**Technical Report: Establishing a Development Environment for Autopsy**

**Date:** 4 July 2025

**Author:** Syed Naveed Hasan

**Objective:** The primary objective of this undertaking was to configure a local development environment on a Windows 11 system for the purpose of building the Autopsy digital forensics platform from its source code. This process is a prerequisite for any subsequent module development or contributions to the core application. The process involved two main open-source projects: The Sleuth Kit (TSK), which provides the core digital forensics analysis engine, and Autopsy, which provides the graphical user interface (GUI).

**1.0 Initial Environment Configuration**

The initial phase involved the setup of the foundational software required for Java-based development.

* **Java Development Kit (JDK):** Based on the project's requirements, JDK 21 was initially installed. A version check in the command prompt confirmed a successful installation (

java 21.0.7 2025-04-15 LTS).

* **Integrated Development Environment (IDE):** Apache NetBeans IDE, version 22, was selected and installed as the development platform.

**2.0 Prerequisite Compilation: The Sleuth Kit (TSK)**

The Autopsy platform is fundamentally dependent on The Sleuth Kit. Therefore, the first major development task was to compile the TSK source code, specifically its Java bindings.

1. The sleuthkit Git repository was cloned into a local directory.
2. The Java bindings sub-project, located at sleuthkit\bindings\java, was opened in the NetBeans IDE. This project is named "Data Models" within the IDE.

**3.0 Troubleshooting the TSK Build Process**

The initial attempt to build the "Data Models" project (Clean and Build command) resulted in a failure. This initiated a comprehensive and iterative troubleshooting process to diagnose and resolve the underlying issues.

**3.1 Issue: JDK Version Incompatibility**

* **Symptom:** The build failed, and an inspection of the project's source files revealed "contains files with errors" indicators within the IDE. The log files showed errors such as “package org.sqlite does not exist.”
* **Analysis:** It was determined that the sleuthkit source code was not compatible with the modern JDK 21. Older projects often have dependencies on specific language features or library versions.
* **Resolution:** A Long-Term Support (LTS) version, JDK 17, was identified as a more compatible alternative. JDK 17 was installed alongside JDK 21, and the NetBeans IDE was configured to recognize it as a valid Java Platform. The "Data Models" project was then explicitly configured to use JDK 17 for its build process.

**3.2 Issue: Dependency Management Failure (Apache Ivy)**

* **Symptom:** After re-configuring the JDK, the build process would hang indefinitely during a dependency resolution phase, specifically the download-ivy target. The process consumed significant time (over 40 minutes) without producing a success or failure message. This pointed towards a network connectivity problem within the build environment.
* **Analysis:** It was hypothesized that a firewall was blocking the build tool (Apache Ant) from downloading its own dependency manager, Apache Ivy. A direct download of a sample dependency via a web browser was successful, confirming that the issue was specific to the Java/Ant environment and not a general network failure.
* **Resolution:** To circumvent the network block, the apache-ivy-2.5.2-bin.zip package was downloaded manually. The ivy-2.5.2.jar file was extracted and manually placed into the C:\dev\sleuthkit\bindings\java\lib directory, where the build script expected to find it. Following this, the init-ivy build target was run successfully.

**3.3 Issue: Manual Dependency Population**

* **Symptom:** After successfully initializing Ivy, the main NetBeans build for the **Autopsy** suite (not TSK) failed again. The build log presented a clear error:

Warning: Could not find file ...\sleuthkit\bindings\java\dist\sleuthkit-4.14.0.jar to copy.

* **Analysis:** This revealed a two-part problem. First, the Data Models build had created sleuthkit-4.13.0.jar, but the newer Autopsy build script required version 4.14.0. Second, subsequent build attempts showed that the Ivy dependency resolution was still not working for the main Autopsy build, leading to further "Could not find file" errors for libraries like sqlite-jdbc, postgresql, and c3p0.
* **Resolution:** A multi-step manual resolution was required:
  1. The successfully built sleuthkit-4.13.0.jar was duplicated. One copy was left as is, and the other was renamed to sleuthkit-4.14.0.jar to satisfy both the older and newer build scripts.
  2. The required library files (sqlite-jdbc-3.49.1.0.jar, postgresql-42.7.5.jar, c3p0-0.10.2.jar, mchange-commons-java-0.3.2.jar, and SparseBitSet-1.3.jar) were manually downloaded from the Maven Central repository.
  3. These .jar files were manually placed into the C:\non\_os\project\7030\sleuthkit\bindings\java\lib\ directory.

**4.0 Native C++ Code Compilation (Visual Studio)**

With the Java portion of the prerequisites resolved, the build process moved to the native C++ code. The Autopsy runtime requires compiled .dll files from libtsk.

