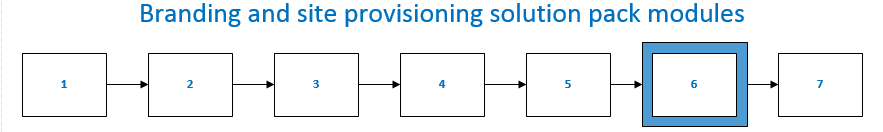
Module 6: Converting full-trust code solutions to the app model

# Introduction

This module provides guidance and introduces techniques that you can use to assess the existing SharePoint 2013 full-trust code (FTC) solutions in your enterprise, and determine whether you can and should convert the code as-is to the SharePoint app model.



# Key terms and concepts

**Table 1. Key terms and concepts**

|  |  |  |
| --- | --- | --- |
| **Term** | **Definition** | **Guidance** |
| CAM readiness | How prepared an organization is to migrate existing solutions, or otherwise adopt, the SharePoint cloud app model (CAM). |  |
| [ContentTypeId](http://msdn.microsoft.com/en-us/library/office/aa543822%28v=office.14%29.aspx) | Unique identifiers of content types that are designed to be recursive. | [ContentTypeId](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.contenttypeid%28v=office.15%29.aspx) (CSOM) |
| [Content type](http://msdn.microsoft.com/en-us/library/office/ms472236%28v=office.14%29.aspx) | A reusable collection of metadata (columns), workflow, behavior, and other settings for a category of information in centralized, reusable way. | [Columns](http://msdn.microsoft.com/en-us/library/office/ms196085%28v=office.14%29.aspx)  [Creating content types](http://msdn.microsoft.com/en-us/library/office/ms460224%28v=office.14%29.aspx)  [Custom information in content types](http://msdn.microsoft.com/en-us/library/office/ms468437%28v=office.14%29.aspx) |
| Content type hub | Part of the Managed Metadata service application; a hub site that publishes content types to other site collections. You can publish, republish, and unpublish content types centrally. | [Publish a content type from a content publishing hub](http://office.microsoft.com/en-us/sharepoint-help/publish-a-content-type-from-a-content-publishing-hub-HA102773265.aspx)  [View error logs for content type publishing](http://office.microsoft.com/en-us/sharepoint-server-help/view-error-logs-for-content-type-publishing-HA102773266.aspx) |
| [Custom field type](http://msdn.microsoft.com/en-us/library/office/ms446361%28v=office.14%29.aspx) | A customized field type created to conform to the business rules specific to an enterprise. | [How to: Customize a field type using client-side rendering](http://msdn.microsoft.com/en-us/library/office/jj220061%28v=office.15%29.aspx) |
| Hosting patterns | Ways of mixing components, requirements, technologies, and goals to match them with the possibilities that are enabled in apps for SharePoint. | [Choose patterns for developing and hosting your app for SharePoint](http://msdn.microsoft.com/en-us/library/office/fp179887%28v=office.15%29.aspx) |
| [Managed metadata](http://msdn.microsoft.com/en-us/library/office/ee559337%28v=office.14%29.aspx) | A hierarchical collection of centrally managed terms that you can use to define and attribute items in SharePoint. In SharePoint 2013 and SharePoint Online, the Managed Metadata Service (MMS) is used to create managed navigation—site navigation powered by taxonomy. | [Managed metadata and navigation in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj163949%28v=office.15%29.aspx)  [Managed navigation in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj163978%28v=office.15%29.aspx) |
| Migration | In the SharePoint context, means migrating custom SharePoint solution from full-trust code (FTC) and the feature framework to client-side code (CSOM). | This document, part of Module 6, provides migration guidance. As well, see the following resources:  [Real world examples of FTC-to-CAM transformations](http://blogs.msdn.com/b/vesku/archive/2014/03/12/spc14-session-recordings-and-slides.aspx) (Vesa Juvonen)  [SharePoint Power Hour – new SharePoint developer APIs and features](http://channel9.msdn.com/Events/SharePoint-Conference/2014/SPC3999) (Rob Howard)  [Upgrade and migrate to SharePoint 2013](http://technet.microsoft.com/en-us/sharepoint/fp142375.aspx) (TechNet) |
| Provisioning patterns | Schemes for using the app model to provision custom functionality to SharePoint sites. | Branding and site provisioning solution pack Module 5  [Site provisioning techniques and remote provisioning in SharePoint](http://blogs.msdn.com/b/vesku/archive/2013/08/23/site-provisioning-techniques-and-remote-provisioning-in-sharepoint-2013.aspx) (Vesa Juvonen) |
| Replacement technique | An iterative way to slowly migrate specific pieces of SharePoint sites from full-trust code solutions to the app model. |  |
| Sideloading | The ability to install an app for SharePoint directly to a site to explicitly bypass the regular governance controls of SharePoint. | [Enable app sideloading in your non-developer site collection](http://blogs.msdn.com/b/officeapps/archive/2013/12/10/enable-app-sideloading-in-your-non-developer-site-collection.aspx) (Apps for Office and SharePoint blog)  [Use sideloading to provision a provider-hosted app](http://code.msdn.microsoft.com/SharePoint-2013-Use-315c531b) |
| Page manipulation  Web part and app part manipulation. | Provision and manipulate a wiki page in a wiki library. |  |

# Assessing existing and targeted solutions

When considering whether to migrate full-trust code solution from SharePoint 2013 on-premises or SharePoint online to the app model, consider costs and benefits from several perspectives:

* Business requirements the solution has served, and the downstream impact of migration
* Budget
* Dependencies
* Institutional knowledge, viewpoints, and principles
* Skills
* Relationships between conceptual, logical, and physical elements (people, processes, and objects)
* Technical, regulatory, and/or process constraints
* An understanding and inventory of the baseline technical and process architecture, and the targeted architecture
* The degree of support that both maintaining existing FTC solutions and migrating to the app model have within the organization

When consider how to migrate FTC solutions to the SharePoint app model, first consider what the original code was designed to accomplish, and for whom. If the business need remains unchanged or has changed only slightly, it may make sense to analyze existing .NET server code and look for equivalents in CSOM. However, in many cases the mappings aren’t one-to-one, and the outcome produced by an equivalent app may produce slightly different effects.

