Managing a Citrix Farm with ADS

andrew.wood@completability.com
graham.dekker@completability.com
April, 2006
Version 1.5

1. Abstract

One of the core goals of Server Based Computing (SBC) is to reduce the total cost of ownership of managing the desktop real estate. Whilst SBC removes complexity from the desktop, in reality it transfers this complexity to the data centre.

The needs of ongoing administration and management of an SBC environment are often overlooked. In organisations with large numbers of servers running multiple applications and services, managing the state of production builds is a time consuming and complex exercise. Managing a Citrix farm presents a variety of ongoing challenges. For example farm administrators need to:-

- Rapidly provision new servers or rebuild existing servers.
- Ensure consistency of server builds; irrespective of which server a user is directed to, the user experience should be consistent.
- Validate that changes and updates (application or otherwise) are easily rolled out and incorporated without compromising consistency or environment integrity.
- Maintain change management processes that are sufficiently robust to validate that existing server builds and newly provisioned servers are identical.

Design to manage, design for change.

This document describes how the authors created a robust, extensible and highly manageable Citrix farm environment, by using Microsoft's Automated Deployment Service (ADS)[1]. The document outlines major farm management issues and provides a summary of enhancements made to the ADS service to facilitate managing server deployment and maintenance. The document also provides the process for creating an unattended build deployment using ADS.

[1]It is the authors understanding that the Microsoft Automated Deployment Service will no longer continue as a standalone product but will be incorporated into Microsoft SMS in the near future.

2. Common Issues Managing Citrix Farms

Deploying a server based computing environment for a sizeable number of users and/or applications is often intended to reduce the total cost of ownership. However, the ongoing administration and management of such an environment is often overlooked in the design phase. There are a number of typical issues that characterise Citrix farm management, these are discussed as follows:-

Lack of a replicable standardised server build.

Servers may have a documented build, but a documented build-by-sheet process is prone to administrative error and is often becomes out of date. There may be a build image available for deployment, but the process to re-create or update that image is poorly understood, difficult and/or time consuming and is often limited to a particular device type and hardware configuration.

Lack of automated applications deployment and installation.

Applications are installed manually and therefore consistency is prone to administrative error. Applications can be contained within an image resulting in challenges in maintaining and updating the application. Or, no procedure is defined for adding or updating applications.

Lack of build update mechanism.

As requirements evolve changes to the platform are required. Changes may need to be performed manually and are therefore prone to error, or simply forgotten. No defined mechanism exists to ensure changes are easily introduced, and are automatically applied to existing and new server deployments.

Deployment work needs to take place after hours.

Server rebuilds, deploying additional servers, and updating applications needs to take place and requires attendance after hours so as to not affect the production network or service. This creates staffing challenges and costs.

Inconsistent server state.

Care and testing in a build process can be undone by administrators failing to keep maintenance procedures such as reboots, disk defragmentation and temporary file cleaning up-to-date. In a sizeable farm it is not uncommon for server states to become misaligned. This invariably leads to reliability and performance issues.

Lack of testing and development environments.

A test and development environment's primary function is to allow for structured development and validation of changes prior to deployment to production. Test and development environments are often incorporated as an afterthought, or deployed and then not kept current within the production environment. The challenge of maintaining the test and development environments can become overwhelming for busy administrators.

An in-effective test and development environment can result in the live platform deteriorating over time.

3. Principles of Deployment Design for Citrix Farms

3.1. Build Management

Consistency with Scripted Unattended

A standardised and scripted server build should be used. A scripted build is required to ensure identical behaviour of all Citrix servers in the farm. Even highly skilled and experienced administrators cannot perform a manual build that is consistent 100% of the time.

A scripted build is the foundation of automated server deployment and provisioning.

Versioning

Build customisation scripts should contain documented and version controlled configuration settings. Configuration should be modularised and recorded as part of the installation so that the server build can be interrogated to ascertain what configuration / build version is applied.

Versioning is the principle control mechanism for managing change.

Single Build

The server build should not be dependent on a given hardware configuration, the same build should support multiple hardware configurations so that only one build needs to be supported and maintained.

Design for Change

A typical standard server is built, configured, and left. Service pack and security updates may be applied as they become available, but in the main a standard server's build remains relatively static.

However, within a Citrix environment, which is essentially supporting a desktop service, a server build will need to accommodate more frequent change and yet, a greater level of robustness is required to ensure consistent user experience and service reliability.

The ability to facilitate updates should be incorporated into the deployment design from the outset. The update mechanism should ensure that the update is applied automatically to both existing servers and any new servers that are built.

