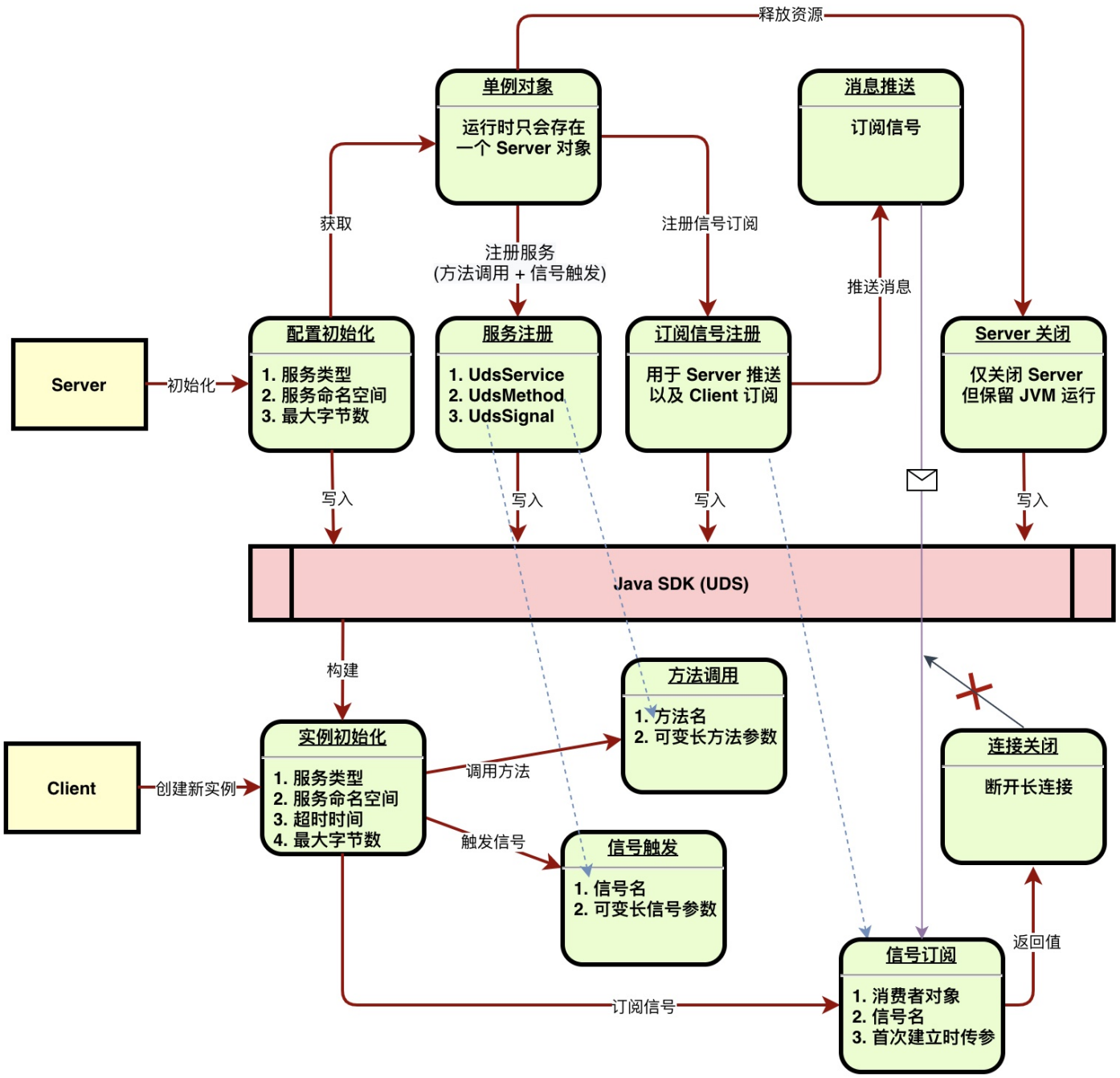
微服务 SDK（Java 版本）

本 SDK 支持 Linux + MacOS 中运行, **不支持** Windows。

文档版本

版本号	编辑人	日期	说明
v1.0.0	吴仙杰	2021.4.13	初始版本
v1.0.1	吴仙杰	2021.4.21	降级 JDK 版本, API 参数调整

SDK 流程图



JDK 版本

JDK ≥ 1.8

SDK 示例程序

> 本节的示例程序可通过以下方式直接运行 (其中将 `<version>` 替换为所使用的版本):
>
> - Server 端运行方式: `java -jar uds-sdk-<version>.jar server test`
> - Client 端运行方式: `java -jar uds-sdk-<version>.jar client test`
>
> 提示: 可运行该包验证 Java UDS (Unix Domain Docket) SDK 是否对当前操作系统有效

