Cello How-To Guide

Security Management

techcello

Contents

1 Security Management

CelloSaaS allows the developers to build security management in their application more easily. The following objects can be secured using CelloSaaS

- Pages/Actions
- Service

- CelloSaaS achieves the security using privileges. Developer need to demand the right privilege in the code for Pages / Actions [presentation tier], Service [Service calls], Data and Fields. Privileges are available to the system via the license package modules and features.
- The Privileges for user is derived from the User's Tenant License Package and his Roles Privilege.
- Privileges are mapped to roles. Any application user will get access to the applications features provided he / she possesses the right privilege based on the user's tenant license.

1.1 Introduction to Privilege

- Privilege is used for fine grained level of access control management.
- Privilege will be associated with features referred to as feature privilege.
- Privilege will be mapped with Entity to restrict the access of an entity actions like add, edit, view, delete and etc.
- Privilege Id and name should be unique. Entity action privileges should follow the naming conversion "{Action}_{EntityId}".

1.2 Feature Privilege

Privileges are identified and configured during the development time and not at runtime.

Example:

While building a Feature called Leave Application Form and Leave List Page, some of the privileges such as Add Leave, View Leave details, Delete Leave, Edit Leave etc can be identified and configured in the system.

Note: In case if a new privilege has to be added in the system, it can only be configured in the code or in the database.

1.3 Configure feature privilege through Fluent API

1.3.1 Through XML Configuration

In Xml configuration add necessary privileges under the Feature xml tag like below.

```
<ModuleConfiguration>
  <Modules>
```

```
    <Module Code="ProjectModule" Name="Project Module" Description="This module contains
feature and usgae of the project">
      <Features>
        <Feature Code="ProjectFeature" Name="Project Feature">
          <Privileges>
            <Privilege Name="Search Project" Description="user can search their Project
list">Search_Project</Privilege>
            <Privilege Name="View Project" Description="user can View their Project
list">View_Project</Privilege>
            <Privilege Name="Add Project" Description="user can add their
Project.">Add_Project</Privilege>
            <Privilege Name="Edit Project" Description="user can Edit their
Project.">Edit_Project</Privilege>
            <Privilege Name="Delete Project" Description="user can delete their
Project.">Delete_Project</Privilege>
            <Privilege Name="Approve Project" Description="user can Approve the
Project.">Approval_Project</Privilege>
          </Privileges>
        </Feature>
      </Features>
    </Module>
  </Modules>
</ModuleConfiguration>
```

