



**National Institute of
Standards and Technology**
Technology Administration
U.S. Department of Commerce

NISTIR 6951

Automated Security Self-Evaluation Tool Technical Documentation

Version 1.03

Mark McLarnon
Marianne Swanson, Editor

NISTIR 6951

***Automated Security Self-
Evaluation Tool Technical
Documentation, Version 1.03***

The National Institute of Standards and
Technology

C O M P U T E R S E C U R I T Y

January 31, 2003

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2003

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov — Phone: (202) 512-1800 — Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Acknowledgements

The author Mark McLarnon of Booz Allen Hamilton and the editor Marianne Swanson of NIST would like to acknowledge the efforts of Mike Indovina, John Wack, Elizabeth Lennon of NIST and Marc Stevens of Booz Allen for their review and assistance in developing the drafts and final copy of this document.

THIS PAGE INTENTIONALLY LEFT BLANK.

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Audience	1
1.3 Description of ASSET	1
1.4 References	1
1.5 License	2
1.6 Conventions	2
2. Development Environment	3
2.1 Operating System Settings	3
2.2 MSDE Settings	3
2.3 ODBC Settings	3
2.4 Visual Café Settings	4
2.4.1 Project Output	5
2.4.2 Third Party JAR Files	7
2.4.3 Classpath Settings	8
2.4.4 Folder Configuration	8
2.4.5 File Requirements	9
2.5 Java Command Line Compilation	15
2.5.1 Compile ASSET with Symantec/WebGain Compiler	15
2.5.2 Compile ASSET with Sun Java Compiler	17
3. ASSET Components	18
3.1 User Interface	18
3.1.1 Swing 1.1	18
3.1.2 MDI Interface	22
3.2 Class Design	23
3.3 ASSET System	24
3.3.1 ASSET System Operation	26
3.3.2 What happens when an assessment is created?	26
3.3.3 How do users navigate questions?	27
3.3.4 What happens when an assessment is completed?	29
3.3.5 Question map construction	32
3.3.6 ASSET System Configuration File	35
3.3.7 ASSET System reports	37
3.4 ASSET Manager	42
3.4.1 ASSET Manager Operation	42
3.4.2 Manager User Interface	43
3.4.3 ASSET Manager Configuration File	43
3.4.4 ASSET Manager Menu	46
3.4.5 Searching in ASSET Manager	47
3.4.6 ASSET Manager Reporting	48
3.5 Sorting in ASSET	49
3.6 ASSET Help	50
3.7 ASSET Core	52
3.8 ASSET Change Password Utility	53
3.9 Procedure Flow Diagram for each Component	55
4. ASSET Database	57
4.1 Database Design	57

4.2	Extensibility of Question Content	57
4.3	Assessment Rollups	58
4.4	Dbmanager	58
4.5	MSDE Data Backup.....	59
5.	Security of ASSET	60
5.1	Introduction	60
5.2	MSDE Network Vulnerability	60
5.3	MSDE Service Packs.....	61
5.4	MSDE Hotfixes.....	61
6.	ASSET Installer and Deployment.....	62
6.1	Windows NT Issues.....	62
6.2	Windows 9x Issues.....	62
6.3	Windows Command Prompt.....	63
	Appendix A— Source Code and Class Diagrams	1
A.1	ASSET System Source Code.....	1
A.2	Class Diagrams	1
	Appendix B— Database	1
B.1	ER Diagram.....	1
B.2	SQL Source Code	3
B.2.1	ASSET System SQL Table Statements	3
B.2.2	ASSET System SQL Stored Procedure Statements.....	17
B.2.3	ASSET Manager SQL Table Statements	56
B.2.4	ASSET Manager SQL Stored Procedure Statements.....	56
	Appendix C— Advanced Troubleshooting.....	1
C.1	MSDE Interoperability.....	1
C.2	Changing the Default Database Password.....	1
C.3	ACL/Security Templates	2
	Appendix D— Design Considerations.....	1
	Appendix E— Acronyms	1
	Appendix F— References.....	1

Tables

Table 1 - JAR file names in ASSET 1.0	5
Table 2 - Required third party JAR files for ASSET	7
Table 3 - System property file contents	36
Table 4 - Available System Reports	38
Table 5 - System Report Headers	41
Table 6 - Manager Property File Contents	45
Table 7 - Available Manager Reports	48
Table 8 - Windows NT Workstation 4.0 Requirements	62

Figures

Figure 1 - ASSET ODBC Authentication Type	4
Figure 2 - Incorrect JDK Compile Errors.....	5
Figure 3 - Create stand-alone archive Project option	6
Figure 4 - MainClass attribute of ADPU project.....	7
Figure 5 - Missing Xerces JAR files compile time error.	8
Figure 6 - Sample system project directory.....	9
Figure 7 – ASSET System question map shows multi-line questions	19
Figure 8 - ASSET System question map using incorrect PLAF.....	20
Figure 9 - Closing action for active assessment in System	23
Figure 10 - Closing action for active report window in System	23
Figure 11 - ASSET System User Interface.....	25
Figure 12 - ASSET System Main Layout.....	25
Figure 13 - ASSET System AssessmentWindow layout.....	26
Figure 14 – Example of System Question Tab	28
Figure 15 - Export Active Assessment Button.....	29
Figure 16 - Multiple Assessment Export Dialog	30
Figure 17 - Sample XML Assessment Export.....	32
Figure 18 - System question map cache file	33
Figure 19 - System property file	35
Figure 20 - Example System Property File.....	37
Figure 21 - System reporting interface.....	38
Figure 22 - Complete System Report.....	39
Figure 23 - System Report Screen Layout	40
Figure 24 - System Reporting Export Dialog.....	41
Figure 25 - Create New Assessment in Manager	43
Figure 26 – Manager property file	44
Figure 27 - Manager Menu Tree Example	46
Figure 28 - ASSET Manager Menu Cache File.....	46
Figure 29 - Manager Search Interface	47
Figure 30 - Choose Browser Wizard	51
Figure 31 - Cancel Choose Browser Wizard.....	52
Figure 32 - ADPU User Interface.....	54
Figure 33 - ADPU Advanced tab	55

Figure 34 - System Procedure Flow	56
Figure 35 - Manager Procedure Flow	56
Figure 36 - ASSET MSDE Communications Configuration	63
Figure A-1 - System UML Class Diagram	A-1
Figure A-2 - Manager UML Class Diagram.....	A-2
Figure B-1 - ER Diagram for System and Manager	B-2
Figure B-2 - System Database Scripts Install Location.....	B-3

1. Introduction

1.1 Purpose

This document is the technical complement to National Institute of Standards and Technology (NIST) Interagency Report (NISTIR) 6885, *Automated Security Self-Evaluation Evaluation Tool (ASSET) User Manual* (hereinafter referred to as the ASSET user manual). It is intended as a development guide for software engineers/database administrators who wish to troubleshoot unique installations of ASSET, reproduce the development version of ASSET, or extend the functionality of ASSET.

1.2 Audience

This report has been written for those who are skilled in software development and relational database design. Users of this report should have detailed knowledge of the Java programming language and its user interface applications programming interfaces (API), such as the Swing API and the Microsoft SQL Server Desktop Engine (MSDE) powered by the relational database programming language Transact-SQL (a derivative of the ANSI SQL92 standard). Readers are assumed to have a moderate to intermediate level of experience with Visual Café Expert Edition, the integrated development environment used to create ASSET.

1.3 Description of ASSET

ASSET was designed to automate the NIST Special Publication 800-26, *Security Self-Assessment Guide for Information Technology Systems*. The ASSET user manual contains a nontechnical description of ASSET. The description in this report focuses on the technical nature of ASSET. ASSET is comprised of three tools: ASSET System, ASSET Manager, and the ASSET Database Password Utility. ASSET System and ASSET Manager are designed to gather and aggregate the results of a self-assessment. The ASSET Database Password Utility is designed to modify the password for the systems administrator ('sa') user account of the MSDE.

1.4 References

This document makes extensive references to Internet resources and additional ASSET documentation. Readers of this document are assumed to have access to the content described in this section. The following publications and websites are referenced in this document:

- NIST Special Publication 800-26, *Security Self-Assessment Guide for Information Technology Systems*, August 2001
<http://csrc.nist.gov/publications/nistpubs/800-26/sp800-26.pdf>
- NIST Interagency Report 6885, *Automated Security Self-Evaluation Tool User Manual*, October 2002
<http://csrc.nist.gov/asset/ASSET-User-Manual.pdf>
- ASSET website (hosted on the NIST Computer Security Resource Center web server)
<http://csrc.nist.gov/asset/>
- Java 2 (version 1.3) API Documentation
<http://java.sun.com/j2se/1.3/docs/api/>

1.5 License

The following license applies to ASSET:

ASSET was developed at the National Institute of Standards and Technology by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code, this software is not subject to copyright protection and is in the public domain. ASSET is an experimental system. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. We would appreciate acknowledgement if the software is used.

ASSET is designed as an open source application, and as such is available for free download from the NIST ASSET website. This license, the source code packages for ASSET, and all documentation are available from the following ASSET website URL:

<http://csrc.nist.gov/asset/>

1.6 Conventions

The following conventions are used within this document. Refer to Appendix E for a complete listing of the acronyms used within this document. The words **system** and **manager** will be used to refer to ASSET System and Manager, respectively. When referring to both tools, or to the entire ASSET solution, the term ASSET will be used. When referring to the ASSET Database Password Utility, the abbreviation **ADPU** will be used. Any source code examples will be presented on numbered lines in 8-point Courier New font as in the following example.

```
1 System.out.println("This is an example");  
2 System.out.println("of java source code.");
```

Any references to Java class names or packages will be written with the full Java class name (including package) in bold font as follows:

- [**javax.swing.JPanel**](#) refers to the class JPanel in the package javax.swing
- Interface **ASSET** refers to the ASSET interface of **System** and **Manager**
- Where applicable, Internet references to the official Java API documentation will be provided in the form of hyperlinks on the class name.

Important notes and warnings will be written in bold-faced font and highlighted as in the following example:

NOTE: This is an important note that the reader is encouraged to remember.

2. Development Environment

This section describes the development environment for ASSET. A complete understanding of the development environment is necessary to reproduce or modify the ASSET tools. ASSET was developed using Web Gain's Visual Café Expert Edition version 4.5 under Windows 2000 Professional service pack 2. The corresponding help files for **system** and **manager** were developed using Macromedia Dreamweaver UltraDev. Although the use of Java offered the possibility of cross platform capabilities, one of the final decisions of the requirements gathering phase was to release the tool for the Windows NT kernel initially (this includes Windows NT workstation, Windows 2000 Professional, and Windows XP professional). Because of the method by which MSDE operates on Windows 95/98/ME, the initial release of ASSET only supports the Windows NT kernel. The settings described here assume an intermediate level of experience with Visual Café. For more information on Visual Café, consult the following URL:

http://www.webgain.com/products/visual_cafe/

2.1 Operating System Settings

No special operating system settings were necessary to configure the development machine. The initial development period was performed when the most current service pack for Windows 2000 was service pack 2. At the time of publication, the most current service pack was service pack 3. Compilation of ASSET has not been attempted under Windows 2000 service pack 3. The following URL lists the changes from Windows 2000 service pack 2 to service pack 3:

<http://support.microsoft.com/default.aspx?scid=kb;en-us;Q320853>

2.2 MSDE Settings

The Microsoft SQL Server Desktop Engine (MSDE) was chosen to provide the database management services for ASSET. Initial development took place using MSDE version 1.0 (installed with Visio 2000 SR-1). The 'sa' user account password was null until changed by the ASSET database password utility. The development machine was also tested under MSDE service pack 3 successfully. Due to the complex nature of the MSDE Package for the web, the version of MSDE that is distributed with Office 2000 and later will also suffice. The database name for system was **SAT_Client**, and the database name for manager was **SAT_Admin**.

Both databases were created with their associated database creation command line scripts named **dbmanager.bat**. These scripts are packaged in the SQL directory but if the path is changed to a command line variable, they can be run from anywhere on the hard drive. These scripts are discussed in detail later in the document. Although not recommended, due to its compatibility with SQLServer, ASSET may be used with an SQLServer installation if necessary. Users can explore this opportunity at their own risk.

<p>NOTE: ASSET has not been tested with MSDE 2000. The 2000 release of both MSDE and its parent product SQLServer included support for new technologies that may affect the method in which assessment data is stored.</p>

2.3 ODBC Settings

ASSET uses two ODBC file system data source names (DSNs). Manager uses a DSN named **MSDE_SAT_Admin_Data**, and system uses a DSN named **LocalServer**. Both DSNs have their default

database set (the ODBC “Change default database to:” option) to the corresponding module of ASSET. System’s DSN, LocalServer, has its default database set to system’s database, **SAT_Client**, and the default database for manager is SAT_Admin. Both DSNs are set to SQL Server authentication using the ‘sa’ account as shown in the following figure for SAT_Client.

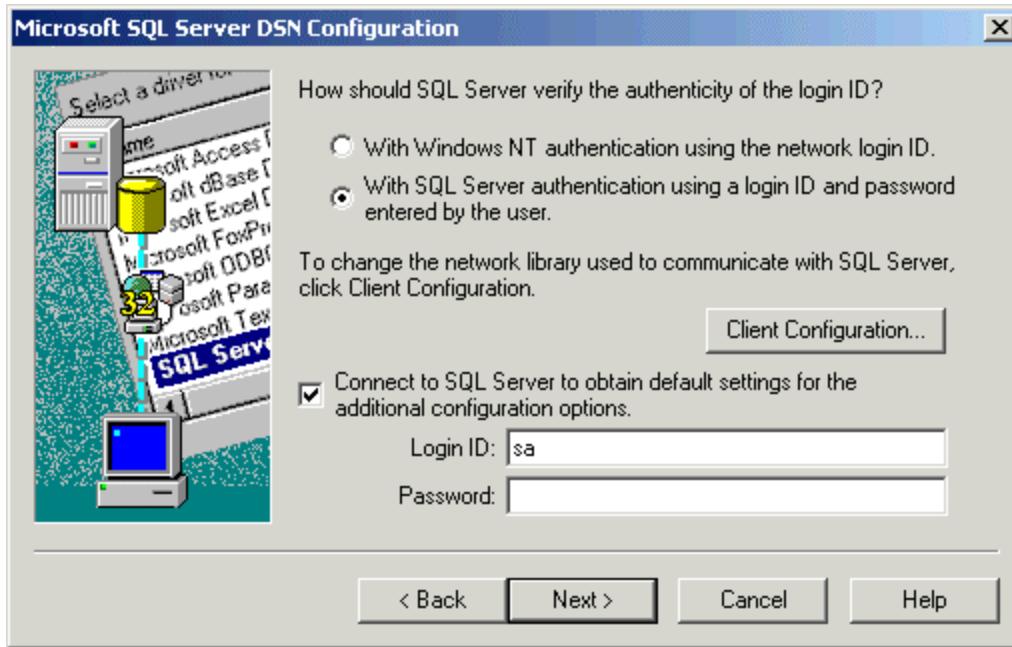


Figure 1 - ASSET ODBC Authentication Type

NOTE: If changed in system and manager configuration files, DSN names and MSDE authentication information are arbitrary. Refer to section “3.3.6 ASSET System Configuration File” for more information.

2.4 Visual Café Settings

This section describes the required configuration options for Visual Café to compile ASSET components. ASSET development occurred within an integrated development environment (IDE) because the interactive nature of the NIST self-assessment process dictates a complex user interface. ASSET was developed using Java 2 version 1.3.0_01. Since the default virtual machine (VM) distributed with Visual Café is Java 2 version 1.2.2, this required a new VM to be installed inside of Visual Café. Although development occurred on Visual Café Expert Edition version 4.5.1, Enterprise Edition 4.5.1 was also tested successfully. ASSET is comprised of four Visual Café projects with one project for system, manager, core files, and ADPU. This discussion will use these names to indicate the project name for the corresponding component of ASSET. The output of each project is discussed in the next section.

Note: Users **MUST** use the JDK version 1.3.0_01 or later as attempting to compile any component of ASSET without this JDK will result in compile errors. See the next figure for an example of compile errors with the incorrect JDK.

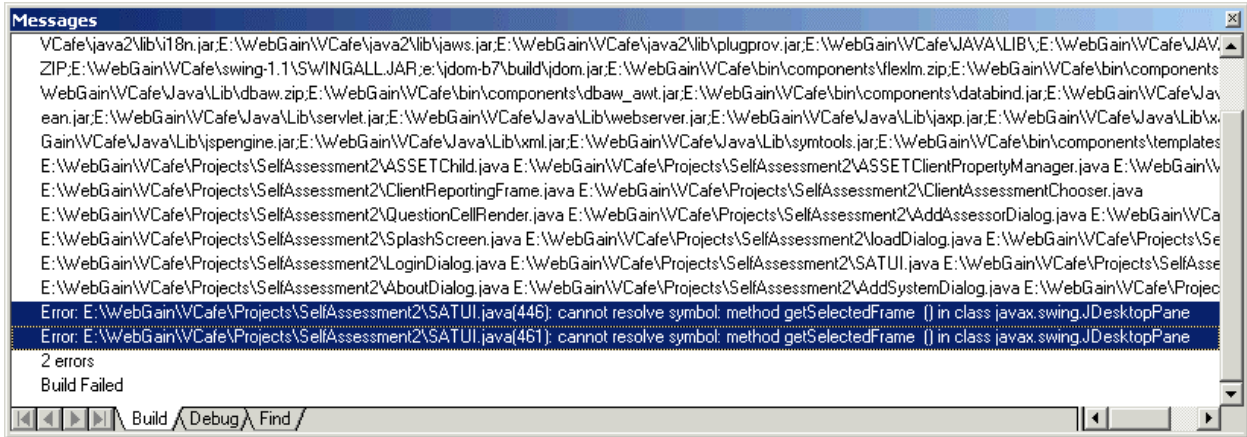


Figure 2 - Incorrect JDK Compile Errors

2.4.1 Project Output

This section describes the required configurations for the output files of each project. Visual Café compiles its projects into JAR files. The following table lists the JAR file and corresponding ASSET component included in version 1.0 of ASSET. Although these JAR file names are arbitrary, these names will be used here as examples.

Jar File Name	ASSET Component
asset.jar	ASSET System
asset_admin.jar	ASSET Manager
asset_core.jar	ASSET Core files
ASSETPassword.jar	ASSET Change Database Password Utility

Table 1 - JAR file names in ASSET 1.0

Each ASSET project uses third party component libraries such as the Swing, and Xerces XML APIs, and even makes use of each other. The required third party JAR files are discussed in the next section. Due to this, certain classes from these third party libraries will exist in the output JAR file for each project. Each JAR file must be configured as a stand-alone archive in the Project Options section or only those classes that are directly referenced with instantiation or import statements will be inserted into the JAR file (all of the parent and dependent classes won't be inserted). This configuration option is shown in Figure 3 below.

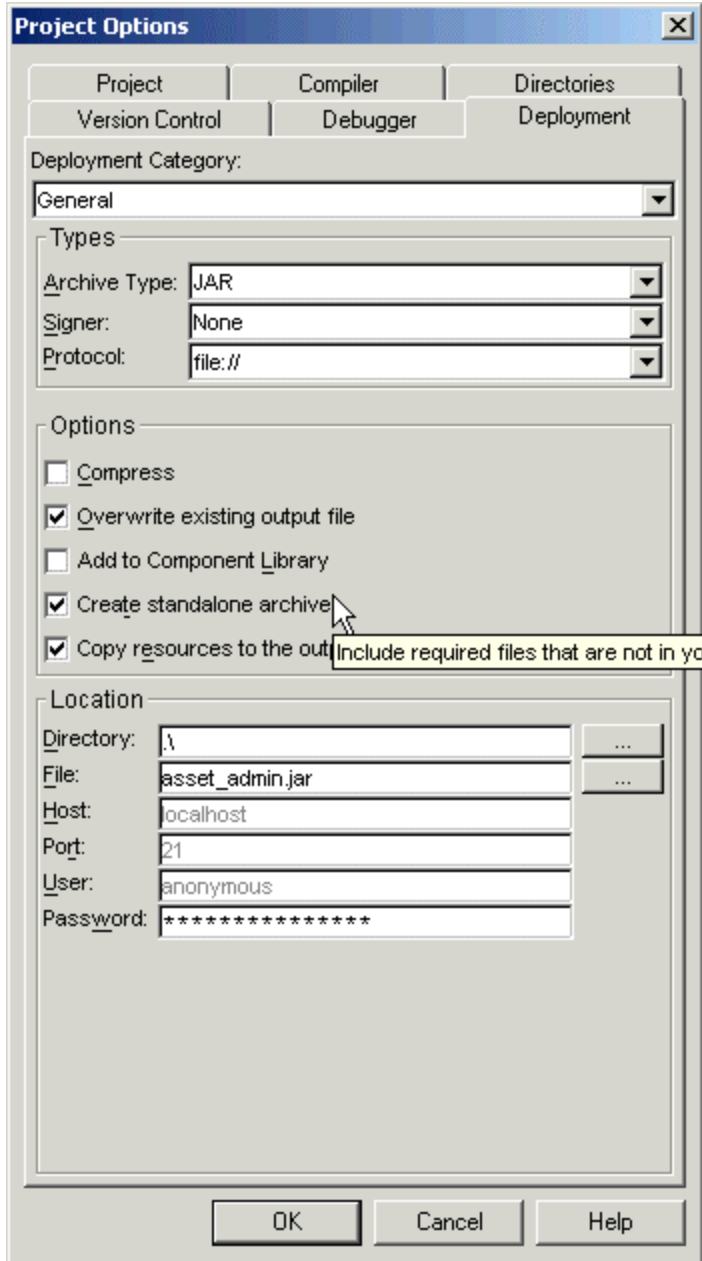


Figure 3 - Create stand-alone archive Project option

If this configuration option is not set, then ASSET users will receive a `java.lang.NoClassDefFoundError`. This means that a required class existed at compile time, but no longer exists at run time. This next configuration option applies only the ADPU project. Once the ADPU project JAR file is built, the Main-Class option must be set in the ADPU JAR manifest file `manifest.mf`. This configuration option determines which class is run when the JAR is double-clicked. In this case, the main class must be set to **ChangePassword**, the main class for the ADPU. To set this option, users can open the ADPU JAR file with WinZip 8.1 and manually edit the file `manifest.mf` to appear as shown in line two of the following Figure 4. Once edited, the file must be reinserted back into the JAR file.

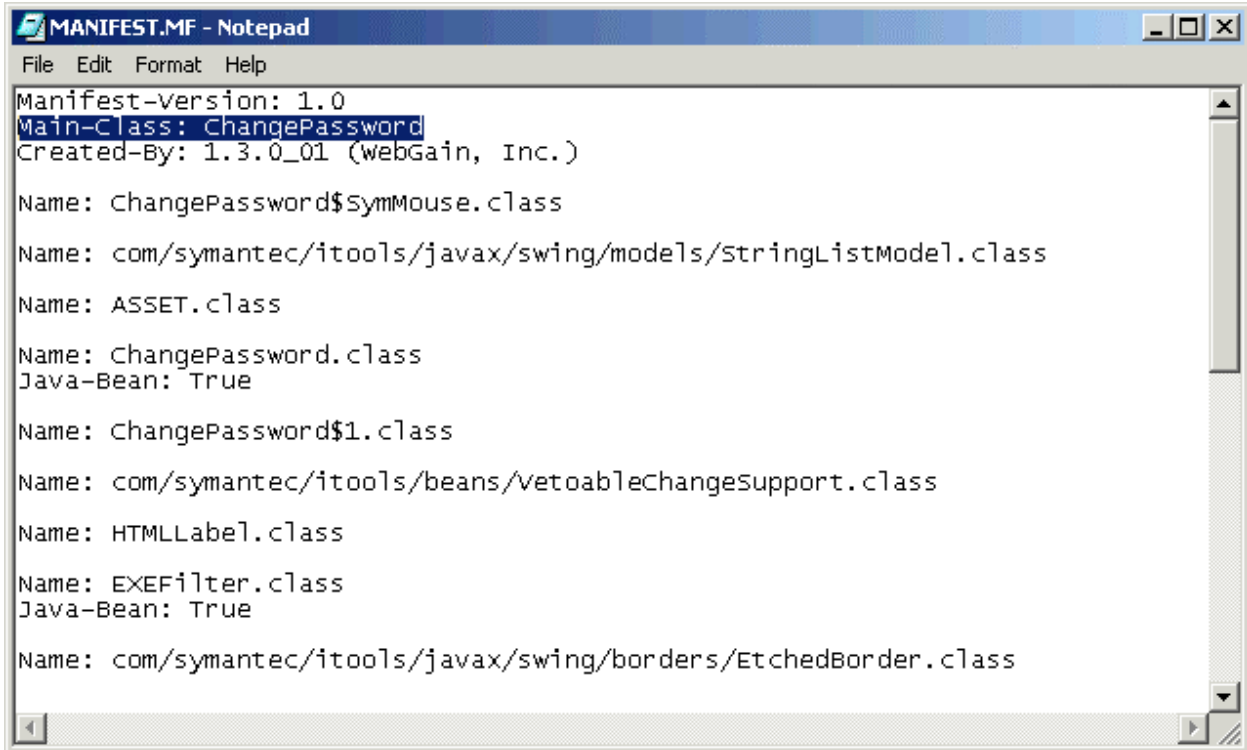


Figure 4 - MainClass attribute of ADPU project

2.4.2 Third Party JAR Files

This section describes the third party JAR file requirements necessary for ASSET to compile. All required JAR files listed below are available free of charge. The following table lists the JAR files necessary to compile system and manager.

JAR Name	URL	Belongs To
jdom.jar	http://www.jdom.org/	System, Manager, Core
xercesImpl.jar	http://gump.covalent.net/jars/latest/xml-xerces2/	System, Manager, Core
xmlParserAPIs.jar	http://gump.covalent.net/jars/latest/xml-xerces2/	System, Manager, Core
calendar.jar		Manager

Table 2 - Required third party JAR files for ASSET

JARs 2 and 3 comprise the Xerces XML parser from the Apache Software foundation. Users must download version 2.0.1 of Xerces or later. Versions prior to 2.0.1 had difficulty during the XML assessment export process in system and manager. It is very important to verify that previous versions of Xerces are **NOT** installed on the development machine and **NOT** inserted into the classpath.

NOTE: Previous versions of the Xerces XML parser are called xerces.jar. Users will receive unpredictable behavior if an incorrect version of Xerces is used.

These JARs must be manually inserted into the classpath of each project, system, manager, and core or they will not compile. This process is described in the next section. If the JAR files are not inserted into the project classpath, you may receive an error similar to the one shown in the Figure 5 below.

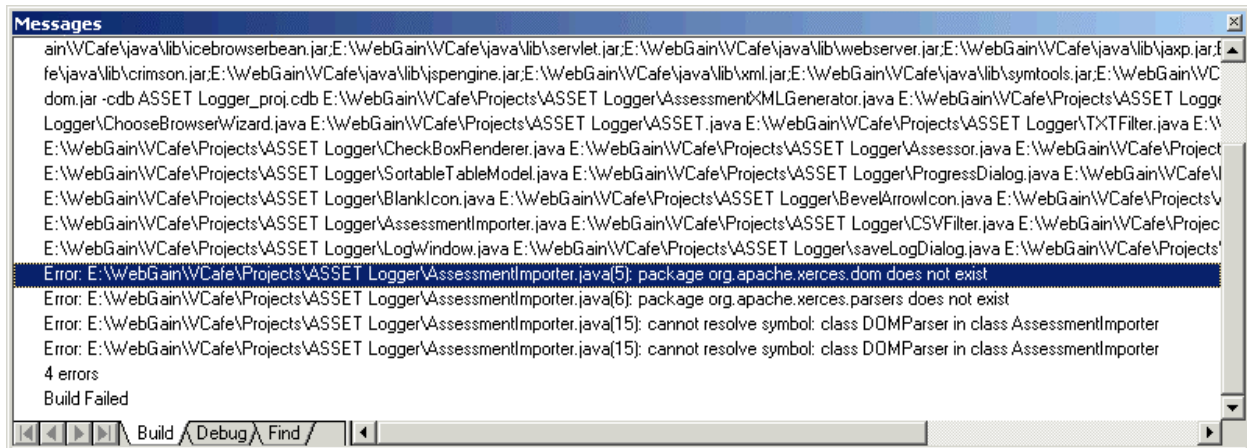


Figure 5 - Missing Xerces JAR files compile time error .

2.4.3 Classpath Settings

This section describes the classpath settings and required JAR files for each project, system, manager, and core. ADPU has no special classpath requirements. It is recognized that setting project specific classpaths is not the only method to allow the Java compiler to find required files. System needs the following JAR files in its project classpath:

- xmlParserAPIs.jar
- asset_core.jar
- xercesImpl.jar

Manager needs the following JAR files in its project classpath:

- asset.jar
- asset_core.jar
- xmlParserAPIs.jar
- xercesImpl.jar
- calendar.jar

Core needs the following files in its project classpath

- xmlParserAPIs.jar
- xercesImpl.jar

2.4.4 Folder Configuration

This section describes the necessary folder configurations for each project. The two main projects, system and manager, require certain sub-directories and files in their respective project directory in order to compile and execute successfully. The core and ADPU projects, on the other hand, have no specific folder requirements. System requires the following directories in order to execute:

- Help
- Images

- data
- sql

The Help directory contains the HTML help files for system, the images directory contains all of the images for the buttons within system and images for the help files, and the data directory is where backups of the current assessment are saved every three minutes. The following image shows a sample of what the system project directory must look like. It should be noted that manager has the same folder requirements as system.

NOTE: Not all of the files seen in the image are required. Particular attention should be paid to the directories mentioned above.

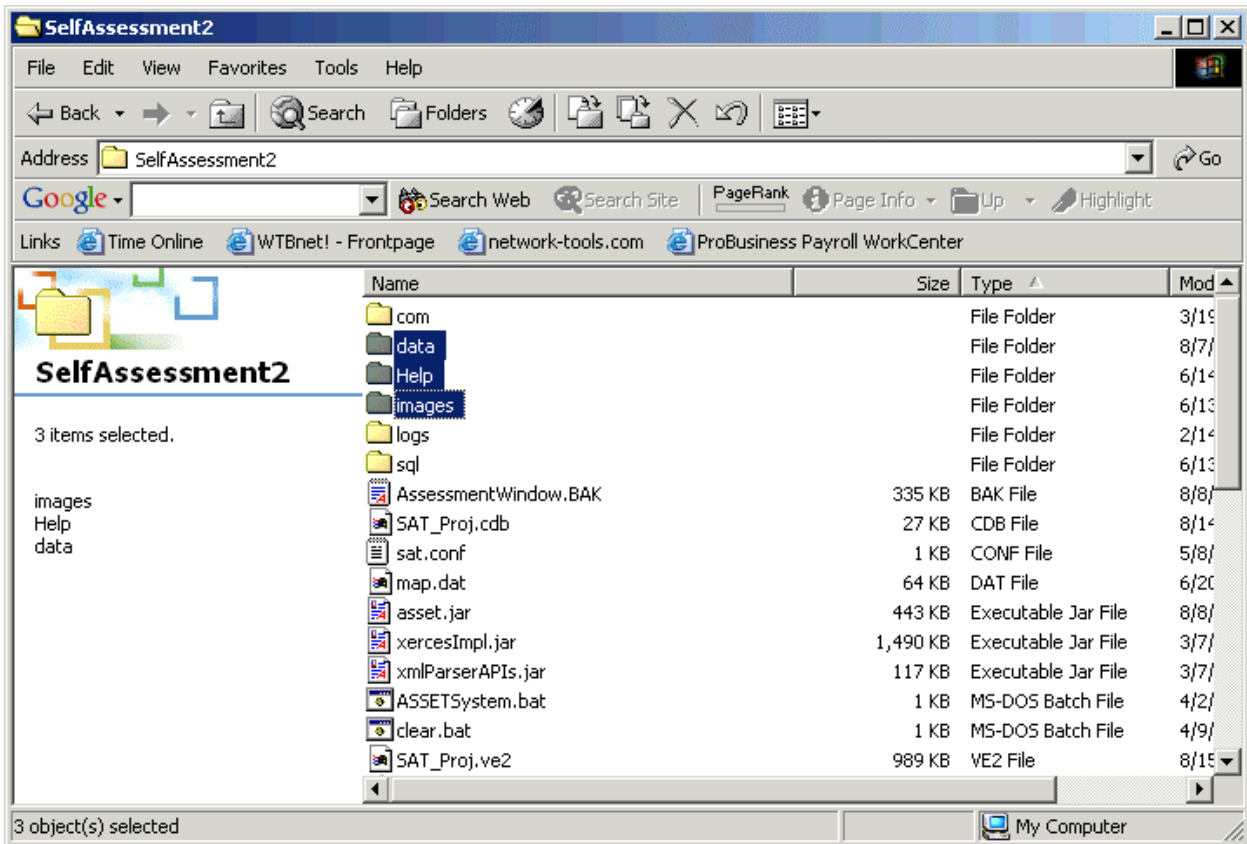


Figure 6 - Sample system project directory

While manager will have the same folder requirements, the contents of these folders differ. The specific file requirements of each folder are discussed in the next section. In addition to this, both manager and system require the sql folder. This folder contains project/database specific SQL scripts to set up and maintain the underlying database. The database specific files are discussed later in the document.

