



WSO2 API Manager 4.2.0 Developer Fundamentals

Product Administration



WSO2 Training

CC by 4.0

Introduction to User Management

- Defining and managing
 - Users
 - Roles
 - Permission

Link - [User Management](#)



Managing User Roles

- Roles contain permissions for users to manage the server.
 - Can be reused.
 - Eliminate the overhead of granting permissions to users individually.
- Following Roles exist by default,
 - Admin
 - Internal/everyone
 - Internal/system
 - Internal/analytics
 - Internal/creator
 - Internal/subscriber
 - Internal/publisher
 - Internal/devops

Link: [User Roles](#)



admin - Provides full access to all features and controls. By default, the admin user is assigned to both the admin and the Internal/everyone roles.

Internal/everyone - This is a predefined role that is used to group all the users (across the user stores) together. When you create a new user, automatically the user belongs to the Internal/everyone role. It does not include any permissions. This role can be used to identify all logged in users.

Internal/system - This is another pre defined role which does not include any permissions. Unlike the Internal/everyone role, this role is not assigned to a user by default.

Internal/analytics - This role can be assigned to users who do not have the publisher or subscriber roles assigned but need permission to view the analytics dashboards.

Internal/creator: A creator is typically a person in a technical role who understands the technical aspects of the API (interfaces, documentation, versions etc.) and uses the API publisher to provision APIs into the Developer Portal. The creator uses the Developer Portal to consult ratings and feedback provided by API users. Creator can add APIs to the Developer Portal but cannot manage their lifecycle. Governance permission gives a creator permission to govern, manage and configure the API artifacts.

Internal/publisher: A person in a managerial role and overlooks a set of APIs across the enterprise and controls the API lifecycle, subscriptions and monetization aspects. The publisher is also interested in usage patterns for APIs and has access to all API statistics.

Internal/subscriber: A user or an application developer who searches the Developer Portal to discover APIs and use them. S/he reads the documentation and forums, ratings/comments on the APIs, subscribes to APIs, obtains access tokens and invokes the APIs.

Internal/devops: Has the ability to perform all the apictl related operations. You can create a new user and assign this role to do the apictl operations.

Create a New User Role

The screenshot displays the WSO2 API Manager console interface. On the left, a sidebar menu shows the navigation path: Home > Add New Role. The main content area is titled 'Add New Role' and shows 'Step 1: Enter Role Details'. The form includes a 'Domain' dropdown set to 'PRIMARY' and a 'Role Name*' text input field containing the value 'creator'. Both the 'Add' button in the sidebar and the 'Role Name*' input field are highlighted with red rectangles. At the bottom of the form, there are 'Next >', 'Finish', and 'Cancel' buttons.

Log into the API Manager carbon console (<https://localhost:9443/carbon>) as admin user.
(username: admin, password: admin)

Go to,

Users and Roles -> Add -> Add new Role

Provide the name of the role and click Finish.

Managing Role Permissions

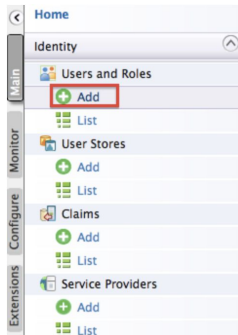
- Permissions can be granted to a role at two levels:
 - **Super Tenant level:** A role with super tenant permissions is used for managing all the tenants in the system and also for managing the key features in the system, which are applicable to all the tenants.
 - **Tenant level:** A role with tenant level permissions is only applicable to individual tenant spaces.

Link : [Role Permissions](#)



Managing Users

- Adding User
 - Log in to the Management Console (<https://<hostname>:9443/carbon>) and click **Add** under **Users and Roles** in the **Main** menu.

This screenshot shows the 'Add New User' form, specifically Step 1: Enter Username and Password. The form includes fields for 'Domain' (set to PRIMARY), 'Username*' (filled with 'apicreator'), 'Password*' (masked with dots), and 'Confirm Password*'. At the bottom, there are 'Next >', 'Finish', and 'Cancel' buttons.This screenshot shows the 'Add New User' form, specifically Step 2: Select Roles of the User. It features a search bar for 'Enter Role Name Pattern (* for all)' and a list of roles. The 'creator' role is selected and highlighted with a red box. Other roles listed include 'admin', 'Internal/analytics', 'Internal/creator', 'Internal/everyone', 'Internal/publisher', 'Internal/subscriber', and 'Internal/system'. At the bottom, there are 'Finish' and 'Cancel' buttons.


Managing Users

Home

Users

Search Users

Select Domain	ALL-USER-STORE-DOMAINS ▾
Enter Username Pattern (* for all)	* <input type="button" value="Search Users"/>
Select Claim URI	Select ▾

Name	Actions
admin	 Change Password  Assign Roles  View Roles  Delete  User Profile
apicreator	 Change Password  Assign Roles  View Roles  Delete  User Profile



Managing Scope-Role Mappings

- WS02 API Manager Publisher and Developer Portals are based on API Manager REST APIs, which are secured with OAuth2. Each resource is bound with specific scopes.
- Each scope is associated with API Manager internal roles. (Internal/publisher, Internal/subscriber, etc.)
- To grant the access to the newly created user with creator role, it should be mapped to a proper internal role.