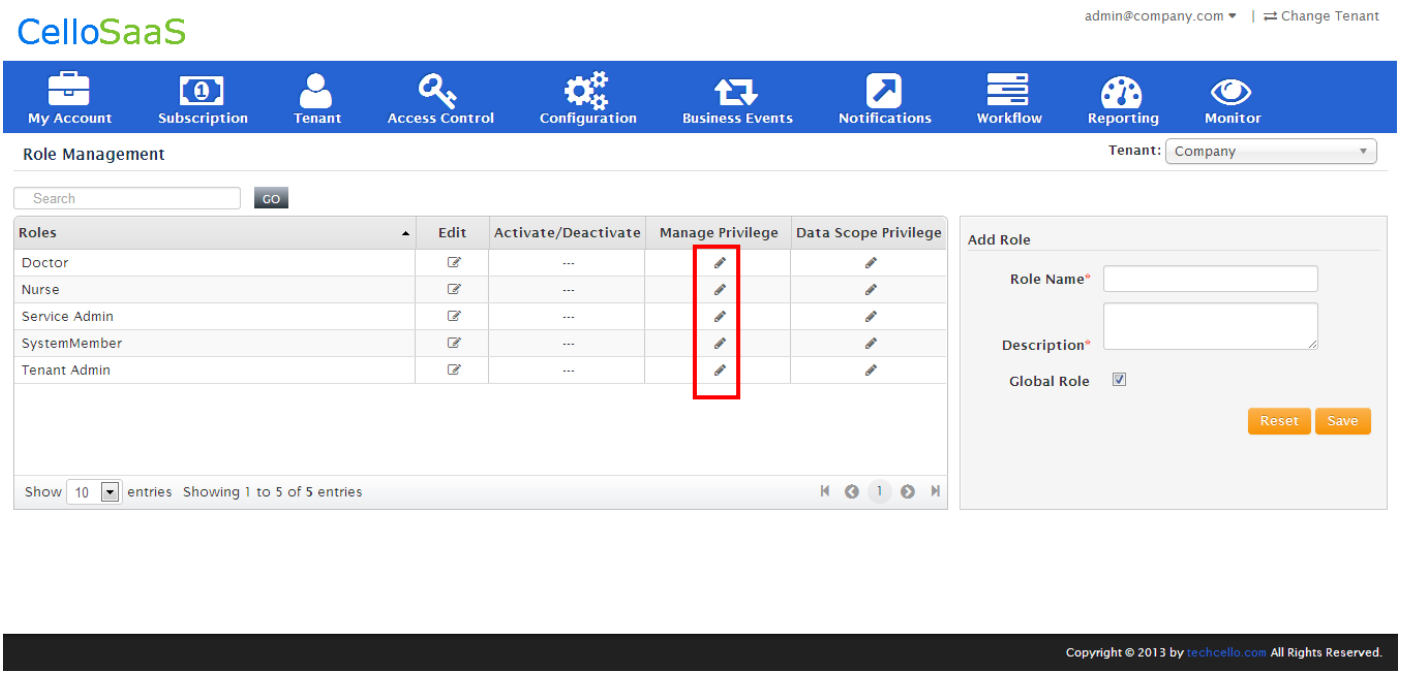
1.3.2 Through Fluent API

In module configuration object add privileges against the feature using WithPrivilege() method.

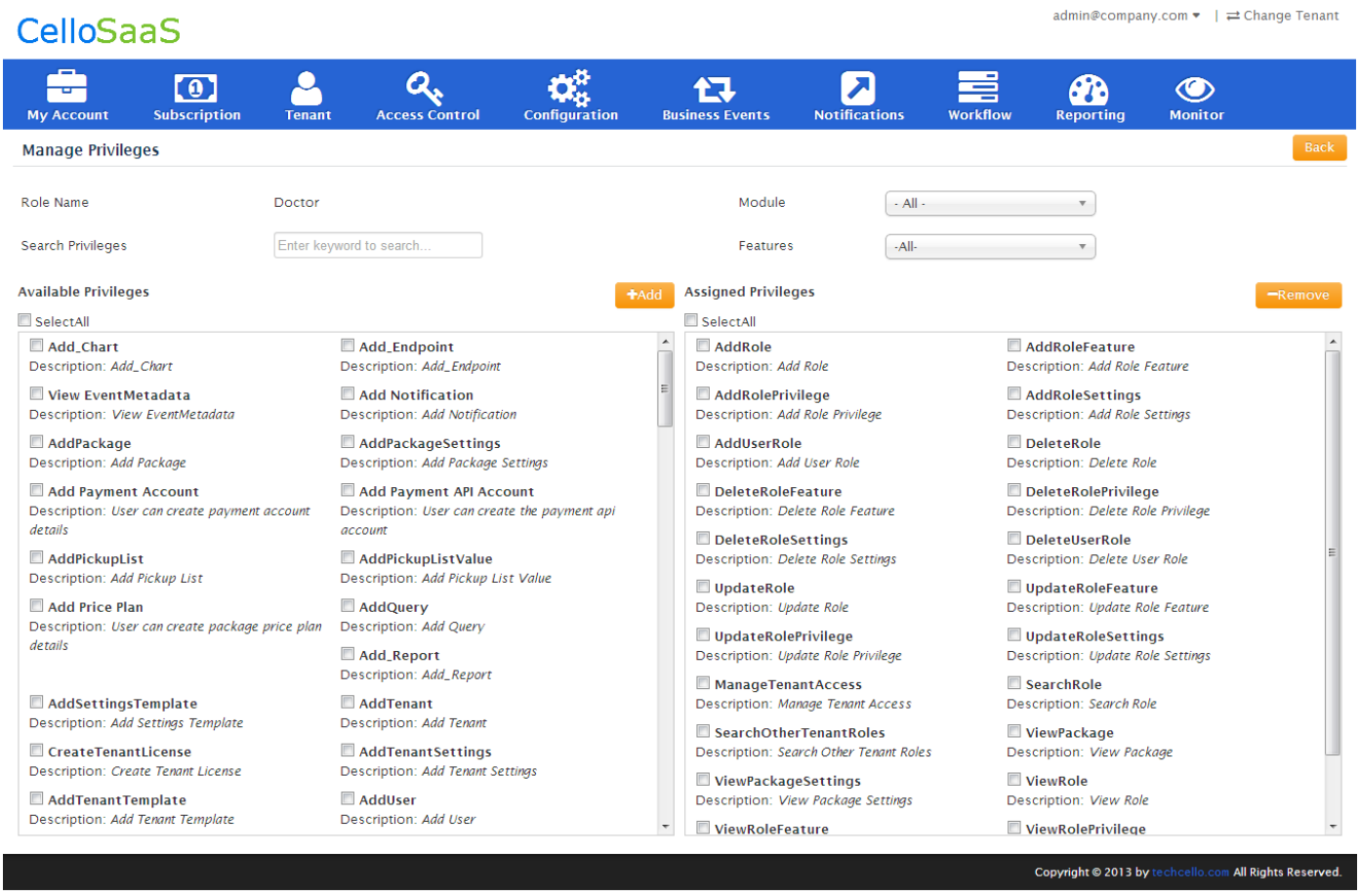
```
public void Configure(ModuleConfig moduleConfiguration)
{
        moduleConfiguration.Add("Employee", "Employee")
        .WithFeatures(f => f.Add("ManageEmployee").WithName("Manage Employee")
      .WithPrivileges((p => p.Add("ViewEmployee").WithName("View Employee")
                .WithDescription("User can View Employee details.")
                .Add("AddEmployee").WithName("Add Employee")
                .WithDescription("User can Add Employee details.")
                .Add("EditEmployee").WithName("Edit Employee")
                .WithDescription("User can Edit Employee details.")
                .Add("DeleteEmployee").WithName("Delete Employee")
                .WithDescription("User can Delete Employee details."))));
}
```

