Key Concepts

The following sections include definitions of identity-related terms and concepts you may come across as you use the WSO2 Identity Server.

Access control

XACML - eXtensible Access Control Markup Language

XACML (eXtensible Access Control Markup Language) is a tool for controlling access to applications. XACML is an XML-based language for access control that has been standardized by the Technical Committee of the OASIS consortium. XACML is very popular as a fine grained authorization method amongst the identity community. See Access Control Concepts for more information.

RBAC - Role Based Access Control

Role-based access control (RBAC) is a type of access control. It is an approach used to restrict access to authorized users based on their role. It is used by the majority of enterprises with more than 500 users. See Access Control Concepts for more information.

ABAC - Attribute Based Access Control

Attribute-based access control (ABAC) is a type of access control. ABAC defines a new access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. See Access Control Concepts for more information.

API security

OAuth

OAuth is an open standard to authorization. OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections and access tokens. See OAuth Concepts for more information.
## Tokens

A token is a simple string that is passed as an HTTP header of a request. Access tokens authenticate API users and applications and ensure better security, e.g., preventing DoS attacks. If a token that is passed with a request is invalid, the request is discarded in the first stage of processing. WSO2 Identity Server supports following token types:

- **Application access token**: This token identifies and authenticates an entire application. You can invoke all the APIs associated with an application using a single application access token.
- **User access token**: This token identifies the end user of an application, e.g., the end user of a mobile application deployed on a different device.
- **Refresh token**: When an access token expires, you have to obtain a new access token to access the API. A refresh token enables to obtain a new access token without a user intervention and use it to regain access to the API.
- **ID token**: This is a JSON Web Token (JWT) containing the user's authentication status and profile information in the form of claims.

## Grant Types

There are many supported grant types in the OAuth2 specification. A grant type is a credential representing the resource owner's authorization (to access its protected resources) used by the client to obtain an access token. See [OAuth Concepts](#) for more information.

## Architecture authentication framework

### Inbound Authenticators

The responsibility of inbound authenticators is to identify and parse all the incoming authentication requests and then build the corresponding response. A given inbound authenticator has two parts.

1. Request Processor
2. Response Builder

For each protocol supported by WSO2 IS, there should be an inbound authenticator. This architecture component includes inbound authenticators for SAML 2.0, OpenID, OpenID Connect, OAuth 2.0, and WS-Federation (passive). In other words, the responsibility of the SAML 2.0 request processor is to accept a SAML request from a service provider, validate the SAML request and then build a common object model understood by the authentication framework and handover the request to it. The responsibility of the SAML response builder is to accept a common object model from the authentication framework and build a SAML response out of it. Both the request processors and the response builders are protocol aware, while the authentication framework is not coupled to any protocol.

See [Identity Server Architecture](#) for more information.
Local Authenticators

The responsibility of the local authenticators is to authenticate the user with locally available credentials. This can be either username/password or even IWA (Integrated Windows Authentication). Local authenticators are decoupled from the Inbound Authenticators. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of local authenticators registered with the service provider corresponding to the current authentication request.

Once the local authentication is successfully completed, the local authenticator will notify the framework. The framework will now decide no more authentication is needed and hand over the control to the corresponding response builder of the inbound authenticator.

You can develop your own local authenticators and plug them into the Identity Server.

See Identity Server Architecture for more information.

Outbound/Federated Authenticators

The responsibility of the federated authenticators is to authenticate the user with an external system. This can be with Facebook, Google, Yahoo, LinkedIn, Twitter, Salesforce or any other identity provider. Federated authenticators are decoupled from the Inbound Authenticators. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of federated authenticators registered with the service provider corresponding to the current authentication request.

A federated authenticator has no value unless it is associated with an identity provider. The Identity Server out-of-the-box supports SAML 2.0, OpenID, OpenID Connect, OAuth 2.0 and WS-Federation (passive). The SAML 2.0 federated authenticator itself has no value. It has to be associated with an Identity Provider. Google Apps can be an identity provider - with the SAML 2.0 federated authenticator. This federated authenticator knows how to generate a SAML request to the Google Apps and process a SAML response from it.

