THREE KEY TECHNIQUES FOR AMAZING OSD SUCCESS
Abstract
This whitepaper details three innovative techniques to maximize OSD success: ensure success during deployment, optimal application mapping, and forcing ConfigMgr to retry when appropriate. These additions to your OSD processes will make you even more successful.

The Author
Simon Burbidge who works as a 1E Consultant.
Other contributors include Troy Martin, Stephen Dunkley, Jerry Dismukes, Reto Egeter, Matt Albert.
Editors: Paul Thomsen, Su Kent
Overview

1E is pleased to share some key techniques that were used in a Windows XP to Windows 7 Operating System deployment (OSD) project that achieved what many would agree is amazing success. The project was for a regional United States bank on a tight deadline with 90,000 clients, many in the bank’s approximately 3700 branches. This was a challenging environment. However, the deployment achieved an Operating System migration rate of 5,000 computers per week, 3,000 of which were at the branches. Those counts are successful builds that were fully functional in terms of applications installed, with 41,000 happy users at the point when this paper was written (April 2014).

The project used Microsoft’s System Center 2007 Configuration Manager (ConfigMgr or SCCM) and 1E’s Nomad, Shopping, and AppClarity products.

Implementing an OSD project can be a bumpy ride, especially considering the plethora of requirements in terms of location, build types and the way organizations like to operate their systems management environment. One of the requirements in this case was for the ability to deploy the majority of build types at all locations. By build type we mean the application profile that identifies the function of the machine within the company. Considering the number of types and the number and dispersal of the locations to consider, utilizing T1 network links or worse, it was clear this really would be a challenge.

How does one measure OSD success? Core considerations are:

- What are the numbers?
- Are you migrating clients at a rate sufficient to meet the project deadline?
- Are you attaining the desired rate without ever compromising branch operations?
- Once a user’s workstation has been migrated, do they always have the applications installed that they need to carry out their function within the business?

In this case success was impressively achieved. The numbers and rates are large for any organization. The techniques described in this paper ensured that branch operations were not compromised and the workstations had the applications required. Here, we describe the key technical methods used that helped to achieve this level of OSD success.
THREE KEY TECHNIQUES FOR AMAZING OSD SUCCESS

Specifically, this paper addresses:

1. How to automatically confirm, at deployment time, that all the critical content needed is already available locally (precached). Missing applications would mean that, for some users, the migration was not a success. For this reason it is critical that in engineering the ConfigMgr deployment task sequence, we ensure that once a commitment is made to install the new image, we have every confidence that the deployment will be successful. But what if non-critical content is not available locally? How do we manage the situation in order to ensure branch integrity while offering the flexibility of allowing OS migration to proceed where appropriate?

2. How to determine which applications are needed by the users and how to provide alternatives in cases where the user needs allow less expensive solutions.

3. If there are application installation issues, how to ensure that application installations are a success during the migration by initiating installation retries and closely monitoring the process.

Details of how the 1E products were fully integrated into the deployment process is not in scope for this particular article. For further information please see www.1e.com/zero-touch-windows-migration/

Should you consider implementing such a solution for your Operating System deployments, contact us at info@1e.com, we would be pleased to work with you.
Three Critical OSD Challenges

Operating System deployments are always challenging for a large number and wide variety of reasons. OSD technical and project experts have solutions for most of those challenges but three issues are especially daunting:

1. Applications must be reinstalled in all locations, including the smallest and most remote, but how do you ensure they’re available when actually needed, as opposed to at earlier project preparation stages?

2. What applications are needed and where, and how can you automatically provide the users with the most appropriate applications, as opposed to possibly inappropriate overly expensive standard applications?

3. If applications fail to install on the first try for random environmental reasons, how can you enable smart retries?

This section defines those issues and later sections provide the strategies and details for how they were addressed.

Precaching

Operating System deployments require large volumes of Operating System image, device driver, application, and software update content. To keep the OSD installations reasonably fast, the content must be available on the same local area network (LAN) as the clients to be upgraded. That’s true in large centralized offices but also in smaller regional or even local offices where there may be only a few clients.

SCCM has options to provide content locally but many organizations find that deploying, managing, and maintaining all those SCCM servers (or server roles) is overly expensive and time consuming. As with this customer, many organizations use 1E’s software-only solution for content distribution, Nomad. Nomad’s architecture makes it easy to deploy and extremely reliable.