Build Integrity

The build process should be verifiable and auditable. The result of each configuration step performed during the build should be logged. Any issues should be automatically flagged prior to an administrator placing a server into production.

Server State Integrity

Server state derives primarily from what is installed on the server. All updates should contain a signature that can be validated. A scheduled automated query should identify updates that are missing or should not be present, and alert accordingly.

Server state also derives from the sequence of process executions. Memory sets can, over time degrade, and become consumed thus compromising consistent user experience. Errant processes may hang and further tie up resources. A scheduled automated maintenance script should perform various maintenance tasks such as disk defragmentation, cleaning of the spool and temporary directories and then perform a reboot to ensure a clean server state.

3.2. Server Provisioning

Server Role

Within a Citrix farm environment, it is common to group applications into server 'silos'. This practice provides a number of benefits:

- Reducing the number of applications on a given server improves system integrity and stability
- Application conflicts are reduced / resolved
- Unreliable / badly behaving applications can be separated from business critical core applications
- Resource 'hogging' applications do not restrict the performance of other applications
- Change management scope of application changes and updates is reduced and only affects a subset of the farm

Additionally, the environment can consist of servers providing roles other than serving applications. For example:

- Dedicated data collector servers
- Web portal servers – i.e. Web Interface
- Support servers – i.e. file and print, database etc.

The deployment design must cater for each server role. To ensure consistency, the build deployment process should remain the same for all roles, with differences that a server's role mandates catered for appropriate actions by the installation scripts.

3.3. Application Provisioning

Consistency with Packaging

As with the core operating system build, an automated packaged application deployment ensures applications are installed and configured in a consistent manner.

Versioning

Application installation should not solely focus on deploying a silent installation process. The installation process should include versioning of the application installation package for use in reporting and update management.

Installation Integrity

The application installation should provide sufficient feedback allowing administrators to validate the success, or otherwise, of the application's installation.

Application Deployment with Build Deployment

Furthermore application deployment should be integrated with the build deployment. This allows server deployment to fulfil its promise of rapid service provisioning.

Application Updates

In a Citrix environment it is not unusual for a set of servers that are initially deployed to be updated with additional application packages or updates at a later date. Issues often arise as new servers are deployed either with the original scripted application, without the patches, or with patches applied in a different order.

Application provisioning design should readily accommodate new applications or application updates – so that both existing and newly deployed servers have applications installed in a consistent manner to ensure a stable user experience.

3.4. Environment Management

Environment Consistency

One of the key benefits of server based computing is the ability to rapidly deploy changes to the infrastructure. To ensure that the environment supports changes that are tested and validated before introducing into the live environment, supporting development infrastructure is critical.

Common practice is to perform development in an isolated development environment. Changes are then verified in a test environment prior to release into the commissioning/production environment.

A frequent challenge is maintaining consistency between environments. Whilst guidelines can be instigated to minimise the risk of this occurring, the amount of effort involved in ensuring the environments are identical can result in shortcuts being taken and therefore, over time, unwanted disparity can arise between the environments rendering the test and development environments ineffective.

The deployment design should ensure that the development and testing environments can be deployed as readily as the live production environment using exactly the same build version and process.

4. Using Microsoft ADS for Citrix Farm Management

4.1. Automation with Microsoft ADS

Microsoft's Automated Deployment Services (ADS) can be used to remotely and automatically deploy operating systems in the Microsoft Windows 2000 Server and Microsoft Windows Server 2003 families, as well as configure, maintain, and manage servers.

Other deployment solutions are available, however ADS was selected at the time of writing, as it was available at no additional cost free for users with a Windows 2003 Enterprise server license.

The procedures and processes written for ADS were intended to be transferable to other deployment solutions if required; as the goal is to achieve a deployment framework that facilitates the on-going management of a large farm environment.

The ADS Server and Application Deployment environment requires additional resources to compliment the core Microsoft ADS Service. These resources include:

- *Customised Automation Scripts and Utilities* which enable the automation of server deployment and device creation.
- *A Network Installation Point* to store software and utilities required in addition to the base operating system environment.
- *An ADS Deployment User* with appropriate administrative privileges to be utilised as a service account for the ADS agent that is used to enable configuration changes and software deployment to be launched on each device as well as add the server into the domain as required.

4.2. Microsoft ADS Components

The following sections offer a brief overview of the components that form the Microsoft ADS service.

4.2.1. Controller Service

The Controller service tracks computers, known as 'Devices' and enables an administrator to group them into 'Sets' and run administrative jobs on the sets. The Administration Agent, which handles communications with the Controller after the device is running a full operating system, is installed with the Controller by default.