E.g., creator -> Internal/creator



Scope: Scopes are Role based access control mechanism which can be enforced to API Resources.

If a scope is associated with a resource, the access token used to invoke the resource must have the required scope.

Managing Scope-Role Mappings

- Log in to the API Manager Admin Portal (<https://<hostname>:9443/admin>) and go to Settings -> Scope Assignments.

The screenshot shows the 'Scope Assignments' page in the API Manager Admin Portal. The page has a header with 'Settings / Scope Assignments' and an 'ADMIN' dropdown. Below the header, there's a sub-header 'Scope Assignments' with a note: 'Scope assignments are only related to internal, APIM-specific scope assignments. They are not related to role permission assignments in the Management Console.' The main content area features a search bar 'Search by Role Name' with an 'Add scope mappings' button and a refresh icon. Below this is a table with two columns: 'Roles' and 'Scope Assignments'. The 'Roles' column lists 'admin', 'Internal/analytics', 'Internal/creator', 'Internal/devops', and 'Internal/integration_dev'. The 'Scope Assignments' column contains 'Scope Assignments' and 'Delete' buttons for each role. At the bottom, there's a pagination control showing 'Rows per page: 5' and '1-5 of 8'.

Roles	Scope Assignments
admin	Scope Assignments Delete
Internal/analytics	Scope Assignments Delete
Internal/creator	Scope Assignments Delete
Internal/devops	Scope Assignments Delete
Internal/integration_dev	Scope Assignments Delete

Rows per page: 5 1-5 of 8

Managing Scope-Role Mappings

- Click on Add scope mappings.

The image displays three sequential screenshots of the 'Add new scope mapping' wizard in a web application.

Screenshot 1: Provide role name
The first step shows a form with a 'Role Name' field containing the text 'creator'. Below the field, a note states: 'Type existing user role, if not create a new role from carbon console first'. The second step, 'Select scope assignments', is indicated by a blue dot.

Screenshot 2: Select scope assignments
The second step shows a 'Mapping role' dropdown menu with 'Internal/creator' selected. A red box highlights this dropdown. Below it, a note states: 'Please check whether the required permissions are assigned to the role to function with the assigned scope, before proceeding'. The third step, 'Custom scope assignments', is indicated by a blue dot.

Screenshot 3: Custom scope assignments
The third step shows a list of permissions under the heading 'Scope Assignments (49)'. The permissions are grouped into categories: 'administrative', 'store', 'publisher', 'Create threat protection policies', 'Update and delete mediation policies', 'Update and delete backend endpoint certificates', 'View backend endpoint certificates', 'Publish API', 'Update and delete client certificates', 'View API', 'Create mediation policies', 'Get/subscribe/configure publisher alerts', 'Update and delete API documents', 'Create API documents', 'Update and delete threat protection policies', 'View Subscription', 'Create API', 'Manage shared scopes', and 'Add client certificates'. A red box highlights the 'Add client certificates' option at the bottom of the list.

Refer

<https://apim.docs.wso2.com/en/4.2.0/administer/managing-users-and-roles/managing-permissions/#adding-api-m-specific-scope-assignments>

Managing Scope-Role Mappings - Role Alias

- New roles can be mapped to existing **Internal/*** roles, created roles, and admin. All the scopes associated with the selected existing role will be mapped to the new role automatically.
- If you want to map the scopes of **Internal/creator** to the new **creator** role, select **Internal/creator** from the drop-down menu and save.
- This will update all scope mappings in the **tenant-conf.json** file with **Internal/creator** as an allowed role resulting in the new **creator** role to be allowed for all scopes allowed for the **Internal/creator** role.



Introduction to User Stores

- A user store is the database where information of the users and/or user roles are stored.
- User Information:
 - ⦿ Usernames, Passwords, First Name, Last Name, Email, etc.
- By default it uses an embedded h2 database.
- Permissions and other authorization related information is stored in a separate database called the user management database, which by default is H2 as well.
- You can also
 - ⦿ Configure a **primary user store** instead of using embedded h2 database.
 - ⦿ Configure several **secondary user stores** as well.
 - ⦿ Configure your own **customized user stores** and connect them with the products as secondary stores.



Docs Link: [User Stores](#)

User Store Types

There are four user store types.

User Store Manager types.

User store type	User store manager class	Description
read_only_ldap	<code>org.wso2.carbon.user.core.ldap.ReadOnlyLDAPUserStoreManager</code>	Use <code>read_only_ldap</code> to do read-only operations for external LDAP user stores.
read_write_ldap	<code>org.wso2.carbon.user.core.ldap.ReadWriteLDAPUserStoreManager</code>	Use <code>read_write_ldap</code> for external LDAP user stores to do both read and write operations.
active_directory	<code>org.wso2.carbon.user.core.ldap.ActiveDirectoryUserStoreManager</code>	Use <code>active_directory</code> to configure an Active Directory Domain Service (AD DS) or Active Directory Lightweight Directory Service (AD LDS). This can be used only for read/write operations. If you need to use AD as read-only, you must use <code>read_only_ldap</code> .
database	<code>org.wso2.carbon.user.core.jdbc.JDBCUserStoreManager</code>	Use <code>database</code> for both internal and external JDBC user stores. This is the user store configuration which is configured by default.