服务对象类

```
package com.qgschina.udssdk.test.service;

import com.qgschina.udssdk.server.annotation.UdsMethod;
import com.qgschina.udssdk.server.annotation.UdsService;
import com.qgschina.udssdk.server.annotation.UdsSignal;
import lombok.Data;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 使用 {@link UdsService} 注解, 标注该对象为一个服务对象
 */
@UdsService
public class TestService {

  /**
   * 注册方法为"方法调用"
   *
   * <ul>
   *   <li>使用 {@link UdsMethod} 注解, 标注该方法为"方法调用"</li>
   *   <li>Client 可通过方法名 {@code echo} 调用该方法, 并得到返回值</li>
   * </ul>
   *
   * 注意: 若返回值存在 <b>{@code byte[]}</b>, 则会被转为 Base64 字符串再给 Client
   *
   * @param bytes 字节数组
   * @param list 列表
   * @return Map, 包含字节数组
   */
  @UdsMethod("echo")
  public Map<String, Object> echoMsg(byte[] bytes,
      List<Map<String, Object>> list) {
    HashMap<String, Object> map = new HashMap<String, Object>() {{
      put("byteArray", new HashMap<String, Object>() {{
        put("bytes", bytes);
        put("string", new String(bytes));
      }});
      put("list", list);
    }};

    printMsg(map);

    return map;
  }

  /**
   * 注册方法为"信号触发"
   *
   * <ul>
   *   <li>使用 {@link UdsSignal} 注解, 标注该方法为"信号触发"</li>
   *   <li>Client 可通过信号名 {@code trigger} 异步调用该方法, 且无需返回值</li>
```

```
     * </ul>
     *
     * @param art POJO
     * @param score 浮点数
     * @throws InterruptedException 可忽略
     */
    @UdsSignal("trigger")
    public void fire(Artifact art, double score) throws InterruptedException {
        // 模拟 2 秒耗时操作
        Thread.sleep(2 * 1000);

        System.out.printf("信号触发 --> 工件: %s, 版本: %d, 评分: %f\n",
            art.getName(), art.getVersion(), score);
    }

    /**
     * 未使用 {@link UdsMethod} 或 {@link UdsSignal} 注解, 故 Client 不可调用该方法
     *
     * @param map Map
     */
    public void printMsg(Map<String, Object> map) {
        System.out.println("方法调用 --> " + map);
    }

    @Data
    public static class Artifact {

        private String name;
        private Integer version;
    }
}
```

