PAUL TECHNOLOGIES

XenAppAudit User Guide

XenApp Farm Audit and Reporting Utility, Version 1.0

Published February 3, 2012

www.paultechnologies.com

Disclaimer

This software application is provided to you "as is" with no representations, warranties or conditions of any kind. You may use and distribute it at your own risk. Andy Paul and Paul Technologies disclaims all warranties whatsoever, express, implied, written, oral or statutory, including without limitation warranties of merchantability, fitness for a particular purpose, title and non-infringement. Without limiting the generality of the foregoing, you acknowledge and agree that (a) the software application may exhibit errors, design flaws or other problems, possibly resulting in loss of data or damage to property; (b) it may not be possible to make the software application fully functional; and (c) Paul Technologies may, without notice or liability to you, cease to make available the current version and/or any future versions of the software application. In no event should the code be used to support of ultra-hazardous activities, including but not limited to life support or blasting activities.

Neither Paul Technologies nor its affiliates or agents will be liable, under breach of contract or any other theory of liability, for any damages whatsoever arising from use of the software application, including without limitation direct, special, incidental, punitive, consequential or other damages, even if advised of the possibility of such damages. You agree to indemnify and defend Andy Paul and Paul Technologies against any and all claims arising from your use, modification or distribution of the code.

USE AT YOUR OWN RISK.

Table of Contents

Disclaimer **2**

Quick Start Guide **4**

- About XenAppAudit 4
- How to use XenAppAudit 4

All about XenAppAudit **5**

Running XenAppAudit **5**

- Requirements 5
- Starting the Data Collection 6
- Launcher Script 6
- XenAppAudit VBScript 7
- XenAppAudit PowerShell Script 8

Using the XenAppAudit Program **9**

- Requirements 9
- Starting the XenAppAudit Program 9
- Import/Export Tab 10
- Applications Tab 13
- Servers Tab 14
- Farm Settings Tab 16
- Running Reports 18

Credits **19**

About Andy **19**

Appendix A – Frequently Asked Questions **20**

Appendix B – Programming Code **23**

- START.BAT 23
- LAUNCHER.VBS 23
- XENAPPAUDIT.WSF 27
- XENAPPAUDIT.PS1 41

Appendix C – Program Workflows **56**

- Start & Launcher Workflow 56
- XenAppAudit.WSF Workflow 57
- XenAppAudit.PS1 Workflow 58

Quick Start Guide

About XenAppAudit

The purpose of the XenAppAudit Utility is to take XenApp audit data collection from days down to minutes. The XenAppAudit Utility provides the following:

- Automated data collection for Citrix XenApp Farms (versions 4 – 6.5)
    - Utilizing VBSCRIPT, MFCOM, WMI, and POWERSHELL scripts
- Data Analysis using Microsoft Access
    - Graphical User Interface
    - Custom data conversion functions
    - Summary Reports for Analysis
- Excel Spreadsheets for Additional Summary Analysis

How to use XenAppAudit

The XenAppAudit data collection scripts are designed to be run on a XenApp server.

- Extract the files to **C:\Program Files\XenAppAudit**
    - The program is tested to run from C:\Program Files\XenAppAudit, however, it should run from any path
- You will need to use an account with FARM ADMINISTRATOR privileges
- Your account should have SERVER ADMINISTRATOR rights as well for remote WMI calls
- Run **START.BAT**
    - If using Windows 2008 with UAC, be sure to RUN AS ADMINISTRATOR
    - For convenience, I recommend opening a CMD window as ADMINSTATOR, then running the batch file
- The START.BAT file will call the LAUNCHER.VBS script.
    - This script will inspect the system to determine OS and XenApp Versions
    - This script will call the associated VB or PowerShell scripts for XenApp data collection
- All scripts are located under the SCRIPTS folder
- Results are available under the RESULTS folder
    - XenAppAudit_be.mdb file containing all collected data
    - Various text files from command line queries
    - Transaction log files of the executed scripts
- Once the audit is complete, open the **XenAppAudit.mdb** (Microsoft Access is required)
    - This file should be in C:\Program Files\XenAppAudit
    - This database is linked to tables in XenAppAudit_be.mdb in the Results folder
    - Run the **Import Data Collection Files for Analysis**
        - This will import the text files from the Results folder
    - You can review the canned reports as well as export formatted to Excel for further analysis or distribution

All about XenAppAudit

This application is my own home-grown XenApp Audit utility. Throughout my tenure as a consultant, I am regularly engaged in performing audits and health checks of existing environments. As much as I despise documentation (don't we all), I found myself running the same queries and checks over and over again.

I have leveraged my limited background as a programmer to create this utility using a mix of batch files, VBScript files, PowerShell commands, MFCOM, and Microsoft Access. The idea is to have a utility I can run in any environment and retrieve the primary information I need to audit the environment.

Once this information is collected, I have created several canned reports and queries to help turn the raw data into usable information. I have seen my "data collection" time reduced from 1-2 days to less than 1 hour in most environments. This allows me more time for question and answer sessions with customers while having empirical data captured to support findings. Overall, I feel I can deliver a better analysis thanks to this tool.

I figured if it helps me, it can help you too! This tool is FREE for anyone to use (or modify) as necessary. There is no guarantee or warranty of any type. If you do distribute or modify, please be sure to give credit where credit is due.

Running XenAppAudit

Requirements

The XenAppAudit utility is designed with two parts:

- Data collection scripts used to populate the backend database
- Front-end Access application used to analyze the data

The data collection scripts are designed to run on a XenApp server. These scripts can be modified and may be able to be run from a remote system, but that has not been tested. The front-end application can run on the XenApp server or a local workstation, but it requires Microsoft Access. The linked tables are fixed to C:\Program Files\XenAppAudit\Results\XenAppAudit_be.mdb, but they can be updated using the Linked Table Manager in Access if necessary.

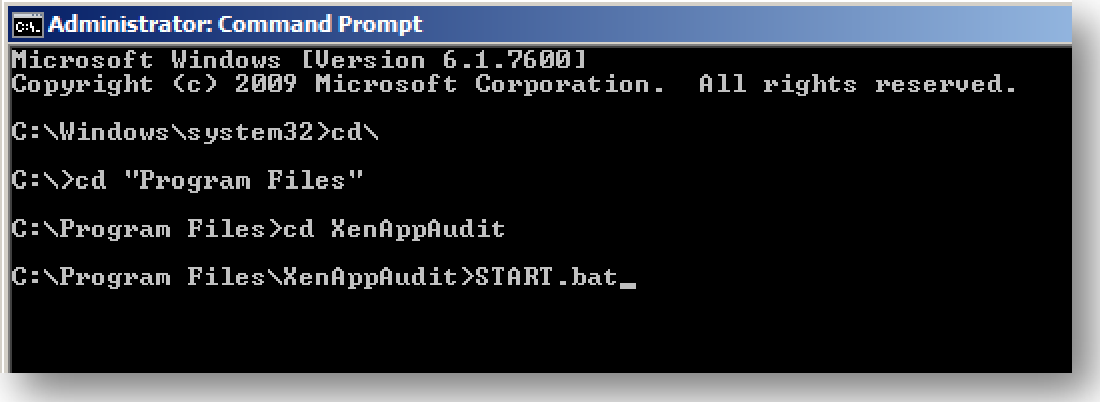
For XenApp 5 and earlier versions, the data collection scripts are MFCOM based and have no special requirements. For XenApp 6 or later, the data collection scripts are PowerShell base and require the Citrix XenApp PowerShell SDK to be downloaded and installed on the XenApp server. The SDK is available from the Citrix web site.

All files can be downloaded from http://paultechnologies.com/xenappaudit/ as a ZIP file. This ZIP file should be extracted to **C:\Program Files\XenAppAudit\.**

Starting the Data Collection

The user account running the data collection should be both a server administrator to enable the remove WMI data calls and a farm administrator in order to enumerate all of the Citrix objects. When running on Windows 2008 with UAC, the scripts should be run as an administrator. For convenience, you can open a command window as administrator, and then run the START.BAT file. This provides the added benefit of watching the scripts in real time.

The initial file to run is START.BAT. This batch file will first call the launcher.vbs script. Once that script is complete, the QueryDS command will execute. The full text of the batch file is available in Appendix B.



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd\

C:\>cd "Program Files"

C:\Program Files>cd XenAppAudit

C:\Program Files\XenAppAudit>START.bat_
```

Launcher Script

The START.BAT file will call the LAUNCHER.VBS script. This script contains logic to determine both the operating system and XenApp version of the server. Based upon this information, the script will in turn call the proper XenApp data collection scripts.

The LAUNCHER script will write all activities to Transcript_Launcher.txt under the RESULTS folder. This script will check the PROCESSOR_ARCHITECTURE environment variable to determine x86 or x64 architecture. Based on the architecture, the script will then check the file version of wfshell to determine the XenApp version.

Based upon the XenApp version and the architecture, the launcher script will then call the proper XenApp data collection script – either XenAppAudit.vbs (XenApp 5 or lower) or XenAppAudit.ps1 (XenApp 6 or later.) Once the data collection subroutine is complete, the launcher script will then perform a series of command line queries and export the data to text files under the RESULTS folder.

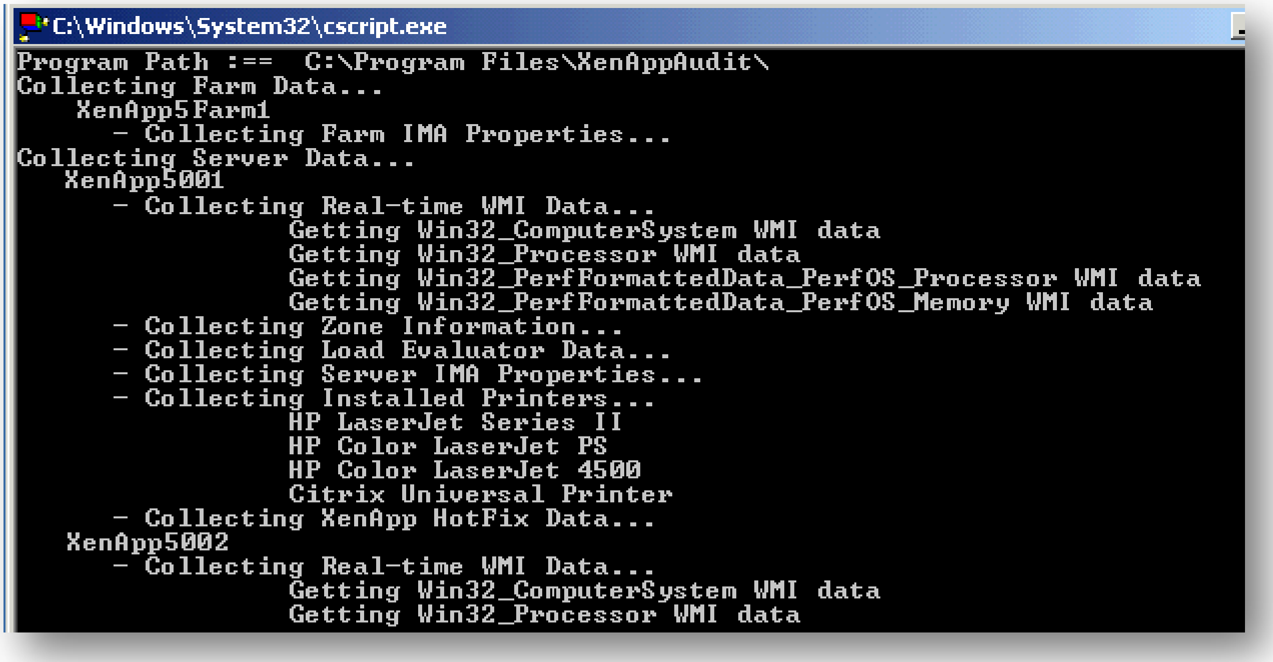
The full text of the script is available in Appendix B; the program logic workflow is available in Appendix C.

XenAppAudit VBScript

The XenAppAudit.WBS script is a vbscript used to call MFCOM objects to document the XenApp Farm settings and record the information in the XenAppAudit_be.mdb database under the RESULTS folder. Access is not required since this script use a Jet Engine connection to the database, which is built into the Windows operating system. The script also records all activities to the Transcript_VBS.txt file located in the RESULTS folder.

This script is used for XenApp 5.0 and previous editions. The script will also make remote WMI calls to each server to gather real-time performance and configuration data. The script captures the following types of information:

- Farm Settings
- Server Settings
    - Printer Drivers on each Server
    - Hotfixes on each Server
    - User Processes running on each Server
- Application Settings
    - Server Assignments for each Application
    - User Assignments for each Application
    - Group Assignments for each Application
- XenApp Policies
- XenApp Administrators
- Load Evaluators
- User Sessions



```
Program Path :==  C:\Program Files\XenAppAudit\
Collecting Farm Data...
    XenApp5Farm1
      - Collecting Farm IMA Properties...
Collecting Server Data...
   XenApp5001
      - Collecting Real-time WMI Data...
                 Getting Win32_ComputerSystem WMI data
                 Getting Win32_Processor WMI data
                 Getting Win32_PerfFormattedData_PerfOS_Processor WMI data
                 Getting Win32_PerfFormattedData_PerfOS_Memory WMI data
      - Collecting Zone Information...
      - Collecting Load Evaluator Data...
      - Collecting Server IMA Properties...
      - Collecting Installed Printers...
                 HP LaserJet Series II
                 HP Color LaserJet PS
                 HP Color LaserJet 4500
                 Citrix Universal Printer
      - Collecting XenApp HotFix Data...
   XenApp5002
      - Collecting Real-time WMI Data...
                 Getting Win32_ComputerSystem WMI data
                 Getting Win32_Processor WMI data
```

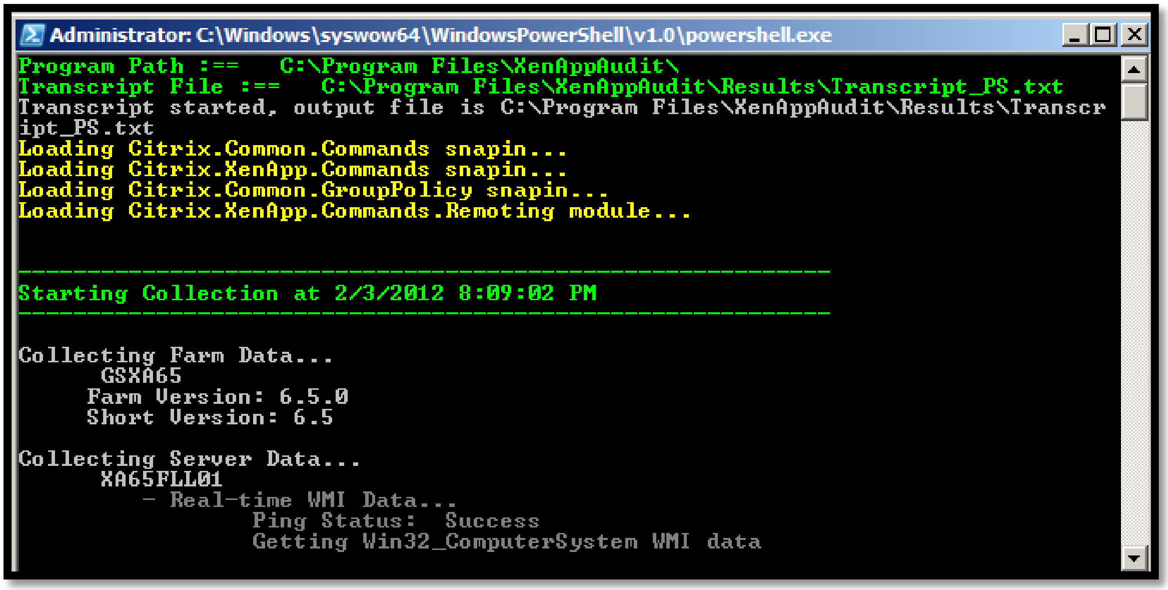
The full text of the script is available in Appendix B; the program logic workflow is available in Appendix C.

XenAppAudit PowerShell Script

The XenAppAudit.PS1 script is a PowerShell script used to document the XenApp Farm settings and record the information in the XenAppAudit_be.mdb database under the RESULTS folder. This script uses the Citrix XenApp PowerShell snap-ins available from the SDK. Access is not required since this script use a Jet Engine connection to the database, which is built into the Windows operating system. The script also records all activities to the Transcript_PS.txt file located in the RESULTS folder.

This script is used for XenApp 6.0 and 6.5 editions. The script will also make remote WMI calls to each server to gather real-time performance and configuration data. The script captures the following types of information:

- Farm Settings
- Server Settings
    - Printer Drivers on each Server
    - Hotfixes on each Server
    - User Processes running on each Server
- Application Settings
    - Server Assignments for each Application
    - User/Group Assignments for each Application
    - Worker Group Assignments for each Application
- Worker Group Settings
    - Server Name Assignments
    - Server Group Assignments
    - OU Assignments
- XenApp Administrators
- Load Evaluators
- User Sessions


Administrator: C:\Windows\syswow64\WindowsPowerShell\v1.0\powershell.exe

```
Program Path :==   C:\Program Files\XenAppAudit\
Transcript File :==   C:\Program Files\XenAppAudit\Results\Transcript_PS.txt
Transcript started, output file is C:\Program Files\XenAppAudit\Results\Transcr
ipt_PS.txt
Loading Citrix.Common.Commands snapin...
Loading Citrix.XenApp.Commands snapin...
Loading Citrix.Common.GroupPolicy snapin...
Loading Citrix.XenApp.Commands.Remoting module...