4.2.2. Network Boot Services (NBS)

This component allows computers without an operating system to be remotely configured to run a selected operating system image. NBS consists of the ADS PXE Service and the Deployment Agent Builder Service.

4.2.3. Image Distribution Service

The Image Distribution service enables the capture and deployment of images locally or remotely. The Image Distribution service image store provides the storage location for images.

4.2.4. Administration Tools

Automated Deployment Services (ADS) includes the ADS Management snap-in, the Sequence Editor, and a set of command-line tools that enable administrators to locally and remotely manage ADS. Using the ADS Management snap-in or the command-line tools, you can monitor devices, group and manage sets of devices, run jobs and task sequences, and manage images.

4.2.5. Devices

A Device is a computer system that has the capacity to be controlled by the ADS Controller service.

Each Device in ADS has a device record in the ADS database. A device in ADS is either controlled or uncontrolled. A controlled device can accept commands from the Controller, while an uncontrolled device cannot.

4.2.6. Network Share Installation Points

A Network Share Installation point serves as a repository for the build image, applications and other resources required for deploying servers. Typically the Network Installation Points are hosted on the ADS server.

4.3. ADS Deployment Network

To facilitate service a dedicated deployment network is implemented. All devices using the ADS environment have at least two network cards; one bound to the live production network, and one bound to the deployment network.

This enables deployment of servers and applications during operational hours as impact on the live production service is minimised.

The Microsoft ADS controller requires a single, 100 megabits per second (Mbps) or faster network to connect to all devices. A DHCP server is also required to serve IP addresses to the deployment network.

4.4. ADS Customisation

This section describes how the ADS environment was customised to facilitate the objectives discussed thus far.

All devices (servers) managed by the ADS controller are added to the ADS database using a custom script (which is also capable of importing devices in batch mode). When devices are added additional properties are assigned to the device either with administrator assigned or, built-in default values. These customised properties are used by the various build and management scripts and processes referred to below.

4.4.1. Build Management

Consistency with Scripted Unattended Installation

The entire Citrix server build process is scripted using the winnt.sif unattended approach plus additional modularised installation, customisation and tuning scripts.

Versioning

The creation of the initial build image for deployment (see 'Creating a Build for ADS Deployment') is automated through a custom script. This includes the registration of the image with ADS with a given version number. When this image is deployed the version is written into the registry and onto the user's desktop background as part of the build installation.

This version number can then be used in a number of ways. Firstly, post-deployment the build uses the version number to query and determine any applicable build patches (see Design for Change). Secondly the build version (see Server State Integrity) is used as part of integrity checks to ensure the correct builds are in use.

Single Build

The use of device properties allows the use of a single build. The build installation leverages the device properties to ascertain the server role and therefore perform role-specific actions as well as the common single build installation. A single build can then be deployed either to a live production blade server, or to a development virtual server.

Design for Change

To facilitate rapid change for existing live servers as well as ensure any newly deployed servers are consistent a 'patching' system is in place.

The final action the build process performs is to check for current updates for the particular build version. A centralised xml file is maintained for each build version which contains a list of the updates that should be installed, their name and location, and the order of installation.

Therefore when a build modification is required, an update is developed and the centralised build image XML file is updated. A pre-created customised job is scheduled on the ADS controller to deploy the update to live servers. Any newly deployed servers using the original base build will automatically install the post-build updates. This means that a new server will have a build that is consistent with all servers deployed with the same build – both in terms of the original 'major' build version and all the subsequent minor build patches and hotfixes.

This approach ensures that introducing updates is not a laborious task.

Periodically updates are folded into a newer build image (at least yearly is recommended).

Build Integrity

When a build is deployed to a server the server is automatically added to its corresponding published applications. However to ensure the server is placed into production at a time when the administrator chooses, a 'disabling' load evaluator is applied to the server as part of the build process. When the administrator wishes to place the server into production a custom ADS task is used. Before the correct production load evaluator is applied by the 'enable' ADS task, a popup view of the build log is shown to the administrator for verification and approval.

Server State Integrity

A nightly task is scheduled to perform a server reboot after cleaning the spooler folder, removing any unused profiles from the profile directory and undertaking a system drive defragmentation.

Also scheduled nightly, is an integrity check that registry stamps are present on the server for each of the applications and build updates specified in the centralised application and build update XML files. Any anomalies are written to the server's event log for retrieval by the monitoring and alerting framework.

4.4.2. Server Provisioning

Server Roles

The ADS service is extended to include the ability to assign all servers an administrator defined 'role'. This role is defined as part of the task to add a device to ADS. The build image build includes all options for each of the different roles in the farm. The build process queries the server role, and installs appropriate components accordingly.