**4.1 Issue: C++ Build Failure (Missing Dependencies)**

* **Symptom:** The initial attempt to build the tsk.sln solution in Visual Studio 2022 resulted in 33 failed projects. The primary error was in the libtsk project: error MSB3073: The command "copy ...\packages\sleuthkit-libewf...".
* **Analysis:** This indicated that the C++ dependency manager, NuGet, had failed to automatically restore the required native libraries (such as libewf and zlib) into the packages directory.
* **Resolution:** The project's build file (libtsk.vcxproj) was manually edited. All XML blocks that referenced the failing NuGet packages—specifically the <ImportGroup Label="ExtensionTargets"> and <Target Name="EnsureNuGetPackageBuildImports"...> sections, as well as the <PreBuildEvent> copy commands—were removed. This severed the dependency on the failing automatic restoration process.

**4.2 Issue: JNI Header Not Found**

* **Symptom:** After removing the NuGet dependencies, the build failed again, but with a new root cause: error C1083: Cannot open include file: 'jni.h': No such file or directory. This error occurred in the

libtsk\_jni project.

* **Analysis:** The libtsk\_jni project is the bridge between C++ and Java. It requires the Java Native Interface (JNI) header files to compile. The Visual Studio project was not configured to know the location of the installed JDK.
* **Resolution:** The properties for the libtsk\_jni project were modified in Visual Studio. The paths C:\Program Files\Java\jdk-17\include and C:\Program Files\Java\jdk-17\include\win32 were added to the "Additional Include Directories" for the "Release | x64" configuration.

**4.3 Issue: Runtime Linker Error (UnsatisfiedLinkError)**

* **Symptom:** After successfully compiling all 34 C++ projects in Visual Studio (========== Rebuild All: 34 succeeded, 0 failed... ), running Autopsy from NetBeans produced a fatal error dialog. The runtime log showed

java.lang.UnsatisfiedLinkError: no libcrypto-1\_1-x64 in java.library.path.

* **Analysis:** This indicated that while the C++ code was now compiled, the Java Virtual Machine could not find the required OpenSSL DLLs at runtime. These DLLs were part of the NuGet packages that we had bypassed.
* **Resolution:** The necessary files, libcrypto-vc142-x64-1\_1\_0.dll and libssl-vc142-x64-1\_1\_0.dll, were located in the downloaded NuGet package cache (...\\packages\\openssl-vc142.1.1.0\\build\\native\\bin\\x64). These two files were manually copied into the final build output directory (C:\\non\_os\\project\\7030\\sleuthkit\\win32\\x64\\Release\\), where all other native libraries resided.

**5.0 Conclusion**

The endeavor to establish a functional development environment for Autopsy module creation on a modern Windows 11 system proved to be an illustrative case study in the challenges of building complex, multi-language, open-source software. While the initial setup of the Java Development Kit (JDK) and NetBeans IDE was straightforward, the subsequent compilation of the required Sleuth Kit (TSK) and Autopsy projects revealed a series of deeply intertwined and brittle dependency issues.

The troubleshooting process, documented in this report, uncovered a cascade of failures. The root causes were identified as:

Strict Version Coupling: The TSK source code exhibited a strong dependency on older versions of the JDK (specifically JDK 17), failing to compile with the more contemporary JDK 21.

Fragile Dependency Management: The build scripts' reliance on Apache Ivy for fetching Java libraries and NuGet for C++ libraries was not robust. These automated processes failed silently in the development environment, necessitating a significant manual "shopping list" approach to download and place dozens of required .jar and .dll files.

Cross-Language Build Complexity: The build process was not self-contained, requiring a separate, successful compilation of the C++ tsk.sln solution within Visual Studio before the main Java-based build in NetBeans could even proceed. This C++ build, in turn, had its own misconfigured project files that required manual editing to locate the JDK's native interface headers (jni.h).

The critical takeaway from this exercise is that the Autopsy development environment, at least when built from source on a modern Windows platform, appears to lack the resilience and up-to-date support necessary for rapid development. The reliance on specific, and in some cases outdated, versions of the JDK, Visual Studio, and numerous third-party libraries creates a significant barrier to entry. The build process is not atomic and depends on a precise and fragile sequence of manual interventions.

Given that these extensive environmental and dependency challenges are tangential to the primary research objective of this dissertation, a strategic decision was made in consultation with my supervisor. The effort required to maintain this brittle build environment was deemed a significant risk to the project timeline. Consequently, this development path has been abandoned in favor of a more direct and maintainable approach: the creation of a standalone Python application that will leverage digital forensics libraries to achieve the project's goals. This pivot allows the focus to return to the core research, mitigating the risk of further delays due to platform-specific configuration complexities.