Upgrading SharePoint 2007 to SharePoint 2010

Anders Rask, SharePoint Solution Architect
Århus, March 2010

twitter: @andersrask

blog: http://andersrask.sharepointspace.com

version 1.1 (updated May 9th 2010)
CONTENTS

1. INTRODUCTION ........................................................................................................... 4
1.1. Outline ....................................................................................................................... 4
1.2. Acknowledgements ................................................................................................. 4
1.3. Updates ...................................................................................................................... 4
2. UPGRADEX FARM FROM SHAREPOINT 2007 TO SP2010 ......................................... 5
2.1. The upgrade cycle ................................................................................................. 5
2.1.1. Learn .................................................................................................................. 6
          Requirements and prerequisites ............................................................................. 6
          Upgrade methods .................................................................................................... 9
          Downtime mitigation processes ............................................................................. 16
2.1.2. Prepare ............................................................................................................... 19
          Document environment ......................................................................................... 19
          Manage customizations ....................................................................................... 19
          Choose upgrade strategy ...................................................................................... 24
2.1.3. Test ..................................................................................................................... 24
          Build test farms ....................................................................................................... 25
          Document and install customizations ................................................................. 25
          Use real data ........................................................................................................... 25
          Evaluate techniques ............................................................................................... 25
          Find issues early ..................................................................................................... 26
2.1.4. Implement .......................................................................................................... 27
          Build/upgrade farms .............................................................................................. 27
          Deploy customizations ......................................................................................... 27
          Minimize downtime ............................................................................................... 27
          Monitor progress ................................................................................................. 27
2.1.5. Validate .............................................................................................................. 28
          Upgrade event failures ........................................................................................... 28
          UI/UX issues .......................................................................................................... 28
          Data issues .............................................................................................................. 28
2.2. Visual upgrade ....................................................................................................... 28
2.3. No International Domain Name support .................................................................. 28
3. UPGRADEX SOLUTIONS AND CODE ....................................................................... 29
3.1. Recompilation ....................................................................................................... 29
3.2. Upgrading Custom Site Definitions ...................................................................... 29
1. INTRODUCTION

1.1. Outline

This document describes guidance for upgrading a SharePoint Products and Technologies 2007 (SP2007) farm to SharePoint Products and Technologies 2010 (SP2010). The various approaches to upgrade will be described and the pros and cons of each approach will be considered.

The first part of this document will discuss the process of performing an upgrade from SP2007 to SP2010 to include preparation, methodology and finalization.

The second part of this document will focus on the upgrading custom solutions and will discuss some of the various tools and features available to assist in this process.

The final part will look at actions that should be started, to prepare a solution for an upgrade.

I have tried to encompass both the operations and development angle of the upgrade process in this white paper. This have sometimes forced me to not dig as much into a given subject as I probably would have liked to do, in an attempt to get this document finished.

1.2. Acknowledgements

A few people helped me by reading the initial drafts and suggesting changes: Mike Watson (http://www.sharepointmadscientist.com), Paul Swider (http://www.paulswider.com) and Wictor Wilén (http://wictorwilen.se). I know you guys are busy, so thanks a lot for some great input!

In my research for this white paper I have read a lot of blogs and specs, and watched a lot of screen casts on the subject. I have tried to give credit where credit is due, but should I have missed accreditation let me know and I will include it.

I will also appreciate any feedback and corrections from the ever growing SharePoint community.

**Note:**

This document should be considered a work-in-progress. As very few actual upgrades has been carried out at this point in time (SharePoint 2010 still being in beta) prescriptive guidance is scarce. It is my plan to keep this white paper up to date as best practices become established.

1.3. Updates

<table>
<thead>
<tr>
<th>Date</th>
<th>Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/9/2010</td>
<td>Chapter 3.1 – 3.3 updated with more info</td>
</tr>
<tr>
<td></td>
<td>Added IDN upgrade and BDC upgrade</td>
</tr>
</tbody>
</table>
2. UPGRADING FARMS FROM SHAREPOINT 2007 TO SP2010

Note:
There is no upgrade path from the public beta version of SP2010 to the RTM when released. Beta can be used to evaluate the product and to test upgrades, but since it is not a supported product, upgrade is not supported!

2.1. The upgrade cycle

When talking upgrade of complex SharePoint solutions, it is important to emphasize that this initially is an iterative approach:

- **Learn**
  - find out all about requirements, prerequisites, documentation, the upgrade process, downtime mitigation, common issues

- **Prepare**
  - document environment thoroughly, upgrade existing documentation, find and manage customizations, choose upgrade strategy, performance test existing hardware

- **Test**
  - build a test farm using real data, evaluate migration techniques, find issues early

- **Implement**
  - upgrade farms, deploy customizations, minimize downtime, monitor progress

- **Validate**
  - upgrade event failures, UI/UX issues, data issues
2.1.1. Learn

Requirements and prerequisites

**Software and hardware**
The biggest change in architecture from 2007 to SP2010 is, that all servers, including SQL server, must run 64-bit. This is mainly because of scalability issues, the need for large amounts of RAM on the server and to focus support on one version.

The minimal requirements for hardware is pt specified to be:

<table>
<thead>
<tr>
<th>Component</th>
<th>Minimum requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>64-bit, four cores</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GB for developer or evaluation use</td>
</tr>
<tr>
<td></td>
<td>8 GB for single server and multiple server farm installation for production use</td>
</tr>
<tr>
<td>Hard disk</td>
<td>80 GB for system drive</td>
</tr>
<tr>
<td></td>
<td>For production use, you need additional free disk space for day-to-day operations.</td>
</tr>
<tr>
<td></td>
<td>Maintain twice as much free space as you have RAM for production environments.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 1: Source TechNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
</tr>
</tbody>
</table>

In addition to hardware requirements for running SP2010, one must also consider the upgrade process itself and how it may be impacted by hardware as well. For example, the upgrade process may take 4 hours on one set of hardware and 2 hours on another. The speed of the upgrade will be determined in large part by the physical resources available to the SQL server(s). Expect the upgrade to run much faster when the SQL server(s) performing the upgrade has excess processor, memory and physical disk IO capacity. Also, the upgrade process can benefit greatly by scaling out across multiple SQL instances with each instance running one or more upgrade processes. Also be aware that the upgrade itself takes up extra disk space for log files and databases.


Furthermore database server must be 64-bit version of either SQL Server 2005 SP3 with cumulative update 3 (CU) or SQL Server 2008 SP1 with cumulative update 2.

SharePoint 2007 must have SP2 and latest CU (currently April CU), since a lot of the tools used for upgrading SharePoint is baked into the service packs.
Note:
The above prerequisite upgrades can be combined, but must not be part of the SharePoint upgrade itself!
Read more on hardware and software requirements on TechNet

Pre-upgrade check
That the tools are already in place also means that you can start planning an upgrade, by running the pre-upgrade checker. The `stsadm.exe` command `PreUpgradeCheck` can be used to analyze the existing SP2007 site collections, looking for situations that could cause grief during an upgrade, such as customized (unghosted) artifacts, changes in database schemas, missing features and other potential issues and relevant information like Alternate Access Mappings (AAM) url’s, site definitions used and large lists.

It is also important to state, that the tools operations on the databases are read-only! No changes are made on the databases, which makes it a relatively harmless procedure to run, even on your production environment (in comparison with PreScan from 2003-2007 upgrades that would make small alterations to the databases).

During execution `PreUpgradeCheck` will visually display progress in the console.

- **Green** text means everything is fine
- **Yellow** means you will find more information available when digging into the log file with references to KB articles; manual upgrades will also show up here. Below you can see that CAML views are used instead of the new XSLT-based views, this needs to be upgraded manually, also listed are AAM configuration, server and farm info, installed language packs etc.
- **Red** means that there’s an issue that needs attention before an upgrade can be completed successfully. In the example below the upgrade fails to find the xml for an installed feature and also fails on the prerequisites (the server is 32-bit).

![Figure 1: Pre-Upgrade Check in action](source: http://www.wictorwilen.se)

After `PreUpgradeCheck` has finished, it will generate a report in both XML and HTML format and a log file. The `PreUpgradeCheck` runs against a rules database that is extendable. You can select what rules to run by specifying the `rulefiles` parameter followed by a comma-
separated list of rule names. You can also see a list of rules being applied by specifying the listrulefiles parameter.

**Note:**


PreUpgradeCheck can be run on both a single server and a whole farm. There are two obvious benefits of this: running it locally will only stress a single Web Frontend (WFE) server, which is good if run on a production environment. Also you can run PreUpgradeCheck on individual WFE and afterwards compare the reports against each other to spot inconsistencies across the frontend servers.

PreUpgradeCheck is meant to be run a number of times, not just as a one-off event. Identifying customizations and rehearsing upgrade operation is paramount when we want to achieve a successful upgrade with minimal downtime. For this reason IT should run PreUpgradeCheck on a regular basis as an ongoing process towards the Version-To-Version (V2V) upgrade.

Common issues when upgrading is upgrading language packs to latest version, upgrading custom site definitions to take advantage of new SP2010 functionality (for more information, see [http://tinyurl.com/mulfcb](http://tinyurl.com/mulfcb)), missing features (only guid is stored in the database), large lists (SP2010 uses throttling on large lists, so code may fail! See more in code upgrade section) or orphaned artefacts in configuration or content database.

