Architecture

WSO2 Identity Server (WSO2 IS) is a product built on top of WSO2 Carbon. Based on the OSGi specification, it enables easy customization and extension through its componentized architecture. This topic describes the architecture of the Identity Server. The users are given the choice of deployment to on-premise servers, private cloud or public cloud without configuration changes.

WSO2 Identity Server is used directly by multiple users, through its user-friendly Management Console. Apart from the default admin user (with the user name ‘admin’), other users can be created later by the admin users that have the privileges to create a new user, or by signing up. Each user can have roles, where each role can have privileges assigned to them. A user’s roles can be changed at any time by the admin user.

Apart from such registered users, Identity Server is also used as an identity provider for third party applications, which also have their own sets of users.

The following are the topics addressed in this page.

Architecture and process flow | Authentication framework | Provisioning framework | Components of the architecture

**Architecture and process flow**

The following diagram depicts the architecture of the Identity Server and the various processes that take place within it.
Watch the following video for a quick overview of the process flow of the Identity Server architecture and how the various components interact with each other.

Authentication framework

The following are the authenticator types in the authentication framework.

- Inbound authenticators
- Local authenticators
- Outbound/federated authenticators
- Multi-option authenticators
- Multi-factor authenticators

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 Identity Server, there should be an inbound authenticator. This architecture component includes inbound authenticators for Security Assertion Markup Language (SAML) 2.0, 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.

Local authenticators

The responsibility of the local authenticators is to authenticate the user with locally available credentials. This can be either user name/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.

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 Security Assertion Markup Language (SAML) 2.0, 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.

Multi-option authenticators

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 authenticator to allow end users to get multiple
login options. This can be a combination of local authenticators and federated authenticators.

**Multi-factor authenticators**

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 tracks all the authenticators in each step and proceeds to the next step only if the user authenticates successfully in the current step. It is an AND between steps, while it is an OR between the authenticators in a given step.

**Provisioning framework**

The following are the provisioning components available in the provisioning framework.

- Inbound provisioning
- Just-in-time provisioning
- Outbound provisioning

**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 Simple Object Access Protocol (SOAP) based API as well as the System for Cross-domain Identity Management (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.

**Just-in-time provisioning**

Just-in-time (JIT) 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.

**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)
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

## Components of the architecture

The following table lists out the components pertaining to the architecture of the WSO2 Identity Server, which are depicted in the above figure and video.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Process flow</th>
</tr>
</thead>
</table>

- 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)
A Service Provider (SP) is an entity that provides Web services. A service provider relies on a trusted Identity Provider (IdP) for authentication and authorization. In this case, the Identity Server acts as the IdP and does the task of authenticating and authorizing the user of the service provider.

Salesforce and Google Apps are examples of service providers and are used as such in this case.

<table>
<thead>
<tr>
<th>Related links</th>
</tr>
</thead>
<tbody>
<tr>
<td>For information on how to add a service provider to the Identity Server, and do the necessary configurations to integrate the SP with the Identity Server, see Adding and Configuring a Service Provider.</td>
</tr>
</tbody>
</table>
The Inbound Authentication component of the Identity Server can handle any of the following requests.

- **SAML SSO:** Security Assertion Markup Language (SAML) is an OASIS open standard for representing and exchanging user identity and authentication data between parties. SAML provides the web-based Single-Sign-On capability. WSO2 IS supports SAML 2.0.

- **OAuth/OpenID Connect:** OAuth 2.0 has three main phases. They are: requesting an Authorization Grant, exchanging the Authorization Grant for an Access Token and accessing the resources using this Access Token. OpenID Connect is another identity layer on top of OAuth 2.0. OAuth applications can get authentication event information over the ID token and can get the extra claims of the authenticated user from the OpenID Connect UserInfo endpoint. WSO2 IS supports Oauth 1.0 and 2.0.

- **Passive STS:** A Security Token Service (STS) is a software based identity provider responsible for issuing security tokens, especially software tokens, as part of a claims-based identity system.

### Related links

For information on how to configure inbound authentication, see Configuring Inbound Authentication for a Service Provider.
Claim management is a key aspect of the Identity Server, which helps to map local claims to service provider claims and vice versa. It also enables you to map local claims to identity provider claims and vice versa.

Just-in-Time (JIT) provisioning allows you to create users on the fly without having to create user accounts in advance. For example, if you recently added a user to your application, you don’t need to manually create the user in the Identity Server. When they log in with single sign-on, their account is automatically created for them, eliminating the time and effort related to creating the account. Just-in-Time provisioning works with your identity provider to pass the correct user information to the Identity Server.