## Prerequisite: Upgrade from SharePoint 2010 to SharePoint 2013

To migrate from custom FTC solutions to the app model, first upgrade from SharePoint 2010 to SharePoint 2013 on-premises or SharePoint Online.

**Important** You must upgrade from SharePoint 2010 to SharePoint 2013 before you can migrate FTC solution to the app model.

[Branding issues that may occur when upgrading to SharePoint 2013](http://office.microsoft.com/en-us/office365-sharepoint-online-enterprise-help/branding-issues-that-may-occur-when-upgrading-to-sharepoint-2013-HA104052656.aspx)

[SharePoint 2013 upgrade process](http://www.microsoft.com/en-us/download/details.aspx?id=30371)

[Plan your upgrade to SharePoint Online 2013](http://office.microsoft.com/en-us/office365-sharepoint-online-enterprise-help/plan-your-upgrade-to-sharepoint-online-2013-HA104034491.aspx)

[Upgrade databases from SharePoint 2010 to SharePoint 2013](http://technet.microsoft.com/en-us/library/cc303436%28v=office.15%29.aspx)

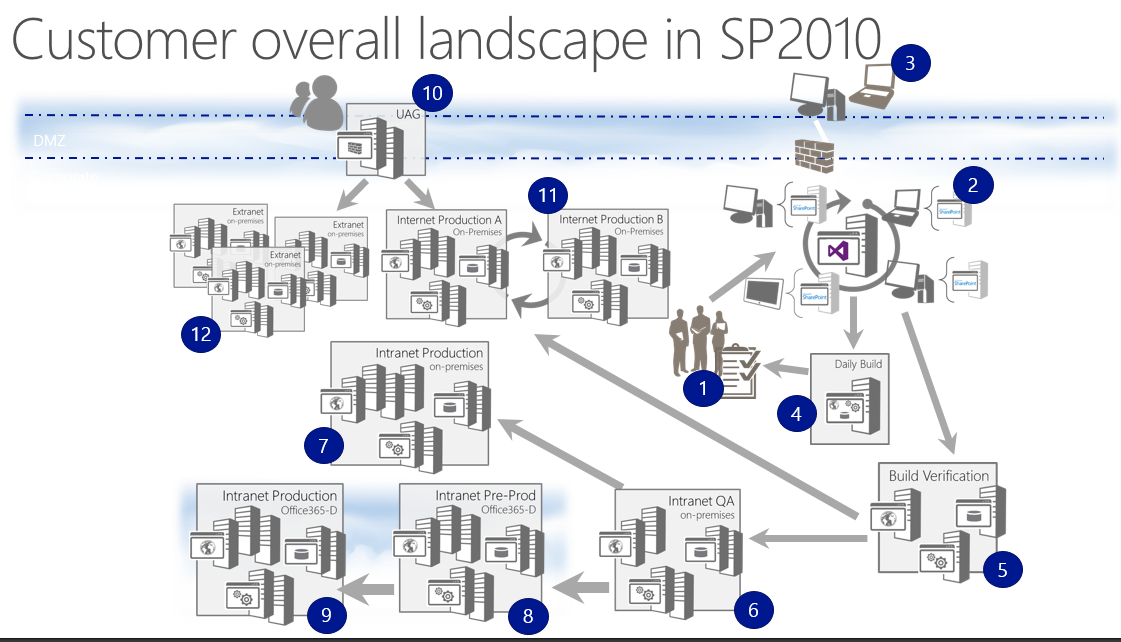
[Upgrade to SharePoint 2013](http://technet.microsoft.com/en-us/library/cc303420%28v=office.15%29.aspx)

[Upgrade web templates to SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/dn175700%28v=office.15%29.aspx)

[Use Feature upgrade to apply new SharePoint 2013 master pages when upgrading from SharePoint 2010](http://msdn.microsoft.com/en-us/library/office/dn673597%28v=office.15%29.aspx)

# Example CAM readiness assessment: Contoso

Contoso is an environmental engineering company. They are a SharePoint Online Dedicated (SPO-D) customer who began assessing their readiness for app model adoption while still running SharePoint 2010. They used it to host their corporate intranet, which employees used to share content. They enabled SharePoint Publishing features, and thoroughly customized site branding. They had over 2000 sites in one site collection. They had created 23 web templates, 61 features, 17 content types, 34 Web Parts, 19 web controls, 21 page layouts, 13 timer jobs, and had customized SharePoint Central Administration. As well, they used hybrid self-service site collection to enable users to request their own sites.



**Figure 1. Contoso’s SharePoint infrastructure landscape**

As part of their pre-migration assessment, they mapped the landscape of their company-wide SharePoint 2010 architecture, which served as a reference for decisions to come. Their workflow followed these steps:

1. Customer requirements are formalized as tasks for the developers.

2. Development environments are virtualized and run SharePoint. Source code is stored in the [Team Foundation Server (TFS)](http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx).

3. External developers use VPN to access dedicated virtual machines and the TFS.

4. The daily build server is used to run automated integration testing.

5. A build verification farm with three servers is used for initial functionality testing. It's somewhat more permanent than the daily build environment. All customizations use it before they're moved to a pre-production or quality assurance environment.

6. Intranet quality assurance or pre-production that's used for user acceptance testing.

7. On-premises intranet production environment hosts some of the SharePoint services.

8. Pre-production environment in Office365 Dedicated, managed by Microsoft

9. Production environment in Office365 Dedicated, managed by Microsoft

10. External access is managed using [ForeFront Unified Access Gateway (UAG)](http://technet.microsoft.com/en-us/library/ff358694.aspx) for selective extranet services.

