Microsoft SharePoint Online for Enterprises

Custom Solution Developer's Guide

Dedicated Plans

Author: Ryan Berg

Published: June 2013

Revised: February 2015

Contents

Introduction

This document provides Microsoft® SharePoint® Online dedicated plan customers with guidance about how to create custom solutions to be deployed to the SharePoint Online environment. It offers recommendations about how to develop custom solutions and describes the established best practices.

This guide consists of the following parts:

- The introduction supplies the context for creating a custom solution, and reviews new information about this guide and SharePoint Online generally.
- Part I summarizes the development process that the customer must follow to design, develop, and submit fully trusted code (FTC) to Microsoft for deployment to the SharePoint Online environment. Fully trusted code requires deployment or execution by Microsoft on the hosted servers based on Microsoft SharePoint 2013. This is typically Microsoft .NET–based code and dependent files, but can also include Windows PowerShell® scripts or batch files.
- Part II provides best practices to developers for SharePoint development and testing. This part also describes the services, features, and configurations that the SharePoint Online service supports.

Note

For supplemental information about partner access and deploying claims authentication solutions, see the SharePoint Online Partner Access SDK, which is available to current customers on the Customer Extranet site.

Most of the content in this guide addresses best practices for developing custom solutions for both Microsoft SharePoint 2010 and Microsoft SharePoint 2013 in the SharePoint Online environment. However, some material is for only one of those two versions. In these cases, the context will indicate which version is being addressed.

Although the process in Part I applies only to fully trust custom solutions that are deployed by Microsoft, the guidance and information in Part II is recommended to customers who are developing any type of customization for the SharePoint Online environment.

What Custom Solution Developers Should Know

Readers of this document should be familiar with professional SharePoint development, and have an understanding of the following technologies:

- Microsoft .NET Framework
- Microsoft ASP.NET

They must also understand how to create and test scalable SharePoint solutions, and how to plan solutions that are upgradeable. For resources to get started with SharePoint development, see Appendix A: SharePoint Developer Resources later in this document.

SharePoint 2013 introduces a new application development model that enables developers to build SharePoint applications that do not require full trust code to be run on the server. Starting with SharePoint 2013, the Cloud App Model is the preferred development model for SharePoint Online Dedicated.

Customers must ensure that custom solutions are tested for performance, scalability, security, and stability prior to submission to Microsoft. The SharePoint Online hosted service does not include a hosted development environment for the customers, so it is the responsibility of the customer or a designated

development team to set up and maintain a development and test environment. To assist the customer in creating an environment that mimics the production environment, Microsoft provides the SharePoint Online Build Guide (available to current customers only, on the Customer Extranet site).

Guidance Updates in This Release

Here are the most important changes to this guide and the custom solution development process:

- Guidance on using the new Cloud App Model introduced in SharePoint 2013
- Updates to existing extensible features in SharePoint 2013 platform

About MSOCAF

Microsoft provides the Microsoft SharePoint Online Code Analysis Framework (MSOCAF) to customers for use in analyzing full trust custom solutions, testing the deployment of the custom solutions, and submitting them for installation in the SharePoint Online environment. This tool is not required for apps that leverage the new Cloud App Model. Customers can obtain MSOCAF from the MSOCAF Download Site. MSOCAF Help provides guidance on using MSOCAF.

MSOCAF uses an extensible framework to run a set of executable rules against a custom solution prior to submitting the custom solution for approval and deployment into pre-production and production environments. For more in-depth information about MSOCAF, see the SharePoint Online Custom Solution Policies and Process document. Here's what developers need to know:

- MSOCAF must be used to validate all custom solutions developed for SharePoint Online by both customers and independent software vendors (ISVs). Validation includes testing of deployment and rollback in the customer's test environment.
- Custom solutions must use solution package (.wsp) files.
- For details about MSOCAF rules, see Appendix D: Rules Enforced by MSOCAF later in this document.
- In addition to validating with MSOCAF, you must conduct additional manual test cases, which are collected in the Custom Solution Test Cases document (available to current customers only, on the Customer Extranet site). You must run your custom solutions through these manual tests in their test environment prior to submission.

PART I: Full Trust Custom Solution Development Process

⚠ **NOTE:** *Information in this section does not apply to the SharePoint 2013 Cloud App Model.*

Introducing custom code in the SharePoint Online production environment—or revising existing code—requires your development team to follow the process in the SharePoint Online Custom Solution Policies and Process document. Here are the steps in that process:

1. Gather requirements.
2. Create high-level design (HLD) document.
3. Microsoft reviews HLD.
4. Develop custom solution.
5. Test, package, and validate with MSOCAF.
6. Microsoft reviews, validates, and deploys.

⚠ **Important**
You should not proceed with any custom solution development work before Microsoft approves the HLD document. If you proceed with coding the custom solution, but the design is rejected, it could lead to time lost in the overall development cycle and potentially a delay in deployment of the custom solution. If the custom solution plan includes third-party components that are to be purchased as part of the custom solution, we recommend that you purchase these components after the HLD is reviewed.

For information and resources about SharePoint development, see Appendix A: SharePoint Developer Resources later in this document.

Solution Files

Microsoft requires that customer-developed solutions use solution packages with a .wsp file extension to deploy custom solutions to the SharePoint Online environment. A custom solution sometimes requires numerous solution packages. These solution packages can be built within the Visual Studio® development system or manually.

Solution files with a .wsp file extension are in CAB-based format that contains a directory structure and definition files that are understood by the SharePoint deployment infrastructure. SharePoint 2013 imports the solution into the SharePoint store, and then deploys it to the farm. In addition to the directory and definition files, a solution file contains a set of SharePoint Features: logical units of functionality that are self-contained. All configuration files within the solution packages should be pre-populated with production environment values to avoid deployment errors.

If you design a solution that requires more than 10 .wsp files, you should reconsider its architecture. It is difficult to manage and deploy that many .wsp files within a single deployment window, and the solution risks rejection due to complexity.

If a solution developed by an ISV does not use solution packages, it can still be submitted to the SharePoint Online operations team for review. However, all custom solutions that are developed by an in-house customer development team must be packaged in .wsp files. For further details about solution packaging and files, see Building Block: Solutions in the MSDN Library.

Analysis of Third-Party Solutions

For ISV custom solutions that use Windows Installer (.msi) files, the .wsp files contained in the .msi file must be submitted separately from the .msi file. The .wsp can only be successfully analyzed if it is separate from the .msi file.

Required Directory Structure and Components in the Deployment Package

The required directory structure for the custom solution deployment package consists of a root directory named for the solution being submitted, and the following subdirectories:

- \Solutions artifacts
- \Release documents
- \Source code
- \Installation scripts
- \Test documents

When building the deployment package, also refer to Appendix C: Custom Solution Deployment Package Checklist.

Table 1. Custom Solution Directory Structure and Package Components

Root Directory and Subdirectories	Package Components	Description
Root Directory		Contains the subdirectories that contain the components. Name this directory for the solution being submitted.
\Solutions artifacts		Contains the solution packages, including final tested code. This folder must contain at least one .wsp file. MSOCAF uses this directory to create a CAB file, which is placed in the root directory. (For ISV custom solutions that use .msi files, the .wsp files contained in the .msi file must be submitted separately from the .msi file.)
	Solution packages	Include the final versions of all the solution packages, including final tested code. The MSOCAF report is automatically added to this folder by MSOCAF.
	Deployment manifest	You must create an XML manifest file to utilize the Test Deployment and Rollback features within MSOCAF for deployment to your test environment. This file must have the file name DeploymentManifest.xml.
	Public debug symbols	To aid in the possible troubleshooting scenarios, include the public debug symbols for all custom code.

Root Directory and Subdirectories	Package Components	Description
		This symbol file is automatically created by Visual Studio as long as either **pdb only** or **full** is selected from the **Debug Info** setting, which you can access via the Advanced settings on the Build page in Visual Studio.
\Release documents		Contains all the relevant documents related to the release. This folder must contain at least one .doc or .docx file.
	Deployment guide	Create a comprehensive set of instructions for deploying the custom solution, following the most current deployment guide template provided by Microsoft. The guide can include third-party licensing information, monitoring information, and scheduling information.
	Third-party licensing document	If using a third-party component or solution, provide the licensing details and any necessary keys or certificates for installation. Note: Remember that licenses are required for the pre-production, primary, and secondary environments.
	Scheduling Information	State the expected date for code-drop submission and deployment. Also include the expected duration of downtimes for deployment configuration and post-deployment configuration. (Include this information in the deployment guide.)
	Dependencies list	Submit a list of all dependencies for the code. This can include accounts and passwords, web services, databases, other solutions, features, updates, tool sets, and libraries.
	Rollback plan	Provide a rollback plan for any components that will be deployed manually. Include screen shots of how the environment must look after a successful rollback.
	Support troubleshooting document	Submit end-user troubleshooting guidance to Microsoft with one or two test accounts, if appropriate, to diagnose problems that are related to your dependencies. If a user account is provided, it should be set up with low privilege and used for no other purpose.

Root Directory and Subdirectories	Package Components	Description
	Monitoring troubleshooting document	Provide all details related to monitoring the custom solution, including logging details and documentation about all instrumentation. This list commonly takes the form of a table or error codes, the corresponding severity, and root cause. Include in the Monitoring section of the deployment guide and include additional Monitoring Troubleshooting documents as needed.
\Source code		Contains the source code.
	Source code	Include the project files and entire source code (validated through MSOCAF and manual test cases), if developed in house. We treat all source code with the utmost confidentiality. For details about the treatment of customer source code, see the "Pre-Existing Work" section of your organization's contract. Note: If the solution is a third-party, off-the-shelf application, no source code is required for submission.
\Installation scripts		Contains any installation or uninstallation scripts that must be run as part of the deployment. The installation scripts should refer to the appropriate file from the \Solutions artifacts folder for any defined deployment packages. For example: Stsadm.exe -o addsolution -filename ..\Solutions artifacts\xyz.wsp
\Test documents		Contains all of the test documentation.
	Test documents and results	For testing and validation purposes, it is extremely important to supply a copy of the manual test cases, test plan, test scenarios, unit test results, and all performance and scale testing that has been performed related to the code (and that is not covered by MSOCAF). This is even more important for all code that has a dependency (web service, data, or system) that cannot be tested during validation by Microsoft.

Root Directory and Subdirectories	Package Components	Description
		Include details about all elements of the custom solution that your organization wants deployed, including any third-party components and other solutions that the organization has purchased.

Updates to an Existing Custom Solution

When updating an existing custom solution, the existing documentation must also be updated. If the design has changed and now requires a feature enhancement or other change to functionality, you must first submit a new HLD document. If existing custom code is being updated *only to address bugs*, with no fundamental design changes, no HLD document is necessary and only the documentation in the deployment package must be updated.

In the case of an update, provide the following in addition to the components detailed in Table 1:

- **A summary of changes.** This must summarize what has changed from the previous version (for example, "Master page layout changed, referenced a new CSS file, and updated the underlying JavaScript.").
- **The original solution source code.** As a part of solution analysis, the new source code is validated against this original source code to ensure that no new changes or code issues were introduced other than the planned change.
- **Anticipated problems during upgrade.** Submit details about any errors, rendering issues, or other problems that might be expected when upgrading the custom solution.
- **Updated support documentation.** Include updates for the end-user troubleshooting guide and other documents. Describe anything that has changed and any new features that could potentially require support or monitoring.

Depending on which deployment action you choose for updating a custom solution, you must include documentation for the required steps:

- **Upgrade steps.** The upgrade must include the necessary steps for upgrading the solution. We recommend the upgrade method instead of the reinstall method, because only the upgraded components must be submitted for review, rather than a full deployment package.
- **Rollback steps.** The rollback must include the necessary scripts for restoring the previous version of the solution. If the custom solution stores any configuration data that is not part of the custom solution, the steps must include information on how to back up that information.
- **Reinstall steps.** The reinstall requires that you submit a completely new custom solution deployment package for review, and must include the necessary scripts for uninstalling the solution and reinstalling the new version.

If the solution is causing issues in the SharePoint Online environment or the customer requests a rollback, Microsoft follows these rollback steps to restore the previous version.

Note

Site collection features and site-scoped features cannot be deactivated by the rollback process.

1. Deactivate the features that are scoped for the web application and the farm.
2. Retract and delete the solution.

Note

For custom solutions that contain any configurations, the configurations must be backed up prior to this step, or else they will be lost.

3. Add the previous versions of the solution to the solution store and deploy to the farm.
4. Activate the features that are scoped for the web application and the farm.

Validate and Submit the Deployment Package

A CR number is necessary for you to submit the custom solution using MSOCAF. Open the CR with the service delivery manager (SDM). You must deliver the custom solution package at least 15 business days before the business day on which you want the solution to be deployed, to ensure that Microsoft can test and validate the solution and perform the deployment in a timely manner in the production environment.

It is not necessary to submit your custom solution to Microsoft from the same server on which you performed analysis of the solution. Analysis can be performed on one server and the folder structure copied to a different server for submission, as long as all of the prerequisites for submission detailed in this document are met.

Microsoft has a Performance Test Policy for custom solutions (available to current customers only, on the Customer Extranet site). Refer to this policy for information about testing requirements.

To validate and submit the custom solution deployment package:

1. Ensure that the root directory for the custom solution deployment package contains all required directories and components (listed in Table 1 earlier in this document).
2. Use MSOCAF to analyze the deployment package.
3. If any errors are reported, fix the errors and then rerun analysis.
4. Provide the required justifications for any remaining errors that will not be fixed.
5. Use MSOCAF to test the deployment of the custom solution in your test environment, and then to test the rollback.

Note

Any changes made after this testing will require steps 2, 3, and 4 to be repeated before you can proceed to step 6.

6. Use MSOCAF to submit the custom solution deployment package to Microsoft.
7. MSOCAF communicates with a web service hosted by SharePoint Online to facilitate submission of the deployment package.

Customer Responsibilities

- Adequately test all custom code before submitting it to Microsoft.
- Describe, test, and document any requirements concerning performance, scale, and language.
- Request that the SDM create a CR, and use the CR number given by the SDM when submitting the custom solution through MSOCAF.

Custom Solution Review and Deployment

The custom solution review and deployment steps are performed by Microsoft. For more information about these steps, see the SharePoint Online Custom Solution Policies and Process document.

Important

A submitted deployment package must contain the required directory structure, all files, and the most current templates as outlined in Required Directory Structure and Components in the Deployment Package earlier in this document; otherwise it is rejected.

As part of the deployment package review, Microsoft reviews the scope of the solution and determines whether it is a small, medium, large, or extra-large custom solution, based on the amount of code and number of .wsp files included in the custom solution deployment package. For more information about solution packages and .wsp files, see Solution Files earlier in this document. Part of the review includes testing, as described in Microsoft Testing of Custom Solutions later in this document.

Reasons for Custom Solution Rejection

FTC solutions must pass through a review and approval process, which includes submittal of the HLD document and validation using MSOCAF. There are various reasons why custom solution submissions to SharePoint Online can be rejected. Here are some of the most common mistakes that result in rejection. Avoid these mistakes, and your solution is much more likely to be accepted on first submission.

For complete information about submitting custom solutions, see the SharePoint Online Custom Solution Developer's Guide, available from the Microsoft Download Center.

Documentation Error

- The name of an artifact in the solution deployment documentation doesn't match an artifact name in the folders.
- Instructions are not included in the relevant document sections, especially for manual installations.

Script Failure in Lab

- Scripts use hard-coded values from the development environment (for example, site collection URL and domain names). Scripts should use values for the production environment, not for the development or pre-production environment.

Missing Deployment Instructions

- No rollback or retraction scripts are included in the submission, or scripts do not meet all rollback or retraction requirements.

Deployment Failure

- The installation sequence is incorrect; for example, an attempt to activate a feature before the feature has been created.
- There are script errors; for example, a feature not deactivated during rollback.
- Unsupported commands are used; for example, SharePoint 2010 functionality that has been deprecated in SharePoint 2013.

Policy Violation

- The complete policies are detailed in the SharePoint Online Custom Solution Policies and Process document, available from the Microsoft Download Center. Make sure your code does not violate any of the described policies, unless you received an exception when your HLD was reviewed.

PART II: Best Practices

This section presents recommended guidelines that have been developed by Microsoft for custom SharePoint solution development and performance testing.