1.4 Role Privilege

- Each role will be mapped with one or many privileges.
- Role and privilege combination must be unique for each tenant.
- Select **Access Control -> Manage Roles**. This page will list the available roles for logged-in user tenant. For more details refer **Role Management** in click here.
- Click Manage Privilege icon to map privilege for the particular role.



- Role privilege page will display all feature privileges based on the tenant license package.
- This page contains two sections named Available Privileges and Assigned Privileges.



- Assigned Privilege section contains the privileges of selected role.
- Available privilege section contains the privileges that can be assigned.

Adding privilege [s] to a role

- Select privilege [s] from Available privilege section and click Add button to add to a role.

Revoking privilege [s] from a role

- Select privilege [s] from Assigned privileges section and click Remove button to delete a privilege from a role.

Modules & Feature filtering for Privileges

- Module lists the available modules from tenant license package.
- Feature lists the available features from tenant license package.
- You can filter the privileges based on Modules and Features.

1.5 User Privileges

To get the logged in user privileges from his identity use ***CelloSaaS.Library.UserIdentity.Privileges***. This contains the privileges for the user that has logged in to the system.

1.6 Page Level Security

- Controllers must inherit from ***CelloSaaS.View.CelloController*** to perform action method level security restrictions.
- Page level access is obtained by the configuration in "***EntityPermission.Config***" file.
- All URL's mapped to the required access rule which is then checked automatically whenever that page or action is called
- CelloSaaS menu provider is integrated with entity permission rule so turn off the menus to the page access rule is not met.
- Access rule is combination of Privileges, Roles, Identity and Tenant Settings.
- Example: ***"P:Add_Employee AND R:GR$Tenant_Admin".*** If the logged-in user has Add Employee privilege and Tenant Admin role then user can access the particular page.
- CelloSaaS provide the default entity permission configuration for CelloSaaS admin pages.