11. Internet environment consists from two different farms which change the roles between tem as part of the release cycle from production to pre-production and vice versa. Done to reach high availability requirements for the .com sites.

12. Multiple extranet farms that are used mainly for collaboration.

They weren’t happy with the fact that it took them six weeks to update something using their FTC-based release model in SharePoint 2010, which inspired them to change their approach. As a forward-looking company, they embraced cloud-readiness, valued business agility and cost-effective solutions, and were looking for a customizable alternative to SharePoint 2010 that would help them reduce long-term costs.

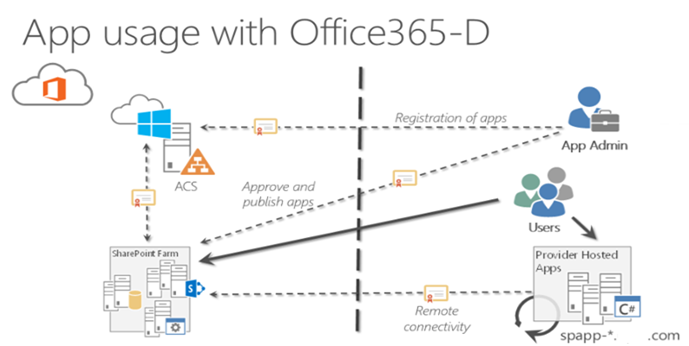
They recognized that changing their approach would first require some planning. They compared their FTC solutions to the capabilities offered by the SharePoint app model, and quickly realized that there is rarely a one-to-one correspondence between their SharePoint 2010 code componentization strategies or their .NET server APIs, and the componentization approach they’d need to adopt to meet their business needs with apps for SharePoint and the CSOM API.

Rather than starting their assessment by looking specifically at technology factors, they focused first on their business requirements. They reviewed and recorded business requirements that considered resources, goals, viewpoints, skills, and constraints. They realized that one of the main sources of pain from their SharePoint 2010 implementation was complexity, so they decided to prioritize simplicity.

Contoso chose to take a long-term view, and assessed the long-term implications for their business of making specific tradeoffs. When the time came to assess specific technical details, they realized that the initial release of the SharePoint app model could meet most needs. By assessing gaps and migrating the pieces that were ready first, they could gradually manage change.

First, they upgraded SharePoint 2010 MySites to SharePoint 2013 OneDrive for Business in December 2013. Around that same time, they created new SharePoint 2013 team sites, which would be the targets for SharePoint 2010 to SharePoint 2013 site migration. They upgraded their intranet sites in March 2014, and completed their Team sites migration in July 2014 (expected).

Contoso took a decisively simpler approach to their SharePoint implementation. In the new environment, users interact with apps for SharePoint, which communicate remotely with the SharePoint farm. System administrators made these apps available to users by registering them in [Access Control Services](http://azure.microsoft.com/en-us/documentation/articles/active-directory-dotnet-how-to-use-access-control/), which communicates with Contoso’s SharePoint farm. Administrators retain the sole rights to approve and publish apps for SharePoint to the farm. Figure 2 represents this system.

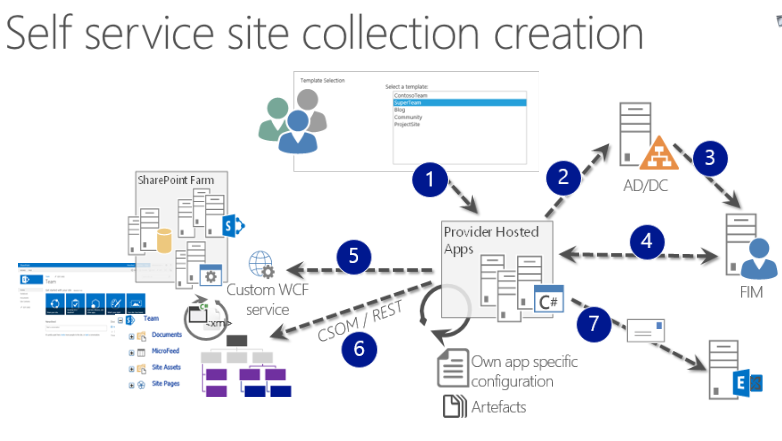


**Figure 2. App usage with Office 365 Dedicated**

Contoso grants users the ability to request site creation through a [custom self-service site creation user interface](http://code.msdn.microsoft.com/SharePoint-2013-Use-an-app-f457b229) (code sample). As described in Modules 3 through 5 and demonstrated in associated samples in the [sample pack](http://code.msdn.microsoft.com/SharePoint-2013-Sample-81b03d1e), apps for SharePoint are central to the new implementation. The provider-hosted apps for SharePoint use Active Directory lookups and [ForeFront Identity Manager](http://technet.microsoft.com/en-us/forefront/cc470030.aspx) to manage identities, and uses CSOM and REST calls to a custom Windows Communication Foundation (WCF) web service to serve and get app-specific configuration information and artifacts.

[Choose patterns for developing and hosting your app for SharePoint](http://msdn.microsoft.com/en-us/library/office/fp179887(v=office.15).aspx)

[How to: Create a basic provider-hosted app for SharePoint](http://msdn.microsoft.com/en-us/library/office/fp142381(v=office.15).aspx)



**Figure 3. Self-service site collection creation**

1. Self-service site collection creation functionality is hosted on the SharePoint provider-hosted apps platform.

2. Contoso wanted to reuse some aspects of their old model (Figure 1), so the SharePoint provider-hosted app is creating by default three [Active Directory groups](http://technet.microsoft.com/en-us/library/cc739393%28v=WS.10%29.aspx) as part of the request process in asynchronous way for the requested sites.

3. New Active Directory groups are synchronized to [ForeFront Identity Manager](http://www.microsoft.com/en-us/server-cloud/products/forefront-identity-manager/default.aspx#fbid=haBDZoiL5YQ), which enables administrators to edit the Active Directory groups.