------------------------------------------------------------
Starting Collection at 2/3/2012 8:09:02 PM
------------------------------------------------------------

Collecting Farm Data...
      GSXA65
     Farm Version: 6.5.0
     Short Version: 6.5

Collecting Server Data...
      XA65FLL01
         - Real-time WMI Data...
                 Ping Status:  Success
                 Getting Win32_ComputerSystem WMI data
```

The full text of the script is available in Appendix B; the program logic workflow is available in Appendix C.

Using the XenAppAudit Program

Requirements

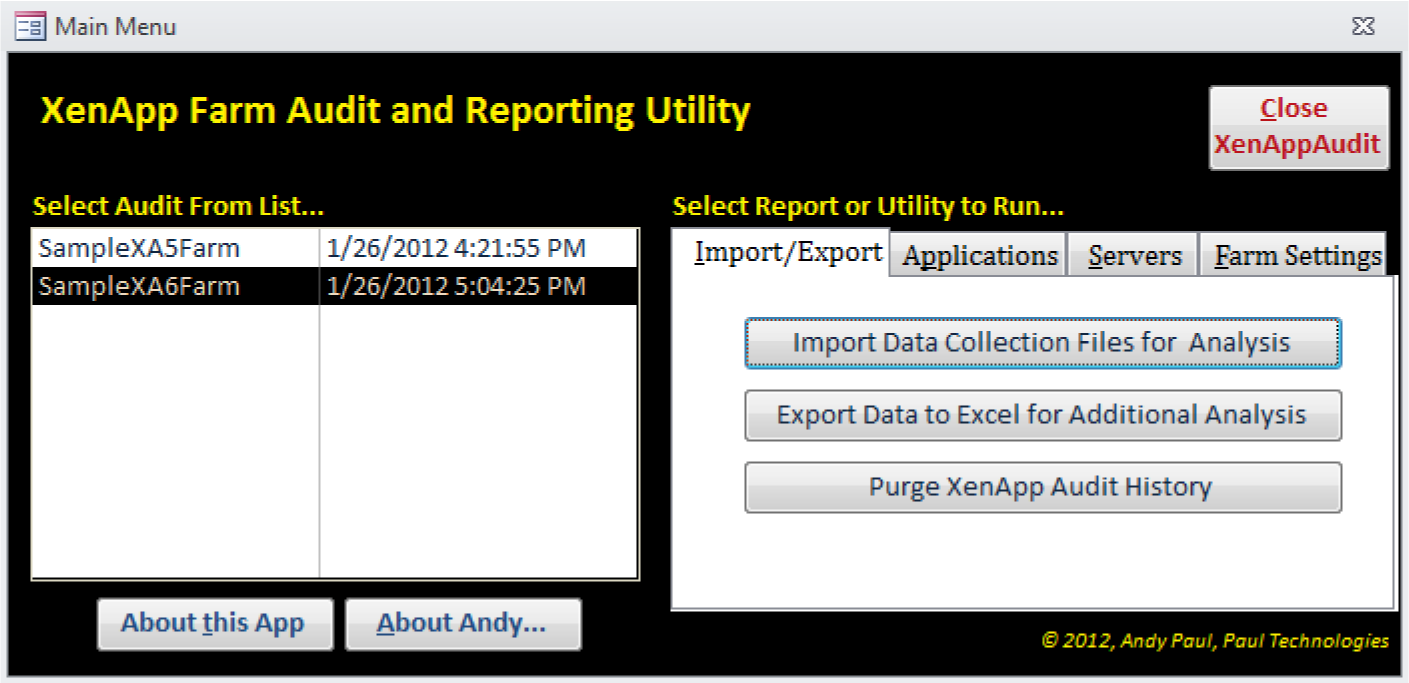
The XenAppAudit utility is designed with two parts:

- Data collection scripts used to populate the backend database
- Front-end Access application used to analyze the data

The front-end application requires Microsoft Access. It is written using Microsoft Access 2010, but is compatible with Microsoft Access 2003 or later. The linked tables are fixed to C:\Program Files\XenAppAudit\Results\XenAppAudit_be.mdb, but they can be updated using the Linked Table Manager in Access if necessary.

Starting the XenAppAudit Program

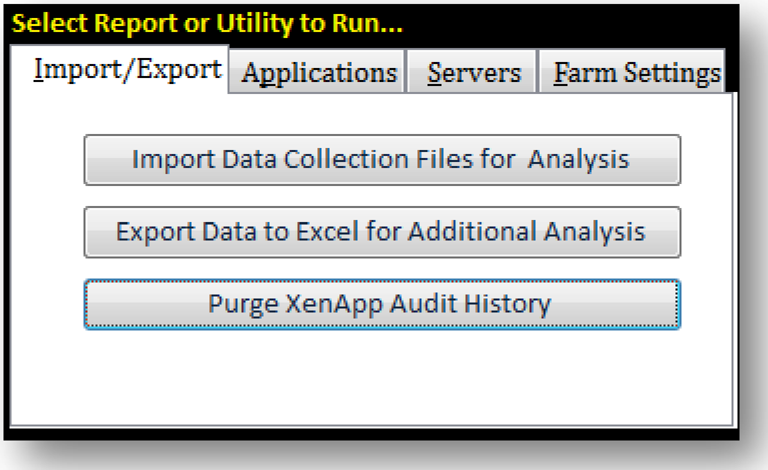
To start the XenAppAudit program, double-click on **XenAppAudit.mdb** located in C:\Program Files\XenAppAudit\. The database is configured with startup options to set the application title (XenApp Farm Audit and Reporting Utility), the startup form (Main Menu, shown below), limit the menus, and hide the navigation panes. This provides for a clean user interface; however, the special access keys are still allowed. You can press F11 view the Navigate Pane if necessary.


Main Menu
XenApp Farm Audit and Reporting Utility
Close XenAppAudit
Select Audit From List...
Select Report or Utility to Run...
Import/Export
Applications
Servers
Farm Settings
Import Data Collection Files for Analysis
Export Data to Excel for Additional Analysis
Purge XenApp Audit History
About this App
About Andy...
© 2012, Andy Paul, Paul Technologies

SampleXA5Farm	1/26/2012 4:21:55 PM
SampleXA6Farm	1/26/2012 5:04:25 PM

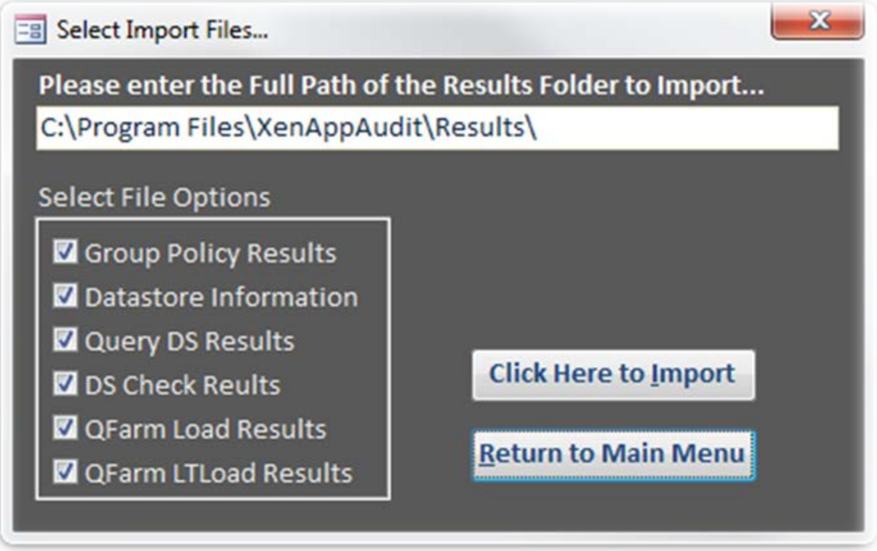
The SAMPLE folder includes a sample back-end database which contains a couple of sample audits, as shown in the image above. The back-end database in the RESULTS folder is blank; any new audits will appear in the list using the Farm Name and the date of the audit. The commands tab will be disabled until an Audit is selected from the list. Once an audit is selected all commands become available.

Import/Export Tab


Select Report or Utility to Run...
Import/Export
Applications
Servers
Farm Settings
Import Data Collection Files for Analysis
Export Data to Excel for Additional Analysis
Purge XenApp Audit History

Import Data Collection Files for Analysis

This command opens another window, allowing you to enter the path for the files and select the files to import. These files are generated as part of the LAUNCHER.VBS script and are stored in the RESULTS folder by default.



Please enter the Full Path of the Results Folder to Import...
C:\Program Files\XenAppAudit\Results\

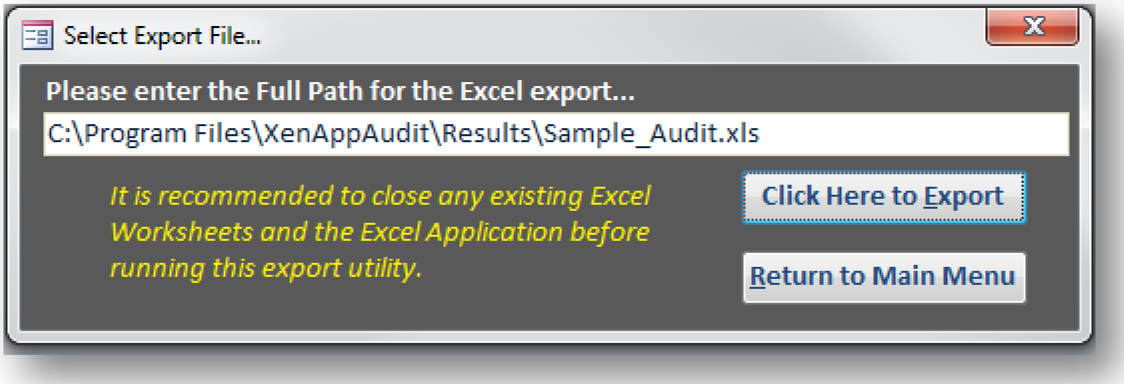
Select File Options
- ☑ Group Policy Results
- ☑ Datastore Information
- ☑ Query DS Results
- ☑ DS Check Reults
- ☑ QFarm Load Results
- ☑ QFarm LTLoad Results

In addition to importing the text files and storing the data for review later, the program also parses the QueryDS results and places the information into the database as part of the Server Health section of the Servers Full Details Report.

For more information on QueryDS, please see CTX106318.

Export Data to Excel for Additional Analysis

This command opens another window, allowing you to specify the Microsoft Excel file name for export. The export utility requires Microsoft Excel to be installed on the workstation running the XenAppAudit program. I recommend closing any open Excel files and applications prior to running the export utility.



Please enter the Full Path for the Excel export...

C:\Program Files\XenAppAudit\Results\Sample_Audit.xls

It is recommended to close any existing Excel Worksheets and the Excel Application before running this export utility.

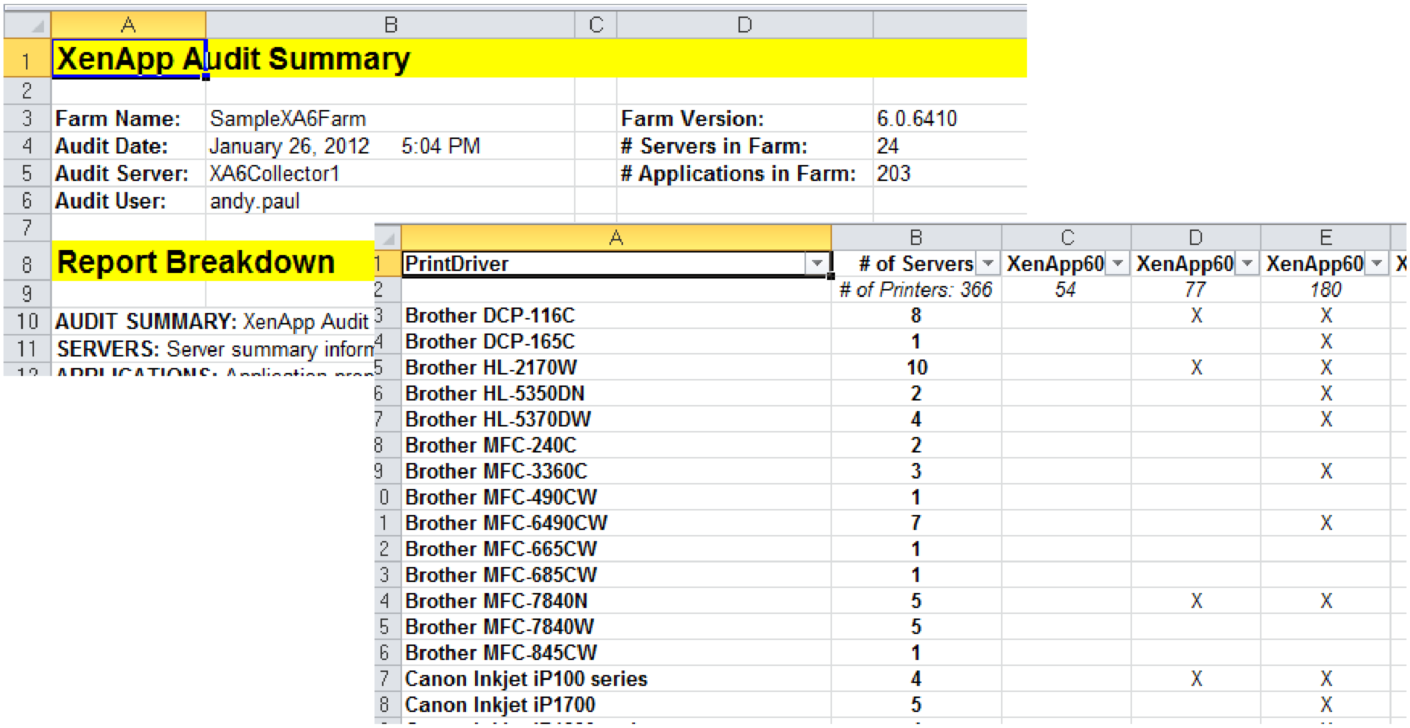
Clicking the Click Here to Export button will initialize the export routine which will perform the following tasks:

- Create the Excel File
- Export the data to Excel
- Format the Excel worksheets
- Insert additional calculations
- Apply AutoFilter to the Excel worksheets

The Excel Sheet will contain the following tabs:

- **AUDIT SUMMARY:** XenApp Audit summary information
- **SERVERS:** Server summary information for all servers, including real-time data
- **APPLICATIONS:** Application properties summary information
- **APPS-SERVERS:** Tabular analysis of which Applications are available on which Servers
- **PRINTERS:** Tabular analysis of which print drivers are installed on which Servers
- **HOTFIXES:** Tabular analysis of which XenApp HotFixes are applied on which Servers
- **WORKER GROUPS-APPS:** Tabular analysis of which Applications are assigned to which Worker Groups in the Farm (if applicable, only available for XenApp 6 or higher)
- **WORKER GROUPS-SERVERS:** Tabular analysis of which Servers are assigned to which Worker Groups in the Farm (if applicable, only available for XenApp 6 or higher)
- **APP PUB DETAILS:** Application details for command line, user, groups and servers

A couple of sample Excel exports are included in the SAMPLE folder.



XenApp Audit Summary

Farm Name:	SampleXA6Farm	**Farm Version:**	6.0.6410
Audit Date:	January 26, 2012 5:04 PM	**# Servers in Farm:**	24
Audit Server:	XA6Collector1	**# Applications in Farm:**	203
Audit User:	andy.paul		

Report Breakdown

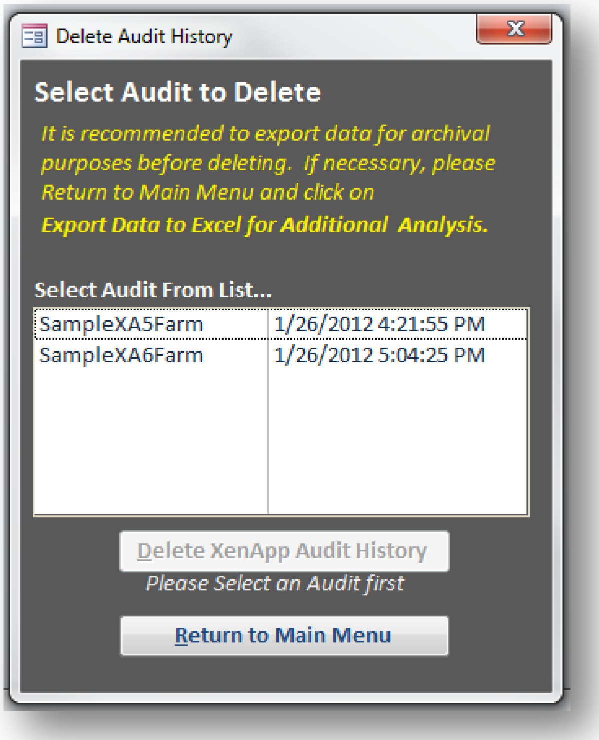
AUDIT SUMMARY: XenApp Audit

SERVERS: Server summary inforn

PrintDriver	# of Servers	XenApp60	XenApp60	XenApp60
	# of Printers: 366	*54*	*77*	*180*
Brother DCP-116C	8		X	X
Brother DCP-165C	1			X
Brother HL-2170W	10		X	X
Brother HL-5350DN	2			X
Brother HL-5370DW	4			X
Brother MFC-240C	2			
Brother MFC-3360C	3			X
Brother MFC-490CW	1			
Brother MFC-6490CW	7			X
Brother MFC-665CW	1			
Brother MFC-685CW	1			
Brother MFC-7840N	5		X	X
Brother MFC-7840W	5			
Brother MFC-845CW	1			
Canon Inkjet iP100 series	4		X	X
Canon Inkjet iP1700	5			X

Purge XenApp Audit History

This command opens another window, allowing you to delete database entries for an Audit. Once deleted, the data cannot be recovered. It is recommended to export the data to Excel prior to deletion.



Select Audit to Delete

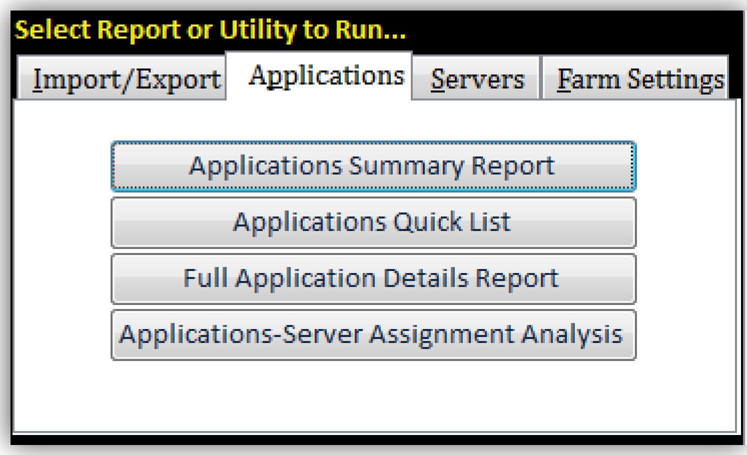
It is recommended to export data for archival purposes before deleting. If necessary, please Return to Main Menu and click on ***Export Data to Excel for Additional Analysis.***

Select Audit From List...

SampleXA5Farm	1/26/2012 4:21:55 PM
SampleXA6Farm	1/26/2012 5:04:25 PM

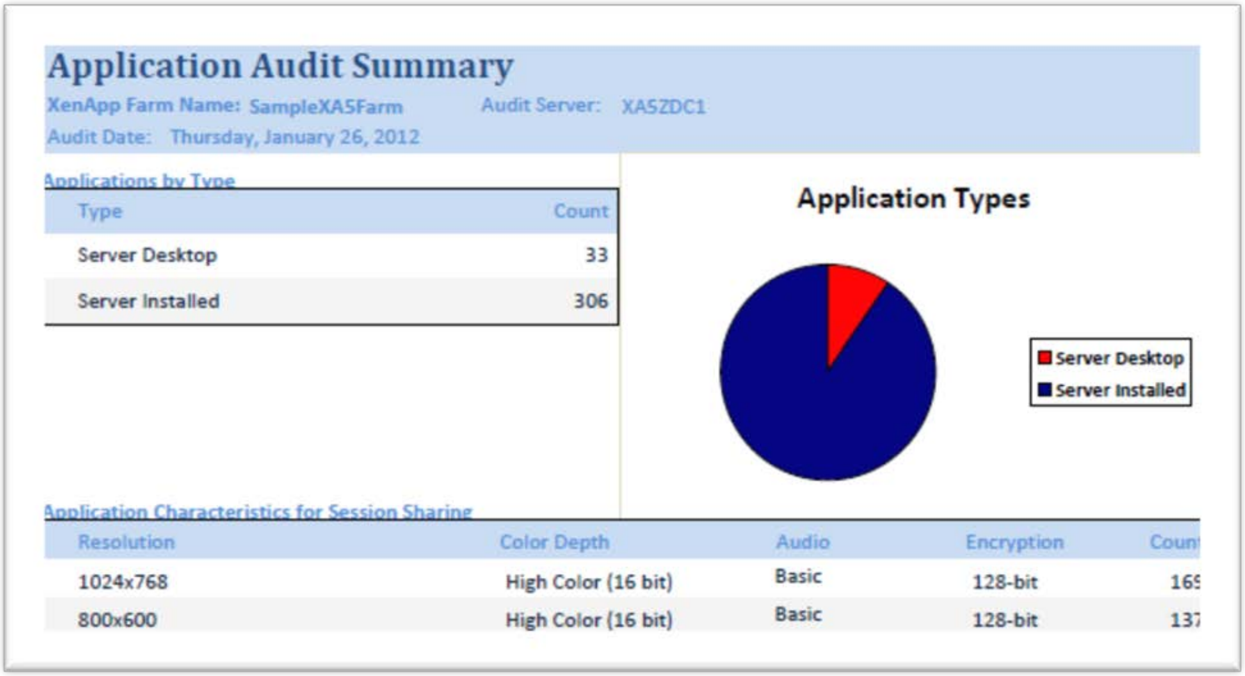
Please Select an Audit first

Applications Tab


Select Report or Utility to Run...
Import/Export
Applications
Servers
Farm Settings
Applications Summary Report
Applications Quick List
Full Application Details Report
Applications-Server Assignment Analysis

Applications Summary Report

This report provides a high level summary of the applications, including a breakdown of application type, session sharing properties, most frequently used commands, and top running processes.



Application Audit Summary

XenApp Farm Name: SampleXA5Farm Audit Server: XA5ZDC1
Audit Date: Thursday, January 26, 2012

Applications by Type

Type	Count
Server Desktop	33
Server Installed	306

Application Characteristics for Session Sharing

Resolution	Color Depth	Audio	Encryption	Coun
1024x768	High Color (16 bit)	Basic	128-bit	16
800x600	High Color (16 bit)	Basic	128-bit	13

Applications Quick List

This command opens a data sheet listing the applications and base application properties as well as server, user, and group summaries. This is the same data exported to Excel under the Applications tab.

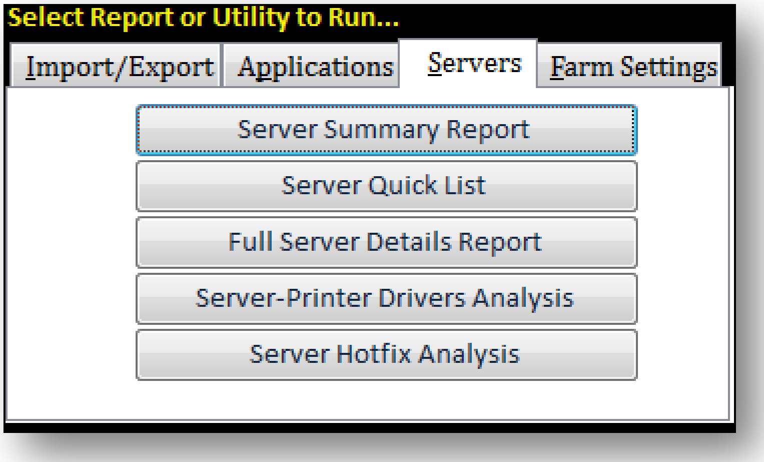
Full Application Details Report

This report provides all the details for each application. This includes a listing of all properties, the command line, and a listing of all users, groups and servers assigned to each application.

Applications-Server Assignment Analysis

This report is a cross-tab analysis showing which applications are published against which servers. This is the same data exported to Excel under the Apps-Servers tab.

Servers Tab



Server Summary Report

This report provides a high level summary of the servers in the Farm. This includes identifying Zone Data Collectors and election preferences, hotfix summary, load evaluator summary, server model summary, and a top loads summary. This report also contains graphs for the real-time CPU and RAM utilizations for all servers in the Farm. A sample report is available on the next page.

Server Quick List

This command opens a data sheet listing the servers, loads, sessions, CPU and RAM, and number of applications published. This is the same data exported to Excel under the Servers tab.

Full Server Details Report

This report provides all the details of each server. This includes IP information, hardware properties, real-time performance data, number of sessions, top users, server health, and all hotfixes currently applied.

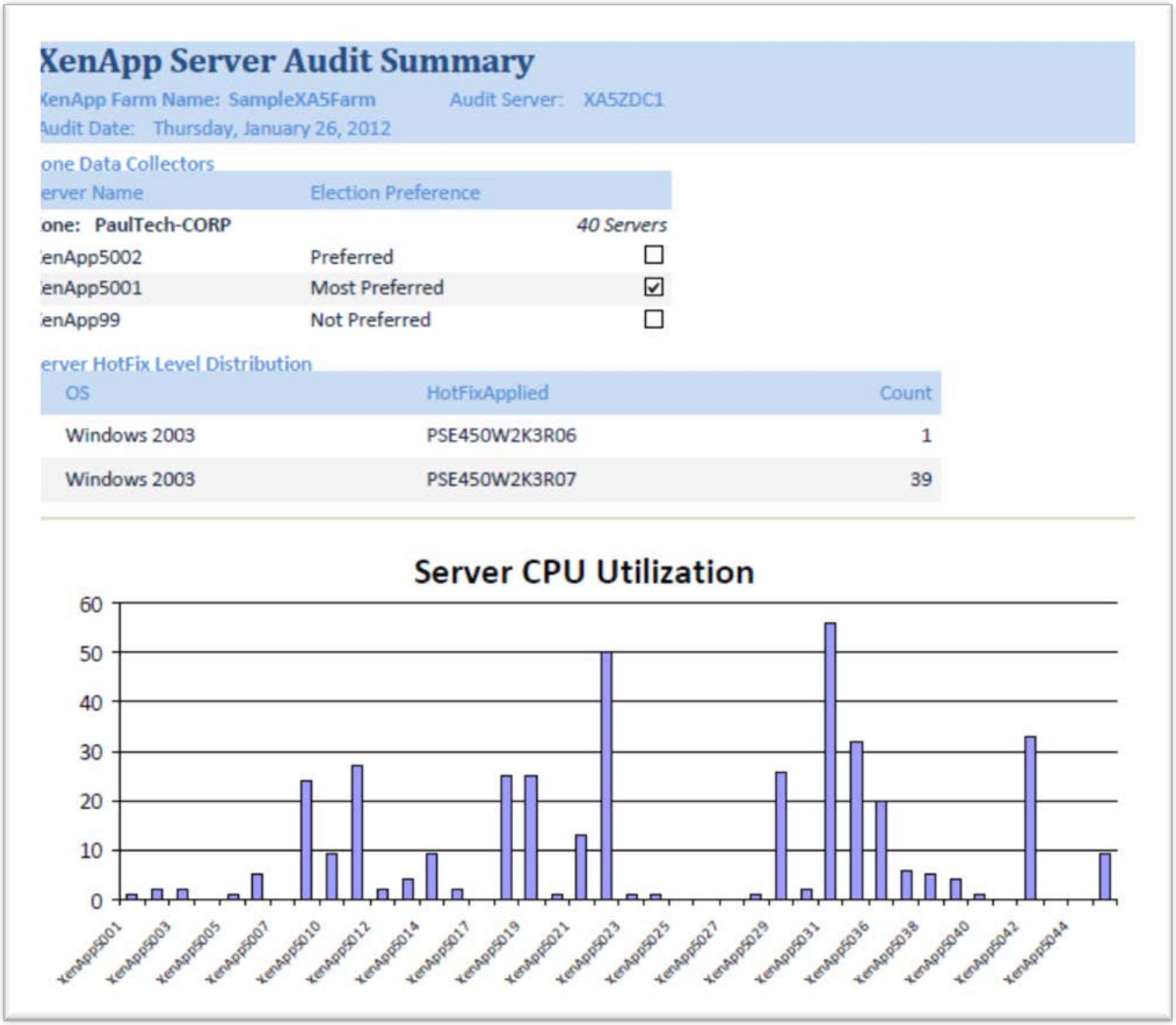
Server-Printer Drivers Analysis

This report is a cross-tab analysis showing which printer drivers are installed on which servers. This is the same data exported to Excel under the Printers tab.

Server Hotfix Analysis

This report is a cross-tab analysis showing which hotfixes are applied to which servers. This is the same data exported to Excel under the Hotfixes tab.

Sample Server Summary Report:



XenApp Server Audit Summary

enApp Farm Name: SampleXA5Farm Audit Server: XA5ZDC1

udit Date: Thursday, January 26, 2012

one Data Collectors

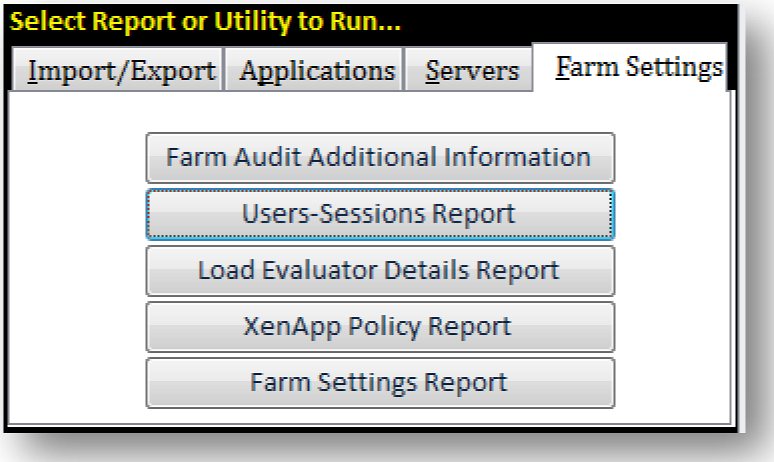
erver Name	Election Preference	
one: PaulTech-CORP		*40 Servers*
enApp5002	Preferred	☐
enApp5001	Most Preferred	☑
enApp99	Not Preferred	☐

erver HotFix Level Distribution

OS	HotFixApplied	Count
Windows 2003	PSE450W2K3R06	1
Windows 2003	PSE450W2K3R07	39

Server CPU Utilization

Farm Settings Tab

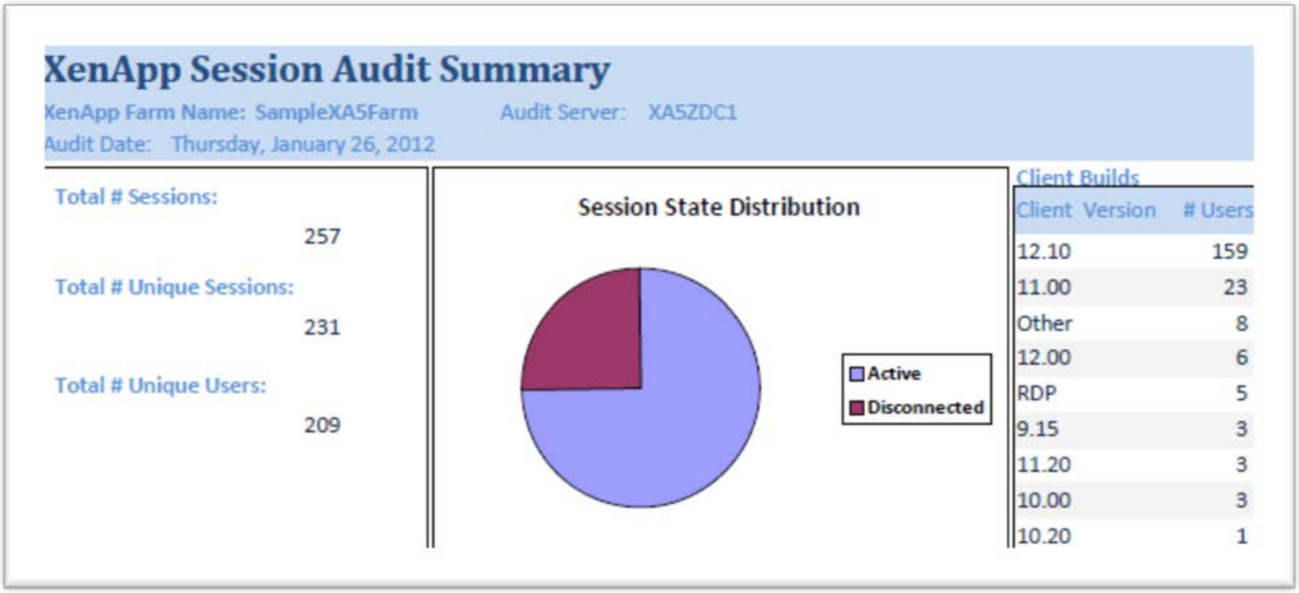


Farm Audit Additional Information

This report shows some of the imported data, such as the datastore connection information, DSCheck results, QFarm results and Group Policy results. A sample report is available on the next page.

Users-Sessions Report

This report shows a breakdown sessions and unique users, distribution of session state, list of clients in use, and the top sessions by resources consumed.



XenApp Session Audit Summary

XenApp Farm Name: SampleXA5Farm Audit Server: XA5ZDC1
Audit Date: Thursday, January 26, 2012

Total # Sessions: 257

Total # Unique Sessions: 231

Total # Unique Users: 209

Client Builds

Client Version	# Users
12.10	159
11.00	23
Other	8
12.00	6
RDP	5
9.15	3
11.20	3
10.00	3
10.20	1

Load Evaluator Details Report

Report is currently not available but will be in future releases.

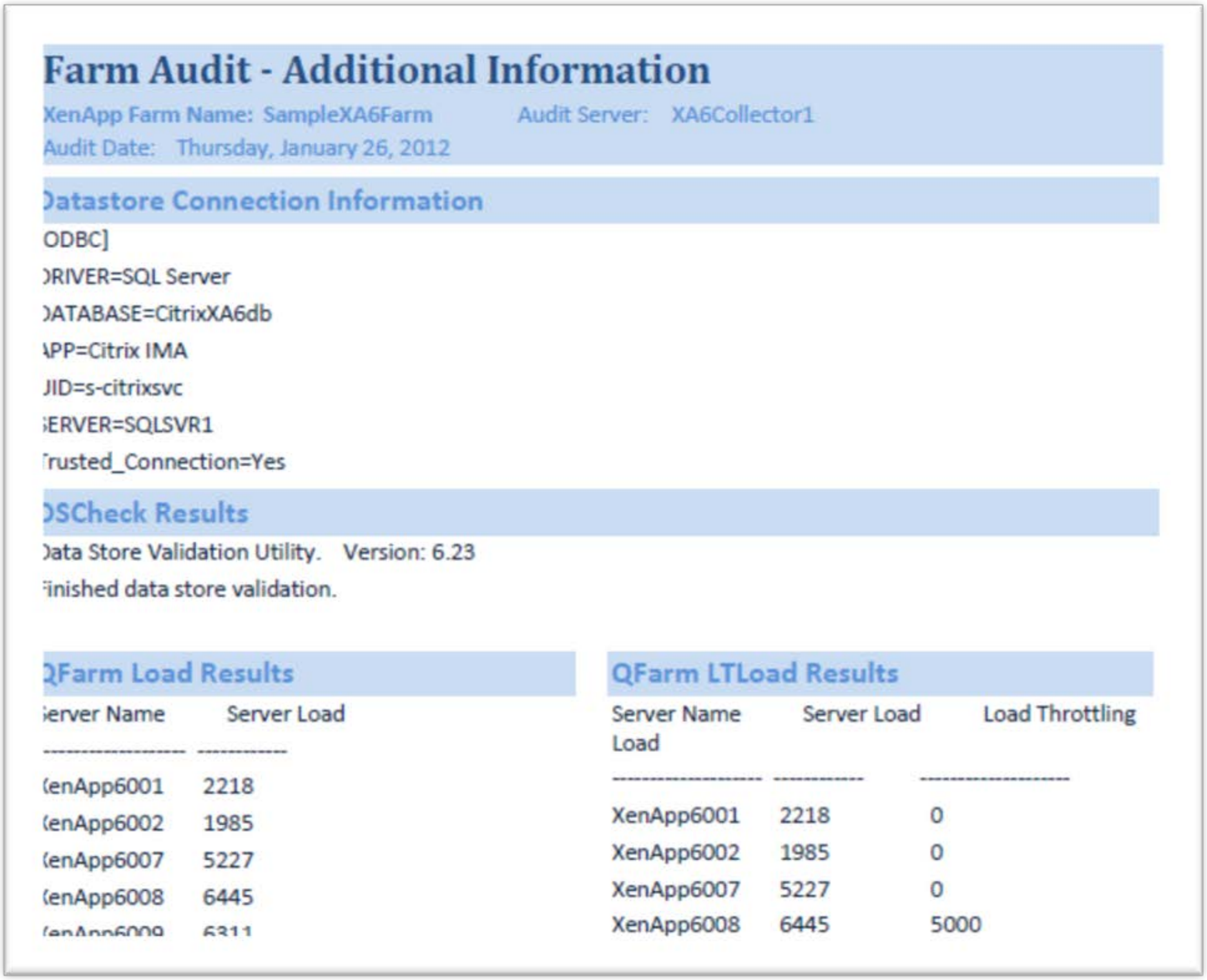
XenApp Policy Report

Report is currently not available but will be in future releases.

Farm Settings Report

Report is currently not available but will be in future releases.

Sample Farm Audit Additional Information Report:



Farm Audit - Additional Information

XenApp Farm Name: SampleXA6Farm   Audit Server: XA6Collector1
Audit Date: Thursday, January 26, 2012

atastore Connection Information