**Related links**

- For information on mapping local claims and service provider claims, see Configuring Claims for a Service Provider.
- For mapping local claims and identity provider claims, see Configuring Claims for an Identity Provider.
- For information on configuring Just-in-Time provisioning, see Configuring Just-In-Time Provisioning for an Identity Provider.
Local authenticators are authentication processes available within the Identity Server itself. User name/password authentication happens by authenticating the credentials entered against the values in the user store connected to the Identity Server.

**Related links**

- For more information on how to configure local authenticators in the Identity Server, see Configuring Local and Outbound Authentication for a Service Provider.
- For more information about Windows-based authentication, see Integrated Windows Authentication.

The IN channel of the authentication framework sends the authentication request to the local authenticators component.

The local authenticator does the authentication by checking the user name and password or by using Integrated Windows Authentication (IWA). Once this is authentication is done, it provides the authentication response to the OUT channel of the authentication framework.
Federated authenticators are authentication processes that are not available within the Identity Server. These need to be configured to reach out to external applications to do the authentication process and send the response back to the Identity Server.

Federated authenticators

- Open ID Connect
- OAuth
- SAML SSO
- Passive STS
- Facebook
- Yahoo
- Google
- Microsoft
- SMS OTP
- Email OTP
- Twitter
- IWA Kerberos
- Office365

The IN channel of the authentication framework sends the authentication request to the federated authenticators component.

The federated authenticators do the authentication by checking the authentication request in the specified authenticator. For example, if Facebook is configured, the authentication process will reflect that. Once this authentication is done, it provides the authentication response to the OUT channel of the authentication framework.

Related links

For more information on how to configure federated authenticators with the various identity providers, see Configuring Federated Authentication.
Identity providers perform authentication. To receive authentication requests from the Identity Server, configurations need to be done at the identity provider side as well. Identity providers are also known as external applications. The protocol specific authenticators (SAML2, OpenID Connect, WS-Federation (Passive)) represent applications that use these protocols for authentication requests.

The authentication request comes in from the federated authenticators component and is sent to the relevant identity provider (External application). The user is authenticated and logged in to the relevant external application.

A single authentication request can require authentication from multiple external applications.
The provisioning framework is responsible for all provisioning work done by the Identity Server. This framework integrates with the User Store Manager component and also receives provisioning requests from the authentication framework.

The JIT provisioning component of the OUT channel in the authentication framework sends the request to the provisioning framework. This occurs if the user is not added into the user store and needs to be added on-the-fly.

The provisioning framework sends the user details to the user store manager and the user is added. Once the user is added, the user store manager contacts the provisioning framework with the response.

The SCIM and SOAP requests that arrive from the service provider are added to the user store manager. This information is also sent to the provisioning framework. The provisioning framework sends this along to the outbound provisioning component.
### Authorization manager

WSO2 Identity Server contains an advanced entitlement auditing and management. It provides entitlement management for any REST or SOAP calls. WSO2 Identity Server provides attribute and claim-based access control via XACML, WS-Trust, OpenID Connect and claim management. WSO2 Identity Server also provides role-based access control (RBAC) and fine-grained policy-based access control via XACML.

WSO2 Identity Server provides a friendly user interface for policy editing. It also supports multiple Policy Information Point (PIP) and policy distribution to various Policy Decision Points (PDPs). It provides a high-performance network protocol (over Thrift) for PEP/PDP interaction, and policy decision and attribute caching. Notifications are provided for policy updates. Moreover, the WSO2 Carbon TryIt tool that comes bundled with the Identity Server lets the user explore the policy impact.

**Related links**

For more information on how to use and manage entitlement within the Identity Server, see [Working with Entitlement](#).

### IdP and SP configurations

The identity provider and service provider configurations provide the basis for all actions that happen within the authentication framework and provisioning framework.

**Related links**

- For more information on how to configure the service provider, see [Adding and Configuring a Service Provider](#).
- For more information on how to configure the identity provider, see [Adding and Configuring an Identity Provider](#).

Authorization does not play a direct role in the process flow but as a component, it integrates with various other components in the Identity Server. This is primarily a functionality that can be managed using APIs that are written to perform authorization tasks.

The identity provider and service provider configurations go to both the authentication framework and provisioning framework.

- [IdP and SP Configurations](#) → [Authentication Framework](#)
- [IdP and SP Configurations](#) → [Provisioning Framework](#)
Inbound provisioning requests can come in the form of SCIM or SOAP.

The System for Cross-domain Identity Management (SCIM) specification is designed to make managing user identities in the WSO2 Identity Server easier. Identity provisioning is a key aspect of any Identity Management Solution. In simple terms, it is to create, maintain and delete user accounts and related identities in one or more systems or applications in response to business processes which are initiated either by humans directly or by automated tasks.