4. Once Active Directory groups have been created, site provisioning can continue in an asynchronous manner.

5. Using a custom Windows Communication Foundation (WCF) extension, site collections are created in Office 365. This customization was implemented before Office 365 natively supported creating site collections using CSOM in the Office365 Dedicated environment.

6. The SharePoint provider-hosted app uses normal CSOM and REST capabilities to apply required branding and other settings to the newly created site collection. By adding native support for site collection creation, the required changes only require a limited set of code.

7. Once site creation is successful, email is sent to requestors informing them that their provisioned site collection is available to use.

Contoso provisioned their Team sites and OneDrive for Business sites with clean and minimally customized branding—Microsoft recommends minimal branding for Team sites and OneDrive for Business sites. For example, Contoso’s intranet home page provides a fully personalized experience for users based on their organization and location, powered by JavaScript. The design is based on SharePoint page layouts.

[How to: Create a basic SharePoint provider-hosted app for SharePoint](http://msdn.microsoft.com/en-us/library/office/fp142381%28v=office.15%29.aspx)

[How to: Create a page layout in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj822368%28v=office.15%29.aspx)

# Migration techniques and considerations

Because existing sites depend on full-trust code, the migration was challenging. A key to gradual migration was first to reduce the dependency on page layouts and master pages by systematically controlling and reducing the elements used on sites. The following techniques for replacing FTC code with apps for SharePoint functionality powered the gradual upgrade.

## Gradual page layout and master page replacement

System administrators wrote PowerShell scripts that enabled remote operations that managed branding elements, configured Web Parts and replaced them on pages, app part instances, and other elements. These scripts are controlled in the UI with provider-hosted apps for SharePoint written in CSOM to update content on the SharePoint farm.

## Gradual Web Part and web control replacement

Contoso ran remote operations to replace Web Parts on pages. First, they replaced custom Web Parts developed for their SharePoint 2010 installation with out-of-the-box Web Pages available in SharePoint 2013. This step reset the baseline so that Contoso could consider Web Part customizations with their refreshed understanding of their business requirements. Next, they replaced default out-of-the-box Web Parts with instances of app parts.

Server-side code from web controls was replaced with JavaScript on either the master page or on the page layout.

[Migrating a Web Part to an App Part](http://msdn.microsoft.com/en-us/office/dn448480.aspx) (Yina Arenas)

## Sideloading: Using app parts to provision the Web Parts on SharePoint sites

As you’re considering deploying app parts that duplicate or approximate existing SharePoint 2013 site functionality, you can upload individual app parts to your app catalog or sideload app parts across all of the site collections on your SharePoint farm. To “sideload” an app part means to deploy it without reading from or writing to the app catalog.

You can upload individual apps to your app catalog (which can contain one or more app parts), sideload app parts across all of the site collections on your SharePoint farm, or both, at the same time or at different times.

### Why sideload an app for SharePoint?

From a development perspective, sideloading enables development against site collections that use templates that are not compatible with the developer site (i.e., Search template and Project template).

When you enable sideloading, anyone with permissions to install an app can do so, even if that app is not sourced from your enterprise’s app catalog or from the Office Store. If [you have a Professional or Enterprise subscription](http://technet.microsoft.com/en-us/library/sharepoint-online-service-description.aspx), use the developer site template to create a new site collection for testing apps. Alternatively, you can use app sideloading to load and test apps.

### Enabling and disabling sideloading

You must have tenant admin permissions in a multi-tenant (MT) environment and farm admin permissions in a single tenant environment to enable sideloading. You will receive an **Access Denied** error if you do not have tenant admin permissions.

**Note** If you develop an app for SharePoint in Visual Studio against any of the following site URLs without the required permissions, Visual Studio will throw the “Sideloading of apps is not enabled on this site” error and the app installation will fail.

**To enable sideloading (PowerShell)**

1. Install the [SharePoint Online management shell](http://www.microsoft.com/en-us/download/details.aspx?id=35588).
2. Save the following code to your machine as “EnableSideLoading.ps1”.

add-type -Path 'C:\Program Files\SharePoint Online Management Shell\Microsoft.Online.SharePoint.PowerShell\Microsoft.SharePoint.Client.dll'

Write-Host 'To enable SharePoint app sideloading, enter the Site Url, user name, and password'

$siteurl = Read-Host 'Site Url'

$username = Read-Host "User Name"

$password = Read-Host -AsSecureString 'Password'

try

{

[Microsoft.SharePoint.Client.ClientContext]$cc = New-Object Microsoft.SharePoint.Client.ClientContext($siteurl)

[Microsoft.SharePoint.Client.SharePointOnlineCredentials]$spocreds = New-Object Microsoft.SharePoint.Client.SharePointOnlineCredentials($username, $password)

$cc.Credentials = $spocreds

$web = $cc.Web

$cc.Load($web);

$cc.ExecuteQuery();

$site = $cc.Site;

$developerguid = new-object System.Guid "AE3A1339-61F5-4f8f-81A7-ABD2DA956A7D"

$site.Features.Add($developerguid, $true, [Microsoft.SharePoint.Client.FeatureDefinitionScope]::Farm);

$cc.ExecuteQuery();

Write-Host -ForegroundColor Green 'Sideloading enabled on site' + $siteurl

}

catch

{

Write-Host -ForegroundColor Red 'Error encountered when trying to enable sideloading' $siteurl, ':' $Error[0].ToString();

}

1. Run PowerShell as an administrator and execute **EnableSideLoading.ps1**.
2. Provide the URL for the site collection where sideloading should be enabled.
3. Enter your user credentials. You must be a Tenant Admin to activate this feature.
4. Once the script has run successfully, install apps from Visual Studio.