```
ODBC]
RIVER=SQL Server
ATABASE=CitrixXA6db
PP=Citrix IMA
ID=s-citrixsvc
ERVER=SQLSVR1
rusted_Connection=Yes
```

SCheck Results

```
ata Store Validation Utility.   Version: 6.23
inished data store validation.
```

Farm Load Results

```
erver Name        Server Load
----------------- -----------

enApp6001    2218
enApp6002    1985
enApp6007    5227
enApp6008    6445
enApp6009    6311
```

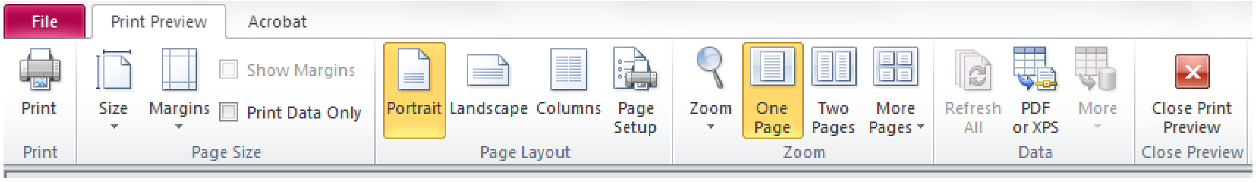
QFarm LTLoad Results

```
Server Name        Server Load        Load Throttling
Load
------------------ ------------       ------------------