2.4.5 File Requirements

This section addresses the non-source code file requirements for each project. Without these necessary files, the projects may compile, but execution errors will almost certainly result. Both core and ADPU have no external file requirements other than their source code.

2.4.5.1 System File Requirements

This section addresses the non-source code file requirements of system. The data folder of System has no contents until the first time the program is executed. This is the directory where temporary active assessment backups are stored. To clear this directory, a batch script named **clear_asset_backups.bat** is included. The configuration file for System is named **sat.conf**; this is required in the root of the System project directory. This filename is hard coded and cannot change. As discussed in the folder requirements section, system requires the following folders

- Help
- Images
- data
- sql

The following HTML files compose the Help directory for System:

- GettingStarted.html
- HelpFilesForClient.html
- Index.html
- Performing.html
- Questions.html
- Terms.html

The following files compose the images directory for System:

- about.gif
- checkbox.gif
- completed.gif
- connection_error.gif
- copy.gif
- cut.gif
- default.gif
- delete.gif
- flagged.gif
- folder.gif
- help.gif
- mbsa_example.gif
- minimize_selfassessment.gif
- msde_busy_error.gif
- msde_delete_files_example.gif
- new.gif
- next.gif
- open.gif
- paste.gif
- plus.gif
- previous.gif
- print.gif
- props.gif

- refresh.gif
- report.gif
- save.gif
- SplashScreen.jpg
- sqllog_question.gif
- system_tray_connection_error.gif

The following files compose the sql directory for System:

- 1_assessments.sql
- 29_assessment_questions.sql
- 31_sat_users.sql
- 3_assessment_question_responses.sql
- 42_assessors.sql
- 6_publications.sql
- 7_pubReferences.sql
- 8_components.sql
- 91_conn_components.sql
- 9_component_criticality.sql
- dbmanager.bat
- sp_CurrentControlAreas.sql
- sp_CurrentQuestionMap.sql
- sp_CurrentQuestionSet.sql
- sp_DeleteSavedQuestion.sql
- sp_getAllComponentInformation.sql
- sp_getAssessorReviewersPerAssessment.sql
- sp_getAssessorsPerAssessment.sql
- sp_getComponentCriticalities.sql
- sp_getConnectedSystems.sql
- sp_getConnectedSystemsPerAssessment.sql
- sp_getCriticalListing.sql
- sp_getCriticalResults.sql
- sp_getNAQuestions.sql
- sp_getPolicyResults.sql
- sp_getPolicyResultsQuestionText.sql
- sp_getPrimaryAssessor.sql
- sp_GetQuestionCount.sql
- sp_getQuestionResults.sql
- sp_getQuestionResultsPerPolicy.sql
- sp_getQuestionResultsPerSection.sql
- sp_getRBDMQuestions.sql
- sp_GetSavedAssessments.sql
- sp_getSingleConnectedSystems.sql
- sp_GetSingleSavedAssessment.sql
- sp_GetSingleSavedAssessmentByAssessmentID.sql
- sp_GetStoredAssessors.sql
- sp_getSubmittedQuestionCount.sql
- sp_getSummaryQuestionResults.sql

- sp_InsertAssessmentQuestionResponse.sql
- sp_SaveAnsweredQuestion.sql
- sp_SaveAssessment.sql
- sp_SaveBookmark.sql
- sp_SaveComponent.sql
- sp_SaveComponentCriticalities.sql
- sp_SaveConnectedComponents.sql
- sp_SaveNewAssessor.sql
- sys_Component_Types.sql
- sys_controlareas.sql
- t_action_fillPublications.sql
- u_action_fillassess_ques.sql
- v_action_fillComponentTypes.sql
- w_action_fillControlAreas.sql
- x_action_fillPublicationReferences.sql

2.4.5.2 Manager File Requirements

This section addresses the non-source code file requirements of Manager. The data folder of System has no contents until the first time the program is executed. This is the directory where temporary active assessment backups are stored. To clear this directory, a batch script named **clear_asset_backups.bat** is included. The configuration file for Manager is named **admin.conf** and is required in the root of the Manager project directory. This filename is hard coded and cannot change. As discussed in the folder requirements section, Manager requires the following folders

- Help
- Images
- data
- sql

The following HTML files compose the Help directory for Manager:

- asset.html
- createassessment.html
- glossary.html
- index.html
- processassessments.html
- reporting.html
- userinterface.html

The following files compose the ASSET System sub-folder of the Help directory for Manager:

- GettingStarted.html
- HelpFilesForClient.html
- Index.html
- Performing.html
- Questions.html
- Terms.html

The following files compose the images directory for Manager:

- about.gif
- checkbox.gif
- checkmark.gif
- completed.gif
- connection_error.gif
- copy.gif
- cut.gif
- default.gif
- delete.gif
- examine.gif
- find16.gif
- flagged.gif
- folder.gif
- help.gif
- mbsa_example.gif
- minimize_selfassessment.gif
- msde_busy_error.gif
- msde_delete_files_example.gif
- new.gif
- next.gif
- open.gif
- paste.gif
- plus.gif
- previous.gif
- print.gif
- props.gif
- refresh.gif
- report.gif
- save.gif
- SplashScreen.jpg
- sqllog_question.gif
- system_tray_connection_error.gif
- userinterface.gif

The following files compose the sql directory for Manager:

- 1_assessments.sql
- 29_assessment_questions.sql
- 31_sat_users.sql
- 3_assessment_question_responses.sql
- 42_assessors.sql
- 6_publications.sql
- 7_pubReferences.sql
- 8_components.sql
- 91_conn_components.sql

- 9_component_criticality.sql
- dbmanager.bat
- sp_CountSensitivities.sql
- sp_CurrentControlAreas.sql
- sp_CurrentQuestionMap.sql
- sp_CurrentQuestionSet.sql
- sp_DeleteAssessmentInformation.sql
- sp_DeleteAssessorAccountability.sql
- sp_DeleteComponentCriticality.sql
- sp_DeleteComponentInformation.sql
- sp_DeleteConnectedComponents.sql
- sp_DeleteQuestionResponses.sql
- sp_DeleteSavedAssessment.sql
- sp_DeleteSavedQuestion.sql
- sp_getAllAssessmentInformation.sql
- sp_getAllComponentInformation.sql
- sp_getAllQuestions.sql
- sp_GetAssessmentCount.sql
- sp_getAssessorReport.sql
- sp_getAssessorReviewersPerAssessment.sql
- sp_getAssessorsPerAssessment.sql
- sp_getComponentCriticalities.sql
- sp_getComponents.sql
- sp_getConnectedSystems.sql
- sp_getConnectedSystemsPerAssessment.sql
- sp_getCriticalListing.sql
- sp_getCriticalResults.sql
- sp_getNAQuestions.sql
- sp_getPolicyResults.sql
- sp_getPolicyResultsQuestionText.sql
- sp_getPrimaryAssessor.sql
- sp_GetQuestionCount.sql
- sp_getQuestionResults.sql
- sp_getQuestionResultsPerPolicy.sql
- sp_getQuestionResultsPerSection.sql
- sp_getQuestions.sql
- sp_getRBDMQuestions.sql
- sp_GetSavedAssessments.sql
- sp_getSingleConnectedSystems.sql
- sp_GetSingleSavedAssessment.sql
- sp_GetSingleSavedAssessmentByAssessmentID.sql
- sp_GetStoredAssessors.sql
- sp_getSubmittedQuestionCount.sql
- sp_getSummaryQuestionResults.sql
- sp_getSummarySystemOrganizations.sql
- sp_getSummarySystemSensitivity.sql
- sp_getSystemsByType.sql
- sp_getSystemTypes.sql

- sp_InsertAssessmentQuestionResponse.sql
- sp_QuerySingleSavedAssessment.sql
- sp_ResetActiveAssessments.sql
- sp_SaveAnsweredQuestion.sql
- sp_SaveAssessment.sql
- sp_SaveBookmark.sql
- sp_SaveComponent.sql
- sp_SaveComponentCriticalities.sql
- sp_SaveConnectedComponents.sql
- sp_SaveNewAssessor.sql
- sp_SearchSavedAssessments.sql
- sp_SearchSavedAssessmentsPartial.sql
- sp_SetActiveAssessments.sql
- sp_SetActiveComponents.sql
- sys_Component_Types.sql
- sys_controlareas.sql
- t_action_fillPublications.sql
- u_action_fillassess_ques.sql
- v_action_fillComponentTypes.sql
- w_action_fillControlAreas.sql
- x_action_fillPublicationReferences.sql

2.5 Java Command Line Compilation

This section addresses users who wish to compile ASSET from the Java command line. Although possible to do, this process is complicated due to the number of required JAR files that are necessary to prevent the compiler from complaining. The process of implementing modifications to the user interface becomes harder when using the command line tools to compile ASSET since developers cannot see the approximate changes in the interface before the code executes as they can within an IDE like Visual Café. There are two possible methods of compiling ASSET on the command line, using the Symantec/WebGain compiler sj.exe (wjc.exe) and using the normal Sun Java compiler javac.exe.

2.5.1 Compile ASSET with Symantec/WebGain Compiler

This section discusses the necessary arguments to supply to the Symantec/WebGain compiler to compile ASSET. These sample commands require Visual Café installed in order to function. The command line examples here are taken directly from the ASSET Visual Café project settings (View>Messages>Build) for each respective ASSET component. The syntax to wjc takes the following form:

```
C:\>wjc -sysclasspath <Insert System classpath here> -O <for optimize class files> -d <Sets the working directory for the current project> -classpath <Includes the project specific classpath settings, and operating system classpath values> -make <Builds out of date files> -cdb <Specify compilation database> <List of files to compile>
```

The following example command line syntax will compile ASSET System (line breaks are indicated by the '→' character):

```
C:\>wjc                                     -sysclasspath                               →
E:\WebGain\VCafe\bin\..\JDK13\JRE\LIB\RT.JAR;E:\WebGain\VCafe\bin\..\JDK13\LIB\TOOLS.JAR  -O      -d
E:\WebGain\VCafe\Projects\SelfAssessment2\    -classpath →
```

```

E:\WebGain\VCafe\Projects\SelfAssessment2\xmlParserAPIs.jar;E:\WebGain\VCafe\Projects\ASSETLogger
\asset_core.jar;E:\WebGain\VCafe\Projects\SelfAssessment2\xercesImpl.jar;:\WebGain\VCafe\jdk13\jr
e\lib\rt.jar;E:\WebGain\VCafe\jdk13\lib\tools.jar;E:\WebGain\VCafe\java\lib\collections.zip;E:\We
bGain\VCafe\java\lib\jaxp.jar;E:\WebGain\VCafe\java\lib\xalan.jar;E:\WebGain\VCafe\java\lib\crims
on.jar;E:\WebGain\VCafe\java\lib\jspengine.jar;E:\WebGain\VCafe\java\lib\xml.jar;E:\WebGain\VCafe
\java\lib\symtools.jar;E:\WebGain\VCafe\bin\components\templates.jar;e:\jdom-b7\build\jdom.jar -
make -cdb SAT_Proj.cdb E:\WebGain\VCafe\Projects\SelfAssessment2\ASSETChild.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\ASSETClientPropertyManager.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\DATFilter.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\ClientReportingFrame.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\ClientAssessmentChooser.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\QuestionCellRender.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\AddAssessorDialog.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\QuestionResponse.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\SplashScreen.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\loadDialog.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\HybridTreeNode.java →
E:\WebGain\VCafe\Projects\SelfAssessment2>LoginDialog.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\SATUI.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\AssessmentWindow.java →
E:\WebGain\VCafe\Projects\SelfAssessment2>AboutDialog.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\AddSystemDialog.java →
E:\WebGain\VCafe\Projects\SelfAssessment2\SystemtoAssess.java →

```

The following example command line syntax will compile ASSET Core (line breaks are indicated by the ‘→’ character):

```

wjc -sysclasspath →
E:\WebGain\VCafe\bin\..\JDK13\JRE\LIB\RT.JAR;E:\WebGain\VCafe\bin\..\JDK13\LIB\TOOLS.JAR -O -d
E:\WebGain\VCafe\Projects\ASSET Logger\ -classpath →
E:\WebGain\VCafe\Projects\xmlParserAPIs.jar;E:\WebGain\VCafe\Projects\xercesImpl.jar;E:\WebGain\VC
afe\Projects\ASSETLogger\;E:\WebGain\VCafe\java\lib\jaxp.jar;E:\WebGain\VCafe\java\lib\xalan.jar
;E:\WebGain\VCafe\java\lib\crimson.jar;E:\WebGain\VCafe\java\lib\jspengine.jar;E:\WebGain\VCafe\j
ava\lib\xml.jar;E:\WebGain\VCafe\java\lib\symtools.jar;E:\WebGain\VCafe\bin\components\templates.
jar;e:\jdom-b7\build\jdom.jar -make -cdb ASSET Logger_proj.cdb → E:\WebGain\VCafe\Projects\ASSET
Logger\AssessmentXMLGenerator.java → E:\WebGain\VCafe\Projects\ASSET Logger\XMLFilter.java →
E:\WebGain\VCafe\Projects\ASSET Logger\ChooseBrowserWizard.java →
E:\WebGain\VCafe\Projects\ASSET Logger\ASSET.java → E:\WebGain\VCafe\Projects\ASSET
Logger\TXTFilter.java → E:\WebGain\VCafe\Projects\ASSET Logger\EXEFilter.java →
E:\WebGain\VCafe\Projects\ASSET Logger\CheckBoxRenderer.java → E:\WebGain\VCafe\Projects\ASSET
Logger\Assessor.java → E:\WebGain\VCafe\Projects\ASSET Logger\TableSorter.java →
E:\WebGain\VCafe\Projects\ASSET Logger\SortableTableModel.java → E:\WebGain\VCafe\Projects\ASSET
Logger\ProgressDialog.java → E:\WebGain\VCafe\Projects\ASSET Logger\SortButtonRenderer.java →
E:\WebGain\VCafe\Projects\ASSET Logger\BlankIcon.java → E:\WebGain\VCafe\Projects\ASSET
Logger\BevelArrowIcon.java → E:\WebGain\VCafe\Projects\ASSET Logger\ASSETPropertyManager.java →
E:\WebGain\VCafe\Projects\ASSET Logger\AssessmentImporter.java →
E:\WebGain\VCafe\Projects\ASSET Logger\CSVFilter.java → E:\WebGain\VCafe\Projects\ASSET
Logger\HTMLLabel.java →
E:\WebGain\VCafe\Projects\ASSET Logger\LogWindow.java → E:\WebGain\VCafe\Projects\ASSET
Logger\saveLogDialog.java → E:\WebGain\VCafe\Projects\ASSET Logger\OpenLogDialog.java →

```

The following example command line syntax will compile ASSET Manager (line breaks are indicated by the ‘→’ character):

```

wjc -sysclasspath →
E:\WebGain\VCafe\bin\..\JDK13\JRE\LIB\RT.JAR;E:\WebGain\VCafe\bin\..\JDK13\LIB\TOOLS.JAR -g -d
E:\WebGain\VCafe\Projects\ASSET Manager\ -classpath →
E:\WebGain\VCafe\Projects\SelfAssessment2\asset.jar;E:\WebGain\VCafe\Projects\ASSET
Logger\asset_core.jar;E:\WebGain\VCafe\Projects\ASSET
Manager\xmlParserAPIs.jar;E:\WebGain\VCafe\Projects\ASSET →
Manager\xercesImpl.jar;E:\WebGain\VCafe\Projects\ASSET Manager\calendar.jar;→
E:\WebGain\VCafe\java\lib\xml.jar;E:\WebGain\VCafe\java\lib\symtools.jar;E:\WebGain\VCafe\bin\com
ponents\templates.jar;e:\jdom-b7\build\jdom.jar -make
-cdb ASSETManager_proj.cdb E:\WebGain\VCafe\Projects\ASSET → Manager\AssessmentFrame.java
E:\WebGain\VCafe\Projects\ASSET → Manager\AddSystemTypeDialog.java
E:\WebGain\VCafe\Projects\ASSET →

```

```

Manager\QuestionFrame.java          E:\WebGain\VCafe\Projects\ASSET      →
Manager\ASSETManagerPropertyManager.java  E:\WebGain\VCafe\Projects\ASSET      →
Manager\ReportingFrame.java E:\WebGain\VCafe\Projects\ASSET →
Manager\AdminLoginDialog.java E:\WebGain\VCafe\Projects\ASSET → Manager\HybridTreeNode.java
E:\WebGain\VCafe\Projects\ASSET → Manager\EditQuestionDialog.java
E:\WebGain\VCafe\Projects\ASSET →
Manager\AdminSplashScreen.java E:\WebGain\VCafe\Projects\ASSET → Manager\AssessmentChooser.java
E:\WebGain\VCafe\Projects\ASSET → Manager\TextAreaRenderer.java E:\WebGain\VCafe\Projects\ASSET
→
Manager\AboutDialog.java E:\WebGain\VCafe\Projects\ASSET → Manager\SATAdmin.java
E:\WebGain\VCafe\Projects\ASSET → Manager\QuestionResponse.java E:\WebGain\VCafe\Projects\ASSET
→ Manager\DateChooser.java E:\WebGain\VCafe\Projects\ASSET → Manager\TextAreaEditor.java
E:\WebGain\VCafe\Projects\ASSET → Manager\MenuCellRenderer.java E:\WebGain\VCafe\Projects\ASSET
→ Manager\AssessmentInfoDialog.java

```

2.5.2 Compile ASSET with Sun Java Compiler

This section describes the use of the Sun Java compiler `javac.exe` to compile ASSET. As stated in section 2.4, ASSET requires version 1.3.0_01 or later of the JDK to compile successfully. The following syntax shows what the input to `javac.exe` looks like:

```
C:\>javac -cp <operating system and project specific classpath> <Java classes to compile>
```

3. ASSET Components

This section explains the components of ASSET, ASSET System, and ASSET Manager in greater detail. There are several key issues that influenced the development of ASSET, and a lack of understanding of these development issues can hinder future ASSET development. ASSET development issues include user interface (UI) needs such as screen layouts or common Java UI “quirks,” class design (object representation) and data flow, and finally, component specific issues.

3.1 User interface

This section describes the UI issues of ASSET. UI issues become important when considering the possibility of compiling ASSET for the Linux/Unix platform. ASSET System, ASSET Manager, and the ADPU use the Swing/Java Foundation Classes (JFC) version 1.1 API as distributed with Visual Café to provide the UI functionality. The UI design for System follows a tabbed format using the JTabbedPane component ([javax.swing.JTabbedPane](#)). The UI design for Manager models the Microsoft Management Console (MMC) due to its “advanced” nature. The user design for each ASSET component is discussed later in the document. Swing API specific issues are discussed in the next section.

Some unique graphical ‘quirks’ exist with ASSET and are a result of a common Java UI “quirk.” Java was designed as a multi-threaded environment but the threads responsible for executing program code are not the same threads responsible for painting screens and drawing pictures that the user can see. The priority for what actions get executed first in most cases goes to the execution threads, not the graphical ones. As with many Java programs, users may experience situations where text appears written over top of other text or form components drawn over top of others. If this occurs, it usually is a result of the graphical threads’ request for screen updates being cached to allow execution threads to execute first. Simply waiting a few seconds, or minimizing and then maximizing the screen, can solve this problem.

3.1.1 Swing 1.1

As stated in the previous section, ASSET uses the Swing API for its UI. The main reason for the use of Swing is the common appearance of the UI across versions of the Java Virtual Machine (JVM), Java Runtime Environment (JRE), and more importantly, across multiple operating systems. Specifically, of the three available “flavors” of Swing, ASSET uses the Java or “Metal” pluggable look and feel (PLAF). The full class name of the Metal PLAF is [javax.swing.plaf.metal.MetalLookAndFeel](#). This version of Swing aims for a common appearance across all JVMs with Swing support. ASSET was designed with maximum functionality in mind; assessments were meant to be only one service that ASSET provides. To allow future services to be added to ASSET to supplement self-assessments, all services such as open/save/report on assessments in System and search/delete/report on assessments in Manager were designed as layered components called JInternalFrames ([javax.swing.JInternalFrame](#)).

3.1.1.1 Metal PLAF and System Question Map

The use of the Metal PLAF can be tied to very specific components of ASSET System and Manager. Due to the length of some of the questions, on the ASSET System question map, it was desirable to have the question text span multiple lines to avoid forcing the user to use a horizontal scroll bar to read the entire question. The next figure shows an example of a multi-line question on the System question map.

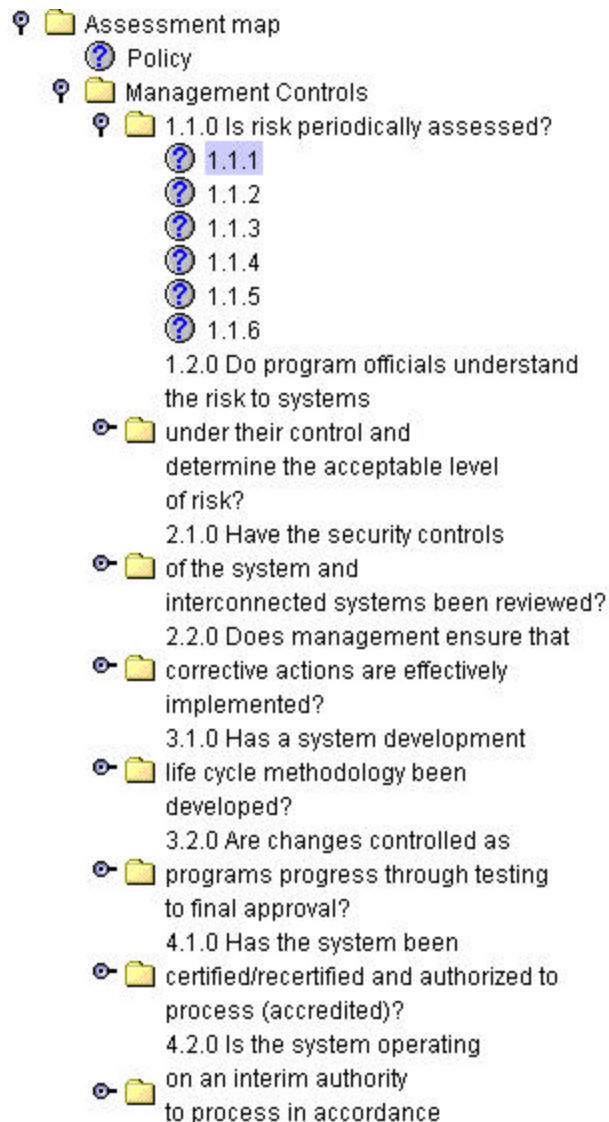


Figure 7 – ASSET System question map shows multi-line questions

Without the Metal PLAF, multi-line questions would have been impossible to implement. The image below shows the same question map using the Windows PLAF (`com.sun.java.swing.plaf.windows.WindowsLookAndFeel`). The following URLs provide additional resources for Swing PLAFs:

- <http://developer.java.sun.com/developer/techDocs/hi/jlf-home.html>
- <http://www.gargoylesoftware.com/papers/plafdiff.html>
- <http://java.sun.com/docs/books/tutorial/uiswing/misc/plaf.html>

In the figure below, note the original multi-line questions are now cut off, and only the first line of each question is visible. Also note that the highlight color has changed to a darker, less visible shade of blue.

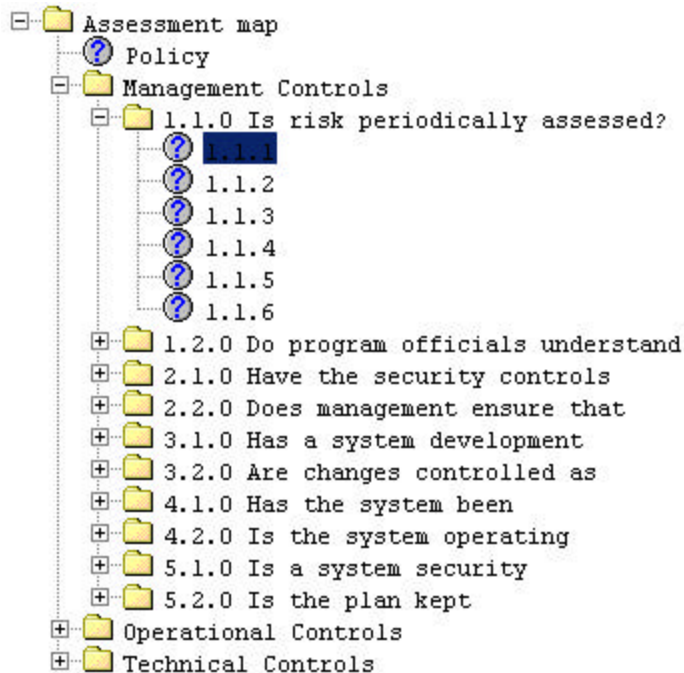


Figure 8 - ASSET System question map using incorrect PLAF

3.1.1.2 Keyboard Navigation

This section addresses keyboard navigation of ASSET as it relates to Section 508 compliancy. As stated in the requirements specification for ASSET, System and Manager will make every effort to conform to Section 508. The following URL is the homepage for Section 508 compliancy:

<http://www.section508.gov/>

Version 1.1 of the Swing API introduced a new method of navigating a Java user interface via the keyboard. Normal keyboard navigation uses the TAB key (to move forward) and SHIFT-TAB keys (to move backwards) to change the focus of the currently selected item on-screen. The ability to navigate an application without the need of a mouse is a requirement of Section 508 Accessibility standards. The Swing API requires the use of the CTRL-TAB key (to move forward) and CTRL-SHIFT-TAB (to move backwards) keys to change focus. The following excerpt of Java code (line 3) from ASSET System shows a “textbook” example of how to override the default key pressed for a text component to capture the TAB key being pressed.

```

1 private void obj_fieldTabKeyPressed(java.awt.event.KeyEvent event) {
2     try {
3         if (event.getKeyChar() == KeyEvent.VK_TAB) {
4             this.id_panel_proceed_button.requestFocus();
5         }
6     } catch (Exception e) {
7         e.printStackTrace();
8     }
9 } /* end method obj_fieldTabKeyPressed(event) */

```

When first implemented, this method resulted in no response from the system, but using the CTRL-TAB key combination instead to navigate forward results in the correct operation (the selection focus moves to the next component on-screen). Those components that do not already have an action handler for the TAB key will permit the above code, yet components such as the policy definition table ([javax.swing.JTable](#))

and the assessment objective field ([javax.swing.JTextPane](#)) of System already have actions defined for when the TAB key is pressed; therefore the CTRL-SHIFT-TAB key combination is required.

In addition to the ability to navigate throughout an entire active assessment using the keyboard, users need the ability to use the keyboard Enter key to trigger the actions of the various buttons within the assessment tabs. The JButton ([javax.swing.JButton](#)) does not capture a KeyPressed event by default, and so each button need a key listener class attached to it. The following source code snippet shows the key listening class from the Add Assessor Dialog of System:

```
1 class SymKey extends java.awt.event.KeyAdapter
2 {
3     public void keyTyped(java.awt.event.KeyEvent event)
4     {
5         Object object = event.getSource();
6         if (object == email_field)
7             emailField_keyTyped(event);
8         else if (object == add_assessor_button)
9             addAssessorButton_keyTyped(event);
10        else if (object == cancel_button)
11            cancelButton_keyTyped(event);
12        else if (object == existing_assessors)
13            existingAssessors_keyTyped(event);
14    }
15 }
```

Line 9 from the source code above shows the method that is executed if the OK button captures a KeyPressed event. The following source code snippet from the same class is the action handler method for line 9 from the previous example.

```
1 private void proceedKeyEnterPressed(java.awt.event.KeyEvent event) {
2     try {
3         if (event.getKeyChar() == KeyEvent.VK_ENTER) {
4             if (this.JTabbedPane.getSelectedIndex() == 0) {
5                 this.addAssessor();
6             }else {
7                 this.usePreExistingAssessor(null);
8                 this.addAssessor();
9             }
10        }
11    }catch (Exception e) {
12        e.printStackTrace();
13    }
14 }
```

This action handler uses the KeyEvent class as a parameter just as the TAB navigation action handler did from a previous example. Line 3 shows the if/then conditional that compares the key character with the appropriate static variable from the KeyEvent class. Every keyboard action event throughout both System and Manager are implemented in this same fashion.

The only other caveat regarding keyboard navigation pertains to ASSET System. When a user is answering questions for an active assessment and decides to change the currently selected tab to the Critical Element table, the quickest way to do so is with the mouse selecting the last tab at the top of the System window. Ordinarily the focus of System is on the question next/back/clear buttons when a user is answering assessment questions. In order for a user to navigate to the critical element tab, they would need to use the SHIFT-CTRL-TAB combination to cycle the selection focus backwards until they reached the tab strip which has the default focus of the question tab. In an effort to streamline this process, the Next button on System recognizes the capital 'S' character as the keyboard command to shift the focus to the Summary table from the Question tab.

3.1.2 MDI Interface

As stated in section 3.1, ASSET components are developed as `JInternalFrame` objects. The use of an `JInternalFrame` object enables a multiple document interface (MDI) design. The idea behind an MDI design was that any future services developed for System or Manager would be presented as `JInternalFrame` objects. Within the Swing architecture, `JInternalFrame` objects are hosted inside of `JDesktopPane` ([javafx.swing.JDesktopPane](#)) objects; otherwise the `JInternalFrame` objects would not be visible when added to a `JPanel` or `JFrame`. The following URL describes the operation of adding a `JInternalFrame` object to a `JDesktopPane`:

<http://java.sun.com/docs/books/tutorial/uiswing/components/internalframe.html>

This article also includes several best practice rules for using internal frames. The ASSET System `AssessmentWindow` class inherits from the `JInternalFrame` object. When a new assessment is created, an instance of `AssessmentWindow` is created and added to the `JDesktopPane` object. The use of an MDI interface influences several areas of ASSET including the display of system and error messages, and custom operations when the internal windows are closed. Both of these references deal with the interface design of ASSET, and this is discussed further later in the document.