**To disable sideloading (PowerShell)**

1. Save the following code as “DisableSideLoading.ps1”.

add-type -Path 'C:\Program Files\SharePoint Online Management Shell\Microsoft.Online.SharePoint.PowerShell\Microsoft.SharePoint.Client.dll'

Write-Host 'To disable app sideLoading, enter Site Url, username and password'

$siteurl = Read-Host 'Site Url'

$username = Read-Host "User Name"

$password = Read-Host -AsSecureString 'Password'

try

{

[Microsoft.SharePoint.Client.ClientContext]$cc = New-Object Microsoft.SharePoint.Client.ClientContext($siteurl)

[Microsoft.SharePoint.Client.SharePointOnlineCredentials]$spocreds = New-Object Microsoft.SharePoint.Client.SharePointOnlineCredentials($username, $password)

$cc.Credentials = $spocreds

$web = $cc.Web

$cc.Load($web);

$cc.ExecuteQuery();

$site = $cc.Site;

$developerguid = new-object System.Guid "AE3A1339-61F5-4f8f-81A7-ABD2DA956A7D"

$site.Features.Remove($developerguid, $true);

$cc.ExecuteQuery();

Write-Host -ForegroundColor Green 'Sideloading disabled on site' + $siteurl

}

catch

{

Write-Host -ForegroundColor Red 'Error encountered when trying to disable sideloading' $siteurl, ':' $Error[0].ToString();

}

2. Run Powershell as an administrator and execute **DisableSideLoading.ps1**.

3. Provide the URL for the site collection where sideloading should be disabled.

4. Enter your user credentials. To deactivate this feature, you must be a Tenant Admin.

### The perils of sideloading

Sideloading is a feature designed for developers and is not enabled by default. Faulty apps pose risks to their host web or host site collection: administrators should take appropriate precautions before sideloading apps. Sideloading poses the following risks:

* Sideloading explicitly bypasses SharePoint’s standard governance controls. Site Collection Administrators can uninstall the app.
* The user experience of sideloaded apps isn’t the same as apps that are published to the app catalog. For example, sideloaded apps have no metadata, and they must be updated manually.
* If you sideload with a process that has an app-only context, the app will not install properly. However, sideloading with a process that has a user context works as expected.

Disable sideloading immediately after the app in installed. The sideloading script introduces the possibility for malicious attack from a user with tenant admin rights as long as the sideloading process exists in the window where the script is running. Once the process closes the window, the threat is gone.

**Note** You can sideload provider-hosted apps once app principals have been registered. Sideloading SharePoint-hosted apps is not recommended.

The **Trust It** button is based on permissions that are defined in [appmanifest.xml](http://msdn.microsoft.com/en-us/library/office/fp179918%28v=office.15%29.aspxhttp:/msdn.microsoft.com/en-us/library/office/fp179918(v=office.15).aspx). Rather than modify permissions in the appmanifest.xml file, developers should update the client ID and other settings in appmanifest.xml to be sure they match the sideloading script.

### Example sideloading code (C#)

The following code example demonstrates how to enable or disable sideloading and install the application to a SharePoint site collection. You can use the following example regardless of whether the app is installed. The example is based on the assumption that the provider-hosted app that has its client secret registered will be provisioned to a site collection.

Guid \_sideloadingFeature = new Guid("AE3A1339-61F5-4f8f-81A7-ABD2DA956A7D");

string \_url = GetUserInput("Please Supply the SharePoint Online Site Collection URL: ");

/\* Prompt for Credentials \*/

Console.WriteLine("Enter Credentials for {0}", \_url);

string \_userName = GetUserInput("SharePoint Username: ");

SecureString \_pwd = GetPassword();

ClientContext \_ctx = new ClientContext(\_url);

\_ctx.ApplicationName = "AMS SIDELOADING SAMPLE";

\_ctx.AuthenticationMode = ClientAuthenticationMode.Default;

//For SharePoint Online

\_ctx.Credentials = new SharePointOnlineCredentials(\_userName, \_pwd);

string \_path = GetUserInput("Please supply path to your app package:");

Site \_site = \_ctx.Site;

Web \_web = \_ctx.Web;

try

{

\_ctx.Load(\_web);

\_ctx.ExecuteQuery();

//Make sure sideloading is enabled. Sideloading requires tenant admin

//permissions to activate, or SharePoint throws an

//exception. The ProcessFeature is an extension method; see OfficeAMS

//(https://officeams.codeplex.com/) OfficeAMS.Core sample for details.

\_site.ProcessFeature(\_sideloadingFeature, true);

try

{

var \_appstream = System.IO.File.OpenRead(\_path);

AppInstance \_app = \_web.LoadAndInstallApp(\_appstream);

\_ctx.Load(\_app);

\_ctx.ExecuteQuery();

}

catch

{

throw;

}

//Ensure that the sideloading feature is disabled when sideloading completes or if SharePoint throws an exception.

\_site.ProcessFeature(\_sideloadingFeature, false);

}

catch (Exception \_ex)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(string.Format("Exception!"), \_ex.ToString());

Console.WriteLine("Press any key to continue.");

Console.Read();

}

### Application lifecycle management

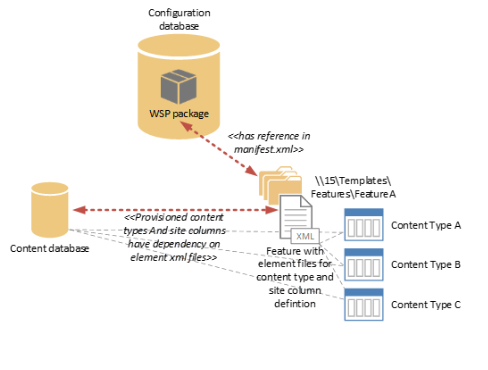
Automating sideloading using a script can help reduce the impact on site governance. Sideloading does not suppress the security check or compensate existing security requirements. It does enable the programmatic installation of an app, which is trusted once it is registered. Once the app is installed using sideloading, the fact that it is trusted is not detectible to users .