XenApp6001    2218         0
XenApp6002    1985         0
XenApp6007    5227         0
XenApp6008    6445         5000
```

Running Reports

When running reports in the XenAppAudit program, you should see the Print Preview ribbon bar. This ribbon bar will allow you to control the print options as well as preview the report on screen and zoom in for greater detail.

The ribbon bar also includes an option to print to PDF or XPS for distribution. A couple of sample reports in PDF format are included in the SAMPLE folder.



Credits

Although this application is my own home-grown XenApp Audit utility, built from scratch, I cannot claim full credit. Many individuals have added their knowledge and ideas to my own over time; so if I leave out any one individual, I apologize, it is not intentional.

Specifically, I would like to thank the follow individuals for their direct or indirect contributions to this project:

- Ron Kraft
- Moin Kahn
- Shawn Ragsdale
- Duane Bradley
- Brandon Shell
- Jason Conger
- Arthur Penn

And I am sure many others, unnamed but their help is always appreciated!

If you do distribute or modify, please be sure to give credit where credit is due. Thanks!

About Andy

Andy Paul is an accomplished virtualization architect, instructor and speaker. He has designed and delivered virtualization projects for Fortune 500 companies, public and private healthcare organizations and higher-education institutions. He has also served as a lead technical trainer, adjunct professor and guest speaker for multiple organizations.

He is a Microsoft Certified IT Professional, Certified Citrix Integration Architect for Virtualization and a Certified Citrix Instructor. He also holds the advanced degrees of Master of Science in Managing Information Technology and a Master of Business Administration.

His roles have included database programming and administration, network engineering, datacenter management, systems engineering, system virtualization, disaster recovery and application delivery. He is not married to any one technology, but believes in using the best tools available for a holistic solution.

Andy's specialties include: Virtual Desktop Infrastructure, Application Virtualization, Terminal Services, Server Based Computing, Citrix Technologies, System Architecture, Network/Systems Design and Engineering, Policy/Procedure Development, Project Management, Server Virtualization/Consolidation

Andy can be contacted via LinkedIn, Twitter, or email at andy.paul@paultechnologies.com.

Appendix A – Frequently Asked Questions

Q: What is the licensing cost for XenAppAudit?

A: There is no licensing cost. XenAppAudit is free for anyone to use.

Q: Where can I buy XenAppAudit?

A: XenAppAudit is not for sale. It is free for anyone to use.

Q: Is this program supported by Citrix?

A: No. This program is of my own creation. It is not supported in any way by Citrix, GlassHouse or Microsoft. This is my own pet project; however, I did leverage Citrix's knowledge base and SDKs for XenApp.

Q: What are the requirements to run XenAppAudit?

A: You will need XenApp Farm Administrator access as well as Server-level Administrator Access. When running on Windows 2008 or higher with UAC enabled, you will need to run as an administrator. When running XenApp 6 or XenApp 6.5, you will need the associated SDK installed on the server that is running the data collection scripts.

Q: What versions of XenApp are supported?

A: This utility has been tested against Citrix Presentation Server 4.0 and all current Citrix XenApp versions (4.5, 5.0, 6.0, and 6.5). There are some data elements that may not be available for all versions.

Q: How do I use XenAppAudit?

A: After extracting the ZIP file (preferably under C:\Program Files\XenAppAudit), simply run the START.BAT file. It will inspect your system and call the necessary data collection scripts. All executions are recorded in a transcript file located under the RESULTS folder. After the data collection is complete, you can then launch the front end application by opening XenAppAudit.mdb.

Q: How long does it take XenAppAudit to run?

A: That greatly depends on how large your Farm is, how many servers are the Farm, and your network topology. I generally see between 5 and 20 minutes, but some audits have taken as long as 45 minutes. I use a lot of WMI calls in the code to gather real-time data. These calls tend to be the longest running item. If you need to run the audits quickly, you can comment out or eliminate the WMI calls.

Q: Do I need to run the XenAppAudit utility on all servers?

A: No, you only need to run it on one server in the Farm. I typically run this on a Data Collector, but that is my personal preference.

Q: Can I run it on a non-XenApp server?

A: No, I wrote the scripts assuming the execution server is a XenApp server. However, you may be able to modify the scripts to suite your need.

Q: Can I run XenAppAudit multiple times?

A: Yes. It is designed to be run multiple times to allow subsequent audits and data comparison.

Q: How can I run XenAppAudit against multiple environments?

A: There are a couple of options. You can install the utility on one server in each environment and keep them stand alone. Also, you can copy the back end database (XenAppAudit_be.mdb) between servers to have one back-end. Another option is to modify the data collection scripts to use a network path to reference the RESULTS folder, as opposed to a local path.

Q: I have a problem with the XenAppAudit utility, where can I get support?

A: This utility is delivered as-is, with no warranties. Although I cannot provide direct support, I am happy to answer any email inquiries. Contact information is available under the “About Andy...” form. You may also post questions on the XenAppAudit page. Either for of inquiry will be answered as soon as possible.

Q: What language is used to run XenAppAudit?

A: XenAppAudit uses a variety of scripts and programming objects, including Batch Files, VBScript, PowerShell, Visual Basic for Applications, and Microsoft Access.

Q: Why did you use Microsoft Access?

A: My first professional job was as a Microsoft Access database developer, so I tend to fall back on Access when possible. I find Access to be easy to use for data analysis and customization. It provides and easy to use GUI, user forms, reports, and queries.

Q: Do I need Microsoft Access to use XenAppAudit?

A: Yes and no. You DO NOT need Access to run the XenAppAudit collection utilities. The Jet Engine database ODBC connections are built into Windows, so there are no special drivers needed, so the script can open the back-end database (XenAppAudit_be.mdb) and insert all the auditing data. However, you will need Access 2003 or higher to run the front-end utility (XenAppAudit.mdb), which is written in Access 2010.

Q: Can I change the code behind XenAppAudit?

A: Absolutely! I purposely chose to use Access and various scripts as opposed to a compiled EXE for this reason. Feel free modify any of the scripts or Access objects as necessary to fit your needs. Of course, changing the scripts may make it difficult to gain assistance if need be. If you find a flaw or have improvements after looking at the scripts, you can email me the recommended changes as well. I figure share and share alike!

Q: Can I redistribute XenAppAudit?

A: Sure. The easiest way is to link to this site. However, feel free to redistribute as necessary. I only ask that you give credit where credit is due.

Q: Can I re-brand XenAppAudit?

A: Please don’t. Feel free to modify and distribute as necessary, but a lot of time and energy went into this project. I only ask that you give credit where credit is due, and re-branding this effort would take away from that philosophy.

Q: I want to modify something on Access, but I don’t see the objects.

A: The XenAppAudit.mdb contains all the logic, but the objects are hidden. If you need to see them, press F11. Alternately, you can hold down the SHIFT key when opening the file and prevent the application start-up routine. The actual tables (and data) are linked to the XenAppAudit_be.mdb. Any table changes should be made there, then the linked tabled updated in the front-end application.

Appendix B – Programming Code

START.BAT

```
@ECHO OFF
cls
ECHO --- BEGINNING XEN APP AUDIT ---
ECHO .................................
cscript /nologo launcher.vbs

cd Results
set resultspath=%cd%
cd..
cd Scripts
queryds.exe /table:LMS_ServerLoadTable > "%resultspath%\QueryDSResults.txt"
cd..

ECHO .................................
ECHO --- XEN APP AUDIT COMPLETE! ---
```

LAUNCHER.VBS

```
'           File:              Launcher.vbs
'           Description:       Audit XenApp Farm Launching Utility Script
'           Requirements:      Run on XenApp Server
'           Created by:        Andy Paul
'                                    www.paultechnologies.com
'
'           Assumes C:\Program Files\XenAppTemp location but can be modified in code as
needed.
'           For x64 systems, calls 32-bit processes for MSAccess Database accessability
'
'
on error resume next
xenappver = "ERROR"
Dim  progpath : progpath="C:\Program Files\XenAppAudit\"
'Get Program Path from actual execution location
curpath = Replace(WScript.ScriptFullName, WScript.ScriptName, "")
Set wshShell = CreateObject( "WScript.Shell" )
Set objFSO = CreateObject("Scripting.FileSystemObject")
progpath = curpath

' Display the result
Dim resultspath :  resultspath=progpath & "Results"
Dim scriptspath : scriptspath=progpath & "Scripts"
varCloseOut = 0
Set objFSO = CreateObject("Scripting.FileSystemObject")
strFilePath = resultspath & "\Transcript_Launcher.txt"
Set objResultsFile = objFSO.OpenTextFile(strFilePath, 8, True, 0)

objResultsFile.WriteLine "USER NAME       :  " & wshShell.ExpandEnvironmentStrings(
"%USERNAME%" )
objResultsFile.WriteLine "COMPUTER NAME   :  " & wshShell.ExpandEnvironmentStrings(
"%COMPUTERNAME%" )
```

```
objResultsFile.WriteLine "PROGRAM PATH    :  " & progpath
objResultsFile.WriteLine "DATE STAMP      :  " & Now()
objResultsFile.WriteLine "-----------------------------------------------"
dim filesys, xenappver
Set filesys = CreateObject("Scripting.FileSystemObject")

var64 =  wshShell.ExpandEnvironmentStrings( "%PROCESSOR_ARCHITECTURE%" )

if var64 = "AMD64" then
      objResultsFile.WriteLine "       ...  Running 64-bit Operating System"
      xenappver = filesys.GetFileVersion("c:\Program Files
(x86)\Citrix\System32\wfshell.exe")
else
      objResultsFile.WriteLine "       ...  Running 32-bit Operating System"
      xenappver = filesys.GetFileVersion("c:\Program Files\Citrix\System32\wfshell.exe")
end if

if xenappver = "ERROR" then
      msgbox "ERROR: Unable to Determine XenApp Version. XenAppAudit is designed to be
run on a XenAppServer.",vbCritical,"Unknown XenApp Version"
            objResultsFile.WriteLine "       .....Exiting due to error with XenApp
Install..."
            varCloseOut = 1
end if
objResultsFile.WriteLine "-----------------------------------------------"
if varcloseout = 0 then
      if var64 = "AMD64" then
            if left(xenappver,1) = 6 then
                  objResultsFile.WriteLine "       ...  XenApp " & left(xenappver,3)
                  if filesys.fileExists("C:\Program Files\Citrix\XenApp Server
SDK\Citrix.XenApp.Sdk.ps1") then
                        objResultsFile.WriteLine "       ...  Launching PowerShell XenApp
Collection"
                        objResultsFile.WriteLine "       ...  Transcript available under
..\Results\Transcript.txt"
                        objResultsFile.WriteLine "       ...  PowerShell Script begin: "
& now()
                        cmdstr =
"%SystemRoot%\syswow64\WindowsPowerShell\v1.0\powershell.exe -executionpolicy
Unrestricted -file " & chr(34) & scriptspath & "\XenAppAudit.PS1" & chr(34)
                        objResultsFile.WriteLine "       ..... Executing: " & cmdstr
                        WshShell.Run cmdstr, 1, true
                        objResultsFile.WriteLine "       ...  PowerShell Script complete:
" & now()
                  elseif filesys.fileExists("C:\Program Files\Citrix\XenApp 6.5 Server
SDK\Citrix.XenApp.Sdk.ps1") then
                        objResultsFile.WriteLine "       ...  XenApp 6.5"
                        objResultsFile.WriteLine "       ...  Launching PowerShell XenApp
Collection"
                        objResultsFile.WriteLine "       ...  Transcript available under
..\Results\Transcript.txt"
                        objResultsFile.WriteLine "       ...  PowerShell Script begin: "
& now()
                        cmdstr =
"%SystemRoot%\syswow64\WindowsPowerShell\v1.0\powershell.exe -executionpolicy
Unrestricted -file " & chr(34) & scriptspath & "\XenAppAudit.PS1" & chr(34)
```

```
                        objResultsFile.WriteLine "       ..... Executing: " & cmdstr
                        WshShell.Run cmdstr, 1, true
                        objResultsFile.WriteLine "       ...  PowerShell Script complete: 
" & now()
                  else
                        msgbox "ERROR: Unable to find XenApp SDK. XenApp SDK is required 
for the XenApp Audit to run.",vbCritical,"XenApp 6 SDK Required."
                        objResultsFile.WriteLine "       .....Exiting due to error with 
SDK..."
                        varCloseOut = 1
                  end if
            else
                  objResultsFile.WriteLine "       ...  XenApp " & left(xenappver,3)
                        objResultsFile.WriteLine "       ...  Launching VBScript/MFCOM 
XenApp Collection"
                        objResultsFile.WriteLine "       ...  Transcript available under 
..\Results\Transcript.txt"
                        objResultsFile.WriteLine "       ...  VBScript begin: " & now()
                        cmdstr = "C:\Windows\SysWOW64\cscript.exe /NOLOGO " & chr(34) & 
scriptspath & "\XenAppAudit.wsf" & chr(34)
                        objResultsFile.WriteLine "       ..... Executing: " & cmdstr
                        WshShell.Run cmdstr, 1, true
                        objResultsFile.WriteLine "       ...  VBScript complete: " & 
now()
            end if
      else
            if left(xenappver,1) = 6 then
                  objResultsFile.WriteLine "       ...  ERROR: XenApp " & 
left(xenappver,3)
            else
                  objResultsFile.WriteLine "       ...  XenApp " & left(xenappver,3)
                        objResultsFile.WriteLine "       ...  Launching VBScript/MFCOM 
XenApp Collection"
                        objResultsFile.WriteLine "       ...  Transcript available under 
..\Results\Transcript.txt"
                        objResultsFile.WriteLine "       ...  VBScript begin: " & now()
                        cmdstr = "C:\Windows\System32\cscript.exe /NOLOGO " & chr(34) & 
scriptspath & "\XenAppAudit.wsf" & chr(34)
                        objResultsFile.WriteLine "       ..... Executing: " & cmdstr
                        WshShell.Run cmdstr, 1, true
                        objResultsFile.WriteLine "       ...  VBScript complete: " & 
now()
            end if
      end if
end if

if varCloseOut = 0 then
      strComputer = "."
      Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
      Set colOperatingSystems = objWMIService.ExecQuery _
        ("Select * from Win32_OperatingSystem")
      For Each objOperatingSystem in colOperatingSystems
            varOSName = objOperatingSystem.Caption
            varOSNumber =  objOperatingSystem.Version
      Next
      call TextDataCollection

end if

'Close Out
CLOSEOUT:
objResultsFile.WriteLine "-----------------------------------------------"
objResultsFile.WriteLine "AUDIT COMPLETE  :  " & Now()
objResultsFile.Close
Set wshSystemEnv = Nothing
Set wshShell     = Nothing
wscript.quit
' *******************************************
sub TextDataCollection
'oShell.run "cmd /K CD C:\ & Dir"
'scriptspath
'resultspath
objResultsFile.WriteLine "-----------------------------------------------"
objResultsFile.WriteLine "ADDITIONAL DATA COLLECTION SCRIPTS"
objResultsFile.WriteLine "       ...  QFARM"
      cmdstr = "cmd.exe /C QFARM > " & chr(34) & resultspath & "\XenAppAudit_Data.txt" &
chr(34)
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  QFARM /ZONE"
      cmdstr = "cmd.exe /C QFARM /ZONE >> " & chr(34) & resultspath &
"\XenAppAudit_Data.txt" & chr(34)
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  QFARM /OFFLINE"
      cmdstr = "cmd.exe /C QFARM /OFFLINE >> " & chr(34) & resultspath &
"\XenAppAudit_Data.txt" & chr(34)
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  QFARM /LOAD"
      cmdstr = "cmd.exe /C QFARM /LOAD > " & chr(34) & resultspath & "\QFARM_LOAD.txt" &
chr(34)
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  QFARM /LTLOAD"
      if left(xenappver,3) = "4.0" then
            objResultsFile.WriteLine "       ..... Not applicable for Presentation Server
4"
      else
            cmdstr = "cmd.exe /C QFARM /LTLOAD > " & chr(34) & resultspath &
"\QFARM_LTLOAD.txt" & chr(34)
            objResultsFile.WriteLine "       ..... Executing: " & cmdstr
            WshShell.Run cmdstr, 1, true
      end if
objResultsFile.WriteLine "       ...  DSCHECK"
      cmdstr = "cmd.exe /C DSCHECK > " & chr(34) & resultspath & "\DSCheckResults.txt" &
chr(34)
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  GPRESULT"
      if left(varOSNumber,1) >= 6 then 'Win2008+
            cmdstr = "cmd.exe /C GPRESULT /R > " & chr(34) & resultspath &
"\GPResult.txt" & chr(34)
      else
            cmdstr = "cmd.exe /C GPRESULT > " & chr(34) & resultspath & "\GPResult.txt" &
chr(34)
      end if
      objResultsFile.WriteLine "       ..... Executing: " & cmdstr
      WshShell.Run cmdstr, 1, true
objResultsFile.WriteLine "       ...  QUERYDS"
      objResultsFile.WriteLine "       ..... Run upon completion of all other scripts"
objResultsFile.WriteLine "       ...  DATASTORE"
      targetfile = resultspath & "\datastore.txt"
      filename = "C:\Program Files (x86)\Citrix\Independent Management
Architecture\mf20.dsn"
            if filesys.fileExists(filename) then filesys.copyfile filename, targetfile
      filename = "C:\Program Files\Citrix\Independent Management Architecture\mf20.dsn"
            if filesys.fileExists(filename) then filesys.copyfile filename, targetfile
end sub
```

XENAPPAUDIT.WSF

```
<package>
<job id="Farm">

      <comment>
            File:             XenAppAudit.wsf
            Description:      Audit XenApp Farm and write Access Database for Analysis
            Requirements:     XenApp 4.0, 4.5, or 5.0
            Created by:       Andy Paul
                                    www.paultechnologies.com
      </comment>

<runtime>
      <description>
            A comprehensive script recording farm, application, server, etc. properties.
            Assumes C:\Program Files\XenAppTemp location but can be modified in code as
needed.
            For x64 systems, run from C:\Windows\SysWOW64\cmd.exe
      </description>

      <example>
            cscript /nologo XenAppAudit.wsf
      </example>
</runtime>

<reference object="MetaFrameCOM.MetaFrameFarm"/>
<reference object="MetaFrameCOM.MetaFramePolicy"/>

<script language="VBScript">

' Common Objects and Variables
On Error Resume Next
Dim RunDate : RunDate = Now()
WScript.Echo "Starting collection at " & RunDate
```

```
xenappver = "0.0"
Dim  progpath : progpath="C:\Program Files\XenAppAudit\"
Set wshShell = CreateObject( "WScript.Shell" )
'Get Program Path from actual execution location
curpath = Replace(WScript.ScriptFullName, WScript.ScriptName, "")
progpath = Replace(curpath, "Scripts\", "")
wscript.echo "Program Path :==  " & progpath
Dim resultspath :  resultspath=progpath & "Results"
Dim scriptspath : scriptspath=progpath & "Scripts"
Set objFSO = CreateObject("Scripting.FileSystemObject")
strFilePath = resultspath & "\Transcript_VBS.txt"
Set objResultsFile = objFSO.OpenTextFile(strFilePath, 8, True, 0)
objResultsFile.WriteLine "Starting collection at " & RunDate

      Dim ErrorChecking : ErrorChecking = 1
      Dim objShell             : Set objShell = CreateObject("WScript.Shell")
      Dim objNetwork           : Set objNetwork = CreateObject("WScript.Network")
      dim objConn              : Set objConn = CreateObject("ADODB.Connection")
      Dim conn_string    : conn_string = "provider=microsoft.jet.oledb.4.0;data source=" &
resultspath & "\XenAppAudit_be.mdb"
                                    objConn.open conn_string
      dim objRS                : Set objRS = CreateObject("ADODB.Recordset")
                                    objRS.CursorLocation = 3
      dim objRS2               : Set objRS2 = CreateObject("ADODB.Recordset")
                                    objRS.CursorLocation = 3
      dim ServerInfoRS   : Set ServerInfoRS = CreateObject("ADODB.Recordset")
                                    objRS.CursorLocation = 3
      strComputer = "."
      Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")