3.1.2.1 System and Error Messages in an MDI Interface

The Swing API introduced a new method of displaying messages graphically called a `JOptionPane` ([javafx.swing.JOptionPane](#)). Previous methods of displaying messages or “alert boxes” to users within Java included the `Dialog` ([java.awt.Dialog](#)) class within the Abstract Window Toolkit (AWT) of the JDK version 1.1.x, and the `JDialog` ([javafx.swing.JDialog](#)) component within Swing API. These methods are all considered “heavyweight” when compared to the `JOptionPane` class. Because the `JDesktopPane` object is a layered component, it was necessary to tie any system messages displayed to the `JDesktopPane` parent object or run the risk of having a dialog box be tied to a layer of parent that wouldn’t necessarily be visible. To accomplish this, the ASSET interface ASSET defined two methods called `displayErrorDialog(String)` and `displayStatusDialog(String)`. These methods represented instances of the `JOptionPane` class tied to the parent class (System, Manager). The error method code is shown in the following example:

```
1 public void displayErrorDialog(String msg) {
2     try {
3         JOptionPane.showMessageDialog(this, msg, "Error:", → JOptionPane.ERROR_MESSAGE);
4     } catch (Exception e) {
5         e.printStackTrace();
6     }
7 }
```

3.1.2.2 Custom Closing Actions for Internal Windows

This section addresses the actions that occur when internal windows are closed in System and Manager. Within System, there are two types of `JInternalFrames` that can be open, an active assessment (`AssessmentWindow`) and the reporting window (`ClientReportingFrame`). When a user clicks the close icon of an active assessment, they could lose active question response data if this action is not properly caught and processed. With this scenario in mind, each `JInternalFrame` object implements the `ASSETChild` interface. This interface is discussed in detail in the next section. When a user tries to close an active assessment in System, they are met with the following dialog:

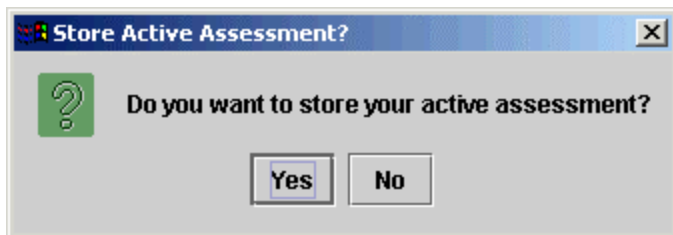


Figure 9 - Closing action for active assessment in System

When a user attempts to close the parent ASSET System SATUI class with “x” button, the same prompt occurs. What happens is the parent class SATUI calls the method `getAllFrames()` of `JDesktopPane`. This allows SATUI to deal with each open frame individually showing this prompt. The following code excerpt from the SATUI method `exitApplication()` handles this specific operation:

```

1 Toolkit.getDefaultToolkit().beep();
2 opened_frames = this.JDesktopPanel.getAllFrames();
3 for (int i = 0; i < opened_frames.length; i++) {
4     this.JDesktopPanel.setPosition(opened_frames[i], 0);
5     child = (ASSETChild) opened_frames[i];
6     {
7         if (child != null) {
8             child.parentApplicationClosing();
9         }
10    }
11 }
12 System.exit(0);

```

Line 8 shows the actual method of the `ASSETChild` interface that gets called, `parentApplicationClosing()`, as this is the method that displays the dialog shown in Figure 9 - Closing action for active assessment in System. When a user tries to close the reporting window only after they have run a report, they are met with the following dialog:

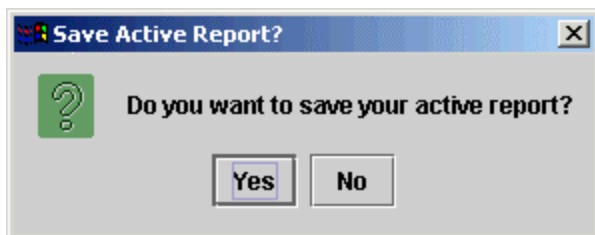


Figure 10 - Closing action for active report window in System

The same action handling code from the active assessment example Figure 10 - Closing action for active report window in System, applies for the reporting window. The next section addresses the class and interface design of ASSET.

3.2 Class design

This section addresses the class design of ASSET components with focus on the Java interfaces that the components may implement as mentioned in previous sections. ASSET makes use of the following Java interfaces:

- ASSET
- ASSETChild

These interfaces are packaged/distributed within Core and System components of ASSET to promote object reuse. It is the components of ASSET that implement these interfaces that are the focus of this section. The ASSET interface defines the following methods:

```
1 public interface ASSET {
2     public void log(String buffer);
3     public void setServiceName(String name);
4     public void saveLogOnException();
5     public void setStatusModeMsg(String msg);
6     public void setStatusMsg(String msg);
7     public void displayErrorDialog(String msg);
8     public void displayStatusDialog(String msg);
9     public Connection connectionGenerator();
10    public void openDefaultHelpFile();
11    public String getAbsolutePath(String local_path);
12    public void defineAction(String action);
13    public void defineTerm(String term);
14    public void setActiveTextComponent(JTextComponent component);
15    public void startNewAssessment();
16    public ASSETPropertyManager getPropertyManager();
17    public void openOldAssessment(String id, String name);
18    public void openReportingFrame();
19 }
```

The System SATUI class and Manager SATAdmin classes implement the ASSET interface. To demonstrate the power of the ASSET interface, the method `setActiveTextComponent(JTextComponent)` is designed to inform SATUI when the currently selected text component changes. This allows the Edit menu to provide Windows clipboard Cut/Copy/Paste capabilities. The ASSETChild interface defines the following methods:

```
1 public interface ASSETChild {
2     public boolean getSupportsSave();
3     public void store();
4     public void saveLocally();
5     public void parentApplicationClosing();
6 }
```

The AssessmentFrame and ReportingFrame classes of Manager and the ClientReportingFrame and AssessmentWindow classes of System implement the ASSETChild interface.

3.3 ASSET System

This section describes the operation and design of ASSET System. Users should note that the ASSET user manual provides a detailed description on the operation and installation of System. The intended purpose of System was the major influence behind its design. The following figure shows a new assessment created in System.

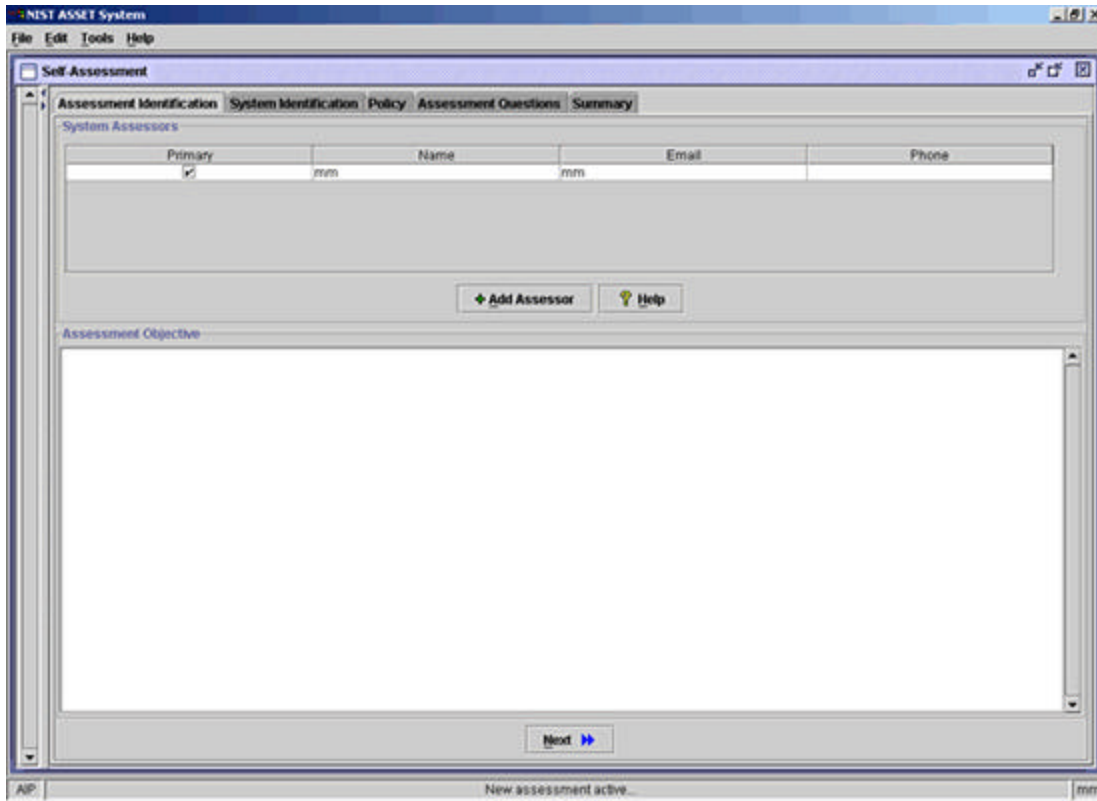


Figure 11 - ASSET System User Interface

As described earlier, when a new assessment is created, a new instance of the AssessmentWindow component is created and added to the JDesktopPane of SATUI with the add () command. From this figure, it is evident that ASSET System uses the JTabbedPane component ([javafx.swing.JTabbedPane](http://java.sun.com/javafx/2/javafx-2-2-ga/javafx-2-2-ga-javadoc/javafx.swing/javafx.swing.JTabbedPane.html)) to provide the five tabs shown: Assessment Identification, System Identification, Policy, Assessment Questions, and Summary. The parent class SATUI uses a BorderLayout for a layout manager as shown in the following figure:

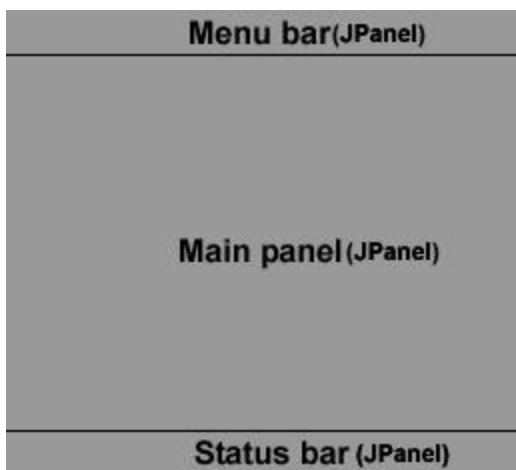


Figure 12 - ASSET System Main Layout

The menu bar is positioned in the north, the JDesktopPane and JInternalFrame(s) are positioned in the Center, and the status bar is positioned in the south. The status bar is implemented as a single component

with a border layout of its own. The leftmost panel is designed to show the current mode operation, the center panel is designed to display general messages similar to the status bar of common web browsers, and the rightmost component displays the user that is currently logged in. The JTabbedPane component must be added to a JScrollPane ([javax.swing.JScrollPane](#)) in order to be visible. The following figure describes the AssessmentWindow layout:

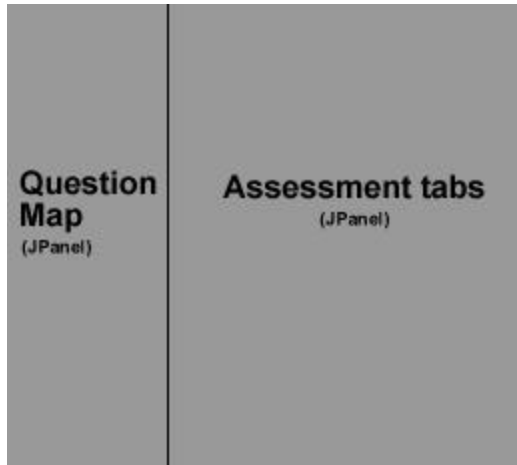


Figure 13 - ASSET System AssessmentWindow layout

The AssessmentWindow class uses a BorderLayout. The two panels shown in the figure above are displayed inside of a JSplitPaneWrapper (com.symantec.itools.java.swing.**JSplitPaneWrapper**), where the question map is the left panel and the JTabbedPane is the right panel. The construction and management of the question map is discussed later in this section. Each JPanel of the JSplitPaneWrapper is displayed inside of a JScrollPane component to allow the use of scroll bars to view the entire pane. Java refers to this as a “virtual desktop.”

3.3.1 ASSET System Operation

This section describes the operation of System. This information summarizes the content of the ASSET user manual and the ASSET requirements specification in that the operation of System need only be understood from a user needs perspective. Once developers understand what is expected from System, they will understand its operation better, including how to improve upon it.

System is designed to gather the individual responses to the NIST Self-Assessment checklist. These responses are then imported into Manager. Because System tracks responses to questions, it should be considered as a finite state machine. A response to a checklist question in System has 16 parts and, therefore, a unique state. Each question should be available to be changed an infinite number of times at the user’s discretion. There are two parts to assessment completion, assessment identification and question answering. Assessment identification itself has two subsections, assessor/purpose identification and componentidentification and classification. During question answering, users have the ability to view the results of the current critical element. Users have the ability to bookmark their current progress by saving assessment responses to the database or local file.

3.3.2 What happens when an assessment is created?

The process of creating a new assessment is initiated by the System parent class SATUI. Within SATUI, the following actions are performed when a new assessment is created:

1. Instantiate new instance of AssessmentWindow class (execute AssessmentWindow constructor).
2. Set primary assessor or new assessment.
3. Add new assessment to JDesktopPane and set visible.



Within the AssessmentWindow constructor, the following steps occur:

1. Initialize graphical components within AssessmentWindow.
2. Initialize action handlers for graphical components.
3. Set column headers for assessor and connected system table.
4. Initialize file chooser dialogs for exporting assessment data.

3.3.3 How do users navigate questions?

This section addresses how users navigate the questions within an assessment. The System Question Map was introduced in section 3.1.1.1. Within the SAT_Client database, questions are stored in a separate database table. These questions are extracted from the database by the stored procedure **current_question_map**. Each question in the self-assessment process is numbered in a three-part fashion. As an example, the last question in the assessment checklist is 17 . 1 . 9. The first part of this number **17** is the major section the user is in. There are 17 possible major sections in the checklist. The second number **1** is the minor section. There are 36 possible minor sections in the checklist. This number also indicates the critical element being computed at the time. The last number **9** is the question number. Due to the number of required questions within the checklist, it was not feasible to allow System to visit the database more than a limited number of times. System solves this method by holding all of the question information within the nodes of the question map. There are a total of 278 questions within the question table:

- 17 questions from the checklist are policy-related. These take the form of 1.0.0, 2.0.0, 3.0.0, etc.
- 36 of the questions are critical elements. These questions take the form of 1.1.0, 1.2.0, 2.1.0, etc.
- The other 225 are subordinate questions that the user is required to complete. These questions take the form of 1.1.1, 1.1.2, 1.1.3, etc.

The initial attempt of constructing the question map was to use basic SQL SELECT statements, selecting the control areas first followed by the major section, then the critical element sections, and finally the subordinate questions. This took the appearance of a three-level loop evaluated asymptotically at $O(n^3)$. The SQL syntax to create the question map is discussed in detail later in the document. Figure 7 – ASSET System question map shows multi-line questions,” shows the appearance of the questions within System. The following figure shows an example of the question tab of System. Those questions in the map below that a user is permitted to answer are indicated with an  icon. Those questions indicated by the  icon are critical element questions that are computed by the system.

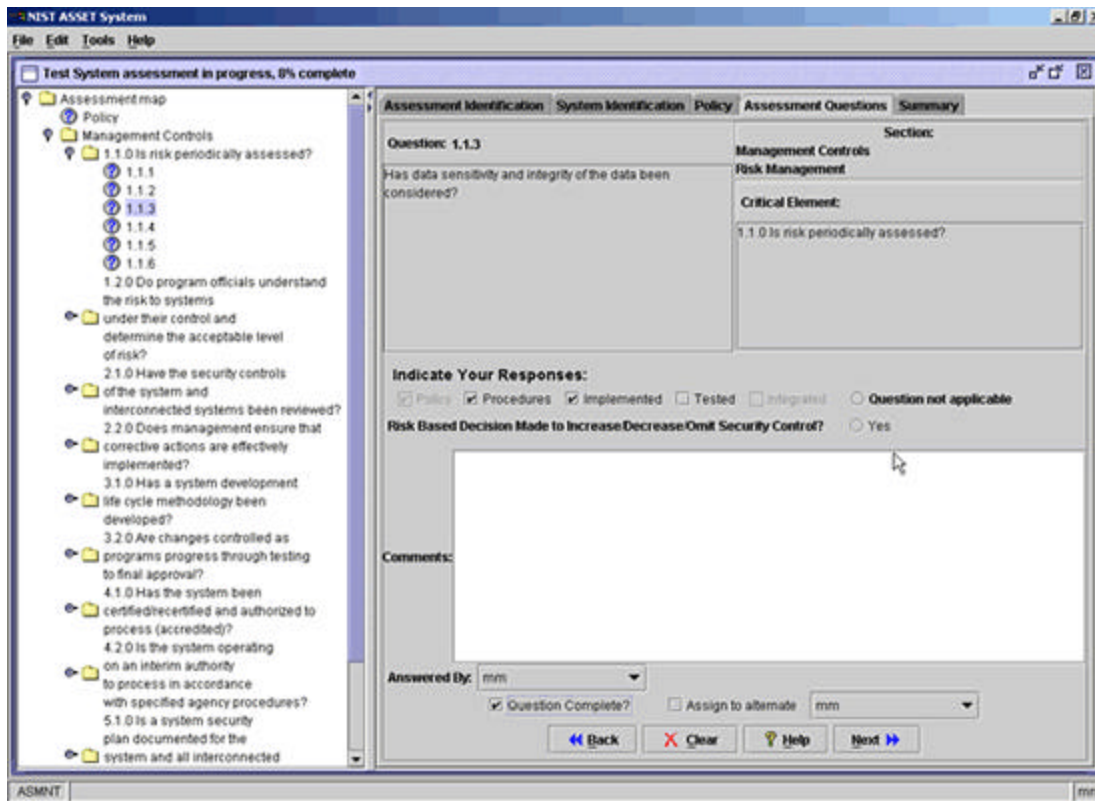


Figure 14 – Example of System Question Tab

The component on the left side of the JSplitPaneWrapper is referred to as the question map in System. This is a JTree component (javax.swing.JTree). The right side of the JSplitPaneWrapper is the question display panel. Users are permitted to navigate the questions within an assessment in one of three ways:

- Backwards
- Forwards
- Randomly

The Back and Forward buttons permit backwards and forwards navigation, and the question tab listens for selection events of various questions. To allow users to click the map with their mouse, System implements the Swing TreeSelectionListener interface. This method requires the `valueChanged(TreeSelectionEvent)` method. When a user clicks on one of the available questions, this method displays the corresponding question in the display panel.

NOTE: Within the class definition for AssessmentWindow, all possible question navigation methods, backwards, forwards, and random (using question map) actually call the same method in the end, `displayQuestion(String)`. This was done to promote a uniform display process.

The questions are designed for the user to navigate forwards using the Next button until they reach the final question, 17.1.9. When they reach this final question, their progress is automatically saved within the database. The other two methods of navigation are provided for the user's benefit should they wish to change an answer.

3.3.4 What happens when an assessment is completed?

When a user answers all of the questions within System, their assessment is considered completed. The integer variable `ques_progress` tracks the number of completed questions as a percentage completed. Once this percentage reaches 100, the assessment is automatically saved to the database and the user has several options:

1. Open/Edit new assessment
2. Generate reports
3. Export assessment information to submit to another System installation or to Manager

It is the export process that is the focus of this section. The user has two options for exporting assessment results from System. Their first option is to export an active assessment as shown in the following figure.

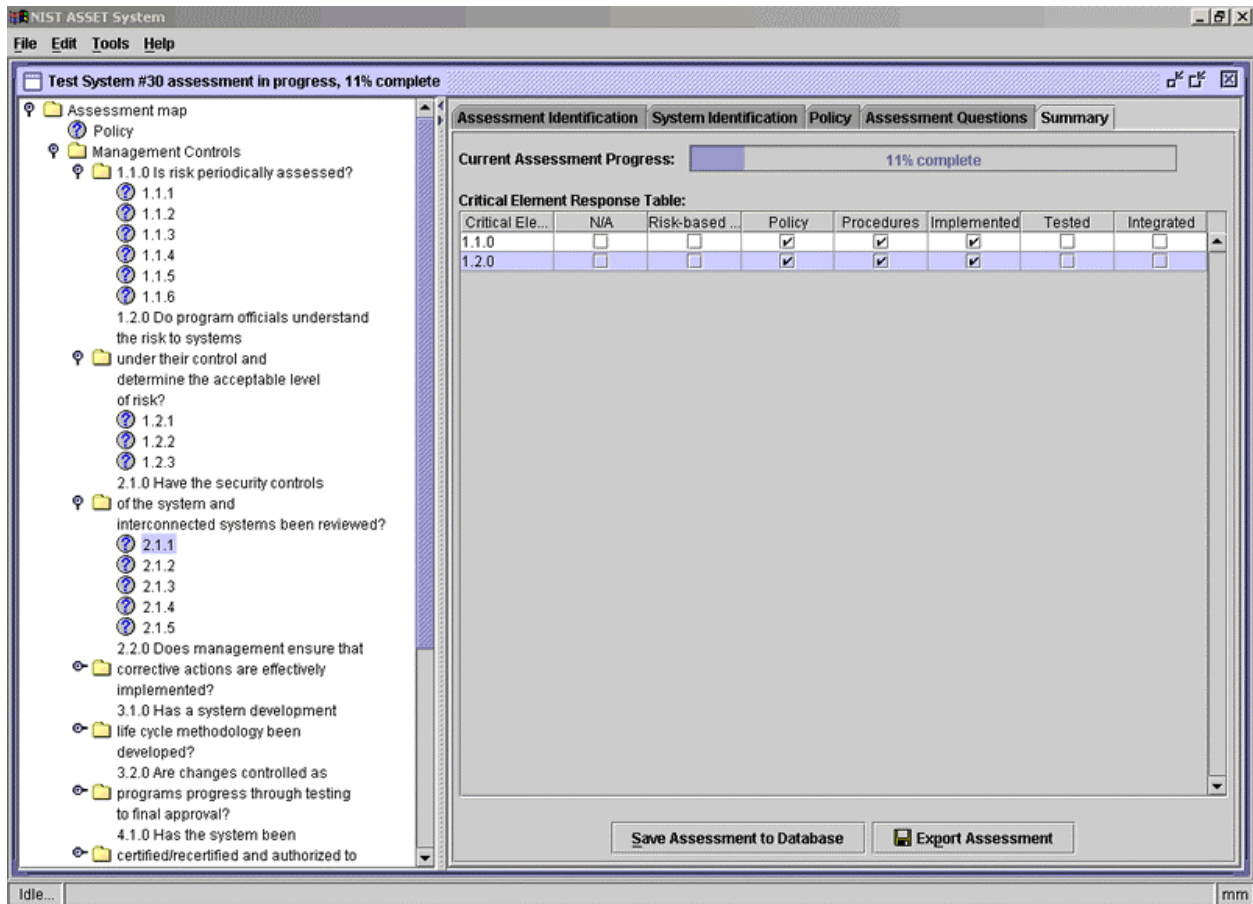


Figure 15 - Export Active Assessment Button

The export button for an active assessment is located at the bottom of the summary tab. When a user presses this button, they invoke the `AssessmentXMLGenerator` class to export the active assessment results to XML. This class extends the `Thread` class ([java.lang.Thread](#)) to allow it to run in the background. The `AssessmentXMLGenerator` calls the database independently from `System` or `Manager`. To properly initialize the `AssessmentXMLGenerator` class, it must be passed a vector of assessment identifiers to export in addition to an instance of the `ASSETPropertyManager` class.

The second option for exporting assessments is to use the Load Assessment dialog accessible from the Start menu. This option is preferred for users who wish to send multiple assessments to Manager. This dialog box is shown in the next figure.

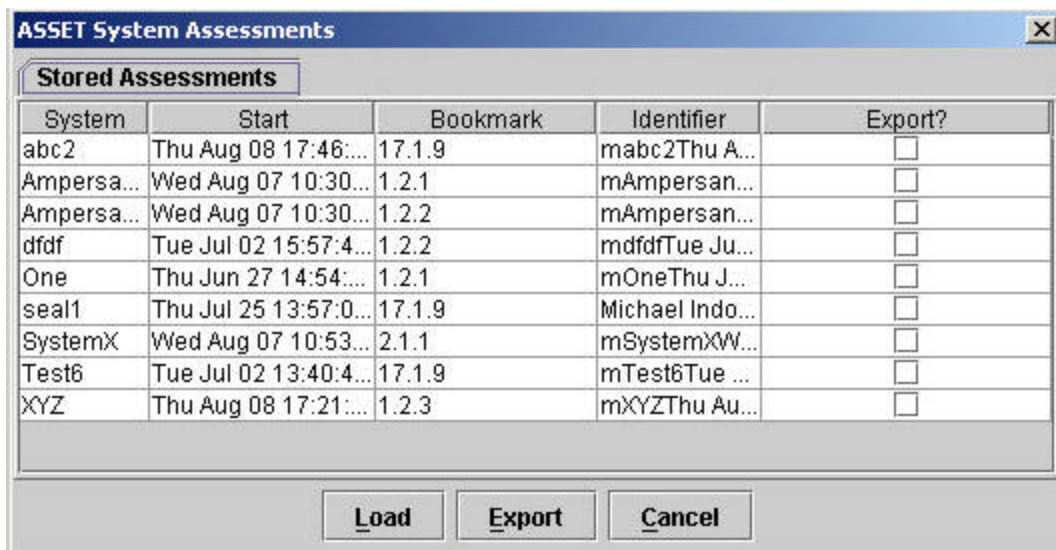


Figure 16 - Multiple Assessment Export Dialog

Users will note in the previous figure that the fifth column “Export” is a checkbox to indicate which assessment to export. This design was implemented using the custom cell rendering process for JTables. This process is outlined further at the following URL:

<http://javaalmanac.com/egs/javax.swing.table/CustRend.html>

Completed assessment export information is stored in a well-formed XML file format. The document type definition (DTD) for an assessment export file is as follows:

```
<!ELEMENT answeredby ( #PCDATA ) >
<!ELEMENT ar ( #PCDATA ) >
<!ELEMENT assessment ( location, system, assessmentname, startdate, completed, bookmark,
primary_assessor, progress, assessors, criticalities, connectedsystems, policy, critical,
questions ) >
<!ELEMENT assessment_obj ( #PCDATA ) >
<!ELEMENT assessmentname ( #PCDATA ) >
<!ELEMENT assessments ( assessment+ ) >
<!ELEMENT assessor ( fullname, email, phone ) >
<!ELEMENT assessors ( assessor ) >
<!ELEMENT availability ( #PCDATA ) >
<!ELEMENT bce ( #PCDATA ) >
<!ELEMENT bookmark ( #PCDATA ) >
<!ELEMENT cmnts EMPTY >
<!ELEMENT completed ( #PCDATA ) >
<!ELEMENT component ( name, bce, pain ) >
<!ELEMENT confidentiality ( #PCDATA ) >
<!ELEMENT connectedsystems ( component ) >
<!ELEMENT critical ( element+ ) >
<!ELEMENT criticalities ( confidentiality, integrity, availability ) >
<!ELEMENT element ( number, policy, procedures, implemented, tested, integrated, rbdm, na ) >
<!ELEMENT email ( #PCDATA ) >
<!ELEMENT fullname ( #PCDATA ) >
<!ELEMENT implemented ( #PCDATA ) >
<!ELEMENT integrated ( #PCDATA ) >
<!ELEMENT integrity ( #PCDATA ) >
<!ELEMENT location ( #PCDATA ) >
```



```

<!ELEMENT na ( #PCDATA ) >
<!ELEMENT name ( #PCDATA ) >
<!ELEMENT num ( #PCDATA ) >
<!ELEMENT number ( #PCDATA ) >
<!ELEMENT objective ( number, state ) >
<!ELEMENT pain ( #PCDATA ) >
<!ELEMENT phone EMPTY >
<!ELEMENT policy ( #PCDATA | objective )* >
<!ELEMENT primary_assessor ( #PCDATA ) >
<!ELEMENT procedures ( #PCDATA ) >
<!ELEMENT progress ( #PCDATA ) >
<!ELEMENT question ( number, policy, procedures, implemented, tested, integrated, rbdm, cmnts,
ar, na, reviewer, completed, answeredby ) >
<!ELEMENT questions ( question+ ) >
<!ELEMENT rbdm ( #PCDATA ) >
<!ELEMENT reviewer ( #PCDATA ) >
<!ELEMENT startdate ( #PCDATA ) >
<!ELEMENT state ( #PCDATA ) >
<!ELEMENT system ( name, num, location, type, assessment_obj ) >
<!ELEMENT tested ( #PCDATA ) >
<!ELEMENT type ( #PCDATA ) >

```

Figure 17 shows an example of an assessment export file opened within Internet Explorer.

NOTE: Users experienced with XML can edit the information contained within this file before importing it into Manager, but care should be taken as there is risk in destroying the referential integrity of the assessment.

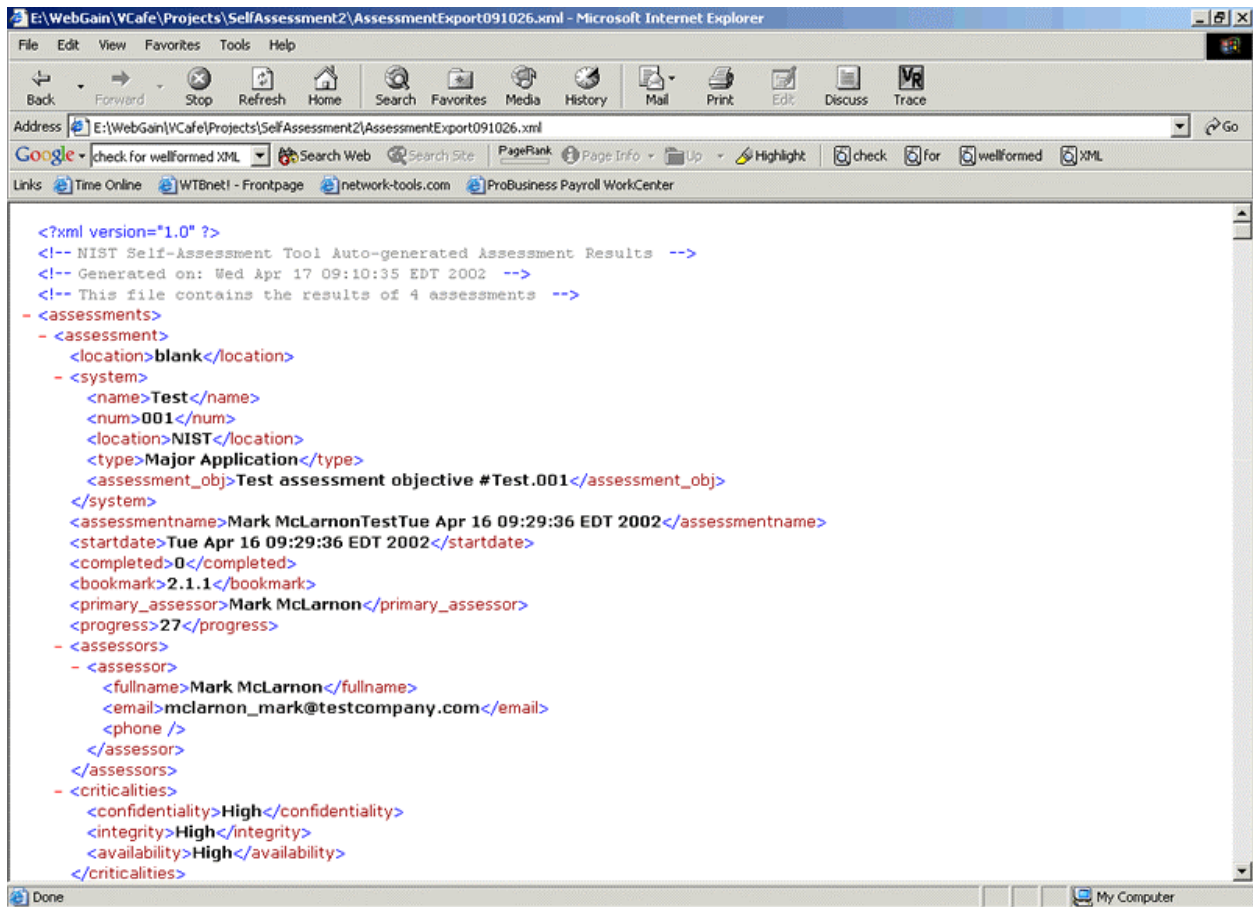


Figure 17 - Sample XML Assessment Export

The assessment import process is discussed in detail in the Manager section.

3.3.5 Question map construction

Within System, the question map construction is a two-step process. The map is not constructed during an assessment until the user completes the identification process, the first and second tabs. Due to its complexity, the question map is serialized to a local outfile named map.dat as shown by the following folder listing for System.