**Test-SPContentDatabase**

To complement PreUpgradeCheck reports, as part of the pre-upgrade testing you should run the SP2010 PowerShell (PS) command Test-SPContentDatabase. This command compares a content database and a web application against each other checking for problems. It can be used against both old 2007 content database and the upgraded SP2010 database.

The tool will check for orphans, missing site definitions, features, assemblies etc.

In other words it will warn you if it detects any potential problems with matching a specific web application and database, such as creating orphans by adding a database that is already in the farm.

![Figure 2: Test-SPContentDatabase example output](image)
So where Pre-upgrade Check is used to detect issues on the SP2007 environment, Test-SPContentDatabase can be used to for example analyze a SP2010 farm before attaching a content database to it.

Upgrade methods

Part of the learning process is knowing your options! There’s several ways to upgrade your SharePoint solution and even hybrid variations. Each method has its pros and cons, with concern for downtime, hardware costs etc.

In-Place upgrade

An in-place upgrade means that the upgrade is done directly on the production server. Since this means closing down the farm for the duration of the upgrade, this approach causes downtime for the solution.

On the other hand the approach means that the existing server hardware can be reused (if within specifications and adhering to prerequisites) and that configurations and customizations done on the server is kept. E.g. you don’t need to recreate a complete farm using solutions and manual configuration.

![Figure 3: In-place upgrade (source TechNet)](http://technet.microsoft.com/en-us/library/cc303423(office.14).aspx)

Doing an in-place upgrade, first install SP2010 on *all* servers in farm -start with the server hosting the Central Administration (CA). Then install language packs. Now run configuration wizard up to point where wizard tells you to configure other servers in farm -start with CA. When wizard is on same step on all servers complete wizard on CA continue on other servers.

As an option you can end up running Visual Upgrade (the new SP2010 look for editing sites, with Ribbons etc.).

If you have problems during in-place upgrade, the PS command Upgrade-SPContentDatabase can be used to resume an upgrade.

**Note:**


Pros:

- customizations are kept
- farm-wide settings preserved

Cons:
a risky approach since you don’t have a fallback strategy should issues arise

downtime while upgrading (can be mitigated with AAM redirects, see hybrid model below)

all content databases are upgraded in sequence causing more downtime

a power outage or disk space problem during upgrade could leave upgrade in an unsupported state

**Database attach**
The database attach approach requires you to create a new farm on new hardware. This farm is then configured, and customizations and artifacts are deployed. Now you backup your old farm, detach it taking it offline and attach it to the new farm (discard temporary content database in new farm).

Attaching the new database could be done with either the PowerShell command `Mount-SPContentDatabase -name <newdb> -WebApplication <url>`, or use `stsadm -o addcontentdb -url <url> -databasename <dbname> [-preserveolduserexperience true|false]`. The last approach should be preferred if you want control over the UI upgrade (e.g. Ribbons), since it honors the version switch for UI, whereas the PS command forces the new UI (at least until RTM version).

This method is also viable for SSP database and upgrade user profile information into the database, but you cannot upgrade search database by using this method.

If you have problems during db attach upgrade that you need to address before continuing, the upgrade process is designed so that it can be resumed even in the event of power outage or if you run out of space during the upgrade process: run the PS command `Upgrade-SPContentDatabase` to resume an upgrade.

**Note:**

**Pros:**
- can upgrade multiple content databases in parallel (less downtime)
- you can use this method to consolidate multiple farms into one farm
- you can upgrade hardware as well as software
- you have an opportunity to clean out the old server and get a “fresh” install

**Cons:**
- server and farm settings are not upgraded (mitigation: scripted installs)
- the settings of the target farm must exactly match the settings on the source farm
- customizations are not upgraded (mitigation: solution deployment, scripted configurations with PowerShell)
- copying databases over network takes time (plan this!)
- requires direct access to SQL server
Hybrid approach 1: Read-only databases

Hybrid approaches gives you the possibility of combining different approaches when upgrading SP2010. One such approach is the R/O databases approach. Basically this is a db attach upgrade but with a downtime mitigation strategy where you continue to provide read-only access to content database during the upgrade.

Start by setting up and configuring a new farm, then transfer customizations to new farm and test. Now set content databases to read only (directly in SQL) on original farm while upgrade in progress on new farm (since Sp2 SharePoint will detect that the database is read-only so that the UI respects this). Backup content database from original farm and perform database upgrade on the new farm in parallel. Optionally use AAM for long-running upgrades to redirect requests (see more on this approach later). Map sites from new farm to old farm while upgrade is in progress.

**Note:**

You can configure the READ_ONLY database availability option by using Transact-SQL. More about how to use the SET clause of the ALTER DATABASE statement: [http://go.microsoft.com/fwlink/?LinkId=148362](http://go.microsoft.com/fwlink/?LinkId=148362).

**Pros:**

- Existing farm can continue to run in read-only mode causing minimal downtime for end users
- can upgrade multiple content databases in parallel (less downtime)
- you can use this method to consolidate multiple farms into one farm
- You can upgrade both software and hardware

**Cons:**
- Server and farm settings are not upgraded (mitigation: scripted installs)
- Customizations are not upgraded (mitigation: solution deployment, scripted configurations with PowerShell)
- Copying databases over network takes time (plan this!)
- Requires direct access to SQL server

**Hybrid approach 2: Detach databases**

Another hybrid approach is a variation over the in-place upgrade: This approach combines the in-place upgrade’s ability to keep configurations and customizations while adding the parallel upgrade approach from db attach positively affecting downtime for the upgrade:

Take the original farm off-line, detach content database from original farm, run in place upgrade on original farm servers in parallel, services and configuration databases. Then attach content databases to the original farm and upgrade content.

**Pros:**
- Customizations are kept
- Farm-wide settings preserved
- Save time by upgrading multiple db’s at the same time

**Cons:**
- Copying databases over network takes time (plan this!)
- Requires direct access to SQL server

**Hybrid approach 3: Detach databases (with temporary farm)**

This approach is very similar to the above hybrid scenario, but it introduces a new small farm that is used temporarily to store the content databases as they are being upgraded:

Set up temporary small farm (both WFE and applications running on same hardware) running SP2010 and then take the original farm offline. Detach the content databases from the original
farm and run an in-place upgrade on original farm. Now attach content databases to temp farm and upgrade content in parallel. Finally re-attach content databases to the original farm.

**Farm before upgrade**

**Topology in transition**

**Upgrade complete**

---

**Figure 7: Hybrid: Detach databases with temporary farm (source: TechNet)**

**Pros:**

- Same as hybrid 2 approach above +
- Reduce downtime since upgrade is carried out in parallel on temp farm

**Cons:**

- Same as hybrid 2 approach above +
- New hardware needed for temp farm (could be some existing test server)

**AAM hybrid: detach databases**

The AAM hybrid should be seen as a last ditch operation, and is only viable for very specific situations, like if you cannot upgrade your farm over a weekend.

The reason for this being that it is operationally fairly difficult to set up. It also isn’t perfect; since it has issues with links (different URL’s on new and old farm). Furthermore it gives you double work (e.g. governance of security, double hardware, double maintenance).

The upgrade is related to what in the old version was called Gradual Upgrade (no longer supported). Basically db attach is used to upgrade content databases one at the time over a longer period. AAM is then used on the new farm to redirect users that request pages that haven’t yet been upgraded to the old farm (http://WSSOld). Over time (could be weeks or even months) all content databases are upgraded one at the time.

Compared to Gradual Update the granularity here is entire content databases, not site collections.

When the databases are upgraded the old databases could be kept as read-only as a kind of post view upgrade to look at old content to compare with new.


**Updating Services**

Services have been totally reworked in SP2010. There is no longer a Shared Services Provider (SSP) site, but instead you got the possibility to scale out the services to individual servers (through proxies) with individual databases. This flexibility is great in terms of scaling out, but adds complexity to upgrade scenarios. You really need to plan beforehand what services are in use in
the farm, and where they should be placed after upgrading to SP2010. Also some services are
split up into two separate services, where one is completely new. Depending on the upgrade
approach manual work is needed to fully upgrade the service architecture.

Another important design change from 2007 to SP2010 is that where some services was specific
to Microsoft Office SharePoint Server (MOSS) -some even only in Enterprise edition, they now all
reside inside Microsoft SharePoint Foundation (formerly Windows SharePoint Services (WSS)). This
should cause solution architects to consider the new possibilities available for the customers’
farm, maybe even change existing solutions to make use of these new possibilities.

Important:

Even with in-place upgrades, not all configurations are kept after upgrade. These settings, such as
timer job configurations, must be collected before upgrade and re-applied post-upgrade.

Below is an illustration of SSP architecture before and after an upgrade:

If you have a single SSP, all proxies for service applications are added to the default proxy group.
The following diagrams show the changes to your farm that are made during in-place upgrade.

Services infrastructure before upgrade:

![Single SSP farm before upgrade](Source: TechNet)

![Single SSP farm after upgrade](Source: TechNet)

Figure 8: SSP before and after upgrade (Source: TechNet)

Note:

If you have multiple SSP’s, they will be upgraded together and after the upgrade you will have
multiple proxy groups!


