAuthenticated ??
            bool isAuthenticated = identity.Identity.IsAut
            Console.WriteLine("isAuthenticated:{0}", isAut

            //List of claims values
            Console.WriteLine("Claims:");
            foreach (var claim in identity.Claims)
            {
                Console.WriteLine("Claim:{0} Value:{1}", c
            }

            Console.WriteLine("Roles:");
            var isAdmin = identity.IsInRole("Admin");
            Console.WriteLine("isAdmin:{0}", isAdmin);
            var isSuperAdmin = identity.IsInRole("SuperAdr
            Console.WriteLine("isSuperAdmin:{0}", isSuperA
            var isUser = identity.IsInRole("User");
            Console.WriteLine("isUser:{0}", isUser);
```

Execution result:

Figure 8. Code execution result.



```
isAuthenticated:True
Claims:
Claim:http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name Value:Sample User
Claim:http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier Value:1
Claim:http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country Value:PL
Claim:http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress Value:sampleuser@test.pl
Claim:http://schemas.microsoft.com/ws/2008/06/identity/claims/role Value:Admin
Claim:http://schemas.microsoft.com/ws/2008/06/identity/claims/role Value:User
Roles:
isAdmin:True
isSuperAdmin:False
isUser:True
```

(https://www.future-processing.com/blog/wp-content/uploads/2015/03/Figure-8.-Code-execution-result..png)

In web application AuthorizeAttribute.Roles can be used to verify user role.

```
[Authorize(Roles = "SuperAdmin,Admin ")]
public class HomeController : Controller
{
//...
}
```

Authorization mechanism , as well as claims values can be used to verify user rights – some additional claim called e.g. userPermission can be added and this claim can contain information about user permission.

In a real environment user identity (ClaimsIdenity) is created using a security token received from STS . Below, sample code with which allows creating ClaimsIdenity object from SecurityToken.