Development Environments

- **Use native SharePoint functionality when possible.** You should always evaluate whether native SharePoint functionality can be used instead of writing custom solutions.
- **Evaluate the use of Cloud App Model.** When possible, you should evaluate whether the custom solution need can be met by using the new Cloud App Model. The client-side functionalities have been enhanced in SharePoint 2013 to encompass a broad set of APIs.
- **Minimize the amount of custom code.** The less code you need to write, the smaller the risk that bugs are introduced into the system. Also, the more quickly an application can be fixed and re-installed, the less downtime a system will experience. Sharing common code reduces the code footprint.
- **Replicate the SharePoint Online environment.** To help you build your local lab and test environments, we provide the SharePoint Online Build Guide with every service release (available to current customers only, on the Customer Extranet site). This guide describes the build settings and product versions that are used in the SharePoint Online production environment, so that you can mimic these settings in your development and test environments.
- **Ensure a complete software installation.** In the case where a development environment will have everything installed on the same server, see Requirements for Developing SharePoint Solutions in the MSDN Library.
- **Use source control and automated build processes.** Whether a project includes one or many developers, source control is very important. We recommend Visual Studio Team System 2010 Team Foundation Server for source control and build automation. Having a build automation capability makes it easy to create new environments as needed.
- **Integrate each developer's environment.** Developers work within their own environment and use source control to share and integrate their work. Each developer on a team requires an environment similar to the configuration outlined in the MSDN article Requirements for Developing SharePoint Solutions. A separate integration environment should also be maintained for formal daily builds. With a product like Visual Studio Team Foundation Server, an automated build process can be established where the checked-in code is compiled and deployed (usually scripted) to the central development server. The MSDN article Introduction to Team Development provides useful guidance on team development.
- **Create logical solution versions.** Version the DLLs and .wsp files in your solution. This is a significant help when determining whether the update of a solution has been successful.
- **Integrate FxCop into the build process.** Because MSOCAF validates solutions using FxCop and other tools, it can be beneficial to integrate these tools as a part of the actual build process. Fixing any issues identified by FxCop earlier in the process is much easier than waiting for MSOCAF to identify these after the custom solution is packaged. For more information, see FAQ: How do I run FxCop during a post-build event? in the Visual Studio Code Analysis team blog.

Note

There are community-based FxCop rules that are specifically designed for SharePoint development. You can add these rules to the base FxCop rules and include them in the build process. For more information, see SharePoint FxCop Rules on the Internet.

- **Move business logic to separate classes.** When planning the application architecture, it is often beneficial to separate each specific functionality into its own class. This allows for easier reuse and easier testing of the functionality. For example, when implementing a custom solution based on a timer job in SharePoint, we recommend that the actual functionality be implemented in a separate class rather than implementing it directly in the Execute method of the custom timer job class.

When the functionality is separated from the SharePoint implementation, it becomes much easier to write test applications (typically console applications) that invoke and test the functionality. After the specific functionality is tested and verified, you can then write and test the code to invoke the functionality from the actual SharePoint custom solution code.

- **Use unique namespaces for custom solutions.** Each feature of a SharePoint custom solution is deployed to the web front-end (WFE) server's file system, into a folder corresponding to the feature name. To minimize the risk of having two features overwriting each other's files, we recommend that each of the features be named with a unique namespace. For instance, instead of calling a feature Contract, call it [Company Name].[Solution Name].HR.Contract.
- **Use ADO.NET to connect to databases.** Microsoft supports connectivity to databases hosted outside the SharePoint Online data center using ADO.NET, although service-oriented architecture (web services) is preferred. We recommend that developers consider using a Windows Communication Foundation (WCF) application instead of a SOAP-based service whenever possible. For more information, see the MSDN article Guidelines and Best Practices for creating Windows Communication Foundation applications. When you do use ADO.NET to connect to a database hosted in another data center, make sure to follow the guidelines described in the section on connecting to SQL Azure™. These connections are often very latent, and you need to plan for security in making the connection. For information about connecting to custom databases, see Access Custom Database later in this document.
- **Use claims-based authentication and authorization.** By using a claims-based approach, it is much easier to perform line-of-business (LOB) integration to systems that are claims-aware. For more information about claims, see A Guide to Claims-Based Identity and Access Control Second Edition — Book Download in the Microsoft Download Center. Customers who choose to purchase the Partner Access feature can also develop SAML claims-based solutions as described in the SharePoint Online Partner Access SDK (available to current customers on the Customer Extranet site). Before beginning, verify that your SharePoint Online farm uses Windows Claims or SAML authentication. A claims-enabled custom solution will not work in a SharePoint farm that uses Windows Classic authentication.
- **Develop asynchronous code that connects to an Internet address.** We have observed slow crawl times for code or components that connect to Internet URLs. Examples are stock feeds, RSS feeds, and JavaScript elements from remote services like Google Analytics. We recommend that you write code using an asynchronous call system such as Asynchronous JavaScript and XML (AJAX) or jQuery.
- **Ensure that dependencies to external systems are managed correctly.** For solutions that have dependencies on external services, such as on-premises web services, you should ensure that the external systems are tested to handle the load and are highly available, and that the solution fails gracefully in the event that the external services are unavailable.

Recommended Guidelines for Increasing SharePoint Site Performance

Performance is usually a high priority for any website deployment, including SharePoint deployments. One of the bottlenecks to performance is often data retrieval, especially in navigation components.

Therefore, it is important to make sure that all navigation code is optimal. For more information about how to write well-performing code in SharePoint 2010, see Writing Efficient Code in SharePoint Server in the MSDN Library. Also make sure that proper caching is used to improve site performance as described in Object Caching Techniques in the MSDN Library.

- Configure with correct caching. All custom solutions should use the appropriate caching mechanisms. For example, the home pages of any portals or other frequently read sites must be configured to use output caching.

The output caching feature can be used on a site (or site collection) only if the SharePoint Server Publishing feature is activated for the site. For more information, see the Microsoft TechNet article Planning for Caching and Performance.

- **Scope Content by Query Web Parts properly.** Ensure that all Content by Query Web Parts is scoped properly. For example, if you must retrieve information from just one site, or from one list in a site collection, it is much less resource-intensive to configure the Web Part to read only that site or list. You should archive any content that is not necessary and that may be inadvertently considered in the scope of the Content by Query Web Part. Finally, ensure that the numbers of items in the lists that are being queried by the Content by Query Web Part do not grow too large, because this can also cause performance issues.
- **Use indexes on lists that are data sources.** You can substantially improve query performance by ensuring that at least one column in a view is indexed. In addition, if you are sorting or filtering in your view, you should use an indexed column. Using indexed columns in your views also helps to ensure that your queries do not trigger large list throttling in SharePoint 2013. For example, unexpected throttling may occur when retrieving fewer than 5,000 items if the list has more than 5,000 items, and if you are sorting on a non-indexed column. For more information about designing and working with large lists, go to SharePoint Server 2010 performance and capacity test results and recommendations in the Microsoft Download Center, and then download DesigningLargeListsMaximizingListPerformance.docx.
- **Avoid cross-site rollups.** We strongly recommend avoiding cross-site queries. SharePoint caching does not support rolling up of content from multiple site collections. This includes the out-of-the-box Content Based Query Web Part. If you must use cross-site queries, follow the guidance in Search Aggregated View in the MSDN Library. For more information about cross-site aggregation, see Aggregating Content within a Site Collection later in this document.
- **Use client-side query mechanisms.** For a better user experience, create components that run queries asynchronously from the client browser. This allows pages to load faster, so users can see some page content while other data is being retrieved. It can also reduce queuing on the SharePoint web front ends, which improves the overall experience for all users. SharePoint 2010 has very good support for this type of implementation, with features like a client-side object model, REST-based data access, numerous web service interfaces, and client query libraries like jQuery.
- **Use the Silverlight Web Part for Silverlight applications.** SharePoint 2013 includes a Silverlight Web Part, which supports the hosting of custom Silverlight applications. The Silverlight Web Part reads the actual Silverlight application from a Silverlight application (XAP) file. This XAP file can be stored in a document library in SharePoint, or it can be packaged as a solution file and submitted to Microsoft for deployment to the _layouts folder. The Silverlight application can then use the SharePoint client object model or web services to access and manipulate objects in SharePoint 2013.

Using SharePoint Web Services

Table 2 specifies whether the web services in SharePoint Foundation 2010 are supported by SharePoint Online and can be used in custom solutions. Similarly, Table 3 specifies whether the web services in SharePoint Server 2013 are supported by SharePoint Online.

Note

Whenever calls are made to these web services, the calls must be instrumented by using either SPMonitoredScope or SPDiagnosticsService. For more information, see Using SPMonitoredScope and SPDiagnosticsService Class in the MSDN Library.

Table 2. SharePoint Foundation 2010 Web Services

Web Service	Supported	Description
Admin web service	No	Provides methods for managing a deployment of SharePoint Foundation, such as for creating or deleting sites.
Alerts web service	Yes	Provides methods for working with alerts for list items in a SharePoint Foundation site.
Authentication web service	No	Provides classes for logging on to a SharePoint Foundation site that is using forms-based authentication.
CellStorage web service	Yes	Enables client computers to synchronize changes made to shared files that are stored on a server.
Copy web service	Yes	Provides methods for copying items between locations in SharePoint Foundation.
Diagnostics web service	Yes	Enables client computers to submit diagnostic reports that describe application errors that occur on the client.
DspSts web service	Yes	Provides a method for performing queries against lists in SharePoint Foundation.
DWS web service	Yes	Provides methods for managing Document Workspace sites and the data they contain.
Forms web service	Yes	Provides methods for returning forms used in the user interface when working with the contents of a list.
Imaging web service	Yes	Provides methods for creating and managing picture libraries.
Lists web service	Yes	Provides methods for working with lists and list data.
Meetings web service	Yes	Provides methods for creating and managing Meeting Workspace sites.
People web service	Yes	Provides methods for working with security groups.
Permissions web service	Yes	Provides methods for working with the permissions for a site or list.

Web Service	Supported	Description
SharedAccess web service	Yes	Provides a method that determines whether a document is being co-authored.
Sharepointemailws web service	No	Provides methods for remotely managing distribution groups.
Site Data web service	Yes	Provides methods that return metadata or list data from sites or lists in SharePoint Foundation.
Sites web service	Yes	Provides methods for working with websites.
Spsearch web service	Yes	Provides methods for remotely performing searches within a SharePoint Foundation deployment.
Users and Groups web service	Yes	Provides methods for working with users and groups.
Versions web service	Yes	Provides methods for managing file versions.
Views web service	Yes	Provides methods for working with list views.
Web Part Pages web service	Yes	Provides methods for sending and retrieving Web Part information to and from web services.
Webs web service	Yes	Provides methods for working with websites and content types.

Table 3. SharePoint Server 2013 Web Services

Web Service	Supported	Description
OrganizationProfileService web service	Yes	Provides an interface for remote clients to read and create organization profiles.
PublishedLinksService web service	Yes	Provides a published links interface for remote clients to read and create published links.
Search web service	Yes	Provides methods that can be used to remotely query SharePoint Server search.
Social Data web service	Yes	Provides an interface for remote clients to read, create, and manipulate social data.
UserProfileChangeService web service	Yes	Provides a user profile interface for remote clients to read and create user profiles.
UserProfileService web service	Yes	Provides a user profile interface for remote clients to read and create user profiles.

Using the SharePoint Cloud App Model

SharePoint 2013 introduces a new application development model that enables developers to build SharePoint applications that do not require full trust code to be run on the server. Starting with SharePoint 2013, the Cloud App Model is the preferred development model for SharePoint Online Dedicated.

The Cloud App Model supports three different hosting options:

1. **SharePoint-Hosted:** The entire app lives in SharePoint. HTML and JavaScript may be used, but no full trust code is allowed.
2. **Autohosted:** The portion of the app that contains full trust code is hosted in Windows Azure and optionally SQL Azure. These Azure components are provisioned invisibly and multitennancy is managed automatically. *This hosting option is not available in SharePoint Online Dedicated, or on-premises environments.*
3. **Provider-Hosted:** Similar to Autohosted in that any full trust code is hosted outside of SharePoint. However, the hosting of that full trust code is left completely up to the developer. This code may be hosted on an on-premises web server, or any other 3rd party hosting or cloud service. If supported, SharePoint Online Dedicated will provide guidance on configuring this hosted option.

It's important to note that the Autohosted option listed above is *not* supported by SharePoint Online Dedicated. You must use either the SharePoint-Hosted or Provider-Hosted models. The Provider-Hosted model also requires a trust relationship to be setup between SharePoint Online Dedicated and the customer provided endpoint that hosts the full trust code.

For additional details, see Getting started developing apps for SharePoint in the MSDN library.

Note

The above information is still in beta and specific guidance will be released after 13.1.

Using the SharePoint 2013 Client Object Model

SharePoint Foundation 2013 provides a variety of options for building and integrating applications and LOB systems with SharePoint technology using client-side object models. These options promote easy access to SharePoint functionality from remote clients. We strongly urge that, whenever possible, you employ client-side object models for your solution. This will allow you to bypass the custom solution review process and streamline deployment of the solution.

The client object model provides simple ways to add, read, update, and manage information that is stored in SharePoint. The primary goal of the client object model is to provide consistent and efficient APIs for manipulating SharePoint objects remotely. The client object model supports three programming models: JavaScript, .NET managed, and Silverlight. Client object model applications can also use the built-in support for Windows Communication Foundation (WCF) services and Representational State Transfer (REST) interfaces, in addition to using the legacy ASP.NET web services. For more information, see Using the Client APIs in the MSDN Library.

Writing Custom Solutions

Custom solutions, or "farm solutions," are full-trust code that is deployed to the SharePoint Online environment by Microsoft. To ensure proper integration and support, refer to the guidelines described in Table 4. It is not a comprehensive list of the guidelines for custom solution development, so if a topic is omitted, do not assume that it is allowed.

Table 4. Recommended Guidelines for SharePoint Custom Solutions

Guideline	Description
Modular deployable feature solutions	To be deployed, custom solutions (including resource files) must be contained within a SharePoint Feature, and packaged as a SharePoint solution package. The SharePoint Feature framework enables you to package sets of interrelated custom solutions as a single Feature, while also permitting inclusion of multiple Features in a solution package. Solution packages are the required form of deployment for custom solutions into the hosted SharePoint Online environment. This is because solution packages can be deployed, upgraded, and removed gracefully. For more information about solution packages, see Solutions Overview in the MSDN Library.
Configure custom solutions for production environment	Configure your custom solutions for the production environment, not for the PPE or test environment. For example, after the custom solution is turned over to Microsoft, you should not submit requests to edit URLs to change them from the PPE to production environment. Instead, configure your custom solution with URLs appropriate to the production environment. Then include comments in the web.config file where changes need to be made for the PPE, and we will make the appropriate changes. Alternatively, you can include notes in the deployment guide specifying the configuration details that are specific to the PPE environment, and we will make changes accordingly.
Do not attempt direct SharePoint database access	Custom solutions should not directly access SharePoint Online databases. SharePoint data should only be updated using the SharePoint object model, without attempting direct access to the SQL Server® data structures. This protects both the SharePoint Online service during upgrades as well as data integrity.
Use Secure Store Service	If your custom solution connects to LOB applications, on-premises databases, or SharePoint Online databases, you can use the Secure Store Service so that user names and passwords are encrypted. However, we do not support the use of Secure Store Service for delegated identity. This means that you cannot use Secure Store Service to pass SharePoint user credentials from the SharePoint farm to an on-premises system. For more information, see Access Custom Database later in this guide.
Referencing SPWeb or SPSite objects	In general, when referencing the SPWeb or SPSite objects, you should employ either a "using" statement or an explicit call of the Dispose method to ensure proper use and disposal of the memory objects. For detailed guidance on exceptions to this general rule, see the MSDN article Best Practices: Using Disposable Windows SharePoint Services Objects.

