ChatGPT

Installing OpenJDK 21 in Codex (Ubuntu 24.04) Without Ca-Certificates Errors

Background and Problem Description

The **openai/codex-universal** environment (based on Ubuntu 24.04) has encountered failures when installing OpenJDK 21 due to issues with the `ca-certificates-java` package. During installation, the post-install scripts for Java's CA certificates can error out with messages like:

```
org.debian.security.UnableToSaveKeystoreException: ...
Caused by: java.io.FileNotFoundException: /etc/ssl/certs/java/cacerts (No such
file or directory)
dpkg: error processing package ca-certificates-java (--configure): installed ca-
certificates-java package post-installation script subprocess returned error
exit status 1
```

This error indicates the Java keystore (`cacerts`) could not be saved because the target directory/file was missing [1] [2]. In a CI build (non-interactive cloud environment), such failures break the setup script – for example, in the *WristLingo* project, which requires Java 21 for Kotlin 2.2.20 and Android Gradle Plugin 8.12.3, the Java install step can halt due to this issue.

Why does this happen? In a clean Ubuntu environment (like a fresh container or Codex VM), the directory `/etc/ssl/certs/java` might not exist initially. The `ca-certificates-java` package's trigger script (`/etc/ca-certificates/update.d/jks-keystore`) expects this directory to be present and a Java runtime available. If not, the attempt to generate or update the `cacerts` keystore fails [1]. This is a known issue in Ubuntu's Java packaging – earlier bugs show that if Java isn't fully configured or the directory is missing, the certificate update step errors out [1]. In some cases the failure is because the `java` / `keytool` command wasn't found when the trigger ran [3], or because the target path didn't exist. By default, Debian/Ubuntu's OpenJDK uses a system-wide keystore at `/etc/ssl/certs/java/cacerts`, often symlinked into the JDK's security directory, so the packaging scripts need to create and update this file.

Causes and Known Workarounds

Several approaches have been identified to address or avoid this installation failure:

- **Pre-create the keystore directory:** Ensuring that `/etc/ssl/certs/java` exists (creating it if necessary) before installing Java avoids the "No such file or directory" issue. A reported workaround for the bug is simply:

```
if [ ! -d "/etc/ssl/certs/java" ]; then mkdir -p /etc/ssl/certs/java; fi
```

which prevents the update script from failing on a missing directory [4].

- **Adjust package installation order:** Some have suggested installing `ca-certificates-java` **before** installing the JDK/JRE. The idea is to let the certificate package configure itself (or at least get the directory set up) prior to adding Java. For example, one solution for a similar issue with OpenJDK 11 was:

```
apt-get install -y ca-certificates-java && apt-get install -y openjdk-11-jre-headless
```

This was recommended as a fix for a related bug [5]. In practice, however, ordering alone may not reliably solve the problem in all cases – if `ca-certificates-java` runs its trigger but no Java is present, it might still fail (older versions had a bug where the script would try to call `java` and error out if missing [3]). Newer package fixes have improved this, but it can still be race-prone in single APT transactions.

- **Ignoring the error (not recommended):** The Codex discussion notes that one workaround was to temporarily disable `set -e` (error exit) so the script doesn't abort on the `ca-certificates-java` failure [6]. For example:

```
set +e  # turn off immediate exit on error
sudo apt-get install -y openjdk-21-jre-headless ca-certificates-java
set -e  # turn it back on
```

This would let the installation continue despite the error. **However, this is not a good practice** – it masks the failure and leaves `ca-certificates-java` unconfigured (meaning the Java truststore may not be properly set up). It's better to fix the root cause so the install succeeds cleanly.

- **Delaying trigger execution (deferred configuration):** The most robust solution involves controlling when the certificate update trigger runs. Debian's package manager allows deferring triggers using options like `-o Dpkg::Options::=--no-triggers` during installation, then manually executing the triggers after ensuring the environment is ready. This method was successfully used by community members in the Codex Q&A to consistently install Java 21 without errors [7]. By doing so, we can install all required packages in one go (to satisfy dependencies) but postpone the `jks-keystore` update until Java is fully in place and the directory exists.

Reliable Installation Solution

Combining the insights above, the **most reliable approach** is to explicitly prepare the environment and leverage deferred trigger handling. In summary, the solution will:

1. **Ensure the keystore directory exists** – Create `/etc/ssl/certs/java` and remove any stale or partial `cacerts` file/symlink if one exists. This addresses the **FileNotFound** error by guaranteeing the path is ready [4].
2. **Install OpenJDK 21 and certificate packages with triggers disabled** – Use `apt-get` with `--no-install-recommends` (to avoid unnecessary GUI components) and the `--no-triggers` option for the problematic packages. This installs the Java runtime (and development kit, if needed) along with `ca-certificates` and `ca-certificates-java` **without immediately running the post-install trigger scripts** [7]. In this step, we also set `DEBIAN_FRONTEND=noninteractive` to ensure APT doesn't prompt for any input.
3. **Manually trigger the certificate keystore update** – After installation, explicitly run the trigger for `ca-certificates-java` (e.g. via `dpkg --triggers-only ca-certificates-java`) and then run `update-ca-certificates -f`. This generates the `/etc/ssl/certs/java/cacerts` keystore now that Java and all dependencies are in place [8]. Essentially, we're **delaying the** `jks-keystore` **update** to a safe point in time.
4. **Symlink the keystore into the JDK, if needed** – Ubuntu's OpenJDK package normally expects the JDK's own `lib/security/cacerts` to point to the system keystore. In some cases (depending on package ordering), that symlink might not be set up, so we create it manually if it's missing [9]. This ensures that the Java installation will use the updated truststore (important for Gradle, Kotlin, and other tools to be able to make HTTPS connections).
5. **Verify the keystore contents** – As a final sanity check, use the Java `keytool` to list the certificates in the keystore, confirming that it contains many entries (the default CAs) and is not empty [10]. This step is optional but recommended in CI for debugging.

By following these steps, we can guarantee a clean Java 21 setup in the Codex environment on every run. It avoids interactive prompts, handles the known bug gracefully, and minimizes installed components (headless JDK/JRE, no GUI libraries). This approach preserves full Gradle and Android toolchain functionality: Gradle will have a Java 21 JDK available for compilation, and the system truststore will be properly configured so that build scripts and dependency downloads (which use HTTPS) work without certificate issues.

Step-by-Step Solution Outline

To summarize the solution in a concise order of execution:

1. **Prepare environment for non-interactive install:** Set `DEBIAN_FRONTEND=noninteractive` and ensure `set -euo pipefail` is used for robust scripting. Update PATH if running in minimal shell (so that system binaries like `apt-get` and `update-ca-certificates` are found).
2. **Create keystore directory and clean old artifacts:** `mkdir -p /etc/ssl/certs/java` and remove any existing `/etc/ssl/certs/java/cacerts` file or symlink. This prevents the known missing-directory error and clears out any leftover broken symlink from previous attempts [4].

3. **Install OpenJDK 21 and required cert packages** *without triggering immediate config:* Run a single apt command to install Java 21 and the certificates packages: `openjdk-21-jdk-headless` (for the JDK, which includes the JRE), `ca-certificates`, and `ca-certificates-java`. Use `--no-install-recommends` to keep it lean (no GUI) and `-o Dpkg::Options::=--no-triggers` to defer running any triggers [7]. This ensures the installation doesn't fail mid-process. For example:

   ```
   apt-get install -yqq --no-install-recommends -o Dpkg::Options::=--no-triggers \
       openjdk-21-jdk-headless ca-certificates ca-certificates-java
   ```

   (The `-yqq` flags auto-approve and quiet the output; adjust verbosity as needed.)
4. **Run certificate triggers manually:** Now that all packages are installed but not fully configured, execute the certificate update triggers. Using `dpkg --triggers-only ca-certificates-java` will run the pending trigger for the Java certificates package (ignoring others). Follow that with `update-ca-certificates -f` to force-update the CA bundle and generate the Java keystore file [8]. These commands will populate `/etc/ssl/certs/java/cacerts` without errors.
5. **Link the JDK's cacerts (if not already linked):** Find the Java installation path (e.g. via `readlink -f $(which java)`) and ensure that `$JAVA_HOME/lib/security/cacerts` points to the system keystore. If not present, create a symlink:

   ```
   ln -s /etc/ssl/certs/java/cacerts "$JAVA_HOME/lib/security/cacerts"
   ```

   This step makes sure any Java process uses the updated truststore [9]. (On Ubuntu, the OpenJDK package usually sets this up, but it can be missing if triggers were skipped, so we fix it manually.)
6. **Verify the keystore:** (Optional) Use `keytool -list -keystore /etc/ssl/certs/java/cacerts -storepass changeit` to list the keystore entries, confirming that it's populated with dozens of default CAs (output can be piped to `head -n 5` just to show a few lines). This verifies success [10].