      Set wshShell = CreateObject( "WScript.Shell" )
      strUserName =  wshShell.ExpandEnvironmentStrings( "%USERNAME%" )
      strServerName = wshShell.ExpandEnvironmentStrings( "%COMPUTERNAME%" )

      var64 =  wshShell.ExpandEnvironmentStrings( "%PROCESSOR_ARCHITECTURE%" )
      if var64 = "AMD64" then
            xenappver = objFSO.GetFileVersion("c:\Program Files
(x86)\Citrix\System32\wfshell.exe")
      else
            xenappver = objFSO.GetFileVersion("c:\Program
Files\Citrix\System32\wfshell.exe")
      end if

      '************** Header Info
      objRS.Open "SELECT top 1 * FROM tblAudit" , objConn, 3, 3
            objRS.AddNew
                  objRS("AuditDate") = RunDate
                  objRS("AuditServer") = strServerName
                  objRS("AuditUser") = strUserName
            objRS.Update
      objRS.Close

'*********** FARM SETTINGS
WScript.Echo "Collecting Farm Data..."
objResultsFile.WriteLine "Collecting Farm Data..."
```

```
      Dim objFarm : Set objFarm = WScript.CreateObject("MetaFrameCOM.MetaFrameFarm")
      objFarm.Initialize(MetaFrameWinFarmObject)
      Dim objWinFarm : Set objWinFarm = objFarm.WinFarmObject
      Dim objSpeedBrowse : Set objSpeedBrowse =
CreateObject("MetaframeCOM.MetaFrameSpeedBrowse")

      If objWinFarm.IsCitrixAdministrator = 0 then
            WScript.Echo "You must be a Citrix admin to run this script"
            WScript.Quit 0
      End If
      wscript.echo "   " & objFarm.FarmName
      objResultsFile.WriteLine "   " & objFarm.FarmName

      objRS.Open "SELECT top 1 * FROM tblAudit Where AuditDate = #" & rundate & "#" ,
objConn, 3, 3
            objRS.movefirst
                  AuditID = objRS("AuditID")
                  objRS("AuditFarmName") = objFarm.FarmName
            objRS.Update
      objRS.Close
      wscript.echo "      - Collecting Farm IMA Properties..."
      objRS.Open "SELECT top 1 * FROM tblFarms" , objConn, 3, 3
            objRS.AddNew
                  objRS("FarmName") = objFarm.FarmName
                  objRS("FarmVersion") = xenappver
                  objRS("MaxConnectionsPerUser") = objWinFarm.MaxConnectionsPerUser
                  objRS("EnforceConnectionsLimitOnAdmins") =
objWinFarm.EnforceConnectionsLimitOnAdmins
                  objRS("LogOverLimitDenials") = objWinFarm.LogOverLimitDenials
                  objRS("EnableICAKeepAlive") = objWinFarm.EnableICAKeepAlive
                  objRS("ICAKeepAliveTimeout") = objWinFarm.ICAKeepAliveTimeout
                  objRS("NoRedundantGraphics") = objWinFarm.NoRedundantGraphics
                  objRS("AlternateCachingMethod") = objWinFarm.AlternateCachingMethod
                  objRS("ICAVideoBufferSize") = objWinFarm.ICAVideoBufferSize
                  objRS("DegradationBias") = objWinFarm.DegradationBias
                  objRS("NotifyDegradation") = objWinFarm.NotifyDegradation
                  objRS("EnableACR") = objWinFarm.EnableACR
                  objRS("LogACRAttempts") = objWinFarm.LogACRAttempts
                  objRS("LegacyMFServerCompatibleMode") =
objWinFarm.LegacyMFServerCompatibleMode
                  objRS("DCRespondToClientBroadcast") =
objWinFarm.DCRespondToClientBroadcast
                  objRS("RASRespondToClientBroadcast") =
objWinFarm.RASRespondToClientBroadcast
                  objRS("UseClientLocalTime") = objWinFarm.UseClientLocalTime
                  objRS("DisableClientLocalTimeEstimation") =
objWinFarm.DisableClientLocalTimeEstimation
                  objRS("EnableDNSAddressResolution") =
objWinFarm.EnableDNSAddressResolution
                  'objRS("NDSPreferredTree") = objWinFarm.NDSPreferredTree
                  objRS("EnableContentRedirection") = objWinFarm.EnableContentRedirection
                  objRS("EnableRemoteConsoleConnections") =
objWinFarm.EnableRemoteConsoleConnections
                  objRS("EnableSNMPAgent") = objWinFarm.EnableSNMPAgent
                  objRS("SNMPLogonTrap") = objWinFarm.SNMPLogonTrap
                  objRS("SNMPLogoffTrap") = objWinFarm.SNMPLogoffTrap
                  objRS("SNMPDisconnectTrap") = objWinFarm.SNMPDisconnectTrap
                  objRS("SNMPThresholdExceededTrap") =
objWinFarm.SNMPThresholdExceededTrap
                  objRS("SNMPThresholdValue") = objWinFarm.SNMPThresholdValue
                  objRS("SpeedBrowseEnable") = objSpeedBrowse.Enable
                  objRS("AuditID") = AuditID
                  objRS("SessionReliability") = objWinFarm.SREnabled
                  objRS("SessionReliabilityPort") = objWinFarm.SRPort
                  objRS("SessionReliabilityKeepAlive") = objWinFarm.SRTimeOut
                  objRS("LicenseServer") = objWinFarm.LicenseServerName
                  objRS("LicensePort") = objWinFarm.LicenseServerPort
            objRS.Update
      objRS.Close

      objRS2.Open "SELECT top 1 * FROM tblFarms Where AuditID = " & AuditID & " ORDER BY
FarmID DESC" , objConn, 3, 3
            objRS2.movefirst
            FarmID = objRS2("FarmID")
      objRS2.Close

' ************* SERVERS
WScript.Echo "Collecting Server Data..."
objResultsFile.WriteLine "Collecting Server Data..."
      Dim objWinServer
      objRS.Open "SELECT top 1 * FROM tblServers" , objConn, 3, 3

      For Each strServer In objFarm.Servers
      Set objWinServer = strServer.WinServerObject

      'WMI Server Info
      strComputer = strServer.ServerName
      wscript.echo "   " & strServer.ServerName
      objResultsFile.WriteLine "   " & strServer.ServerName
      wscript.echo "       - Collecting Real-time WMI Data..."
      Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" &
strComputer & "\root\cimv2")

      wscript.echo "                 Getting Win32_ComputerSystem WMI data"
      Set colSettings = objWMIService.ExecQuery ("Select * from Win32_ComputerSystem")
            For Each objComputer in colSettings
                  strMemory = int(objComputer.TotalPhysicalMemory/1000000/1024)
                  strProcs = objComputer.NumberOfProcessors
                  strMfg = objComputer.Manufacturer
                  strModel = objComputer.Model
                  CPUPercent = 0
                  RAMPercent = 0
            Next
      wscript.echo "                 Getting Win32_Processor WMI data"
      Set colItems = objWMIService.ExecQuery("Select * from Win32_Processor")
            For Each objItem in colItems
                  strClockSpeed = objItem.MaxClockSpeed
            Next
      wscript.echo "                 Getting Win32_PerfFormattedData_PerfOS_Processor WMI
data"
      Set colItems = objWMIService.ExecQuery("SELECT * FROM
Win32_PerfFormattedData_PerfOS_Processor WHERE Name = '_Total'")
```

```
            For Each objItem In colItems
            CPUPercent = objItem.PercentProcessorTime
            Next
      wscript.echo "                 Getting Win32_PerfFormattedData_PerfOS_Memory WMI
data"
      Set colItems = objWMIService.ExecQuery("SELECT * FROM
Win32_PerfFormattedData_PerfOS_Memory")
            For Each objItem In colItems
                  RAMPercent = objItem.PercentCommittedBytesInUse
            Next

      'Zone Info Routine
            wscript.echo "      - Collecting Zone Information..."
            For Each strZone In objFarm.Zones
                  For Each strZoneServer In strZone.Servers
                        if strZoneServer.ServerName = strServer.ServerName then
                              sZoneName = strZone.ZoneName
                              sZoneRank = fnZonePref(strZoneServer.ZoneRanking)
                              sZDC = False
                              if strZone.DataCollector = strServer.ServerName Then sZDC =
TRUE
                        end if
                  Next
            Next

      'Load Eval Routine
      wscript.echo "      - Collecting Load Evaluator Data..."
            set objServer = CreateObject("MetaFrameCOM.MetaFrameServer")
            objServer.Initialize 6, strServer.Servername
            Set objLE = objServer.AttachedLE
            objLE.Loaddata(True)
            evaluatorName = objLE.LEName

      wscript.echo "      - Collecting Server IMA Properties..."
      'Write to Database
      objRS.AddNew
            objRS("AuditID") = AuditID
            objRS("ServerName") = strServer.ServerName
            objRS("ServerFolder") = mid(strServer.ParentFolderDN, 9)
            objRS("XenAppBuild") = objWinServer.MFWinBuild
            objRS("XenAppLoad") = objWinServer.ServerLoad2
            objRS("ServerIPAddress") = strServer.IPAddress
            objRS("LoadEvaluator") = evaluatorName
            objRS("SessionCount") = strServer.SessionCount
            objRS("OperatingSystem") = objWinServer.WinNTVerMajor & "." &
objWinServer.WinNTVerMinor
            objRS("ServerMake") = strMfg
            objRS("ServerModel") = strModel
            objRS("ServerRAM") = strMemory
            objRS("ServerCPU") = strProcs
            objRS("ServerCPUSpeed") = strClockSpeed
            objRS("ZoneName") = sZoneName
            objRS("ZoneElectionPreference") = sZoneRank
            objRS("IsDataCollector") = sZDC
            objRS("ServerVideoBuffer") = objWinServer.ICAVideoBufferSize
            objRS("XenAppXML") = objWinServer.XMLPortNumber
            objRS("ContentRedir") = objWinServer.EnableContentRedirection
            'objRS("XenAppVersion") = objWinServer.MPSVersion
            objRS("XenAppEdition") = objWinServer.MPSEdition
            objRS("CPUPercent") = CPUPercent
            objRS("RAMPercent") = RAMPercent
      objRS.Update

      'Retrieve Record ID for Child Tables
      ServerInfoRS.Open "SELECT top 1 * FROM tblServers Where AuditID = " & AuditID & "
ORDER BY ServerID DESC" , objConn, 3, 3
            ServerInfoRS.movefirst
            ServerID = ServerInfoRS("ServerID")
      ServerInfoRS.Close

      'Server Printers
      wscript.echo "      - Collecting Installed Printers..."
            objRS2.Open "SELECT top 1 * FROM tblServersPrinters" , objConn, 3, 3
            For Each strDriver in strServer.PrinterDrivers
            wscript.echo "                 " & strDriver.DriverName
            objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("ServerID") = ServerID
                  objRS2("PrintDriver") = strDriver.DriverName
            objRS2.Update
            next
      objRS2.Close

      'Server Hotfixs
      wscript.echo "      - Collecting XenApp HotFix Data..."
            objRS2.Open "SELECT top 1 * FROM tblServersHF" , objConn, 3, 3
            For each strHotfix in objWinServer.HotFixes
            objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("ServerID") = ServerID
                  objRS2("HotFixName") = strHotFix.Name
            objRS2.Update
            next
      objRS2.Close


If strServer.SessionCount > 0 then

      'Server Processes
      wscript.echo "      - Collecting Process Info..."
            objRS2.Open "SELECT top 1 * FROM tblProcesses" , objConn, 3, 3
            'WMI Call

      Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" &
strComputer & "\root\cimv2")
      Set colProcessList = objWMIService.ExecQuery("Select * from Win32_Process")
      For Each objProcess in colProcessList
            colProperties = objProcess.GetOwner(strNameOfUser,strUserDomain)

            if strNameofUser = "SYSTEM" or strNameofUser = "LOCAL SERVICE" or
strNameofUser = "Ctx_StreamingSvc" or strNameofUser = "" or strNameofUser =
"ctx_cpsvcuser" or strNameofUser = "Ctx_ConfigMgr" or strNameofUser = "NETWORK SERVICE"
Then
                  'do Nothing
            Else
                  wscript.echo "                  Process: " &  objProcess.Name  & " is
owned by: " & strNameOfUser
                  objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("ServerID") = ServerID
                  objRS2("ServerName") = strcomputer
                  objRS2("ProcessName") = objProcess.Name
                  objRS2("ProcessID") =  objProcess.ProcessID
                  objRS2("ThreadCount") =  objProcess.ThreadCount
                  objRS2("PageFileUsage") =  objProcess.PageFileUsage
            objRS2("PageFaults") =  objProcess.PageFaults
            objRS2("WorkingSetSize") =  objProcess.WorkingSetSize
                  varProcTime = (CSng(objProcess.KernelModeTime) +
CSng(objProcess.UserModeTime)) / 10000000
                  objRS2("ProcessorTime") =  varProcTime
            objRS2("ProcessOwner") = strNameOfUser
                  objRS2("SessionID") =  objProcess.SessionID
                  objRS2("HandleCount") = objProcess.HandleCount
                  objRS2("ParentProcessID") =  objProcess.ParentProcessID
                  objRS2("VirtualMemorySize") = objProcess.VirtualSize
                  objRS2.Update
            end if
      next
      objRS2.Close
End If

      Next
      objRS.Close

' ************* APPLICATIONS
WScript.Echo "Collecting Application Data..."
objResultsFile.WriteLine "Collecting Application Data..."
      objRS.Open "SELECT top 1 * FROM tblApps" , objConn, 3, 3
      For Each strApp In objFarm.Applications
      strApp.LoadData(TRUE)
      wscript.echo "   " & strApp.AppName
      objResultsFile.WriteLine "   " & strApp.AppName
      Set AppObject = CreateObject("MetaFrameCOM.MetaFrameApplication")
      AppObject.Initialize MetaFrameWinAppObject,strApp.DistinguishedName
'     wscript.echo "      - Collecting IMA Application Properties..."
      AppObject.LoadData(TRUE)

      Set AppData = AppObject.WinAppObject

      dim WinType : WinType =
fnScreenRes(AppData.DefaultWindowType,AppData.DefaultWindowWidth,
AppData.DefaultWindowHeight, AppData.DefaultWindowScale)
      dim ColorDepth : ColorDepth = fnColorDepth(AppData.DefaultWindowColor)
      Dim CPUPriority : CPUPriority = fnCPUPriority(AppData.CPUPriority)
      dim Audio : Audio = fnAppAudio(AppData.DefaultSoundType)
      dim Encrypt : Encrypt = fnEncrypt(AppData.DefaultEncryption)
      Dim ApplicationType : ApplicationType =
fnappType(AppObject.AppType,AppData.DefaultInitProg)

      objRS.AddNew
            objRS("AuditID") = AuditID
            objRS("AppDN") = strApp.DistinguishedName
            objRS("BrowserName") = strApp.BrowserName
            objRS("AppName") = strApp.AppName
            objRS("AppCommandLine") = AppData.DefaultInitProg
            objRS("AppWorkingDir") = AppData.DefaultWorkDir
            objRS("AppType") = AppObject.AppType
            if AppObject.AppType = 17 then
                  objRS("AppType") = "Content"
                  set dContent = AppObject.ContentObject2
                  objRS("AppName") = dContent.ContentName
                  objRS("AppCommandLine") = dContent.ContentAddress
            end if
            if AppObject.AppType = 3 then
                  if len(AppData.DefaultInitProg) > 0 then
                        objRS("AppType") = "Server Installed"
                  else
                        objRS("AppType") = "Server Desktop"
                  end if
            end if
            if AppObject.AppType = 38 then objRS("AppType") = "Streamed to Server"
            objRS("Folder") = mid(AppData.ParentFolderDN, 14)
            objRS("WindowType") = WinType
            objRS("ColorDepth") = ColorDepth
            objRS("AudioQuality") = Audio
            objRS("Encryption") = Encrypt
            objRS("Priority") = CPUPriority
            if AppData.EnableApp = 0 then objRS("Disabled") = TRUE
            objRS("PNAgentFolder") = AppData.PNFolder
      objRS.Update

      'Retrieve Record ID for Child Tables
      objRS2.Open "SELECT top 1 * FROM tblApps Where AuditID = " & AuditID & " ORDER BY
AppID DESC" , objConn, 3, 3
            objRS2.movefirst
            AppID = objRS2("AppID")
      objRS2.Close
'     wscript.echo "      - Collecting Application Server Assignments..."
      objRS2.Open "SELECT top 1 * FROM tblAppsServers" , objConn, 3, 3
      Set AppServers = AppObject.Servers
      For Each strAppServer in AppServers
            objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("AppID") = AppID
                  objRS2("AppServerName") = strAppServer.ServerName
            objRS2.Update
      next
      objRS2.Close

'     wscript.echo "      - Collecting Application Group Assignments..."
      objRS2.Open "SELECT top 1 * FROM tblAppsGroups" , objConn, 3, 3
      Set AppGroups = AppObject.Groups
      For Each strGroup in AppGroups
            objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("AppID") = AppID
                  objRS2("GroupName") = strGroup.AAName & "\" & strGroup.GroupName
            objRS2.Update
      Next
      objRS2.Close

'     wscript.echo "      - Collecting Application User Assignments..."
      objRS2.Open "SELECT top 1 * FROM tblAppsUsers" , objConn, 3, 3
      Set AppUsers = AppObject.Users
      For Each strUser in AppUsers
            objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("AppID") = AppID
                  objRS2("UserName") = strUser.AAName & "\" & strUser.UserName
            objRS2.Update
      Next
      objRS2.Close
      Next
      objRS.Close

' ************* POLICIES
WScript.Echo "Collecting Policy Data..."
objResultsFile.WriteLine "Collecting Policy Data..."
      objRS.Open "SELECT top 1 * FROM tblPolicies" , objConn, 3, 3
      For Each strPolicy in objFarm.Policies(MetaframeUserPolicyObject)