All server roles within the farm can then be assured of having a common and consistent core operating system build.

When a server is added to the ADS environment, administrators also define which silo the server will belong to. A silo is not only defined by what applications are installed but also what Citrix Management Console (CMC) server folder it resides in, and what CMC application folder the silo's corresponding published applications reside in. The build process then uses the ADS device 'role' property to determine which CMC folders to place the server in and therefore which published applications to add the server to.

4.4.3. Application Provisioning

Consistency with Packaging

Every application installation will require a number of different steps to install successfully. To ensure consistency of approach, an application factory packaging guide is used as a step-by-step guide to the process of packaging applications. This packaging guide sets out naming conventions, standardised registry stamping and application locations.

Versioning

The above methodology introduces a versioning system not only for the application but for the application package also.

When the application is installed the version number of the package is written to the registry. This is then used by the automated nightly integrity verification task to ensure that the expected application packages are on all servers.

This version number can then be used in a number of ways. Firstly, post-deployment, the build uses the version number to query and determine any applicable build patches (see Design for Change). Secondly, the build version (see Server State Integrity) is used as part of integrity checks to ensure the correct builds are in use.

Installation Integrity

Applications are installed by an ADS task in sequence according to the silo the server belongs to. Feedback of the installation is presented back to the ADS server console so that the administrator can be certain that the applications have installed correctly.

Application Deployment with Build Deployment

ADS tasks are available for both server build and application installation.

Application Updates

In the same manner as build updates, application updates are easily implemented. A custom ADS deployment task was developed to allow application deployment. Application updates are packaged and amended to the centralised application list xml file for any given silo. The custom ADS application deployment task queries this xml file and installs any application that is not present. The same xml file is used in the initial server deployment as in provisioning application updates. In this way newly built servers are 100% consistent with updated live servers.

4.4.4. Environment Management

Environment Consistency

Within the ADS environment, each servers is assigned a *FarmEnvironment* value. This value describes the environment the server belongs i.e. 'Live' or 'Test' etc. and is implemented by the custom add device script. The value of this property is utilised by the scripted build to ensure that a server is provisioned into the correct environment – be it 'Live', 'Test' or some other value as required and defined by an administrator.

Differences between environments can include domain membership, OU placement, different Citrix farms, and different data stores. This is fully catered for by the build process.

This ensures that, for example, a testing environment can be deployed as readily as a live production environment using exactly the deployment platform (if required), the same build version and the same deployment process.

4.4.5. Additional Customisations

Additional customisations were made to leverage ADS to allow administrators to perform common tasks that were not available within the CMC. Some of these are referred to previously.

Server Activation / Deactivation

Occasionally administrators may need to make servers unavailable to users for maintenance or fault resolution. A custom ADS task allows a server or a set of servers to be instantaneously removed or re-added to operation by applying or removing a 'de-activating' load evaluator.

Print Drivers

For consistency and ease of management ADS was chosen as the mechanism to deploy print drivers to the environment. A custom ADS task is available to deploy printer drivers specified in a centralised xml file. Adding a print driver to the environment is simply a case of copying the driver files to a central location, updating the print driver list xml file and performing the print driver deployment ADS task.

Run Task

A custom ADS task was developed to allow administrators to perform any custom action they require on any group of servers.

5. Creating a Build for ADS Deployment

5.1. Creating a Base Unattended Installation

This process creates a disk partition on the ADS existing server. This partition hosts an unattended installation setup i.e. unattended.txt files, OS install files, as well as additional network/source code information. An image of this installation partition is then captured and stored in ADS. The image is then deployed to a Device and used in the creation and installation of a working server.

Note: This process is not documented within the ADS help file - for further information consult MS KB 820758.

The assumptions for this process are:

a) The ADS services are up and running on the network.
b) The ADS server has available a dedicated disk partition of @ 800Mb – this partition should not be the initial boot partition.
c) The Windows 2003+SP1 source media is available via disc or network share.

The Unattended Installation configuration process is as follows:

1. Validate the installation process on the target host device by performing a manual install. Confirm the following:
    a. Does the host require OEM raid/device driver information?
    b. Does the host require OEM network drivers?

2. On the ADS Server, create an 800 megabyte (MB) partition and format it with the NTFS file system. Make sure that the partition that you create is assigned a drive letter. For example, on a server that is already running Windows Server 2003, create an additional NTFS partition and assign it the E drive letter. **Note:** The volume label of this system drive will become the volume label of the system drive volume of the new server.

3. Validate that the ADS Template unattended installation file 'c:\program files\microsoft ads\adsunattended.txt' is available. This is a dummy template file to allow the initial installation of the base unattended installation.