**User Profiles**

User Profiles are now split up in two services:

- User Profile Service
- Managed Metadata Service (new in SP2010)

If you run an in-place upgrade, the managed metadata service is automatically enabled and configured. If you upgrade using db attach you will need to enable and configure Managed metadata before upgrading!

Persisted properties relating to profiles are also preserved when using in-place upgrades:

- MySiteHostURL
- SearchCenterURL
- EnablePersonalFeaturesforMultipleDeployments
- ProfileStoreLanguage
- ProfileStoreLanguagePacksApplied
- ProfileStoreCollationID
- DaysWorthOfEventsToKeep

On the other hand a db attach approach will not preserve these properties since they are stored in configuration database. You also will need to enable and configure the Managed Metadata service before you upgrade the User profile service to make taxonomy data part of the upgrade.

If you have taxonomy data that needs to be migrated (if you planned meta data before upgrading), use the `Move-SPProfileManagedMetadataProperty` command in PS.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>To upgrade and use taxonomy data, the User Profiles Service proxy and Managed Metadata Service proxy must be in the same proxy group.</td>
</tr>
</tbody>
</table>

**My Sites**

If you use My Sites, make sure you upgrade the My Site host at the same time as you upgrade the user profiles. Also make sure you upgrade My Sites host as part of the intranet migration process!

When you upgrade My Site host it will automatically upgrade to the new look and feel of SP2010, so any customizations on personal and shared My Site pages will be lost!

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You don’t need to upgrade all the My Sites themselves at the same time as doing the User Profile upgrade, just the host!</td>
</tr>
</tbody>
</table>

**Search**

You cannot use db attach to upgrade search data. Instead you should configure search in your new farm before or after the upgrade.

If you use in-place upgrade, you should review and adjust search topology after upgrade to suit new recommendations and requirements.
For db attach approach you need to export XSN files and UDCX files before upgrading and import them into new farm after upgrade:

- Export-SPInfoPathAdministrationFiles
- Update-SPInfoPathAdminFileUrl to update links if url is different in new farm

You cannot use in-place for FormsServices.

**Excel Services**

Excel Services is still a local service (it runs service in same farm that consumes it).

If you upgrade Excel Services using in-place upgrade: configuration info stored in SSP is automatically moved from SSP db to configuration database.

When using the db attach approach, you need to reconfigure Excel Services on the new farm.

After upgrade (db attach and in-place), a new unattended service account must be provisioned for Secure Store Service.

**Business Data Catalog (BDC)**

When you do an in-place upgrade, data from SSP is moved to a new dedicated database and a new service application is created.

BDC is not upgraded in a db attach upgrade process.

Old BDC Connections are run using Application Registry Backwards compatible service. The interface for this is kept in the old SSP admin site. New development should **not** be done in Application Registry Service, as this service is only meant to be used for upgrading BDC from SP2007!

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>If no BDC services were available for the old solution, the SSP site can be deleted after upgrade!</td>
</tr>
</tbody>
</table>

Consider moving the BDC profile pages to a new location, as these were hosted in the SSP web application.

**Single Sign-On (SSO)**

The SSO service is replaced with Secure Store Service in SP2010. Use the PS cmdlets below to upgrade application definitions:

- Upgrade-SPSingleSignOnDatabase
- Upgrade-SSOConnectionString
- Upgrade-SecureStoreConnectionString
- Upgrade-SecureStorePassphrase

Notice that passwords are not upgraded, so these will need to be configured post-upgrade. Also you must manually set Secure Store Service the default SSO provider after the upgrade is done.

**Downtime mitigation processes**

Usually you would like to minimize downtime during an upgrade. Several parameters affect your downtime: The chosen upgrade model, server performance, size of farm and databases, how well you tested etc.

There are different processes that you can use to minimize downtime.
Give users read access during upgrade
One way is setting the source database to read-only during an upgrade. This will enable end users to access their data without changing it (SharePoint detects the SQL lock on the database and enforce UI trimming accordingly). The users will then only detect downtime when the solution is switched to the new farm.

Upgrading in parallel
To minimize the time used to run the upgrade use parallel upgrades. You can do parallel database attach (number of parallel upgrades depends on hardware) and create multiple temporary farms to do in-place upgrade and db attach on.

Content database attach with AAM redirection is another way to reduce downtime.

Avoid surprises – test!
More subtle approaches could be to optimize farm before upgrade, to avoid surprises during the production upgrade: make sure you follow recommendations from pre-upgrade checker, split large content databases into smaller ones, test (on real data) – the more you rehearse the upgrade process, and the more “real” the test environment and test data are, the more certain you will be on a successful upgrade.

Common issues that is only found through testing includes missing dependencies (features not deployed to new farm, or missing on one or more WFE), UI change (CSS will break if you just upgrade to the new UI without upgrading CSS), lack of space (for example on SQL server, you should expect x2-x3 space – especially depending on # of document versions - increase during an upgrade), there’s almost always some manual post-upgrade configuration that depending on setup needs to be done (for example configuring additional settings on Forms Authentication providers for claims-based web application).

Clean up before upgrade
It is very hard to predict the amount of time an upgrade will take. Performance will vary on a lot depending on farm metrics:

- # site collections
- # webs
- # lists
- # document versions
- Document versions size
- # documents
- # links
- Overall DB size

To mitigate the above, do general “spring cleaning” on your site collections: delete unused sites, lists and documents. Clean up in number of versions for documents. Split up large content databases.

Note:
Remember to backup your databases before cleaning up!

STSADM.EXE has operations to automate part of this procedure
Delete live site collection:
stsadm -o DeleteSite -url <URL> [-deleteadaccounts {True | False}] [-gradualdelete]

Delete orphaned site collection:
stsadm -o DeleteSite -force [-gradualdelete] -siteid <site ID> -databasename <database name> -databaseserver <database server name>

Delete live site:
stsadm -o DeleteWeb -url <URL>

Delete orphaned site:
stsadm -o DeleteWeb -force -webid <Web ID> -databasename <database name> -databaseserver <database server name>

Since the amount of versions directly affect the time it takes to upgrade, consider manually deleting old document versions, or create a tool to automate this task.

Clean up unused templates, features and web parts. Again this is a manual process, but a custom tool could automate the process (for example listing all unused templates and giving you the option to delete them).

Repair data issues:
stsadm -o DatabaseRepair -url <url> -databasename <database name> [-deletecorruption]
stsadm -o ForceDeleteList -url <url>

Check and remove locks on site collections (when doing backups):
stsadm -o getsitelock -url <url>
stsadm -o setsitelock -url <url> -lock {none | noadditions | readonly | noaccess}

Revise hardware and server settings
Performance also varies based on hardware and software metrics such as (in order of importance):

- SQL disk I/O per sec.
- SQL DB to disk layout
- SQL temp db optimizations (one per cpu)
- SQL CPU & memory
- WFE CPU & memory
- Network bandwidth & latency

Revising the hardware and configuring server software before upgrading will help bring down the amount of downtime for an upgrade.
2.1.2. Prepare

Document environment
If your environment is not documented, this is the time to do this! If it is documented, this is the time to revise your documentation to ensure its up to date!

You should document hardware, software, customizations (see more below) and configurations. This will assist you in estimating the scope of the upgrade, and make disaster recovery after a failed upgrade much easier.

Manage customizations
Probably one of the most common reasons for a failed upgrade is not knowing the extent of customizations on your farm. Are all customizations done using solution deployment? Are manual special case customizations that cannot easily be solved using solution deployment documented? And are these special cases in sync across WFE?

Note:
An upgrade is an excellent time to enforce government policies. If “rogue” customizations is found this should be followed up with guidance on packaging artifacts in solutions, using features etc.

To answer these questions, you have a number of tools to help you, but you will also have to dig in GAC, bin, 12 hive, Solutions store, Add/remove programs, etc. to get an overview of the customizations on the farm.

Examples of customizations include custom site/list definitions, themes and changed CSS, master pages, page layouts, content types, custom web parts, custom web controls, event handlers, customized/un-ghosted pages, application pages, custom timer jobs, AAM’s etc.

The following section will try to shed some light on how to identify customizations in your farm:

Pre-upgrade check
First of all run pre-upgrade check tool on both farm and individual servers (running on individual servers and then comparing reports will give you a hint of how similar your WFE are).

Note:
List of all WSS/ MOSS Pre-Upgrade Check KB articles: [http://support.microsoft.com/kb/960577](http://support.microsoft.com/kb/960577)
Joel Oleson has a good blog post on the subject [http://www.sharepointjoel.com/Lists/Posts/Post.aspx?ID=238](http://www.sharepointjoel.com/Lists/Posts/Post.aspx?ID=238)

Customized/Unghosted files
Pre-upgrade does a good job checking for customizations, but does not detect files customized (unghosted) in SharePoint Designer (SPD). A tool like Gary Lapointes gl-enumunghostedfiles (part of stsadm extensions [http://stsadm.blogspot.com/2009/02/downloads.html](http://stsadm.blogspot.com/2009/02/downloads.html)) can help identifying and reghosting these customizations.