With or without Nomad, the best way to ensure content is available locally when OSD installations are done is to precache the content on the LANs. With dozens, hundreds, or even thousands of locations to be precached, and so many kinds of content, this is not a trivial process. Therefore OSD specialists will plan and implement precaching well ahead of the time the OSD installs are actually done.

This process should be centrally monitored, but to ensure that computers are upgraded fully successfully the content availability should be checked on each client as the OS upgrade starts. This is not an intrinsic SCCM feature so a solution is required to make these checks.
THREE KEY TECHNIQUES FOR AMAZING OSD SUCCESS

Mapped Applications

One of the most time consuming set of OSD project tasks is to identify applications that are installed in the organization, determine which are still being used, testing for Operating System compatibility, determining what should be made available (and to whom), and repackaging.

1E’s AppClarity allows you to focus on only those installed applications that matter most to users. It normalizes the vast volumes of inconsistent data available from ConfigMgr so that you can readily understand which applications you should provide to your users. This includes accounting for how all the applications are actually used.

Shopping, enables you to provide an enterprise app store for your users to request software and other resources. Shopping integrates with AppClarity so that the right applications can be automatically configured for installation during deployment with the minimum of effort from your administrators. This determining of the appropriate application is known as application mapping.

For this project “mapped applications” refers to Windows 7 packaged applications being mapped to corresponding legacy applications based on their actual usage. The mapping details are stored in the 1E Shopping database. Prior to deployment legacy applications are identified using AppClarity. You specify in Shopping that legacy applications with certain usage should be replaced by Windows 7 compatible or alternative applications. The list of Windows 7 mapped applications to install during deployment is then automatically obtained through a web service function call to the 1E Shopping website during execution of the native MDT Gather task sequence step.

For example, a client with Visio 2003 installed where the user has recently used the application would have Visio 2013 installed. If the user had not used Visio 2003 then nothing would be installed, saving that license cost. If the user had used Visio 2003 but not recently, then a Visio viewer would be installed, again with no licensing cost. If the user later needed to edit Visio documents, they could go to Shopping to get it automatically installed.

While application mapping had considerable benefits for the customer, the impact on the precaching strategy had to be considered.
Installation Retrying

The host SCCM environment of this organization was well resourced and effectively managed with the 90,000 clients being served by four network load balanced management point servers.

However, as the rate of migrations increased it was clear that the deployment process was having to compete with the other scheduled systems management activities to clients not targeted for migration, such as patch and package deployments. When migrations coincided with large scale update or package deployments, it was observed that a small, arbitrary number of application installations were being ‘skipped’ with no attempt to install being made by the task sequence despite it being apparent that the package was present in the Install Software step base variable.

The cause appeared to be contention when accessing policy from the MPs. Examination of the client logs revealed that the task sequence makes no attempt to retry download of package policy at the time the base variable is enumerated and installation attempted - if the initial attempt to communicate with the management point server is unsuccessful, the step immediately moves onto the next package in the list.

Given that the root cause of this issue is sporadic, there is good reason to hope that on a retry the installation would succeed. ConfigMgr task sequences do not attempt such retries so a solution had to be implemented.
Maximizing Precaching Success

The Requirements

User workstations at all locations consisted of both standardized builds where the application profile was well known in advance and machines that may have had an ad-hoc number of different business applications installed over time. The remit from the customer and challenge for 1E was to facilitate upgrading both standard and ad-hoc builds while ensuring that at completion the computer was fully functional for the user.

The migration plan implemented by 1E and the customer, meant that deployments were initially piloted with a small number of clients in order to prove that critical design requirements were satisfied e.g. operating the task sequence with their disk encryption software, Nomad task sequence integration and the ability of 1E Shopping to facilitate the installing of legacy or mapped applications during deployment.

At the same time, discussions took place with the customer around which direction the precaching strategy would take. Precaching was identified as a critical activity if we were to stand a chance of attaining the deployment rate demanded by the customer. Much effort was put into designing how this was going to proceed and how eventually, this was to be managed within the deployment task sequence itself. These discussions eventually spawned the project’s precaching methodology which is described here in this paper.

With the precaching requirements clearly defined we were able to push ahead with further development of the deployment and precaching task sequences. After continued piloting under all deployment scenarios proved successful, the move was made into production. Again, a conservative approach was initially used with the number of clients gradually increasing over time. This approach not only meant that confidence could be gained incrementally but was also aligned with the parallel effort to test and package the several hundred of user applications for Windows 7.