1.7 Default Entity Configuration

```
<EntityPermission>
  <EntityCategory>
    <add name="UI">
      <Entity>
        <add name="/tenantuserassociation/othertenantuserslist" AuthorizationRule="S:ShareUsers AND R:GR$Tenant_Admin" />
        <add name="/tenantuserassociation/linkedbyothertenantslist" AuthorizationRule="S:ShareUsers" />
        <add name="/home/admin" AuthorizationRule="P:View_Tenant OR P:View_Package OR P:View_TenantTemplate OR P:View_User OR (S:ShareUsers AND R:GR$Tenant_Admin) OR
             P:View_Role OR P:Manage_TenantAccess OR P:Manage_Entity OR P:Manage_DataView OR P:View_SettingsTemplate OR P:Manage_ModuleConfiguration OR P:View_Rules OR
             P:View_PickupList OR R:GR$Product_Admin OR P:View_Notification OR P:View_Audit OR P:View_Databackup OR R:GR$Tenant_Admin"/>
        <add name="/home/tenant" AuthorizationRule="P:View_Tenant OR P:View_Package OR P:View_TenantTemplate"/>
        <add name="/tenant/*" AuthorizationRule="P:View_Tenant"/>
        <add name="/package/packagelist" AuthorizationRule="P:View_Package"/>
        <add name="/settingstemplate/tenantsettingstemplate" AuthorizationRule="P:View_TenantTemplate"/>
        <add name="/home/user" AuthorizationRule="P:View_User OR (S:ShareUsers AND R:GR$Tenant_Admin)"/>
        <add name="/user/*"  AuthorizationRule="P:View_User"/>
        <add name="/home/defaultaccess" AuthorizationRule="P:View_Role OR P:Manage_TenantAccess"/>
        <add name="/roles/*" AuthorizationRule="P:View_Role"/>
        <add name="/entitytenantscope/*" AuthorizationRule="P:Manage_TenantAccess"/>
        <add name="/home/configuration" AuthorizationRule="P:Manage_Entity OR P:Manage_DataView OR P:View_SettingsTemplate OR P:Manage_ModuleConfiguration OR P:View_Rules
                                                           OR P:View_PickupList OR R:GR$Product_Admin"/>
        <add name="/data/*" AuthorizationRule="P:Manage_Entity"/>
        <add name="/dataview/*" AuthorizationRule="P:Manage_DataView"/>
        <add name="/settingstemplate/index" AuthorizationRule="P:View_SettingsTemplate"/>
        <add name="/configuration/moduleconfiguration" AuthorizationRule="P:Manage_ModuleConfiguration"/>
        <add name="/rules/*" AuthorizationRule="P:View_Rules"/>
        <add name="/configuration/pickuplist" AuthorizationRule="P:View_PickupList"/>
        <add name="/configuration/relationshipdetailslist" AuthorizationRule="P:View_PickupList"/>
        <add name="/masterdata/*" AuthorizationRule="R:GR$Product_Admin"/>
        <add name="/notificationconfig/notificationmasterdetails" AuthorizationRule="P:View_Notification"/>
        <add name="/audit/*" AuthorizationRule="P:View_Audit"/>
        <add name="/events/searcheventaudit" AuthorizationRule="P:View_Audit"/>
        <add name="/notificationconfig/manageauditdetails" AuthorizationRule="P:View_Audit"/>
        <add name="/databackup/*" AuthorizationRule="P:View_Databackup"/>
        <add name="/databackup/managebackuprequest" AuthorizationRule="P:Create_Databackup"/>
        <add name="/cache/refreshcache" AuthorizationRule="R:GR$Product_Admin OR R:GR$Tenant_Admin"/>
        <add name="/home/reporting" AuthorizationRule="P:View_Query OR P:View_Chart OR P:View_Report"/>
        <add name="/querybuilder/*" AuthorizationRule="P:View_Query"/>
        <add name="/chartdetails/*" AuthorizationRule="P:View_Chart"/>
        <add name="/reportbuilder/*" AuthorizationRule="P:View_Report"/>
        <add name="/tablesources/*" AuthorizationRule="P:View_Report"/>
        <add name="/home/workflow" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/workflow/*" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/workflowdesigner/index" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/workflowdesigner/createnewdesign" AuthorizationRule="P:Add_WorkflowDesign"/>
        <add name="/workflowdesigner/managedesign" AuthorizationRule="P:Edit_WorkflowDesign"/>
        <add name="/workflowdesigner/deleteworkflow" AuthorizationRule="P:Delete_WorkflowDesign"/>
        <add name="/workflowinstance/searchwfinstance" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/workflowinstance/searchwftaskinstance" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/workflowdesigner/publish" AuthorizationRule="P:Publish_WorkflowDesign"/>
        <add name="/wfdashboard/index" AuthorizationRule="P:View_WorkflowDesign"/>
        <add name="/events/index" AuthorizationRule="P:View_EventMetadata"/>
        <add name="/tenant/tenantsettings" AuthorizationRule="P:View_TenantSettings"/>
        <add name="/serviceendpoint/*" AuthorizationRule="P:View_Endpoint"/>
        <add name="/serviceendpoint/manage" AuthorizationRule="P:Edit_Endpoint OR P:Add_Endpoint"/>
        <add name="/serviceendpoint/delete" AuthorizationRule="P:Delete_Endpoint"/>
      </Entity>
    </add>
  </EntityCategory>
</EntityPermission>
```

- Entity permission will have two category named UI and BL.
- Page level configurations are to be added in the UI section under the Entity tag. Set the URL or action in name attribute. Specify the access rule in AuthorizationRule attribute.