Example Installation Script

Below is a **bash script** implementing the above steps. It is designed to be **non-interactive and CI-safe** – it will not prompt for input and will exit on any failure (thanks to `-euo pipefail`). It uses only APT (apt-get) for installation and avoids any GUI components by using headless packages. You can include this in your CI setup to reliably install OpenJDK 21 in the Codex Ubuntu 24.04 environment:

```
#!/usr/bin/env bash
set -euo pipefail
export DEBIAN_FRONTEND=noninteractive
export PATH="/usr/sbin:/usr/bin:/sbin:/bin:$PATH"

# 1. Ensure the Java CA certs directory exists and is clean
mkdir -p /etc/ssl/certs/java
rm -f /etc/ssl/certs/java/cacerts   # remove any stale file or symlink if present
```

```
# 2. Install OpenJDK 21 (headless JDK) and certificate packages without running
triggers
apt-get update -qq
apt-get install -yqq --no-install-recommends -o Dpkg::Options::=--no-triggers \
    openjdk-21-jdk-headless ca-certificates ca-certificates-java

# 3. Manually trigger the Java CA certificate keystore generation
dpkg --triggers-only ca-certificates-java || true
update-ca-certificates -f

# 4. Ensure the JDK's security directory uses the system cacerts keystore
if command -v java >/dev/null 2>&1; then
  JAVA_BIN="$(readlink -f "$(command -v java)")"
  JAVA_HOME="$(dirname "$(dirname "$JAVA_BIN")")"
  if [[ -d "$JAVA_HOME/lib/security" && ! -e "$JAVA_HOME/lib/security/
cacerts" ]]; then
    ln -s /etc/ssl/certs/java/cacerts "$JAVA_HOME/lib/security/cacerts" || true
  fi
fi

# 5. Verify that the cacerts keystore has been created and populated
keytool -list -keystore /etc/ssl/certs/java/cacerts -storepass changeit | head -
n 5
```

This script will **guarantee a clean Java 21 installation** without the `ca-certificates-java` failure. Let's break down what it does:

- It pre-creates the target directory and removes any existing `cacerts` entry to avoid file-not-found errors [4].
- It installs OpenJDK 21 JDK (headless) along with `ca-certificates` and `ca-certificates-java` in one go, *but defers trigger execution* using `--no-triggers` [7]. We include `ca-certificates` (the base system cert bundle) explicitly to ensure it's present; this package usually is installed on Ubuntu by default, but we include it for completeness in minimal environments.
- It then runs the certificate update triggers manually: first targeting only the Java certificates package's trigger [11], and then running a full `update-ca-certificates` for good measure. This sequence creates the `/etc/ssl/certs/java/cacerts` file with proper CA entries.
- Next, it creates a symlink in the JDK's own directory if one isn't already there [9]. On Ubuntu, after the above steps, `$JAVA_HOME/lib/security/cacerts` should typically already point to `/etc/ssl/certs/java/cacerts` (or be that file). We double-check this; if the file is missing, we link it. This step ensures that if Java or Gradle tries to access the default keystore (usually via the JDK's `lib/security/cacerts` path), it will get the updated centralized keystore.
- Finally, it uses the `keytool` utility (part of the JDK) to list the keystore content [10]. The output should list multiple trusted CA certificates, confirming success. If this command fails or shows an error, it means something went wrong in the earlier steps (which would also cause the script to exit due to `set -e`).

Conclusion

By using the above method, you **avoid the** `ca-certificates-java` **bug** in a repeatable way. The key is to manage the timing of the certificate keystore generation: installing all packages first (headless JDK 21 and certs) and then running the update when the filesystem and Java tools are ready. This approach was verified in the Codex environment and recommended by the community [12] [8]. It does not rely on GUIs or interactive prompts, making it ideal for CI/CD pipelines.

With this setup, the WristLingo project's build requirements are met: Java 21 is properly installed for Kotlin and AGP, all without manual intervention. Gradle, the Android SDK (API 36 with Build Tools 36.0.0), and the NDK remain fully functional since Java is configured correctly. The solution minimizes additional packages by using the headless JDK and skipping recommended extras, reducing setup time and image size.

In summary, **the best practice** for installing OpenJDK 21 on Ubuntu 24.04 (Codex or similar cloud CI environments) is to preemptively handle the Java CA certificates setup. Manual directory creation, deferred trigger execution, and symlinking (as needed) ensure that `ca-certificates-java` can complete without errors and that your Java installation is robust and ready for secure operations. [1] [7]

Sources:

- Ubuntu ca-certificates-java bug report – missing directory causing keystore update failure [1] [4]
- OpenAI Codex discussion on Java 21 install issue and workaround script [2] [12]
- Related Stack Overflow Q&A on fixing OpenJDK install by ordering ca-certificates-java [5]
- Launchpad bug #1998065 on JRE installation triggers (fix for Java not found during ca-cert update) [3]

---

[1] [4] Bug #1895435 "update-ca-certificates fails due to no /etc/ssl/c..." : Bugs : ca-certificates-java package : Ubuntu
https://bugs.launchpad.net/ubuntu/+source/ca-certificates-java/+bug/1895435

[2] [6] [7] [8] [9] [10] [11] [12] installation of ca-certificates-java fails · openai codex · Discussion #1728 · GitHub
https://github.com/openai/codex/discussions/1728

[3] Bug #1998065 "JRE 19/Jammy: package ca-certificates-java 2019090..." : Bugs : ca-certificates-java package : Ubuntu
https://bugs.launchpad.net/ubuntu/+source/ca-certificates-java/+bug/1998065

[5] docker - head: cannot open '/etc/ssl/certs/java/cacerts' for reading: No such file or directory in Debian Image Java 11 - Stack Overflow
https://stackoverflow.com/questions/61305727/head-cannot-open-etc-ssl-certs-java-cacerts-for-reading-no-such-file-or-dir