# Office AMS: Tenant permissions

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to pull permissions from each site collection and site in a SharePoint Online tenant. | * Office 365 Multi Tenant (MT) |
| Solution: | Core.PermissionListing, version 1.0 |
| Author: | Brian T. Jackett (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# Scenario 1: Display permissions across all “non-my site” site collections

This scenario shows how to retrieve all permissions from each “non-My Site” site collection. This scenario relies on the SharePoint Search Service Application to be implemented and configured to crawl any site collection that you wish to pull permissions from. It also borrows from the Contoso.Management.SiteEnumeration project for querying the search service to return search results for site collections. Retrieval of SharePoint permissions will be performed via CSOM in a provider hosted app.

## Display permissions

In the current version of the app there is no user interaction. When the app is launched the CSOM code will query the tenant for any non-My Site site collections that exist in the search index. This is performed by the **GetSites** method which also relies on the **ProcessQuery** helper method (adapted from Contoso.Management.SiteEnumeration solution).

private List<Site> GetSites(ClientContext clientContext)

{

List<Site> sites = new List<Site>();

KeywordQuery keywordQuery = new KeywordQuery(clientContext);

string keywordQueryValue = "contentclass:\"STS\_Site\"";

ProcessQuery(keywordQueryValue, sites, clientContext, keywordQuery, 0);

return sites;

}

private int ProcessQuery(string keywordQueryValue, List<Site> sites, ClientContext ctx, KeywordQuery keywordQuery, int startRow)

{

int totalRows = 0;

keywordQuery.QueryText = keywordQueryValue;

keywordQuery.RowLimit = 500;

keywordQuery.StartRow = startRow;

keywordQuery.SelectProperties.Add("Title");

keywordQuery.SelectProperties.Add("SPSiteUrl");

keywordQuery.SortList.Add("SPSiteUrl", Microsoft.SharePoint.Client.Search.Query.SortDirection.Ascending);

SearchExecutor searchExec = new SearchExecutor(ctx);

ClientResult<ResultTableCollection> results = searchExec.ExecuteQuery(keywordQuery);

ctx.ExecuteQuery();

if (results != null)

{

if (results.Value[0].RowCount > 0)

{

totalRows = results.Value[0].TotalRows;

foreach (var row in results.Value[0].ResultRows)

{

sites.Add(new Site

{

Title = row["Title"] != null ? row["Title"].ToString() : "",

Url = row["SPSiteUrl"] != null ? row["SPSiteUrl"].ToString() : ""

});

}

}

}

return totalRows;

}

Once the site collections have been returned then they are processed one at a time and the permissions for each root web and subwebs are processed via the **ProcessRoleAssignments** method. Each *SecurableObject* is first checked to see if it contains unique permissions.

clientContext.Load(securableObject, x => x.HasUniqueRoleAssignments);

clientContext.ExecuteQuery();

if (!securableObject.HasUniqueRoleAssignments)

{

Response.Write("Same perms as parent" + "<br/>");

}

If the permissions are different then each of the role assignments is enumerated and the principal type (group or user), login name, and the assigned permission level are output to the app.

else

{

RoleAssignmentCollection roleAssignments = securableObject.RoleAssignments;

clientContext.Load<RoleAssignmentCollection>(roleAssignments);

clientContext.ExecuteQuery();

foreach (RoleAssignment roleAssignment in roleAssignments)

{

Principal member = roleAssignment.Member;

RoleDefinitionBindingCollection roleDef = roleAssignment.RoleDefinitionBindings;

clientContext.Load(member);

clientContext.Load<RoleDefinitionBindingCollection>(roleDef);

clientContext.ExecuteQuery();

foreach (var binding in roleDef)

{

string output = string.Format("[{0}]{1}: {2}<br/>", member.PrincipalType, member.LoginName, binding.Name);

Response.Write(output);

}

}

}