Use the following Authorization Rule formats.

	Format	**Example**
Role	R:RoleId	R:GR$Tenant_Admin
Privilege	P:PrivilegeId	P:View_Employee
Identity	I:Identity	I:admin@company.com
Tenant Settings	S:TenantSettingsId	S:ShareUsers

- Access rule can be constructed as a complex expression containing AND, OR and NOT.
- You can also combine permission, role and tenant settings.

Example:

AuthorizationRule="S:ShareUsers AND R:GR$Tenant_Admin AND P:View_User"

1.8 Action Level Security

- You can check the necessary privileges before start executing the method in Services.
- You can do either using **CheckPrivilge** method or **PrivilegeContext**.
- Use **CheckPrivilege** method to check required privilege for a service method.

Namespace : CelloSaaS.Services.AccessControlManagement
Service: AccessControlService
Method : CheckPrivilege(string privilege)

```
/// <summary>
/// This method is used to Check if the user's role has access to the privilege

bool CheckPrivilege(string privilege)
```

Sample:

```
public EmployeeDetails GetEmployeeDetailsByEmployeeID(string employeeID)
{
    IAccessControlService accessControlService = (IAccessControlService)
    ServiceLocator.GetServiceImplementation( typeof(IAccessControlService));

     // Check whether the user has permission to add user details or not.
     if (!accessControlService.CheckPrivilege(
    ManageEmployeeEmployeeDetailsConstants.ViewEmployeeDetails))
     {
                throw new UnauthorizedAccessException("Permission to access Get EmployeeDetails By
                EmployeeID is denied.");
     }

    // Write logic to get employee details by employee id.
}
```

- If logged-in user doesn't have privilege to access the particular method then throw ***UnauthorizedAccessException***.
- You can use ***PrivilegeContext*** to check the privilege in another way.
- To access the service, demand the privilege by using ***CelloSaaS.Library.Context*** which is provided by CelloSaaS.

- Context has the following ethods:

```
/// This method is used to create PrivilegeContext instance and that will be add in HttpContext.Current.Items with “Privilege” as key
void SetPrivilegeContext(string privilegeName);
```

- Usually privilege will be demanded from the controller before calling Proxy methods by using SetPrivilegeContext method.
- Example:
- CelloSaaS.Library.Context.SetPrivilegeContext(“Edit_EmployeeDetails”);
- In services, current privilege will be taken from HttpContext by using GetCurrentPrivilegeContext method.

```
/// This method is used to get the PrivilegeContext instance which is currently present.
CelloSaaS.Library.PrivilegeContext GetCurrentPrivilegeContext();
```

Example

```
public EmployeeDetails GetEmployeeDetailsByEmployeeID(string employeeID)
{
    string currentPrivilege = string.Empty;
    PrivilegeContext privilegeContext =Context.GetCurrentPrivilegeContext();
    if (privilegeContext != null)
    {
        currentPrivilege = privilegeContext.PrivilegeName;
    }

    if (!string.IsNullOrEmpty(currentPrivilege) && !UserIdentity.HasPrivilege(currentPrivilege))
    {
        throw new UnauthorizedAccessException("Permission to access Get EmployeeDetails By EmployeeID is denied.");
    }

    // Write logic to get employee details by employee id.
}
```

1.9 Service Level Security

- Service level security is possible via Entity Permission ***config*** file.
- Need to add the Service Contract name under the Entity Category **BL**.
- Add the Contract name under entity node. Add the necessary method names with the authorization rule under Entity Sub Element.

Example

```
<EntityPermission>
 <EntityCategory>
  <add name="BL">
   <Entity>
    <add name="CelloSaaS.ServiceContracts.UserManagement.IUserDetailsService">
     <EntitySubElement>
      <add name="SearchUserDetails" AuthorizationRule="P:View_User"></add>
     </EntitySubElement>
    </add>
   </Entity>
  </add>
 </EntityCategory>
</EntityPermission>
```

2 Contact Information

Any problem using this guide (or) using Cello Framework. Please feel free to contact us, we will be happy to assist you in getting started with Cello.

Email: support@techcello.com

Phone: +1(609)503-7163

Skype: techcello