There are two parts in a federated authenticator.

1. Request Builder
2. Response Processor

Once the federation authentication is successfully completed, the federated authenticator will notify the authentication framework. The framework will now decide no more authentication is needed and hand over the control to the corresponding response builder of the inbound authenticator.

Both the request builder and the response processor are protocol aware while the authentication framework is not coupled to any protocol.

You can develop your own federated authenticators and plug them into the Identity Server.

See Identity Server Architecture for more information.
Request-path Authenticators

This is a special type of authenticator. Request-path authenticator is always a local authenticator. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of request-path authenticators registered with the service provider corresponding to the current authentication request. Then the framework will check whether there is any request-path authenticator applicable for the initial authentication request. In other words, a request path authenticator will get executed only if the initial authentication request brings the applicable set of credentials with it.

The request-path authenticators always require the user credentials to be present in the initial authentication request itself. This does not need any end-user interactions with the Identity Server.

Once the request-path authentication is successfully completed, the request-path authenticator will notify the authentication framework. The framework will now decide no more authentication is needed and hand over the control to the corresponding response builder of the inbound authenticator.

See Identity Server Architecture for more information.

Multi-option authentication

The service provider can define how to authenticate users at the Identity Server, for authentication requests initiated by it. While doing that, each service provider can pick more than one authenticators - so, the end user will get multiple login options. This can be a combination of local authenticators and federated authenticators.

See Identity Server Architecture for more information.

Multi-factor authentication

The service provider can define how to authenticate users at the Identity Server, for authentication requests initiated by it. While doing that, each service provider can define multiple steps and for each step it can pick more than one authenticator. The authentication framework will track all the authenticators in each step and will proceed to the next step only if the user authenticates successfully in the current step. It's an AND between steps while its an OR between the authenticators in a given step.

See Identity Server Architecture for more information.

Architecture provisioning framework
Inbound provisioning

Inbound provisioning focuses on how to provision users to the Identity Server. Out-of-the-box, the Identity Server supports inbound provisioning via a SOAP-based API as well as the SCIM 1.1 API. Both the APIs support HTTP Basic Authentication. If you invoke the provisioning API with Basic Authentication credentials, then where to provision the user (to which user store) will be decided based on the inbound provisioning configuration of the resident service provider.

The SCIM API also supports OAuth 2.0. If the user authenticates to the SCIM API with OAuth credentials, then the system will load the configuration corresponding to the service provider who owns the OAuth client id. If you plan to invoke the SCIM API via a web application or a mobile application, we would highly recommend you to use OAuth instead of Basic Authentication. You simply need to register your application as a service provider in Identity Server and then generate OAuth keys.

See Identity Server Architecture for more information.

Just-in-time (JIT) provisioning

Just-in-time provisioning talks about how to provision users to the Identity Server at the time of federated authentication. A service provider initiates the authentication request, the user gets redirected to the Identity Server and then Identity Server redirects the user to an external identity provider for authentication. Just-in-time provisioning gets triggered in such a scenario when the Identity Server receives a positive authentication response from the external identity provider. The Identity Server will provision the user to its internal user store with the user claims from the authentication response.

You configure JIT provisioning against an identity provider - not against service providers. Whenever you associate an identity provider with a service provider for outbound authentication, if the JIT provisioning is enabled for that particular identity provider, then the users from the external identity provider will be provisioned into the Identity Server's internal user store. In the JIT provisioning configuration you can also pick the provisioning user store.

JIT provisioning happens while in the middle of an authentication flow. The provisioning can happen in a blocking mode or in a non-blocking mode. In the blocking mode, the authentication flow will be blocked till the provisioning finishes - while in the non-blocking mode, provisioning happens in a different thread.

See Identity Server Architecture for more information.
Outbound provisioning

Outbound provisioning talks about provisioning users to external systems. This can be initiated by any of the following.