Simple Object Access Protocol (SOAP) is a protocol for exchanging XML-based messages over a network, normally using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build on. SOAP services are defined using Web Services Definition Language (WSDL) and are accessible using a URL that is known as a SOAP endpoint. Here, a SOAP API is used to provision users to the Identity Server.

Related links

For more information on configuring inbound provisioning, see Configuring Inbound Provisioning for a Service Provider.
WSO2 Identity Server implements flexible user store via built-in LDAP (powered by ApacheDS), external LDAP, Microsoft Active Directory or any JDBC database. It provides an API for integrating identity management to any application. WSO2 Identity Server allows tenants/organizations to configure their user stores through the admin console. WSO2 Identity Server supports multiple profiles per user using its flexible profile management feature.

**Related links**

- For more information, see User Management Architecture
- For more information on how to configure user stores, see Configuring the Realm.
- For more information on how to work with users and roles, see Configuring Users, Roles and Permissions.

The user store manager receives provisioning requests from the provisioning framework. These provisioning requests are handled and the relevant user store is updated. The request can affect multiple user stores if the configuration is such. Once this request has been handled, an update is sent back to the provisioning framework.

The inbound provisioning component sends SCIM and SOAP provisioning requests on to the user store manager.

The user store manager receives the provisioning request, acts on it and sends it on to the provisioning framework where it has to be sent on for outbound provisioning.
**Claim manager**

A claim is a piece of information about a particular subject. It can be anything that the subject is owned by or associated with, such as name, group, preferences, etc. A claim provides a single and general notion to define the identity information related to the subject. Claim-based identity is a common way for any application to acquire the identity information. It provides a consistent approach for all applications by hiding the lower level implementation. Claims are also used in identity propagation, by packaging the claims into one or more tokens (such as SAML). These tokens are then issued by an issuer; commonly known as a security token service (STS).

### Related links

- For more information on managing claims, see [Claim Management](#).
- For information on how to configure claims on the service provider side, see [Configuring Claims for a Service Provider](#).
- For information on how to configure claims on the identity provider side, see [Configuring Claims for an Identity Provider](#).

**XACML**

XACML does not play a direct role in the process flow but as a component, it integrates with various other components in the Identity Server. Primarily, it integrates with the following four components:

- Authentication framework
- Authorization manager
- Provisioning framework
- User store manager

XACML (eXtensible Access Control Markup Language) is ideally a part of the authorization manager component but it is depicted separately due to its unique role in the Identity Server architecture. XACML is an XML-based language for access control that has been standardized by the Technical Committee of the OASIS consortium. XACML is popular as a fine grain authorization method among the community. However, there are aspects of XACML that enable it to surpass being just a fine grain authorization mechanism. XACML describes access control policy language, request/response language and reference architecture. The policy language is used to express access control policies (who can do what when). The request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses).
<table>
<thead>
<tr>
<th><strong>Auditing</strong></th>
<th><strong>Identity manager</strong></th>
</tr>
</thead>
</table>
| WSO2 Identity Server supports auditing of privileged operations using distributed auditing system (XDAS). It also allows you to monitor and collect standard access and performance statistics. The Analytics component of WSO2 Identity Server supports monitoring session and authentication statistics. | Auditing does not play a direct role in the process flow but as a component, it integrates with various other components in the Identity Server. IS can be configured to produce audit logs for all of its components but the following components are most commonly used for logging details.  
- Authentication framework  
- Provisioning framework  
- User store manager |
| **Identity manager** | **Identity Manager** |
| Enterprise IT Systems are constantly changing; their perimeters are expanding and their policies keep changing. Therefore, in such a rapidly evolving world, security solutions need to be forward thinking and innovative. They need to be configurable in order to keep pace and adapt to rapid changes. This can be achieved by the identity manager component because it caters to security requirements at hand as well as looking toward the future. It has a very customizable user interface and can be easily implemented in order to ensure maximum security for your system. | Identity manager does not play a direct role in the process flow but as a component, it integrates with various other components in the Identity Server, primarily the user store manager. |
The Outbound Provisioning component of the Identity Server can send provisioning requests to applications that support the following connectors:

- SCIM
- SPML
- Google
- Salesforce

These connectors reach out to identity providers that perform the provisioning.

The provisioning request comes into the outbound provisioning component from the provisioning framework. This request will go to the relevant connector.

---

Related Links

For more information on how to configure outbound provisioning connectors, see Configuring Outbound Provisioning Connectors for an Identity Provider.

---

Related Links

For further reading about the architecture in an Identity and Access Management solution, see the following article: Identity Architect Ground Rules: Ten IAM Design Principles.