### Troubleshooting

Sometimes, Visual Studio will return a “Sideloading of apps is not enabled . . . “ error. This happens because Visual Studio calls the [IsAppSideloadingEnabled method of the AppCatalog class](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.appcatalog.isappsideloadingenabled%28v=office.15%29.aspx) for the site collection URL () parameter defined on the property. **IsAppSideloadingEnabled** returns **True** if the site is a developer site or if the sideloading feature is enabled. Sideloading won’t work for a site collection on a site that is not a developer site unless the sideloading feature is enabled.

## Migrating content types and site columns

Many SharePoint 2010 developers used the feature framework to create declaratively create site columns and content types, while others created them using FTC.



**Figure 4. Site column dependencies on the feature framework**

The SharePoint configuration database stores WSP [solution](http://msdn.microsoft.com/en-us/library/office/aa543214(v=office.14).aspx) files that each reference the manifest.xml file, which is part of the [solution schema](http://msdn.microsoft.com/en-us/library/office/ms442108%28v=office.15%29.aspx). The content types and site columns (fields) that are provisioned to the site depend on element.xml files [(feature.xml](http://msdn.microsoft.com/en-us/library/office/ms475601%28v=office.15%29.aspx)).

If you created site columns and content types using server-side code, there is no dependency on the feature framework or farm solutions. Other SharePoint features, such as page layouts, depend on the [Content Type ID](http://msdn.microsoft.com/en-us/library/office/aa543822%28v=office.14%29.aspx), which SharePoint uses to recursively and uniquely define [content types](http://msdn.microsoft.com/en-us/library/office/ms472236%28v=office.14%29.aspx).

### Known issue

Site administrators have the ability to retract solutions from targeted site collections to which they had previously been installed. During the installation process, SharePoint creates a directory structure on the web front-end that includes XML files that describe various features. When feature framework elements are installed using solution packages, SharePoint creates new folder structures with element.xml files in the 15 hives. By default, the content database has a dependency on these files in the 15 hive, which is the root issue. This works in most cases.

However, once a solution creates a content type or site column, it cannot be deleted even after it has been retracted. When the solution is retracted, the front-end XML files are deleted, but the rows in the database remain.

Re-activating the solution restores the XML files that define the site columns, which makes the site columns usable again.

### Analyzing migration needs for site columns and content types

At first glance, site columns and content types may seem like a very difficult feature subset to assess for migration readiness. If you’ve created few or no custom content types and site columns, and used a centralized and organized approach to create custom content types, then migrating these elements will be easier. However, if content type and site column development were less targeted or business requirements have changed significantly since you created them, pre-migration analysis may be more involved.

When assessing custom content types and site columns, consider whether:

* The content type or site column available out-of-the-box can do what your custom SharePoint 2010 code does.
* Web applications subscribe to custom types available in the content type hub.
* You’re using custom content types and site columns to define [managed metadata in SharePoint 2010](http://msdn.microsoft.com/en-us/library/office/ee559337%28v=office.14%29.aspx).
* Custom site columns associate with more than one custom or out-of-the-box content type.
* The relationships that multiple content types have with multiple libraries and each other.
* There’s a different way to meet your business need than using custom content types.
* A required page layout depends on the custom content type.

If there’s a way to simplify custom content types, site columns, and their interrelationships, do that before migrating.

### Creating a custom ContentTypeId in CSOM

The functionality for creating a content type ID in CSOM is available in the [SharePoint 2013 Service Pack 1 Client Components SDK](http://www.microsoft.com/en-us/download/details.aspx?id=35585). You can [create content types with specific IDs using CSOM (Vesa Juvonen)](http://blogs.msdn.com/b/vesku/archive/2014/02/28/ftc-to-cam-create-content-types-with-specific-ids-using-csom.aspx), which is required to ensure that you have consistent content types across the enterprise. Now that it’s possible without full-trust code, you can create content types using the app model.

// Open a connection to the Office 365 tenant.

ClientContext cc = new ClientContext(siteUrl);

cc.AuthenticationMode = ClientAuthenticationMode.Default;

cc.Credentials = new SharePointOnlineCredentials(userName, pwd);

// Load reference to content type collection.

Web web = cc.Web;

ContentTypeCollection contentTypes = web.ContentTypes;

cc.Load(contentTypes);

cc.ExecuteQuery();

// Create a ContentTypeCreationInformation object

ContentTypeCreationInformation newCt = new ContentTypeCreationInformation();

// Set the name for the content type

newCt.Name = "Contoso Document";

//Inherit from an out-of-the-box document - 0x0101 and assign.

newCt.Id = "0x0101009189AB5D3D2647B580F011DA2F356FB2";

// Set the content type to be available from a specific group.

newCt.Group = "Contoso Content Types";

// Create the content type.

ContentType myContentType = contentTypes.Add(newCt);

cc.ExecuteQuery();

**Note** When you assign the **Id** property directly, you cannot set the **ParentContentType** property for the **ContentTypeCreationInformation** object. If you do, SharePoint will throw an exception.

### Creating content types in the app model

You can use CSOM to create site columns and content types, and then add site columns to the content type. You can also localize site columns and content types using CSOM APIs introduced in the [SharePoint 2013 Client Components SDK for Service Pack 1 (SP1)](http://www.microsoft.com/en-us/download/details.aspx?id=35585). This package includes an updated version of the Microsoft.SharePoint.Client.dll, and for SharePoint on-premises and pre-convergence, its file set version is 15.0.4569.1000. For MT and ST, the file version is 16.0.4569.1000.

You can create a new content type in CSOM with the [ContentTypeCreationInformation](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.contenttypecreationinformation(v=office.15).aspx) class. The following code shows you how.

**Note** The ability to set ContentTypeId was newly introduced in SharePoint 2013 Service Pack 1.

ContentTypeCollection contentTypes = web.ContentTypes;

cc.Load(contentTypes);

cc.ExecuteQuery();

foreach (var item in contentTypes)

{

if (item.StringId == "0x0101009189AB5D3D2647B580F011DA2F356FB2")

return;

}

// Create a ContentTypeInformation object.