Guideline	Description
Avoid hard-coding SharePoint artifacts	The locations of SharePoint artifacts (like web services, web apps, site quota template names, and lists) should always be configurable values and should not be hard-coded in your custom solutions. This will prevent problems such as features working in the PPE but not working in the production environment or vice versa.
Reduce unnecessary round trips	Use caching as appropriate to reduce unnecessary round trips. Improper management of the number of round trips required for interactions with custom solutions can be exacerbated by latency conditions. This results in a poor end-user experience, while potentially affecting back-end service performance.
Minimize attack surface	Input surfaces in custom solutions must include boundary checks, input data integrity checks, and appropriate exception handling to prevent the potential for cross-site scripting and SQL injection.
Specify Code Access Security (CAS) policy and safe controls	When packaging a solution, you should include a Code Access Security (CAS) policy for that solution. If necessary, include your assembly in the safe controls list through the solution. With CAS, code is given explicit permissions and can run only under those permissions, regardless of the permissions of the user executing the code. Requiring CAS and registering safe controls allows SharePoint Online to maintain a secure environment while allowing custom solutions.
Error handling and logging	When writing error-handling code, inherit from the SPException class to ensure that the errors are consistent in appearance with any errors from SharePoint Server. Trap errors at all possible entry points, including constructors. Use logging where appropriate to help troubleshoot errors. Be sure to log all errors on .aspx and Web Part pages to the Unified Logging Service (ULS) log.
Use delegate controls	Determine whether the functionality or custom solution to be developed needs to be done as a Web Part or if it can be done using delegate controls. By using delegate controls, you can easily replace existing controls that are already in the page using less custom code. For more information about delegate controls, see Delegate Control (Control Templatization) in the MSDN Library.
XML mapping	Use the AppSettings object to implement the XML mapping. This can be provided out of the box using the existing Settings framework in .NET Framework 3.5. Avoid creating custom XML files and a strongly typed object for this purpose.
Avoid unsafe updates	Avoid using AllowUnsafeUpdates. Use ValidateFormDigest() and, if necessary, interact with SharePoint Server objects using elevated privileges. In cases where AllowUnsafeUpdates must be used, set AllowUnsafeUpdates to False in the try-catch finally block, or a Dispose() method as required by the IDisposable interface to avoid security holes.

Guideline	Description
Deployment logging	Include detailed messages in the event logs during installation and de-installation to enable appropriate operational troubleshooting.
Limit results when querying large lists and updating large lists	For details about limiting results when you query and update large lists, see Best Practices: Common Coding Issues When Using the SharePoint Object Model in the MSDN Library.
Provide source code and documentation to Microsoft	Whenever possible, provide source code for third-party solutions, along with documentation, to ensure support.
Provide licensing disclosure for third-party solutions	Full disclosure must be provided to Microsoft for licensed solutions that are to be deployed to the SharePoint Online environment. This must be provided as part of the custom solution deployment package.
Include a failover server in web.config	All custom database connections should be configured with a failover partner; otherwise, the database will be unavailable in the event of a failover. **Syntax**: *connectionString="Data Source=SERVER_NAME;Failover Partner=SERVER_NAME;Initial Catalog=DATABASE_NAME;Integrated Security=True;Pooling=True"*

General Guidelines for Custom Solutions

Plan carefully when installing dependencies. Solutions have configuration and application dependencies. Dependencies are not always straightforward, nor are they static over the course of development or even over the life cycle of the code in production. Some dependencies are used by multiple applications. The development team must do proper planning, especially when there are dependencies between solutions.

Limit the number of .wsp files in a custom solution. Large numbers of .wsp files that need to be tracked and managed pose a challenge for deployment. The more .wsp files in a custom solution, the greater the possibility that something will not be installed correctly, the longer the solution will take to deploy, and the greater the chance of mixing incompatible versions of solution packages. We recommend that you scrutinize your deployment plan and keep the number of solution packages to the minimum number needed for the project. Keep in mind that a solution package can contain several features, so there is no necessity to have one solution package for each feature.

Write effective code. Avoid creating unnecessary objects in code. This includes creating SPSite and SPWeb objects as described in these MSDN articles: Optimizing Code Performance and Avoid Unnecessary Construction of SPWeb and SPSite Objects.

Schema-based Development

Not all SharePoint artifacts require compiled code. You can create certain types of custom solutions using XML definition files that can be packaged and deployed as .wsp files.

These non-code artifacts include:

- Site columns

- Content types
- List definitions
- Site templates
- Web templates

Note

Organizations can centrally manage content types without having to deploy them to each site collection separately, by using the content type hub functionality in SharePoint 2013.

Branding

Master pages are a feature of ASP.NET and share common functionality in SharePoint Foundation and SharePoint Server 2013. Master pages define a predesigned style and layout template for site collections and subsites. SharePoint 2013 provides support for master pages for both content pages and application pages. For information about best practices when modifying master pages, on SharePoint 2013 refer to Web Content Management and Branding in the MSDN Library.

Although you can apply master pages manually after site creation, we recommend that you automate this by using feature stapling, as described below.

Feature Stapling

Feature stapling is a method by which you can apply or attach a Feature to all new sites that use a given site definition. This preserves the site definition integrity without the complexity and overhead of using code to activate Features. Feature stapling is specifically designed to apply other Features to one or more site definitions, in addition to making those Features available to any new sites created from any site definition.

For more information, see Feature Stapling in the MSDN Library.

Instrumentation

SharePoint 2013 includes two new classes that can be used for instrumentation purposes.

- **SPMonitoredScope** can be used by custom solutions that run in the HTTP context of SharePoint (Web Parts, Field Controls).
- Custom solutions that do not run in the HTTP context (timer jobs, event receivers, and feature activation callouts) should use the **SPDiagnosticsService** classes to log information in the ULS logs. For more information, see Logging for SharePoint Developers in the MSDN Library.

SPMonitoredScope

By using the SPMonitoredScope class, custom solutions can add information that is shown when using the Developer Dashboard functionality of SharePoint 2013. The Developer Dashboard facilitates the debugging and troubleshooting of issues with page rendering times by providing additional performance and tracing information. For more information, see Using SPMonitoredScope in the MSDN Library.

SPDiagnosticsService

By using the SPDiagnosticsService class, custom solutions can easily write to the ULS log and to the event log. As a general guideline, each custom solution should use SPDiagnosticsService calls before and after time-consuming calls or when calls outside SharePoint Server are made. For more information, see SPDiagnosticsService class in the MSDN Library.

Enable Cross-Domain Access

You can enable cross-domain access to SharePoint Online services using Silverlight. There are two methods for doing so:

- clientaccesspolicy.xml file
- crossdomain.xml file

Whichever solution you use, it must be deployed like any other custom solution, using a .wsp file and following the custom solution deployment process and policies. Do not deploy the .xml file using SharePoint Designer; doing so will result in the file not being recognized. The deployment process is described in full detail in the SharePoint Online Custom Solution Policies and Process document available from the Microsoft Download Center.

For details on enabling cross-domain access, see Making a Service Available Across Domain Boundaries in the MSDN Library.

Audience Targeting

If you plan to develop a custom solution that uses SharePoint's audience targeting functionality or the Membership Web Part, a CR may be required to enable the importing of Active Directory groups (this setting is disabled by default). You need to plan ahead to allow time for the group import CR approval process to complete before deploying your custom solution. For more information about the CR process and timeline, see the SharePoint Online Configuration Request Guidelines, available to customers on the Customer Extranet site. Enabling group import will impact the performance of the User Profile Synchronization process. The process may take days, rather than hours, to complete.

Writing Web Parts

Web Parts can serve as "building blocks" for custom web applications and pages. Solutions that include Web Parts may be written and deployed by either the administrator of a website or the individual users of the website to provide easy functional customization. Table 5 lists guidelines that are specific to Web Part development. Questions regarding use of custom Web Parts should be directed to the SDM. For more information about writing Web Parts, see Building Block: Web Parts in the MSDN Library.

Table 5. Recommended Guidelines for Using Web Parts

Guideline	Description
Use SharePoint Server architecture	The SharePoint Online architecture provides several frameworks for functionality, such as feature deployment. All custom Web Parts should use the SharePoint Server architecture to ensure consistent behavior across the application.
Configuration of Web Parts	The configuration of the Web Parts being deployed should allow the flexibility of deploying to the web application level or lower. This improves supportability and maintenance. You are encouraged to include configuration data with the Web Part itself as metadata.
Web Part connections	The SharePoint Web Part infrastructure provides a standardized set of connection interfaces that allow Web Parts to exchange information with each other at run time. You should use these when it is necessary to pass data between Web Parts or between a list and a Web Part. There are a number of SharePoint Server filter Web Parts that already emit data. Your Web Part should be able to receive that data if connection with those Web Parts is desired.
Separate user interface (UI) and presentation layer	When possible, abstract the presentation layer from the code logic by using either user controls (ASCX) or XSL. If the Web Part requires form input, you should utilize user controls; otherwise, the Web Part should output raw XML and apply an XSLT to format the output. When using an XSL, store the XSLT as a Web Part property, allowing for modification from the tool pane. Avoid using a data tables/dataset approach and HTML formatting in the Web Part.
Provide source code and documentation to Microsoft	Source code for third-party Web Parts solutions, whenever possible, should be provided with good documentation to ensure good support.
Provide licensing disclosure for third-party Web Parts	Full disclosure must be provided to Microsoft for licensed Web Parts that are to be deployed to the SharePoint Online environment. You provide this information as part of the custom solution deployment package.

Writing Timer Jobs

Timer jobs are a key building block for performing activities in the SharePoint environment. To provide proper monitoring, you should include logging in the Execute method of the timer jobs. Also try to separate the business logic for a timer job into a separate class that is invoked by the Execute method of the timer job. For more information, see How to: Run Code on All Web Servers in the MSDN Library.

Writing Event Receivers

When writing event receivers that are to be deployed to SharePoint Online, you must implement tracing calls for each event. Select method overloads carefully to ensure that the calling context is the one expected. For information about best practices for event receiver development, see Best Practices with Event Receivers in the MSDN Library.

Writing Feature Receivers

When Feature receivers are used, developers must implement tracing calls for each of the methods that they override.

Calling Web Services

Whenever the code in the custom solution is calling a web service, proper instrumentation should be implemented in the custom solution. To aid in troubleshooting, each call to a web service must have a logging call both prior to and after the call. This information can be used by the SharePoint Online operations team for troubleshooting purposes.

Configuration Management

SharePoint can be configured by modifying the web.config file or through changes to search settings (scopes, managed properties), audience information, and other settings that are typically accessed via the Central Administration page.

Custom solutions that must change the configuration of SharePoint must be implemented using Feature receivers. The Feature receivers can use the appropriate SharePoint APIs to update both web.config and the SharePoint settings. Microsoft reserves the right to reject any custom solutions that require manual configuration of web.config. For more information, see How to: Add and Remove Virtual Web.config Settings in the MSDN Library.

Storing Application Configuration

Applications that must store configuration data (such as URLs or list names) should specify these properties in the <properties> section in the feature.xml file. During feature activation, the applications can then read the properties and store them in the hierarchical object store provided by SharePoint or in a SharePoint list. For more information, see Managing Application Configuration in the MSDN Library.

Windows Claims Authentication

All new SharePoint Online Dedicated customer farms are now built using Windows Claims authentication. Customer farms that were built prior to 11.3 can still use Windows Classic authentication for their web applications, but will be migrated to Windows Claims sometime before their upgrade to the next version of SharePoint Online. Depending on the type of customization, it is possible that code that worked prior to the migration to Windows Claims may not work as expected after migration due to changes in the way identity is managed. Although the authentication store is still Active Directory, Windows Classic authentication is different than Windows Claims. It is very important to understand the Identity Claim, how it works on Windows Claims, and the impact it can have on custom solutions.

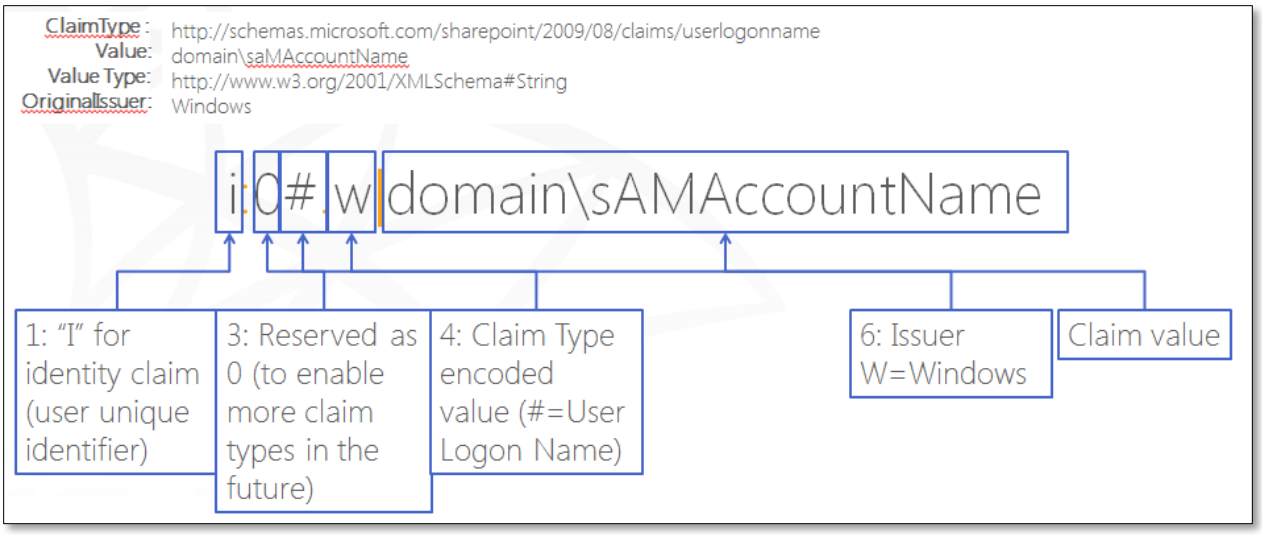

Figure 1. Identity Claim (Windows)

Note

If the claim is encoded, then the casing for encoded claims MUST be lower case and invariant culture, upper case MUST not be used. Characters 1 through 5 (i:0#.w|) are case-sensitive. These restrictions apply only to the encoded claim string. Thus ToUpper corrupts the encoded string and the EnsureUser method breaks with the above error. The output of the claim methods are all in uppercase

The typical Windows Claims authentication logon user value will look similar to that shown in Figure 1 above. It differs substantially from the sAMAccountName you usually get with Windows Classic. This difference will have significant impact on scenarios where custom solutions rely on that value.

Here are some of the major scenarios where custom solutions may rely heavily on the user objects:

- Any code that uses the following classes:
  - System.Security.Principal.WindowsPrincipal
  - System.Security.Principal.WindowsIdentity
  - System.Security.Principal.NTaccount
  - System.Security.Principal.SecurityIdentifier
- The use of Thread.CurrentPrincipal where the expectation is that a WindowsPrincipal is returned.
- Any hard-coded user or application identities in the code, as well as code that parses or otherwise constructs a user name on the fly; a common example is site provisioning code that provides a user identity for the primary site collection owner.
- Outbound WCF calls.
- Client-side object model code (that is, Sandboxed solutions) that calls out-of-the-box or custom WCF endpoints.
- Code that changes Thread identity or otherwise impersonates a user.
- My Site and profile customizations.

Consider the simple scenario of programmatically adding a user to an existing group. The easiest way to avoid complications with claims encoding while constructing the user name is to use the SPWeb.EnsureUser method. Otherwise there will be logon issues. This is simpler than having to work around to get the encoding for an account name. It's also easier, because it's only necessary to pass in the

account name instead of the four parameters normally needed to add a user. EnsureUser automatically takes care of encoding the name and simplifies the code.

```
string login = "Domain\UserName";
string groupName = "Group";
using (SPSite theSite = new SPSite("http://team.contoso.com"))
     {
       using (SPWeb theWeb = theSite.OpenWeb())
       {
         SPUser user = theWeb.EnsureUser(login);
         SPGroup group = theWeb.Groups[groupName];

group.AddUser(user);
       }
     }
```

The same recommendation applies to similar scenarios like these:

- Adding a user as site collection administrator.
- Adding a user and using SPRoleAssignment to provide access to a site.

There are cases where it might be necessary to pass the encoded claims string as logon name. If the goal is just to get the encoded claim value for a particular user, then we recommend using SPClaimProviderManager.

For example, to get an encoded claims string, you may have to call SPClaimProviderManager.ConvertIdentifierToClaim Method:

```
SPClaim spclaim =
 ConvertIdentifierToClaim(“domain\accountname”,
SPIdentifierTypes.WindowsSamAccountName);
String encodedClaimString = null;
if (null != identity)
       {
           try
           {
               encodedClaimString = claimManager.EncodeClaim(identity);
           }
           catch
           {
                // just catch any exceptions and handle them appropriately
           }
       }
```

For more information on SPClaimProviderManager, see SPClaimProviderManager Members in the MSDN Library.

A simple scenario in which to pass the encoded string is when it is necessary to create a site collection and assign an owner to the site collection.

The owner login should be passed as encoded string. For example, here the owner login should have the claims encoded string. Note that the `encodedClaimString` variable below would be populated with a claims encoded string value via a process similar to the previous example:

```
SPWebApplication webApp = new SPSite("http://MySiteCollection").WebApplication;
SPSiteCollection siteCollections = webApp.Sites;

SPSite newSiteCollection = siteCollections.Add("sites/Site_Collection",
    "encodedClaimString", "Email_Address");
```

SPSiteCollection.Add Methods has several overloads. For more information, see SPSiteCollection.Add Method in the MSDN Library.

