

# ESDK Java 1.4.X

## INSTALLATION GUIDE

### 1 Overview

ESDK packages are specific to the product language (C/C++ or Java) and include both the ETA and EMA products. This *ESDK Java Installation Guide* describes procedures to install and build ESDK Java. These instructions apply to ESDK versions 1.2 and higher. Because steps to install are general to the ESDK, they apply to both ETA and EMA.

Developer may obtain ESDK Java Archive (JAR) files and their dependencies from the ESDK Java package. Starting with ESDK 1.2, JAR files are also hosted in Maven Central. Prior to ESDK 1.2, developers built their code using ANT (**build.xml**). Starting with ESDK 1.2, the ANT build is replaced by a build powered by Gradle (<https://gradle.org/>). In this release, you can use Gradle to do the following:

- If needed, download JAR files and associated dependencies from Maven Central (refer to Section 5)
- Build the ESDK package's Java examples (refer to Section 6) and product source code (refer to Section 5.2)

### 2 Obtaining the Package

You can obtain the Elektron SDK in either of the following ways:

- Download the package from the Developer Community Portal at the following URL: <https://developers.refinitiv.com/elektron/elektron-sdk-java/downloads>.
- Clone the ESDK from the GitHub repository (located at <https://github.com/Refinitiv/Elektron-SDK>) by using the following command:

```
git clone https://github.com/Refinitiv/Elektron-SDK.git
```



**Tip:** You can also download the package from GitHub via the browser:

- Browse to the URL <https://github.com/Refinitiv/Elektron-SDK/releases>

- Each release will have the following options listed beneath it's release name:



- To download a compressed package, click **zip** or **tar.gz**.

## 3 Gradle

The ESDK package includes **build.gradle** files throughout its directories to assist in building libraries and examples via a single command. After building the libraries and examples, you develop and compile your custom applications in the same manner as versions prior to ESDK 1.4.

**build.gradle** files specify the location of the product's Java dependencies (which can be local or remote). By default, the package's **build.gradle** files are set to pull its dependencies from locations on Maven Central (for details, refer to Section 5.1). If needed, you can configure Gradle files to pull dependencies (such as Apache or Mockito) from other URLs or locations.

# 4 Package File and Directory Changes

Notable changes in the ESDK package include:

- To support simultaneously hosting of files on Maven Central, JAR filenames have changed: all JAR files now include the package version (**Version**) as a suffix in their name (e.g., **upa-3.2.0.0.jar**). As a result of this change, you must update your class path(s) to use the appropriate filename(s).
- For multicast connections, you need the JNI libraries (same as previous ESDK versions) as well as the new **librsslRelMcast** libraries located in the **Eta/Libs/rssl** directory.
- Directory structure changes as follows:

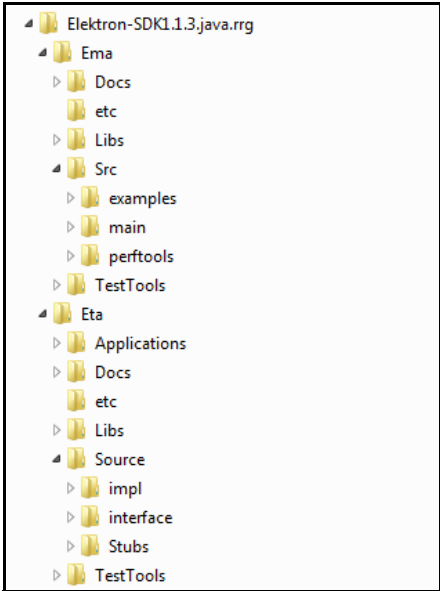
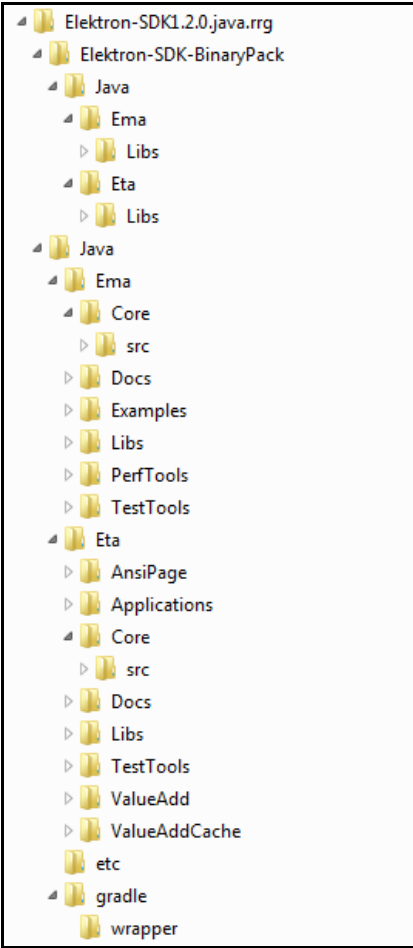
ELEKTRON SDK JAVA VERSION 1.1.3 PACKAGE	ELEKTRON SDK JAVA VERSION 1.3 PACKAGE
<p>The ESDK Java package prior to the version1.2 release, used the following high-level structure (included here for the sake of comparison):</p>  <pre>Elektron-SDK1.1.3.java.rrg ├── Ema │   ├── Docs │   │   └── etc │   ├── Libs │   └── Src │       ├── examples │       ├── main │       └── perftools ├── TestTools ├── Eta │   ├── Applications │   ├── Docs │   │   └── etc │   ├── Libs │   └── Source │       ├── impl │       ├── interface │       └── Stubs └── TestTools</pre>	<p>Starting with Version 1.2, the ESDK Java package (as obtained from the GSG) uses a directory structure that differs significantly from previous versions. To support Gradle, package directories have changed with introduction of a <b>Core</b> directory that stores the <b>src</b> directory content. The following diagram illustrates the new top-level directory structure.</p>  <pre>Elektron-SDK1.2.0.java.rrg ├── Elektron-SDK-BinaryPack │   ├── Java │   │   ├── Ema │   │   │   └── Libs │   │   ├── Eta │   │   │   └── Libs │   └── Java │       ├── Ema │       │   └── Core │       │       └── src │       ├── Docs │       ├── Examples │       ├── Libs │       ├── PerfTools │       └── TestTools ├── Eta │   ├── AnsiPage │   ├── Applications │   └── Core │       ├── src │       ├── Docs │       ├── Libs │       ├── TestTools │       ├── ValueAdd │       └── ValueAddCache ├── etc └── gradle     └── wrapper</pre>

Table 1: ESDK Java Package Structures

The ESDK package's version structure (illustrated in Table 1) is determined by how Gradle generates artifacts. Each subdirectory has its own **src** folder. All the source code in that folder can generate a jar file. Thus, the contents of the **ValueAdd** folder generate the **upaValueAdd-Version.jar**, the contents of the **Eta/Core** directory generate the **upa-Version.jar**, the contents of the **Ema/Core** directory generate the **ema-Version.jar**, and etc.

Starting in ESDK Version 1.2, a new library **librsslRelMcast** in directory **Elektron-SDK-BinaryPack/Java/Eta/Libs/rssl** accounts for the shared reliable multicast library. This library is dynamically loaded by **librssl** whenever the Reliable Multicast transport is selected.

Additionally, the DACS library was moved to directory **Elektron-SDK-BinaryPack/Java/Eta/Libs**.

## 5 Downloading Java Dependencies

For ease of product use, Thomson Reuters maintains its ESDK Jar files on Maven Central.

- Note:**
- ESDK JAR files dependencies (i.e., Apache, Mockito, etc) are maintained on Maven Central by their third party producers.
  - If you obtain the ESDK from GitHub, you must also clone the **Elektron-SDK-BinaryPack** to access ESDK non-open sourced JAR file dependencies (e.g., JDACS, JNI libraries, etc). If the **Elektron-SDK-BinaryPack** is absent, Gradle automatically clones this binary pack when compiling the ESDK source code.



**Tip:** ESDK packages obtained from Thomson Reuters's GSG or via the Developer Community Portal (<https://developers.thomsonreuters.com/elektron/elektron-sdk-java/downloads>) continue to include all needed Java dependencies: you do not need to use Gradle to download or build your ESDK dependencies. However, you will still need to run Gradle to build examples (refer to Section 6).