Test the content database
In SP2010 there’s a new tool available that will help identifying missing customizations: the PS cmdlet Test-SPContentDatabase can detect problems before you attach a content database to a farm. You can see this cmdlet as a compliment to pre-upgrade checker report,
plus it works on both SP2010 and 2007 databases, so it is very useful to point at an upgraded database to check if assemblies, site definitions or features are missing or if there are undetected orphans. It also will show metrics for table sizing on a content database, which can be useful for detecting content approaching the software boundaries of the product.

**Note:**
Joel Oleson walks through the syntax and uses of Test-SPContentDatabase on his blog: [http://www.sharepointjoel.com/Lists/Posts/Post.aspx?ID=288](http://www.sharepointjoel.com/Lists/Posts/Post.aspx?ID=288)

**EnumAllWebs**
Another tool to determine impact of customizations is stsadm -o enumallwebs. This command can be used to list the ID and sitemap status for all site collections and sub-sites in a specified content database. Especially sitemap status (InSiteMap="True | False") is useful, as this tells you if a site collection is orphaned in the content database (this could happen if a content database has been attached to a web application that already contained a site collection with the same URL). An orphan can both be a site only registered in content database, or a site only registered in the configuration database. Such orphans will need to be handled before upgrading the database.

**Note:**

Always remember to backup your content database before deleting any sites or site collections!

**Deployment Advisor from Quest**
Deployment Advisor (DA) developed by Quest Software Inc., is a new tool due for release soon: One of the main purposes of this tool is to give Operations a way to get a sanity check on a given SharePoint farm: have the server been configured in compliance with best practices in the field? Does WFE contain unique configurations or customizations? Is the farm ready for an upgrade? Answering these and other questions makes Operations able to assess risks for SharePoint farms in regards to hardware, patches, customizations, security and performance. DA scans the farm against an extendable rules engine that describes best practices for SharePoint within categories such as Performance, SP2010 upgrade, Availability, Search, Security, Supportability and areas such as Antivirus, Farm Configuration, IIS, Network, Server and SQL.
In an upgrade scenario, you can use DA to compare WFE servers (one of the ideas behind the tool is for it to be “the WinDiff of SharePoint”) with regards to configuration, customization, patches etc. You can also look at the specific farm with regards to 2010 upgrade issues. Here it will tell you what critical issues that need to be resolved before an upgrade can take place, such as upgrading to a 64-bit architecture on both web servers and SQL servers:

You can also examine the SP2010 upgrade readiness for a specific server:

One very powerful feature of DA is its ability to compare servers to each other. This mind you is across metrics such as hardware, software, patch-level, files on server, services on server etc. This proves useful both if you want to compare different WFE in the same farm, but also if you want to prepare for an upgrade: Say you create a clean install of SP2007, fully patched and following best practices. Then you compare that to the server you want to upgrade. That gives you the possibility to detect if files like `core.js` or other “Microsoft owned” files has been customized on
the server in question. You can even filter on basically anything (like %.js) to fine tune your comparison. Very neat!

<table>
<thead>
<tr>
<th>Name</th>
<th>Old Size</th>
<th>New Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>_mylayouts_123追捧.js</td>
<td>11680</td>
<td>117200</td>
</tr>
<tr>
<td>_mylayouts_123超强.js</td>
<td>9599</td>
<td>30222</td>
</tr>
<tr>
<td>_mylayouts_123超强core.js</td>
<td>17191.25</td>
<td>66601</td>
</tr>
<tr>
<td>_mylayouts_123超强protect.js</td>
<td>29909</td>
<td>205944</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>57997</td>
<td>50229</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>67161</td>
<td>60951</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>213556</td>
<td>216064</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>37707</td>
<td>39707</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>151891</td>
<td>159639</td>
</tr>
<tr>
<td>_mylayouts_123超强mmp.js</td>
<td>13397</td>
<td>13853</td>
</tr>
</tbody>
</table>

**Figure 9: Comparing servers in Deployment Advisor showing Core.js is customized**

In general this comparison against a “best practice server” is also useful if you take over a farm and want to quickly get an overview on the general state of the server by comparing metrics like BuildVersion, patch-level etc. with your “golden” server.

**Manual inspection**

A manual inspection of your farm could include

- checking in Visual Studio and Solution store if everything is packaged in solutions
- any manual editing of web.config (note that this will need to be checked both in relation to differences in web.config on different WFE in the farm, and across environments (devtest, integration test, preprod, prod)
- any manual xcopy operations.

These manual steps should be documented and if possible mitigated with solution deployment.

**Places to check for customizations**

- _layouts, features, sitedefinitions
- GAC
- add/remove programs (3rd party)
- timerjobs, event receivers
- http handlers/modules/iis customizations

Pre-upgrade check does detect database customizations, but other kind of modifications of Out-Of-The-Box files such as webtemp files, application pages etc. will not be picked up.

A way of detecting these customizations is using the above mentioned Deployment Advisor, Windif (or similar) to detect differences

- a) from files as they were OOTB (install a clean farm and compare)
- b) between WFE on the same farm
- c) between environments

Also inspect code, looking for hacks that may cause problems. A good developer would always mark these special cases with some kind of code comment.

Since STP files are no longer supported, look for these in your development environment. A way to upgrade STP files to WSP packages is by restoring them on a SP2007 site that is then in-place upgraded. After fixing any visual issues the template can be exported as a WSP package that can then either be used to create new sites from using the UI, or be exported to Visual Studio 2010 and be packaged for deployment. Both the export and import tools has a tendency to
import too much, so count on using time cleaning up the solutions before they are ready for deployment. More on this in a later chapter on upgrading code.

**Other tools for detecting customizations**

SPDiag version 2 is good for farm insight such as AAM’s or finding deployed solutions using the SnapShot tool. Diagnostics tool is also handy for detecting any discrepancies regarding best practices on configuration of the farm (Part of SharePoint administration Toolkit 4.0 that can be downloaded here [http://technet.microsoft.com/en-us/library/cc508987.aspx]).

WssAnalyzeFeatures. This tool verifies if the feature definition files for all installed features are available on the file system, if the features used on a site collection are installed on the server (download from MSDN Code here [http://code.msdn.microsoft.com/WssAnalyzeFeatures]).

Bamboo SharePoint Analyzer can help you get an overview of your farm topology, installed patches on servers, solutions and features deployed etc. (available here [http://community.bamboosolutions.com/media/p/7160.aspx]).

SharePoint Feature Administration and Clean Up Tool can help locating faulty features in your farm (available from Codeplex [http://featureadmin.codeplex.com]).

**Collect customizations**

When all customizations has been collected, create a list of customizations along with source, environment and action required to move customization (could also be not to move it, e.g. if it’s a SP 2007 specific customization). The list should also contain third party add-ins and assemblies.

When collecting customizations try and assess whether this customization is still relevant on the new platform:

1. Keep the customization. Choose if customization can be ported to new platform without issues.
2. Replace or redo customization. Choose if customization has visual or functional issues on the new platform, but you want to keep the customization.
3. Discard customization. Choose this if customization is no longer relevant.

The following table illustrates common customizations and recommendation for that customization.

<table>
<thead>
<tr>
<th>Customization type</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site templates (STP files)</td>
<td>STP files are a deprecated feature in SharePoint Server 2010. New site templates in SharePoint Server 2010 are saved as WSP files (solution packages). A site that was provisioned by using a site template will be upgraded, but you will be unable to create new sites based on that template. If you want to be able to create new sites, you can create and deploy a solution package instead.</td>
</tr>
<tr>
<td>Site definition</td>
<td>Migrate sites to a supported, predefined site definition, then apply custom features by using solution deployment. You can also continue to use a custom site definition. You do not have to create a new site definition based on SharePoint Server 2010. However, if you must perform custom upgrade actions for the definition, you might have to create an upgrade definition file for that site definition. For more information, see Upgrade Definition Files [<a href="http://go.microsoft.com/fwlink/?LinkId=182339">http://go.microsoft.com/fwlink/?LinkId=182339</a>] on MSDN.</td>
</tr>
<tr>
<td>Feature</td>
<td>Evaluate, then redesign or redeploy if necessary.</td>
</tr>
<tr>
<td>Workflows and server controls</td>
<td>Depends on the solution. Contact the vendor to find out whether there is an updated solution. If a workflow is compatible with the new version, redeploy.</td>
</tr>
<tr>
<td>Event handler</td>
<td>Rewrite and redeploy as a feature.</td>
</tr>
<tr>
<td>Managed paths (inclusions/exclusions)</td>
<td>Re-create inclusions for a database attach upgrade. Exclusions are assumed and do not have to be re-created.</td>
</tr>
<tr>
<td>Themes</td>
<td>Because of the extensive changes to the UI, custom themes based on Office SharePoint Server 2007 will not work in SharePoint Server 2010. Use Visual Upgrade to continue to use the sites in the old user experience until you can create and apply a new theme based on SharePoint Server 2010.</td>
</tr>
<tr>
<td>Toolbar actions</td>
<td>Move to the ribbon (Fluent UI).</td>
</tr>
<tr>
<td>Master pages and CSS files</td>
<td>Rework to accommodate the new user experience.</td>
</tr>
</tbody>
</table>
Choose upgrade strategy
When customizations are collected, it is time to plan what upgrade strategy should be chosen, and determine order of operations (what sites goes first? should sites be split up?).