If you are working with the PickerEntity class, the PickerEntity.Key property will be looking for an identifier of a database record. While assigning value, the value should be the encoded claim value, as in this example:

```
PickerEntity pe = new PickerEntity();
          SPSecurity.RunWithElevatedPrivileges(delegate()
          {
              using (SPSite thisSite = new SPSite(this.Site.ID))
              {
                  pe.Key = “ClaimsEncodedStringForDomainUser”;
               }
          });
```

When you work with the client object model, the UserCreationInformation.LoginName property will expect the encoded string. For example, you have to pass the claims encoded user:

```
     ClientContext clientContext = new
ClientContext(“https://team.contoso.com/”);
            GroupCollection collGroup = clientContext.Web.SiteGroups;
            Group oGroup = collGroup.GetById(6);

            UserCreationInformation userCreationInfo = new
UserCreationInformation();
            userCreationInfo.Email = "alias@somewhere.com";
            userCreationInfo.LoginName = "ClaimsEncodedStringForDomainUser";
            userCreationInfo.Title = "Admin";

            User oUser = oGroup.Users.Add(userCreationInfo);
            clientContext.ExecuteQuery();
```

Here are other scenarios where you may need to pass the encoded strings:

- When you construct the SPQuery and pass any identity filters.
- When you work with the SPUtility class under Microsoft.SharePoint.Utilities namespaces.

There are several methods that may not yield correct results if you do not pass the encoded string. For example, when you leverage the SPUtility.GetPrincipalsInGroup to enumerate the principals of the users in a group, you may have to pass the encoded user string for the method. If you do not do so, the method shown below will return null:

```
SPPrincipalInfo[] principals = SPUtility.GetPrincipalsInGroup(temWeb,
“ClaimsEncodedStringForGroup”, 100, out reachedMax);
```

For more information, see SPUtility Members in the MSDN Library.

When you work with SharePoint identities in Windows PowerShell, you have to create an SPClaimsPrincipal for the Identity. For example, use the following Windows PowerShell snippet to create a site collection and assign an owner:

```
#Get-SPWebTemplate
>$template = Get-SPWebTemplate "STS#0"

#Get User Principal

>$cp = New-SPClaimsPrincipal -Identity "redmond\SiteOwner" -IdentityType 1

#Create Site Collection

>$sp = New-SPSite http://servername:port -OwnerAlias $cp.ToEncodedString() -
Template $template
```

Another common example of code that may work differently in the Windows Claims environment is code that parses a domain-qualified user name and expects to find a backslash separating the domain and user name. The first part of the substring is not going to return the domain name alone, because it contains the claims encoding values with the domain name.

```
SPUser user = GetUser(); //arbitrary method that returns an SPUser
string domain = user.LoginName.Substring(0,
     user.LoginName.IndexOf(“\\”));
string message = “The domain is: “ + domain;  //this will display unexpected clai
ms encoding information before the domain info.
```

Aggregating Content

If you want to aggregate content, you can deploy a custom solution that consumes list data and then displays results in either a Web Part or a custom control. The following techniques should be used to write custom solutions that will minimize the impact on performance when deployed to the SharePoint Online environment. Such custom solutions can be designed to aggregate content within a single site collection or across several of them.

Data can be consumed for aggregation by using the search interface within the Query Web service, by using the List Web service, by using the REST API, or by using the SharePoint client object model. Among these options, we recommend using the search interface, because it typically has the lowest impact on the production environment and better query performance. However, the design of the custom solution

should consider the required freshness of the results, because results from a search-based method of consuming the data will only be refreshed after each search crawl (typically once a day). That is usually sufficient to meet most roll-up and reporting requirements. For more information, see List Aggregation Patterns in the MSDN Library.

Consuming Data with Search Using Search Object Model or Query Web Service

The Query Web service exposes the robust search functionalities of SharePoint for both structured and unstructured contents. Custom solution developers can also leverage the search object model to query data against the site collection. Both the Query web service and object model support returning multiple result types.

- The Query Web Service is a SOAP-based ASMX web service, and supports a number of operations, including querying and getting search results, getting query suggestions, and getting metadata such as managed properties of a list.
- The search object model provides a unified interface for querying against different search providers. It provides a unified interface for searching against different locations and engines. It also allows for combining and merging of results. Using the search object model, you can create custom Web Parts to execute search queries and display the search results returned.

You can consume the Query Web service reference, where <server> is the SharePoint site from which you want to retrieve the results:

- http:// <server>/_vti_bin/search.asmx

A custom Web Part can consume the Query Web service as follows:

1. Instantiate the web service.
2. Use the credentials of the user running the client application.
3. Execute the QueryEx method, returning the results to a DataSet.
4. Set the DataGridView data source to the first table in the DataSet object, which contains the relevant results. (GetXMLString can be written to collect the data.)

```
QueryWebServiceProxy.QueryService queryService = new QueryWebServiceProxy.QuerySe
   rvice();
queryService.Credentials = System.Net.CredentialCache.DefaultCredentials;
System.Data.DataSet queryResults = queryService.QueryEx(GetXMLString());
resultsGrid.DataSource = queryResults.Tables[0];


StringBuilder xmlString = new StringBuilder("<QueryPacket xmlns='urn:Microsoft.Se
   arch.Query'>
<Query><SupportedFormats><Format revision='1'> urn:Microsoft.Search.Response.Docu
   ment:Document
</Format></SupportedFormats><Context><QueryText language='en-US' type='STRING'>")
   ;
xmlString.Append(<query to be executed>);
xmlString.Append("</QueryText></Context></Query></QueryPacket>");

return xmlString.ToString();
```

Consuming Data with the List Web Service

Through ECMAScript: The ECMA client object model can be used in Web Parts or controls within a page by referencing an ECMAScript (JavaScript) file to asynchronously consume the List Web service.

- Reference <SharePoint:ScriptLink Name="sp.js" ……. />. The file is located in the path "Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS
- You can also use jQuery with JavaScript by adding a reference to jquery.js file in your custom solution.
- To execute any JavaScript function on a page load event, put references to the JavaScript and jQuery files inside the ExecuteOrDelayUntilScriptLoaded function.
- The List Web service can be called in your JavaScript code, and you can use the XmlHttpRequest object directly or make use of a JavaScript library that wraps the XmlHttpRequest object.
- Make sure that the page is loading the JavaScript library.
- When the page is loaded, the SharePoint List Web service must be called. This can be accomplished by making use of the jQuery AJAX method (for example, calling http://yoursite/_vti_bin/lists.asmx). This method can post the necessary SOAP envelope message to the List Web service. The XML of the SOAP envelope can easily be copied from the .NET Web service test form of the desired web method (for example, http://yoursite/_vti_bin/lists.asmx?op=GetListCollection).

For example, the code sample below calls the GetListCollection web method when the page is loaded. The complete parameter of the AJAX method is actually a pointer to another JavaScript function that can be called asynchronously when the web service call is done.

```
$(document).ready(function() {
   var soapEnv =
   "<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
   \
   <soapenv:Body> \
   <GetListCollection xmlns='http://schemas.microsoft.com/sharepoint/soap/'> \
   </GetListCollection> \
   </soapenv:Body> \
   </soapenv:Envelope>";

$.ajax({
   url: "http://yoursite/_vti_bin/lists.asmx",
   type: "POST",
   dataType: "xml",
   data: soapEnv,
   complete: processResult,
   contentType: "text/xml; charset=\"utf-8\""
   });
   });
```

The `processResult` function is called when the response XML of the web service call is received. In this method a loop is created which will iterate over every List element of the response XML. For every List element a `<li></li>` element is added to the element with the ID attribute set to data.

```
function processResult(xData, status) {
   $(xData.responseXML).find("List").each(function() {
   $("#data").append("<li>" + $(this).attr("Title") + "</li>");
```

```
    });
    }
```

Through Silverlight: A Silverlight Web Part or a Silverlight application hosted within a Silverlight Web Part can be designed to asynchronously consume the List Web service. You can add a web reference to "http://my/sites/your-alias- here/_vti_bin/lists.asmx" and create a proxy for the List Web service.

```
ListsSoapClient proxy = new ListsSoapClient();
proxy.GetListItemsCompleted += new EventHandler<GetListItemsCompletedEventArgs>(p
    roxy_GetListItemsCompleted);
XElement Query = new XElement("Query");
XElement viewfield = new XElement("ViewFields");
XElement QueryOptions = new XElement("QueryOptions");
proxy.GetListItemsAsync("MyContacts", null, Query, viewfield, null, QueryOptions,
    null);
```

Then the callback method `GetListItemsCompletedEventArgs` can be used to process list data that is retrieved.

Consuming Data with the REST API

A RESTful service models SharePoint lists as HTTP resources that can be addressed by a URL. You can append query strings to the URLs in order to specify filter criteria or query logic. The SharePoint REST API is based on the Open Data protocol (OData) for web-based data services, and can return output in the Atom and AtomPub syndication formats (exchanging XML data over HTTP) or in JavaScript Object Notation (JSON) format.

The following examples show some URLs that can be consumed to access the list data through the REST API:

- To return the contents of the Parts list in XML format as an Atom feed
  `http://localhost/_vti_bin/listdata.svc/Parts`
- To return the Parts list item with an ID value of 3 as an Atom feed
  `http://localhost/_vti_bin/listdata.svc/Parts(3)`
- To return the Parts list as an Atom feed, ordered by the Name field
  `http://localhost/_vti_bin/listdata.svc/Parts?$orderby=Name`

To improve performance when querying list data for any custom solution deployed to the SharePoint Online environment, use the REST API asynchronously as follows:

1. Create an HTTP request using the `POST` verb.
2. Use the service URL of the list to which you want to add an entity as the target for the `POST`.
3. Set the content type to `application/json`.
4. Serialize the JSON objects that represent your new list items as a string, and add this value to the request body.
5. Use jQuery to access the list data with the REST API:

```
var url = 'http://localhost/sites/sharepointlist/_vti_bin/listdata.svc/InventoryL
ocations';
var inventoryLocation = {};
// Insert a new Part location.
inventoryLocation.PartId = $('#hidPartId').val();
inventoryLocation.BinNumber = $('#binText').val();
inventoryLocation.Quantity = $('#quantityText').val();
var body = Sys.Serialization.JavaScriptSerializer.serialize(inventoryLocation);
```

```
$.ajax({
         type: 'POST',
         url: url,
         contentType: 'application/json',
         processData: false,
         data: body,
         success: function ()
         {
           alert('Inventory Location Saved.');
         }
        });
```

Consuming Data with the Client Object Model

A custom solution can also call the client object model to consume the list data. For more information about leveraging the client object model, refer to the TechNet blog article Using the SharePoint Online 2010 Client Object Model.

Displaying Aggregate Data to the User

Regardless of how a custom solution consumes the list data, you can develop an ECMA Web Part, a Silverlight Web Part, a custom Web Part, or even a custom control that will display the data after it has been retrieved from the server. An asynchronous technique like using AJAX or jQuery should be used to query and render the aggregated data to prevent the code doing the aggregated list look up from blocking the page load.

- **ECMAScript Web Part.** An out-of-the-box ECMAScript Web Part can consume data via ECMA Script object model loaded from the Web Part. You can call the client object model through Java Script to access the list data from SharePoint, or you can use REST API or the List Web service.
- **Silverlight Web Part.** An out-of-the-box Silverlight Web Part can be designed to use the List Web service, the REST APIs, or the client object model, or to access the data from lists. It can also be used to host compiled Silverlight application (XAP) files.
- **Custom Web Part.** Any custom Web Parts used for aggregating content should be designed in such a way that they will run on the client and asynchronously fetch the data from the SharePoint environment.
- **User control or dialog control within a page.** As an alternative to a Web Part, you can also use a user control or dialog box in your page to display the site collection rollup contents hosted within an AJAX or Silverlight control. The key to developing such controls is to minimize the number of post backs and round trips to the server. These controls can cache the data at the client side, improving the performance by reducing round trips to the server.

Minimizing Load Time

- **For ECMAScript.** Decide whether to use inline scripts or a different JavaScript file for your application. For a small amount of JavaScript (less than 10 kilobytes) where functions are not reused across controls or Web Parts, it may be preferable to use the inline scripting approach, especially if the JavaScript file will not be cached.
- **For a Silverlight application.** The load time for Silverlight applications corresponds to the size of the XAP file. Though the XAP files can be cached like other web resources, there is still a performance penalty on page load, because the XAP file still needs to be parsed.

Improving Performance with Caching

In the case of AJAX clients, maximize browser caching of resources by referencing JavaScript library files in a central location.

In addition, you should also use the cache option in jQuery AJAX calls whenever possible:

```
$.ajax({
   type: "GET",
   url: "test.js",

   Cache = true,
   dataType: "script"
 });
```

Upgrading Features Within Custom Solutions

SharePoint 2013 provides rich support for Feature upgrading and versioning. You have two ways of handling Feature upgrades:

- Defining activities as CustomUpgradeActions in the Feature definition file that gets triggered when a Feature is upgraded
- Implementing the FeatureUpgrading events in the Feature receiver

The Feature definition can also contain information that defines the range of versions to upgrade. This means that developers can choose different upgrade actions for different versions of the solution. For more information, see Feature.xml Changes in the MSDN Library.

When upgrading schema-based artifacts defined by Feature elements, you should consider whether the Feature has been customized in the content database (for example content type, site column), or whether it is found only in the file system of the web front-end server. This is important, because a non-customized schema-based artifact exists only in the file system of the web front-end server and can thus be more easily upgraded. If, however, the Feature is customized or exists in the content database, it cannot be upgraded simply by upgrading the Feature definition file.

An example of a Feature that exists in the content database is a custom content type that is associated with a list. Whenever a content type is associated to a list, the schema of the content type is written to the content database. When upgrading this category of customized Features, this must be done using the SharePoint object model.

Writing Manual Installation Scripts

If possible, you should avoid using scripts outside of .wsp files. For those components that must be deployed manually using scripts (rather than with MSOCAF), follow these best practices:

- If you must use scripts outside of .wsp files, try to use only one script. At most, you can use two scripts: one pre-deployment script and one post-deployment script.
- Make sure your scripts comply with the deployment feature in MSOCAF to facilitate automated deployment.
- Use Pause and Sleep commands in scripts.

Use Pause and Sleep Commands in Scripts

Use Pause and Sleep commands in installation scripts. The Pause command can be used to hold the batch file currently running until the user presses any key. This helps the user to verify the output or wait until the previous command execution has succeeded, in scenarios such as:

- Pauses between adding and deploying a solution
- Pauses between deploying a solution and any other immediate command
- Pauses between retracting and deleting a solution
- Pauses between deleting a solution and any other immediate command
- After any command that requires a delay to execute

Similarly, the Sleep command in the Windows PowerShell command-line interface enables you to pause Windows PowerShell activity for a specified period of time for such scenarios.

Do not use "–ErrorAction SilentlyContinue" or "-EA 0"

All scripts involve enumeration of SharePoint objects are required to have file based logging in place to support trouble shooting needs. Here is a code sample for creating log files and write log entries.
At the minimum, **we require customers to log the script execution start and end timestamp and execution status (Success, Failure). The log file will be saved at the current execution directory.**

```
$date = Get-Date
$outFile = ".\<file_name>_" + $date.ToFileTime() + ".log"

{
    try{
        Write-host “Start Execution”;
        Write("Begin Processing") >> $outFile

        #Logic

        Write(“Status: Success”) >> $outFile
    }
    catch
    {
        write("Error while processing. Message " + $Error[0].Exception) >> $outFile
        Write(“Status: Failed”) >> $outFile
    }
    finally
    {
        Write(“End Processing ”) >> $outFile
        Write-host “End Execution”;
    }
}
```

Users get to control what happens when a cmdlet issues a non-terminating error. This is done via the common parameter –ErrorAction ("-EA" is the alias).

Action preference used with -ErrorAction are:

- "SilentlyContinue": Do not print, continue – Not allowed in MSO scripts.
- "Continue": Print, continue (this is the default)

- "Stop": Halt the command or script
- "Inquire": Ask the user what to do

Use of "SilentlyContinue" parameter suppresses any error messages that might occur during script execution. These error messages are never printed on console and any valid failures will not be identified during script run. Make sure –ErrorAction SilentlyContinue or -EA 0 is not used in your scripts. In case of any known issues to be ignored during script execution, ensure the errors to be ignored are properly called out in script execution instructions in the deployment guide

Recommended Guidelines for Testing

The following guidelines for the testing process are used and recommended by the SharePoint Online internal development teams.