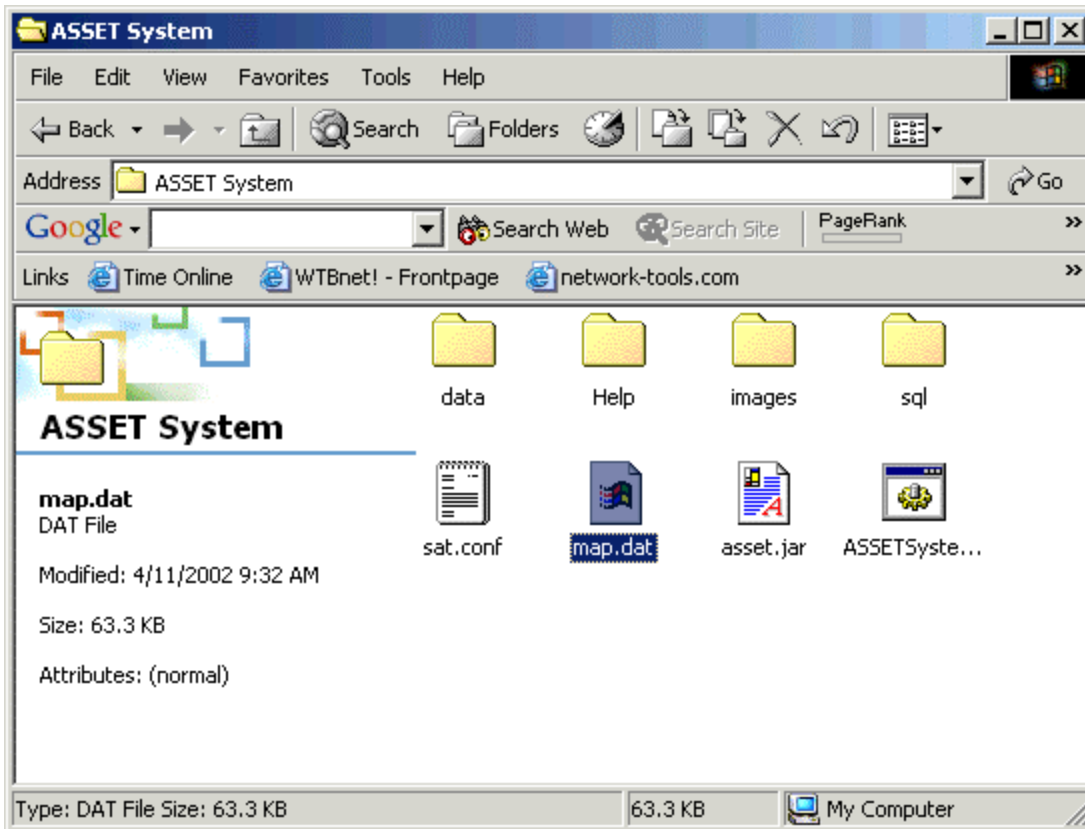


Figure 18 - System question map cache file

The first step in the build process is to check for the presence of the cached question map. This is done by checking for the existence of the file from the figure above. If the map cache file is not located, the second step is to rebuild the map from scratch. The following code snippet performs this first step:

```

1 try {
2     File check_exists = new
3     File(this.parent.getPropertyManager().getProperty("sat.map.path"));
4     if (check_exists.exists()) {
5         FileInputStream fis = new
6         FileInputStream(this.parent.getPropertyManager().getProperty("sat.map.path"));
7         ObjectInputStream ois = new ObjectInputStream(fis);
8         cache_data = (Vector) ois.readObject();
9         this.progress_model = (DefaultTreeModel) cache_data.elementAt(0);
10        this.progress_tree.setCellRenderer(custom_renderer);
11        this.progress_tree.setModel(this.progress_model);
12        this.setMapStatus(true);
13        this.node_map = (Hashtable) cache_data.elementAt(1);
14        this.progress_tree.setVisible(true);
15        this.root_node = (DefaultMutableTreeNode)
16        this.progress_model.getRoot();
17    }else {
18        this.rebuildTree();
19    }
20    this.prgMonitor.setVisible(false);
21 }catch (Exception e) {
22     this.parent.log(e.toString());
23 }

```

Lines 2-4 show the conditional check for the existence of the file “map.dat.” This property is represented by the System property **sat.map.path**. Lines 16-18 are the alternative of this conditional; if the cache is

not located, then the method `rebuildTree()` is executed. The `rebuildTree()` method takes the following form:

1. Create root tree node
2. Insert policy node
3. Extract current control areas
4. Loop through current control areas
 - a. Extract questions for specific control area
 - b. Loop through questions for specific control area
 - i. Insert question into tree
 - ii. Insert question into node hashtable (enables lookups)
5. Serialize tree to local file `map.dat`.

The loop that processes questions examines the question number (major section, minor section, and question number) and based on a conditional, inserts a critical element or a normal question. Users should also note that this is the spot where the policy table is computed as well. When a question is extracted from the database that has a minor section of 0 and a question number of 0, this question is inserted into the policy table model. The policy table model is an instance of the `DefaultTableModel` class defined as follows:

```
DefaultTableModel criticalElementModel = new DefaultTableModel() {
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
```

This definition of the `DefaultTableModel` class redefines the `isCellEditable(int, int)` method to prevent any of the cells within the table from being modified by a mouse event. When each question is inserted into the tree, they are inserted into the root node of the `JTree` as follows:

```
this.progress_model.insertNodeInto(question, critical_element, question_count);
```

Each question is wrapped within an instance of the `HybridTreeNode` class, which extends the `DefaultMutableTreeNode` class. The node hashtable **node_map** is the data structure that is searched when a user moves to a new question. The method `displayQuestion()` calls the `findQuestionNode(String)` method. This method definition is as follows:

```
private HybridTreeNode findQuestionNode(String question_num) {
    HybridTreeNode node = null;
    try {
        node = (HybridTreeNode) this.node_map.get(question_num);
    } catch (Exception e) {
        this.parent.log("ERROR: node does not exist");
        this.parent.log(e.toString());
    }
    return node;
}
```

This search method was implemented because there is no feasible method to access the elements in the question map as the user moves forward in an assessment. The `valueChanged(TreeSelectionEvent)` method of the `TreeSelectionListener` interface publishes what the currently selected node is. When a user moved forwards or backwards, they didn't generate a `TreeSelectionEvent` by default. With the question map, once the correct node to be displayed is located, this node is selected with a `TreeSelectionEvent` by the following method from `AssessmentWindow`:

```
private void selectNode(HybridTreeNode node) {
```

```

try {
    if (node != null) {
        this.progress_tree.scrollPathToVisible(new TreePath(node.getPath()));
        this.progress_tree.setSelectionPath(new TreePath(node.getPath()));
    }else {
    }
}catch (Exception e) {
    this.parent.log(e.toString());
}
}
}

```

3.3.6 ASSET System Configuration File

This section describes the System configuration file **sat.conf**. ASSET was designed to be extensible, conforming to the Open-closed principle where the elements of System or Manager that were configurable are abstracted to a property file. Within System, this property file controls such things as the username/password combination that is used to authenticate to MSDE. The System property file is shown in the following folder listing of the System program folder. This property file must always be located in this spot or System will not load successfully.

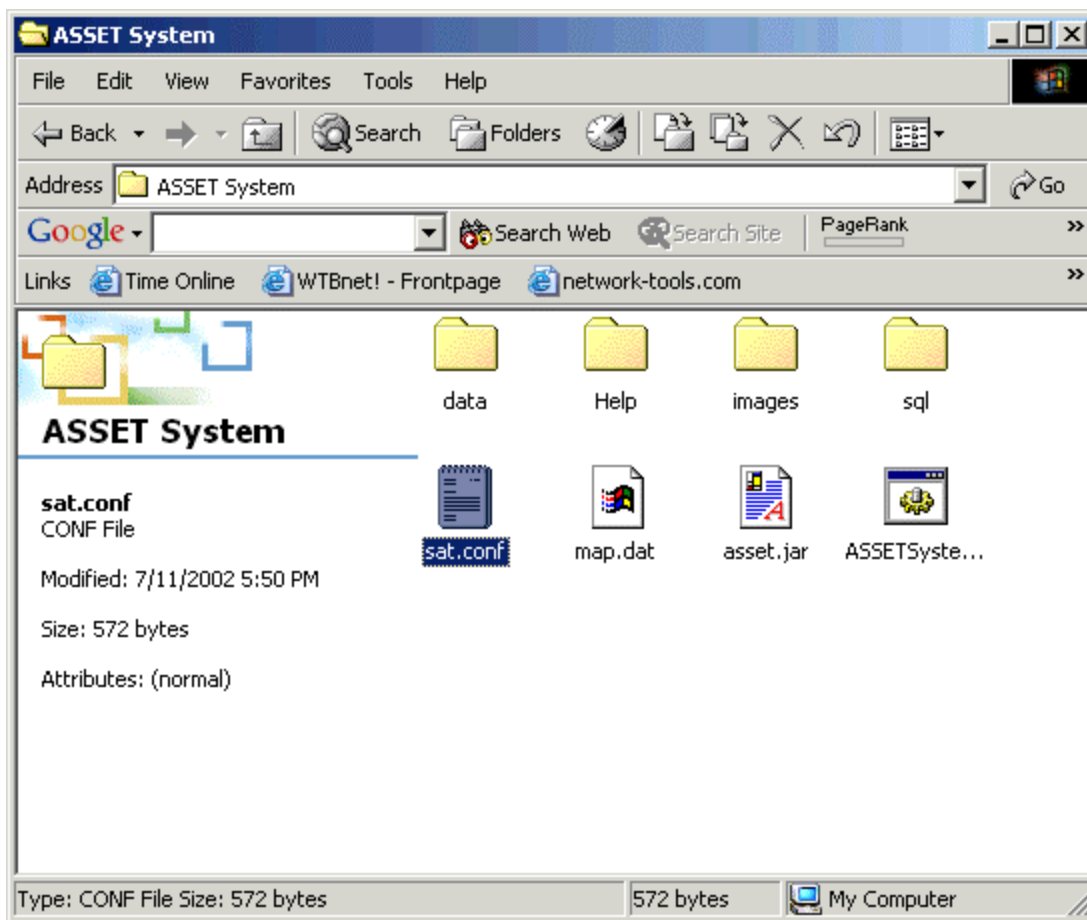


Figure 19 - System property file

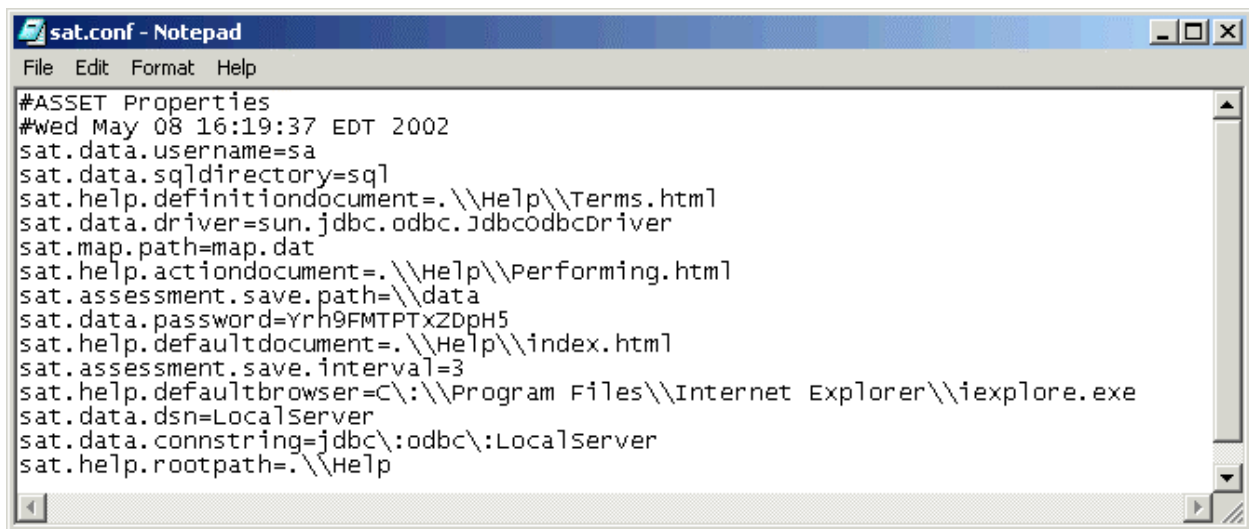
The properties contained within **sat.conf** follow the standard UNIX property file format of NAME=VALUE. The following table describes the properties contained within **sat.conf** and their corresponding purposes within System.

Property Name	Default Value	Description
sat.data.username	Sa	The user name used to connect through ODBC to the MSDE system. This defaults to the standard 'sa' account within MSDE. Provided that customized account creation procedures are provided, this may be changed to a new value.
sat.data.sqldirectory	sql	The folder where the SQL files are located to build the database.
sat.help.definitiondocument	.\Help\Terms.html	The help document used to define terms within System This relates to those components of System with corresponding help buttons.
sat.data.driver	sun.jdbc.odbc.JdbcOdbcDriver	The JDBC driver used to connect to the MSDE system. This was done to allow different database drivers to be used.
sat.map.path	map.dat	The local path where the question map is serialized.
sat.help.actiondocument	.\Help\Performing.html	The help document that contains the descriptions for each action that has an associated help button within System.
sat.assessment.save.path	\\data	The local path where the active assessment backups are stored.
sat.data.password	System dependent	This is the corresponding password to the property value sat.data.username which again is used to authenticate to the MSDE system.
sat.help.defaultdocument	.\Help\index.html	This is the default Help document for System.
sat.assessment.save.interval	3	This is interval of minutes that System will wait to back up the currently active assessment. Within System, this is translated into milliseconds for the <code>Thread.sleep()</code> method call.
sat.help.defaultbrowser	C:\Program Files\Internet Explorer\iexplore.exe	This is the system-independent path to the user's web browser which they will use to view the help files. This option can be configured within the system Help menu using the Change default browser option.
sat.data.dsn	LocalServer	The ODBC DSN that is used to connect to MSDE. This can be changed to a valid DSN.
sat.data.connstring	jdbc:odbc:LocalServer	The connection string supplied to the JDBC connection process. NOTE here that the JDBC/ODBC bridge is being used.
sat.help.rootpath	.\Help	The root location for the help files of System. This value is used to assemble a file object with which to open a help file.

Table 3 - System property file contents

Within System, these properties were accessible from the ASSETClientPropertyManager object. These properties were inherited from the ASSETPropertyManager object. The ASSETPropertyManager object essentially is a convenience wrapper for the Java Properties class ([java.util.Properties](#)). The core class ASSETPropertyManager contains the property access and modification methods from which System and Manager inherit.

The ASSETPropertyManager contains the ability to add new properties as well. This allows a user to change the default browser used to view help files. This also is the reason why when the property file is saved, the properties are printed with escape characters denoted with the ‘\’ character. This is a function of the `save(OutputStream, String)` method of the Properties class. The following figure shows an example of the System property file. Note that all properties that refer to the path for a particular resource are denoted as a relative path (relative to the parent path of C:\Program Files\NISTASSET\ASSET System).



```
#ASSET Properties
#wed May 08 16:19:37 EDT 2002
sat.data.username=sa
sat.data.sqldirectory=sql
sat.help.definitiondocument=.\Help\Terms.html
sat.data.driver=sun.jdbc.odbc.JdbcOdbcDriver
sat.map.path=map.dat
sat.help.actiondocument=.\Help\Performing.html
sat.assessment.save.path=\\data
sat.data.password=Yrh9FMTPTxZDpH5
sat.help.defaultdocument=.\Help\index.html
sat.assessment.save.interval=3
sat.help.defaultbrowser=C:\\Program Files\\Internet Explorer\\iexplore.exe
sat.data.dsn=LocalServer
sat.data.connstring=jdbc\:odbc\:LocalServer
sat.help.rootpath=.\Help
```

Figure 20 - Example System Property File

3.3.7 ASSET System reports

This section describes the possible reports for System. The reporting functionality offered within System was meant to provide a supplementary view of assessment data for only **one system at a time**. Because System reports deal with one system at a time, the System reporting process requires that a single assessment be chosen in order to run a report. The reporting functionality offered by Manager was meant to provide an organization-wide view of assessment results. The following table describes each report offered by System and its corresponding purpose, and the stored procedure that implements this report within the database. The stored procedures are discussed later in the document.

Report	Purpose	Stored Procedure
Summary of topic areas by levels of effectiveness	Lists the ITSAF level reached for each of the 17 topic areas of a single assessment.	get_policy_results_and_text
List of non-applicable questions	Lists those questions marked as Not-applicable for a specific assessment.	get_na_questions
List of risk-based decisions	Lists those questions marked as Risk-based decision made for a specific assessment	get_rbdm_questions
System Summary	Liists the applicable response elements for all of the questions for a particular assessment.	get_summary_question_results

Table 4 - Available System Reports

The following figure shows the user interace of the System reporting frame. When System reporting is open, the status bar mode indicator reads “RPRT.” Note that similar to an active assessment, this interface is also implemented as a JInternalFrame. This allows a user to switch back and forth between an active assessment and the reporting frame should both windows be minimized on the virtual desktop.

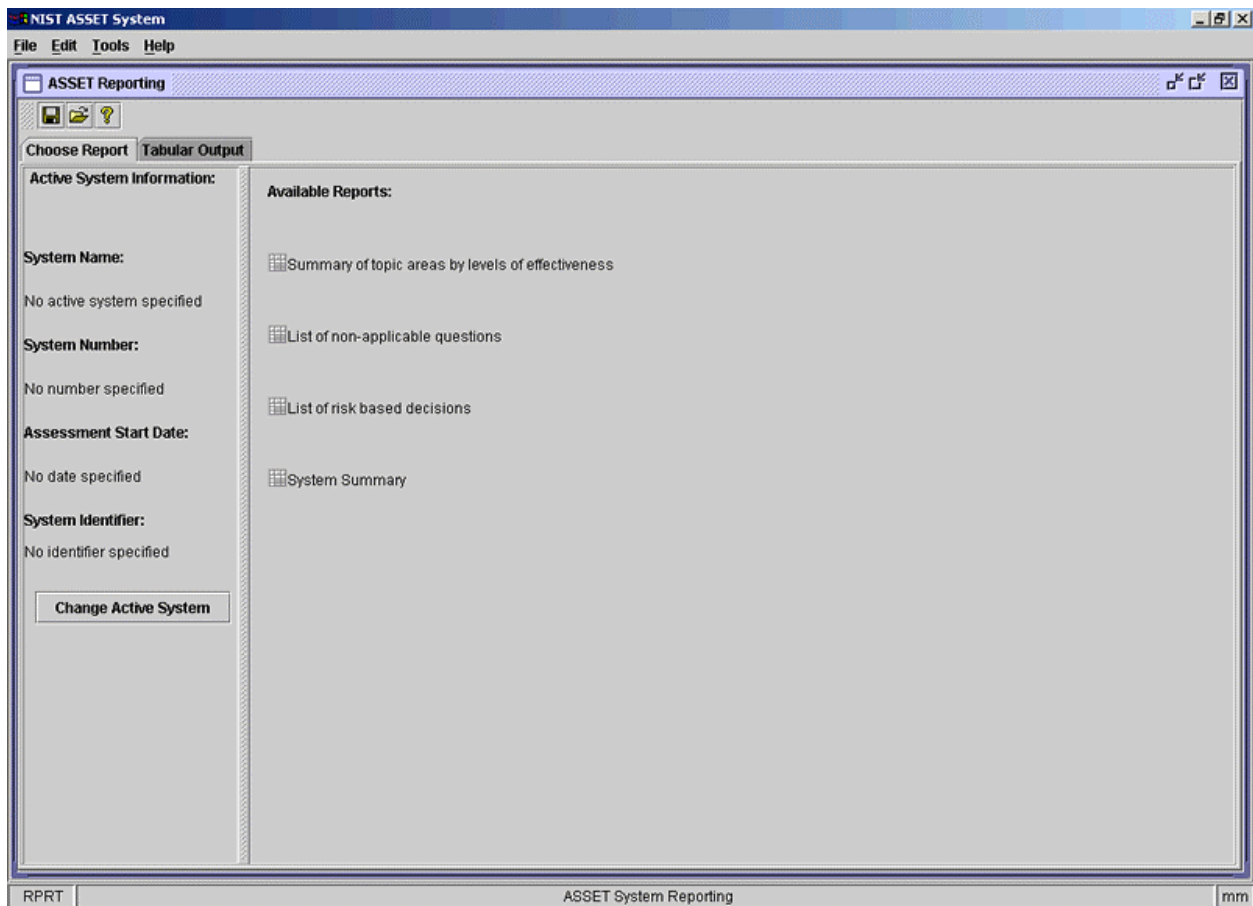


Figure 21 - System reporting interface

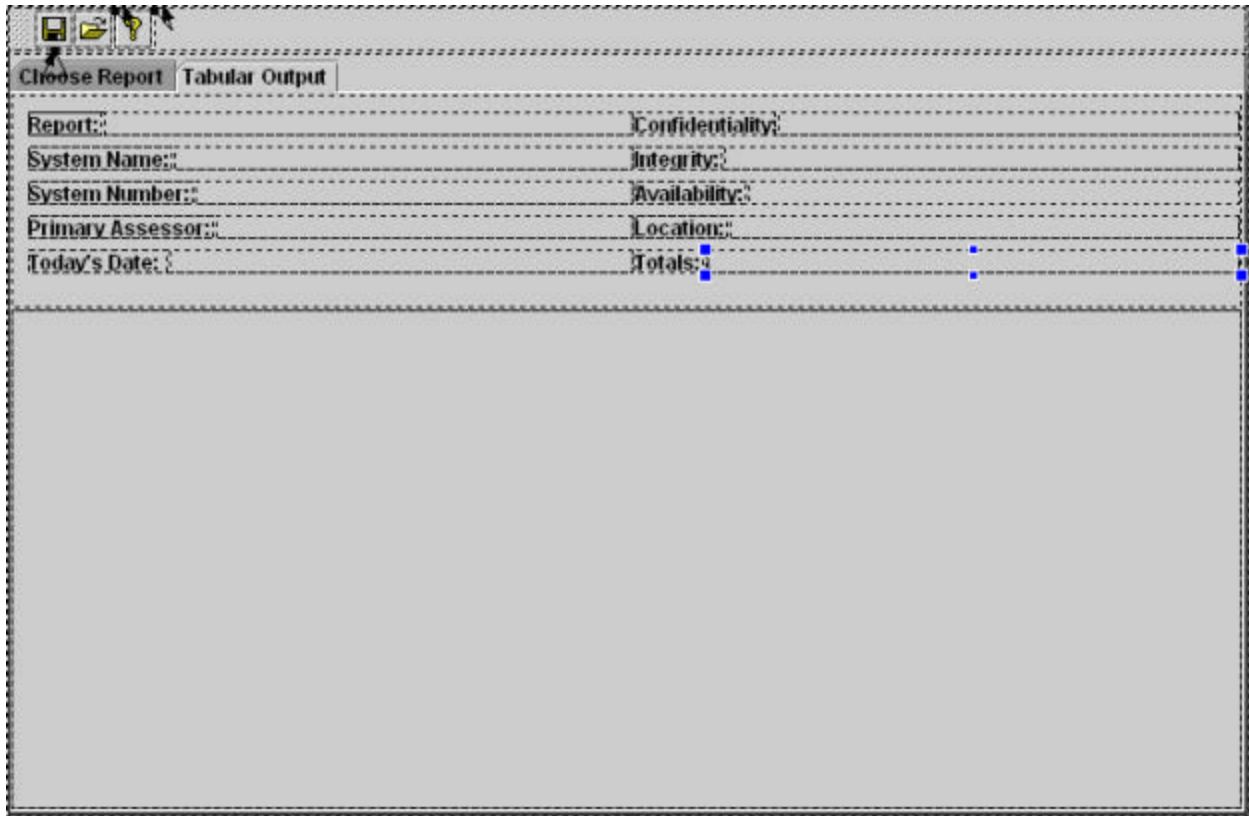


Figure 23 - System Report Screen Layout

Each report within System has the same report headers, whereas the column headers differ as shown in the previous figure. The following table describes the report headers for System and their meaning.

Header	Meaning
Report	Lists the title of a particular report. The possible titles are listed in Table 4 - Available System Reports.
System Name	Lists the name of the system in question as specified in the System Name field during the assessment identification. This is a arbitrary user field.
System Number	Lists the number of the system in question as specified in the System Number field during the assessment identification. This is a arbitrary user field.
Primary Assessor	Lists the primary assessor of the selected assessment. This is the person that was logged into System when the assessment was created.
Today's Date	Lists the current date and time that the report was run. This is provided by the <code>Date().toString()</code> method call.
Confidentiality	Lists the confidentiality level of the selected system as chosen by the user during the assessment identification.
Integrity	Lists the integrity level of the selected system as chosen by the user during the assessment identification.
Availability	Lists the availability level of the selected system as chosen by the user during the assessment identification.
Location	Lists the location of the system as determined by the user during the assessment identification.
Totals	Lists the number of records returned.

Table 5- System Report Headers

The column headers for each report are as follows, the report headers for the System Summary report are:

- Question #
- Question Text
- Level Reached
- N/A
- Risk-Based Decision?
- Comments
- Answered By
- Completed?
- Refer To

The report headers for the Topic Area summary report are:

- Topic #
- Topic Area
- Level

The report headers for the Non-applicable and Risk-based decision reports are:

- Question Number
- Question Text
- Comments

These headers are encapsulated within a Vector object, which is passed to the table model of the results table. Each report can be exported to a tab-delimited file using the save function on the reporting tool bar. The following figure shows the JFileChooser export dialog.

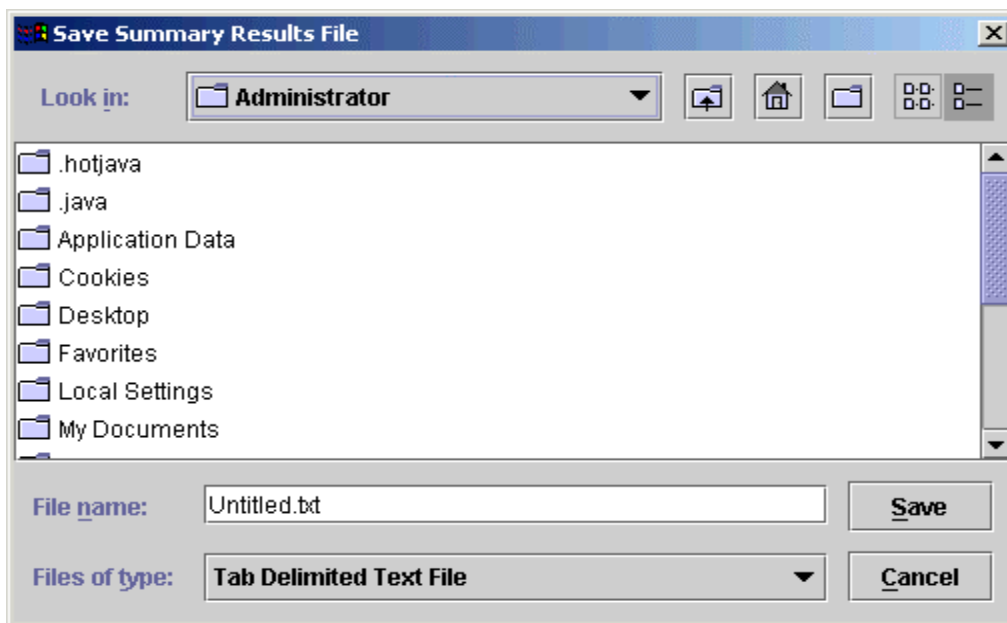


Figure 24 - System Reporting Export Dialog

3.4 ASSET Manager

This section describes the operation and design of Manager. Users should note that the ASSET User Manual provides a detailed description on the operation and installation of Manager. The intended purpose of Manager was the major influence behind its design, even more so than System. In short, Manager is designed to aggregate the results from one or more instances of System. Since not all organizations have a widespread enterprise or the necessary manpower to need multiple instances of System, Manager offers the same functionality as System in both assessments gathering and reporting.

3.4.1 ASSET Manager Operation

The primary function of Manager is importing and searching imported assessments. In an effort to promote component reuse, the same code that provides importing functionality in System is used in Manager. As stated previously, Manager also has the ability to create/open/export assessments in a similar method to System. The following syntax (taken from **ASSETManager.bat**) launches Manager. Note the presence of the System archive file in the classpath; this file is required for Manager to execute:

```
java -cp "...\calendar.jar;..\ASSET
System\asset.jar;..\xercesImpl.jar;..\xmlParserAPIs.jar;..\asset_core.jar;asset_admin.jar"
SATAdmin →
```

Within Manager, users have the option to open a previously saved assessment or create a new assessment. The following code excerpt from the main Manager class **ASSETManager** shows the action of creating a new assessment:

```
1 public void startNewAssessment() {
2 try {
3     this.assessment = null;
4     this.setStatusModeMsg(" AIP ");
5     this.setStatusModeMsg("Creating new assessment window now, please hold.");
6     this.assessment = new AssessmentWindow(this);
7     this.assessment.setClosable(true);
8     this.assessment.setPrimaryAssessor(this.primary_assessor, →
        this.primary_assessor.getDataVector());
9     workspace.add(this.assessment);
10    this.assessment.setMaximum(true);
11    this.assessment.setVisible(true);
12 }catch (Exception e) {
13     e.printStackTrace();
14 }
15 }/* end method startNewAssessment() */
```

The ability to create a new assessment forced the structure of the Manager database to match the System database. The only real difference between the two databases is the stored procedures. When a new assessment is saved within Manager, the assessment information is saved within the Manager database, not the System database. This difference is discussed further in the database section. The following figure shows an example of a new assessment created with Manager. Note the use of **JInternalFrames** similar to System's user interface.

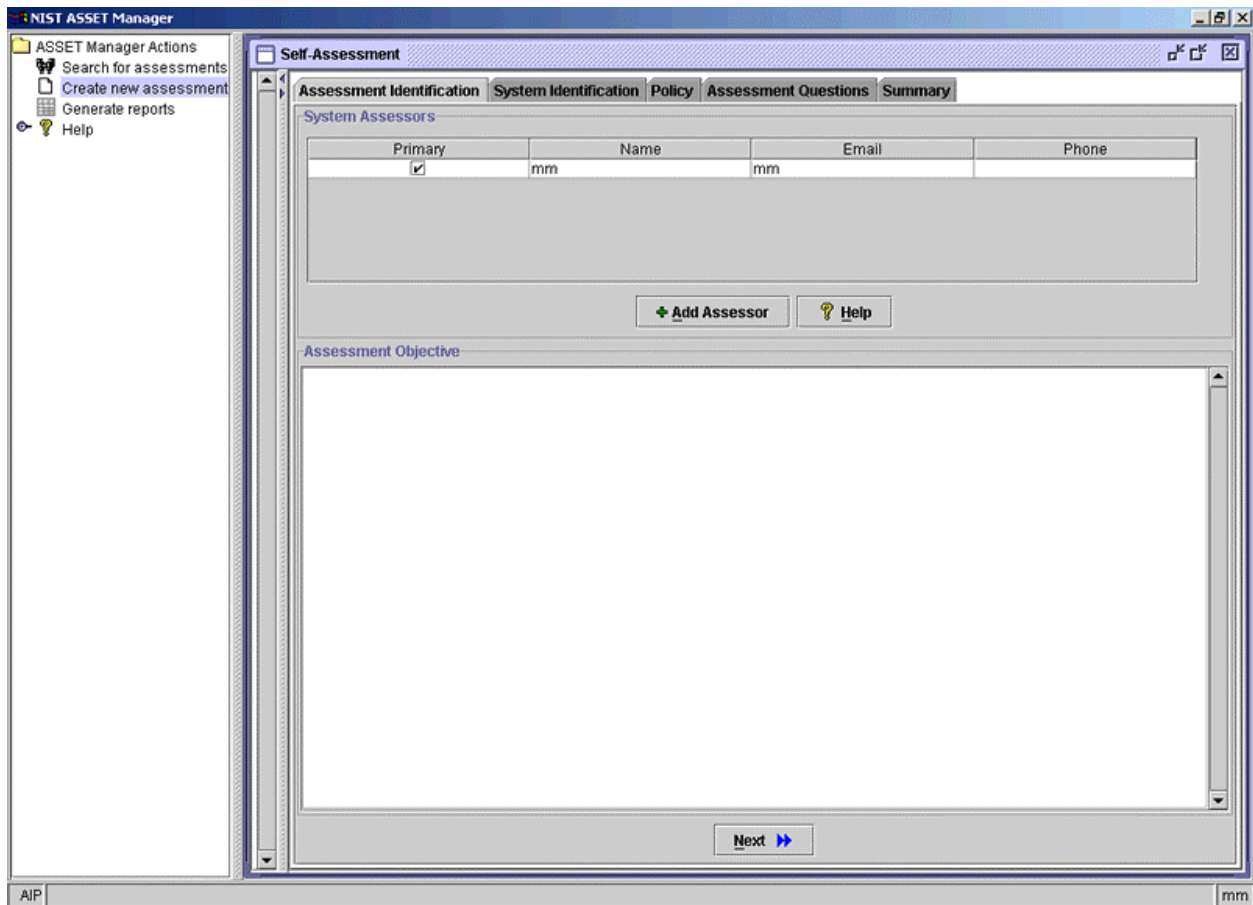


Figure 25 - Create New Assessment in Manager

The AssessmentImporter class, a part of the ASSET core package, provides import functionality to System and Manager. Similar to the AssessmentXMLGenerator class, the AssessmentImporter class also extends the Thread class to allow an import to run in the background. Once assessments have been imported, Manager can generate reports on these assessments.