**Note:**

Even SharePoint behind the scenes will set recovery model to Simple during an upgrade (applicable for beta 2 in-place upgrade), you should still expect your SQL server to require x2-x3 of its current space –especially if you have a lot of versions on your documents. This is in part caused by the fact that databases aren’t shrinked automatically after an upgrade for time saving reasons.
A How-To will come out shortly on TechNet on how to detect databases that need shrinking.

The strategy should include means to limit downtime, and document expected downtime, and describe actions for spring cleaning as described earlier.

It should also include a rollback strategy and a plan for when an upgrade should be abandoned and recovery of the old farm should start, any hardware upgrades due to new requirements, or space requirements.

**Note:**

It’s a good idea to do a performance analysis on your server hardware so you know beforehand if you should upgrade.

2.1.3. Test
The importance of testing before, during and after an upgrade cannot be stressed enough!

*It is imperative for the success of an upgrade that we have a test environment that we trust to be similar to the one we are going to upgrade in production.*

There are so many things that can go wrong during an upgrade, that without proper testing you could end up with either a long downtime, a site that’s not properly upgraded (missing features) or worse.
Build test farms
When you build test farms it is important that the metrics of the farm is kept as close to the production farm as possible!

Both with regard to hardware, software, configuration, customizations and content they should be kept similar. The more similar your test farm is to the real thing, the higher the probability of everything running smooth during the actual upgrade in production.

For hardware for example, the space on the disks plays an important factor: you would like to discover any space related issues during testing rather than having to add more disks during production upgrade.

If the test environment is virtual, it should also be kept as close to the real farm as possible. You should for example run SQL server and the farms on different virtual images.

If your tests environment isn’t identical you should keep it as similar as possible to the original: if you have multiple servers for a role (like 5 WFE) you should have at least 2 servers with that role in your test setup!

Document and install customizations
Use the worksheet mentioned above to document and install customizations and configurations.

Use real data
When you test the upgrade process, keep your content as close to production data as possible. This approach will help you identify trouble areas and determine upgrade performance. For example issues may rise due to large lists that you would not find on test data.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You don’t necessarily have to have all content on your upgrade test environment at the same time. Say you have 60 content databases with terabytes of data; it could be hard to convince your IT department to give you that kind of storage for a test farm. Instead test the content databases one at the time –just make sure you tested them all before attempting a real upgrade!</td>
</tr>
</tbody>
</table>

Evaluate techniques
After choosing the upgrade method you should do a test upgrade. This is just a preliminary test to catch any problems during the upgrade, and to rehearse the actual process.

After the upgrade evaluate how things went, improve your techniques and do it again. And again!

Evaluating also means troubleshooting problems, hunting for errors and validation of the result.

Review log files
To review the results of an upgrade, there’s several log files of interest:

- pre-upgrade checker log file (in 12/LOGS dir)
- psconfig log file (in 14/LOGS dir)
- upgrade log file (in 14/LOGS dir)
  - find most recent log and look for a given correlation id
- upgrade error log file (in 14/LOGS dir)

If you search for and find the phrase “Upgrade session finished successfully!” the upgrade was went well.
If the above entry was not found, search for ERROR and WARNING in upgrade log:

- ERROR indicates failures such as failing components and faulty database connections
- WARNING indicates issues such as missing features or components.

Warnings should not be ignored. They may not break your upgrade process, but warnings should be investigated so you know what the impact will be on your system.

**Review sites**

For individual WFE you can also try and run `stsadm -o localupgradestatus` to find out if sites were skipped. If this is the case, you should restart the upgrade process.

The before mentioned PS cmdlet `Test-SPContentDatabase` can also be used after an upgrade to validate if the content database has issues.

Verify that the sites actually work using a browser, do a search crawl of the site and verify the crawl log for issues.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since security scope has changed for deploying custom code in SharePoint, all test reviews should be done with a user with as low privileges as possible</td>
</tr>
</tbody>
</table>

**Reviewing artifacts**

A non-exhaustive list of things to check for when validating an upgraded site includes:

**Web parts**

- extra or missing web parts
- broken web part pages
- do they render correct
- are any pages still checked out

**Style and appearance**

- images display correctly
- CSS showing appropriately
- themes showing appropriately
- js working correctly (check for script errors)

**Permissions**

- does the appropriate people and groups still have correct permission level

**Customized (unghosted) pages**

- are customizations still in place
- should customizations still be there in upgraded farm

**Find issues early**

Finding issues early ensures a higher success rate for the upgrade - the earlier we detect the problems the better.

If you have multiple environments (as you should!), you can also use finding issues early to not repeat the problems found in test, in the subsequent environments such as integration test, pre-prod and prod, learning and improving the upgrade along the way.
2.1.4. Implement

Build/upgrade farms
First upgrade all farms to support the prerequisites for upgrade: upgrade to 64-bit, upgrade server OS, upgrade SQL server to supported versions, SP and CU, upgrade SharePoint to supported version, SP and CU.

The process of getting the servers in a supported state can be combined, as long as you don’t combine the prerequisite upgrade with the SP2010 upgrade.

Also upgrade hardware and build test farms.

Depending on the chosen upgrade model, upgrade the services and content databases accordingly.

Configure all valid settings, such as timer jobs, as recorded earlier.

Prefer scripted configurations over manual ones, to minimize human error and ensure consistency across platforms. This is the case both for OS installations and server installations.

Deploy customizations
Again depending on upgrade model, it might be necessary to deploy all or at least some customizations. Make sure this is done as solutions whenever possible to ensure a consistent deploy across WFE.

Minimize downtime
Make sure that the SQL server is up for the job. When upgrading to SP2010 SQL server quickly becomes a bottleneck, so make sure it has plenty of space and horsepower if you want to minimize the time it takes to do an upgrade.

Also consider making the content database read-only on the existing environment, while you upgrade a copy of this database in the background. Since SP2 SharePoint will detect that the database is read-only and trims the UI accordingly. This feature was added specifically with upgrade scenarios in mind!

If you are doing db-attach upgrade, upgrading content databases in parallel will reduce the time it takes to upgrade. It is also possible to upgrade in parallel to a temporary farm to make the upgrade even faster.

Monitor progress
Upgrade logs is now split up so that there’s only one upgrade log per session, and a separate log for errors, making it easier to see how the upgrade went.

The command line tools for upgrade now have status indicators that will visually show the progress of the upgrade. Also the upgrade status page in Central Administration (CA) tracks the progress and history of upgrades on the upgrade status page.
Use the above to ensure upgrade process is on schedule, and be ready to “pull the plug” on the upgrade if you can see you are running out of time for the upgrade and need to recover the old installation.

2.1.5. Validate
After upgrade is complete, you need to validate that the upgraded system really works. This means checking logs, checking rendering and checking that the database doesn’t have hidden issues.

Upgrade event failures
Reviewing the different logs associated with upgrade will give you a good indicator if everything really went fine. Look in the chapter Review log files above for more information.

If issues are found, find out how to fix it, and restart or resume the upgrade!

UI/UX issues
Visually checking the upgraded farm will tell you if some of the functionality developed for the old version of SharePoint needs to be redesigned to look properly or even to work in SP2010. This includes HTML, CSS and JS issues, but could also be XHTML compliance issues.

Also pages that fail to upgrade visually might be unghosted/customized in the old farm. You will then have to identify why the page was customized, determine if it is necessary to keep the customization, and then reghost the page in question.

Data issues
Check for orphaned items or database corruption using stsadms (see earlier chapter on orphaned items).

Other data related issues are connectivity issues to data sources. Check that these work where used.

2.2. Visual upgrade
By default the old look and feel of SP2007 and WSS3 is retained when doing an upgrade, but the site administrator has the ability to preview and change to the new SP2010 look and feel using the UI.

When doing a db attach upgrade using stsadms.exe, setting preserveolduserexperience switch to true | false will enforce the UI accordingly.

You could also automate the upgrade by utilizing PowerShell and/or the object model. For example using the SPSite. VisualUpgradeWebs method (consider including this code in a SPLongOperation since it, depending on the size of the site collection, could take a while to finish).

In the ONET.XML of a custom site definition the UIVersion attribute in the Project element can be set to 3 or 4 to enforce UI version.

2.3. No International Domain Name support
If you are upgrading a web content management site, and is using International Domain Names (IDN), it is worth mentioning that the support for IDN that was there in SP2007 was removed in SP2010!
The only reason I have heard for this is, that “Support of internationalized domain names (IDNs) has been deprecated”. Not sure how to interpret that, but the fact is that it no longer works in SP2010, so if you used this in SP2007 you will need to delete all IDN settings in your SP2007 farm before upgrading.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
</table>

### 3. UPGRADING SOLUTIONS AND CODE

#### 3.1. Recompilation

Existing code that utilizes object model and runs within IIS will continue to work without recompilation (if compiled for AnyCPU or 64-bit). As when upgrading from SPS 2003 to SharePoint 2007 the upgrade process inserts assembly binding redirects from old assemblies to new assemblies (here 12.0.0.0 to 14.0.0.0) making the code automatically redirect to the new SharePoint dll's.