      Set objPolicy = CreateObject("MetaframeCOM.MetaframePolicy")
      objPolicy.Initialize MetaframeUserPolicyObject, strPolicy.name
      objPolicy.LoadData(TRUE)

      objRS.AddNew
            objRS("AuditID") = AuditID
            objRS("PolicyName") = objPolicy.Name
            objRS("PolicyPriority") = objPolicy.Priority
            objRS("UserPolicy_DisableClientAudioMapping") =
objPolicy.UserPolicy.DisableClientAudioMapping
            objRS("UserPolicy_DisableClientCOMPortMapping") =
objPolicy.UserPolicy.DisableClientCOMPortMapping
            objRS("UserPolicy_DisableClientDriveMapping") =
objPolicy.UserPolicy.DisableClientDriveMapping
            objRS("UserPolicy_DisableFloppyDriveMapping") =
objPolicy.UserPolicy.DisableFloppyDriveMapping
            objRS("UserPolicy_DisableHardDriveMapping") =
objPolicy.UserPolicy.DisableHardDriveMapping
            objRS("UserPolicy_ConnectCDPolicy") = objPolicy.UserPolicy.ConnectCDPolicy
            objRS("UserPolicy_DisableNetDriveMapping") =
objPolicy.UserPolicy.DisableNetDriveMapping
            objRS("UserPolicy_ConnectClientDrives") =
objPolicy.UserPolicy.ConnectClientDrives
            objRS("UserPolicy_ConnectCDAtLogon") = objPolicy.UserPolicy.ConnectCDAtLogon
            objRS("UserPolicy_DisableClientLPTPortMapping") =
objPolicy.UserPolicy.DisableClientLPTPortMapping
            objRS("UserPolicy_PrinterBandWidth") = objPolicy.UserPolicy.PrinterBandWidth
            objRS("UserPolicy_PrinterBandWidthLimit") =
objPolicy.UserPolicy.PrinterBandWidthLimit
            objRS("UserPolicy_ConnectCPPolicy") = objPolicy.UserPolicy.ConnectCPPolicy
            objRS("UserPolicy_ConnectCPAtLogon") = objPolicy.UserPolicy.ConnectCPAtLogon
            objRS("UserPolicy_ConnectCPDeviceType") =
objPolicy.UserPolicy.ConnectCPDeviceType
            objRS("UserPolicy_DefaultToMainClientPrinter") =
objPolicy.UserPolicy.DefaultToMainClientPrinter
            objRS("UserPolicy_EnableDefaultToMainCP") =
objPolicy.UserPolicy.EnableDefaultToMainCP
            objRS("UserPolicy_DisableClientPrinterMapping") =
objPolicy.UserPolicy.DisableClientPrinterMapping
            objRS("UserPolicy_DisableOEMVirtualChannels") =
objPolicy.UserPolicy.DisableOEMVirtualChannels
            objRS("UserPolicy_ConcurrentSessionsPolicy") =
objPolicy.UserPolicy.ConcurrentSessionsPolicy
            objRS("UserPolicy_ConcurrentSessionsLimit") =
objPolicy.UserPolicy.ConcurrentSessionsLimit
            objRS("UserPolicy_ShadowUsersPolicy") =
objPolicy.UserPolicy.ShadowUsersPolicy
            objRS("UserPolicy_ShadowPolicy") = objPolicy.UserPolicy.ShadowPolicy
            objRS("UserPolicy_AllowShadow") = objPolicy.UserPolicy.AllowShadow
            objRS("UserPolicy_MustNotifyShadowee") =
objPolicy.UserPolicy.MustNotifyShadowee
            objRS("UserPolicy_DisableShadowerInput") =
objPolicy.UserPolicy.DisableShadowerInput
            objRS("UserPolicy_EstimateCLTPolicy") =
objPolicy.UserPolicy.EstimateCLTPolicy
            objRS("UserPolicy_UseCLTPolicy") = objPolicy.UserPolicy.UseCLTPolicy
            objRS("UserPolicy_EncryptionLevelPolicy") =
objPolicy.UserPolicy.EncryptionLevelPolicy
            objRS("UserPolicy_RequiredEncryptionLevel") =
objPolicy.UserPolicy.RequiredEncryptionLevel
            objRS("UserPolicy_DisableAutoClientUpdate") =
objPolicy.UserPolicy.DisableAutoClientUpdate
      objRS.Update
      Next
      objRS.Close

' ************* ADMINS
WScript.Echo "Collecting Administrators Data..."
objResultsFile.WriteLine "Collecting Administrators Data..."
      objRS.Open "SELECT top 1 * FROM tblMFAdmins" , objConn, 3, 3
      For Each strAdmin in objFarm.Admins
            objRS.AddNew
            objRS("AuditID") = AuditID
            objRS("MFAdminName") = strAdmin.AccountName
            objRS("MFAdminRole") = fnAdminLevel(strAdmin.AdminType)
            objRS.Update
      Next
      objRS.Close

' ************* LOAD EVALUATORS
WScript.Echo "Collecting Load Evaluator Data..."
objResultsFile.WriteLine "Collecting Load Evaluator Data..."
      objRS.Open "SELECT top 1 * FROM tblLoadEvals" , objConn, 3, 3
      For Each strEvaluator In objFarm.LoadEvaluators
            objRS.AddNew
            objRS("AuditID") = AuditID
            objRS("LEName") = strEvaluator.LEName
            objRS("LEDesc") = strEvaluator.Description
      objRS.Update

      'Retrieve Record ID for Child Tables
      objRS2.Open "SELECT top 1 * FROM tblLoadEvals Where AuditID = " & AuditID & " ORDER
BY LoadEvalID DESC" , objConn, 3, 3
            objRS2.movefirst
            LoadEvalID = objRS2("LoadEvalID")
      objRS2.Close

      'MetaFrameLoadEvaluator
      objRS2.Open "SELECT top 1 * FROM tblLoadEvalsRules" , objConn, 3, 3

      'Set objLE = CreateObject("MetaframeCOM.MetaframeLoadEvaluator")
      'Set objRules = CreateObject("MetaframeCOM.MetaframeLMRules")
      'objRules.Initialize MetaFrameLMRules, strEvaluator.LEname
      'objRules.LoadData(TRUE)

            For each objRule in strEvaluator.Rules
                  objRS2.AddNew
                  objRS2("AuditID") = AuditID
                  objRS2("LoadEvalID") = LoadEvalID
                  objRS2("LERuleType") = objRule.RuleType
                  objRS2("LERuleHWM") = objRule.HWM
                  objRS2("LERuleLWM") = objRule.LWM
                  objRS2.Update
            next
            objRS2.close
      Next
      objRS.Close

' ********* SESSIONS
wScript.Echo "Collecting Session Data..."
objResultsFile.WriteLine "Collecting Session Data..."
      objRS.Open "SELECT top 1 * FROM tblSessions" , objConn, 3, 3

      For Each Session In objFarm.Sessions
      Set Logon_Time=Session.LogonTime(True)
      Session_Connected=Right("00" & Logon_Time.Month, 2)&"/"& Right("00" &
Logon_Time.day, 2)&"/"& Right("2000" & Logon_Time.Year,4) &" "&Right("00" &
Logon_Time.Hour,2)&":"&Right("00" & Logon_Time.Minute, 2)
            objRS.AddNew
                  objRS("AuditID") = AuditID
                  objRS("UserName") = Session.UserName
                  objRS("ClientBuild") = Session.ClientBuild
                  objRS("ClientName") = Session.ClientName
                  objRS("ServerName") = Session.ServerName
                  objRS("SessionID") = Session.SessionID
                  objRS("SessionName") = Session.SessionName
                  objRS("SessionState") = fnSessionState(Session.SessionState)
                  objRS("LogonTime") = Session_Connected
            objRS.Update
      next
      objRS.Close

' *********************************************
wscript.echo "Audit Completed at " & now()
objResultsFile.WriteLine "Audit Completed at " & now()
objResultsFile.close
wscript.quit
' *********************************************

Function fnScreenRes(WindowType,wWidth,wHeight,wScale)
      Select Case WindowType
            Case 0
                  fnScreenRes = "1600x1200"
            Case 1
                  fnScreenRes = "640x480"
            Case 2
                  fnScreenRes = "800x600"
            Case 3
                  fnScreenRes = "1024x768"
            Case 4
                  fnScreenRes = "1280x1024"
            Case 5
                  fnScreenRes = wWidth & "x" & wHeight
            Case 6
                  fnScreenRes = wScale & "%"
            Case 7
                  fnScreenRes = "100%"
            Case Else
                  fnScreenRes = "Unknown: " & WindowType
      End Select
      End Function

Function fnColorDepth(ColorDepth)
      Select Case ColorDepth
            Case 1
                  fnColorDepth = "16 Colors"
            Case 2
                  fnColorDepth = "256 Colors"
            Case 3
                  fnColorDepth = "High Color (16 bit)"
            Case 4
                  fnColorDepth = "True Color (24 bit)"
            Case Else
                  fnColorDepth = "Unknown: " & ColorDepth
      End Select
      End Function

Function fnAppAudio(IMAAudio)
      Select Case IMAAudio
            Case 1
                  fnAppAudio = "None"
            Case 2
                  fnAppAudio = "Basic"
            Case Else
                  fnAppAudio = "Unknown: " & IMAAudio
      End Select
      End Function

Function fnEncrypt(IMAEncrypt)
      Select Case IMAEncrypt
            Case 1
                  fnEncrypt = "Basic"
            Case 2
                  fnEncrypt = "Logon Only"
            Case 3
                  fnEncrypt = "40-bit"
            Case 4
                  fnEncrypt = "56-bit"
            Case 5
                  fnEncrypt = "128-bit"
            Case Else
                  fnEncrypt = "Unknown: " & IMAEncrypt
      End Select
      End Function

Function fnAppType(AppType,AppCmd)
      Select Case AppType
            Case 3
                  If len(AppCmd) > 0 Then
                              fnAppType = "Server Installed"
                        Else
                              fnAppType = "Server Desktop"
                  End If
            Case 17
                  fnAppType = "Content"
            Case 38
                  fnAppType = "Streamed to Server"
            Case Else
                  fnAppType = "Unknown: " & fnAppType
      End Select
'StreamedToClient
'StreamedToClientOrInstalled
'StreamedToClientOrStreamedtoServer
      End Function

Function fnCPUPriority(IMACPU)
      Select Case IMACPU
            Case 1
                  fnCPUPriority = "Low"
            Case 2
                  fnCPUPriority = "Below Normal"
            Case 3
                  fnCPUPriority = "Normal"
            Case 4
                  fnCPUPriority = "Above Normal"
            Case 5
                  fnCPUPriority = "High"
            Case Else
                  fnCPUPriority = "Unknown: " & IMACPU
      End Select
      End Function
```

```
Function fnZonePref(ZonePref)
	Select Case ZonePref
		Case 1
			fnZonePref = "Most Preferred"
		Case 2
			fnZonePref = "Preferred"
		Case 3
			fnZonePref = "Default Preference"
		Case 4
			fnZonePref = "Not Preferred"
		Case Else
			fnZonePref = "Unknown: " & ZonePref
	End Select
	End Function

Function fnSessionState(SessionState)
	Select Case SessionState
		Case 1
			fnSessionState = "Active"
		Case 2
			fnSessionState = "Connected"
		Case 3
			fnSessionState = "Connecting"
		Case 4
			fnSessionState = "Shadowing"
		Case 5
			fnSessionState = "Disconnected"
		Case 6
			fnSessionState = "Idle"
		Case 7
			fnSessionState = "Listening"
		Case 8
			fnSessionState = "Resetting"
		Case 9
			fnSessionState = "Down"
		Case 10
			fnSessionState = "Initializing"
		Case 11
			fnSessionState = "Stale"
		Case 12
			fnSessionState = "Licensed"
		Case 13
			fnSessionState = "Unlicensed"
		Case 14
			fnSessionState = "Reconnected"
		Case Else
			fnSessionState = "Unknown: " & SessionState
	End Select
	End Function

Function fnAdminLevel(AdminRole)
	Select Case AdminRole
		Case 1
			fnAdminLevel = "Read Only"
		Case 2
```

```
                  fnAdminLevel = "Custom"
            Case 3
                  fnAdminLevel = "Full"
            Case Else
                  fnAdminLevel = "Unknown: " & AdminRole
      End Select
      End Function


</script>
</job>
</package>
```

XENAPPAUDIT.PS1

```
#           File:                 XenAppAudit.XA6.ps1
#           Description:          Audit XenApp Farm and write Access Database for Analysis
#           Requirements:         XenApp 6.0, XenApp PS SDK
#           Created by:           Andy Paul
#                                       www.paultechnologies.com
#
#           A comprehensive script recording farm, application, server, etc. properties.
#           Assumes C:\Program Files\XenAppTemp location but can be modified in code as
needed.
#           For x64 systems, run PowerShell x86
#
$curlocation = Get-location
$curdir = $curlocation.path
$progpath = $curdir -replace("Scripts")
if($progpath.substring($progpath.length - 1, 1) -ne "\" ){$progpath += "\"}
Write-Host "Program Path :==  "  $progpath -ForegroundColor Green
$resultsPath = $progPath + "Results"
$scriptsPath = $progPath + "Scripts"
$transcriptFile = $resultsPath + "\Transcript_PS.txt"
Write-Host "Transcript File :==  " $transcriptfile -ForegroundColor Green
start-transcript -path $transcriptFile
#Add Citrix SDK Snapins, from Citrix.XenApp.Sdk.ps1
if (Get-PSSnapin Citrix.Common.Commands -ea 0)
{
      Write-Host "Citrix.Common.Commands snapin already loaded" -ForegroundColor Yellow
}
else
{
      Write-Host "Loading Citrix.Common.Commands snapin..." -ForegroundColor Yellow
      Add-PSSnapIn Citrix.Common.Commands
}

# If the snap-in is registered (i.e. this is a XenApp server), then load it
if (Get-PSSnapin Citrix.XenApp.Commands -ea 0)
{
      Write-Host "Citrix.XenApp.Commands snapin already loaded" -ForegroundColor Yellow
}
else
{
      if (Get-PSSnapIn "Citrix.XenApp.Commands" -registered -ea 0)
      {
            Write-Host "Loading Citrix.XenApp.Commands snapin..." -ForegroundColor Yellow
            Add-PSSnapin "Citrix.XenApp.Commands"
      }
}

if (Get-PSSnapin Citrix.Common.GroupPolicy -ea 0)
{
      Write-Host "Citrix.Common.GroupPolicy snapin already loaded" -ForegroundColor
Yellow
}
else
{
      Write-Host "Loading Citrix.Common.GroupPolicy snapin..." -ForegroundColor Yellow
      Add-PSSnapIn Citrix.Common.GroupPolicy
}

#Import the remoting commands scripts module
if (Get-Module Citrix.XenApp.Commands.Remoting)
{
      Write-Host "Citrix.XenApp.Commands.Remoting module already loaded" -ForegroundColor
Yellow
}
else
{
      Write-Host "Loading Citrix.XenApp.Commands.Remoting module..." -ForegroundColor
Yellow
      Import-Module -name Citrix.XenApp.Commands.Remoting
}


Write-Host
Write-Host
$AuditDate = Get-Date -Format G
Write-Host "------------------------------------------------------------" -ForegroundColor
Green
Write-Host "Starting Collection at"$AuditDate -ForegroundColor Green
Write-Host "------------------------------------------------------------" -ForegroundColor
Green
$AuditID = 0
$dbName = $resultsPath + "\XenAppAudit_be.mdb"

$localhost=[environment]::MachineName
$localuser=[environment]::UserName

#Write-Host "Press any key to continue ..." -ForegroundColor Blue
#$x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")

$adOpenStatic = 3
$adLockOptimistic = 3
$connection = New-Object -com ADODB.Connection

$connection.Open("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=$dbName")
$objRSAudit = New-Object -com ADODB.Recordset
$objRSAudit.Open("Select top 1 * from tblAudit", $connection, $adOpenStatic,
$adLockOptimistic)
      $objRSAudit.AddNew()
      $objRSAudit.Fields.item('AuditServer').value = $localhost
      $objRSAudit.Fields.item('AuditDate').value = $AuditDate
      $objRSAudit.Fields.item('AuditUser').value = $localuser
      $objRSAudit.Update()
$objRSAudit.Close()

Write-Host
Write-Host "Collecting Farm Data..."

$farm = Get-XAFarm
$farmname = $farm.farmname
Write-Host "     " $farmname  -ForegroundColor Gray

$qryString = "Select top 1 * from tblAudit where AuditDate = #"
$qryString += $AuditDate
$qryString += "#"

$objRSAudit.Open($qryString, $connection, $adOpenStatic, $adLockOptimistic)
      $objRSAudit.MoveFirst()
      $AuditID = $objRSAudit.Fields.item('AuditID').value
      $objRSAudit.Fields.item('AuditFarmName').value = $farmname
      $objRSAudit.Update()
$objRSAudit.Close()

$objRSFarm = New-Object -com ADODB.Recordset
$objRSFarm.Open("Select top 1 * from tblFarms", $connection, $adOpenStatic,
$adLockOptimistic)
      $objRSFarm.AddNew()
      $objRSFarm.Fields.item('AuditID').value = $AuditID
      $objRSFarm.Fields.item('FarmName').value = $farmname
      $objRSFarm.Fields.item('FarmVersion').value = $farm.serverversion
      $objRSFarm.Update()
$objRSFarm.Close()

[string]$XenAppVersion = $farm.serverversion
Write-Host "     Farm Version:" $XenAppVersion
Write-Host "     Short Version:" $XenAppVersion.substring(0,3)

$qryString = "Select top 1 * from tblFarms where AuditID="
$qryString += $AuditID
$qryString += " ORDER BY FarmID DESC"

#Write-Host $qryString -ForegroundColor Yellow

$objRSFarm.Open($qryString , $connection, $adOpenStatic, $adLockOptimistic)
      $objRSFarm.MoveFirst()
      $FarmID = $objRSFarm.Fields.item('FarmID').value
$objRSFarm.Close()

#Write-Host "Press any key to continue ..." -ForegroundColor Blue
#$x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")

Write-Host
Write-Host "Collecting Server Data..."