Configuring a Secondary User Store

1. Log in to the carbon console and, go to User Stores > Add
2. Select the user store type
3. Enter the user store domain.
4. Enter the user store connection properties
5. Click the Add button to add the user store.

Management Console
Signed-in as admin@carbon.super | Sign-out | Docs | About

Home > Identity > User Stores > Add

Add New User Store

User Store Manager

User Store Manager Class: org.wso2.carbon.user.core.jdbc.UniqueIDJDBCUserStoreManager
Depending on the class, properties needs to be defined.

Domain Name: acme.com

Description:

Define Properties For Acme.Com

Property Name	Property Value	Description
Connection URL *	jdbc:mysql://localhost:3306/	URL of the user store database
Connection Name *	userstore	Username for the database
Connection Password *	root	Password for the database
Driver Name *	com.mysql.jdbc.Driver	Full qualified driver name

Optional

Advanced

Test Connection Add Cancel

Configuring a Secondary User Store Manually

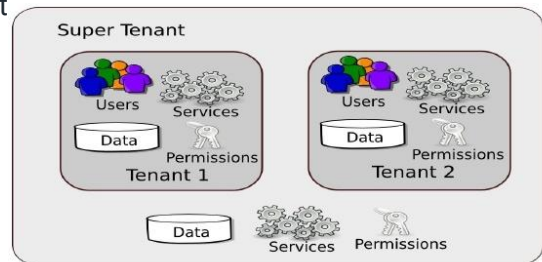
- When you configure multiple user stores, you must **give a unique domain name to each user store** in the `<DomainName>` element.
- If it is the configuration of a super tenant, save the secondary user store definitions in `<APIM_HOME>/repository/deployment/server/userstores` directory.
- If it is a general tenant, save the configuration in `<APIM_HOME>/repository/tenants/<tenantid>/userstores` directory
- The secondary user store configuration file must have the same name as the domain with an underscore (_) in place of the period. For example, if the domain is `wso2.com`, name the file as `wso2_com.xml`
- Only one file should contain the definition for one user store domain.

Configuring a Secondary User Store Manually



Introduction to Multi Tenancy

- Maximize resource sharing by allowing multiple users (tenants) to log in and use a single server/cluster at the same time, in a tenant-isolated manner.
- Ensures optimal performance of the system's resources such as memory and hardware
- Secures each tenant's personal data.
- Register tenant domains using the Management Console.
- Features of multi tenanted environment
 - Tenant isolation
 - Data isolation
 - Execution isolation
 - Performance Isolation



Link : [Introduction to Multi tenancy](#)

What is Multi Tenancy?

Logically isolated entities, sharing the same infrastructure.

All tenants share the same database, user store and other services. But, one tenant's information cannot be accessed by another tenant.

- **Tenant isolation** Each tenant has its own domain, which the other tenants cannot access.
- **Data isolation** Each tenant can manage its data securely in an isolated manner.
- **Execution isolation** Each tenant can carry out business processes and workflows independent of the other tenants. No action of a tenant is triggered or inhibited by another tenant.
- **Performance Isolation** No tenant has an impact on the performance of another tenant.

Multi Tenancy

- An Individual Tenant can perform the following:
 - ◉ Deploying artifacts
 - ◉ Applying Security
 - ◉ User Management
 - ◉ Data Management
 - ◉ Request Rate Limiting
 - ◉ Response Caching
- Methods for sharing resources among tenants
 - ◉ Private Jet mode
 - ◉ Separation at hardware level
 - ◉ Separation at JVM level
 - ◉ Native multi tenancy



- **Private Jet mode** : This method allows the load of a tenant ID to be deployed in a single tenant mode. A single tenant is allocated an entire service cluster. The purpose of this approach is to allow special privileges (such as priority processing and improved performance) to a tenant.
- **Separation at hardware level** : This method allows different tenants to share a common set of resources, but each tenant has to run its own operating system. This approach helps to achieve a high level of isolation, but it also incurs a high overhead cost.
- **Separation at JVM level** : This method allows tenants to share the same operating system. This is done by enabling each tenant to run a separate JVM instance in the operating system.
- **Native multi tenancy** : This method involves allowing all the tenants to share a single JVM instance. This method minimises the overhead cost.

Updating WSO2 API Manager Using WSO2 Updates 2.0

- **WSO2 Updates 2.0** : WSO2 Updates include improvements that are released by WSO2, on top of a released WSO2 product version. With updates, you do not have to wait until the next product version release to get the product enhancements and security fixes.
- Refer <https://updates.docs.wso2.com/en/latest/updates/overview/> for more details on how to update WSO2 APIM 4.2.0 using Updates 2.0.





Let's try it out!

Working with Tenants