Code that runs outside IIS and utilizes the object model (workflows, feature receivers, timer jobs etc.) will either need recompilation or binding redirects to work with SP2010.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>In SP2007 for a number of reasons it proved problematic to version assemblies. This often collided with both good development practice, and company rules on development lifecycles. These issues are well documented on the internet (including workarounds to get SPWebConfigModification class to add assembly redirects) so I won’t dig into that here. However the problem does no longer exist in SP2010 because that you now can specify assembly redirects directly in your solution manifest (see more below). It is not an automated process, so you will need to do it manually, but it is a lot easier than it was in SP2007, so developers should definitely consider using AssemblyVersion in code that is expected to have a long lifecycle!</td>
</tr>
</tbody>
</table>

#### 3.2. Upgrading Custom Site Definitions

As discussed in earlier chapter regarding visual upgrade, you can decide to keep the visuals as version 3 or you can decide to upgrade the visuals to version 4 adding the new layouts and tools such as the ribbon.

This choice also affects how you want to upgrade your Custom Site Definitions (CSD).

If you don’t plan on upgrading the visuals to version 4, most CSD should work as is, depending on how much is going on inside the CSD.

In other words if you only used a CSD to add a new artifacts or change the basic layout of pages, you might be better off by using a new SP2010 site definition as a basis for re-creating that same functionality in the upgraded farm, or as close as you can get. Then add upgrade logic to your feature (see upgrading features later in this chapter).

For more advanced scenarios, a better option would be to upgrade the functionality of the old site definition to match the new site definition. This involves changing the ONET.XML, since this has changed radically in the new version.
3.2.1. Upgrade definition files

The purpose of Upgrade Definition Files (UDF) is to transform existing sites customized in the previous version of the product to take advantage of features in the new version.

The UDF xml file maps custom lists, files and features from the old custom site definition to the new custom site definition during a schema or version upgrade.

Though there are major changes to the product from SP2007 to SP2010, the paradigm shift isn’t as big as from SPS2003 to SP2007, where the feature concept was introduced, making the ONET.XML contain noticeably smaller. Hence the UDF for this version will be less complex, and most of the times not needed at all, depending on what customizations were done in the custom site template.

The OOTB upgrade files for SP2010 can be found in 14\CONFIG\UPGRADE and can serve as a guide for upgrading your custom site definitions by selecting the site definition the custom site definition was based on. The custom UDF should be placed in the above mentioned folder and be given a unique name that begins with the name of the site definition (e.g. SPSNEWSCUSTOM_upgrade.xml).

Note:


3.3. Upgrading Solutions

There are a few noteworthy changes in Solution packages regarding upgrades.

In SP2007 it was tricky to add binding redirects in a consistent manner (SPWebConfigModification) since the runtime element is stored in another xml namespace (it could be done but it was tricky). Now this can be added declaratively as part of the solution manifest:

```xml
<Solution ...>
  <Assemblies>
    <Assembly DeploymentTarget="GlobalAssemblyCache" Location="MyWebPart.dll">
      <BindingRedirects>
        <BindingRedirect OldVersion="1.0.0.0" NewVersion="1.1.0.0" />
      </BindingRedirects>
    </Assembly>
  </Assemblies>
</Solution>
```

This will add an assembly binding element to the web.config files for the assembly in question, redirecting code that uses the old assembly to point to the new assembly.

Solutions can now also have dependencies declared in their manifest files. Three important things to note regarding solution dependencies though: solution dependencies does not automatically secure that dependent solutions are deployed. They just give you an error if you try to activate a solution that is dependent on another solution, and that solution isn’t deployed. Also you cannot have a farm based solution that is dependent on a user solution (Sandboxed solution). Last but not least: you will not receive any errors if you try and retract a solution that another solution is dependent on!
3.4. Versioned Features

Upgrading artifacts within features were always a pain-point in SP2007.

Upgrade scenarios for features in SP2007 would often mean adding a new dependent feature containing code in a feature call-out trying to change what needed to be changed in the feature.

The good news is that upgrading features has received some attention in the new version, so it is now possible to upgrade features both declaratively and programmatically. You can even declare branches for different actions depending on version, or have element manifests being applied at update only.

Up until now the version attribute in the Feature.xml manifest served no purpose. This has changed in the new version where the version attribute is used to detect if a given feature instance (SPFeature) needs to be upgraded. This is done by comparing the feature instance version with the feature definition (SPFeatureDefinition) version, hence securing that artifacts are identical whether it was just activated in a new version or upgraded from an old version.

The upgrade behavior can both be defined declaratively in the feature.xml and in an event triggered when a feature is upgraded.

3.4.1. Declarative feature upgrade

The declarative feature upgrade manifest contains an UpgradeActions element. Here you can declare actions that should only be applied for certain feature version ranges, including adding fields to content types, provision, move or rename files and more.

Currently the following elements can be placed inside an UpgradeActions element:

- VersionRange
- CustomUpgradeAction
- ApplyElementManifests
- AddContentTypeField
- MapFile

The optional VersionRange element gives you the opportunity to target feature upgrade to specific version ranges (e.g. between version 1.0.0.0 and version 1.2.0.0). The declarative logic specified inside the VersionRange elements will then only be executed if the version falls inside the version range. This gives you the ability to branch upgrades with different behavior for each version.

CustomUpgradeActions contains actions and parameters for custom code that is referenced in the UpgradeActions element (see more below under programmatic feature upgrade).

CustomUpgradeActions can both be placed inside the UpgradeActions element and inside the VersionRange element.

ApplyElementManifests is what you probably will use a lot when upgrading features: it will include an elements manifest that is only triggered on upgrade. This makes it easy to add new artifacts to an existing feature. The element can be placed under UpgradeActions or VersionRange elements.

The optional AddContentTypeField makes it possible to easily add new fields to existing content types. By adding a PushDown="TRUE" attribute to the element the change is pushed down from the site content types to every list content type. This was really a pain to do both declaratively and in code in SP2007, so that’s a really helpful change in SP2010.
MapFile can be used to move or rename files during feature upgrade.

### 3.4.2. Programmatic feature upgrade

The changes are not only declarative. There are several changes to the object model regarding feature upgrades.

For once there’s now a `FeatureUpgrading` event that gets called for each matching `VersionRange` when a feature is upgraded. You can pass parameters to this event declaratively through the `CustomUpgradeAction`.

New in SP2010 is also the Feature Upgrade Query Object Model. This can be used to query across farm to determine what features are installed and what versions they have, if they need to be upgraded, and then upgrade features accordingly. A `QueryFeature` method has been added to `SPSite`, `SPContentDatabase`, `SPWebApplication`, `SPWebService` and `SPAdministrationWebApplication` classes. These methods can be used to determine what features need upgrading in the relevant scope.

**Note:**


To do the actual upgrade you call the `Upgrade` method on a deployed feature (`SPFeature`) and have it update to a new version.

**Note:**

The `SPFeatureDefinition` class already contained a version property. New in SP2010 is that `SPFeature` also contains a version property. This version does not necessarily correspond to the `SPFeatureDefinition` version: Upgrading a feature definition does not upgrade the feature instance itself. You will can use the query object model to obtain feature instances that need to be upgraded and programmatically call `Upgrade()` to upgrade to the new version. Read more on `SPFeature` version property here [http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spfeature.version(office.14).aspx](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spfeature.version(office.14).aspx)

### 3.5. Customizations against deprecated/changed UI

Customizations done in Central Administration and SSP Admin UI will also have to be re-implemented.

Central Administration has been completely restructured, and SSP has been replaced completely, so configuration links won’t show up as expected.

Since the HTML and CSS has changed in the new versions, depending on the layout the customized pages will look different in the new UI, even if the UI was done carefully emulating the existing configuration pages using the same controls!

If these links are still needed, they should be moved prior to an upgrade.

For application pages consider changing the `MasterPageFile` attribute with the `DynamicMasterPageFile` attribute. This will make the application page reference the site master page rather than application.master.
3.6. Security changes

3.6.1. Web Parts
As with SP2007 ASP.NET web parts should be preferred. WSS web parts while being phased out are still supported, but there are really no good reasons to use them anymore:

Web Part Page Services Components (WPSC) that was part of WSS web parts would allow you to do client-side connections, but the new feature in SP2010 called Client Object Model exceeds anything WPSC would ever allow you to do. Also AJAX (including postbacks) is now natively supported. Other reasons to use SharePoint web part classes include part cache, but this can easily be solved in ASP.NET web parts using runtime cache.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
</table>

The Client Object Model is also the reason that new Cross-site scripting (XSS) safeguards have been implemented in SP2010. Properties in web parts that can be changed by contributors, combined with Client Object Model are a XSS risk. This is why custom properties in web parts now require at least Designer level (previously it only took Contributor level). The new XSS safeguards are the RequiresDesignerPermissionAttribute that can be applied to properties in web parts and SafeAgainstScript safe control. Both are designed to limit access to viewing and saving properties in web parts.

Note that all web parts are affected by these new security measures (including old SP2007 web parts). This means you should review existing web parts to check if this new restriction breaks functionality, validate the risk of XSS and evaluate if you can risk setting the SafeAgainstScript SafeControl to true (false is default!).

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSS Safeguard only affects shared web parts, not Personal or personalized properties.</td>
</tr>
</tbody>
</table>

3.6.2. Sandboxed Solutions
Sandboxed solutions are a new concept in SP2010. Sandboxed solutions address a common problem in SP2007: you would have farm administrators would like to keep their servers up and running with good response times, and secure from malicious code. But you would also have developers that were told to develop custom functionality. Testing code before deploying it to a farm is both time consuming and difficult. Even with several test levels such as unit tests, smoke tests, functional test, load test and integration test, you will often not discover problems with the code until it is too late: in your production environment.