3.4.2 Manager User Interface

This section discusses the Manager user interface. As shown in the previous figure, the Manager user interface model is an active assessment within System. Microsoft system administrators might recognize the inherent design of Manager as it relates to the Microsoft Management Console (MMC). The design of the MMC is similar to the design in Figure 13 - ASSET System AssessmentWindow layout, where the navigation menu occupies the left side of the screen and the Search and Reporting window occupy the right side. The Searching and Reporting windows are both instances of JInternalFrame.

3.4.3 ASSET Manager Configuration File

This section describes the Manager configuration file **admin.conf**. The structure of **admin.conf** is similar to System's property file **sat.conf** as described in section 3.3.6, ASSET System Configuration File. This property file is located in the root of the Manager directory as shown in the following figure.

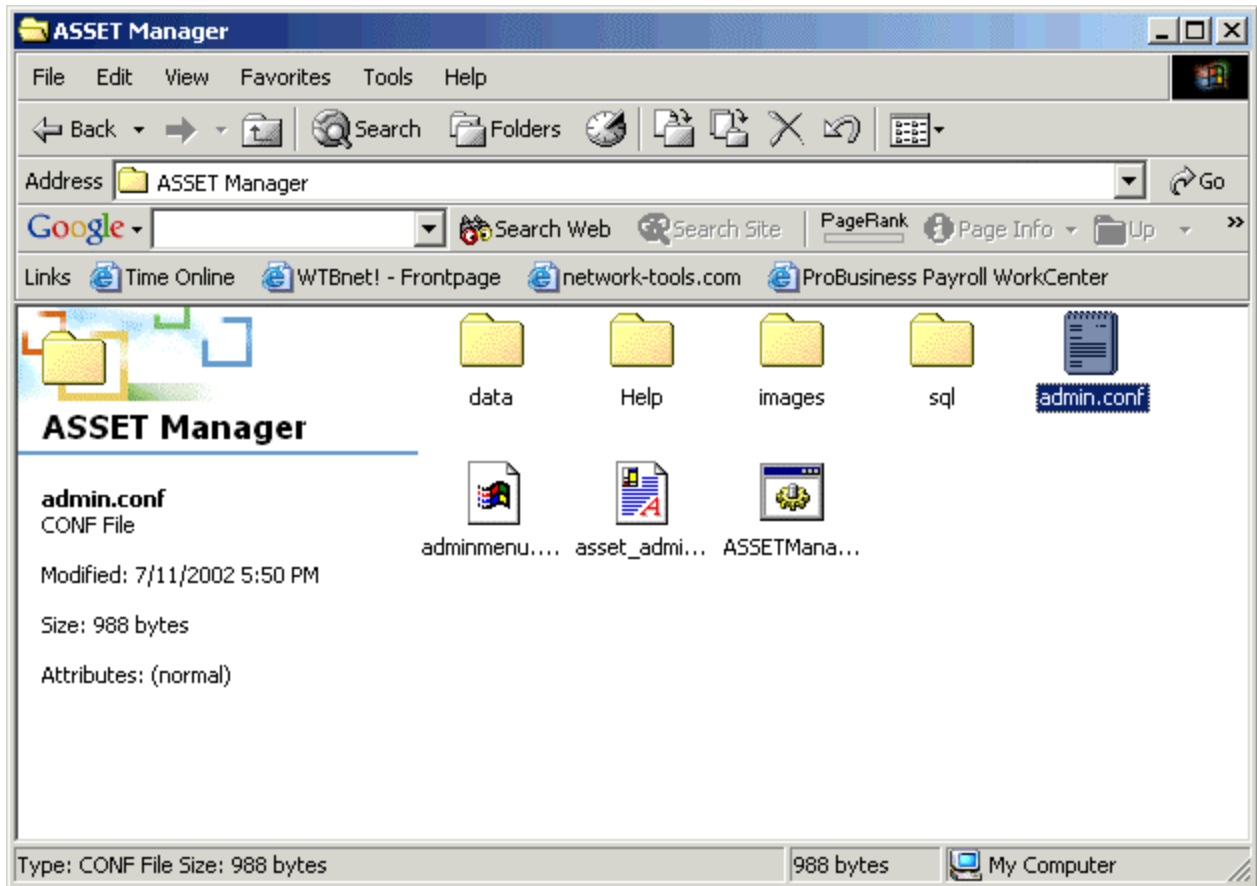


Figure 26 – Manager property file

The properties contained within **admin.conf** follow the standard UNIX property file format of NAME=VALUE. The following table describes the properties contained within **admin.conf** and their corresponding purposes within Manager.

Property Name	Default Value	Description
sat.help.defaultdocument	.\Help\ASSETSystem\index.html	This is the default Help document for System.
sat.help.rootpath	.\Help\ASSETSystem	The root location for the help files of System. This value is used to assemble a file object with which to open a help file.
sat.manager.help.assessment document	.\Help\processassessments.html	The help document that describes how to process assessments.
sat.data.username	sa	The user name used to connect through ODBC to the MSDE system. This defaults to the standard 'sa' account within MSDE. Provided that customized account creation procedures are provided, this may be changed to a new value.
sat.data.sqldirectory	sql	The folder where the SQL files are located to build the database.
sat.data.connstring	jdbc\:odbc\:MSDE_SAT_Admin_Data	The connection string supplied to the JDBC connection process. NOTE here that the JDBC/ODBC bridge is being used.
sat.data.driver	sun.jdbc.odbc.JdbcOdbcDriver	The JDBC driver used to connect to the MSDE system. This was done to allow different database drivers to be used.
sat.splashscreen.imagepath	images/SplashScreen.jpg	The file name of the Manager splash screen image.
sat.manager.help.defaultdocument	.\Help\index.html	The default Help document for Manager.
sat.assessment.save.interval	3	The time interval in minutes for the local save of an active assessment.
sat.display.defaultplaf	javax.swing.plaf.metal.MetalLookAndFeel	The default look and feel (PLAF) for Manager.
sat.help.defaultbrowser	C:\Program Files\Internet Explorer\iexplore.exe	This is the system-independent path to the user's web browser which they will use to view the help files. This option can be configured within the system Help menu using the Change default browser option.
sat.splashscreen.delay	6	The amount of time in seconds that the splash screen is displayed.
sat.help.actiondocument	.\Help\ASSETSystem\Performing.html	The help document that contains the descriptions for each action that has an associated help button within System.
sat.data.dsn	LocalServer	The ODBC DSN that is used to connect to MSDE. This can be changed to a valid DSN.
admin.menu.path	adminmenu.dat	The file name where the menu information for Manager is serialized and stored.
sat.assessment.save.path	data	The local path where the active assessment backups are stored.
sat.help.definitiondocument	.\Help\ASSETSystem\Terms.html	The Help document that defines terms and actions within Manager.
sat.map.path	map.dat	The local path where the question map is serialized.
sat.manager.help.reportingdocument	.\Help\reporting.html	The help document that describes the reporting process in Manager
sat.data.password	System dependent	This is the corresponding password to the property value sat.data.username which again is used to authenticate to the MSDE system.

Table 6 - Manager Property File Contents

3.4.4 ASSET Manager Menu

This section discusses the Manager menu tree. This menu was designed in a similar method to the System question map discussed in the System section. The Manager menu tree functions in place of a file menu or “tool bar” as is common to many office productivity applications such as Microsoft Office. The Manager menu is shown in the figure contained within the section entitled “ASSET Manager Operations,” and shown again as a cutout of greater detail in the following figure.



Figure 27 - Manager Menu Tree Example

Similar to the System question map, the Manager menu is an instance of JTree, which uses instances of DefaultMutableTreeNode for each of its tree nodes. The difference from the System question map is that the Manager menu uses a custom cell renderer for each of its nodes, MenuCellRenderer. Similar to the System question map, the Manager menu uses local serialization to cache the contents of this JTree and its data model. This cache file is named menu.dat and is shown in the following figure.

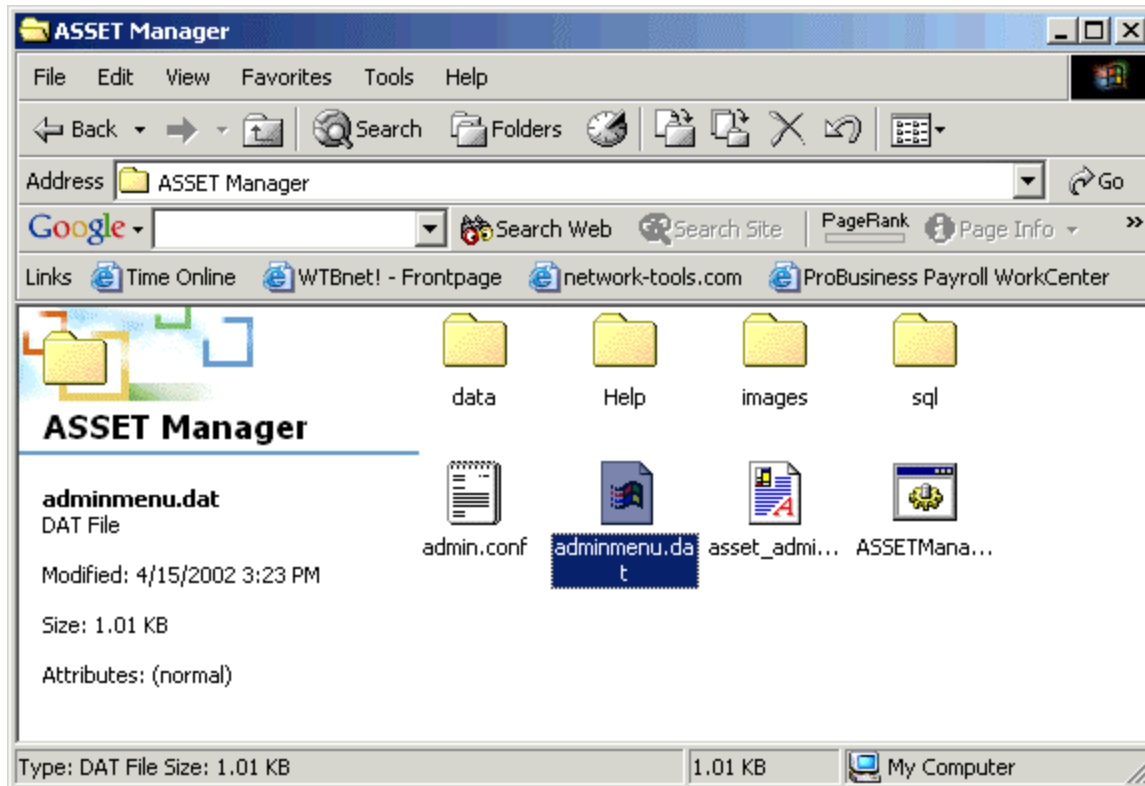


Figure 28 - ASSET Manager Menu Cache File

If users delete this file, the administration menu will be rebuilt each time and resaved to this local file.

NOTE: Users may also experience a situation where the Menu options are obviously incorrect. When the menu items are things like fruit and sports, these are items that Java uses as the default menu to a JTree. This is caused by the table model becoming corrupted. Users can rectify this situation if they delete the menu cache file and restart Manager.

3.4.5 Searching in ASSET Manager

This section describes searching in ASSET Manager. Searching is the primary function of Manager. The Manager user interface revolves around users querying the Manager database and generating reports from the search results. After a user has performed a search of the Manager database, they can select an arbitrary subset of their search results to develop a report. The search window is shown in the following figure:

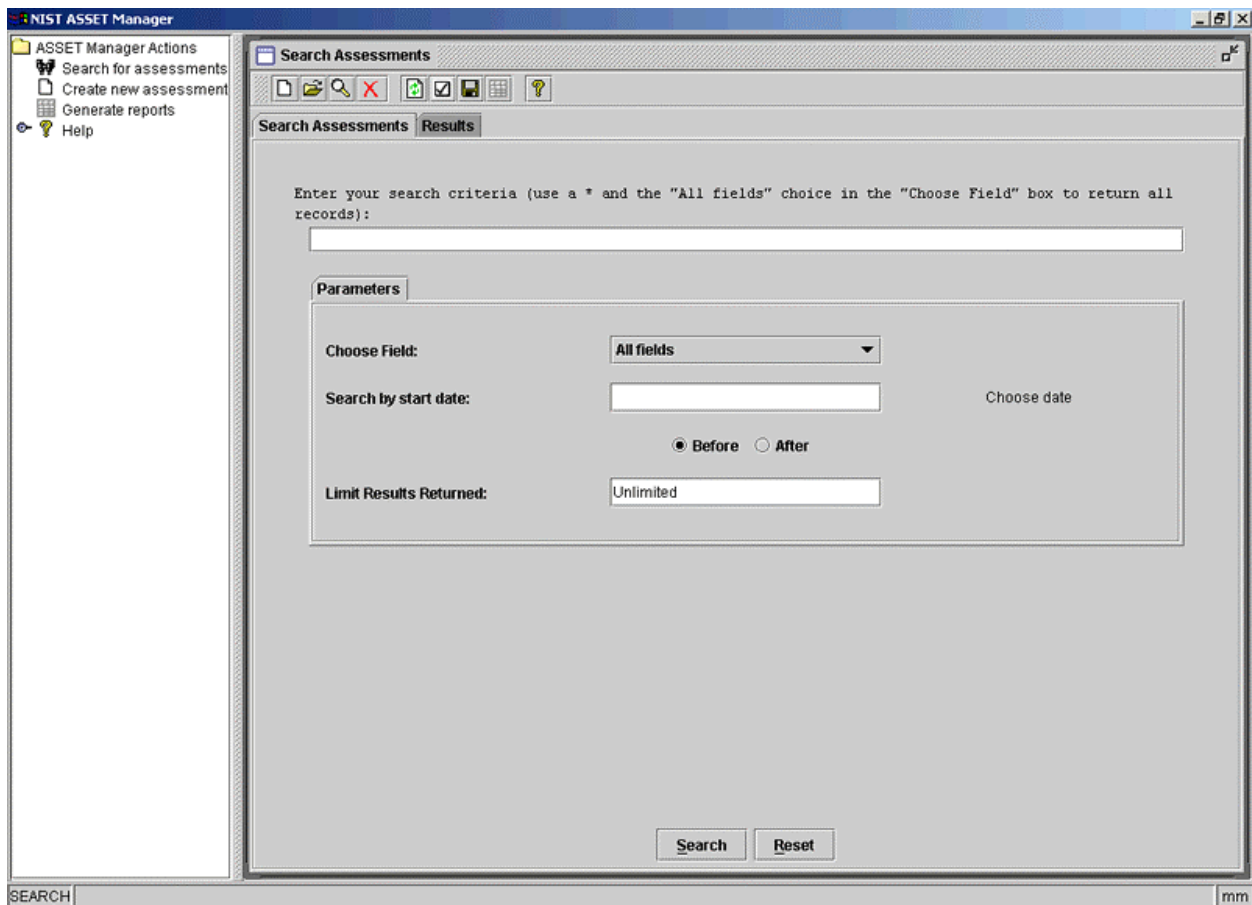


Figure 29 - Manager Search Interface

This user design follows both System's AssessmentWindow class discussed in the System section and the ClientReportingFrame class for Manager reporting. The only exception is the design of the parameters tab in the middle of the figure above. This tab is managed by a GridLayoutManager with three columns and four rows making a total of 12 cells. Those cells that appear empty are actually filled with empty JLabels.

Users have the option with the Choose Field dropdown box to choose which field from the database they wish to search. The default option is All Fields. This means that if a user enters a '*' in the search text box they will return all records from the database. Searching the manager database feeds into the next section on Manager Reporting.

3.4.6 ASSET Manager Reporting

This section details ASSET Manager reporting. The design of the Manager reporting user interface closely follows that of System reporting user interface. Manager reporting also contains the ability to run all of the reports of System. The following table lists the reports that Manager can run and their associated stored procedures that implement the reporting code within the database.

Report	Purpose	Stored Procedure
Summary of all Systems	This report lists all of the systems within the Manager database and the associated level that each system has reached for each of the control areas within the Manager.	get_all_assessment_information
Summary of System Types	This system lists all of the systems within the Manager database and their associated system type.	get_systems_by_type
Summary of Systems by Sensitivity	This system lists all of the systems within the Manager database and their associated System sensitivity levels for confidentiality, integrity, and availability.	get_system_sensitivity
Summary of Systems by Organization	This system lists all of the systems within the Manager database and their associated organization/location within an enterprise.	get_system_organization

Table 7 – Available Manager Reports

The following headings are used for the System Summary Report: (Note that 17 of these headings are the 17 major control areas from the initial version of the NIST SP 800-26. This number may change in the future.)

- Name
- Number
- Type
- Organization
- Primary Assessor
- Confidentiality
- Integrity
- Availability
- Risk Management
- Review of Security Controls
- Life Cycle
- Authorize Processing (Certification and Accreditation)
- System Security Plan
- Personnel Security
- Physical Security

- Production, Input/Output Controls
- Contingency Planning
- Hardware and Systems Software Maintenance
- Data Integrity
- Documentation
- Security Awareness, Training, and Education
- Incident Response Capability
- Identification and Authentication
- Logical Access Controls
- Audit Trails
- Completed?

The following headers are used for the System Type report:

- Name
- Number
- Type

The following report headers are used for the System Sensitivity report:

- Name
- Number
- Confidentiality
- Integrity
- Availability

Manager reporting contains the ability to draw its reporting data from an arbitrary subset of the entire Manager database. After a user searches the Manager database for arbitrary assessments, they can mark a subset of their search results as 'active.' Once they mark a subset of the search results, they can open the Manager reporting from the Search window tool bar. Once opened, any of the reports that they generate will be drawn from the subset of assessments.

3.5 Sorting in ASSET

This section addresses the option of sorting data in System and Manager. It is important to note that sorting was not an initial requirement of ASSET; it was proposed as a convenience to System and Manager. Specifically, the data models for the each of the JTables within System and Manager initially contained the ability to sort data in two directions. The sorting method proposed was based on information from the following URL:

<http://www2.gol.com/users/tame/swing/examples/JTableExamples5.html>

This method utilized the following classes:

- SortableTableModel.java
- TableSorter.java
- SortButtonRenderer.java
- BevelArrowIcon.java
- BlankIcon.java

These classes are contained within the Core project. This sorting method used the SortButtonRenderer (extends CellRender class) class to enable the column headers to act like JButtons. The SortableTableModel extends the DefaultTableModel to enable the table model data vectors to be sorted. The following code snippet shows how this sorting method is implemented:

```
1 SortButtonRenderer renderer = new SortButtonRenderer();
2 TableColumnModel model = table.getColumnModel();
3 int n = headerStr.length;
4 for (int i = 0; i < n; i++) {
5     model.getColumnModel(i).setHeaderRenderer(renderer);
6     model.getColumnModel(i).setPreferredWidth(columnWidth[i]);
7 }
8 JTableHeader header = table.getTableHeader();
9 header.addMouseListener(new HeaderListener(header,renderer));
```

Lines 5 and 6 show how the SortButtonRenderer object is attached to the table columns. The problem with this sorting process as it relates to ASSET is that it is not an in-memory sort, only graphical. For instance, the loadDialog class uses a JTable to list the available assessments to load from the database. To do this, when a user chooses an assessment and then presses the load assessment button, the action handling method uses the currently selected row on the JTable to determine which assessment was selected. If this sorting method were implemented, and the user was to sort the table columns, then the assessments that were displayed on-screen would be different from the actual contents in memory.

Sorting also played a part in the reporting for System and Manager. If a report was sorted after it was executed, and then saved to an outfile, the order in which the report contents were displayed on-screen was different from the the order in which the data was written to the outfile. In addition, certain columns within the various SAT reports did not accept the sorting feature in testing. Because sorting was not an initial requirement of ASSET, it was abandoned in place of other more important requirements. Sorting is an important feature to have in ASSET and therefore may be considered in a future release of ASSET.

3.6 ASSET Help

This section addresses the ASSET help files for both System and Manager. The specific file names of the help files for System and Manager were named in the File Requirements section earlier in the document. The help content for System and Manger is presented in HTML format viewable by any common web browser that is HTML 3.2 compliant. The structure of System help is designed to be viewed in one of two ways:

- Pressing associated help button for specific action
- Accessing help files from Help menu option on SATUI

The help files for System are reproduced within Manager because of the ability of Manager to interact with assessments directly. Manager's help files and hyperlink interactivity are structured in the same way as System. The method by which System and Manager communicate with the help files is through anchors within the HTML source code for the help pages. Initially the help files were intended to do things like explain each of the ITSAF levels that a question has achieved, or define confidentiality, integrity, and availability in terms of NIST SP 800-26. So as shown in the following snippet of HTML from System help (taken from terms.html), an HTML anchor defines the location of the term "boundary controls."

```
<P><font face="Arial, Helvetica, sans-serif" size="2"><I><B><A NAME="boundarycontrols">Boundary
Controls</A></B></I>- the security measures that protect an interconnected system
from unauthorized access.</font></P>
```

The structure of the anchor text is that the anchor is comprised of the lower case concatenation of the word or action it is defining without any whitespace, so in the case of the previous example, if a user wanted to define what “boundary controls” meant, they would need to search for an anchor “boundarycontrols.” Lines 12 and 13 of ASSET.java show the two methods that open help files for system, `defineTerm(String)` and `defineAction(String)`. The parameter for each of these methods is the HTML anchor to display in the associated help file. The following snippet of Java code shows how `defineTerm(String)` is coded within System.

```
public void defineTerm(String term) {
    this.execute(this.props.getProperty("sat.help.defaultbrowser") + " " +
    this.getFullPath(this.props.getProperty("sat.help.definitiondocument")+ "#" + term);
}
```

This method uses two properties from System’s configuration file, `sat.help.defaultbrowser`, and `sat.help.definitiondocument`. Refer to the section describing the System configuration file for more information. With regards to the previous example, the inclusion of the '#' character means that the hyperlink that is going to be opened by the user’s default browser will be to a named anchor on the HTML page of term definitions.

NOTE: Normally, interacting in Java source with the underlying OS causes platform dependencies. The execute method of SATUI, however, does not use Windows-specific commands to execute the system’s web browser and can work the same way under Unix/Linux.

The property `sat.help.defaultbrowser` can be changed to match whatever web browser the user prefers. Users change this through the Change default browser option on the System help menu. The `ChangeBrowserWizzard` class performs this change; all it does is use a `JFileChooser` to search for a file with a `.exe` extension that the user chooses. Once they choose a browser, this property is reset in the System config file. Because all installations of Windows contain Internet Explorer by default, this was the default value to set this property. If the property was not set to Internet Explorer’s executeable and instead to Netscape, and users did not have this browser installed, they would think there was a problem with ASSET when it complained about not being able to launch a binary that was not present. The following dialog is displayed when a user attempts to change their default browser.

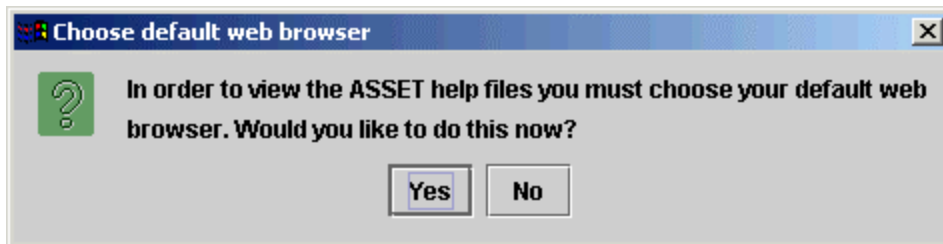


Figure 30 - Choose Browser Wizard

If the user presses “Yes,” they are next presented with a `JFileChooser` to choose the browser to view the help files. This `JFileChooser` uses a file filter to display files with an `.exe` extension only. If the user presses “No” at the previous dialog, they are presented with the following warning dialog.



Figure 31 - Cancel Choose Browser Wizard

3.7 ASSET Core

This section discusses the Core project of ASSET. The concept behind the core project was to package all of the components that are used by both System and Manager within a single JAR, **asset_core.jar**. Because both System and Manager then require this JAR file in their runtime classpaths, the startup scripts for both applications included this in their classpath. The following command line syntax launches System (taken from **ASSETSystem.bat**):

```
C:\>java -cp ..\asset_core.jar;asset.jar;..\xercesImpl.jar;..\xmlParserAPIs.jar SATUI
```

This syntax shows **asset_core.jar** included within the runtime classpath of System. The following table lists and describes the source code files that comprise Core:

File	Description
AssessmentImporter.java	Class that provides assessment import functionality to both System and Manager
AssessmentXMLGenerator.java	Class that provides assessment export to System and Manager
Assessor.java	Class that encapsulates assessor information
ASSET.java	Parent interface implemented by System and Manager
ASSETPropertyManager.java	Parent property manager class for System and Manager
BevelArrowIcon.java	Class used to provide sorting
BlankIcon.java	Class used to provide sorting
CheckBoxRenderer.java	Enables checkbox actions within JTable columns
ChooseBrowserWizard.java	Allows user to chose default browser for help files
CSVFilter.java	File filter for JFileChooser
EXEFilter.java	File filter for JFileChooser
HTMLLabel.java	Label used for reporting user interface
LogWindow.java	Logging class
OpenLogDialog.java	Logging class
ProgressDialog.java	Dialog that displays JProgressBar
saveLogDialog.java	Logging class
SortableTableModel.java	Class used to provide sorting
SortButtonRenderer.java	Class used to provide sorting
TableSorter.java	Class used to provide sorting
TXTFilter.java	File filter for JFileChooser
XMLFilter.java	File filter for JFileChooser

To build Core within Visual Café, these files were added to the Core project window and compiled. No special configurations were necessary. Once deployed to asset_core.jar, System and Manager's project settings were modified to include this JAR in their project classpaths.

3.8 ASSET Change Password Utility

The ADPU is a utility application that interacts with the MSDE osql.exe utility to change the password of the MSDE 'sa' account as recommended by the following CERT bulletin:

<http://www.kb.cert.org/vuls/id/635463>

The ADPU project is composed of the following files:

- ChangePassword.java
- ConfFilter.java
- PasswordGenerator.java

The PasswordGenerator class generates random password strings using the java SecureRandom ([java.security.SecureRandom](#)) class. The ADPU is distributed within the JAR **ASSETPassword.jar**. This file has its main class specified within the JAR manifest file (manifest.mf) to allow the ADPU to be launched with javaw.exe. To launch the ADPU, users need to double click on the JAR file. The password change happens during the installation of ASSET. The process happens in two steps to add a layer of complexity. First the password is set to an arbitrary value of 'nis7@ss3t.' Then the ADPU utility is run. The interface of the utility is shown in the following figure. Due to the size of the window, no layout manager was used to manage the user interface of ADPU; all of the components on the ADPU are placed statically.



Figure 32 - ADPU User Interface

There are two options on how to set the password within this interface. Users can choose a password and enter it into the text box shown above, or they may choose the “Suggest a password” link. If users choose to enter their own password, they will be prompted to re-enter the password a second time for validation. The “Suggest a password” link launches the PasswordGenerator object. The options for the APDU are configured on the advanced tab. Here users can control how big of a password the automated generator can produce, and which configuration files to store the new password in, System, Manager or both. The maximum size password that MSDE can support is 128 characters. Also, if the MSDE is installed to a drive other than C:\, the advanced tab will allow the user to specify the exact path name to the osql.exe utility. Users have the option to reload the configuration files on the advanced tab. The advanced tab of the ADPU is shown in the following figure.

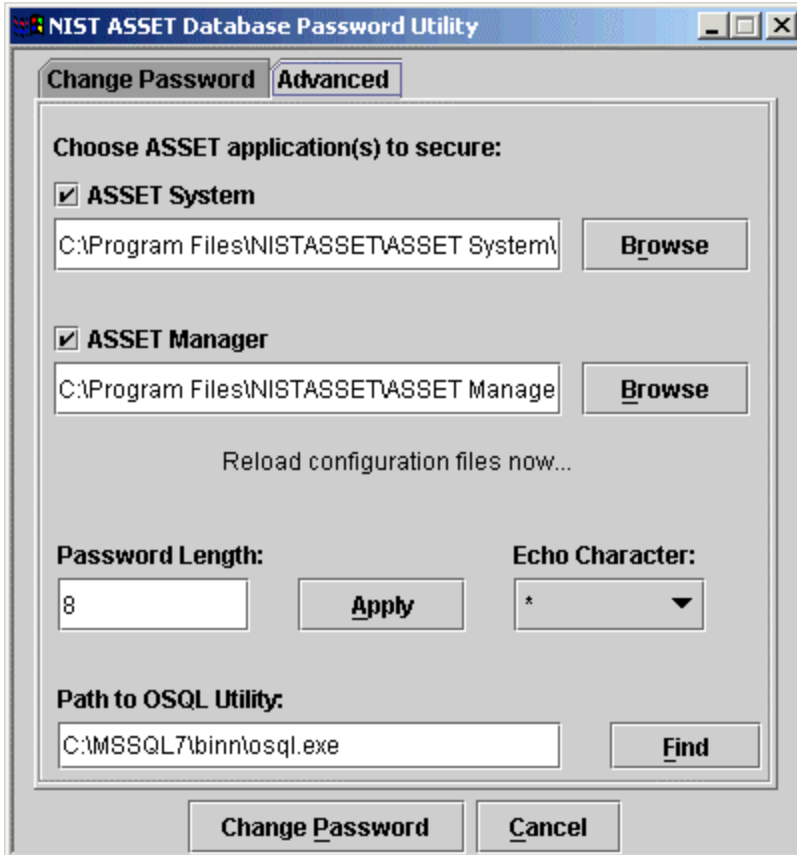


Figure 33 - ADPU Advanced tab

3.9 Procedure Flow Diagram for each Component

This section presents a process diagram for the intended operation of System and Manager. The system procedural flow diagram is presented in the following figure. This figure details the possible operations that a user can perform at a high level while system is active.

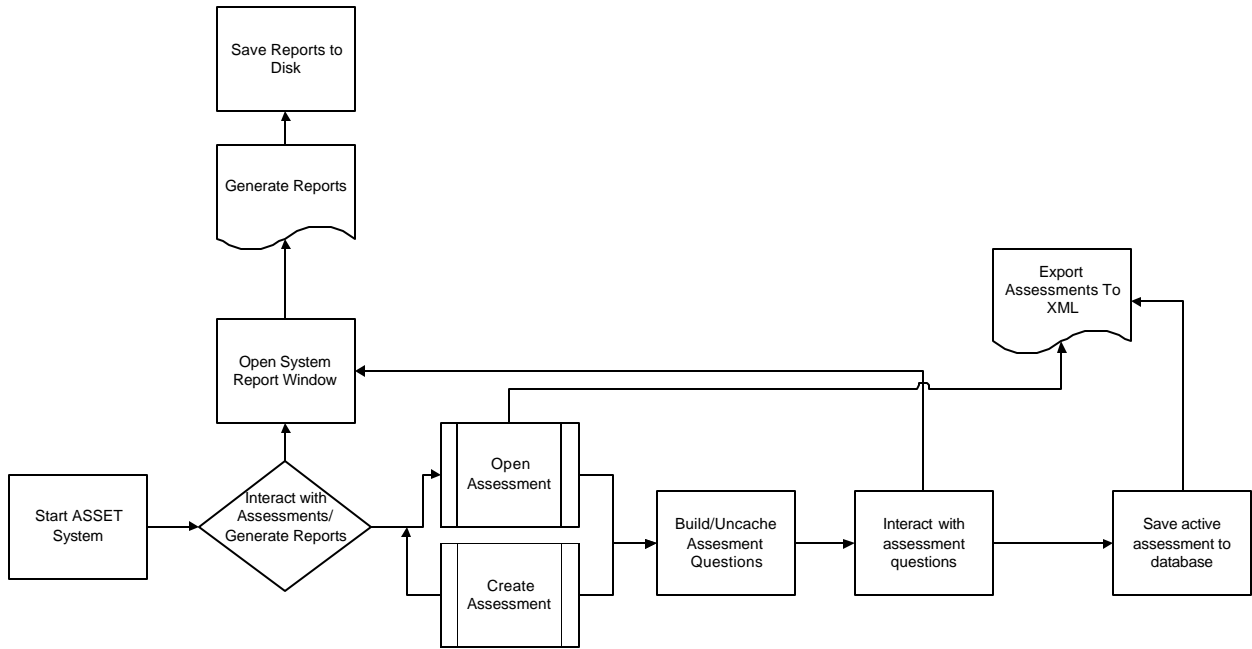


Figure 34 - System Procedure Flow

The procedural flow diagram for Manager is shown in the following figure. This diagram indicates the possible operations that a user can perform while Manager is active.