Scope of Testing

- Do not limit testing to only what has changed. Regressions in the code can impact seemingly unrelated portions of the custom solution. We recommend full test passes, even for small updates. Ultimately, the extent of testing is at your discretion.
- The secret to success with testing against SharePoint is to adhere to a process and an agile methodology. For guidance on testing concepts and techniques, see Testing SharePoint Solutions in the MSDN Library. For details about our testing policy and your testing responsibilities, see the SharePoint Online Performance Test Policy (available to current customers only, on the Customer Extranet site).

Microsoft Testing of Custom Solutions

When presented with code from customers, Microsoft performs the following tests:

1. Check for memory leaks.
    - Check for object disposal.
    - Does not use Microsoft Win32 APIs (unmanaged code).
    - Verify that file handles are closed.
    - Verify that connections are closed.
    - Run MSOCAF.
2. Determine whether the error handling and exception handling are sufficient.
    - Verify that all the protected methods can handle exceptions, and log exceptions to the ULS log.
    - Verify that exception handling in the Feature receivers is re-thrown; if not, the Feature is only partially activated.
    - Verify that proper instrumentation and error handling has been included in the custom solution.
    - For events that require response from the SharePoint Online operations team, write them to a customer-specific event source, of the form *customer_log*, for example *contoso_log*. This custom event source is created by your development team in the event log. For further details about event logs, see How to: Create and Remove Custom Event Logs in the MSDN Library.
    - Verify that any event logs that are generated from in-house or third-party code have unique IDs and include source, category, type, and description.

3. Walk through the code to determine whether all custom solutions fall within the scope of the custom solution policy presented in this document.
4. Evaluate whether all cases where elevated privileges are requested by the code are needed.
    - Verify that the relevant SafeControl entries exist in web.config.
5. Evaluate deployment.
    - Verify that the deployment does not overwrite any system files (pre-existing SharePoint Server 2013 files on the file system).
    - Verify that the solution can be removed, added to, or upgraded when it is run multiple times.
    - Verify that uninstall scripts are provided to roll back to previous version.
6. Verify that the code runs correctly against the current production version of SharePoint.
7. Evaluate cross-site scripting vulnerabilities for all data input.
    - Make sure to use Server.HtmlEncode.
    - Test using scripts provided at the MSDN reference site Testing Your Web Applications for Cross-Site Scripting Vulnerabilities.
    - Reference: How To: Protect from Injection Attacks in ASP.NET.
8. Test for protection against SQL injection, especially if a custom database is used.
    - Reference: How To: Protect from SQL Injection in ASP.NET.
9. Test for security vulnerabilities.
10. Microsoft Code Analysis Tool .NET (CAT.NET) v1 CTP - 64 bit is a binary code analysis tool that helps identify common variants of prevailing vulnerabilities that can give rise to common attack vectors like cross-site scripting (XSS), SQL injection, and XPath injection. MSOCAF makes use of CAT.NET.

Additional Recommendations

Here are some guidelines that we recommend, but are not a basis for solution rejection:

- CSS, JavaScript, images, and styles should be on the hard disk, to use IIS compression (GZip).
- Caching should be used appropriately, invalidating the cache as appropriate.
- Verify that code is thread safe.
- Verify SQL queries for Enterprise Search, as described in this reference article:
    - Reference: Best Practices: Writing SQL Syntax Queries for Relevant Results in Enterprise Search

Test Environments

Test environments are different from development environments, and testing should take place in a farm environment (with two web front-end servers and separate servers for service applications and content databases) rather than a standalone environment. A test environment should mimic the SharePoint Online production environment, and should include one or more web applications, a copy of one or more SQL Server content databases, and at least two web front-end servers. To build the test environment, refer to the SharePoint Online Build Guide (available to current customers only, on the Customer Extranet site).

The goal in testing the custom solutions is to find any failure points (see next section) and test them. Speed is not important; in fact, for testing purposes memory should be limited, to see what happens. If an application passes in this type of environment, it probably doesn't have large memory requirements.

Failure Points

The following are likely causes of failure at some time in the life of the solution. They are considered by the SharePoint Online engineering team to be the most common failure points.

- **Network and infrastructure configuration.** The network configuration in the test environment is different from the SharePoint Online production environment. For example, the test configurations do not have correct trusts and forest structure, or the URLs are not configured for Secure Sockets Layer (SSL).
- **Altered computer configurations.** A developer configures his own computer, and then writes a solution to mimic those (altered) settings. Then the solution is installed in a clean environment where the developer has access. The failure point comes when the developer "fixes" an issue on his own computer, but the fix can't be duplicated in the clean environment.
- **Dependencies.** To minimize the impact on users, for any custom solutions that depend on external resources or on other custom solutions, you must code them to fail gracefully, when the resource or solution they depend on is unavailable.
- **The human element: variation.** There are numerous ways to deploy solutions, but the best course of action is to adhere to a process and deploy the same way every time, eliminating as far as possible any human variables. For example, when the same tasks must be assigned to different individuals, failure may result.
- **The human element: rushing.** Simple changes can be more complicated than they look. For example, you may make a very minor change, such as changing the name of a Web Part. However, due to time constraints this "minor text change" is not properly understood by the tester. Consequently, no testing is done, and the install fails. Why? The developer did remember to change the XML manifest file to point to the new Web Part name, but forgot to change the name in the Feature receiver. That's just one scenario; many similar things could occur.
- **Solution files are fragile.** A change in the code or even a change to the structure of the website can keep a solution from deploying properly. There is no developer tool that can show everything that the solution is going to do. The only thing you can do is test each change. It can be very tough to know whether the latest version of the solution is working or not, especially because developers rarely wipe their computers clean after each install. It is really the responsibility of your test teams to verify that the solution works—in an upgrade situation, for a fresh install, or in the production environment as it is configured today.
- **Multiple web front-end (WFE) servers.** The environment hosted by Microsoft contains multiple WFE servers. It is not uncommon that the code or deployment process will fail when multiple WFE servers are in a server farm. Make at least one pass against a farm configuration with two WFE servers.
- **Incomplete removal of previous versions.** A previous version may not be completely removed from development integration or a test environment, causing a number of false positives to be returned. Clean-up prior to the installation of a new build is critical to successful development on SharePoint Online. Also, a detailed list of steps for a complete retraction is necessary in the event of a failed install or failed smoke test.
- **Deployment scripts without adequate pauses for longer running operations.** The SharePoint Online environment consists of farms with multiple WFE servers. Because of this multiserver environment, it can take a while for timer jobs to complete. Dependent operations like retractions and deletions require a pause between them to allow for time to validate that prerequisite operations have completed.
- **Access design.** A solution may not be designed to work correctly in an environment with one-way trusts.

- **Authentication.** A solution may not be designed to work correctly with the authentication method (Windows Claims or SAML claims) that is used on the SharePoint web application.

Required Performance Tests

It is your responsibility to test custom solutions for performance, and to provide the test results to us as a part of the custom solution deployment package. You must conduct the performance testing according to the guidelines provided in this document.

To validate the impact of deploying the custom solution, you must compare a baseline of the performance of the system prior to the deployment of the custom solution with post-deployment performance. The tests must measure the impact on the SharePoint Servers, such as CPU and memory. To do this, you must collect the performance counters listed in step 2 below. The tests must also measure the performance of the page. This can be measured by capturing the time to last byte (TTLB) setting using a web test.

Step 1: Establish a Test Environment

The load testing environment should have the same caching configuration as the SharePoint environment, as described in the build document. This should include the following:

- Binary large object (BLOB) caching should be enabled.
- If the site has the publishing feature enabled, SharePoint Object caching should be enabled.

Step 2: Establish the Baseline with Initial Testing

Before testing the custom solution, establish a baseline by testing prior to deployment of the custom solution. This should include the following actions:

- Perform an IIS reset.
- Warm up the system by accessing and invoking the same functionality that will be tested.

The tests should access all the pages and functionality that are impacted by the custom solution prior to actually deploying them.

Run these tests using a variety of different user loads. At minimum, the tests should be performed at a targeted number of requests per second. Capture the following counters on the SharePoint WFE servers:

- Processor(_Total) \ percentage of processor time
- Process(w3wp) \ percentage of processor time
- ASP.NET \ Request Execution Time
- ASP.NET Apps v2.0.50727 \ requests per second
- Memory \ percentage of usage peak
- Memory \ available megabytes
- Time to last byte (TTLB) information for each page.

Step 3: Run the Tests with the Custom Solutions

Deploy the custom solutions to the farm and perform the same tests as in step 2 above.

Logging Requirements for Custom Solutions

It is critical that each custom solution provide a means of monitoring its health. For custom solutions that cannot be monitored, Microsoft may deem them to be unsupportable. Custom solutions should be designed to write errors to the event log only when action is required from the SharePoint Online

operations team. All other errors—those that contain only events that do not require any actions from the SharePoint Online operations team—must be written to the ULS log where possible.

> **Important**
> *Microsoft reserves the right to reject a custom solution that does not implement the appropriate instrumentation.*

If we find that a custom solution feature is adversely impacting a core service or service stability, we reserve the right to disable the custom solution. In such a case, Microsoft notifies you immediately after stability has been achieved or the core service is back online. To minimize the impact on users if we must disable a custom solution, you must code any interdependent custom solutions to fail gracefully.

Microsoft Responsibilities

- Evaluate whether the appropriate logging information is provided in the custom solution deployment package, and determine whether all necessary monitoring is included.
- If a custom solution adversely affects a core service or service stability in general, disable the custom solution and notify the customer.

Customer Responsibilities

- Ensure that the only times that the custom solution writes information to the event log are when the errors require involvement from the SharePoint Online operations team.
- Ensure that the custom solution writes all other information to the ULS log, to assist with assessing the health of the custom solution.

The Event Log

If the custom solution contains the ability to write to the event log to report errors or problems, Microsoft may configure Microsoft Operations Manager or Microsoft System Center Operations Manager to record and respond to the events using alerts. This configuration occurs during the same change window as the deployment. However, it is critical that the custom solution provide a means of monitoring its health. Without one, Microsoft is dependent on users reporting problems with the custom solution, and has no means to independently assess or measure the health of the custom solution.

All custom solutions from your development team must include and use a customer-specific event source in the event log, of the form *customer_log*, for example *contoso_log*. This event source can be created either with a Windows PowerShell script or in the code itself during feature activation.

From a Windows PowerShell script: A simple one-time cmdlet to create a new event log (**Customer_Log**) and new event source (**NewApp**) can be executed on each web front-end (WFE) and application server in the farm, as described in the New-EventLog Help topic in the TechNet Library. For example:

```
C:\PS>new-eventlog -source NewApp -logname Customer_Log
```

In custom solution code: Code used in the custom solution to activate the feature can create a new event log (**Customer_Log**) and new event source (**NewApp**). For example:

```
if (!System.Diagnostics.EventLog.SourceExists("NewApp"))
System.Diagnostics.EventLog.CreateEventSource("NewApp","Customer_Log");
```

In either case, an event source instance must be created to write to the event log. For example:

```
EventLog eventLog1 = new EventLog();
eventLog1.Log = "Customer_Log";
eventLog1.Source = "NewApp";
eventLog1.WriteEntry("This is a simple event log entry");
```

For further details about creating and writing to custom event logs, refer to these articles in the MSDN Library: How to: Create and Remove Custom Event Logs and How to: Write Entries to Event Logs. Ensure that new entries can be continuously written to the event log by setting OverflowAction to OverwriteOlder.

In the Monitoring section of the deployment guide, you can provide instructions for actions that must be taken in the event of a logged error. Proactive operational monitoring is critical, and any custom solution–related errors that will be monitored must be logged in the event log.

Third-party solutions can write to the Application event log, but must use unique IDs, and specify the source, category, type, and description of the event.

Important

The error strings must be adequately descriptive of the error that occurred.

Within the custom solution deployment package, the deployment guide must include the following information for all events:

- Event type
- Event source
- Event category
- Event ID
- Error string
- Description of error
- Impact of error

In addition, each event must have a troubleshooting article associated with it.

Unified Logging Service

For events logged to the Unified Logging Service (ULS) log, each custom solution must implement a uniquely named categorization using the SPDiagnosticsArea class in SharePoint 2013. This approach allows events from the custom solution to be grouped together or filtered.

SharePoint Online Configuration

The ways in which the configuration of the SharePoint Online environment can be changed are defined in Table 6 below. Whether a particular configuration is supported is shown in Table 7. If a configuration is not listed, please contact the SDM to learn whether it is supported.

Table 6. SharePoint Online Configurations

Configuration	Supported	Comment
Custom-managed path	No	SharePoint Online does not support custom-managed paths for site collections.
SharePoint database schema change	No	Changes directly to the database schema are not supportable.
SharePoint database data access	No	No access is provisioned directly to the SharePoint databases. All access to the databases is via the object model (OM) or web services.
Modifying built-in SharePoint files	No	SharePoint Online does not support any changes to the SharePoint files on the file system. This includes shipped site definitions.
Web services access	Yes	Almost all web services that ship with SharePoint Server are exposed. For more information, see Using SharePoint Web Services earlier in this document.
WCF Data Services access	Yes	All WCF Data Services (formerly known as ADO.NET Data Services) that ship with SharePoint Server are exposed.
Silverlight Web Part	Yes	SharePoint 2013 includes an out-of-the-box Silverlight Web Part.
SharePoint Designer editing	Yes	Any changes that can be made with SharePoint Designer 2013 (SharePoint Designer 2010 is not compatible with SharePoint 2013) are supported.
Changing mapped host name after deployment	No	SharePoint Online does not support changing the hosted domain name—for example, sharepoint.contoso.com—after the environment has been deployed. The customer should carefully choose the host name of its environment before service adoption.
Visual Studio installation on deployment	No	Microsoft does not allow Visual Studio to be installed on a web front-end (WFE) server for troubleshooting.
Installation of Microsoft Office client on server	No	Microsoft does not allow Microsoft Office clients to be installed on a WFE server.

Configuration	Supported	Comment
Publishing of service applications for cross-farm consumption	No	Microsoft does not publish service applications for consumption from remote farms.
Consuming on-premises service applications	No	SharePoint Online does not support service application proxies that would connect to on-premises service applications. The one exception to this is the FAST Search Query Search Service Application.
Using RBS Providers	No	Microsoft does not allow a remote BLOB store (RBS) provider to move the storage of large binary data (BLOBs) from SQL Server databases in the SharePoint Online environment to a commodity storage solution.

SharePoint Online Customization Types

Table 7 describes the various ways that SharePoint can be customized and whether or not they are supported within the SharePoint Online environment. These apply to SharePoint 2013 . Non-supported types are either planned for a later release of the SharePoint Online service or have been determined at this time to pose a risk to the environment, add additional operational overhead, or be potentially difficult to upgrade to future versions of SharePoint Online.

Table 7. Support for Custom Solution Types

Customization Type	Supported	Comment
SharePoint Feature	Yes	A SharePoint Server Feature is a server-side custom solution at the file system level. It can contain modular items that can be installed and activated in a SharePoint Online environment.
Feature stapling	Yes	Feature stapling, also known as Feature site template associations, is the ability for a Feature to be attached to all new instances of sites that use a given site definition, without modifying the site definition or creating code routines to activate the feature on each site.
Feature event receiver	Yes	A Feature event receiver, or Feature receiver, is a server-side code routine that is called as part of certain key events in the lifetime of a Feature, such as installation, activation, deactivation, or removal.

Customization Type	Supported	Comment
Web application	No	A web application is code or content installed in a new web application directory, or as a subdirectory under an existing web application. It provides functionality that SharePoint technology does not. Supporting additional web applications poses a burden on the SharePoint Online operations team.
Web service	Yes	A web service is server-side code that exposes code routines to be called from remote systems. The web service must be hosted in one of the deployed web applications.
Windows Server service	No	The Windows Server® service is installed on the server to perform major actions. It runs in a separate process space from the SharePoint Server instance. A service has a relatively high cost for deployment and manageability.
SharePoint timer job	Yes	A timer job is a built-in type of SharePoint Server object that can perform various tasks within the SharePoint Online environment on a scheduled or one-time event basis.
Scheduled task	No	A scheduled task permits the running of code on the Windows-based server on a defined schedule. On event occurrence, the code is run. There is no trigger from within SharePoint Server. Scheduled tasks are not supported, because they run outside of SharePoint Server.
Web Part	Yes	A Web Part is a modular component that is used within the SharePoint environment to perform activities or display information.
Event handler	Yes	An event handler is a web application scoped feature that contains server-side code routines. It is called when registered events occur within the SharePoint environment. Custom event handlers can be attached to site collections, sites, lists, or document libraries.
Backward-compatible event handler	No	A backward-compatible event handler is a server-side code routine that is called when a registered event occurs within a SharePoint Server document library. Backward-compatible event handlers can only be attached to and have a scope of a document library.