测试主类

```
package com.qgschina.udssdk.test;

import com.qgschina.udssdk.client.Client;
import com.qgschina.udssdk.client.constant.UdsCode;
import com.qgschina.udssdk.client.model.UdsResult;
import com.qgschina.udssdk.common.constant.ServiceType;
import com.qgschina.udssdk.common.model.NamespaceResultData;
import com.qgschina.udssdk.server.Server;
import com.qgschina.udssdk.test.service.TestService;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Objects;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

/**
 * 用于对 UDS 服务的一个可用性测试类, 包含 Server 端和 Client 端
 * <p>
 * 注意: 本 SDK 仅支持在 Linux 下运行
 */
public class MainTest {

    /**
     * 假定一个用于"信号订阅"的信号名
     */
    private static final String SUB_SIGNAL = "sub_shutdown";

    /**
     * 用于让主线程有机会等待其他线程执行完后再执行
     */
    private static final CountDownLatch latch = new CountDownLatch(1);
```

```java
/**
 * Main 方法
 *
 * @param args 命令行参数,
 *             参数 1: {@code [server|client]},
 *             参数 2: {@code namespace}
 * @throws InterruptedException 可忽略
 */
public static void main(String[] args) throws InterruptedException {
  if (args == null || args.length != 2) {
    System.out.println("运行方式 :)\n"
        + "java -jar uds-sdk-<version>.jar "
        + "<server/client> <namespace>");
    return;
  }

  String runType = args[0];
  String namespace = args[1];

  if (Objects.equals("server", runType)) {
    // SDK Server 端使用方式
    useServer(namespace);
    return;
  }

  if (Objects.equals("client", runType)) {
    // SDK Client 端使用方式
    useClient(namespace);
    return;
  }

  System.out.println("参数错误 :(");
}

/**
 * SDK Server 端使用方式
 *
 * @param namespace 用于开启服务的命名空间
 * @throws InterruptedException 可忽略
 */
private static void useServer(String namespace) throws InterruptedException {
  // 初始化 Server 配置, 参数分别为:
  // 1. 服务类型
  // 2. 服务子命名空间
  // 3. 允许数据传输的最大字节数, 单位: MB
  Server.init(ServiceType.CUSTOM, namespace, 10);

  // 获取 Server 单例对象
  Server server = Server.getInstance();

  // 注册服务对象 (方法调用及信号触发)
  server.registerService(new TestService());

  // 注册信号订阅
  server.registerSubSignal(SUB_SIGNAL);

  System.out.println("模拟仅运行 10 秒的 Servlet 容器...");
  boolean countDown0 = latch.await(10, TimeUnit.SECONDS);
  System.out.printf("10 秒已到 (%s), 模拟关闭 Servlet 容器\n", !countDown0);

  System.out.println("消息推送: 服务即将关闭");
  // 消息推送
  server.send(SUB_SIGNAL, "服务即将关闭");

  // 关闭 Server
  // 注意: 通常 Server 都不是通过自身程序关闭的, 故执行本方法并不会关闭 JVM
  server.stop();

  // 模拟正常关闭 JVM
  System.exit(0);
```

```
}

/**
 * SDK Client 端使用方式
 *
 * @param namespace 用于连接服务的命名空间
 * @throws InterruptedException 可忽略
 */
private static void useClient(String namespace) throws InterruptedException {
  // 创建新的 Client, 参数分别为:
  // 1. 服务类型
  // 2. 服务子命名空间
  // 3. 等待 Server 端返回的超时时间, 单位: 秒 (对订阅不生效)
  // 4. 允许数据传输的最大字节数, 单位: MB
  Client client = new Client(ServiceType.CUSTOM, namespace, 10, 10);

  // 信号订阅 (长连接)
  System.out.println("信号订阅 --> ");
  UdsResult<Client.UdsConnection> subResult = client
      .subSignal(obj -> {
        System.out.println("\n>>>>>>>>>>>>>>>>>>>>>>>>");
        System.out.println("订阅结果: " + obj);
        System.out.println("<<<<<<<<<<<<<<<<<<<<<<<<\n");
      }, SUB_SIGNAL);
  checkUdsCode(subResult);

  // 获取当前命名空间下的所有可被方法调用的方法
  System.out.println("所有方法 --> ");
  UdsResult<NamespaceResultData> allMethods = client.getAllMethods();
  checkUdsCode(allMethods);

  // 获取当前命名空间下的所有可被触发或订阅的信号
  System.out.println("所有信号 (触发和订阅) --> ");
  UdsResult<NamespaceResultData> allSignals = client.getAllSignals();
  checkUdsCode(allSignals);

  // 获取当前命名空间下的所有可被调用的方法, 及可被触发或订阅的信号
  System.out.println("所有方法和信号 --> ");
  UdsResult<NamespaceResultData> allMethodsAndSignals =
      client.getAllMethodsAndSignals();
  checkUdsCode(allMethodsAndSignals);

  // 方法调用
  System.out.println("方法调用 --> ");
  UdsResult<Object> resultEcho =
      client.callMethod("echo",
          "Java 微服务 SDK v0.0.1".getBytes(StandardCharsets.UTF_8),
          new ArrayList<String>() {{
            add("C++");
            add("Java");
            add("Python");
          }});
  checkUdsCode(resultEcho);

  // 信号触发
  System.out.println("信号触发 --> ");
  UdsResult<Void> resultSignal =
      client.triggerSignal("trigger",
          new HashMap<String, Object>() {{
            put("name", "Java 开发包");
            put("version", 1);
          }}, 99.9999999);
  checkUdsCode(resultSignal);

  System.out.println("模拟一个仅运行 12 秒的 Client 服务");
  boolean reached0 = latch.await(12, TimeUnit.SECONDS);
  System.out.printf("12 秒已到 (%s), 模拟关闭 Client 服务\n", !reached0);

  // 关闭与 Server 端建立的信号订阅长连接
  subResult.getData().disconnectSub();
```

```
  }

  /**
   * 检查 UDS 调用返回结果的响应码
   *
   * @param result UDS 返回结果
   */
  private static void checkUdsCode(UdsResult<?> result) {
    if (result.getCode() == UdsCode.SUCCESS) {
      System.out.println("正常返回: " + result);
      System.out.println("=======================\n");
      return;
    }
    if (result.getCode() == UdsCode.NOT_CONNECTED) {
      throw new RuntimeException("未连接: " + result);
    }
    if (result.getCode() == UdsCode.OVER_TIME) {
      throw new RuntimeException("超时: " + result);
    }
    if (result.getCode() == UdsCode.METHOD_CALL_ERROR) {
      throw new RuntimeException("函数调用失败: " + result);
    }
    if (result.getCode() == UdsCode.SIGNAL_SUB_ERROR) {
      throw new RuntimeException("信号订阅失败: " + result);
    }
  }
}
```