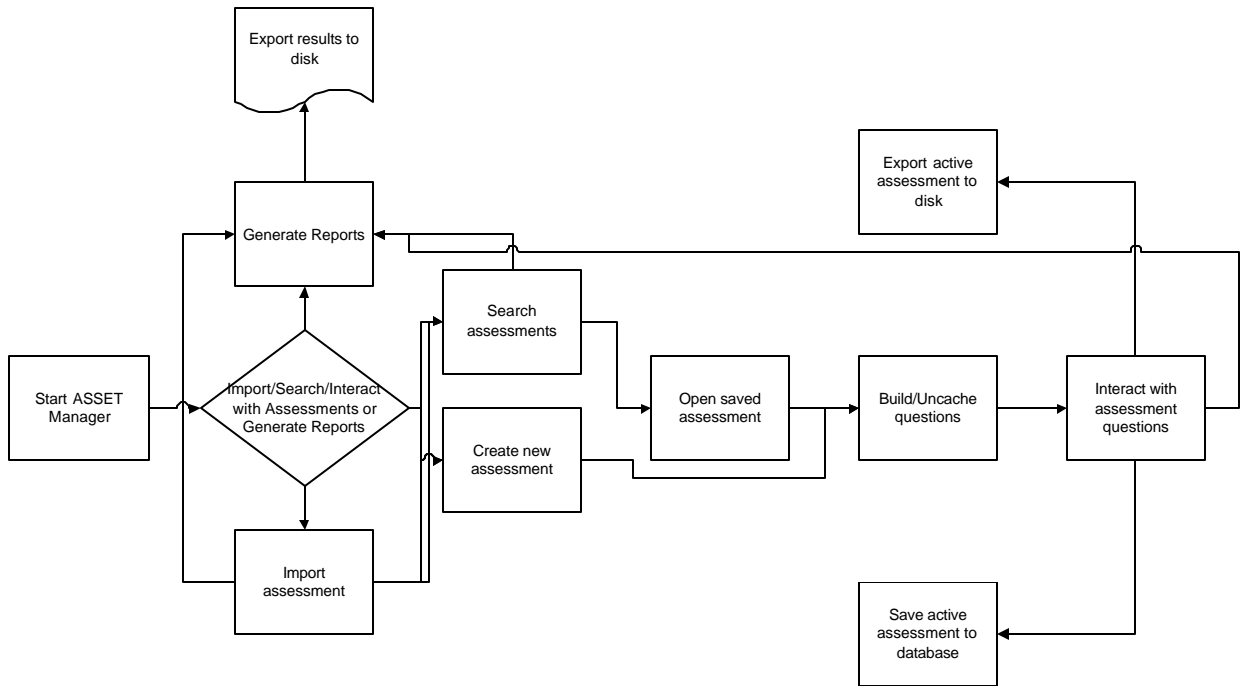


Figure 35 - Manager Procedure Flow

4. ASSET Database

4.1 Database Design

This section discusses the the ASSET database design for System and Manager. It is important to note that the inherent design of the System and Manager database are exactly alike with the exception of the stored procedures. The relational database technology used within ASSET has been implemented by the relational database management system (RDBMS) MSDE 1.0. ASSET communicates with the MSDE through the Java database connectivity open database connectivity (JDBC/ODBC) API version 2.0. The actual SQL statements that implement the ASSET databases are presented in Appendix B of the document. This section discusses why the tables were implemented in the way they were and the method by which the databases are constructed during the ASSET installation process.

Initially to promote a simple deployment process, the ASSET database was intended to be implemented over Microsoft Access. Access does not promote the advanced SQL processing that MSDE does with Transact-SQL (T-SQL). The use of T-SQL permitted the use of stored procedures to perform the most common storage and processing actions of System and Manager during active assessments and reporting. For example, the action of saving an active assessment required saving each of the questions that had been answered. This action would then have to be performed as many times as there are subordinate questions in the checklist. The use of the stored procedure `insert_assessment_question_response` allowed for a significant performance gain over supplying SQL strings to the JDBC/ODBC bridge and on to MSDE each time the question was to be saved. Even with the argument that a sophisticated RDBMS like MSDE can cache the first SQL statement supplied to it and execute each repeated call to the same SQL from its cached version, when System or Manager was closed, this cached execution plan would be lost.

4.2 Extensibility of Question Content

One of the initial requirements of System was the ability for system administrators to add their own questions to the checklist and customize other questions for their own enterprise. The customization of the checklist was intended to be initiated by ASSET Manager who would then publish updated checklist versions, in XML format similar to assessment exports, to each installation of ASSET System within its enterprise. Managers would also have the ability to delete questions from the checklist but only those questions that they added or customized. In a sense the operation would be a rollback instead of delete operation. Readers will note that the `assessment_questions` table contains an integer field **version**. This field is designed to track the version of a question, indicating the number of versions of the same question there are in the database. When Manager publishes a new set of questions to System, System inserts the questions contained in this checklist into the `assessment_questions` table and updates the version number of the current question with the same question number.

Within this version of System and Manager, each time a new assessment was to be created, the questions asked would only reflect a version equal to 0. This would require that the question map be checked against the database each time an assessment was opened/created. This is still a performance benefit over the failed design of the question map as discussed earlier in the document. The difficult question to answer when considering multiple versions of the questions within the checklist was the method in which a user would open a saved assessment which was completed with a version of the checklist that is no longer current. The user of System would have updated the checklist questions after they completed this assessment.

4.3 Assessment Rollups

The “assessment rollup” process describes the operation where output from System is exported to XML format and then imported into Manager. Because most organizations are organized in a hierarchical format, to properly perform a rollup of all of the assessment data within an enterprise, Manager would need some notion to distinguish where assessment data originated.

It was decided that the location identification within Manager would be performed manually in version 1.0 of ASSET. However, as shown in the following snippet of XML from a sample assessment export, future versions of System will be able to identify their place within a distributed organization.

```
<?xml version="1.0"?>
<!--NIST Self-Assessment Tool Auto-generated Assessment Results-->
<!--Generated on: Wed Apr 17 09:10:35 EDT 2002-->
<!--This file contains the results of 4 assessments-->
<assessments>
  <assessment>
    <location>
      blank
    </location>
  </assessment>
  <assessment>
    <location>
      blank
    </location>
  </assessment>
  <assessment>
    <location>
      blank
    </location>
  </assessment>
  <assessment>
    <location>
      blank
    </location>
  </assessment>
</assessments>
</system>
...
```

The location tag included in the example above is meant to indicate the location where this assessment was performed. Future versions of Manager could then be able to perform searches of the Manager databases on this field.

4.4 Dbmanager

This section describes the creation of the System and Manager databases. It was envisioned that these databases would be created and destroyed several times in the life cycle of ASSET on a user’s computer. Because of this, the database creation logic was packaged as a Windows batch script and accessible from every installation of System and Manager within the SQL directory of each component of ASSET. This script is named dbmanager.bat for both System and Manager. Dbmanager uses the osql.exe utility provided with MSDE to interact with the MSDE system. Readers should note that osql.exe communicates with MSDE through ODBC. Because of this dependency on the osql.exe utility, the script needs to know where to locate the script if it is not within the System path.

Dbmanager uses the osql.exe utility in a trusted manner as indicated by the -E command flag. This allows any user to execute the script without having to authenticate to the MSDE system. The following options are available to MSDE:

- Script usage
- Adding new stored procedure
- Deleting stored procedure
- Deleting the entire database
- Creating the entire database (including the stored procedures)

The following syntax shows how to create the System database from the command line:

```
C:\Program Files\NISTASSET\Asset System\sql\>dbmanager createdb
```

The following syntax shows how to clear the entire System database and any saved data:

```
C:\Program Files\NISTASSET\Asset System\sql\>dbmanager deletedb
```

These commands can be executed by ASSET users to restore database integrity by deleting the existing database and then reinstalling it.

NOTE: Users must first save assessment data in the database to a file, since deleting the database will remove all trace of a user's assessment data (stored in the database) from their computer.

The design of Dbmanager allows organizations to customize the ASSET databases to suit their own needs. They can add new stored procedures and change the structure of the ASSET tables as needed. Users simply need to note that this operation must be performed at their own risk.

4.5 MSDE Data Backup

This section addresses the ability of ASSET to utilize the backup procedures provided by the MSDE system. Several users have requested instructions on how to perform this. Officially the backup procedures endorsed by NIST are utilizing System and Manager to export assessment data to XML format. The following URLs describe backup procedures for MSDE:

<http://www.swynk.com/sqlscripts/backuprest7.asp>

<http://www.richmondsys.co.uk/richmond-products/kb/supportdesk/database/DB00000008.htm>

<http://nsupport.elronsoftware.com/support/wi6web.nsf/d6bd51df55a92eb8852566e2005bb596/da17fc1e00f9d26085256b8a007d7692?OpenDocument>

5. Security of ASSET

5.1 Introduction

This section describes the security considerations of ASSET. The security of the systems on which ASSET was installed, and the data that it gathers, have been primary concerns influencing ASSET development from the start. Much of the security of ASSET revolves around the MSDE system. The following Microsoft URL lists vulnerabilities associated with MSDE:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp?productid=31&servicepackid=0&submit1=go&isie=yes>

Components of ASSET such as the ADPU and custom network configurations serve to strengthen the security of the machine on which ASSET is installed. Actual procedures for performing the security options mentioned in this section are detailed in the ASSET user manual and are not discussed here.

5.2 MSDE Network Vulnerability

A default installation of MSDE opens a network port, TCP port 1433 that leaves a machine vulnerable to exploitation. A malicious user can connect to a vulnerable MSDE installation through ODBC, ADO, MDAC, or some other database connectivity API and execute the stored procedure xp_cmdshell. This SP gives MSDE users access to an operating system command shell executing at the permissions of the user account that started the MSDE service sqlservr.exe, which is normally the LocalSystem account. This by itself is not as critical as the notion that Microsoft recommends that if users want SQL Server to be able to use network shares to save data, then they must start the services with an account of higher privilege like the Administrator account.

ASSET mitigates this vulnerability in two ways. First, ADPU application resets the 'sa' password to a user-defined value. The second way is to disable MSDE network communications, closing the TCP port that is opened during the MSDE installation (described in the ASSET user manual). Many "script kiddie" tools that attempt to attack SQL Server and MSDE depend upon a blank 'sa' password. This security process is not foolproof in that the password of the 'sa' account is stored on the hard drive in the ASSET installation directory. Users must work with these security procedures to provide host-based security for their machine. The following output from the nmap port scanner shows TCP port 1433 as being open on a default installation of MSDE.

```
# nmap (V. 2.54BETA15) scan initiated Wed Sep 11 10:21:06 2002 as: nmap -sT -oN msdel.out
10.0.0.26
Interesting ports on CLIENT (10.0.0.26):
(The 1529 ports scanned but not shown below are in state: filtered)
Port      State      Service
13/tcp    open      daytime
37/tcp    open      time
135/tcp   open      unknown
139/tcp   open      unknown
427/tcp   open      unknown
445/tcp   open      unknown
1025/tcp  open      listen
1032/tcp  open      iad3
1433/tcp  open      unknown

# Nmap run completed at Wed Sep 11 10:27:33 2002 -- 1 IP address (1 host up) scanned in 387
seconds
```

The following output from nmap shows TCP port 1433 as closed after executing the instructions for disabling network communications as specified in the ASSET User manual.

```
# nmap (V. 2.54BETA15) scan initiated Wed Sep 11 10:33:38 2002 as: nmap -sT -oN msde2.out 10.0.0.26
Interesting ports on CLIENT (10.0.0.26):
(The 1530 ports scanned but not shown below are in state: filtered)
Port      State      Service
13/tcp    open      daytime
37/tcp    open      time
135/tcp   open      unknown
139/tcp   open      unknown
427/tcp   open      unknown
445/tcp   open      unknown
1025/tcp  open      listen
1032/tcp  open      iad3

# Nmap run completed at Wed Sep 11 10:40:05 2002 -- 1 IP address (1 host up) scanned in 387 seconds
```

5.3 MSDE Service Packs

ASSET has been tested with service pack 3 of MSDE. This did not affect the operation of System or Manager in any adverse manner. After the release of the ASSET tool in June 2002, service pack 4 for MSDE was published by Microsoft. The following URL describes the fixes that service pack 3 provides:

<http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q274797&>

Service pack 3 is a cumulative fix that includes all of the fixes that service pack 1 and 2 provided. If users run vulnerability scanners against a machine with ASSET installed, they will note that the scanner lists several vulnerabilities for the MSDE system. An explanation of these fixes is provided in the Help FAQ for System and Manager. Service pack 4 for MSDE has not been tested yet. The download URL for service pack 4 is as follows:

<http://www.microsoft.com/sql/downloads/sp4.asp>

In addition to MSDE service packs, future version of ASSET will include the MSDE version 2000 instead of version 1.0.

5.4 MSDE Hotfixes

Users that have not installed any of the MSDE service packs or the latest service pack should absolutely consider installing the MSDE hotfixes. The URL presented earlier in this section describes the patch information for MSDE:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp?productid=31&servicepackid=0&submit1=go&isie=yes>

Similar to many other Microsoft products, patches will be continually available as new vulnerabilities are discovered. The Microsoft security site is a suitable download location for future patches for the MSDE system.

6. ASSET Installer and Deployment

6.1 Windows NT Issues

This section addresses installer issues for Windows NT. Even though ASSET was developed on Windows 2000 Professional, it will not run on Windows NT Workstation 4.0 by default. To be specific, it is the MSDE that cannot be installed on NT. The following table of requirements is for users who wish to install ASSET on Windows NT Workstation 4.0:

Component	Download Location
Service Pack 6a (All applicable hotfixes)	http://www.microsoft.com/ntserver/nts/downloads/recommended/SP6/allSP6.asp
Windows Installer 2.0	http://www.microsoft.com/downloads/release.asp?ReleaseID=32832&area=search&ordinal=2
MDAC 2.6.1	http://www.microsoft.com/data/download_270RTM.htm

Table 8 - Windows NT Workstation 4.0 Requirements

NT workstations do not have these components installed by default. If installing ASSET on Windows NT, the items listed above must be installed before attempting to install ASSET. The ASSET installer will fail when trying to install the MSDE without these components installed. Microsoft Office 2000 Premium includes an install of the MSDE on disc 2. If Office 2000 Premium or later is installed, it includes the requirements listed in the previous table and will allow ASSET to install.

NOTE: Deployment testing revealed that Windows NT installations took at least 1-3 minutes longer than Windows 2000 Professional and Windows XP. Users should exercise patience when installing on Windows NT and should not press the Cancel button to attempt to abort. Instead, they should use the uninstallation procedures defined in the ASSET user manual if they are sure the installation has failed.

6.2 Windows 9x Issues

ASSET cannot install on the Windows 9x family of operating systems (e.g., Windows 95/98/ME) due to an incompatibility in the operation of MSDE. MSDE uses TCP/IP to communicate with clients under the 9x kernel because it cannot support named pipes. In theory, if a user was to install MSDE manually and configure the ASSET databases by hand (because the **dbmanager.bat** script will not run under Windows 9x), then they can reconfigure the ASSET configuration files to point to a new ODBC DSN. As discussed in the security section, ASSET uses MSDE configured with named pipes to communicate with clients as shown in the following figure.

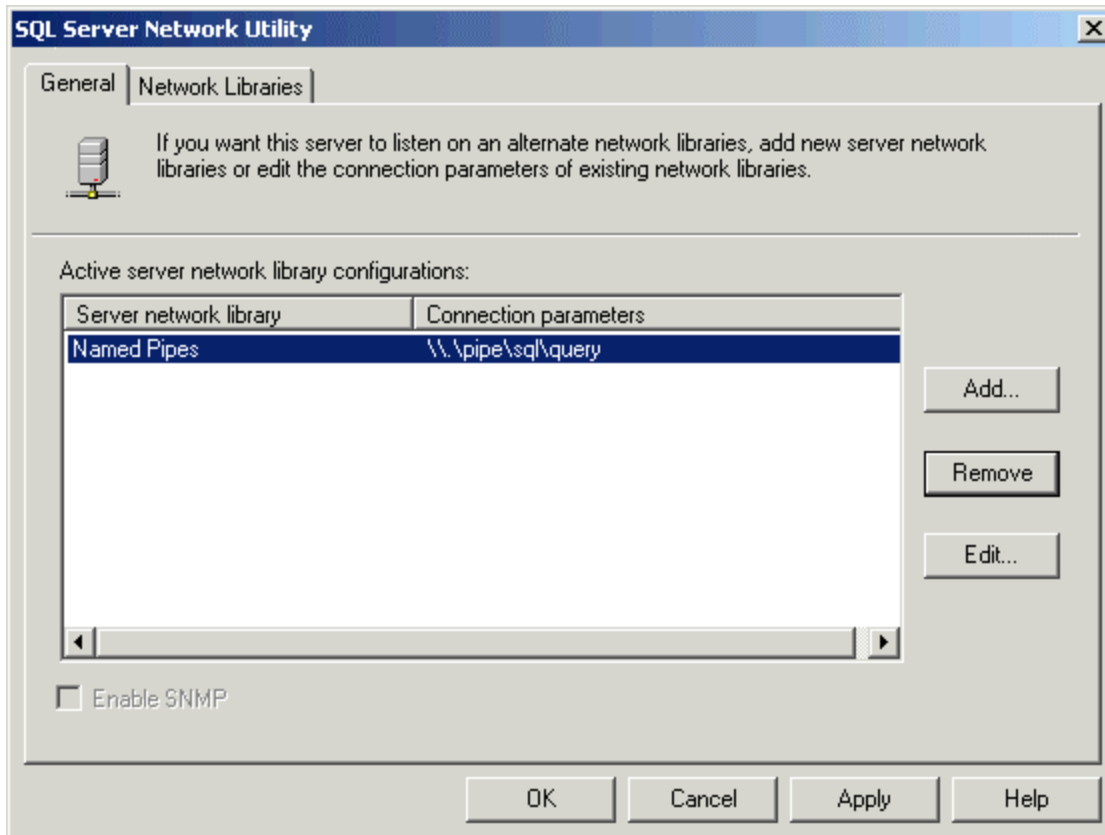


Figure 36 - ASSET MSDE Communications Configuration

6.3 Windows Command Prompt

This section addresses why the Windows command prompt appears when System or Manager is started, but not ADPU. Sun recommends that console applications be configure to start with the virtual machine javaw.exe instead of java.exe. This version of the JVM closes the command window after the application is started. System and Manager use classes from the Xerces and JDOM XML parsers to import and export assessment information. Visual Café's stand-alone archive configuration option (Project Options window) does not import all of the required classes from these JAR files and therefore requires these JARs be accessible at runtime. Using javaw.exe throws ClassNotFound (**sun.tools.java.ClassNotFound**) exceptions for these classes. Because of this, System and Manager are started with batch scripts. The following syntax from **ASSETSystem.bat** starts System:

```
C:\>java -cp ..\logger.jar;asset.jar;xercesImpl.jar;xmlParserAPIs.jar SATUI
```

The following syntax from **ASSETManager.bat** starts Manager:

```
java -cp ..\calendar.jar;..\ASSET System\asset.jar;..\xercesImpl.jar;..\xmlParserAPIs.jar;..\asset_core.jar;asset_admin.jar" SATAdmin →
```

These batch scripts cause the command prompt to remain open after each application is started. In theory, if one was to copy all of the classes from the JAR files listed in the commands above, they could configure the JAR manifest files to launch with javaw instead of java.exe. This would cause the archive

files to be several MB in size. ADPU, on the other hand, does not require **java.exe** to start because all of its required classes are contained within the JAR file ASSETPassword.jar.

NOTE: The main class attribute for ASSETPassword.jar must be set manually each time it is rebuilt because Visual Café does not properly set the main class of the JAR file, even though it may be specified in the project settings window.

Appendix A—Source Code and Class Diagrams

A.1 ASSET System Source Code

The source code for ASSET System, Manager, Core, and Change Password Utility can be found at the NIST ASSET website, <http://csrc.nist.gov/asset>.

A.2 Class Diagrams

This section presents the Unified Modeling Language (UML) class diagrams for System, Manager, and ADPU. No diagram is necessary for the Core project, since Visual Café copies the required contents of Core into System and Manager where appropriate. These class diagrams were created using Object Insight's JVision. These diagrams do not include all subclasses (e.g., those denoted with a \$ in Visual Café), only the most important classes. Each diagram will be presented on a separate page and will not include discussion; they are meant to allow developers to understand how the classes within each project relate to one another.

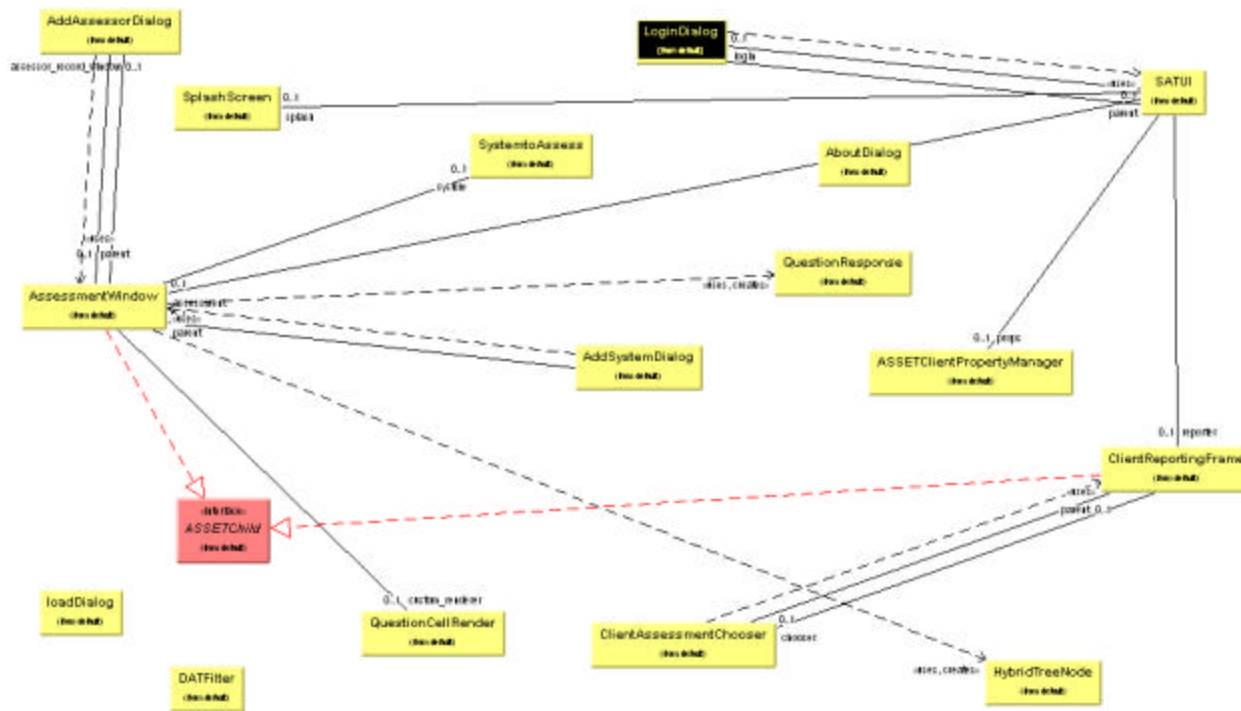


Figure A-1 - System UML Class Diagram

Appendix B—Database

B.1 ER Diagram

This section presents the ER diagram for the System and Manager databases. This ER diagram was created using ERWin 4. The ER diagram in this section does not show the relationship of stored procedures to the Manager or System database, only the referential integrity of the involved tables. The ER diagram is presented on the next page.

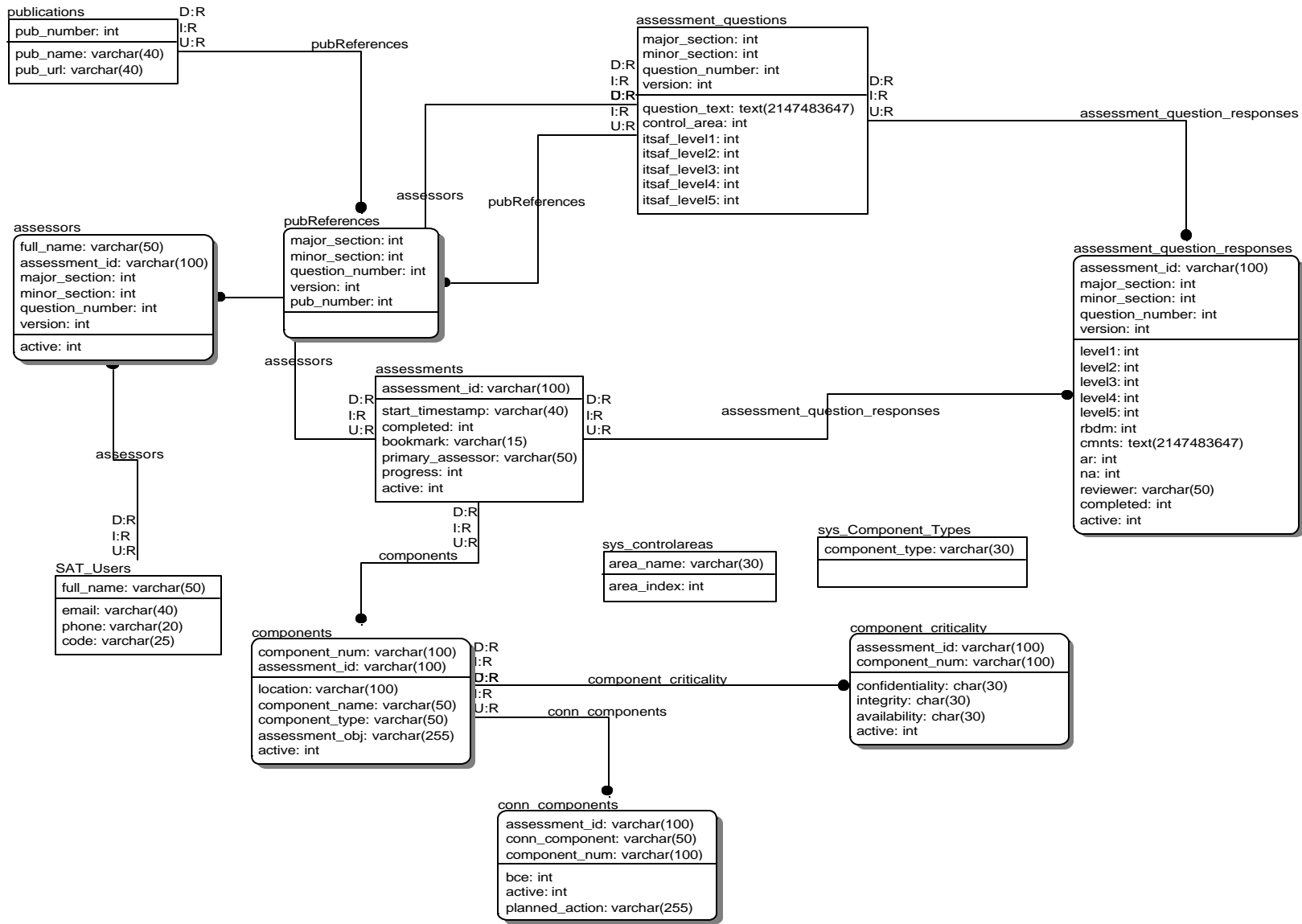


Figure B-1 - ER Diagram for System and Manager

B.2 SQL Source Code

This section presents the SQL statements for the creation of the ASSET databases. The ASSET databases are written using Transact-SQL (T-SQL), the dialect of SQL understood by the MSDE and SQL Server. The statements include both table creation statements and stored procedure creation statements. They will each be presented on a separate page. Many of the SQL scripts included with System also apply to Manager since Manager needs the ability to act as System. Where applicable, if a script has been included with System and it applies to Manager, only a reference to the System copy of the script will be added rather than duplicating the SQL statement twice.

The SQL presented in this section is distributed with ASSET System and Manager, and can be executed after ASSET has been installed if needed. Provided that any assessment information that was entered into System is saved to XML, the System database can be deleted and restored again. For more information, see the dbmanager.bat operation section in the body of the document. The SQL for System and Manager is located in the SQL directory underneath the respective parent directories (ASSET System and ASSET Manager) within the C:\Program Files\NISTASSET install location. The following figure shows a partial listing of the SQL directory contents for System.