Customization Type	Supported	Comment
Custom site definition	No	A site definition is a server-side collection of files that defines the structure of one or more site templates. When a site is created with a custom site definition, it is dependent on that site definition in the future, including when upgrading to a new version of SharePoint Server.
Site template	Yes	A site template is a package that contains a customized site design based on an existing site definition. This package is typically created by using the Save Site As Template functionality.
Web template	Yes	A web template is similar to a site template, but it is typically created by developers. For more information, see Web Templates and this post on MSDN. A web template persists in the SharePoint database as a solution file, with a .wsp extension. The template is stored in the Solutions gallery of the site collection. A web template can also be deployed as a sandboxed solution.
List definition	Yes	A list definition provides the schema for a SharePoint Server list.
List template	Yes	A list template is a package that contains a set of custom solutions from a base list structure. A list template should not be confused with a list definition.
Field type	Yes	A field type is an extensible method for allowing new common data structures within SharePoint Server for use in lists or other locations that can use SPField objects.
Master page	Yes	A master page defines the layout, fonts, and formatting for the pages that are associated to it.
Application Master Page	Yes	An application master page defines the layout, fonts, and formatting for the administration pages that are associated to it.
Page layouts	Yes	A page layout is a page template that controls how the page looks and exactly which elements (such as lists and libraries) should be present on the page. The publishing page layout can be attached to a master page that defines its layout and formatting.

Customization Type	Supported	Comment
Custom application page mapping	Yes	Custom application page mapping functionality allows for substitution of specific application pages used by SharePoint. For more information, see SPWebApplication.UpdateMappedPage in the MSDN Library.
_layouts page	Yes	A _layouts page is any page stored in the _layouts virtual directory within SharePoint Server.
Custom CSS file	Yes	A Cascading Style Sheet (CSS) file is used to format the contents of web pages. It contains style rules that define how to display HTML elements.
SharePoint theme	Yes	A SharePoint Server theme is a collection of graphics and cascading style sheets that can modify how a website looks. Add a manual rule to ensure that all solutions are only targeted for O15 site collections.
Content type	Yes	A content type is a reusable collection of settings that can be applied to a certain category of content. Content types enable the management of metadata and behaviors of a document, item, or folder type in a centralized, reusable way.
Column template (column definition)	Yes	A site column, also known as a field, is a reusable column definition, or template, that can be assigned to multiple lists across multiple SharePoint sites. Site columns decrease duplication of work and help ensure consistency of metadata across sites and lists.
Delegate control	Yes	A delegate control allows certain predefined elements of the SharePoint environment to be replaced (like a field control) by other controls through SharePoint Features.
Form template	Yes	A form template is a user-control–based (ASCX-based) control template that brings together multiple controls into a nested structure, where each control consists of templates and can be used to richly extend list item forms.
Custom action	Yes	A custom action is an item (command) that is added to the user interface.
Coded workflow	Yes	A coded workflow is a Windows Workflow Foundation workflow that references an assembly for the workflow pipeline.

Customization Type	Supported	Comment
No-code workflow (declarative workflow)	Yes	A no-code workflow, also known as a declarative workflow, is a sequential workflow pipeline that is configured on a list or document library. It requires no installation of server-side code.
Workflow activity	Yes	A workflow activity is a compiled class that is used as a part of the modular steps in an Azure Workflow. An activity is used as part of an Azure Workflow action that will be run. Azure Workflow activities run as server-side code when used as activities within the SharePoint environment.
Workflow condition	Yes	A workflow condition is a compiled class that is used as a part of the modular steps in a workflow. A condition is used to determine when a workflow action will be run. Workflow conditions run as server-side code when used as conditions within the SharePoint environment.
Solutions that customize or change the Central Administration web application	No	Central Administration (CA) is currently not provisioned on all farm WFE servers as a best practice. In the event that a custom solution is deployed to CA, it is deployed only to the computers where CA is provisioned. This results in a situation where, if there is the need to provision a new CA on a different server on the farm, the custom solution is not deployed to the new server. This causes the features using the solution to malfunction.
Document icon	Yes	A document icon is a graphical item that is used to represent corresponding documents that are exposed within document libraries. Each preconfigured document type in SharePoint has a corresponding document icon entry in a server configuration file, but other file formats may lack appropriate icons. Document icons can only be modified by submitting a CR through the SDM.

Customization Type	Supported	Comment
iFilter or Connector (protocol handler)	Yes	An iFilter is a server-side component that is used by the indexing system to index documents that are identified as having a file format that is associated with the iFilter. Custom iFilters require deployment to all servers in the environment that do indexing; they also require changes to mapped properties for search. In previous SharePoint products connectors were known as protocol handlers. A connector enables indexing applications like SharePoint Server to systematically crawl the nodes of a data store to extract relevant information to include in the index. Each connector is used to index a specific type of data store. For a list of default connectors, please see Default connectors in SharePoint Server 2013.
Document converter	No	A document converter is a custom executable file that takes a document of one file type, and generates a copy of that file in another file type.
Information management policy	Yes	An information management policy is a set of rules for a certain type of important content. Policy enables administrators to control and evaluate who can access the information, how long to retain information, and how effectively people are complying with the policy itself.
External content types	Yes (unless a new SSO account is required)	External content types are reusable metadata descriptions of connectivity information and data definitions, plus the behaviors you want to apply to a certain category of external data in your SharePoint 2013 environment.
Excel user-defined function (UDF)	Yes	A user-defined function (UDF) is server-side managed code, run by Microsoft Excel® Calculation Services to provide capabilities that are not included in the base product. These include: • Functions that are not built into Excel. • Custom implementations to built-in functions. • Custom data feeds for legacy or unsupported data sources. • Application-specific data flows.

Customization Type	Supported	Comment
InfoPath Forms Services custom code	Yes	Microsoft InfoPath® Forms Services custom code is uploaded and hosted by the server-side InfoPath Forms Services. **Note:** These InfoPath forms are considered custom solutions and must go through the custom solution submission and validation process. The code-behind .dll must be extracted from the .xsn template file, so it can be successfully analyzed using MSOCAF.
InfoPath form view control	Yes	An InfoPath form view control is a server-side InfoPath Forms Services viewer control that can display a web browser view of an InfoPath form.
HTTP handler	No (only supported if not used for authentication)	An HTTP handler is used to handle file requests within the ASP.NET 3.5 framework. For example, within ASP.NET 3.5 and SharePoint Server 2013, ASPX page requests and ASMX web service requests are served by HTTP handlers.
HTTP module	No	An HTTP module is a class that handles runtime events in ASP.NET 3.5 and SharePoint Server 2013.
Pluggable authentication provider	No	A pluggable authentication provider is a module used by ASP.NET 3.5 and SharePoint Server to replace the built-in authentication providers. Custom modules require a high level of testing to help ensure security.
Pluggable single sign-on (SSO) provider	No	A pluggable SSO provider is a modular component that can be used to handle the storage and mapping of credentials for use in connecting with third-party or back-end systems. This custom solution type requires a high level of security review and scrutiny.
STSADM command extensions	No	A custom STSADM command extension is server-side code that allows the Stsadm utility to have new commands available to run from the command line.
Console applications	No	A console application must be run manually by the SharePoint Online operations team.
Inline code	No	Inline code is server-side code that is directly embedded into a web page. Inline code has a high level of risk compared to compiled assemblies.

Customization Type	Supported	Comment
Web.config settings change	Yes	The web.config file controls most settings for the ASP.NET environment that SharePoint Server 2013 is built on. Any modifications to web.config settings *must* be added or removed using the SPWebApplication class and SPWebConfigModifications property, so that they will be applied on all the WFE servers. Manual modification of the web.config settings is *not* supported.
Custom security policy	Yes	A custom security policy grants code access security (CAS) rights to certain server-side code components that run under a specific web application context.
COM server	No	A component object model (COM) server is an unmanaged library that typically runs on a server with full permissions. COM objects typically perform poorly when interops are set up between the COM object and .NET code.
Sandboxed solution	Yes	A sandboxed solution is a solution file that can be uploaded to individual site collections by site collection administrators.
External data lists	Yes	In SharePoint 2013, the external data list type is used for displaying content that comes from Business Connectivity Services (BCS) Enterprise content types.
Silverlight client object model customization	Yes	The Silverlight client object model is a client-side solution that runs in the browser; SharePoint 2013 provides support for writing such customizations.
Service application	No	A service application is a service that can be consumed by multiple web applications. It is managed via the Central Administration site.
Excel JavaScript object model (JSOM)	Yes	The Excel JavaScript object model enables client-side JavaScript scripts to run in the browser and utilize services provided by SharePoint 2013.
Ribbon extensions	Yes	The Ribbon is the new user interface component in SharePoint 2013 that replaces traditional toolbars.
Solutions using the dialog box platform	Yes	SharePoint 2013 provides a new programming interface that uses dialog boxes to show information and interact with the user.

Customization Type	Supported	Comment
Client object model	Yes	A client object model is a programming interface that can be called from client-side solutions. SharePoint 2013 provides: Silverlight, ECMAScript, and Microsoft .NET–managed.
REST	Yes	A representational state transfer (REST) is a form of interface that can be used from custom applications.
User profile exports	No	SharePoint Online does not support user profile exports.
Managed metadata	Yes	SharePoint 2013 provides a model for centrally managing metadata.
Conversion services	No	SharePoint 2013 provides interfaces for data conversions.
Claims provider	Yes (claims augmentation only)	A claims provider is a software component or service that issues claims and packages claims into security tokens. Your claims provider solution must deploy the assemblies and all of its dependencies to the global assembly cache (GAC) of all server roles: web front end and application server. Failure to do so can result in service outages. For more information about avoiding issues with custom claims providers, see Learning About Claims-Based Authentication in SharePoint 2010 in the MSDN Library.
Windows PowerShell	No (except where MSOCAF functionality is not supported)	SharePoint Online does not support running manual Windows PowerShell™ scripts in the hosted environment.

Custom Solutions That Require Database Services

When developing solutions, you may need to have database services provided for your custom solutions. If a custom solution requires database services, the following options are available:

- **Host the custom database on premises and access it via Business Connectivity Services (BCS).** In this scenario the custom database is hosted on premises in the customer's data center. The custom solution makes use of BCS to connect to and consume the database services.
- **Host the database solution on the Microsoft SQL Azure™ technology platform.** In this scenario the custom database is hosted on SQL Azure by Microsoft. The custom solution deployed to SharePoint Online connects to the cloud-based databases. For more information, see SQL Azure Support in Office 365 Dedicated Plans later in this document.

Note

The SQL Azure services are not a part of the SharePoint Online contract, and must be acquired separately.

- **Host the custom database in the Microsoft data center.** SharePoint Online supports the use of a SQL Server database, following the guidelines described in Table 8. If a database custom solution item is not listed in the table, do not assume that it is therefore allowed. Contact the SDM for further information. SharePoint Online supports only one custom database to be utilized by all custom solutions developed by the customer. One additional custom database is allowed for each approved third-party custom solution that requires a custom database.

Note

Databases that are resource-intensive—that have high storage, CPU, connections, and memory requirements—cannot be hosted. You have the option to host a resource-intensive SQL Server database on your own servers in your own data centers.

Table 8. Custom Database Support Guidelines

Database Custom Solution	Supported	Comment
SQL Server database	Yes	The Online Services infrastructure uses SQL Server hosted on the existing SharePoint Online infrastructure. The current versions supported by Online Services are SQL Server 2008 R2 for SharePoint 2010 farms .
Custom non-SQL Server database	No	SharePoint Online does not support database servers other than SQL Server. You can choose to host such databases on their own premises and integrate them with SharePoint Online using the WCF interface with a web service.

Database Custom Solution	Supported	Comment
More than one custom SQL Server database	No (with exception for ISV)	SharePoint Online hosts a single SQL Server database for customers on the shared SQL Server infrastructure. Each customer that wants to deploy its own custom solutions is allowed one custom database for that purpose. One additional custom database is allowed for each approved third-party custom solution that requires a custom database.
Direct database access	Yes	Ensure that proper security measures are in place. Applications that make direct access to on-premises **or cloud-based** databases using ADO.NET should design for highly latent connections.
Custom-developed Web Parts/SharePoint Online pages with custom database integration	Yes	Custom applications must be isolated from out-of-the-box features, and should be running as an independent entity. Failure of the custom database access should not cause the entire farm to fail. No standard existing functionality (for example, Search and workflows) should fail due to a failure on a custom database. AJAX UI implementation is recommended when connecting to a custom database to minimize page load impact if the custom database connection becomes slow.
Custom database as an authentication or credential store	No	The database cannot be used as an authentication store. No credentials are allowed to be stored.
Access to SQL Server	No	Customer access to the SQL Server database using the SQL Server client tools is not permitted under any circumstances, even during troubleshooting.
Backups of the custom SQL Server database	No	Microsoft cannot provide a copy of a customer's custom SQL Server database.
Installation of a copy of a SQL Server database or scripting the creation of a custom SQL Server database	Yes	The creation of the custom SQL Server database within the hosted SharePoint Online environment can be done by providing Microsoft with a copy of the SQL Server database (not populated) or through the use of scripting to create the database and associated schema in a Feature receiver.

Database Custom Solution	Supported	Comment
Upgrade path	Yes	All databases hosted by Microsoft are subject to the same upgrade strategy as the SQL Server product offering. Customers will be notified of plans to upgrade the SQL Server build or version in advance (with six weeks' notice) of the actual exercise and are responsible for testing that their custom database and the functionality they are using will work with the new build or version. As of the writing of this document, SharePoint Online supports only SQL Server 2008 R2 for SharePoint2010 .
SQL Server Reporting Services Web Parts (Restricted to pre-SP2 SQL Server 2005 version)	No	The SQL Server team has released Web Parts to display data within a Reporting Services system. However, only pre-SP2 versions of SQL Server 2000 and SQL Server 2005 Reporting Services Web Parts can work within a hosted environment. The current design of the Reporting Services Web Parts requires the Reporting Services server to be hosted within the same domain as the SharePoint Online environment. Customers who want to use the shipped Reporting Services Web Parts should know that SharePoint Online supports only the pre-SP2 version of SQL Server 2005.

SQL Azure Support in Office 365 Dedicated Plans

Office 365 dedicated plan customers can use SQL Azure in their SharePoint Online farm with certain restrictions. The following guidelines must be followed when using SQL Azure:

- SQL Azure can be used with SharePoint Business Connectivity Services (BCS) to create a Business Data Connectivity (BDC) application. A BDC application can be created in SharePoint Designer to generate an external content type (ECT) for the SQL Azure data. SharePoint Designer can also create a new list based on the ECT, which will automatically render the data in the site. Note that creating the BDC application requires you to create a Secure Store Service application definition to encrypt and store the credentials used to connect to SQL Azure. Credentials used to access any SQL Azure database must always be encrypted. For a complete walk-through on using SharePoint Designer to create a BDC application to SQL Azure and using the Secure Store Service to encrypt the credentials, see Creating an External Content Type Based on SQL Azure in MSDN.
- Any components that request data asynchronously from a client application that is hosted in a SharePoint site, such as a Silverlight control, must **only** retrieve data from the list based on the ECT. We do not support storing connection credentials in a control that is hosted on a client computer, because that assembly can be disassembled and the credentials retrieved.
- Components that do not run in a SharePoint page, such as a custom timer job, can make a direct connection to SQL Azure to retrieve data. However, as stated above, the credentials for connecting to SQL Azure must be encrypted and stored in the Secure Store Service. The application can read the credentials out of the Secure Store Service and use that to create the connection string to retrieve data.

- Components that run on the server and in a SharePoint page are **only** allowed to make connections and request data from SQL Azure if the results are cached using a server cache mechanism, such as the ASP.NET Server cache. Queries to SQL Azure are frequently much more latent than queries to a local instance of SQL Server, which is why caching is required. Those requests must also meet the following requirements:
    - Data must be cached a minimum of 20 minutes between requests.
    - Requests must use a lock() pattern when refreshing the cache to ensure that there is not contention for resources or multiple queries executed for the same content.
- All connection strings to SQL Azure must use secure connections, as described in How to: Connect to SQL Azure Using ADO.NET in the MSDN Library. These are the important points to remember:
    - You must include Encrypt = True and TrustServerCertificate = False in the connection string.
    - Your connection timeout should be at least 30 seconds.
- There is one additional configuration change needed to allow connections to SQL Azure from the SharePoint Online farm:
    - You must get the external IP address for your production farm and, optionally, PPE environments, and add them to the list of allowed IP addresses in the SQL Azure firewall. You can contact the SDM or service manager to open a request to get the IP addresses you should use.