Sandboxed solutions is a subset of a standard solution: it is limited both in regards of object model and performance to run within a process called User Code Service (SPUCWorkerProcess.exe) that runs within a very limited Code Access Security policy (wssUserCode.config, that should not be edited!) and only on selected servers in the farm. It also uses a limited subset of the SharePoint API (reflected in Visual Studio intellisense).

A solution that runs within the sandbox is monitored on an array of metrics such as CPU, queries to database, unhandled exceptions etc. You can set up quota limit that the code needs to stay within. If this quota is exceeded, warnings will go out to Operations and when a limit has been reached, the code is temporarily disabled. You can build custom solution validators that allows only certain types of artifacts (e.g. web parts) or code signed with specific signatures.
While this new concept makes a lot of sense, it also means that you need to be aware of this when you upgrade your existing solutions:

Code-wise you will need to review your solutions, so that they will still work within the solution sandbox, since sandboxed solutions run against a subset of the API and with a limited CAS policy, a lot of the stuff you did yesterday (like web service calls, or calling code that is not marked with AllowPartiallyTrustedCallers=True) will no longer work!

It is possible to make calls to the “real” API, but it requires you to move the code to what’s called a full trust proxy in a separate assembly that goes in the GAC, and call the proxy from the sandbox.

You can choose to ignore sandboxed solutions and just upgrade your old 2007 solutions as what is now called Farm Solutions, but all in all the concept of sandboxed solutions will need to be addressed before upgrading a farm. There are good reasons to use the sandbox, including improved security, better monitoring and in the end a more stable and better performing farm, and looking ahead, all new development that fall inside what can be achieved as sandboxed solutions should be developed as such!

With regard to architecture, it should be considered to dedicate server(s) to run sandboxed solutions further isolating custom code from the rest of the farm.

Note:

For more information on SandBoxed solutions check out:
http://blah.winsmarts.com/2009-12-
SharePoint_2010_Sandboxed_Solutions_The_Definitive_Guide.aspx

For more information on custom solution validators check out the API:

For more information on full trust proxies see the API:

3.7. Large List Throttling

There’s a new performance related feature in SP2010 called Large List Query Throttling: Queries that touch large lists will fail based on predefined thresholds set in CA.

There is a good chance that this could cause problems for legacy code, especially if development is being done as an administrative user! Also if development and test environment does not have realistic data volumes, code could fail without this being caught before deployment.

For this reason you should start developing against least privileges, and always try to have as realistic data as possible in your environment (for lists it would even make sense to have lists that are a lot larger than in production).

Even if you only select a small subset of items from a large list, the API and database still need to do a table scan to select the appropriate items. Hence a small query on a large list will be throttled and throw an exception. This can be resolved by adding an index on the list that matches the field that is used in CAML query to filter the list.

It is possible to override the Resource Throttling: SPQueryThrottleOption.Override if Object Model Override is set to Yes in CA and if the user executing the query has Full Read permissions.
3.8. Deprecated API’s

When you recompile your old SP2007 code for SP2010, you will see warnings for types and methods that have been deprecated in SP2010. Most of these will continue to work without breaking anything in SP2010, but you are encouraged over time to upgrade the code since Microsoft no longer will invest in these API’s.

### Note:

Get a list of deprecated types and methods made obsolete in SP2010 and SP2007 on MSDN: [http://code.msdn.microsoft.com/sps2010deprecated](http://code.msdn.microsoft.com/sps2010deprecated)

Chris Auld mentioned a plug-in for Reflector that would catch obsolete methods and warn against code that could have problems in SP2010, for example in relation to sandboxed solutions. Tool should become available at [http://www.syringe.net.nz/blog](http://www.syringe.net.nz/blog)

3.9. Hardcoding issues

If you have hardcoded references to anything residing in the old 12-hive (aka SharePoint root folder: c:\program files\common files\microsoft shared\web server extensions\12) these should be updated to point to the 14 folder instead!

### 3.10. Upgrading the look & feel to the new version

If you choose to go with the new visual upgrade like the Ribbon, developer dashboard etc., you need to manually add these controls to your master pages and page layouts.

After upgrading the solution to SP2010, in Site Settings > Site Collection Administration select Visual Upgrade > ”Apply the new User Interface to All Sites”. Click Update All Sites.

This will change the appearance to the new interface.

While the site settings page itself properly will upgrade without issues, but if you are using a custom site definition, you will need to manually replace the old UI controls:

Since SharePoint distinguish between v3 (SP2007) and v4 (SP2010) master pages (v3 master pages are filtered out in the standard Master page view), start by creating a new blank v4 master page using SharePoint Designer 2010 (SPD) and replace the content with the content of the v3 master page.

- Delete the page editing toolbar (PublishingConsole) tag prefix and associated controls
- Delete site action (PublishingSiteAction) tag prefix and associated controls (including the SPSecurityTrimmedControl wrapper control)
- Add core.js if not already present as a ScriptLink control
- Copy the ribbon DIV html and control (SPRibbon) from v4.master and paste it into the new master at the very top of the body (inside FORM element)
- Add register tag prefixes for ribbon (MUISelector)
- If you use breadcrumb control, this is contained in ribbon, so remove control and surrounding HTML from master
• Copy the developer dashboard control (DeveloperDashboard) from v4.master and insert it into the bottom of the body of the new master

**Note:**

Further customizations can be done (such as maintaining the position of the ribbon while scrolling) info on upgrading an existing master page to the SharePoint Foundation master page can be found on MSDN: [http://msdn.microsoft.com/en-us/library/ee539981(office.14).aspx](http://msdn.microsoft.com/en-us/library/ee539981(office.14).aspx)

---

### 3.11. Upgrading projects to Visual Studio 2010

Part of upgrading your code should be migrating from VS2005/VS2008 to VS2010. There are a bunch of new cool features for SharePoint in the new VS2010, so it is recommendable to upgrade existing projects to the new development platform. Also it will make upgrading existing code easier.

If your projects were created using VSeWSS you can download a VS2010 template that will upgrade your projects to VS2010 SharePoint projects. After migration you will need to manually consolidate your artifacts using Feature Designer and Packaging Explorer.

**Note:**


If you use other tools like STSDEV or WSPbuilder, you can consider a number of different more or less manual approaches:

The first approach is a manual approach where you basically build your project structure up manually importing code and artifacts as you go:

• First you need to evaluate what your visual studio projects contain.
  - If you have separated your different logic into tiers for data access, business logic and presentation, there is a good chance that these class library projects can be copied directly into VS2010.
  - For visual studio projects containing artifacts create an empty SP2010 project. Here you must choose between creating a sandboxed solution or a farm solution – the choice will depend on what customizations are done in the project, since sandboxed solutions put a lot of restrictions on what can be done. sandboxed solutions should be preferred, but will probably require a lot more effort on refactoring the code to keep within the sandbox boundaries.

• Use the new VS2010 feature called Mapped Folders to map the SharePoint root (aka 14-hive) folders you need for your project. Add your existing artifacts into the relevant folders
  - To take full advantage of VS2010 you can also create some of the artifacts (such as web parts) from scratch using the corresponding template and then copy/paste the code and declarative xml from your existing files.

• VS2010 now has a feature called Replaceable Parameters that basically are tokens that are replaced after manifest transformation. The tokens are extendable and include tokens for things like $Sharepoint.Project.AssemblyFullName$ Consider replacing
• For artifacts that need to be provisioned to document libraries you create Modules and add your existing content to the modules.

• Features can either be created manually or added through the feature Manifest Template (<featurename>.Template.xml). The features added through designer and Manifest Template is merged into a single manifest file for the feature.

• Add the artifacts to the Package (Package.package file in project folder) using Package Explorer or Package Designer.

• For artifacts currently not supported by VS2010 (for example custom site definitions) add the relevant xml from your existing manifest.xml files to Package.Template.xml that can be found nested under Package folder. Artifacts listed in Package.Template.xml are merged with Package artifacts during packaging into a single solution manifest file.

Note:

Read more on MSDN about the structure and files in SharePoint project types: http://msdn.microsoft.com/en-us/library/ee476619(VS.100).aspx#projectcomponents

To ease this manual process you can instead choose to import SharePoint solution packages (WSP) into VS2010 using the Import SharePoint Solution Package project type. As of now this template works best for simple WSP packages, but hopefully it will become better in the final release:

• First create a WSP file containing the artifacts you need to migrate to VS2010.

• Create a new Import SharePoint Solution Package project in VS2010 and select WSP file when asked.

• If not supported artifacts was contained in the WSP you might get a warning, but don’t count on it. The import still has a lot of beta hiccups, so for example custom site definitions disappear after an import and so does assemblies for CAG. I don’t know if this will be fixed for the final release, but still the tool is still a huge help when you want to convert existing projects.