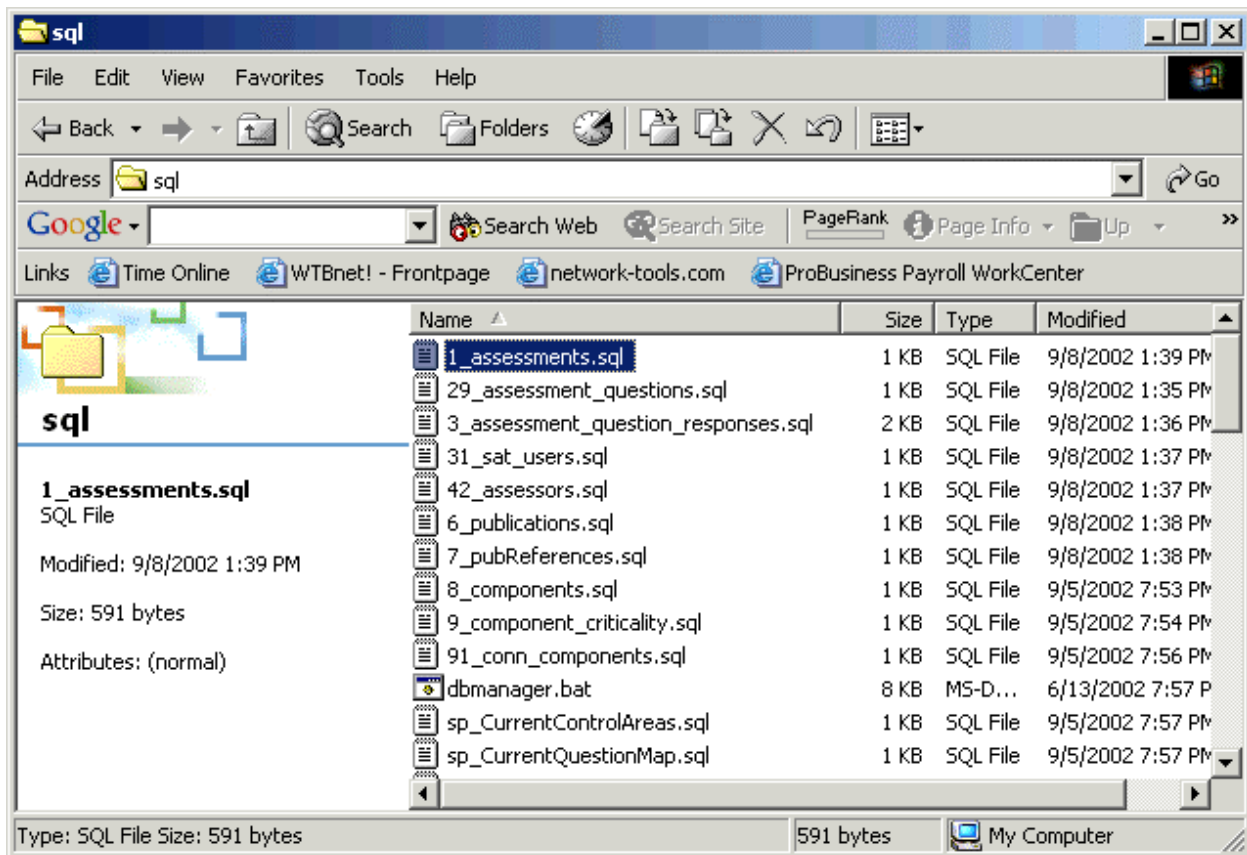


Figure B-2 - System Database Scripts Install Location

B.2.1 ASSET System SQL Table Statements

The following series of SQL statements presents the table creation statements for ASSET System and their corresponding file names within the ASSET distribution. These statements are executed by the

dbmanager script during the installation of System, and if the System database is ever rebuilt, then they are executed by the user from the command line. These files are located in:

C:\ProgramFiles\NISTASSET\ASSETSystem\sql

```
-- This SQL statement creates the assessments table for ASSET System.
-- Note the primary key of this table is the string 'assessment_id'. This
-- field is a concatenation of the System name, primary assessor, and start
-- date of component being assessed through ASSET System.
-- Author - Mark McLarnon
create table assessments (
    assessment_id    varchar(100) unique,
    start_timestamp  varchar(40),
    completed        integer,
    bookmark         varchar(15),
    primary_assessor varchar(50),
    progress         integer,
    primary key (assessment_id)
);
```

Included from file **1_assessments.sql**

```
-- This SQL statement creates the table assessment_questions for the ASSET
-- System database. Note in this case that the question number as listed in
-- the 800-26 checklist is broken into three parts, major_section,
-- minor_section, and question_number. Both the assessment_question_responses
-- table and the assessors table will reference this table through
-- referential integrity.
-- Author - Mark McLarnon
create table assessment_questions (
    major_section      integer,
    minor_section      integer,
    question_number    integer,
    version            integer,
    question_text      text,
    control_area       integer,
    itsaf_level1       integer,
    itsaf_level2       integer,
    itsaf_level3       integer,
    itsaf_level4       integer,
    itsaf_level5       integer,
    primary key (major_section, minor_section, question_number, version)
);
```

Included from file **29_assessment_questions.sql**

```

-- This SQL statement creates the assessment_question_responses table for
-- ASSET System. This table holds a response to a checklist question. THE
-- checklist questions are contained within the assessment_questions table,
-- this table holds a foreign key reference to questions table.
-- The second foreign key is to the assessors table, this table is designed
-- to track who answered what question. Assessors holds a foreign key
-- reference to the SAT_Users table.
-- Author - Mark McLarnon
create table assessment_question_responses (
    assessment_id    varchar(100),
    major_section    integer,
    minor_section    integer,
    question_number  integer,
    version          integer,
    level1          integer,
    level2          integer,
    level3          integer,
    level4          integer,
    level5          integer,
    rbdm            integer,
    cmnts           text,
    ar              integer,
    na              integer,
    reviewer        varchar(50),
    completed       integer
    primary key (assessment_id, major_section, minor_section,
question_number, version),
    constraint asID foreign key (assessment_id) references assessments
(assessment_id),
    constraint qmv1 foreign key (major_section, minor_section,
question_number, version) references assessment_questions (major_section,
minor_section, question_number, version)
);

```

Included from file **3_assessment_question_responses.sql**

```
-- This SQL statement creates the SAT_Users table. This table is
-- referenced by the assessors table. Note that the fields here are
-- all stored as strings, the phone number is meant only as a
-- convenience, not to be used in some type of numerical computation or
-- complex sort.
-- Author - Mark McLarnon
create table SAT_Users (
    full_name varchar(50) not null,
    email      varchar(40),
    phone      varchar(20),
    code       varchar(25),
    primary key (full_name)
);
```

Included from file **31_sat_users.sql**

```
-- This SQL statement creates the assessor table. This table is responsible
-- for storing which assessor answered which question for an active
-- assessment. Note the three foreign key references here to the
-- assessment_questions, SAT_Users table, and the
-- assessment_question_responses table.
-- Author - Mark McLarnon
create table assessors (
    full_name          varchar(50) not null,
    assessment_id      varchar(100),
    major_section      integer,
    minor_section      integer,
    question_number    integer,
    version            integer,
    primary key (full_name, assessment_id, major_section, minor_section,
question_number, version),
    constraint asID2 foreign key (assessment_id) references assessments
(assessment_id),
    constraint email foreign key (full_name) references sat_users
(full_name),
    constraint qmv2 foreign key (major_section, minor_section,
question_number, version) references assessment_questions (major_section,
minor_section, question_number, version)
);
```

Included from file **42_assessors.sql**

```
-- This SQL statement creates the publications table. This table is designed
-- to store a publication reference for a particular reference.
-- In version 1.00 of ASSET, this table is not currently used. It was
-- designed to hold the references that a particular question had to a
-- particular reference source.
-- Author - Mark McLarnon
create table publications (
    pub_number integer not null,
    pub_name   varchar(40),
    pub_url    varchar(40),
    primary key (pub_number)
);
```

Included from file **6_publications.sql**


```
-- This SQL statement creates the pubReferences table. This table holds a 1
-- to many relationship with the publications table. This setup allows a
-- single question to reference more than one publication without violating
-- 3NF.
-- Author - Mark McLarnon
create table pubReferences (
    major_section    integer unique,
    minor_section    integer unique,
    question_number  integer unique,
    version          integer,
    pub_number       integer,
    primary key (major_section,minor_section,version,question_number,
pub_number),
    constraint pNum foreign key pub_number) references publications
(pub_number),
    constraint qmv3foreign key (major_section, minor_section,
question_number, version)references assessment_questions (major_section,
minor_section, question_number, version)
);
```

Included from file **7_pubReferences.sql**

```
-- Purpose - ASSET System database script
-- This SQL script creates the components table. The context is
-- straightforward except for the component_num field. This field is designed
-- to hold the contents of the Component Number field from the ASSET System
-- user interface. Initially this field was a number but was changed to allow
-- alphanumeric characters. To make this change, 9 edits to the SQL scripts
-- were required.
-- Author - Mark McLarnon
create table components (
    component_num varchar(100),
    assessment_id varchar(100),
    location      varchar(100),
    component_name varchar(50),
    component_type varchar(50),
    assessment_obj varchar(255),
    primary key (component_num, assessment_id),
    constraint asID4 foreign key (assessment_id) references assessments
(assessment_id)
);
```

Included from file **8_components.sql**

```
-- This SQL statement creates the component_criticality table. This table
-- references the components table with the asID9 foreign key reference.
-- Author - Mark McLarnon
create table component_criticality (
    assessment_id      varchar(100),
    component_num      varchar(100),
    confidentiality    char(30),
    integrity          char(30),
    availability       char(30),
    primary key (component_num, assessment_id),
    constraint asID9 foreign key (component_num, assessment_id) references
components(component_num, assessment_id),
);
```

Included from file **9_component_criticality.sql**

```
-- This SQL statement creates the conn_components table. This table holds a
-- foreign key reference to the components table through the asID14 foreign
-- key.
-- Author - Mark McLarnon
create table conn_components (
    assessment_id      varchar(100),
    component_num      varchar(100),
    conn_component     varchar(50),
    bce                integer,
    planned_action     varchar(255),
    primary key (assessment_id, component_num, conn_component),
    constraint asID14 foreign key (component_num, assessment_id) references
components (component_num, assessment_id),
);
```

Included from file **91_conn_components.sql**

```
-- This table holds the component types for when a user starts a new
-- assessment and has to choose what type of component it is.
-- Author - Mark McLarnon
create table sys_Component_Types (
    component_type varchar(30),
    primary key (component_type)
);
```

Included from file **sys_Component_types.sql**

```
-- This table holds the control areas for the checklist. It is used to build
-- the question map because on the map, the questions are organized by
control -- area
-- Author - Mark McLarnon
create table sys_controlareas (
    area_name    varchar(30) unique,
    area_index   integer,
    primary key (area_name)
);
```

Included from file **sys_controlareas.sql**

B.2.2 ASSET System SQL Stored Procedure Statements

This section lists the stored procedures (SPs) from System exactly as contained within the database creation scripts. Both System and Manager rely heavily on stored procedures for their performance benefits over supplying SQL strings through JDBC to MSDE. The use of stored procedures also promotes the extensibility of ASSET to work on alternate RDBMS platforms because no platform-specific SQL code is contained within the ASSET source. These files are found in the following directory:

C:\Program Files\NISTASSET\ASSET System\sql

```
-- Stored Procedure - current_control_areas
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the current control areas from
-- the sys_controlareas table within System. It is called in the
-- rebuildMap() method of System to rebuild the System question map when
-- needed.
-- Author - Mark McLarnon
create procedure current_control_areas AS
SELECT sys_controlareas.area_name, sys_controlareas.area_index
FROM sys_controlareas
GO
```

Included from file **sp_CurrentControlArea.sql**


```

-- Stored Procedure - current_question_map
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts all of the relevant question info
-- from the assessment_questions table to build the question map. The reason
-- that the question text is extracted as well is that when a new question is
-- to be displayed, while System is running, it simply asks the data
-- structure that holds the map contents what the new question number and
-- text are rather than having to call the database each time.
--
-- The variable cntrlarea is a string that represents the current control
-- area being mapped. This means that this stored procedure is only
-- called as many times as there are control areas.
-- Author - Mark McLarnon
create procedure current_question_map
(
    @cntrlarea varchar(25)
)
AS
SELECT
    assessment_questions.major_section,
    assessment_questions.minor_section,
    assessment_questions.question_number,
    assessment_questions.question_text,
    assessment_questions.itsaf_level1,
    assessment_questions.itsaf_level2,
    assessment_questions.itsaf_level3,
    assessment_questions.itsaf_level4,
    assessment_questions.itsaf_level5
FROM
    assessment_questions
WHERE
    ((
        (
            assessment_questions.control_area
        )
    )
    =
        @cntrlarea))
GO

```

Included from file **sp_CurrentControlAreas.sql**

```
-- Stored Procedure - current_question_set
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts all of the relevant question info
-- from the question table where the version number matches 0. Although the
-- version flag is not used in version 1.00 of ASSET, it's intent is to only
-- extract questions that match version 0 each time, these are the most
-- current versions of the questions in the checklist. The intent for future
-- releases of ASSET is to allow users to add new questions to the checklist
-- using ASSET Manager.
-- Author - Mark McLarnon
create procedure current_question_set
AS
SELECT assessment_questions.*
FROM assessment_questions
WHERE assessment_questions.version = 0
GO
```

Included from file **sp_CurrentQuestionSet.sql**

```

-- Stored Procedure - delete_saved_question
-- ASSET Component - ASSET System
-- Purpose - This stored procedure is called when a user presses the clear
-- button within ASSET System for a question that they have previously
-- answered. If they have saved their progress to the database, then without
-- this information although the answer in memory is reset to empty, when the
-- database was loaded again, it would extract the old answer from the
-- assessment_question_responses table and list that as the correct response
-- for this supposedly deleted question.
--
-- The assessment_id variable dictates which assessment the question
-- corresponds to. The remaining three variables, major_section,
-- minor_section, and question_number, compose the three-part question number
-- as listed in the NIST Special Pub 800-26.
-- Author - Mark McLarnon
create procedure delete_saved_question
(
    @assessment_id    varchar(100),
    @major_section    integer,
    @minor_section    integer,
    @question_number  integer
)
AS
IF (EXISTS (SELECT major_section, minor_section, question_number FROM
assessment_question_responses WHERE assessment_id = @assessment_id))
BEGIN
    DELETE from
        assessment_question_responses
    WHERE
        assessment_id = @assessment_id
    AND
        major_section = @major_section
    AND
        minor_section = @minor_section
    AND
        question_number = @question_number
END
GO

```

Included from file **sp_DeleteSavedQuestion.sql**

```

-- Stored Procedure - get_all_component_information
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts all of the relevant component
-- info from both the components table and the component_criticality table.
--
-- The variable component_number is used to identify which component info to
-- extract from the components table. The variable assessment_id is used to
-- extract only the component info that corresponds to a particular
-- assessment. This operates under the theory that if the same component is
-- assessed twice, the information could change between the time of the two
-- assessments.
-- Author - Mark McLarnon
create procedure get_all_component_information
(
    @component_num    varchar(100),
    @assessment_id    varchar(100)
)
AS
SELECT
    *
FROM
    components
INNER JOIN
    component_criticality
ON
    components.component_num = component_criticality.component_num
AND
    components.assessment_id = component_criticality.assessment_id
WHERE
    components.component_num = @component_num
AND
    components.assessment_id = @assessment_id
GO

```

Included from file **sp_getAllComponentInformation.sql**

```
-- Stored Procedure - get_assessorreviewers_per_assessment
-- ASSET Component - ASSET System
-- Purpose - This stored procedure is used to extract all of the assessors
-- that contributed to a specific assessment. This information is used to
-- populate the Assign to Alternate Reviewer box on the question tab in
-- System. The variable assessment_id indicates which assessment to extract
-- assessors for.
-- Author - Mark McLarnon
create procedure get_assessorreviewers_per_assessment
(
@assessment_id    varchar(100)
)
AS
SELECT
    DISTINCT(SAT_Users.full_name),
    SAT_Users.email,
    SAT_Users.phone
FROM
    SAT_Users
INNER JOIN
    assessment_question_responses
ON
    SAT_Users.full_name = assessment_question_responses.reviewer
WHERE
    assessment_question_responses.assessment_id = @assessment_id
GO
```

Included from file **sp_getAssessorReviewersPerAssessment.sql**

```

-- Stored Procedure - get_assessors_per_assessment
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the assessors from the SAT_Users
-- table inner-joined with the assessors table to fill the Answered By choice
-- box on the question tab within System.
--
-- The variable assessment_id is used to determine which assessment to
-- extract assessors information for.
-- Author - Mark McLarnon
create procedure get_assessors_per_assessment
(
@assessment_id    varchar(100)
)
AS
SELECT
    DISTINCT(SAT_Users.full_name),
    SAT_Users.email,
    SAT_Users.phone
FROM
    assessors
INNER JOIN
    SAT_Users
ON
    SAT_Users.full_name=assessors.full_name
WHERE
    assessors.assessment_id = @assessment_id
GO

```

Included from file **sp_getAssessorsPerAssessment.sql**

```
-- Stored Procedure - get_component_criticalities
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the component criticality choices
-- from the component_criticality table for a specific component and a
-- specific assessment. It is used when an assessment is opened again as its
-- values set the Confidentiality, Integrity, Availability drop-down boxes on
the System
-- identification tab.
-- Author - Mark McLarnon
create procedure get_component_criticalities
(
    @component_num    varchar(100),
    @assessment_id    varchar(100)
)
AS
SELECT
    *
FROM
    component_criticality
WHERE
    component_num = @component_num
AND
    assessment_id = @assessment_id
GO
```

Included from file **sp_getComponentCriticalities.sql**

```
-- Stored Procedure - get_conn_systems
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the connected systems for a
-- particular System. It is used when the systems assessment information is
-- loaded from the database.
-- Author - Mark McLarnon
create procedure get_conn_systems
AS
SELECT
    DISTINCT component_num,
    conn_component,
    bce,
    planned_action
FROM conn_components
GO
```

Included from file **sp_getConnectedSystems.sql**


```
-- Stored Procedure - get_conn_systems_per_assessment
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the connected systems for a
-- particular assessment. It inner-joins the components and conn_components
-- table to do so using the variables component_num (which component are they
-- connected to) and assessment_id
-- (for which assessment).
-- Author - Mark McLarnon
create procedure get_conn_systems_per_assessment
(
    @component_num    varchar(100),
    @assessment_id    varchar(100)
)
AS
SELECT
    *
FROM
    conn_components
WHERE
    component_num = @component_num
AND
    assessment_id = @assessment_id
GO
```

Included from file **sp_getConnectedSystemsPerAssessment.sql**

```
-- Stored Procedure - get_critical_listing
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the critical element question
-- response from the assessment_question_reponses table for a particular
-- assesment. Recall that these responses are computed automatically.
-- Author - Mark McLarnon
create procedure get_critical_listing
(
    @assessment_id    varchar(100)
)
AS
SELECT
    major_section,
    minor_section,
    question_number
FROM
    assessment_question_responses
WHERE NOT
    minor_section = 0
AND
    question_number = 0
AND
    assessment_id = @assessment_id
GO
```

Included from file **sp_getCriticalListing.sql**

```
-- Stored Procedure - get_critical_results
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the critical element question
-- response from the assessment_question_reponses table for a particular
-- assessment. Recall that these responses are computed automatically.
-- It is similar to the get_critical_listing SP except this one extracts all
-- of the information instead of just the question number.
-- Author - Mark McLarnon
create procedure get_critical_results
(
    @assessment_id    varchar(100)
)
AS
SELECT *
FROM
    assessment_question_responses
WHERE NOT
    minor_section = 0
AND
    question_number = 0
AND
    assessment_id = @assessment_id
GO
```

Included from file **sp_getCriticalResults.sql**

```

-- Stored Procedure - get_na_questions
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the Non-Applicable questions for
-- a particular assessment. It is used to generate one of the reports for
-- System.
-- Author - Mark McLarnon
create procedure get_na_questions
(
    @assessment_id    varchar(100)
)
AS
SELECT
    assessment_questions.question_text,
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number,
    cmnts
FROM
    assessment_questions
    INNER JOIN
    assessment_question_responses
    ON
        assessment_question_responses.major_section =
assessment_questions.major_section
    AND
        assessment_question_responses.minor_section =
assessment_questions.minor_section
    AND
        assessment_question_responses.question_number =
assessment_questions.question_number
WHERE
    assessment_question_responses.assessment_id=@assessment_id
AND
    assessment_question_responses.na = 1
AND NOT
    assessment_question_responses.question_number = 0
ORDER BY
    assessment_question_responses.major_section
GO

```

Included from file **sp_getNAQuestions.sql**

```
-- Stored Procedure - get_policy_results
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the policy results from the
-- database for a particular assessment. All that this really corresponds to
-- is the questions within the assessment_question_responses table that have
-- a minor number of 0 and a question number of 0 and for which level 1 was
-- marked as 1 (achieved policy).
-- Author - Mark McLarnon
create procedure get_policy_results
(
    @assessment_id    varchar(100)
)
AS
SELECT *
FROM assessment_question_responses
WHERE minor_section = 0 AND question_number = 0 AND assessment_id =
@assessment_id
GO
```

Included from file **sp_getPolicyResults.sql**

```

-- Stored Procedure - get_policy_results_and_text
-- ASSET Component - ASSET System
-- Purpose - This procedure extracts the policy questions from the
-- assessment_question_responses table including the text of the question. It
-- is used to generate a report for System.
-- Author - Mark McLarnon
create procedure get_policy_results_and_text
(
    @assessment_id    varchar(100)
)
AS
SELECT
    assessment_questions.question_text,
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number,
    assessment_question_responses.level1
FROM
    assessment_questions
    INNER JOIN
    assessment_question_responses
    ON
        assessment_question_responses.major_section =
assessment_questions.major_section
    AND
        assessment_question_responses.minor_section =
assessment_questions.minor_section
    AND
        assessment_question_responses.question_number =
assessment_questions.question_number
WHERE
    assessment_question_responses.assessment_id=@assessment_id
AND
    assessment_question_responses.minor_section = 0
AND
    assessment_question_responses.question_number = 0
GO

```

Included from file **sp_getPolicyResultsQuestionText.sql**

```
-- Stored Procedure - get_primary_assessor
-- ASSET Component - ASSET System
-- Purpose - This SP extracts the primary assessor for a particular
-- assessment; it is used to load a particular assessment.
-- Author - Mark McLarnon
create procedure get_primary_assessor
(
  @assessment_id    varchar(100)
)
AS
SELECT DISTINCT(SAT_Users.full_name), SAT_Users.email, SAT_Users.phone
FROM
  assessments
INNER JOIN
  SAT_Users
ON
  SAT_Users.full_name = assessments.primary_assessor
WHERE
  assessments.assessment_id = @assessment_id
GO
```

Included from file **sp_getPrimaryAssessor.sql**

```
-- Stored Procedure - get_question_count
-- ASSET Component - ASSET System
-- Purpose - This procedure extracts the number of questions from the
-- database. It is used to compute the progress for a particular assessment.
-- Author - Mark McLarnon
create procedure get_question_count
AS

SELECT count(*) AS QuestionCount FROM assessment_questions WHERE
    (
        assessment_questions.minor_section = 0
        AND
        assessment_questions.question_number = 0
    )
OR
    (
        NOT assessment_questions.question_number = 0
    )
GO
```

Included from file **sp_GetQuestionCount.sql**


```

-- Stored Procedure - get_question_results
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the question results information
-- for a particular question. Note that it inner-joins the
-- assessment_question_responses table with the assessors table to also
-- extract which assessor answered the question.
-- Author - Mark McLarnon
create procedure get_question_results
(
    @assessment_id    varchar(100)
)
AS
SELECT
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number,
    level1,
    level2,
    level3,
    level4,
    level5,
    rbdm,
    cmnts,
    ar,
    na,
    reviewer,
    completed,
    assessors.full_name
FROM
    assessment_question_responses
    INNER JOIN
        assessors
    ON
        assessment_question_responses.assessment_id =
assessors.assessment_id
        AND
        assessment_question_responses.major_section =
assessors.major_section
        AND
        assessment_question_responses.minor_section =
assessors.minor_section
        AND
        assessment_question_responses.question_number =
assessors.question_number
WHERE
    assessment_question_responses.assessment_id = @assessment_id
GO

```

Included from file **sp_getQuestionResults.sql**

```

-- Stored Procedure - get_question_results_per_policy
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the question result information
-- for a set of questions based on which parent policy item it resides under.
-- It is used to generate a report for System.
-- Author - Mark McLarnon
create procedure get_question_results_per_policy
(
    @assessment_id    varchar(100),
    @critical_major   integer
)
AS
SELECT
    assessment_question_responses.major_section AS major_section,
    assessment_question_responses.minor_section AS minor_section,
    assessment_question_responses.question_number AS question_number,
    assessment_question_responses.level1 AS level1,
    assessment_question_responses.level2 AS level2,
    assessment_question_responses.level3 AS level3,
    assessment_question_responses.level4 AS level4,
    assessment_question_responses.level5 AS level5,
    na
FROM
    assessment_question_responses
WHERE
    assessment_question_responses.assessment_id = @assessment_id
AND
    assessment_question_responses.major_section = @critical_major
AND NOT
    assessment_question_responses.minor_section = 0
AND NOT
    assessment_question_responses.question_number = 0
GO

```

Included from file **sp_getQuestionResultsPerPolicy.sql**

```

-- Stored Procedure - get_question_results_per_section
-- ASSET Component - ASSET System
-- Purpose - This question extracts the subordinate question results from
-- the assessment_question_responses table for a particular critical
-- element. This stored procedure is used to load an assessment from the
-- database.
-- Author - Mark McLarnon
create procedure get_question_results_per_section
(
    @assessment_id    varchar(100),
    @critical_major   integer,
    @critical_minor   integer,
    @critical_number  integer
)
AS
SELECT
    assessment_question_responses.major_section AS major_section,
    assessment_question_responses.minor_section AS minor_section,
    assessment_question_responses.question_number AS question_number,
    assessment_question_responses.level1 AS level1,
    assessment_question_responses.level2 AS level2,
    assessment_question_responses.level3 AS level3,
    assessment_question_responses.level4 AS level4,
    assessment_question_responses.level5 AS level5,
    rbdm,
    cmnts,
    ar,
    na,
    reviewer,
    completed,
    assessors.full_name AS full_name
FROM
    assessment_question_responses
    INNER JOIN
        assessors
    ON
        assessment_question_responses.assessment_id =
    assessors.assessment_id
    AND
        assessment_question_responses.major_section =
    assessors.major_section
    AND
        assessment_question_responses.minor_section =
    assessors.minor_section
    AND
        assessment_question_responses.question_number =
    assessors.question_number
WHERE
    assessment_question_responses.assessment_id = @assessment_id
    AND
    assessment_question_responses.major_section = @critical_major
    AND
    assessment_question_responses.minor_section = @critical_minor
GO

```

Included from file **sp_getQuestionResultsPerSection.sql**

```

-- Stored Procedure - get_rbdm_questions
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the Risk-Based Decision questions
-- for a particular assessment. It is used to generate one of the reports for
-- System.
-- Author - Mark McLarnon
create procedure get_rbdm_questions
(
    @assessment_id    varchar(100)
)
AS
SELECT
    assessment_questions.question_text,
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number,
    cmnts
FROM
    assessment_questions
    INNER JOIN
    assessment_question_responses
    ON
        assessment_question_responses.major_section =
assessment_questions.major_section
    AND
        assessment_question_responses.minor_section =
assessment_questions.minor_section
    AND
        assessment_question_responses.question_number =
assessment_questions.question_number
WHERE
    assessment_question_responses.assessment_id=@assessment_id
AND
    assessment_question_responses.rbdm = 1
AND NOT
    assessment_question_responses.question_number = 0
GO

```

Included from file **sp_getRBDMQuestions.sql**

```
-- Stored Procedure - get_saved_assessments
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts a list of the saved assessments
-- from the database table assessments. It is used to populate the load
-- assessment dialog.
-- Author - Mark McLarnon
create procedure get_saved_assessments
AS
SELECT
    components.component_name AS component_name,
    components.component_num AS component_num,
    components.location AS component_location,
    components.component_type AS component_type,
    components.assessment_obj AS component_assessment_obj,
    assessments.assessment_id AS id,
    assessments.start_timestamp,
    assessments.completed,
    assessments.bookmark,
    assessments.primary_assessor
FROM
    assessments
INNER JOIN
    components
ON
    assessments.assessment_id=components.assessment_id
ORDER BY components.component_name
GO
```

Included from file **sp_GetSavedAssessments.sql**

```
-- Stored Procedure - get_connected_systems
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the connected system for a
-- particular system from the get_connected_systems table. It is used during
-- the assessment load process.
-- Author - Mark McLarnon
create procedure get_connected_systems
(
    @component_num    varchar(100)
)
AS
SELECT *
FROM conn_components
WHERE component_num = @component_num
GO
```

Included from file **sp_getSingleConnectedSystems.sql**

```

-- Stored Procedure - get_single_saved_assessments
-- ASSET Component - ASSET System
-- Purpose - This stored procedure extracts the information for a particular
-- assessment based on a component name and assessment start time. It is used
-- during the assessment export process.
-- Author - Mark McLarnon
create procedure get_single_saved_assessment
(
    @component_name    varchar(50),
    @start_timestamp   varchar(40)
)
AS
SELECT
    components.component_name AS component_name,
    components.component_num AS component_num,
    components.location AS component_location,
    components.component_type AS component_type,
    components.assessment_obj AS component_assessment_obj,
    assessments.assessment_id AS assessment_id,
    assessments.start_timestamp AS start_timestamp,
    assessments.completed, assessments.bookmark,
    assessments.primary_assessor,
    assessments.progress
FROM
    assessments
INNER JOIN
    components
ON
    assessments.assessment_id=components.assessment_id
WHERE
    components.component_name = @component_name
AND
    assessments.start_timestamp = @start_timestamp
ORDER BY components.component_name
GO

```

Included from file **sp_GetSingleSavedAssessment.sql**

```

-- Stored Procedure - get_single_saved_assessment_by_id
-- ASSET Component - ASSET System
-- Purpose - This procedure extracts all of the assessment information
-- for a particular component based on the assessment_id variable.
-- Author - Mark McLarnon
create procedure get_single_saved_assessment_by_id
(
    @assessment_id    varchar(100)
)
AS
SELECT
    components.component_name AS component_name,
    components.component_num AS component_num,
    components.location AS component_location,
    components.component_type AS component_type,
    components.assessment_obj AS component_assessment_obj,
    assessments.assessment_id,
    assessments.start_timestamp,
    assessments.completed,
    assessments.bookmark,
    assessments.primary_assessor,
    assessments.progress
FROM
    components
INNER JOIN
    assessments
ON
    components.assessment_id = assessments.assessment_id
WHERE
    assessments.assessment_id = @assessment_id
GO

```

Included from file **sp_GetSingleSavedAssessmentByAssessmentID.sql**


```
-- Stored Procedure - get_stored_assessors
-- ASSET Component - ASSET System
-- Purpose - This SP extracts the user information from the SAT_Users table.
-- It is used in the add_assessor dialog for existing assessors.
-- Author - Mark McLarnon
create procedure get_stored_assessors
AS
SELECT SAT_Users.full_name, SAT_Users.email, SAT_Users.phone, SAT_Users.code
FROM SAT_Users
GO
```

Included from file **sp_GetStoredAssessors.sql**

```
-- Stored Procedure - get_submitted_question_count
-- ASSET Component - ASSET System
-- Purpose - This SP extracts the question count for a particular assessment
-- by assessment_id.
-- Author - Mark McLarnon
create procedure get_submitted_question_count
(
    @assessment_id    varchar(100)
)
AS
SELECT count (major_section) AS 'count' FROM assessment_question_responses
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_getSubmittedQuestionCount.sql**

```

-- Stored Procedure - get_summary_question_results
-- ASSET Component - ASSET System
-- Purpose - This SP extracts the question response information for a
-- particular question for a particular assessment. It inner-joins the
-- assessment_question, assessment_question_responses, and assessors table.
-- Author - Mark McLarnon
create procedure get_summary_question_results
(
    @assessment_id    varchar(100)
)
AS
SELECT
    assessment_questions.question_text,
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number,
    level1,
    level2,
    level3,
    level4,
    level5,
    rbdm,
    cmnts,
    ar,
    na,
    reviewer,
    completed,
    assessors.full_name
FROM
(
    assessment_questions
INNER JOIN
    assessment_question_responses
ON
    assessment_question_responses.major_section =
assessment_questions.major_section
AND
    assessment_question_responses.minor_section =
assessment_questions.minor_section
AND
    assessment_question_responses.question_number =
assessment_questions.question_number
)
INNER JOIN
    assessors
ON
    assessment_question_responses.assessment_id = assessors.assessment_id
AND
    assessment_question_responses.major_section = assessors.major_section
AND
    assessment_question_responses.minor_section = assessors.minor_section
AND
    assessment_question_responses.question_number =
assessors.question_number

WHERE
    assessment_question_responses.assessment_id = @assessment_id

```

```
ORDER BY
    assessment_question_responses.major_section,
    assessment_question_responses.minor_section,
    assessment_question_responses.question_number
GO
```

Included from file **sp_getSummaryQuestionResults.sql**

```

-- Stored Procedure - insert_assessment_question_response
-- ASSET Component - ASSET System
-- Purpose - This SP saves the assessment question response for a single
-- question during a particular assessment. It is called once for each
-- question that is saved.
-- Author - Mark McLarnon
create procedure insert_assessment_question_response
(
@assessment_id    varchar(100),
@major_section    integer,
@minor_section    integer,
@question_number  integer,
@version          integer,
@level1          integer,
@level2          integer,
@level3          integer,
@level4          integer,
@level5          integer,
@rbdm            integer,
@cmnts           text,
@ar              integer,
@na              integer,
@reviewer        varchar(50),
@completed       integer
)
AS
IF (NOT EXISTS (SELECT * FROM assessment_question_responses WHERE
assessment_id = @assessment_id AND major_section = @major_section AND
minor_section = @minor_section AND question_number = @question_number))
BEGIN
    INSERT INTO assessment_question_responses
    (
        [assessment_id],
        [major_section],
        [minor_section],
        [question_number],
        [version],
        [level1],
        [level2],
        [level3],
        [level4],
        [level5],
        [rbdm],
        [cmnts],
        [ar],
        [na],
        [reviewer],
        [completed]
    )
VALUES
(
    @assessment_id,
    @major_section,
    @minor_section,
    @question_number,
    @version,
    @level1,

```

```

        @level2,
        @level3,
        @level4,
        @level5,
        @rbdm,
        @cmnts,
        @ar,
        @na,
        @reviewer,
        @completed
    )
END
ELSE
    UPDATE assessment_question_responses
    SET level1 = @level1,
        level2 = @level2,
        level3 = @level3,
        level4 = @level4,
        level5 = @level5,
        rbdm = @rbdm,
        cmnts = @cmnts,
        ar = @ar,
        na = @na,
        reviewer = @reviewer,
        completed= @completed
    WHERE assessment_id = @assessment_id
    AND major_section = @major_section
    AND minor_section = @minor_section
    AND question_number= @question_number
GO

```

Included from file **sp_InsertAssessmentQuestionResponse.sql**