Access Custom Database

If your custom solution accesses a custom database, use the Secure Store Service to configure a non-expiring Windows Service account to be stored in encrypted form. Here is an overview of the process for accessing custom databases:

1. Create a non-expiring Windows Service account in the same domain that is trusted by the SharePoint Online farm and that can access the custom database.
2. Grant the account specific rights to the custom database.
3. Create a Secure Store target application, and configure the Windows account as documented in Use Secure Store with SQL Server Authentication in the TechNet Library. In the article, look for these two procedures:
    - To create a target application for SQL Server authentication
    - To set credentials for the target application
4. In your code, impersonate the user in the Secure Store Service to access the database.

Non-Supported SQL Server Features

SharePoint Online supports any standard SQL Server features that are used in development practices on SQL Server, except the following:

- Integration Services (or any SSIS packages, DTS, BCP)
- Reporting Services
- Analysis Services
- Notification Services

- Replication

All SQL Server features that are required by the custom design should be detailed in the HLD document for review and approval by Microsoft. Microsoft provides feedback if a feature that the customer plans to use is not supportable.

Non-standard features that are also not supported include:

- SQL Server 2008 transparent data encryption
- SQL Server 2008 data encryption
- SQL Server data replication
- Linked server configuration
- Cross-database chaining
- C2 audit tracing
- Common criteria compliance
- Server proxy account
- Database Mail or SQL Mail
- Extended stored procedures (XPs)
- Common language runtime (CLR) integration
- Ad hoc remote queries (OPENROWSET AND OPENDATASOURCE support)
- Native XML web service (HTTP endpoints)
- Custom OLE automation
- Service broker or message queuing
- Web Assistant (as deprecated in SQL 2005)
- xp_cmdshell
- Custom SQL Server Agent jobs (maintenance jobs already inclusive of the hosting system)
- Trace flags
- Lightweight pooling

Terms and Conditions

The terms and conditions for hosting a single database for a custom solution are described in this section. If a condition is not listed, this does not mean it is automatically allowed. Contact the SDM for further information.

Database Configuration

Microsoft enforces restrictions on each database. The database cannot grow above 250 GB for SharePoint 2013. The database is configured as follows:

- Full recovery model.
- Compatibility mode: 100 (SQL Server 2008)
- Database ownership is set to sa.
- Database access level for application is set to read/write only.
- A single MDF file and a single LDF file.
- MDF and LDF data files are located in dedicated Data and Log locations respectively. For example, data files (.mdf) should be placed on H:\MSSQL\Data and log files (.ldf) on O:\MSSQL\DATA.
- Data files are initialized at 20 GB for MDF and 5 GB for LDF, with auto-growth set for 200 MB for both file types. Data files are restricted to a maximum size of 80 GB for MDF and 20 GB for LDF.
- Remote query timeout is set to 600 seconds.
- *Not provided:* Full-text search is not enabled, and not allowed when co-hosted on the SharePoint Online back end.

Security Model

- Integrated Windows Authentication only.
- *Not allowed:* SQL Server standard logons.
- *Not provided:* SQL Server manages the sa password, which will not be provided (disabled).

Database Maintenance

All custom databases will have a maintenance window that is similar to that for SharePoint Online databases:

- Weekly full backup
- Daily differential backup
- 15-minute transaction log backup, with file retention of 21 days

High Availability Protection

- Database is mirrored (high-protection mode) within local data center.
- Database is log-shipped to remote data center for disaster recovery.

Connectivity and Access

- SQL Server connectivity using TCP/IP protocol only.
- Access to SharePoint Online custom database via ADO.NET only; no direct access to server or database via SQL Server Management Studio or the OSQL utility.
- Database access level for application is set to Read/Write only.
- *Not provided:* View server state.
- *Not allowed:* the commands create, delete, detach, attach, backup database.

Support

- Custom databases must meet acceptable levels of performance that do not impact SharePoint operations.
- Transaction per second < 100.
- Reads per transaction (one RPC call) < 100,000.
- *Not provided:* Archive data to tape for long-term retention.

Note

Microsoft reserves the right to implement indexes to improve performance of the database if server performance is impacted by the hosted database.

Using Business Connectivity Services

Business Connectivity Services (BCS) is an integration feature of SharePoint 2010 and SharePoint 2013 that provides read and write access to external data in LOB systems, web services, and other external systems. Once an external system has been modeled in BCS, a SharePoint user or developer can create solutions that use external data directly without having to possess expert knowledge about the API of the back-end system. For a more detailed description, see Business Connectivity Services overview (SharePoint Server 2010) in the TechNet Library.

Microsoft recommends using Microsoft SharePoint Designer 2010 or SharePoint Designer 2013 to create "code-free" BCS solutions managed by the customer, rather than full trust code solutions managed by Microsoft. This method allows for quicker deployment and easier maintenance.

For more information, see Microsoft Business Connectivity Services Model in the Microsoft Download Center.

The following table lists the CR templates in the order in which they must be submitted.

CR Template	Purpose	Comments
SPOD-051 (2010/2013) Import BCS Model	Upload the BCS model to the BCS Metadata Store	Provides the mapping between SharePoint and the external LOB system, and describes the entities and behaviors of the data source
SPOD-054 (2010/2013) Set BCS Model Permissions	Identifies the security group that you will add to the **Members** field on SPOD-10-168. This template is also used to assign permission to a BCS Model. Users must be assigned permission to access this model.	This group must be created so the users can access the model.
SPOD-168 (2010/2013) Create or Modify a Target Application	Creates a new target application.	You can define the administrators of the target application. The **Member** group must be provided. This group uses the credentials set in the target application. Depending on the model's needs, multiple target applications might need to be created. You must submit a single CR template for each target application needed.

BCS Solution Authentication Methods

The following table contains information about supported authentication methods for BCS solutions. For more information, see Available Authentication Modes in the MSDN Library.

Table 9 BCS Authentication

Authentication Method	Description	Supported?
Pass-Through Authentication	Passes the credentials of the logged-on user to the external system. This requires that the user's credentials are known to the external system. Note: If the Web application is not configured to authenticate with Windows credentials, the NT Authority/Anonymous Logon account is passed to the external system rather than the user's credentials. This mode is called User's Identity in the BCS administration pages and in SharePoint Designer 2010 or SharePoint Designer 2013.	
RevertToSelf Authentication	When the user is accessing external data from a web browser, this mode ignores the user's credentials and sends the application pool identity account under which the BSC runtime is running on the web server to the external system. When the user is accessing external data from an Office client application, this mode is equivalent to PassThrough mode, because BCS running on the client will be running under the user's credentials. This mode is called BDC Identity in the BCS administration pages and in SharePoint Designer 2010 or SharePoint Designer 2013. Notes: By default, BDC Identity mode is not enabled. You must use Windows PowerShell to enable BDC Identity mode. BDC Identity mode is not supported in hosted environments.	
WindowsCredentials Authentication	For external web services or databases, this mode uses a Secure Store Service to map the user's credentials to a set of Windows credentials on the external system. This mode is called Impersonate Windows Identity in the BCS administration pages and in SharePoint Designer 2010 or SharePoint Designer 2013. For more information, see Configure the Secure Store Service (SharePoint Server 2013) and Plan the Secure Store Service (SharePoint Server 2013) in the TechNet Library.	X
Credentials Authentication	For an external web service, this mode uses a Secure Store Service to map the user's credentials to a set of credentials that are supplied by a source other than Windows and that are used to access external data. The web service should use basic or digest authentication when this mode is used. **Important**: To help preserve security in this mode, Microsoft recommends that the connection between BCS and the external system should be secured by using Secure Sockets Layer (SSL) or Internet Protocol Security (IPSec). This mode is called Impersonate Custom Identity in BCS administration pages and in SharePoint Designer 2010 or SharePoint Designer 2013.	X

Authentication Method	Description	Supported?
DigestCredentials Authentication	For a WCF web service, this mode uses a Secure Store Service to map the user's credentials to a set of credentials using Digest authentication. This mode is called Impersonate Custom Identity – Digest in BCS administration pages and in SharePoint Designer 2010 or SharePoint Designer 2013.	?

Customer Responsibilities

The customer must provide the following information to Microsoft to deploy a BCS solution.

- If connecting to a SQL Server database on the customer premises, hosted by a third party, or in the cloud, the customer must provide:
    - Fully qualified name to SQL Server.
    - IP and port used.
    - Active Directory group name and credentials used to connect to SQL Server.
- If connecting to a web service, the customer must provide:
    - Notification of whether WCF or SOAP-based web services are to be used.
    - URL to WSDL file.
    - URI to the service.
- Provide relevant accounts and groups:
    - Content Access Account – Account used to access LOB system.
    - User Permissions – Active Directory group containing subset of users or All Authenticated Users.
    - Information Worker (IWs)/Developer Permissions - AD group containing IWs/ Developers that will create and save ECTs.

Writing Sandboxed Solutions

SharePoint sandboxed solutions are deprecated in SharePoint 2013 in favor of developing apps for SharePoint, but sandboxed solutions can still be installed to site collections on SharePoint 2013.

SharePoint 2010 introduced a new type of customization called a sandboxed solution, which uses a restricted set of classes from the SharePoint object model. You can deploy this type of solution to individual site collections without requiring a review by Microsoft. Sandboxed solutions represent both a faster deployment path and the opportunity for decentralized customization of individual sites, without the need for centralized ownership or cooperation of development teams. In those cases where a solution must be deployed as full trust, a sandboxed solution would not be applicable.

Sandboxed solutions are packaged as solution files (with a .wsp file extension) that contain assemblies, other non-compiled components, and an XML manifest file. A site collection administrator, or another user with sufficient permissions, can upload the solution package to the solution gallery in the root site of

the site collection. All sandboxed solutions are executed in a unique application domain. Because of this, SharePoint 2010 can monitor the performance and resource use of the sandboxed solutions according to various measurements, such as CPU execution time, memory consumption, and unhandled exceptions.

These measurements are weighted and combined into a single measure named resource points. Microsoft has set absolute limits, detailed in Table 10, for the number of resource points that sandboxed solutions can consume on a per-site collection basis. The available resources are effectively shared among all sandboxed applications in the site collection. Resource usage of the deployed sandboxed solutions is visible at all times to the site collection administrator in the Solution Gallery. All sandboxed solutions in a site collection will be terminated if they collectively exceed the set limits. Refer to the SharePoint Online Custom Solution Policies and Process document for Microsoft policies on termination of sandboxed solutions.

Table 10. Monitored Metrics for Sandboxed Solutions

Resource	Description	Units	Resources per Point	Absolute Limit (in Points)
AbnormalProcessTerminationCount	Abnormally terminated process	occurrence	0.25	1
CPUExecutionTime	CPU execution time for site	seconds	100	60
CriticalExceptionCount	Critical exception events	events	10	3
InvocationCount	Solution invocation events	events	100	TBD
PercentProcessorTime	Percent CPU usage by solution	percentage	85	100
ProcessCPUCycles	Solution CPU cycles	cycles	40000000000	1 x 10^11
ProcessHandleCount	Windows handles count	items	10000	5,000
ProcessIOBytes	Windows handles count	items	10000000	1 x 10^8
ProcessThreadCount	Thread count in overall process	instances	10000	200
ProcessVirtualBytes	Memory consumed	bytes	1000000000	1.0 x 10^9
SharePointDatabaseQueryCount	Number of SharePoint database queries	instances	400	100
SharePointDatabaseQueryTime	Elapsed time to execute query	seconds	20	60
UnhandledExceptionCount	Number of unhandled exceptions	instances	25	3

Resource	Description	Units	Resources per Point	Absolute Limit (in Points)
UnresponsiveProcessCount	Number of unresponsive processes	instances	2	1

Hybrid Solutions

In cases where a sandboxed solution is too limited, it is possible to extend the solution by providing mechanisms that allow the solution to call components that are deployed to the SharePoint environment. These mechanisms enable the developer to expose additional SharePoint functionality, to be used from within a sandboxed solution, while maintaining many of the benefits of the sandbox environment. Hybrid solutions are still preferable to complete full trust solutions, because only the full trust portions of the application need to be reviewed by Microsoft. After the full trust components are deployed, they can also be reused by sandbox and declarative solutions later.

There are three different types of full-trust components that can be used from within a sandboxed solution.

- **Full-trust proxies.** You implement a full-trust proxy in a class that derives from the SPProxyOperation abstract class. That assembly must then be packaged in a solution file and deployed to the global assembly cache in the SharePoint farm. This class then exposes a full-trust proxy that can be called from within the sandboxed environment.

  Note
  Customers must follow the full custom solution review process when developing and deploying a full-trust proxy.

- **External content types.** You can use an external content type to retrieve data from LOB applications, and other external sources, through Business Connectivity Services (BCS).

  Note
  SharePoint Online does not support external content types that require definition of a new single sign-on (SSO) account for accessing the LOB system.

- **Custom workflow activities.** The declarative workflows within a sandbox environment can use full-trust code-based workflow activities that are deployed to the global assembly cache in the SharePoint farm.

Appendix A: SharePoint Developer Resources

This appendix offers links to content that may be useful to developers of custom code for SharePoint Online. It also presents a discussion of recommended guidelines for developers.

Getting Started

The following resources provide a good introduction to the SharePoint platform.

SharePoint 2013

Developing Apps for SharePoint

Choosing the right set of API for SharePoint 2013

Learn about the several sets of APIs that are provided in SharePoint 2013 Preview, including the server object model and the various client object models, and the REST/OData web service.

SharePoint Developer Center

Primary MSDN site for SharePoint developers.

SharePoint 2013 Web Services

Web Services available for remote development.

SharePoint 2013 Administration Toolkit

Details about the SharePoint Administration Toolkit 1.0.

SharePoint Server 2013 SDK

The software development kit for SharePoint Server 2013.

Capacity Management for SharePoint Server 2013

Resources to help with capacity management.

SharePoint Server 2013 capacity management: Software boundaries and limits

Software boundaries that cannot be exceeded by design and thresholds that can be exceeded to accommodate specific requirements.

Claims-Based Authentication Resources

There are a number of resources available to learn about claims authentication in general, how it's used in SharePoint 2010, and how to develop claims-aware applications. This list focuses on topics that are most likely to be useful when doing custom claims development for the SharePoint Online environment.

- Configuring SharePoint 2010 and Active Directory Federation Services v2 End to End
- Configure authentication using a SAML security token
- Claims Walkthrough: Writing Claims Providers for SharePoint 2010
- Incoming Claims: Signing into SharePoint
- The Share-n-Dipity Blog

Appendix B: Testing Resources

Testing resources and testing guidelines are identified in this appendix. It also presents a discussion of best practices for testers.

Online Resources

Checklist for Testing SharePoint Web Parts

A 2003 checklist that largely still applies.

Testing Your Web Applications for Cross-Site Scripting Vulnerabilities

Recommendations about how to handle testing XSS vulnerabilities.

Load Testing and Performance Resources

Chapter 17 – Load-Testing Web Applications (MSDN Patterns and Practices)

General introduction about how to load-test a web application.

Load Testing Kit (SharePoint Server 2010)

Overview and how-to steps for the Load Testing Kit.

MSDN Webcast: Effective Web and Load Testing in Visual Studio Team System

Webcast developer session about using Visual Studio Team System to load-test web applications.

Estimate performance and capacity requirements for portal collaboration environments

A usage profile example for load testing purposes.

How to integrate SP Dispose Check into Visual Studio Solutions

A blog entry that outlines the steps necessary to integrate SP Dispose Check into the integrated Visual Studio runtime environment.

SharePoint Server 2010 capacity management: Software boundaries and limits

This document describes software boundaries and limits of Microsoft SharePoint Server 2010.

Appendix C: Custom Solution Deployment Package Checklist

Use this checklist to make sure you have included all necessary documentation in the deployment package before submitting it through MSOCAF. This documentation is required for Microsoft review and validation of the custom solution. MSOCAF validates the structure of the deployment package and checks for all required files, as described in Required Directory Structure for the Custom Solution Deployment Package earlier in this document. The package must be complete and self-contained without references to separate files, documents, or email messages; otherwise the entire package may be rejected.

General Instructions

- ☐ Use the most current deployment guide template (available to current customers only, on the Customer Extranet site) to indicate the requirements, planned design, and implementation details for the custom solution.
- ☐ Prior to submission, ensure that all documents (including embedded documents, if any) open without errors.
- ☐ Outline what you plan to test and validate (including memory usage, scale, and functionality). It is important that you have validated any third-party products prior to providing the deployment guide to Microsoft.