$objRSServer = New-Object -com ADODB.Recordset
```

```
$objRSServer2 = New-Object -com ADODB.Recordset
$objRSPrinters = New-Object -com ADODB.Recordset
$objRSHF = New-Object -com ADODB.Recordset
$objRStblProcesses = New-Object -com ADODB.Recordset

$objRSServer.Open("Select top 1 * from tblServers", $connection, $adOpenStatic,
$adLockOptimistic)
$Servers = Get-XAServer
foreach($server in $servers){
            #This section shows information about the server name and the edition and
version of XenApp installed

      $strComputer = $Server.ServerName
      Write-Host "     " $strComputer  -ForegroundColor Gray
      Write-Host "         - Real-time WMI Data..." -ForegroundColor DarkGray

      #check PING Status
      $ping = new-object system.net.networkinformation.ping
      $pingreturns = $ping.send($strComputer)
      $pingstatus = $pingreturns.status
      Write-Host "                 Ping Status: " $pingstatus -ForegroundColor DarkGray
      if ($pingstatus -eq "Success")
      {
      $colSettings = get-wmiobject -class "Win32_ComputerSystem" -namespace "root\CIMV2"
-computername $strComputer
      Write-Host "                 Getting Win32_ComputerSystem WMI data"-ForegroundColor
DarkGray
      foreach($objComputer in $colSettings){
            $strMFG = $objComputer.Manufacturer
            $strModel = $objComputer.Model
            $strMemory = $objComputer.TotalPhysicalMemory/1000000/1024
            $strMemory  = [int] $strMemory
            $strProcs = $objComputer.NumberofLogicalProcessors
      }

      $colItems = get-wmiobject -class "Win32_Processor" -namespace "root\CIMV2" -
computername $strComputer
      Write-Host "                 Getting Win32_Processor WMI data"-ForegroundColor
DarkGray
      foreach($objItem in $colItems){$strClockSpeed = $objItem.MaxClockSpeed}


      $colItems = get-wmiobject -class "Win32_PerfFormattedData_PerfOS_Processor WHERE
Name = '_Total'" -namespace "root\CIMV2" -computername $strComputer
      Write-Host "                 Getting Win32_PerfFormattedData_PerfOS_Processor WMI
data"-ForegroundColor DarkGray
      foreach($objItem in $colItems){
            $strCPUPercent = $objItem.PercentProcessorTime
            $strCPUPercent = [int]$strCPUPercent
      }

      $colItems = get-wmiobject -class "Win32_PerfFormattedData_PerfOS_Memory" -namespace
"root\CIMV2" -computername $strComputer
      Write-Host "                 Getting Win32_PerfFormattedData_PerfOS_Memory WMI
data"-ForegroundColor DarkGray
      foreach($objItem in $colItems){
            $strRAMPercent = $objItem.PercentCommittedBytesInUse
            $strRAMPercent = [int]$strRAMPercent
      }
      }
      Else
      {
      $strMFG = "Unknown"
      $strModel = "Ping Failure"
      $strMemory  = $Null
      $strProcs = $Null
      $strClockSpeed = $Null
      $strCPUPercent = $Null
      $strRAMPercent = $Null
      }
      Write-Host "         - XenApp Configuration Data..." -ForegroundColor DarkGray

      $ZoneElection = switch ($Server.ElectionPreference) {
            "MostPreferred" {"Most Preferred"}
        "Preferred" {"Preferred"}
        "DefaultPreference" {"Default Preference"}
        "NotPreferred" {"Not Preferred"}
        default {$Server.ElectionPreference}}

      $objRSServer.AddNew()
            $objRSServer.Fields.item('AuditID').value = $AuditID
            $objRSServer.Fields.item('ServerName').value = $strComputer
            $objRSServer.Fields.item('ServerFolder').value = $Server.FolderPath
            $objRSServer.Fields.item('XenAppBuild').value = $Server.CitrixVersion
            $XenAppLoad = Get-XAServerLoad -ServerName $strComputer
                  $objRSServer.Fields.item('XenAppLoad').value = $XenAppLoad.load
            $objRSServer.Fields.item('ServerIPAddress').value = $Server.IPAddresses[0]
#First IP Address Only
            $LoadEvaluator = Get-XALoadEvaluator -Servername $strComputer
                  $objRSServer.Fields.item('LoadEvaluator').value =
$LoadEvaluator.LoadEvaluatorName
            $ServerFull = Get-XAServer -Servername $strComputer -Full
                  $objRSServer.Fields.item('SessionCount').value =
$ServerFull.SessionCount
            $objRSServer.Fields.item('OperatingSystem').value = $Server.OSVersion
            $objRSServer.Fields.item('ServerMake').value = $strMfg
            $objRSServer.Fields.item('ServerModel').value = $strModel
            $objRSServer.Fields.item('ServerRAM').value = $strMemory
            $objRSServer.Fields.item('ServerCPU').value = $strProcs
            $objRSServer.Fields.item('ServerCPUSpeed').value = $strClockSpeed
            $objRSServer.Fields.item('ZoneName').value = $Server.ZoneName
            $objRSServer.Fields.item('ZoneElectionPreference').value = $ZoneElection
            $ZDC = $False
                  $strZone = Get-XAZone
                  foreach($ZoneServer in $strZone){if ($ZoneServer.DataCollector -eq
$strComputer){$ZDC = $true}}
                  $objRSServer.Fields.item('IsDataCollector').value = $ZDC
            #$objRSServer.Fields.item('ServerVideoBuffer').value =
objWinServer.ICAVideoBufferSize
            #$objRSServer.Fields.item('XenAppXML').value = objWinServer.XMLPortNumber
            #$objRSServer.Fields.item('ContentRedir').value =
objWinServer.EnableContentRedirection
```

```
            #$objRSServer.Fields.item('XenAppVersion').value = $Server.CitrixVersion
            $objRSServer.Fields.item('XenAppEdition').value = $Server.CitrixEditionString
            $objRSServer.Fields.item('CPUPercent').value = $strCPUPercent
            $objRSServer.Fields.item('RAMPercent').value = $strRAMPercent
      $objRSServer.Update()

#ID for Child Tables
$qryString = "Select top 1 * from tblServers where AuditID="
$qryString += $AuditID
$qryString += " ORDER BY ServerID DESC"

#Write-Host $qryString -ForegroundColor Yellow

$objRSServer2.Open($qryString , $connection, $adOpenStatic, $adLockOptimistic)
      $objRSServer2.MoveFirst()
      $ServerID = $objRSServer2.Fields.item('ServerID').value
$objRSServer2.Close()

#Server Printers
Write-Host "         - Installed Printers..." -ForegroundColor DarkGray
$objRSPrinters.Open("SELECT top 1 * FROM tblServersPrinters", $connection, $adOpenStatic,
$adLockOptimistic)
      Update-XAPrinterDriver -ServerName $strComputer
      $PrintDrivers = Get-XAPrinterDriver -ServerName $strComputer
      if($PrintDrivers.count -gt 0){
      foreach($Printer in $PrintDrivers){
      $objRSPrinters.AddNew()
            $objRSPrinters.Fields.item('AuditID').value = $AuditID
            $objRSPrinters.Fields.item('ServerID').value = $ServerID
            if($XenAppVersion.startswith("6.0"))
                  {$objRSPrinters.Fields.item('PrintDriver').value = $Printer
                        Write-Host "                 " $Printer -ForegroundColor
DarkGray}
            else
                  {$objRSPrinters.Fields.item('PrintDriver').value = $Printer.DriverName
                        Write-Host "                 " $Printer.DriverName -
ForegroundColor DarkGray}
      $objRSPrinters.Update()
}}
$objRSPrinters.Close()

#Server HotFixes
Write-Host "         - Installed XenApp HotFixes..." -ForegroundColor DarkGray
$objRSHF.Open("SELECT top 1 * FROM tblServersHF", $connection, $adOpenStatic,
$adLockOptimistic)
      $ServerHF = Get-XAServerHotfix -ServerName $strComputer
      if($ServerHF.count -gt 0){
      foreach($HotFix in $ServerHF){
      $objRSHF.AddNew()
            $objRSHF.Fields.item('AuditID').value = $AuditID
            $objRSHF.Fields.item('ServerID').value = $ServerID
            $objRSHF.Fields.item('HotfixName').value = $HotFix.HotfixName
      $objRSHF.Update()
      }}
$objRSHF.Close()
```

```
#Server Processes
Write-Host "         - Server User Processes..." -ForegroundColor DarkGray
$objRStblProcesses.Open("SELECT top 1 * FROM tblProcesses", $connection, $adOpenStatic,
$adLockOptimistic)
      $colProcesses = get-wmiobject -class "Win32_Process" -namespace "root\CIMV2" -
computername $strComputer
      foreach($objProcess in $colProcesses){
            $ErrorActionPreference = "SilentlyContinue"
            [string]$ProcessUserName = $null
            $ProcessUserName = $objProcess.getowner().user
            [string]$ProcessName = $objProcess.name
            $ErrorActionPreference = "Continue"
            if ($ProcessUserName -eq "SYSTEM") {}
            elseif ($ProcessUserName -eq "LOCAL SERVICE"){}
            elseif ($ProcessUserName -eq "Ctx_StreamingSvc"){}
            elseif ($ProcessUserName -eq "") {}
            elseif ($ProcessUserName -eq "ctx_cpsvcuser") {}
            elseif ($ProcessUserName -eq "Ctx_ConfigMgr") {}
            elseif ($ProcessUserName -eq "NETWORK SERVICE") {}
            elseif ($ProcessUserName -eq $Null) {}
            Else {
                  Write-Host "                  Process: " $ProcessName " is owned by: "
$ProcessUserName -ForegroundColor DarkGray
                  $objRStblProcesses.AddNew()
                  $objRStblProcesses.Fields.item('AuditID').value = $AuditID
                  $objRStblProcesses.Fields.item('ServerID').value = $ServerID
                  $objRStblProcesses.Fields.item('ServerName').value = $strComputer
                  $objRStblProcesses.Fields.item('ProcessName').value = $ProcessName
                  $objRStblProcesses.Fields.item('ProcessID').value =
$objProcess.ProcessID
                  $objRStblProcesses.Fields.item('ThreadCount').value =
$objProcess.ThreadCount
                  $objRStblProcesses.Fields.item('PageFileUsage').value =
$objProcess.PageFileUsage
                  $objRStblProcesses.Fields.item('PageFaults').value =
$objProcess.PageFaults
                  $objRStblProcesses.Fields.item('WorkingSetSize').value =
$objProcess.WorkingSetSize
                  [single]$kernelTime = $objProcess.KernelModeTime
                  [single]$usertime = $objProcess.UserModeTime
                  $objRStblProcesses.Fields.item('ProcessorTime').value =
($kerneltime+$usertime)/10000000
                  $objRStblProcesses.Fields.item('ProcessOwner').value = $ProcessUserName
                  $objRStblProcesses.Fields.item('SessionID').value =
$objProcess.sessionid
                  $objRStblProcesses.Fields.item('HandleCount').value =
$objProcess.HandleCount
                  $objRStblProcesses.Fields.item('ParentProcessID').value =
$objProcess.ParentProcessID
                  $objRStblProcesses.Fields.item('VirtualMemorySize').value =
$objProcess.VirtualSize
                  $objRStblProcesses.Update()
            }}
$objRStblProcesses.Close()
$ErrorActionPreference = "Continue"
}
```

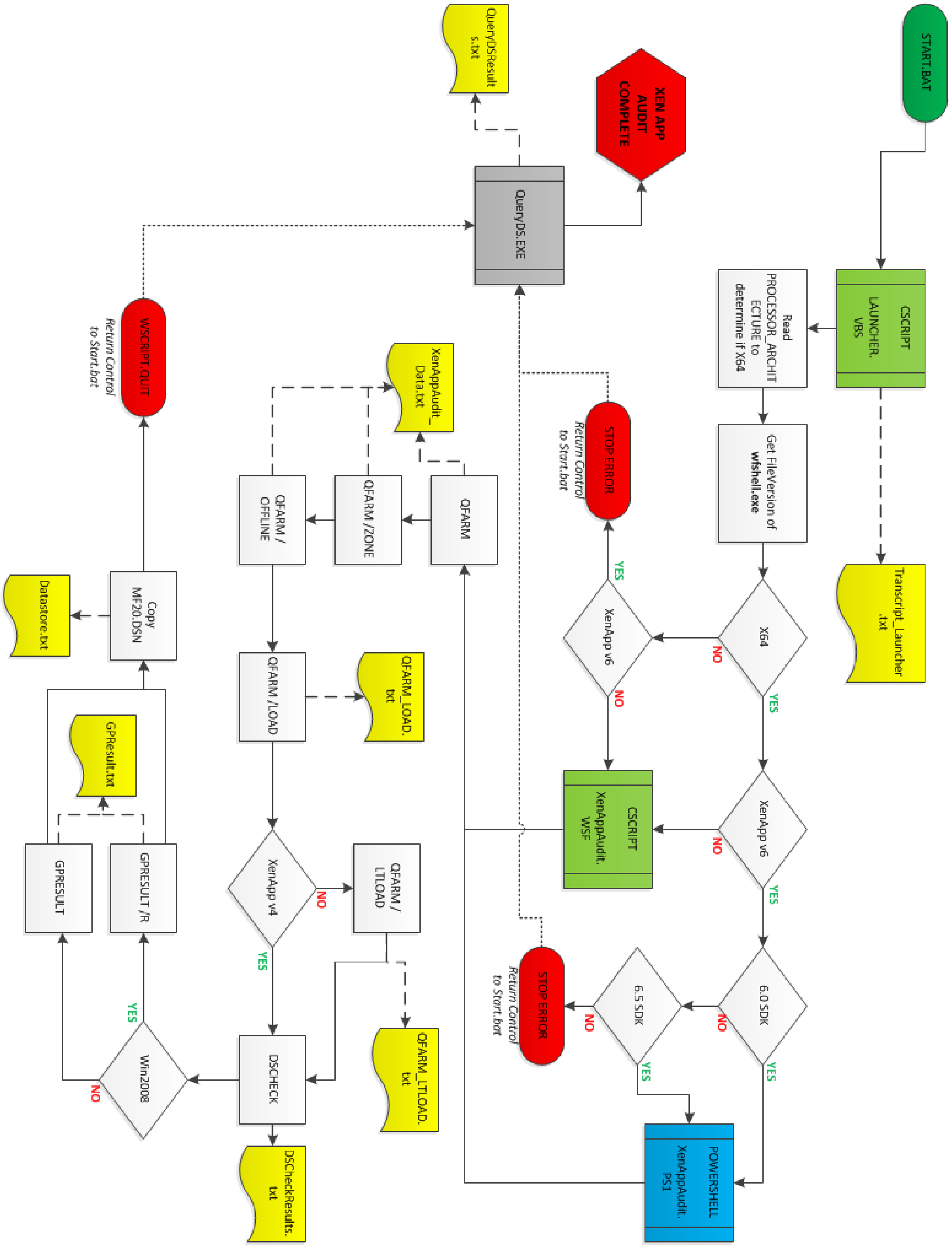
```
$objRSServer.Close()

Write-Host
Write-Host "Collecting Application Data..."

$objRSApps = New-Object -com ADODB.Recordset
$objRSApps2 = New-Object -com ADODB.Recordset
$objRSAppsServers = New-Object -com ADODB.Recordset
$objRSAppsGroups = New-Object -com ADODB.Recordset
$objRSAppsUsers = New-Object -com ADODB.Recordset
$objRSAppsWorkerGroups = New-Object -com ADODB.Recordset

$objRSApps.Open("Select top 1 * from tblApps", $connection, $adOpenStatic, 
$adLockOptimistic)
$Applications = Get-XAApplication
foreach($app in $Applications){
      $AppName = $app.DisplayName
      $BrowserName = $app.BrowserName
      Write-Host "     " $AppName  -ForegroundColor Gray
      Write-Host "         - XenApp Application Properties..." -ForegroundColor DarkGray

      $ColorDepth = switch ($app.ColorDepth)     {
        "Colors8Bit" {"8 Bit"}
        "Colors16Bit" {"16 Bit"}
        "Colors32Bit" {"32 Bit"}
        default {$app.ColorDepth}    }

      $Audio = switch ($app.AudioType)     {
        "None" {"None"}
        "Basic" {"Basic"}
        default {$app.AudioType}     }

      $CpuPriorityLevel = switch ($app.CpuPriorityLevel)     {
        "BelowNormal" {"Below Normal"}
        "Low" {"Low"}
            "Normal" {"Normal"}
            "AboveNormal" {"Above Normal"}
            "High" {"High"}
        default {$app.CpuPriorityLevel}    }

      $ApplicationType = switch ($app.ApplicationType)     {
        "ServerInstalled" {"Server Installed"}
        "ServerDesktop" {"Server Desktop"}
            "Content" {"Content"}
            "StreamedToServer" {"Streamed to Server"}
            "StreamedToClient" {"Streamed to Client"}
            "StreamedToClientOrInstalled" {"Streamed to Client or Installed"}
            "StreamedToClientOrStreamedToServer" {"Streamed to Client or Streamed to 
Server"}
        default {$app.ApplicationType}    }

      $EncryptionLevel = switch ($app.EncryptionLevel) {
        "Basic" {"Basic"}
        "LogOn" {"Logon Only"}
            "Bits40" {"40-bit"}
            "Bits56" {"56-bit"}
            "Bits128" {"128-bit"}
        default {$app.EncryptionLevel}   }

      $objRSApps.AddNew()
            $objRSApps.Fields.item('AuditID').value = $AuditID
            $objRSApps.Fields.item('AppName').value = $AppName
            $objRSApps.Fields.item('BrowserName').value = $BrowserName
            $objRSApps.Fields.item('AppCommandLine').value = $app.CommandLineExecutable
            #ContentAddress Logic for Command Line
            $objRSApps.Fields.item('AppWorkingDir').value = $app.WorkingDirectory
            $AppDN = $app.FolderPath+"/"+$app.AppName
                  $objRSApps.Fields.item('AppDN').value = $AppDN
            $objRSApps.Fields.item('WindowType').value = $app.WindowType
            $objRSApps.Fields.item('ColorDepth').value = $ColorDepth
            $objRSApps.Fields.item('AudioQuality').value = $Audio
            $objRSApps.Fields.item('Encryption').value = $EncryptionLevel
            $objRSApps.Fields.item('Folder').value = $app.FolderPath
            $Disabled = if($app.Enabled){$False} else {$True}
                  $objRSApps.Fields.item('Disabled').value = $Disabled
            $objRSApps.Fields.item('PNAgentFolder').value = $app.ClientFolder
            $objRSApps.Fields.item('AppType').value = $ApplicationType
            $objRSApps.Fields.item('Priority').value = $CpuPriorityLevel
      $objRSApps.Update()

#ID for Child Tables
$qryString = "Select top 1 * from tblApps where AuditID="
$qryString += $AuditID
$qryString += " ORDER BY AppID DESC"

$objRSApps2.Open($qryString , $connection, $adOpenStatic, $adLockOptimistic)
      $objRSApps2.MoveFirst()
      $AppID = $objRSApps2.Fields.item('AppID').value
$objRSApps2.Close()

$AppReport = get-XAApplicationReport $BrowserName

#App Servers
Write-Host "         - Application Server Assignments..." -ForegroundColor DarkGray
$objRSAppsServers.Open("SELECT top 1 * FROM tblAppsServers", $connection, $adOpenStatic,
$adLockOptimistic)
      $ascount = $AppReport.ServerNames
      if($ascount.count -gt 0){
      foreach($AppServer in $AppReport.ServerNames){
      $objRSAppsServers.AddNew()
            $objRSAppsServers.Fields.item('AuditID').value = $AuditID
            $objRSAppsServers.Fields.item('AppID').value = $AppID
            $objRSAppsServers.Fields.item('AppServerName').value = $AppServer
      $objRSAppsServers.Update()
      }}
$objRSAppsServers.Close()

#App WorkerGroups
Write-Host "         - Application Worker Group Assignments..." -ForegroundColor DarkGray
$objRSAppsWorkerGroups.Open("SELECT top 1 * FROM tblAppsWorkerGroups", $connection,
$adOpenStatic, $adLockOptimistic)
      $awgcount = $AppReport.WorkerGroupNames
```

```
      if($awgcount.count -gt 0){
      foreach($AppWG in $AppReport.WorkerGroupNames){
      $objRSAppsWorkerGroups.AddNew()
            $objRSAppsWorkerGroups.Fields.item('AuditID').value = $AuditID
            $objRSAppsWorkerGroups.Fields.item('AppID').value = $AppID
            $objRSAppsWorkerGroups.Fields.item('WorkerGroup').value = $AppWG
      $objRSAppsWorkerGroups.Update()
      }}
$objRSAppsWorkerGroups.Close()

#App Users & Groups
Write-Host "          - Application Account Assignments..." -ForegroundColor DarkGray
$objRSAppsGroups.Open("SELECT top 1 * FROM tblAppsGroups", $connection, $adOpenStatic,
$adLockOptimistic)
$objRSAppsUsers.Open("SELECT top 1 * FROM tblAppsUsers", $connection, $adOpenStatic,
$adLockOptimistic)
$appassignedusers = get-xaaccount -browsername $BrowserName
      foreach($AccessAccount in $appassignedusers){
            if ($AccessAccount.AccountType -eq 4)   #Group
            {$objRSAppsGroups.AddNew()
                  $objRSAppsGroups.Fields.item('AuditID').value = $AuditID
                  $objRSAppsGroups.Fields.item('AppID').value = $AppID
                  $objRSAppsGroups.Fields.item('GroupName').value =
$AccessAccount.AccountDisplayName
            $objRSAppsGroups.Update()}
            if ($AccessAccount.AccountType -eq 1)   #User
            {$objRSAppsUsers.AddNew()
                  $objRSAppsUsers.Fields.item('AuditID').value = $AuditID
                  $objRSAppsUsers.Fields.item('AppID').value = $AppID
                  $objRSAppsUsers.Fields.item('UserName').value =
$AccessAccount.AccountDisplayName
            $objRSAppsUsers.Update()}
      }
      $objRSAppsGroups.close()
      $objRSAppsUsers.close()
}
$objRSApps.Close()