ContentTypeCreationInformation newCt = new ContentTypeCreationInformation();

// Set the name for the content type.

newCt.Name = "Contoso Document";

//Inherit from the out-of-the-boxdocument - 0x0101 and assign.

newCt.Id = "0x0101009189AB5D3D2647B580F011DA2F356FB2";

// Set content type to be available from a specific group.

newCt.Group = "Contoso Content Types";

// Create the content type.

ContentType myContentType = contentTypes.Add(newCt);

cc.ExecuteQuery();

// Use AddFieldAsXml to add fields to the FieldCollection of a site collection.

FieldCollection fields = web.Fields;

cc.Load(fields);

cc.ExecuteQuery();

string FieldAsXML = @"<Field ID='{4F34B2ED-9CFF-4900-B091-4C0033F89944}' Name='ContosoString' DisplayName='Contoso String' Type='Text' Hidden='False' Group='Contoso Site Columns' Description='Contoso Text Field' />";

Field fld = fields.AddFieldAsXml(FieldAsXML, true, AddFieldOptions.DefaultValue);

cc.Load(fields);

cc.Load(fld);

cc.ExecuteQuery();

// Use the FieldLinkCollection and FieldLinkCreationInformation classes to link the fields to the content type.

FieldCollection fields = web.Fields;

Field fld = fields.GetByInternalNameOrTitle("ContosoString");

cc.Load(fields);

cc.Load(fld);

cc.ExecuteQuery();

FieldLinkCollection refFields = myContentType.FieldLinks;

cc.Load(refFields);

cc.ExecuteQuery();

foreach (var item in refFields)

{

if (item.Name == "ContosoString")

return;

}

// Ref does nt

FieldLinkCreationInformation link = new FieldLinkCreationInformation();

link.Field = fld;

myContentType.FieldLinks.Add(link);

myContentType.Update(true);

cc.ExecuteQuery();

# Localizing site titles, descriptions, and content types with CSOM

If your SharePoint installation is localized, you can use CSOM to convert title and description values for content types and site columns.

web.TitleResource.SetValueForUICulture("fi-FI", "Kielikäännä minut");

web.DescriptionResource.SetValueForUICulture("fi-FI", "Kielikäännetty saitti");

You can use this same approach from a list:

list.TitleResource.SetValueForUICulture("fi-FI", "Kielikäännä minut");

list.DescriptionResource.SetValueForUICulture("fi-FI", "Tämä esimerkki näyttää miten voit kielikääntää listoja.");

You can localize the name and description for content types. For fields, you can localize the title and description values.

myContentType.NameResource.SetValueForUICulture("fi-FI", "Contoso Dokumentti");

myContentType.DescriptionResource.SetValueForUICulture("fi-FI", "Tämä on geneerinen Contoso dokumentti.");

fld.TitleResource.SetValueForUICulture("fi-FI", "Contoso Teksti");

fld.DescriptionResource.SetValueForUICulture("fi-FI", "Tää on niiku Contoso metadatalle.");

# Timer jobs

You can use the SharePoint CSOM to create simple timer jobs. However, note that there is no equivalent replacement for timer jobs in the SharePoint app model. You can use any outside platform, such as Windows Azure, to provide timer job functionality.

Old SharePoint timer jobs are simply scheduled executions of code. In the app model, we can use other ways to schedule the code, and then write custom CSOM code or create other remote access points to access sites for required operations. This is described in more detail in the Module 5 document.

### Timer jobs

You can use CSOM to create a simple timer job.

using Microsoft.SharePoint.Client;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.Configuration;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Core.SimpleTimerJob.Console

{

class Program

{

private static string SharePointPrincipal = "00000003-0000-0ff1-ce00-000000000000";

/// <param name="args"></param>

static void Main(string[] args)

{

Uri siteUri = new Uri(ConfigurationManager.AppSettings["url"]);

//Get the realm for the URL

string realm = TokenHelper.GetRealmFromTargetUrl(siteUri);

// Get the access token for the URL.

// Requires this app to be registered with the tenant.

string accessToken = TokenHelper.GetAppOnlyAccessToken(

TokenHelper.SharePointPrincipal,

siteUri.Authority, realm).AccessToken;

//Get client context with access token.

using (var ctx =

TokenHelper.GetClientContextWithAccessToken(

siteUri.ToString(), accessToken))

{

// Let's create a list to the host web and add a new entry for each execution.

if (!ListExists(ctx.Web, "RemoteOperation"))

{

AddList(ctx.Web, ListTemplateType.GenericList, "RemoteOperation");

}

// Add new execution entry to the list time stamp.

// Assume that the web has a list named "Announcements".

List list = ctx.Web.Lists.GetByTitle("RemoteOperation");

ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();

ListItem newItem = list.AddItem(itemCreateInfo);

newItem["Title"] = string.Format("New {0}", DateTime.Now.ToLongTimeString());

newItem.Update();

ctx.ExecuteQuery();

}

}

/// Add a list to a site.

/// <param name="web">The site to be processed can be root web or sub site.</param>

/// <param name="listType">Type of the list</param>

/// <param name="listName">Name of the list</param>

/// <param name="enableVersioning">Enable versioning on the list.</param>

/// <param name="updateAndExecuteQuery">Perform list update and execute the query. Defaults to true. </param>

public static void AddList(Web web, ListTemplateType listType, string listName)

{

ListCollection listCol = web.Lists;

ListCreationInformation lci = new ListCreationInformation();

lci.Title = listName;

lci.TemplateType = (int)listType;

List newList = listCol.Add(lci);

}

/// Checks if list exists on the particular site based on the list Title property.