- Inbound provisioning request (initiated by a service provider or the resident service provider)
- JIT provisioning (initiated by a service provider)
- Adding a user via the management console (initiated by the resident service provider)
- Assigning a user to a provisioning role (initiated by the resident service provider)

WSO2 Identity Server supports outbound provisioning with the following connectors. You need to configure one or more outbound provisioning connectors with a given identity provider, and associate the identity provider with a service provider. All the provisioning requests must be initiated by a service provider - and will be provisioned to all the identity providers configured in the outbound provisioning configuration of the corresponding service provider.

- SCIM
- SPML
- SOAP
- Google Apps provisioning API
- Salesforce provisioning API

See Identity Server Architecture for more information.

Conditional provisioning with roles

If you want to provision a user to an external identity provider, for example to Salesforce or Google Apps, based on the user's role, then you need to define one or more provisioning roles in the outbound provisioning configuration of the corresponding identity provider.

See Configuring Roles for an Identity Provider for more information.

Identity provisioning

Identity Provisioning

Identity provisioning plays a key role in propagating user identities across different SaaS providers. Provisioning is the process of coordinating the creation of user accounts, e-mail authorizations in the form of rules and roles, and other tasks such as provisioning of resources associated with enabling new users. See The Evolution of Provisioning Standards for more information.

SPML - Service Provisioning Markup Language

Service Provisioning Markup Language (SPML) is an XML-based framework developed by OASIS for exchanging user, resource and service provisioning information between cooperating organizations. See The Evolution of Provisioning Standards for more information.
SCIM - System for Cross-domain Identity Management

The System for Cross-domain Identity Management (SCIM) specification is designed to make managing user identities in the WSO2 Identity Server easier. SCIM is an emerging open standard which defines a comprehensive REST API along with a platform neutral schema and a SAML binding to facilitate the user management operations across SaaS applications; placing specific emphasis on simplicity and interoperability as well. See The Evolution of Provisioning Standards for more information.

SOAP security

SOAP - Simple Object Access Protocol

SOAP, originally defined as Simple Object Access protocol, is a protocol specification for exchanging structured information in the implementation of Web services. It relies on XML Information Set for its message format, and usually relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

STS - Security Token Service

The “Security Token Service” component of WSO2 Carbon enables you to configure the generic STS to issue claim-based security tokens. A claim-based security token is a common way for applications to acquire and authenticate the identity information they need about users inside their organization, in other organizations, and on the Internet. This Security Token Service is capable of issuing SAML 1.1 and SAML 2.0 tokens as recommended in WS-Trust and SAML Web Service Token Profile specifications.

WS-Federation - Web Services Federation

WS-Federation (Web Services Federation) describes the management and brokering of trust relationships and security token exchange across Web services and organizational boundaries. WS-Federation is a part of the larger WS-Security framework. For example, WS-Federation builds on the Security Token Service (STS) by providing mechanisms that facilitate interactions. In the WS-Federation Model, an Identity Provider is a Security Token Service (STS).

WS-Security - Web Services Security

Web Services Security (WS-Security, WSS) is an extension to SOAP to apply security to Web services. It is a member of the Web service specifications and was published by OASIS. The protocol specifies how integrity and confidentiality can be enforced on messages and allows the communication of various security token formats.

WS-Trust - Web Services Trust

WS-Trust is a WS-* specification and OASIS standard that provides extensions to WS-Security, specifically dealing with the issuing, renewing, and validating of security tokens, as well as with ways to establish, assess the presence of, and broker trust relationships between participants in a secure message exchange.
SSO and identity federation

Identity Federation

Identity federation enables users to access multiple applications using the same access credentials. This makes access easy, as users do not have to remember a different set of credentials for every application they use. However, the users have to provide their credentials to each one of the applications separately although the credentials used are the same. See Evolution of Identity Federation Standards for more information.

SSO - Single Sign On

On the other hand, SSO enables users to provide their credentials once and obtain access to multiple applications. In SSO, the users are not prompted for their credentials when accessing each application until their session is terminated. See Evolution of Identity Federation Standards for more information.

SAML Metadata

SAML profile requires agreements between system entities regarding identifiers, binding support, end points, certificates and so forth. A metadata specification is used to describe this information in a standard way.