#WorkerGroups
Write-Host
Write-Host "Collecting Worker Group Data..."
$objRSWorkerGroups = New-Object -com ADODB.Recordset
$objRSWorkerGroup2 = New-Object -com ADODB.Recordset
$objRSWorkerGroupsServers = New-Object -com ADODB.Recordset
$objRSWorkerGroupsSGs = New-Object -com ADODB.Recordset
$objRSWorkerGroupsOU = New-Object -com ADODB.Recordset
$objRSWorkerGroups.Open("Select top 1 * from tblWorkerGroups", $connection,
$adOpenStatic, $adLockOptimistic)
$objRSWorkerGroupsServers.Open("Select top 1 * from tblWorkerGroupsServers", $connection,
$adOpenStatic, $adLockOptimistic)
$objRSWorkerGroupsSGs.Open("Select top 1 * from tblWorkerGroupsSG", $connection,
$adOpenStatic, $adLockOptimistic)
$objRSWorkerGroupsOU.Open("Select top 1 * from tblWorkerGroupsOU", $connection,
$adOpenStatic, $adLockOptimistic)
$WGs = Get-XAWorkerGroup
foreach($WG in $WGs){
      $objRSWorkerGroups.AddNew()
            $objRSWorkerGroups.Fields.item('AuditID').value = $AuditID
            $objRSWorkerGroups.Fields.item('WorkerGroup').value = $WG.WorkerGroupName
            $objRSWorkerGroups.Fields.item('Description').value = $WG.Description
            $objRSWorkerGroups.Fields.item('FolderPath').value = $WG.FolderPath
      $objRSWorkerGroups.Update()

      #ID for Child Tables
      $qryString = "Select top 1 * from tblWorkerGroups where AuditID="
      $qryString += $AuditID
      $qryString += " ORDER BY WorkerGroupID DESC"

      $objRSWorkerGroup2.Open($qryString , $connection, $adOpenStatic, $adLockOptimistic)
            $objRSWorkerGroup2.MoveFirst()
      $WorkerGroupID = $objRSWorkerGroup2.Fields.item('WorkerGroupID').value
      $objRSWorkerGroup2.Close()

      foreach($wgserver in $wg.servernames){
      $objRSWorkerGroupsServers.AddNew()
            $objRSWorkerGroupsServers.Fields.item('AuditID').value = $AuditID
            $objRSWorkerGroupsServers.Fields.item('WorkerGroupID').value = $WorkerGroupID
            $objRSWorkerGroupsServers.Fields.item('ServerName').value = $wgserver
      $objRSWorkerGroupsServers.Update()}

      foreach($wgsvrgrp in $wg.servergroups){
      $objRSWorkerGroupsSGs.AddNew()
            $objRSWorkerGroupsSGs.Fields.item('AuditID').value = $AuditID
            $objRSWorkerGroupsSGs.Fields.item('WorkerGroupID').value = $WorkerGroupID
            $objRSWorkerGroupsSGs.Fields.item('ServerGroup').value = $wgsvrgrp
      $objRSWorkerGroupsSGs.Update()}

      foreach($wgou in $wg.OUs){
      $objRSWorkerGroupsOU.AddNew()
            $objRSWorkerGroupsOU.Fields.item('AuditID').value = $AuditID
            $objRSWorkerGroupsOU.Fields.item('WorkerGroupID').value = $WorkerGroupID
            $objRSWorkerGroupsOU.Fields.item('OUName').value = $wgou
      $objRSWorkerGroupsOU.Update()}
}
$objRSWorkerGroupsOU.Close()
$objRSWorkerGroupsSGs.Close()
$objRSWorkerGroupsServers.Close()
$objRSWorkerGroups.Close()


#Admins
Write-Host
Write-Host "Collecting Farm Administrators Data..."
$objRSAdmins = New-Object -com ADODB.Recordset
$objRSAdmins.Open("Select top 1 * from tblMFAdmins", $connection, $adOpenStatic,
$adLockOptimistic)
$mfadmins = Get-XAAdministrator
foreach($mfadmin in $mfadmins){
      $MFAdminType = switch ($mfadmin.AdministratorType){
        "Full" {"Full"}
        "ViewOnly" {"View Only"}
        "Custom" {"Custom"}
        default {$mfadmin.AdministratorType}}

      $objRSAdmins.AddNew()
            $objRSAdmins.Fields.item('AuditID').value = $AuditID
            $objRSAdmins.Fields.item('MFAdminName').value = $mfadmin.AdministratorName
            $objRSAdmins.Fields.item('MFAdminRole').value = $MFAdminType
      $objRSAdmins.Update()
}
$objRSAdmins.close()

#Load Evaluators
Write-Host
Write-Host "Collecting Load Evaluator Data..."
$objRSLoadEval = New-Object -com ADODB.Recordset
$objRSLoadEval2 = New-Object -com ADODB.Recordset
$objRSLoadEvalRules = New-Object -com ADODB.Recordset
$objRSLoadEval.Open("Select top 1 * from tblLoadEvals", $connection, $adOpenStatic,
$adLockOptimistic)
$XALE = Get-XALoadEvaluator
foreach($LE in $XALE){
      $objRSLoadEval.AddNew()
            $objRSLoadEval.Fields.item('AuditID').value = $AuditID
            $objRSLoadEval.Fields.item('LENAme').value = $LE.LoadEvaluatorName
            $objRSLoadEval.Fields.item('LEDesc').value = $LE.Description
      $objRSLoadEval.Update()

#ID for Child Tables
$qryString = "Select top 1 * from tblLoadEvals where AuditID="
$qryString += $AuditID
$qryString += " ORDER BY LoadEvalID DESC"

$objRSLoadEval2.Open($qryString , $connection, $adOpenStatic, $adLockOptimistic)
      $objRSLoadEval2.MoveFirst()
      $LoadEvalID = $objRSLoadEval2.Fields.item('LoadEvalID').value
$objRSLoadEval2.Close()

$objRSLoadEvalRules.Open("Select top 1 * from tblLoadEvalsRules", $connection,
$adOpenStatic, $adLockOptimistic)
      if($LE.ApplicationUserLoadEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Application User
Load"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value =
$LE.ApplicationUserLoad
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $null
            $objRSLoadEvalRules.Update()}
      if($LE.ContextSwitchesEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Context
Switches"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value =
$LE.ContextSwitches[1]
```

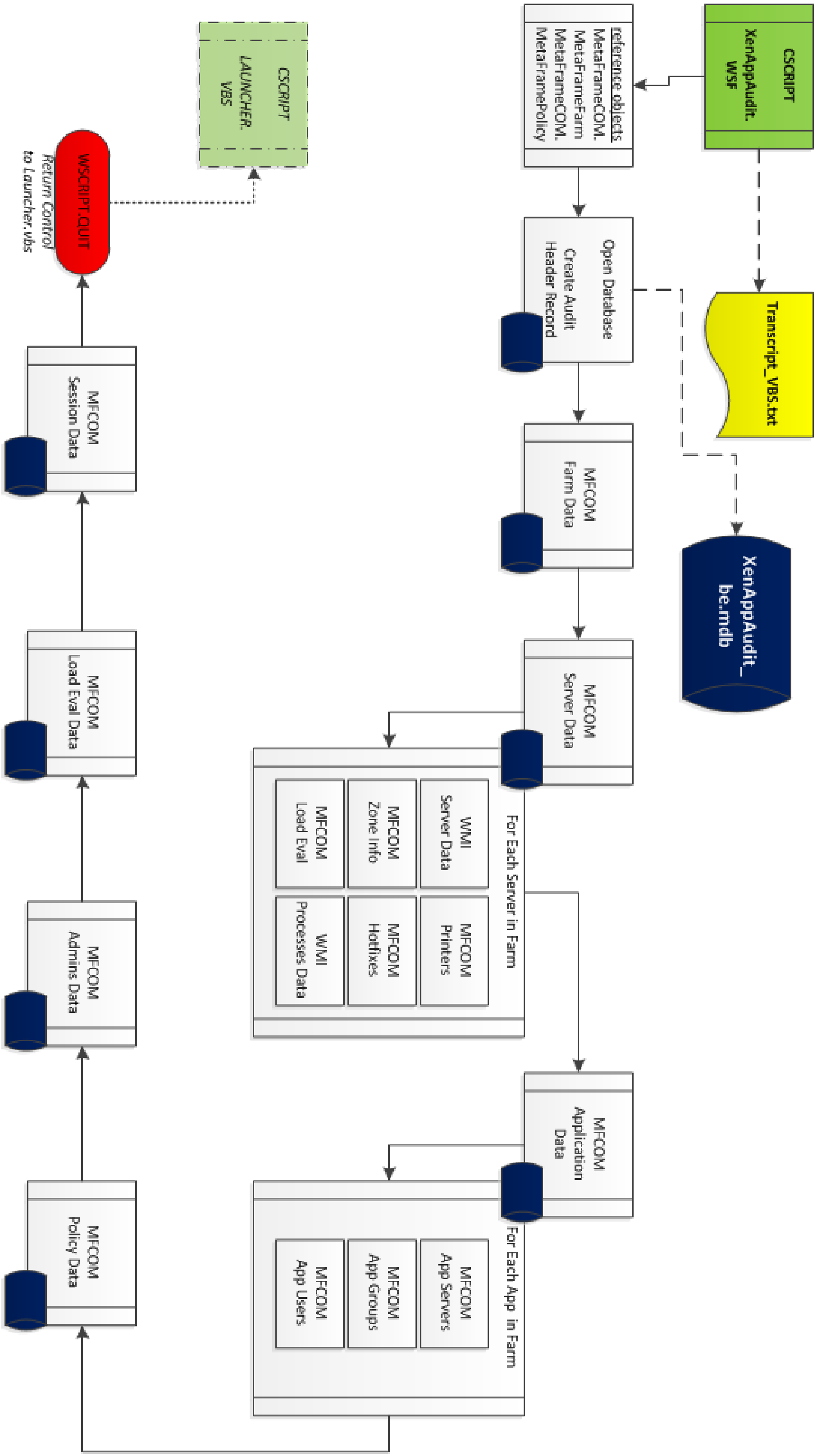
```
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value =
$LE.ContextSwitches[0]
            $objRSLoadEvalRules.Update()}
      if($LE.CpuUtilizationEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "CPU Utilization"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value =
$LE.CpuUtilization[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value =
$LE.CpuUtilization[0]
            $objRSLoadEvalRules.Update()}
      if($LE.DiskDataIOEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Disk Data IO"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.DiskDataIO[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $LE.DiskDataIO[0]
            $objRSLoadEvalRules.Update()}
      if($LE.DiskOperationsEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Disk Operations"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value =
$LE.DiskOperations[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value =
$LE.DiskOperations[0]
            $objRSLoadEvalRules.Update()}
      if($LE.IPRangesEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "IP Ranges"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.IPRanges[-1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $LE.IPRanges[0]
            $objRSLoadEvalRules.Update()}
      if($LE.LoadThrottlingEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Load Throttling"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.LoadThrottling
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $NULL
            $objRSLoadEvalRules.Update()}
      if($LE.MemoryUsageEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Memory Usage"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.MemoryUsage[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $LE.MemoryUsage[0]
            $objRSLoadEvalRules.Update()}
      if($LE.PageFaultsEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Page Faults"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.PageFaults[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $LE.PageFaults[0]
            $objRSLoadEvalRules.Update()}
      if($LE.PageSwapsEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Page Swaps"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.PageSwaps[1]
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $LE.PageSwaps[0]
            $objRSLoadEvalRules.Update()}
      if($LE.ScheduleEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Schedule
Enabled"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $NULL
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $NULL
            $objRSLoadEvalRules.Update()}
      if($LE.ServerUserLoadEnabled){
            $objRSLoadEvalRules.AddNew()
                  $objRSLoadEvalRules.Fields.item('AuditID').value = $AuditID
                  $objRSLoadEvalRules.Fields.item('LoadEvalID').value = $LoadEvalID
                  $objRSLoadEvalRules.Fields.item('LERuleType').value = "Server User
Load"
                  $objRSLoadEvalRules.Fields.item('LERuleHWM').value = $LE.ServerUserLoad
                  $objRSLoadEvalRules.Fields.item('LERuleLWM').value = $NULL
            $objRSLoadEvalRules.Update()}
$objRSLoadEvalRules.Close()
}
$objRSLoadEval.Close()


#Sessions
Write-Host
Write-Host "Collecting Session Data..."
$objRSSessions = New-Object -com ADODB.Recordset
$objRSSessions.Open("Select top 1 * from tblSessions", $connection, $adOpenStatic,
$adLockOptimistic)
$Sessions = Get-XASession -Farm -Full
foreach($Session in $Sessions){
      $SessionState = switch ($Session.State)
    {
        "Active" {"Active"}
        "Connected" {"Connected"}
            "Connecting" {"Connecting"}
            "Shadowing" {"Shadowing"}
            "Disconnected" {"Disconnected"}
        "Idle" {"Idle"}
        "Listening" {"Listening"}
            "Resetting" {"Resetting"}
            "Down" {"Down"}
            "Initializing" {"Initializing"}
        "Stale" {"Stale"}
            "Licensed" {"Licensed"}
            "Unlicensed" {"Unlicensed"}
            "Reconnected" {"Reconnected"}
        default {$Session.State}
    }
      $objRSSessions.AddNew()
            $objRSSessions.Fields.item('AuditID').value = $AuditID
            $objRSSessions.Fields.item('UserName').value = $Session.AccountName
            $objRSSessions.Fields.item('ClientBuild').value = $Session.ClientBuildNumber
            $objRSSessions.Fields.item('ClientName').value = $Session.ClientName
            $objRSSessions.Fields.item('ServerName').value = $Session.ServerName
            $objRSSessions.Fields.item('SessionID').value = $Session.SessionID
            $objRSSessions.Fields.item('SessionName').value = $Session.SessionName
            $objRSSessions.Fields.item('SessionState').value = $SessionState
            $objRSSessions.Fields.item('LogonTime').value = $Session.ConnectTime
      $objRSSessions.Update()
}
$objRSSessions.close()


$connection.Close()
$EndTime = Get-Date -Format G
Write-Host
Write-Host "-----------------------------------------------------------" -ForegroundColor
Red
Write-Host "Audit #" $AuditID "Completed at" $EndTime -ForegroundColor Red
Write-Host "-----------------------------------------------------------" -ForegroundColor
Red
Write-Host
stop-transcript
#Write-Host "Press any key to continue ..." -ForegroundColor Blue
#Write-Host
#$x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")
```

Appendix C – Program Workflows

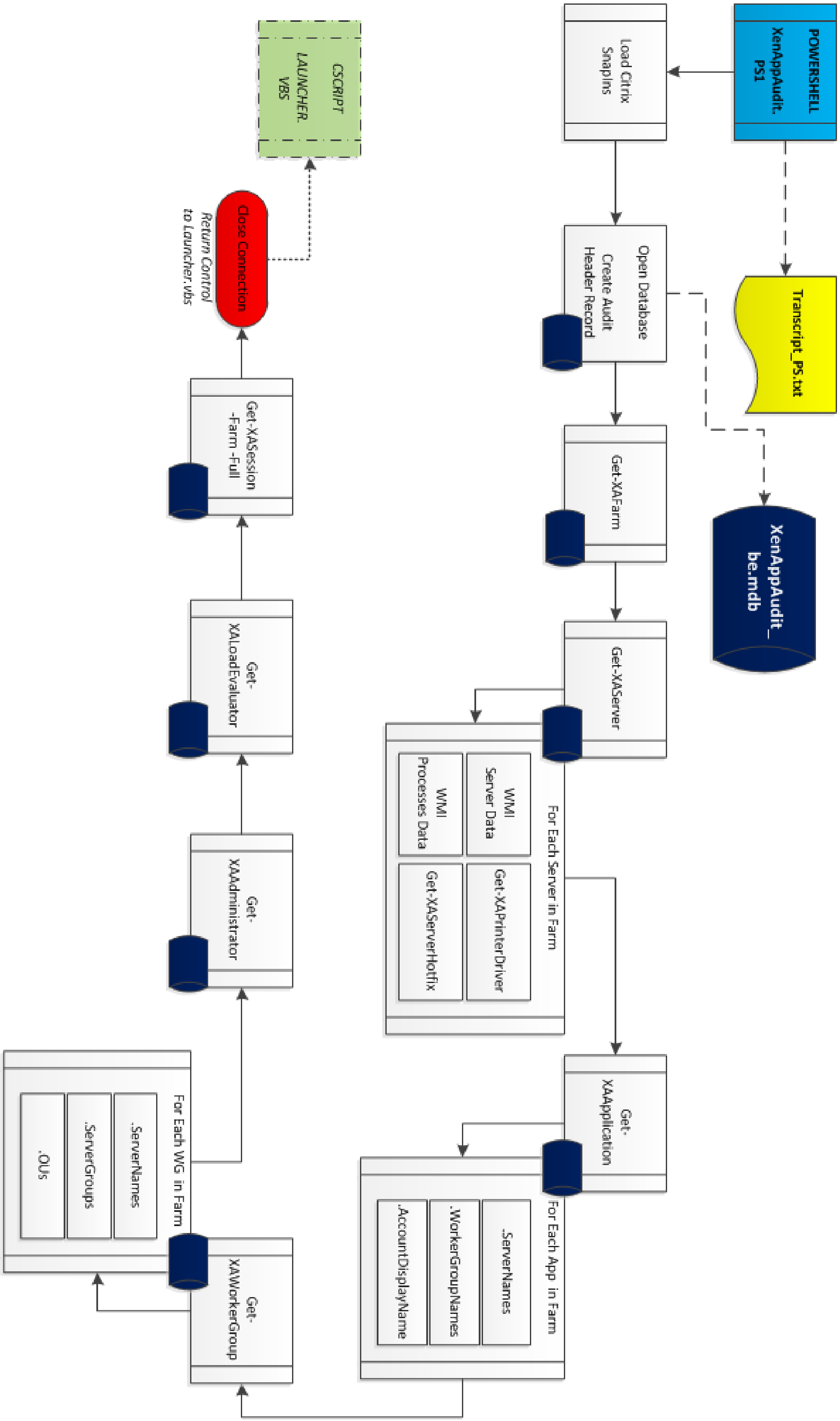
Start & Launcher Workflow



XenAppAudit.WSF Workflow



XenAppAudit.PS1 Workflow


POWERSHELL
XenAppAudit.PS1
Load Citrix SnapIns
Transcript_PS.txt
Open Database
Create Audit Header Record
XenAppAudit_be.mdb
Get-XAFarm
Get-XAServer
For Each Server in Farm
WMI Server Data
WMI Processes Data
Get-XAPrinterDriver
Get-XAServerHotfix
Get-XAApplication
For Each App in Farm
.ServerNames
.WorkerGroupNames
.AccountDisplayName
Get-XAWorkerGroup
For Each WG in Farm
.ServerNames
.ServerGroups
.OUs
Get-XAAdministrator
Get-XALoadEvaluator
Get-XASession -Farm -Full
Close Connection
Return Control to Launcher.vbs
CSCRIPT LAUNCHER.VBS