```

-- Stored Procedure - save_answered_question
-- ASSET Component - ASSET System
-- Purpose - This SP saves accountability information for a question, who
-- answered the question drawn from the SAT_Users table.
-- Author - Mark McLarnon
create procedure save_answered_question
(
    @full_name          varchar(50),
    @assessment_id     varchar(100),
    @major_section     integer,
    @minor_section     integer,
    @question_number   integer,
    @version           integer
)
AS
IF (NOT EXISTS (SELECT * from assessors where assessment_id = @assessment_id
AND major_section = @major_section AND minor_section = @minor_section AND
question_number = @question_number AND version = @version))
BEGIN
    INSERT INTO assessors
    (
        [full_name],
        [assessment_id],
        [major_section],
        [minor_section],
        [question_number],
        [version]
    )VALUES
    (
        @full_name,
        @assessment_id,
        @major_section,
        @minor_section,
        @question_number,
        @version
    )
END
ELSE
    UPDATE assessors
        SET full_name = @full_name,
            version = @version
    WHERE assessment_id = @assessment_id
    AND major_section = @major_section
    AND minor_section = @minor_section
    AND question_number = @version
GO

```

Included from file **sp_SaveAnsweredQuestion.sql**

```

-- Stored Procedure - save_assessment
-- ASSET Component - ASSET System
-- Purpose - This SP saves assessment identification for a particular
-- assessment. It is called during the question map construction procedure,
-- after the assessment id process is complete.
--
-- NOTE the IF/THEN conditional it checks for the existence of the assessment
-- information before being saved to the database. If the information exists,
-- then it is merely updated instead of a new INSERT statement.
-- Author - Mark McLarnon
create procedure save_assessment
(
@assessment_id      varchar(100),
@start_timestamp    varchar(40),
@completed          integer,
@bookmark           varchar(15),
@primary_assessor   varchar(50),
@progress           integer
)

AS

IF (NOT EXISTS (SELECT * from assessments WHERE assessment_id =
@assessment_id))
BEGIN
    INSERT INTO assessments
    (
        [assessment_id],
        [start_timestamp],
        [completed],
        [bookmark],
        [primary_assessor],
        [progress]
    )
    VALUES
    (
        @assessment_id,
        @start_timestamp,
        @completed,
        @bookmark,
        @primary_assessor,
        @progress
    )
END
ELSE
    UPDATE assessments
    SET completed      = @completed,
        bookmark       = @bookmark,
        progress       = @progress
    WHERE assessment_id = @assessment_id
GO

```

Included from file **sp_SaveAssessment.sql**


```
-- Stored Procedure - save_bookmark
-- ASSET Component - ASSET System
-- Purpose - This SP saves the bookmark information for a particular
-- assessment. It is a string that represents the question that a user will
-- proceed to next.
-- Author - Mark McLarnon
create procedure save_bookmark
(
@bkmark          varchar(15),
@assessment_id   varchar(100)
)
AS
UPDATE assessments SET bookmark = @bkmark
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_SaveBookmark.sql**

```

-- Stored Procedure - save_component
-- ASSET Component - ASSET System
-- Purpose - This SP saves component information for a particular assessment.
-- It is called when the assessment identification process completes.
--
-- See SP save_assessment for explanation of IF/THEN conditional.
-- Author - Mark McLarnon
create procedure save_component
(
@component_num          varchar(100),
@assessment_id          varchar(100),
@location               varchar(100),
@component_name         varchar(50),
@component_type         varchar(50),
@assessment_obj         varchar(255)
)
AS
IF (NOT EXISTS (SELECT * from components WHERE component_num = @component_num
AND assessment_id = @assessment_id))
BEGIN
    INSERT INTO components
    (
        [component_num],
        [assessment_id],
        [location],
        [component_name],
        [component_type],
        [assessment_obj]
    )
VALUES
(
    @component_num,
    @assessment_id,
    @location,
    @component_name,
    @component_type,
    @assessment_obj
)
END
ELSE
    UPDATE components
    SET location = @location,
        component_name = @component_name,
        component_type = @component_type,
        assessment_obj = @assessment_obj
    WHERE component_num = @component_num
    AND assessment_id = @assessment_id
GO

```

Included from file **sp_SaveComponent.sql**

```

-- Stored Procedure - save_component_criticalities
-- ASSET Component - ASSET System
-- Purpose - This SP saves the component criticality information for a
-- particular component and assessment. It is called when the assessment
-- identification process completes.
--
-- See SP save_assesment for explanation of IF/THEN conditional.
-- Author - Mark McLarnon
create procedure save_component_criticalities
(
    @assessment_id    varchar(100),
    @component_num    varchar(100),
    @confidentiality  char(30),
    @integrity        char(30),
    @availability     char(30)
)
AS

IF (NOT EXISTS (SELECT * FROM component_criticality WHERE assessment_id =
@assessment_id AND component_num = @component_num))
BEGIN
    INSERT INTO component_criticality
    (
        [assessment_id],
        [component_num],
        [confidentiality],
        [integrity],
        [availability]
    )
    VALUES
    (
        @assessment_id,
        @component_num,
        @confidentiality,
        @integrity,
        @availability
    )
END
ELSE
    UPDATE component_criticality
    SET confidentiality = @confidentiality,
        integrity        = @integrity,
        availability     = @availability
    WHERE assessment_id = @assessment_id
    AND   component_num = @component_num
GO

```

Included from file **sp_SaveComponentCriticalities.sql**

```

-- Stored Procedure - save_connected_components
-- ASSET Component - ASSET System
-- Purpose - This SP saves the connected component information for a
-- particular component and assessment.
--
-- See SP save_assessment for explanation of IF/THEN conditional.
-- Author - Mark McLarnon
create procedure save_connected_components
(
@assessment_id    varchar(100),
@component_num    varchar(100),
@conn_component   varchar(50),
@bce              integer,
@planned_action   varchar(255)
)
AS
IF (NOT EXISTS (SELECT * FROM conn_components WHERE component_num =
@component_num AND conn_component = @conn_component AND assessment_id =
@assessment_id))
BEGIN
    INSERT INTO conn_components
    (
        [assessment_id],
        [component_num],
        [conn_component],
        [bce],
        [planned_action]
    )
    VALUES
    (
        @assessment_id,
        @component_num,
        @conn_component,
        @bce,
        @planned_action
    )
END
ELSE
    UPDATE conn_components
    SET bce = @bce,
        planned_action = @planned_action
    WHERE
        component_num = @component_num
    AND
        conn_component = @conn_component
    AND
        assessment_id = @assessment_id
GO

```

Included from file **sp_SaveConnectedComponents.sql**

```
-- Stored Procedure - save_new_assessor
-- ASSET Component - ASSET System
-- Purpose - This SP saves user information for an assessor that is added to
-- an assessment. It saves the new assessor information to the SAT_Users
-- table.
-- Author - Mark McLarnon
create procedure save_new_assessor
(
@full_name    varchar(50),
@email        varchar(40),
@phone        varchar(20),
@code         varchar(25)
)
AS
IF (NOT EXISTS (SELECT * FROM SAT_users WHERE full_name = @full_name))
BEGIN
    INSERT INTO SAT_Users
    (
        [full_name],
        [email],
        [phone],
        [code]
    )
    VALUES
    (
        @full_name,
        @email,
        @phone,
        @code
    )
END
UPDATE SAT_Users
SET email = @email,
    phone = @phone,
    code = @code
WHERE full_name = @full_name
GO
```

Included from file **sp_SaveNewAssessor.sql**

B.2.3 ASSET Manager SQL Table Statements

The SQL statements used to create the tables of Manager are the same as those to create the tables of System. Refer to the previous section on ASSET System SQL Table Statements for more information. These statements can be found in the following folder:

C:\Program Files\NISTASSET\ASSET Manager\sql

B.2.4 ASSET Manager SQL Stored Procedure Statements

The following twenty-four Manager SPs provided are those that are different from System. All other SPs provided are the same as System.

```

-- Stored Procedure - count_sensitivities
-- ASSET Component - ASSET Manager
-- Purpose - This procedure counts the system names from the ASSET Manager DB
-- based on one of the three sensitivity fields, confidentiality, integrity,
-- and availability. This SP uses the sp_executesql SP provided by T-SQL to
-- provide a dynamic nature. Note the use of the IF/THEN conditional.
-- Author - Mark McLarnon
create procedure count_sensitivities
(
    @field    varchar(50),
    @value    varchar(50)
)
AS
SET QUOTED_IDENTIFIER OFF
DECLARE @buffer nvarchar(1024)

IF (@field = 'confidentiality')
    BEGIN
        SET @buffer = N'SELECT COUNT(*) AS totals from component_criticality
WHERE confidentiality = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END
IF (@field = 'integrity')
    BEGIN
        SET @buffer = N'SELECT COUNT(*) AS totals from component_criticality
WHERE integrity = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END
IF (@field = 'availability')
    BEGIN
        SET @buffer = N'SELECT COUNT(*) AS totals from component_criticality
WHERE availability = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END
GO

```

Included from file **sp_CountSensitivites.sql**

```
-- Stored Procedure - delete_assessment_information
-- ASSET Component - ASSET Manager
-- Purpose - This SP deletes the assessment information from the assessments
-- table based on the parameter assessment_id supplied. This SP is a part of
-- the parent SP that deletes assessment information from all necessary
-- tables.
-- Author - Mark McLarnon
create procedure delete_assessment_information
(
@assessment_id      varchar(100)
)
AS
DELETE from assessments
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_DeleteAssessmentInformation.sql**


```
-- Stored Procedure - delete_assessor_accountability
-- ASSET Component - ASSET Manager
-- Purpose - This SP deletes assessment accountability information from
-- the assessors table based on the parameter assessment_id. This SP is a
-- part of a parent SP that deletes assessment information from all necessary
-- tables.
-- Author - Mark McLarnon
create procedure delete_assessor_accountability
(
@assessment_id      varchar(100)
)
AS
DELETE from assessors
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_DeleteAssessorAccountability.sql**

```
-- Stored Procedure - delete_component_criticality
-- ASSET Component - ASSET Manager
-- Purpose - See SP delete_assessment_information.
-- Author - Mark McLarnon
create procedure delete_component_criticality
(
@assessment_id    varchar(100),
@component_num    varchar(100)
)
AS
DELETE from component_criticality
WHERE assessment_id = @assessment_id AND component_num = @component_num
GO
```

Included from file **sp_DeleteComponentCriticality.sql**

```
-- Stored Procedure - delete_component_information
-- ASSET Component - ASSET Manager
-- Purpose - See SP delete_assessment_information.
-- Author - Mark McLarnon
create procedure delete_component_information
(
@assessment_id      varchar(100),
@component_num      varchar(100)
)
AS
DELETE from components
WHERE component_num = @component_num AND assessment_id = @assessment_id
GO
```

Included from file **sp_DeleteComponentInformation.sql**

```
-- Stored Procedure - delete_connected_components
-- ASSET Component - ASSET Manager
-- Purpose - See SP delete_assessment_information.
-- Author - Mark McLarnon
create procedure delete_connected_components
(
@assessment_id      varchar(100),
@component_num      varchar(100)
)
AS
DELETE from conn_components
WHERE component_num = @component_num AND assessment_id = @assessment_id
GO
```

Included from file **sp_DeleteConnectedComponents.sql**

```
-- Stored Procedure - delete_question_responses
-- ASSET Component - ASSET Manager
-- Purpose - See SP delete_assessment_information.
-- Author - Mark McLarnon
create procedure delete_question_responses
(
@assessment_id      varchar(100)
)
AS
DELETE from assessment_question_responses
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_DeleteQuestionResponses.sql**

```

-- Stored Procedure - delete_saved_assessment
-- ASSET Component - ASSET Manager
-- Purpose - See SP delete_assessment_information.
-- Author - Mark McLarnon
create procedure delete_saved_assessment
(
@assessment_id      varchar(100),
@component_name     varchar(50),
@start_timestamp    varchar(50)
)
AS
DECLARE @component_num      varchar(100)

SET @component_num = (SELECT component_num from components where
component_name = @component_name)

EXECUTE delete_connected_components @assessment_id, @component_num

EXECUTE delete_component_criticality @assessment_id, @component_num

EXECUTE delete_question_responses @assessment_id

EXECUTE delete_assessor_accountability @assessment_id

EXECUTE delete_component_information @assessment_id, @component_num

EXECUTE delete_assessment_information @assessment_id

GO

Included          from          file          sp_DeleteSavedAssessments.sql

```

```

-- Stored Procedure - get_all_assessment_information
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts all assessment information from the assessments
-- components and component_criticality table as inner-joins.
-- Author - Mark McLarnon
create procedure get_all_assessment_information
AS
SELECT
    assessments.assessment_id,
    assessments.completed,
    assessments.primary_assessor,
    components.component_num,
    components.location,
    components.component_name,
    components.component_type,
    component_criticality.confidentiality,
    component_criticality.integrity,
    component_criticality.availability
FROM
    (
        assessments
    INNER JOIN
        components
    ON
        assessments.assessment_id = components.assessment_id
    )
    INNER JOIN
        component_criticality
    ON
        components.component_num = component_criticality.component_num
    AND
        components.assessment_id = component_criticality.assessment_id
    AND
        assessments.active = 1
    AND
        components.active = 1
GO

```

Included from file **sp_getAllAssessmentInformation.sql**

```
-- Stored Procedure - get_all_questions
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts the questions from assessment_questions table
-- inner-joined with the sys_controlareas table.
-- Author - Mark McLarnon
create procedure get_all_questions
AS
SELECT *
FROM
    sys_controlareas
INNER JOIN
    assessment_questions
ON
    sys_controlareas.area_index = assessment_questions.control_area
GO
```

Included from file **sp_getAllQuestions.sql**


```
-- Stored Procedure - get_assessment_count
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts the number of assessments stored in the Manager
-- database.
-- Author - Mark McLarnon
create procedure get_assessment_count
AS
select count(*) AS AssessmentCount
from assessments
GO
```

Included from file **sp_GetAssessmentCount.sql**

```
-- Stored Procedure - get_assessor_report
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts the assessors that were involved with a
-- specific assessment from the assessors table.
-- Author - Mark McLarnon
create procedure get_assessor_report
(
@assessment_id    varchar(100)
)
AS
SELECT *
FROM assessors
WHERE assessment_id = @assessment_id
GO
```

Included from file **sp_getAssessorReport.sql**

```
-- Stored Procedure - get_components
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts the list of components from the components
-- table.
-- Author - Mark McLarnon
create procedure get_components
AS
SELECT *
FROM components
GO
```

Included from file **sp_getComponents.sql**

```
-- Stored Procedure - get_questions
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts a listing of the current questions from the
-- assessment_questions table.
-- Author - Mark McLarnon
create procedure get_questions
(
    @version    integer
)
AS
SELECT *
FROM
    sys_controlareas
INNER JOIN
    assessment_questions
ON
    sys_controlareas.area_index = assessment_questions.control_area
WHERE
    version = @version
GO
```

Included from file **sp_getQuestions.sql**

```
-- Stored Procedure - get_system_organization
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts a listing of all of the systems within the
-- Manager database and their associated location.
-- Author - Mark McLarnon
create procedure get_system_organization
AS
SELECT
    components.component_num,
    component_name,
    location
FROM
    components
WHERE
    components.active = 1
GO
```

Included from file **sp_getSummarySystemOrganizations.sql**

```

-- Stored Procedure - get_system_sensitivity
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts a listing of the systems within the Manager
-- database and their associated sensitivity values, C, I and A.
-- Author - Mark McLarnon
create procedure get_system_sensitivity
AS
SELECT
    components.component_num,
    component_name,
    component_criticality.confidentiality,
    component_criticality.integrity,
    component_criticality.availability
FROM
    components INNER JOIN component_criticality
    ON
    components.component_num = component_criticality.component_num
WHERE
    components.active = 1
AND
    component_criticality.active = 1
GO

```

Included from file **sp_getSummarySystemSensitivity.sql**

```
-- Stored Procedure - get_systems_by_type
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts a complete listing of the systems within the
-- Manager database and either associated system type.
-- Author - Mark McLarnon
create procedure get_systems_by_type
AS
SELECT component_num,
        component_name,
        component_type
FROM
    components
WHERE
    components.active = 1
GO
```

Included from file **sp_getSystemsByType.sql**

```
-- Stored Procedure - get_sys_types
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts a listing of the systems types from the Manager
-- database.
-- Author - Mark McLarnon
create procedure get_sys_types
AS
SELECT *
FROM sys_Component_Types
GO
```

Included from file **sp_getSystemTypes.sql**


```

-- Stored Procedure - query_single_saved_assessment
-- ASSET Component - ASSET Manager
-- Purpose - This SP extracts all of the information for a particular
-- assessment of a particular system from the assessments table inner-joined
-- with the components table.
-- Author - Mark McLarnon
create procedure query_single_saved_assessment
(
    @assessment_id    varchar(100)
)
AS
SELECT
    components.component_name AS component_name,
    components.component_num AS component_num,
    components.location AS component_location,
    components.component_type AS component_type,
    components.assessment_obj AS component_assessment_obj,
    assessments.assessment_id,
    assessments.start_timestamp,
    assessments.completed,
    assessments.bookmark,
    assessments.primary_assessor
FROM
    assessments
INNER JOIN
    components
ON
    assessments.assessment_id = components.assessment_id
WHERE
    assessments.assessment_id = @assessment_id
ORDER BY
    components.component_name
GO

```

Included from file **sp_QuerySingleSavedAssessment.sql**

```
-- Stored Procedure - rpt_reset_active_assessments
-- ASSET Component - ASSET Manager
-- Purpose - This SP resets the systems within the Manager database to all
-- being active.
-- Author - Mark McLarnon
create procedure rpt_reset_active_assessments
AS
UPDATE assessments SET assessments.active = 1
WHERE NOT assessments.active = 1

UPDATE assessment_question_responses SET assessment_question_responses.active
= 1
WHERE NOT assessment_question_responses.active = 1

UPDATE assessors SET assessors.active = 1
WHERE NOT assessors.active = 1

UPDATE components SET components.active = 1
WHERE NOT components.active = 1

UPDATE component_criticality SET component_criticality.active = 1
WHERE NOT component_criticality.active = 1

UPDATE conn_components SET conn_components.active = 1
WHERE NOT conn_components.active = 1
GO
```

Included from file **sp_ResetActiveAssessments.sql**

```

-- Stored Procedure - search_saved_assessments
-- ASSET Component - ASSET Manager
-- Purpose - This procedure searches the manager database based on the string
-- value that the user supplies. The field parameter determines which field
-- the user wishes to search on. The conversion between a string on the
-- Manager UI and the actual field name is accomplished with a hash table.
-- Author - Mark McLarnon
create procedure search_saved_assessments
(
@field      varchar(30),
@value      varchar(100)
)
AS
SET QUOTED_IDENTIFIER OFF
DECLARE @buffer nvarchar(1024)

IF (@field = 'assessment_id')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,        assessments.bookmark,        assessments.primary_assessor
FROM          assessments      INNER          JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE
assessments.assessment_id = '' + CAST (@value AS nvarchar(100)) + ''
        exec sp_executesql @buffer
    END

IF (@field = 'component_num')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,        assessments.bookmark,        assessments.primary_assessor
FROM          assessments      INNER          JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE
components.component_num = '' + CAST (@value AS nvarchar(100)) + ''
        exec sp_executesql @buffer
    END

IF (@field = 'component_name')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,        assessments.bookmark,        assessments.primary_assessor
FROM          assessments      INNER          JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE
components.component_name = '' + CAST (@value AS nvarchar(100)) + ''
        exec sp_executesql @buffer
    END

```

```

IF (@field = 'component_type')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE
components.component_type = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

IF (@field = 'location')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE components.location
= ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

IF (@field = 'start_timestamp')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE
assessments.start_timestamp = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

IF (@field = 'completed')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE
assessments.completed = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

```

```

IF (@field = 'bookmark')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE assessments.bookmark
= ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

IF (@field = 'assessment_obj')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE
components.assessment_obj = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END

IF (@field = 'primary_assessor')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE
assessments.primary_assessor = ''' + CAST (@value AS nvarchar(100)) + ''''
        exec sp_executesql @buffer
    END
GO

```

Included from file **sp_SearchSavedAssessments.sql**

```

-- Stored Procedure - search_saved_assessments_partial
-- ASSET Component - ASSET Manager
-- Purpose - This is a experimental SP that performs the same function as the
-- search_saved_assessments SP except uses partial text searching.
-- Author - Mark McLarnon
create procedure search_saved_assessments_partial
(
@field      varchar(30),
@value      varchar(100)
)
AS
SET QUOTED_IDENTIFIER OFF
DECLARE @buffer nvarchar(1024)

IF (@field = 'assessment_id')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,      assessments.bookmark,      assessments.primary_assessor
FROM          assessments      INNER      JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE          CONTAINS
(assessment_id, '' + CAST (@value AS nvarchar(100)) + '')'
        print @buffer
        exec sp_executesql @buffer
    END

IF (@field = 'component_num')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,      assessments.bookmark,      assessments.primary_assessor
FROM          assessments      INNER      JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE          CONTAINS
(component_num, '' + CAST (@value AS nvarchar(100)) + '')'
        exec sp_executesql @buffer
    END

IF (@field = 'component_name')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num      AS      component_num,      components.location      AS
component_location,          components.component_type      AS      component_type,
components.assessment_obj      AS      component_assessment_obj,
assessments.assessment_id      AS      id,          assessments.start_timestamp,
assessments.completed,      assessments.bookmark,      assessments.primary_assessor
FROM          assessments      INNER      JOIN          components      ON
assessments.assessment_id=components.assessment_id      WHERE          CONTAINS
(component_name, '' + CAST (@value AS nvarchar(100)) + '')'
        print @buffer
        exec sp_executesql @buffer
    END
END

```

```

IF (@field = 'component_type')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS
(component_type, ''' + CAST (@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

IF (@field = 'location')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS (location,
''' + CAST (@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

IF (@field = 'start_timestamp')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS
(start_timestamp, ''' + CAST (@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

IF (@field = 'completed')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS (completed,
''' + CAST (@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

```

```

IF (@field = 'bookmark')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS (bookmark,
''' + CAST (@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

IF (@field = 'assessment_obj')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS (''' + CAST
(@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END

IF (@field = 'primary_assessor')
    BEGIN
        SET @buffer = N'SELECT components.component_name AS component_name,
components.component_num AS component_num, components.location AS
component_location, components.component_type AS component_type,
components.assessment_obj AS component_assessment_obj,
assessments.assessment_id AS id, assessments.start_timestamp,
assessments.completed, assessments.bookmark, assessments.primary_assessor
FROM assessments INNER JOIN components ON
assessments.assessment_id=components.assessment_id WHERE CONTAINS (''' + CAST
(@value AS nvarchar(100)) + ''')'
        exec sp_executesql @buffer
    END
GO

```

Included from file **sp_SearchSavedAssessmentsPartial.sql**


```

-- Stored Procedure - set_active_assessments
-- ASSET Component - ASSET Manager
-- Purpose - This SP sets certain assessments as active as chosen by the
-- user. It uses the sp_executesql SP to execute dynamic SQL; this way the
-- user can
-- set the list of active assessments in real time.
-- Author - Mark McLarnon
create procedure set_active_assessments
(
@values      varchar(1024)
)
AS
DECLARE @buffer nvarchar(1024)
DECLARE @c_values nvarchar(1024)

SET @c_values = CONVERT(nvarchar(1024), @values)

SET @buffer = N'UPDATE assessments SET active = 0 WHERE assessment_id NOT IN
' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE assessments SET active = 1 WHERE assessment_id IN ' +
@c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE assessment_question_responses SET active = 0 WHERE
assessment_id NOT IN ' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE assessment_question_responses SET active = 1 WHERE
assessment_id IN ' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE assessors SET active = 0 WHERE assessment_id NOT IN '
+ @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE assessors SET active = 1 WHERE assessment_id IN ' +
@c_values
exec sp_executesql @buffer
GO

```

Included from file **sp_SetActiveAssessments.sql**

```

-- Stored Procedure - set_active_components
-- ASSET Component - ASSET Manager
-- Purpose - This procedure works in tandem with the set_active_assessments
-- SP.
-- Author - Mark McLarnon
create procedure set_active_components
(
@values      varchar(1024)
)
AS
DECLARE @buffer  nvarchar(1024)
DECLARE @c_values nvarchar(1024)

SET @c_values = CONVERT(nvarchar(1024), @values)

SET @buffer = N'UPDATE components SET active = 0 WHERE component_num NOT IN '
+ @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE components SET active = 1 WHERE component_num IN ' +
@c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE conn_components SET active = 0 WHERE component_num NOT
IN ' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE conn_components SET active = 1 WHERE component_num IN
' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE component_criticality SET active = 0 WHERE
component_num NOT IN ' + @c_values
exec sp_executesql @buffer

SET @buffer = N'UPDATE component_criticality SET active = 1 WHERE
component_num IN ' + @c_values
exec sp_executesql @buffer
GO

```

Included from file **sp_SetActiveComponents.sql**

Appendix C—Advanced Troubleshooting

This section presents all of the advanced troubleshooting procedures for ASSET. It is important to note that all of these procedures are outlined on the NIST ASSET website. This website will always contain the latest troubleshooting procedures, and users should check this first for the latest information.

Perhaps the most important troubleshooting point to make is the operating systems on which ASSET will run successfully:

- Windows 2000 Professional
- Windows XP Professional
- Windows NT Workstation 4.0 (with user action)

C.1 MSDE Interoperability

Other applications that utilize the MSDE system will cause interoperability issues with ASSET. Specifically, Microsoft's diagramming tool Visio has known interoperability problems with all other applications that utilize MSDE. The Auto Discovery and Layout add-on to Visio 2000 Enterprise Edition utilizes the MSDE. Microsoft has stated that Visio 2000 Enterprise Edition and later expects the 'sa' account to have a null password. If the ASSET installation changes the 'sa' account, this will cause Visio to cease to operate as expected.

Multiple laboratory tests have shown that ASSET installs successfully on a system that has all versions of Visio installed. User reports have shown that although users are not always able to successfully change the MSDE password during the installation process, they can do so after the installation has completed. This process is outlined in detail on the ASSET website.

Microsoft Share Point Team Services and Microsoft Project Central both perform the same operation in changing the 'sa' account password and, therefore, will likely not interoperate with ASSET. These products were not tested together on the same machine because it was not intended for ASSET to coexist on machines with these products.

The ASSET installation process requires that the MSDE and even ASSET program files be installed to the C:\ drive. This is referred to by the operating system environment variable %SystemDrive%. Users that have manipulated the ASSET installer to install files to their locations have reported problems with ASSET operation. This is outside of the bounds of the intended operation of ASSET.

C.2 Changing the Default Database Password

If users have installed a product that has changed the password for the 'sa' account like those mentioned in the last section, they will need to know this password so that ASSET can function. When the password is not blank, the password change feature of the ASSET installer will not succeed, and users will have to edit the configuration files of System and Manager manually. This process is outlined in support documents on the ASSET website. For products like Share Point Team Services and Project Central, refer to their documentation on how to obtain the password for the 'sa' account. Future versions of ASSET may utilize an alternate login than the 'sa' account to mitigate these problems.

C.3 ACL/Security Templates

ASSET users that operate on Windows 2000 and Windows XP may notice problems installing and operating ASSET if their machine has a configured security policy. As stated in the ASSET User manual, users need to be logged into their machine with Administrative privileges to install ASSET. If they have used a tool like secedit.exe to custom configure the security of their machine, ASSET will most likely not install. This has to do with the security permissions placed upon the Program Files directories. Of the example security policies provided with Windows 2000 Professional, only the basicwk.inf security policy has been tested to work with ASSET successfully.

Appendix D—Design Considerations

This section outlines the design considerations that have affected the development of ASSET and the section within this document that users can refer to for more information. The design considerations are presented in the following table with an associated reference section within the body of the document.

Consideration	Associated Section
System UI uses Swing Metal PLAF	3.1
No sorting provided in ASSET 1.0	3.5
Keyboard navigation requires SHIFT-TAB instead of TAB only	3.1.1.2
No partial text searching in Manager	3.4.5
No hierarchical distribution of System and Manager	4.3
ASSET Uses MSDE and T-SQL	4.1
UI delivered via JInternalFrames instead of JFrames	3.1.2

Appendix E—Acronyms

ACL	Access Control List
ADPU	ASSET Database Password Utility
API	Applications Programming Interface
ASSET	Automated Security Self-Evaluation Tool
DSN	Data Source Name
DTD	XML Document Type Definition
ER	Entity relationship
GISRA	Government Information Security Reform Act
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
ITL	Information Technology Laboratory
ITSAF	IT Security Assessment Framework
JAR	Java archive
JDBC	Java Database Connectivity
JDK	Java Development Kit
ME	Windows Millennium Edition
MMC	Microsoft Management Console
MSBA	Microsoft Security Baseline Analyzer
MSDE	Microsoft SQL Server Desktop Engine
NIST	National Institute of Standards and Technology
ODBC	Open database connectivity
PLAF	Pluggable Look and Feel
SAX	Simple API for XML Processing
SQL	Select Query Language
SP	Stored procedure
SP 800-26	NIST Special Publication 800-26
T-SQL	Transact SQL
UML	Unified Modeling Language
URL	Uniform Resource Location
XML	Extensible Markup Language
XP	Windows XP

Appendix F—References

This section lists the Internet references cited for design considerations made within the ASSET source code. Readers may also refer to section 1.6 of the document for a listing of the external sources referenced within this document. The URLs listed here are accompanied by a short description.

URL	Description
http://www.jguru.com/faq/view.jsp?EID=316478	Describes method for navigating among components on a Swing user interface using the TAB key.
http://www2.epocworld.com/kbjava.nsf/ce9a3a52e9d969788025670e00429dbd/cc18b44bb062199c802569e50058aaf1?OpenDocument	Describes the use of <code>getKeyChar()</code> method when implementing <code>KeyPressedEvent</code> action handler.
http://www2.gol.com/users/tame/swing/examples/JTableExamples5.html	Describes method for implementing bidirectional sorting within <code>JTables</code> .
http://www.ibiblio.org/xml/books/xmljava/chapters/ch03s03.html	Describes method for saving XML documents to outfiles without using XML parsers or preprocessors.
http://pooh.east.asu.edu/Cet427/ClassNotes/Background/cnBackground-24.html	Provides example on implementing <code>Threads</code> in Java by extending the <code>Thread</code> class instead of implementing the <code>Runnable</code> interface.
http://dbserv.jinr.ru/js/content/java/New-tutorial/ui/swing/filechooser.html	Describes method for programming <code>JFileChooser</code> on Swing.
http://java.sun.com/docs/books/tutorial/uiswing/misc/plaf.html	Web page describing the Pluggable look and feel (PLAF) design of Swing.
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q241397	URL that provides recommendations for backing up MSDE databases using <code>Transact-SQL</code> .
http://www.xml-zone.com/articles/pm020101/pm020101-1.asp	Online article that describes parsing XML files.
http://www.xml.com/1999/09/conformance/reports/report-xml4j-val.html	XML article that describes <code>SAX</code> XML processors and required tag structure.
http://developer.java.sun.com/developer/TechTips/2000/tt0627.html	JDC TechTips article that provides examples of using the <code>SAX</code> and <code>DOM</code> API from the <code>jdom.org</code> .
http://www7.software.ibm.com/vad.nsf/Data/Document2329	Article from IBM that describes the <code>Model-View-Controller</code> architecture.
http://www.esus.com/javaindex/j2se/jdk1.2/javaxswing/editableatomiccontrols/jtable/jtablestrows.html	Article that describes alternate method for implementing sorting procedures within <code>Swing JTable</code> interface.
http://www.codeguru.com/java/articles/141.shtml	Article that describes using <code>JTree</code> to display text across multiple lines. This URL describes the need to use the <code>Metal</code> <code>PLAF</code> .
http://www.esus.com/javaindex/j2se/jdk1.2/javaxswing/editableatomiccontrols/jtable/jtabledetectsel.html	Article that describes the use of <code>ActionHandler</code> to detect the selection of certain cell within <code>JTable</code> using <code>ColumnModelListener</code> class.
http://www.esus.com/javaindex/j2se/jdk1.2/javaxswing/editableatomiccontrols/jtable/jtablewidthcolumn.html	Article that describes the process to set the width of a <code>Column</code> within a <code>JTable</code> on <code>Swing</code> .
http://dbserv.jinr.ru/js/content/java/New-tutorial/ui/swing/filechooser.html	Additional article that describes use of <code>JFileChooser</code> within <code>Swing</code> application, especially method to set selected file name.
http://www.esus.com/javaindex/j2se/jdk1.2/javaxswing/editableatomiccontrols/jtable/jtabletextarea.html	Article that describes method to embed <code>JTextArea</code> within cell of <code>JTable</code> using <code>CustomCellRenderer</code> class.