### 5.1 Downloading ESDK and Dependencies from Maven Central



**Tip:** If you retrieve the ESDK from GitHub, you can use the package's native Gradle files to download Java dependencies from Maven Central. For details, refer to Section 5.2.

Because you can download ESDK libraries and dependencies from Maven Central using several different tools, specific procedural instructions are not included here. Maven uses the following syntax to specify ESDK dependencies:

```
<dependency>
  <groupId>com.thomsonreuters.ema</groupId>
  <artifactId>ema</artifactId>
  <version>3.2.0.1</version>
</dependency>

<dependency>
  <groupId>com.thomsonreuters.upa</groupId>
  <artifactId>upa</artifactId>
  <version>3.2.0.1</version>
</dependency>

<dependency>
  <groupId>com.thomsonreuters.upa.valueadd</groupId>
  <artifactId>upaValueAdd</artifactId>
  <version>3.2.0.1</version>
</dependency>
```

```
<dependency>
  <groupId>com.thomsonreuters.upa.valueadd.cache</groupId>
  <artifactId>upaValueAddCache</artifactId>
  <version>3.2.0.1</version>
</dependency>

<dependency>
  <groupId>com.thomsonreuters.upa.ansi</groupId>
  <artifactId>ansipage</artifactId>
  <version>3.2.0.1</version>
</dependency>
```

Gradle uses the following syntax to specify ESDK dependencies:

```
compile group: 'com.thomsonreuters.ema', name: 'ema', version: '3.2.0.1'
compile group: 'com.thomsonreuters.upa', name: 'upa', version: '3.2.0.1'
compile group: 'com.thomsonreuters.upa.valueadd', name: 'upaValueAdd', version: '3.2.0.1'
compile group: 'com.thomsonreuters.upa.valueadd.cache', name: 'upaValueAddCache',
    version: '3.2.0.1'
compile group: 'com.thomsonreuters.upa.ansi', name: 'ansipage', version: '3.2.0.1'
```

## 5.2 Accessing Java Dependencies when Package is Cloned from GitHub

The ESDK package in GitHub does not include Java dependencies but includes all Gradle files necessary to download and build these dependencies from Maven Central.



**Warning!** To run Gradle, you must:

- Have access to the Internet
  - Specify any proxy (i.e., a firewall) that you use on your network. For instructions on specifying a proxy, refer to Gradle's instructions at the following link: [https://docs.gradle.org/current/userguide/build\\_environment.html#sec:accessing\\_the\\_web\\_via\\_a\\_proxy](https://docs.gradle.org/current/userguide/build_environment.html#sec:accessing_the_web_via_a_proxy).
- 

### ► To download and build dependencies using Gradle:

1. Clone the ESDK package from GitHub (<https://github.com/thomsonreuters/Elektron-SDK>).
2. Open a command (on Windows) or terminal (on Linux) window.
3. Change your directory to the ESDK Java root directory (i.e., **Elektron-SDK/Java**).
4. Issue the appropriate Gradle command as follows:
  - On Windows, issue the command: `gradlew.bat jar`
  - On Linux, issue the command: `./gradlew jar`

## 6 Building Examples

The ESDK requires that you use Gradle to build the Java examples for both EMA and ETA. Because the ESDK comes with a large number of examples, this section discusses how to use Gradle to access the entire list of examples, and basic syntax in running an example.

---

**Note:** To use Gradle, you must have access to the Internet.

---

### 6.1 Listing ESDK Examples

The following procedure assumes that you've already downloaded, and if necessary, built the package's JAR files and dependencies (refer to Section 5).

Before running an ETA or EMA example, you need to know the example's name (for details on using Gradle to run examples, refer to Section 6.3). You can use Gradle to list all ETA and EMA example names.