As stated previously 41,000 machines have completed migration at a rate, within the last month of 5,000 machines per week. A minimal number of deployment failures have been experienced, mainly caused by transient, external factors. Of the 3000 branches computers being migrated per week, the majority of these are machines that exist with a well-defined role within the business and require no additional applications installed above and beyond those assigned in the MDT Roles. However, this number also includes corporate (referred to as “Corp”) computers where any number of mapped applications could be up for installation during deployment through the use of 1E Shopping and AppClarity task sequence integration.
The importance of precaching illustrates that the project relied heavily on Nomad. Nomad’s ability to deploy and cache content far and wide to even the remotest of branches without overwhelming the network has been well proven.

The conundrum however, was that how could we leverage the product to ensure deployment success and in particular content availability past the point of (imaging) no return? We needed to perform a local pre-deployment cache check that took into account location and build type – prior to anything else happening in the task sequence. However, under certain circumstances the customer also required flexibility in terms of allowing key personnel to migrate their corporate computers even at branches.

**The Precaching Methodology**

Way before any deployments were scheduled, 1E and the customer undertook due diligence on the client estate in order to establish which Microsoft Deployment Toolkit (MDT) roles needed to be created, which applications were in use on workstations and which of these would need to be precached (cached using Nomad prior to execution of the deployment task sequence). This was not an insignificant undertaking.

Application packages were prepared. From there all we needed to do was to ensure that content was available where it was needed and check that this was indeed the case before starting any deployment.

In deciding what to precache, content was divided into four types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Precache approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced</td>
<td>Packages explicitly referenced in the deployment task sequence e.g. driver packages, OS image.</td>
<td>All referenced packages at all branch</td>
</tr>
<tr>
<td>MDT Role</td>
<td>Packages referenced in the MDT Roles defined for the organization. All computers will have at least one Role assigned.</td>
<td>All MDT Role packages at all branches</td>
</tr>
<tr>
<td>Strategic</td>
<td>Packages that are not referenced or defined within a MDT role, but are nevertheless vital to many users carrying out their assigned tasks e.g. Office 2013.</td>
<td>All strategic packages at all branches</td>
</tr>
<tr>
<td>Mapped</td>
<td>Applications identified as being used under the legacy OS that must be migrated to the new OS (installed during the deployment process). Installation is achieved using 1E Shopping integrated with AppClarity. Based upon usage data hosted by AppClarity, web functions in 1E Shopping spawned an Install Software type step base variable referenced within the deployment task sequence.</td>
<td>Specific mapped application packages at branches where requested by branch/ business unit managers.</td>
</tr>
</tbody>
</table>
THREE KEY TECHNIQUES FOR AMAZING OSD SUCCESS

With the precaching approach agreed and the applications identified, a precaching task sequence was created and advertised on a recurring schedule to customer workstations several weeks prior to migration.

Controlling the Task Sequence Behavior

With thousands of branch offices and other slow locations and hundreds of applications to consider, guaranteeing that all content is at every location ahead of deployment was difficult. Therefore, this needed to be managed within the task sequence so that it could be double-checked at deployment time, allowing the appropriate action to be taken depending upon the application/build type/location. To this end, the following task sequence design directives were agreed with the customer:

<table>
<thead>
<tr>
<th>Type</th>
<th>Location Type</th>
<th>TS Operation if Content not Precached</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced</td>
<td>Slow - Critical</td>
<td>Fail</td>
</tr>
<tr>
<td>MDT Role</td>
<td>Business critical</td>
<td>If any package was found not precached, then the deployment process will fail gracefully before the point of no return. A custom error message is generated allowing administrators to easily report on which packages were missing.</td>
</tr>
<tr>
<td>Strategic</td>
<td>locations consisting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mainly of branch</td>
<td></td>
</tr>
<tr>
<td></td>
<td>offices but also</td>
<td></td>
</tr>
<tr>
<td></td>
<td>includes some</td>
<td></td>
</tr>
<tr>
<td></td>
<td>server-less Corporate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>offices</td>
<td></td>
</tr>
<tr>
<td>Mapped</td>
<td></td>
<td>Fail or continue but installation to be managed</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
<td>Mapped applications are only ever installed on Corp machines. Precaching all mapped applications was not practicable and yet Corp migrations needed to take place at this type of location. As a mistimed Corp build could well bring down a branch, these builds needed to be very carefully managed. With this in mind, the following approach was to be implemented: At the task sequence pre-flight stage, if a Corp build type is detected within a slow-critical location, the deployment is allowed to continue only if the current time falls within a window designed to allow enough time for the build to complete during out-of-business hours e.g. 19:00 - 21:00. If the current time falls either side of the window, the task sequence errors out. Having established that the build can continue, whether un-cached mapped applications are downloaded by Nomad from the nearest distribution server and installed is controlled through a custom property in the MDT database FilterMappedApps*. The default is for un-cached content NOT to be downloaded. Setting of this property to allow the download of un-cached content would be the responsibility of operations staff assigned to that location in consultation with the Corp user/mapped application owner. The Mapped applications to be installed on any one particular machine can be viewed through the 1E Shopping web portal. Depending on the application profile of the target Corp machine, it may be that further precaching would need to take place before the scheduled deployment. *The term ‘filter’ is used, because setting this property to ‘True’ means those packages discovered as un-cached are then filtered out of the Install Software base variable before allowing the Install Software step to execute. As shown later in this document, Pre-provisioning and configuration of MDT records by customer operations staff is achieved through the 1E Provisioning web interface tool.</td>
</tr>
<tr>
<td>Referenced</td>
<td>Slow - Other</td>
<td>Fail</td>
</tr>
<tr>
<td>MDT Role</td>
<td></td>
<td>If any package was found not precached, then the deployment process will fail gracefully before the point of no return. A custom error message is generated allowing administrators to easily report on which packages were missing.</td>
</tr>
<tr>
<td>Strategic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapped</td>
<td></td>
<td>Continue but installation to be managed</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
<td>The task sequence will continue if mapped application are un-cached. Installation of the un-cached applications to be controlled through FilterMappedApps. Default is to NOT install.</td>
</tr>
</tbody>
</table>
**Task Sequence Implementation**

This section describes how the above decision process was implemented within the task sequence.

Figure 1 illustrates the process of checking that content is present within the build location before beginning image deployment. That process is:
THREE KEY TECHNIQUES FOR AMAZING OSD SUCCESS

Download the “Precache Check” package.

This is the key package and is used to host the files needed to perform all precache checks within this group. This package should have been pre-staged already at all locations, but Nomad will download from a peer cache or the nearest DP if required. The size of the package is kept to a minimum, about 200 KB.

Redirect the Precache Check package source location to Nomad cache.

We need the package path within the task sequence environment to point at the local Nomad cache. This allows us to run all package scripts locally.

Gather.

A “Gather” is performed (MDT database lookup) in order to identify the role-based packages for this machine. We use a custom gather step rather than the native MDT step, because we want to use the Precache Check step rather than the weighty MDT Toolkit package. We use the gather output later to check whether all role-based packages assigned to the machine have been cached locally.

Get the SCCM boundary Type and determine if this is a ‘Slow-Critical’ location.

This step makes a call to a custom web service in order to identify if the computer is in a SCCM fast or slow boundary location and if the site is Slow-Critical or not. The web service examines the ADSite name and from that is able to determine if it's a branch. If this is the case, then the site is automatically deemed Slow-Critical. If the site is not a branch, then it’s checked against those listed in LOCATIONS.INI. A match indicates that it is Slow-Critical. The findings of the service are passed back to the client in the format: <Boundary Type>:<Slow-Critical status> e.g. a response of 1:1 would indicates a Slow boundary location that is also of the critical type.

If the client is identified as being in a slow boundary, we cannot forgive any referenced, role, or strategic packages not being precached on the local machine or subnet and we need the deployment to error-out. Next, if this location is also Slow-Critical, then the current time is compared to the deployment window. If outside, then the deployment will also error out. The window default is set to 19:00-21:00. If any other start or finish time is required, this can be configured on the task sequence step command line.

Note: when the TS is running under the local OS, the script will obtain the local ADSite name from WMI (the SCCM client agent properties) and pass this in the web service.
call to obtain the SCCM boundary type. When under WinPE, the script will pass the
IPAddress and IPSubnet to the web service instead and the web service will perform a
lookup in AD for the corresponding ADSite name. The ADSite lookup in AD, depending
on loading can take several minutes to complete, hence the decision to only use this
method of identifying ADSite if we really have to. Scalability when performing WMI
queries against the SCCM site and ADSI in AD meant that a web service was the way to
go. (The web service can easily be adapted for customers who use IPSubnets or
IPAddress ranges for their site boundaries).