```
private ClaimsPrincipal CreateUserIdentityFromSecurityToke
        {
            var genericToken = token as GenericXmlSecurity
            var handlers = FederatedAuthentication.Federat
            var cToken = handlers.ReadToken(new XmlTextRe
            var identity = handlers.ValidateToken(cToken)
            var userIdenity = new ClaimsPrincipal(identity
            return userIdenity;
```

(https://twitter.com/FutureProcessin
(https://plus.google.com/110939518
(https://www.youtube.com/user/Futu
(https://www.linkedin.com/company processing-sp--
(http://www.goldenline.pl/firma/futu processin
(http://www.slideshare.net/FuturePr

Show menu

TECHNOLOGIES ▾ PROCESSES, STANDARDS AND QUALITY ▾ ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/) CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)

Type

EXISTING SOLUTION BASED ON CLAIMS

Nowadays popularity of Claims-based identity solutions is still increasing. There are many commercial and open source solutions e.g. :

- ADFS (Microsoft)
- Oracle Security Token Service
- WSO2 Identity Server
- Thinktecture IdentityServer

Using WIF, we can also build our own solutions based on claims.

SUMMARY

In this article the approach to authentication and authorisation process based on claims has been presented. What is more, there has been possibility to also present a WIF technology (basic part of WIF), which allows to build claims-based identity solution on .NET platform.

The next article will describe technical approach to STS – demonstration of how to build our STS based on user/password authentication.

← PREVIOUS POST
(https://www.future-processing.pl/blog/certified-tester-advanced-level-test-manager/)

NEXT POST →
(https://www.future-processing.pl/blog/setting-coding-standards-in-a-project/)

RELATED POSTS

(https://www.future-processing.pl/blog/how-to-build-a-fault-tolerant-system/)

Wojciech Ryczko

(https://www.future-processing.pl/blog/keep-it-clean-hexagonal-architecture-in-practice-video-in-polish/)

Daria Sobaszek-Dobska

(https://www.future-processing.pl/blog/privilege-escalation/)

Kamil Górnicz

[How to build a fault tolerant system? (https://www.future-processing.pl/blog/how-to-build-a-fault-tolerant-system/)](https://www.future-processing.pl/blog/how-to-build-a-fault-tolerant-system/)

05.12.2018

In many applications, where computers are used, outages or malfunctions can be expensive, or even disastrous. In that…

READ MORE ▶

Daria Sobaszek-Dębska

[Keep it clean: hexagonal architecture in practice [VIDEO in Polish] (https://www.future-processing.pl/blog/keep-it-clean-hexagonal-architecture-in-practice-video-in-polish/)](https://www.future-processing.pl/blog/keep-it-clean-hexagonal-architecture-in-practice-video-in-polish/)

19.09.2018

Jeszcze przed wakacjami, w siedzibie Future Processing gościliśmy uczestników 28. spotkania społeczności Silesia Java User Group. Jak zwykle…

READ MORE ▶

[Privilege escalation (https://www.future-processing.pl/blog/privilege-escalation/)](https://www.future-processing.pl/blog/privilege-escalation/)

03.07.2018

How privileges work in operating systems? By definition privileges are defined as the delegation of authority over a…

READ MORE ▶

COMMENTS

1 Comment Future Processing 1 Login

♡ Recommend Tweet Share Sort by Newest

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Sameers • 10 months ago

Where can i get complete code or sample project for SAML based authentication?

^ | ∨ • Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

(https://twitter.com/FutureProcessin

(https://plus.google.com/1109395185

(https://www.youtube.com/user/Futu

(https://www.linkedin.com/company processing-sp--

(http://www.goldenline.pl/firma/futu processin

(http://www.slideshare.net/FuturePr

Show menu

Type

TECHNOLOGIES▾ PROCESSES, STANDARDS AND QUALITY▾ ABOUT US (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/ABOUT-US/) CONTACT (HTTPS://WWW.FUTURE-PROCESSING.PL/BLOG/CONTACT/)

MOST POPULAR TAGS

.NET (https://www.future-processing.pl/blog/tag/net/)

Agile (https://www.future-processing.pl/blog/tag/agile-2/)

Android (https://www.future-processing.pl/blog/tag/android/)

Architecture (https://www.future-processing.pl/blog/tag/architecture/)

Automation (https://www.future-processing.pl/blog/tag/automation/)

Best Practices (https://www.future-processing.pl/blog/tag/best-practices/)

C# (https://www.future-processing.pl/blog/tag/c/)

cloud (https://www.future-processing.pl/blog/tag/cloud-2/)

FUTURE PROCESSING

— About us (https://www.future-processing.pl/blog/about-us/)

— Contact (https://www.future-processing.pl/blog/contact/)

— Cookies (https://www.future-processing.pl/blog/cookies/)

ARCHIVES

December 2018 (https://www.futu processing.pl/blog/2018/12/)

November 2018 (https://www.futu .pl/blog

October 2018 (https://www.futu processing.pl/blog/2018/10/)

September 2018 (https://www.futu .pl/blog

August 2018 (https://www.futu processing.pl/blog/2018/08/)

July 2018 (https://www.futu .pl/blog

June 2018 (https://www.futu processing.pl/blog/2018/06/)

May 2018 (https://www.futu .pl/blog

January 2018 (https://www.futu processing.pl/blog/2018/01/)

August 2017 (https://www.futu .pl/blog

June 2017 (https://www.futu processing.pl/blog/2017/06/)

March 2017 (https://www.futu .pl/blog

...more

COOKIES

This website stores cookies on your computer. These cookies are used to improve our website and provide more personalized services to you, both on this website and through other media. To find out more about the cookies we use, see our Cookies policy (/cookies-policy/).

ACCEPT

Technical Blog (https://www.future-processing.pl/blog) by Future Processing (http://www.future-processing.com)

(https://www.facebook.com/FutureP

cloud computing (https://www.future-processing.pl/blog/tag/cloud-computing/)

databases (https://www.future-processing.pl/blog/tag/databases-2/)

Docker (https://www.future-processing.pl/blog/tag/docker/)

Java (https://www.future-processing.pl/blog/tag/java/)

JavaScript (https://www.future-processing.pl/blog/tag/javascript/)

nosql (https://www.future-processing.pl/blog/tag/nosql/)

performance (https://www.future-processing.pl/blog/tag/performance-2/)

PHP (https://www.future-processing.pl/blog/tag/php/)

Process Management (https://www.future-processing.pl/blog/tag/process-management/)

Project Management (https://www.future-processing.pl/blog/tag/project-management/)

Scrum (https://www.future-processing.pl/blog/tag/scrum/)

Security (https://www.future-processing.pl/blog/tag/security-2/)

testing (https://www.future-processing.pl/blog/tag/testing/)

WIF (https://www.future-processing.pl/blog/tag/wif/)

(https://twitter.com/FutureProcessin

(https://plus.google.com/1109395185

(https://www.youtube.com/user/Futu

(https://www.linkedin.com/company
processing-sp--

(http://www.goldenline.pl/firma/futu
processin

(http://www.slideshare.net/FuturePr