Note:

Carsten Keutmann the author of WSPBuilder also has released a beta of WSPBuilder for VS2010. I haven’t had time so far to check this out, but it is available on Codeplex here: http://wspbuiilder.codeplex.com/releases/view/30858

The third way of importing a project would be to “roll your own” import tool. VS2010 has specific interfaces defined for creating extensions of various kinds. For example the ISharePointProjectFeature interface for adding items to features and the ISharePointProjectPackage to add items to packages.

Note:

Since the SharePoint Tools in VS2010 are extendable, we already see a lot of tools by the SharePoint Community. So far most notably Community Kit for SharePoint: Development Tools Edition that contains several enhancements focused on deployment, artifacts and more. CKS:DEV can be found on Codeplex: http://cksdev.codeplex.com/

There are a lot of good reasons to upgrade to VS2010 like F5 debugging, templates for specific tasks, native support for solutions and features, possibility to browse SharePoint sites using Server
Explorer. The list goes on! All this makes SharePoint development a much better experience than
developing in earlier versions of VS.

Note:

More info on importing WSP into VS2010 on Channel9:
For more info on what’s new in VS2010 with regards to SharePoint development read this

3.12. Client upgrades

Be aware that Internet Explorer (IE) 6 no longer is supported for authoring, due to its poor
interpretation of web standards. As part of an upgrade you should plan for upgrading to a
supported browser.

Note:


4. PLANNING

Now that the basics for upgrading SharePoint 2007 to SP2010 have been laid out both regarding
servers and code, it is time to think about what the specific actions should be when doing an
upgrade.

This chapter only contains general recommendations, as the approach will be dictated by
external factors such as if the company that pays for the upgrade is willing to buy new hardware
for either a full db attach upgrade or a hybrid approach involving new hardware.

Also the physical design of the solution, the size of the content databases, and the amount of
customization on the farm will affect the recommended approach, along with demands for
downtime.

4.1. Planning prerequisites

First thing that should be done is bringing the solution in a supported upgradable position.

This includes upgrading any OS used as SharePoint servers from 2003 to 2008 server R2 for all
involved servers.

Make sure SQL server is running 64-bit with latest SP and CU.

For SQL Server this is SP3 with CU3. Optionally consider upgrading to SQL server 2008 SP1 with CU
2 since SQL Server 2005 support lifecycle is terminated in 2011
(http://support.microsoft.com/lifecycle/?p1=2855) another reason for upgrading is
improvements from 2005 to 2008 including better compression, better encryption, higher
availability through improved patching capabilities, throttling and improved locking mitigating
blocking issues, better mirroring, support for Remote BLOB Storage etc. (read here
Also consider upgrading the OS that SQL server is running on to Windows 2008 Server, since no further service packs is considered for 2003 server (http://www.microsoft.com/windows/lifecycle/servicepacks.mspx).

When looking at upgrading the software, also consider upgrading or replacing existing hardware: The upgrade process itself will demand more hard disk space to instantiate a number of new databases, existing databases will grow and log files will take up space as well (transaction model is automatically set to Simple for databases though).

4.2. Planning upgrade model

When choosing the appropriate upgrade model, several things will affect your choice:

For example consider if the servers are already within specifications or if you can expect acceptable performance by upgrading the hardware (scale up)? If this is the case, this speaks for doing an in-place upgrade. On the contrary, if we already now can see that the existing hardware must be replaced, this will necessitate a db attach upgrade.

Another question to be asked are if you have scripted installs. If you don’t have this, this could speak for an in-place upgrade, rather than having to do a manual install, that is prone for human error.

Are customizations as a general rule structured and reproducible (read: solutions and features)? If not, this speaks for doing an in-place upgrade to avoid the process of reproducing customizations on a new server.

Also ask yourself what is acceptable downtime? If downtime is totally unacceptable, favor solutions that mitigate downtime, such as read-only databases.

As described in the chapter on upgrade models, there is also the possibility to choose a hybrid model. For example the read-only databases hybrid approach has a lot speaking for it, with regards to downtime mitigation.

In general in-place upgrade is considered risky, since you won’t be able to easily recover from a failed upgrade. If getting new hardware is out of the question for the upgrade, be sure you have a tested disaster recovery plan that will enable you to re-build your SP2007 farm if need be.

4.3. Planning new Server Architecture

Since the architecture on SP2010 has changed quite a lot compared to that of SP2007, you also need to take this into consideration when doing an upgrade.

Decide how the service architecture should be:

Should new server roles be added to the farm by adding new hardware or by combining roles on existing servers?

Would the farm architecture benefit from isolating certain services, since this is now possible in SP2010?

Default in SP2010 is that all services are disabled. This is good since it indicates that you should consider for each service if it should be enabled.

Sandboxed solutions

Consider isolating Sandboxed solutions on a separate server (remote mode). Remote mode is more scalable, but requires more administrative involvement.

Note:

Remote Binary Large Object Storage (RBS)

SP2007 used integrated storage architecture for Binary Large Objects (BLOB), meaning that the BLOB was stored in the content database along with the metadata.

As content databases grow, so does the time it takes to backup and restore data, hence affecting the Service Level Agreement (SLA) of the farm.

In SP2010, it is possible to store BLOB data separate from the content database using RBS. This allows for storing BLOBS on cheaper storage and has faster backup/restore from SQL server since metadata is stored separately from BLOB’s. RBS defines an interface that allows external BLOB storage providers to support it. In the time of writing there are 5 external providers that either already integrate to, or is in the process of writing providers to integrate to RBS: EMC², OpenText, NetApp, AvePoint and CommVault.

Note:

RBS should not be considered a silver bullet for keeping disaster recovery within SLA, but rather as a specific tool for a specific problem. Also consider that the whole backup/restore picture will be complicated by having to fetch data from several locations. The SQL Filestream RBS provider that SP2010 provides out of the box is supported by both SharePoint and SQL backup and recovery, but support for backup is up to the individual RBS provider.

RBS has several advantages over the existing alternative in SP2007, External BLOB Storage (EBS):

- It has a managed interface with a provider API
- The scope for setting up RBS is per content database, so you can configure one BLOB store provider for one content database and another BLOB store provider for another (in EBS you had farm scope).
- As a consequence of the above, you can have many providers with RBS, where EBS only supported one provider
- You can configure a RBS maintainer to support retention policies, detect orphans etc.
- RBS can be configured through the UI and using PowerShell
- You can migrate BLOBS from one store to another using PowerShell

Note:

Using RBS requires that SP2010 runs on SQL Server 2008 R2. The existing architecture in SP2007, called External BLOB Storage (EBS), is still supported in SP2010, but should be considered deprecated

4.4. Test, test, test

As described earlier, upgrading is very much a trial and error discipline. You cannot expect to upgrade a complex farm with lots of content, customizations and configurations perfect the first time. Even if this is possible, you have no way to tell how long the process would take.

Practicing the upgrade process documenting the farm and customizations along the way, will give you a much better gut feeling when you do the actual upgrade: You will have a good idea on what to do, since you already have done it plenty of times, you will have a certain degree of knowledge about the outcome of the upgrade, and even if something should go wrong, you have documentation ready to recover your old farm if need be.
Using virtual environments to replicate farm setup, where you test for issues after upgrade. If possible consider doing a pilot, where only part of the farm is upgraded and let end users test the site extensively for you with everyday usage.

4.5. Planning operations scheduling
Plan upgrade over a weekend. This will give you time to roll back if something breaks in the upgrade process.

A simple schedule can help you determine if you are on track or if you should consider rolling back the original site:

- Friday 18:00 start backups
- Saturday 0:00 start upgrade of content farm/databases
- Sunday 12:00 upgrade must be effectively complete, or rollback must begin
- Monday 06:00 environment must be up and running

Scheduling should also include a plan for operations staff that should be available during the actual upgrade.

4.6. Planning code upgrade approach
In parallel with the planning and trial upgrade of the farm, the development team should be looking at what to do with the existing customizations.

This could be done as a separate test upgrade, where solutions and features are installed on a test SP2010 environment and tested.

Some things to consider regarding existing solutions, features and code:

- Should code be migrated as farm solutions, or should an effort be made to convert the solutions to sandboxed solutions?
- Should obsolete namespaces, types and methods be addressed?
- When upgrading features consider using the new possibilities available (e.g. new fields in Content Types).
- Does code access large lists, or could lists grow outside specified throttling metrics? Treat code accordingly, and decide how to handle throttle exceptions.
- When reviewing solutions, features and code, think about if the functionality is still relevant - it could either have been replaced by OOTB functionality or the functionality it was addressing could have been removed from the platform (e.g. custom links in SSP).
- Code that run outside IIS should be recompiled with new SharePoint assemblies or binding redirects should be defined along with AssemblyVersion.
- Considering the wealth of new features in VS2010 for developing and deploying SharePoint code, migrating your projects to VS2010 should have a high priority.
4.7. Planning user adoption

Finally you should plan for your end users. SP2010 is an awesome product, but it is also huge and a lot of the ways things was done in SP2007 has changed in SP2010, especially when enabling visual upgrade. Examples include the Ribbon, new templates for Information Workers, and a new and vastly improved SharePoint Designer to mention a few.

Training your site administrators, designers and contributors will prove valuable before doing the actual upgrade ensuring end user adoption from the start.

Note:

There are a lot of online resources for end user training, a lot of who are free. The below link is an example of free online videos to train end users in SP2010:
http://www.point8020.com/SharePointEndUserTraining.aspx