# OFFICE AMS: Provisioning site collections in on-premises from SP APP

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to extend on-premises farm to support site collection creation from provider hosted app.  Notice that after SP2013 April CU (2014), this capability is natively supported in on-premises. This example however does show how to also expose other relevant APIs from the server, which might not be exposed by using oob methods. | * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Provisioning.Services.SiteManager, version 1.0 |
| Author: | Vesa Juvonen (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.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

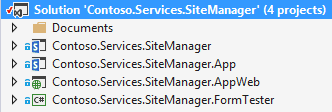
# Introduction

This sample shows how to extend on-premises farm to support site collection creation from provider hosted app. This same pattern can be used to provide other extensions as well, like exposing information management settings and other capabilities which are not natively available in the CSOM.

Site collection creation remotely is natively supported in the Office365 (MT), but same capability is not available for on-premises with CSOM. Specifically for the site collection creation, you could also use the site admin web service (siteadmin.svc) in on-premises, but this capability has been depreciated and is not supported with in Office365-D.

# Solution setup

The solution contains 4 projects:



Below you can find a short description of each of the projects:

**Contoso.Services.SiteManger**: this is the WCF end point which is to be deployed as farm solution to the on-premises farm. It’s important to notice that this is deployed to the farm just to expose needed out of the box

**Contoso.Services.SiteManger.App**: this the SP App project to introduce the app for SharePoint.

**Contoso.Services.SiteManger.AppWeb**: this is the actual provider hosted app hosted in on-premises provider hosted environment

**Contoso.Services.SiteManger.FormTester**: this is simple windows forms tester application to able to test the WCF end points in on-premise or during the WCF end point development.

# Site Provisioning

The actual site collection provisioning happens from the provider hosted app by calling custom WCF end point which have to be deployed to the farm as an extension point. This is good example of so called “smart” on-premises extensions, where we only use the farm solution to expose additional APIs for remote access, rather than actually place the business logic to the farm.

This way we can control the business logic without updating the farm solutions in the farm, which means that we can adjust the behavior without any service breaks or impact on the SharePoint services. Actual site collection creation API is exposed by the WCF end point as *CreateSiteCollection* method. We can control the configuration of the site collection by providing different configuration options using complex data type called SiteData.

SiteManager.SiteManagerClient managerClient = GetSiteManagerClient();

SiteManager.SiteData newSite = new SiteManager.SiteData()

{

Description = "",

LcId = "1033",

OwnerLogin = "contoso\\administrator",

SecondaryContactLogin = "contoso\\vesaj",

Title = DateTime.Now.Ticks.ToString(),

Url = "sites/" + DateTime.Now.Ticks.ToString(),

WebTemplate = "STS#0"

};

string url = managerClient.CreateSiteCollection(newSite);

status.Text = string.Format("Created site collection to {0}.", url);

After the site collection has been created, additional configuration options are handled by using standard CSOM APIs. In this case we assign theme settings to the just created site collection by contacting the root web of it.

using (var ctx = CreateAppOnlyClientContextForUrl(spContext, url))

{

// Deploy theme to web, so that we can set that for the site

Web web = ctx.Web;

ctx.Load(web);

ctx.ExecuteQuery();

DeployThemeToWeb(ctx, web);

//Set the properties for applying custom theme which was jus uplaoded

string spColorURL = URLCombine(web.ServerRelativeUrl, "/\_catalogs/theme/15/contoso.spcolor");

string spFontURL = URLCombine(web.ServerRelativeUrl, "/\_catalogs/theme/15/contoso.spfont");

string backGroundImage = URLCombine(web.ServerRelativeUrl, "/\_catalogs/theme/15/contosobg.jpg");

// Use the Red theme for demonstration

web.ApplyTheme(spColorURL,

spFontURL,

backGroundImage,

false);

ctx.ExecuteQuery();

// Redirect to just created site

Response.Redirect(url);

}