Update: 1E has since developed a “Single Site Download” (SSD) capability for Nomad
(released January 2013). This enables the client agent to copy content from peers
located in remote IPSubnets assigned to a common “site” (an Active Directory site or
similar specific location). This leverages cache information hosted in the 1E
ActiveEfficiency database. For organizations that use SSD, the solution would be
adjusted to incorporate this scenario as well. However, as the customer was running an
older version of the Nomad agent, this functionality was not included in the design.

Check MDT Precaching in Slow Boundary.

For a slow boundary location only, we want to ensure that appropriate packages are
present in the local cache or on the local subnet. The script responsible will use the 1E
Nomad utility PackageStatusRequest.exe to perform this task for each package: this
means referenced, Role and any (strategic) package listed in PRECACHE.INI. If the
script finds that any of the packages are missing, then this information is logged and
the deployment errors-out.

Update: When implementing Nomad SSD, PackageStatusRequest.exe is no longer
used. We will instead make a web service call to the 1E ActiveEfficiency web site.

Error Handling.

Normally the native task sequence error handling group would be sufficient, but this
utilizes the weighty MDT Toolkit package. We therefore again rely on the
PrecacheCheck package to do our error handling for us. In order for us to detect a
failure within the Precache Check group, the error handling group is conditioned on
the non-existence of a flag variable PRECACHED which is set in the last step of the
PrecacheCheck group. Error handling will perform the same job as the native group
with the addition of sending an additional status message indicating which packages
were not precached.

The above process ensures a satisfactory precache status for all referenced and role
based packages.
Managing Mapped Applications

From early project discussions it became apparent that application requirements were a bit of a black hole. Somewhat understandably (before the due diligence exercise gained momentum), no one knew anything about which applications were needed and thus would need to be precached nor where this should take place.

For this reason the decision was made to exclude mapped applications from the precache check described above with the view that this would be handled later in the task sequence State Restore phase.

Eventually, in managing mapped application and their precache status, the decision was made that for slow boundary locations, applications would not be permitted to install if found to be un-cached. It was felt that there were too many unknowns in terms of the number and size of package that may be downloaded. Having said that, the customer still wanted the flexibility of being able to override this behavior for certain key personnel. This functionality was built into the provisioning web portal developed by 1E and deployed with this customer.

Figure 2.

Note: 'Block' refers to un-cached mapped applications only

Figure 3 (see next page) shows how the requirements around mapped applications and precaching were implemented within the task sequence.

The APPLICATIONS base variable contains the list of mapped applications to install. If FILTERMAPPEDAPPS = True (the default) then for all slow boundary locations, the base variable will be filtered. Filtering involves checking the precache status of all packages in the list using PackageStatusRequest.exe.
Successes are added to the variable APPLICATIONS_Filtered. This is then referenced by the Install Software step. For all fast boundary locations, or where FILTERMAPPEDAPPS has been set to False, no precache check is performed and the task sequence is free to download content from the nearest distribution point via Nomad.
Enabling Application Installation Retrying

As mentioned earlier, application installation failures are often due to random environmental issues that will not recur. For this reason it is appropriate to retry the installation before completion of the task sequence in an attempt to affect a successful install. Missing mapped applications at the end of the build was considered a critical issue and as such close monitoring was put into place. To this end, Figure 4 shows how the task sequence was configured.

![Figure 4. Installation Retries](image-url)
The installation retry process is:

1. An initial attempt is made to install role, mapped (filtered or not filtered) and strategic packages (mapped applications that are subject to the initial Precache check) at the start of the State Restore phase.
2. At completion of Install Software, the SCCM client execution history registry key is checked against each of the above packages. If no key exists for a given package ID, then this is classed as a ‘skipped’ failure and added to the RETRYROLE or RETRYMAPPED base variable. If the package key is present, the _State value is retrieved. Any ‘failure’ is added to the retry base variable. This would likely indicate an issue with the package itself.
3. The task sequence continues on to complete installation of referenced packages (e.g. encryption and anti-virus software agents) and to restore user state.
4. Package installations are retried. If either of the retry base variables exist, another Install Software step is executed.
5. The SCCM client execution history is re-checked. Any remaining failed or skipped packages are inserted into a custom status message.

Figure 5 shows the Application Installation Failures List report output for a test deployment. The report leverages the custom message in order to indicate the status of all machines where one or more package failures have been experienced.

Figure 5.