Deployment Guide (Installation Instructions)

The deployment guide must include:

- ☐ Detailed instructions for Microsoft to follow when deploying a custom solution, as well as documentation of any unique issues regarding deployment that Microsoft should be aware of. Deployment activities that are automated using the MSOCAF deployment automation framework via DeploymentManifest.xml do not need to be documented.

Note

For third-party products, do not point to the vendor's documentation. You must provide Microsoft with the necessary documentation in this deliverable by providing a completed MSODeploymentGuide.docx document.

- ☐ (If applicable) The expected duration of down times for deployment configuration and post-deployment configuration of this custom solution. If no downtime is anticipated, indicate 0 minutes.
- ☐ Instructions for verifying the release, to confirm that the package was properly installed and the custom solution works correctly.
- ☐ Any architecture diagrams, such as illustrations of dependencies and data flow.
- ☐ List of any dependencies, such as prerequisite software, related solution deployments, and related standard configurations.
- ☐ Notation of the presence of open source code, if any open source code is used in the custom solution.

Note

As a part of the deployment guide, the customer is required to provide a waiver in writing that the customer will be responsible for supporting any issues that arise from the open source software.

- ☐ A list of all design changes, if this design is not the original HLD for this custom solution.
- ☐ Thorough validation steps for the SharePoint Online operations teams to perform post-deployment.

Rollback Plan

Each solution package must include a rollback plan, in case the deployment must be rolled back.

- ☐ Uninstall and clean-up procedures for any components deployed manually.
- ☐ The previous solution files (.wsp files) to roll back to, if the previous version was not submitted through MSOCAF.

The Rollback Process for Remove Configuration Requests

The rollback process is required for Remove Customization Configuration Requests (CRs). This helps prevent service degradation that can occur after performing a remove action.

There are two ways to provide rollback steps:

Manual Deployment

If your solution was deployed using a manual process (for example, ISV solutions), do the following:

1. Package the solution with removed solution versions, including:
    a) Deployment scripts and feature activation scripts, located in the Installation Scripts/Rollback folder.
    b) The Deployment Guide rollback section with instructions on how to run the rollback scripts.

MSOCAF Deployment

If your solution was deployed using MSOCAF (for example, customer custom solutions), use the appropriate scenario and approach for the rollback package:

Scenario 1 – Solutions contained within one CR with only one version

For solutions that:

- Have only one version of the solution deployed to the Pre-production Environment (PPE) and Production Environment and;
- Are part of the same CR:
    1. Request the "retraction" of the original first version (v1.0) CR through CRAS.

Note
The Remove CR process is not supported in this case.

Scenario 2 – Solutions spread across multiple CRs or more than one version of the solution

For solutions spread across multiple CRs or if the solutions have more than one previous version (like v2.0, v3.0, …):

- Package the solution with removed solution versions, including:
  a) Deployment scripts and feature activation scripts located in the Installation Scripts/Rollback folder.
  b) The Deployment Guide rollback section with instructions on how to run the rollback scripts.

Solution Packages

Each solution package must include:

- ☐ Final tested code for deployment by Microsoft. This needs to have been validated with MSOCAF.

Test Documents and Results

Include details about all elements of the custom solution that you want deployed, including any third-party components and other solutions that you have purchased. This section must include the following test documents:

- ☐ **Test plan** that describes what was tested, why it was tested, how it was tested, and whether multiple WFE servers were tested. This should include details about what assumptions were made, and a usage profile or any data points that describe the test environment.
- ☐ **Test scenarios** that were tested, and what devices, browsers, latency, or permissions were used in testing. Describe the variations based on audience and presentation requirements.
- ☐ **Test results** that describe what tests were run for feature or functionality testing. This should include which clients were used, what the results were, what dependencies were tested, and what failure situations were tested.
- ☐ **Performance and scale test results** that describe what performance or scale tests were run against the custom solution. This should include which usage profile was used and what assumptions were made.

Dependencies List

- ☐ List of all dependencies for the code, such as a particular web service, account, database, solution, feature, patch, tool set, or library.

Event/ULS Log

The event log must include a custom event source with the following components:

- ☐ Table of all event entries that are generated by the custom solution, with event IDs. Table headings typically include error code, severity, and root cause.
- ☐ Troubleshooting instructions for all event entries.

The ULS log must include the following components:

- ☐ Category information that identifies the custom solution.

Client Troubleshooting Documentation

The troubleshooting information must include these two documents:

- ☐ Troubleshooting guidance for client-facing problems.

Note

For a template that shows the correct format for the client-facing troubleshooting document; see the Troubleshooting Guide Template (available to current customers only, on the Customer Extranet site).

- ☐ One or two test accounts for diagnosing problems with related dependencies, if appropriate. The account should be a low-privilege account that is used for no other purpose.

Source Code

All source code must be included in the submission, and must match the current version of the solution packages. Source code is treated with utmost confidentiality by Microsoft. Details about the treatment of customer source code can be found in the customer's contract, in the "Pre-Existing Work" section. If the solution is a third-party off-the-shelf application, no source code is needed for submission.

The source code section must include the following:

- ☐ Project files (where available)
- ☐ Source code, validated through MSOCAF
- ☐ Public debug symbols for all custom code

Note

Ensure that the .zip file format is used for file compression.

Third-Party Licensing Document

The licensing document must include:

- ☐ Any third-party licensing keys or certificates for deployment to the environment.

Note

Be sure to purchase licenses for the primary and secondary data center computers, and for a PPE if available.

- ☐ All details about the testing that has been done against the third-party component or solution.

Customer's Security and Compliance Review Document

We recommend that you have your own security and compliance teams review and approve all custom solutions that are being presented to Microsoft for deployment.

This review document is optional, but if you include one it should contain this information:

- ☐ The customer's Security and Compliance Review approval of the design.
- ☐ All test details from this review process.

Monitoring Document

The monitoring document must include:

- ☐ All monitoring guidance and details, which must include all information about logging and instrumentation for the custom solution.

Update or Revision to an Existing Custom Solution

When updating an existing custom solution and therefore existing documentation, any change to the solution requires submission of a complete new package for review that covers the updated solution.

Provide the following components to Microsoft:

- ☐ Updated versions of *all support documents* that are detailed earlier in this checklist, such as an update for the end-user troubleshooting guide. Include descriptions of anything that has changed and any design changes and new features that could potentially require support or monitoring.
- ☐ Summary of changes between current version and previous version, in adequate detail to describe clearly what has changed. (Use the "Changes in This Release" section of the HLD template.)
- ☐ The original solution source code, for validating against the new source code.
- ☐ As appropriate, either upgrades steps or uninstall (rollback) information for the previous custom solution, along with an uninstall test that can be run to confirm that the custom solution has been removed.
- ☐ Details about any problems that are expected to be encountered during the upgrade: any errors, rendering issues, or other problems that might be expected when upgrading the custom solution.

Appendix D: Rules Enforced by MSOCAF

MSOCAF validates the custom solution code using a set of rules that are based on established best practices for SharePoint Online custom solutions. These rules are listed in Table 11. Rules marked with an asterisk (*) apply only to SharePoint 2010.

Table 11. Rules Enforced by MSOCAF

Rule Tests Against	Description

Use of SPList.Items	SPList.Items selects all items from all subfolders, including all fields in the list. This action consumes a significant amount of memory. Instead of using SPList.Items, use SPList.GetItems(SPQuery query).
	Resolution
	If the goal is to add a new item, then use the following approach:
	`SPList list = web.Lists["MyList"];`
	`SPQuery query = new SPQuery{Query = "0"};`
	`SPListItemCollection items = list.GetItems(query);`
	`SPListItem item = items.Add();`
	If the goal is to enumerate through a list, then use the following approach:
	`SPList list = web.Lists["MyList"];`
	`SPQuery query = new SPQuery;`
	`// Include only the fields you will use.`
	`query.ViewFields = "<FieldRef Name=\"ID\"/><FieldRef Name=\"ContentTypeId\"/>";`
	`query.RowLimit = 2000; // Only select the top 2000.`
	`// Include items in subfolder (if necessary).`
	`query.ViewAttributes = "Scope=\"Recursive\"";`
	`StringBuilder sb = new StringBuilder();`
	`// To make it order by ID and stop scanning the table, specify the OrderBy override attribute.`
	`sb.Append("<OrderBy Override=\"TRUE\"><FieldRef Name=\"ID\"/></OrderBy>");`
	`//.. Append more text as necessary ..`
	`query.Query = sb.ToString();`
	`SPListItemCollection items = list.GetItems(query);`
	If the goal is to get a specific item, then use SPList.Items.GetItemById or SPList.GetItemById(int id, string field1, params string[] fields).

Rule Tests Against	Description
	For more information, see Best Practices: Common Coding Issues When Using the SharePoint Object Model in the MSDN Library.
SPListItemCollection \GetItemByID inside loop	Do not retrieve an instance of an SPListItemCollection inside a loop. Each time a SPListItemCollection is requested SharePoint retrieves the items by making a database call. If you need to access an items collection, retrieve it according to the recommendations above and store the result in a variable outside the loop. **Resolution** Store the Items property return value in a SPListItemCollection variable. The database is queried only once, and then iterates over the result set that is stored within the collection object.
SPQuery without Row Limit property	Do not use a SPQuery object without setting the Row Limit property. Use of an SPQuery object without specifying the value for the Row Limit property the query returns all items, performs poorly, and even fails on large lists. See the rule "Usage of SPList.Items" above for an example of how to implement this. **Resolution** Specify a Row Limit property when using an SPQuery object. For more information, see Best Practices: Common Coding Issues When Using the SharePoint Object Model in the MSDN Library.
SPQuery Row Limit value range	Do not use an SPQuery object with a Row Limit value beyond the range of 1 to 2000. An SPQuery object without a value for Row Limit performs poorly, and even fails on large lists. Specify a Row Limit between 1 and 2000 and, if necessary, page through the list. **Resolution** Specify a Row Limit between 1 and 2000 and, if necessary, page through the list. For more details, see Best Practices: Common Coding Issues When Using the SharePoint Object Model in the MSDN Library.
Timer jobs	MSOCAF verifies that each custom solution that implements a timer job (derived from Microsoft.SharePoint.Administration.SPJobDefinition) is accompanied by a Feature receiver that implements both the FeatureActivated and FeatureDeactivated methods. The FeatureActivated method should contain code that checks to see whether the timer job already exists before creating it. The FeatureDeactivated method should contain code that deletes the timer job. **Resolution** Ensure that any timer job implementation has FeatureActivated and FeatureDeactivated methods. For more information, see Timer Job in the MSDN Library.
Out-of-the-box files modification	Do not modify the out-of-the-box files with a custom solution. MSOCAF checks that the custom solution does not modify the out-of-box SharePoint files. The only out-of-the-box files that a customer can modify are the document icons displayed in the SharePoint environment, and those must be modified by submitting a configuration request through the customer's SDM.

Rule Tests Against	Description
	Resolution Create new files as necessary in the relevant folder.
Verbose logging \ tracing	Verbose logging or tracing information can lead to huge log files and affect performance. MSOCAF checks custom solutions make sure they do not turn on verbose logging. **Resolution** Do not set verbose logging or tracing on.
Deprecated APIs	Using deprecated APIs might block migration to newer versions of SharePoint and is thus not allowed. **Resolution** Avoid using deprecated APIs in the code. For more information, see Microsoft SharePoint Server 2010: Deprecated Types and Methods in the MSDN Code Gallery.
Editing Web config files	SharePoint provides access to a web application through the object model using the SPWebApplication class. The SPWebApplication class has a SPWebConfigModifications collection property that contains all the modifications (SPWebConfigModification objects) made to this web application's web.config file. Do not modify web.config by using StreamWriter, XMLDocument, ConfigManager, or XMLTextWriter. **Resolution** Only modify the web.config file by using the object model.
Inline code ASPX pages	Do not use inline code with full trust on the server, because it is possible for a developer to expose data from anywhere in the SharePoint environment by making a call to elevate the privileges of the code during run time. **Resolution** Compiled assemblies provide a safer alternative than inline code. For more information, see Inline Code in the MSDN Library.
SPListItem.Update() inside loop	Do not use SPListItem.Update() inside a loop. **Resolution** If updates to multiple items are to be made, we recommend using the ProcessBatchData method. This method is designed for processing multiple commands against a SPList without having to open a SPListItemCollection to invoke them. For more information, see SPWeb.ProcessBatchData Method in the MSDN Library.
SPMonitoredScope Web Part check*	Enable SPMonitoredScope for all custom methods of Web Parts (all Functions except OnInit, Render, OnPreRender). **Resolution** Wrap all code in custom methods with SPMonitoredScope for tracking. For more information, see Using SPMonitoredScope in the MSDN Library. *** Note:** This rule only applies to SharePoint 2010-based custom solutions.

Rule Tests Against	Description
SPDiagnostics Service	Be sure to call SPDiagnosticsService.WriteTrace at start and end of functions in timer jobs, event receivers, Feature receivers, and web services. **Resolution** Call SPDiagnosticsService.WriteTrace at start and end of functions in timer jobs, event receivers, Feature receivers, and web services. For more information, see SPDiagnosticsServiceBase.WriteTrace Method in the MSDN Library.
ULS logging	All exception handlers within a custom solution are required to write to the ULS log. MSOCAF detects whether ULS logging is done in all the catch blocks. Additionally, MSOCAF detects whether calls to unsupported APIs for logging are made. **Resolution** Write to the ULS log appropriately. For more details, see SPDiagnosticsServiceBase.WriteTrace Method in the MSDN Library.
BlobCache value editing in web.config	Editing of BlobCache value settings in web.config is prohibited. **Resolution** Do not edit the BlobCache value settings in web.config.
RunWithElevatedPrivileges restrictions	Elevating privileges of the user must be done securely. All code is checked for the delegate usage of RunWithElevatedPrivileges and then checked for instructions for file/folder deletion, user profile deletion, or starting/stopping of critical services. **Resolution** You should not have a RunWithElevatedPrivileges delegate that will perform any of the following tasks: a. Deleting file(s) or folder(s) b. Deleting user profile(s) from SharePoint Online c. Starting or stopping critical services in the SharePoint Online environment
SharePointQueryWebPartsCheck	Errors happening in the Feature receiver on activation and deactivation are logged and thrown back to SharePoint.
Windows Claims Check	The Windows Claims Check rule flags any code that may break when deployed to a web application in Windows Claims authentication mode. It ensures that any customization using System Namespaces and could fail in *Claims* mode are flagged. MSOCAF verifies your code for the usage of the following name spaces and will flag them: • `Microsoft.SharePoint.SPUser.LoginName` • `Microsoft.SharePoint.SPUserInfo.LoginName` • `Microsoft.SharePoint.Utilities.SPPrincipalInfo.LoginName` • `Microsoft.SharePoint.GetPrincipalsInGroup` • `System.Security.Principal.WindowsPrincipal.Identity` • `System.Security.Principal.WindowsIdentity.Name` • `System.Security.Principal.NTaccount.Value`

Rule Tests Against	Description
	• `System.Security.Principal.SecurityIdentifier.Value` For information about claims authentication see Windows Claims Authentication earlier in this guide.

Appendix E: Third-Party Solutions for SharePoint 2013 that Work with SharePoint Online

Customers that purchase these third-party solutions should determine whether the solution can be deployed without an HLD review. Be sure to note the version number; requests for different versions require a review by Microsoft. For solutions that do not require HLD review, the licensing and other configuration information should be included in the deployment guide.

Any application created by a third party or by another division of Microsoft has not been previously reviewed if it is not included in this list. Thus, it must follow the regular custom solution process to be deployed to a customer's SharePoint Online environment.

List of preapproved third-party solutions

Third Party Tool/Solution	Version	Requires HLD Review to Deploy
Nintex Workflow 2013 Enterprise	3.0	Yes
Metalogix SharePoint Extensions Web Service	6.1.0.2	Yes
Symantec Data Loss Prevention SharePoint Online Components	12.0	No
NewsGator	4.1	Yes
DocAve	5.6	Yes
MatchPoint	4.0	Yes

Approval of third-party solutions for deployment to a customer's SharePoint Online environment is for compatibility with the SharePoint Online environment only.

MICROSOFT IS NOT RESPONSIBLE FOR CUSTOMERS' USE OR INABILITY TO USE THIRD-PARTY SOLUTIONS AND IS NOT RESPONSIBLE FOR CUSTOMER DATA PROVIDED TO A THIRD-PARTY ISV AT A CUSTOMER'S REQUEST.