The following diagram is an example of what Gradle prints to the screen (EMA and ETA examples display under the section **Other tasks** toward the end of the command's output):

```
Run EMA Consumer Application Examples tasks
-----
runconsumer100 - This task runs the Consumer example100__MarketPrice__Streaming
runconsumer101 - This task runs the Consumer example101__MarketPrice__QosPriority
runconsumer102 - This task runs the Consumer example102__MarketPrice__Snapshot
runconsumer110 - This task runs the Consumer example110__MarketPrice__FileConfig
runconsumer111 - This task runs the Consumer example111__MarketPrice__UserSpecifiedFileConfig
runconsumer112 - This task runs the Consumer example112__MarketPrice__TunnelingConnection
runconsumer120 - This task runs the Consumer example120__MarketPrice__FieldListWalk
runconsumer130 - This task runs the Consumer example130__MarketPrice__UserDisp
runconsumer140 - This task runs the Consumer example140__MarketByOrder__Streaming
runconsumer200 - This task runs the Consumer example200__MarketPrice__Streaming
runconsumer210 - This task runs the Consumer example210__MarketByOrder__Streaming
runconsumer220 - This task runs the Consumer example220__MarketByPrice__PrivateStream
```

Figure 1. Gradle Output with EMA Consumer Example Names

#### ► Using Gradle to list EMA and ETA examples:

1. Open a command (on Windows) or terminal (on Linux) window.
2. Change your directory to the ESDK root directory.
  - For packages downloaded from GSG or the Developer Portal, the package directory is named **Elektron-SDKVersion.java.rrg**, where **Version** is represented by 3 digits and a letter (e.g., **Elektron-SDK1.3.0.L1.java.rrg**), while the ESDK root directory is **Elektron-SDKVersion.java.rrg/Java**.
  - For packages pulled from GitHub, the ESDK root directory is **Elektron-SDK/Java**.
3. To view the list of ETA examples, issue the appropriate command as follows:
  - On Windows, issue the command: **gradlew.bat Eta:Applications:tasks --all**
  - On Linux, issue the command: **./gradlew Eta:Applications:tasks --all**
4. To view the list of EMA examples, issue the appropriate command as follows:
  - On Windows, issue the command: **gradlew.bat Ema:Examples:tasks --all**
  - On Linux, issue the command: **./gradlew Ema:Examples:tasks --all**



## 6.2 Enabling Logging in EMA

In EMA, to enable logging when running examples, you must activate the logging section in the **build.gradle** file in the **PackageDirectory/Ema/Examples** directory. Remove the forward slashes (//) from the **jvmArgs** line as follows:

```
tasks.withType ( JavaExec ) {  
    // to enable the logger comment out the line below  
    jvmArgs += "-Djava.util.logging.config.file=../Core/src/main/resources/  
        logging.properties"  
}
```

## 6.3 Building and Running an Example Using Gradle

The following procedure assumes that you've already identified the name of the example you want to run (for details, refer to Section 6.1).

### ► Using Gradle to build and run an example:

1. Open a command (on Windows) or terminal (on Linux) window.
2. Change your directory to the ESDK root directory (i.e., **Elektron-SDKVersion.java.rrg/Java**).

Where **Version** is represented by 3 digits and a letter (e.g., **Elektron-SDK1.3.0.L1.java.rrg**).

3. To build and run an example, issue the appropriate command as follows:
  - On Windows, issue the command: **gradlew.bat runExampleName [--args="arguments"]**
  - On Linux, issue the command: **./gradlew runExampleName [--args="arguments"]**

Where:

- **runExampleName** is the name of the example you want to build and run. For example, issuing the command: **gradlew.bat runconsumer270** runs the EMA consumer **example270\_\_SymbolList**.
- **arguments** are options or arguments that you want to add to the Gradle command. The following is a Linux command illustrating the use of arguments when running Gradle:

```
./gradlew runVaConsumer --args="-c localhost:14002 DIRECT_FEED mp:TRI"
```



### Tip:

- You can see a list of all possible arguments by passing the command: **--args="-?"**
- Instead of **--args**, you can use **--PcommandLineArgs**, which functions in an identical manner with the same arguments..

## 7 Additional Resources

Encountering unique situations and scenarios is commonplace when using APIs in new and different ways, and in the face of different IDEs and build environments. For this reason, not every scenario can be addressed in this migration guide. For further information, tips, advice, etc., feel free to reach out to the wider open source community via the forums at the [Developer Community Portal](#). The community also includes tutorials and other getting started details.