/// <param name="web">Site to be processed - can be root web or sub site</param>

/// <param name="listTitle">Title of the list to be checked.</param>

public static bool ListExists(Web web, string listTitle)

{

ListCollection lists = web.Lists;

IEnumerable<List> results = web.Context.LoadQuery<List>(lists.Where(list => list.Title == listTitle));

web.Context.ExecuteQuery();

List existingList = results.FirstOrDefault();

if (existingList != null)

{

return true;

}

return false;

}

}

}

# Situations that require a full migration

The presence of some customizations in FTC require a full migration. These include:

* Custom field types
* Custom site definitions
* Moving from SharePoint publishing sites to SharePoint non-publishing sites (Team sites, OneDrive for Business sites)

All of these scenarios require data cleanup. First, export data from your baseline installation, clean up the data, and then import it into the new environment.

## Migrating custom field types

Custom field types can be created declaratively or with FTC. As with some other SharePoint components built declaratively that depend on the feature framework, there is not a simple way to migrate the functionality to the app model.

You can use **JSLink** to display custom field types on the client. This approach is described in detail in the MSDN Magazine article, [Using JSLink with SharePoint 2013](http://msdn.microsoft.com/en-us/magazine/dn745867.aspx).

[Custom field types](http://msdn.microsoft.com/en-us/library/office/ms446361%28v=office.14%29.aspx) (SharePoint 2010)

[How to: Create a custom field type](http://msdn.microsoft.com/en-us/library/office/bb862248%28v=office.14%29.aspx) (SharePoint 2010)

[How to: Create a custom field type definition](http://msdn.microsoft.com/en-us/library/office/ms415141%28v=office.14%29.aspx) (SharePoint 2010)

[Walkthrough: Creating a custom field type](http://msdn.microsoft.com/en-us/library/office/bb861799%28v=office.14%29.aspx) (SharePoint 2010)

[Code sample: Custom field](http://msdn.microsoft.com/en-us/library/office/ff407242%28v=office.14%29.aspx) (SharePoint 2010)

[Custom field validation](http://msdn.microsoft.com/en-us/library/office/ms434697%28v=office.14%29.aspx) (SharePoint 2010)

## Migrating custom site definitions

Site definitions are cached on the server the first time they are accessed at runtime. When site pages are customized (excluding Web Parts and other customizations based in web browsers), these pages can be stored in the content database. Even though modules (.aspx files, etc.) could be stored in the content database, the site has an identifier that makes it to be created based on the custom site definition. If the site definition is retracted from the farm, the site breaks.

To migrate sites that are created using custom site definitions, create a new site with the out-of-the-box site definition and move your content to it. This is the only way to move the content from sites created using custom site definitions to a new site.

Custom [site definitions and configurations](http://msdn.microsoft.com/en-us/library/office/aa978512%28v=office.14%29.aspx) (SharePoint 2010) cannot be created as apps for SharePoint.

**Note** Custom site definitions are only supported on-premises.

# Governance considerations

You can use apps for SharePoint to manage many site governance considerations for your enterprise. T Some example governance considerations include site stamping, managing the application of variable branding elements to sites and subwebs in the enterprise, and tracking compliance steps and status.

## App stapling and tenant-scoped apps

Like “feature stapling,” which enabled developers to add functionality to specific types of sites in SharePoint, “app stapling” is a way that developers can affix functionality to SharePoint sites using the SharePoint app model. “App stapling” is a way that developers can deploy functionality to a group of existing or new sites that require specific functionality.

### Stapling an app to a SharePoint site

To staple an app to sites, first deploy it through your enterprise’s app catalog. Next, *install* the app to the app catalog, which enables you to deploy it to specific sites in the tenancy.

After you have installed the app, deploy it by selecting Site Content from the App Catalog, then clicking the ellipse (. . . ) and Deployment. On the Manage App Deployments screen, you can add, remove, or retract access to a site. When using app stapling, however, you cannot deploy custom actions and deploy app parts to the host web when the app is deployed to the sites.

The administrator can retract stapled apps from specific sites, or disable them. Site owners can’t remove the app from a site that meets deployment criteria because it was deployed and installed by the administrator.

**Note** Remote events are fired only once—when the app is installed in the App Catalog.

[SharePoint 2013 app deployment through “app stapling”](http://blogs.msdn.com/b/richard_dizeregas_blog/archive/2013/03/04/sharepoint-2013-app-deployment-through-quot-app-stapling-quot.aspx)

[How to: Set up an app catalog on SharePoint](http://msdn.microsoft.com/en-us/library/office/fp123530%28v=office.15%29.aspx)

[How to: Set up an app catalog on SharePoint Online](http://msdn.microsoft.com/en-us/library/office/dn574752%28v=office.15%29.aspx)

[Configure an environment for apps for SharePoint](http://technet.microsoft.com/en-us/library/fp161236.aspx)

### Tenant-scoped apps for SharePoint

In SharePoint 2013, a tenancy is a set of site collections. A tenancy can have an app for SharePoint app catalog that’s specific to its scope.

Every app for SharePoint has an [app scope](http://msdn.microsoft.com/en-us/library/office/fp179896%28v=office.15%29.aspx)—which can be a web scope or a tenant scope. Whether an app has a web scope or a tenant scope is a byproduct of how the site administrator installed the app. When an administrator batch installs an app to a set of sites within the tenancy, the app is tenant scoped.

Depending on the functionality you want to migrate from FTC to the app model, you may decide that app stapling or tenant-scoped apps confer specific advantages, depending on the functionality in the source solution.

**Note** If you install a Store app to the app catalog site, you can also push that app by using the “app stapling” to all sites.

## Tracking compliance checks and status

You may want to write an app for SharePoint that monitors whether a user’s OneDrive for Business site complies with corporate IT standards. For example, at scheduled intervals, you may want to automatically generate and send an email to users who are out of compliance. You can lock users out of sites that are out of compliance.

In the case of a lockout, you may want to provision the ability to unlock by provisioning a new control to the page. Consider, for example, a control that escalates to an administrator to unlock a site.

# Conclusion

This module introduced a general approach and some specific tips for migration FTC solutions to the app model, with specific attention to site branding and site governance.