4. Open a command prompt, change to the folder or the network share that contains the Windows Server 2003 source files, type the following command, where *PathOfAnswerFile* is the path of the unattended Setup answer file, and then press ENTER:

```
Winnt32.exe /noreboot /syspart:Drive:
/tempdrive:Drive: /unattend:PathOfAnswerFile
```

For Example: With a configured 800MB unattended installation partition on drive E and with the installation directory of f:\windist the command line would be:-

```
F:\windist \Winnt32.exe /noreboot /syspart:E:
/tempdrive:E: /unattend:c:\program files\microsoft
ads\adsunattend.txt
```

This will launch a windows installation process similar to the screen shot shown in the following example. This process is now creating a setup partition on drive E:

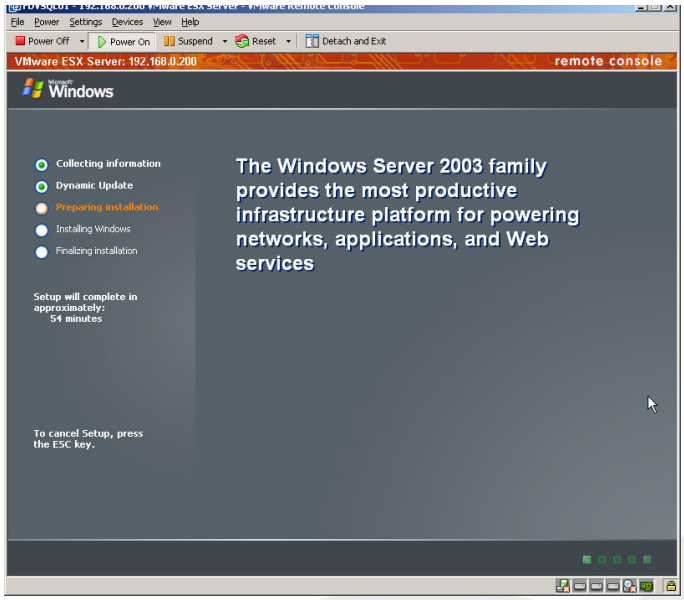

Figure 5-1 Screen Shot of the WinNT32 Installation Process

The Setup Partition will have the following files and folders:

- \WIN_NT.~BT
- \ WIN_NT.~LS
- \ntldr
- \txtsetup.sif
- \boot.ini
- \ntdetect.com

5.2. Customising and Capturing the Base OS Setup

1. Locate the Win_nt.~bt folder, and then rename the Migrate.inf file to Omigrate.inf if it exists.
   When you rename the Migrate.inf file to Omigrate.inf, information that is contained in the partition that you created in step 6.1 is not carried over to the devices. The partition that you created in step 6.1 now contains the installation files that are needed to install Windows Server 2003.

2. Copy the ADS template adswinnt.sif file from the `c:\program files\microsoft ads` directory to the \Win_nt.~bt folder and rename the file to winnt.sif. This ensures that FC custom actions are correctly interpreted during an unattended installation.

   Verify that the domain username and passwords, and product code are correct in the winnt.sif file.

3. Modify the Base Unattended Installation to incorporate customisation options for deployment using ADS Server.
   a. Create a directory under the `\$WIN_NT$.~LS\$OEM$` directory called $1. This will allow the creation of a custom folder structure for source scripts and installation files on the target device. Create a 'Drivers' , a 'BuildStore' and a Windows directory within `\$WIN_NT$.~LS\$OEM$\$1`.
   b. Copy the additional ADS Customisation scripts and files to `\$WIN_NT$.~LS\$OEM$\$1\BuildStore`. If OEM drivers are required, copy OEM Drivers to subdirectories within \$WIN_NT$.~LS\$OEM$\$1\Drivers. Make sure that the winnt.sif file has the following lines in the [Unattended] section:

      OemPreinstall=Yes

      OemPnPDriversPath="drivers\intel_lan;drivers\adaptec_scsi"

4. Use the adsimage.wsf script to capture the ADS image. An example command line would be:

```
adsimage.wsf /add "Image name" /path "D:\~w2k3ads.img""
/description "Image Description"
```

5. The image is now ready for deployment.

6. Conclusion

To allow effective management of large Citrix environments a well planned deployment infrastructure and design is vital.

Administrators should be able to perform common tasks easily and robustly.

Today's organisations require platforms that grow and evolve. Deployment design should cater for this. 'Design for change' should be a cornerstone of deployment design to ensure integrity, reliability and availability of the service.

The authors of this document believe that at the heart of every quality Citrix infrastructure is - a proper and effective deployment design.