包介绍

- `com.qgschina.udssdk.server`：用于 Server 端的相关包及类
- `com.qgschina.udssdk.client`：用于 Client 端的相关包及类
- `com.qgschina.udssdk.common`：可用于 Server 端及 Client 端的相关包及类
- `com.qgschina.udssdk.test`：可直接运行的测试主类，可用于判断 Java UDS SDK 是否支持当前操作系统

API

Server 端

`com.qgschina.udssdk.server.Server`

```
/**
 * 初始化 Server 配置
 * <p>
 * 注意: 该方法只会在 {@link Server#getInstance()} 前生效一次
 *
 * @param type       服务类型
 * @param namespace  服务子命名空间
 * @param maxBytesMb 允许数据传输的最大字节数, 单位: MB
 */
public static void init(ServiceType type, String namespace, int maxBytesMb) {
}

/**
 * 获取 Server 单例对象
 * <p>
 * 必须先执行 {@link Server#init} 完成初始化配置
 *
 * @return Server 单例对象
 */
public static Server getInstance() {
}

/**
```

```
     * 注册服务
     *
     * <ul>
     *   <li>{@link UdsService}: 注解在类名上, 标识服务</li>
     *   <li>{@link UdsMethod}: 注解在方法上, 标识方法调用的方法名</li>
     *   <li>{@link UdsSignal}: 注解在方法上, 标识信号触发的信号名</li>
     * </ul>
     *
     * 注意: 若方法返回值存在 <b>{@code byte[]}</b>,
     * 则会被转为 Base64 字符串再给 Client
     *
     * @param service 拥有特定注解的服务实例化对象
     */
    public void registerService(Object service) {
    }

    /**
     * 注册信号订阅
     *
     * <ul>
     *   <li>Server 进行消息推送的信号名</li>
     *   <li>Client 进行信号订阅的信号名</li>
     * </ul>
     *
     * @param signal 注册为信号订阅的信号名
     */
    public void registerSubSignal(String signal) {
    }

    /**
     * 消息推送
     * <p>
     * 推送消息给 Client
     * <p>
     * 注意: 若数据存在 <b>{@code byte[]}</b>, 则会被转为 Base64 字符串再给 Client
     *
     * @param signal 注册为信号订阅的信号
     * @param data   推送给 Client 的数据
     */
    public void send(String signal, Object data) {
    }

    /**
     * 关闭服务
     * <p>
     * 注意: 通常 Server 都不是通过自身程序关闭的, 故执行本方法并不会关闭 JVM
     */
    public void stop() {
    }
```

Client 端

`com.qgschina.udssdk.client.Client`

```
    /**
     * 创建一个新的 Client 实例
     *
     * @param type       服务类型
     * @param namespace  服务子命名空间
     * @param timeout    等待 Server 端返回的超时时间, 单位: 秒 (对订阅不生效)
     * @param maxBytesMb 允许数据传输的最大字节数, 单位: MB
     */
    public Client(ServiceType type, String namespace, int timeout,
        int maxBytesMb) {
    }

    /**
     * 信号订阅 (长连接)
```

```
     * <p>
     * 当 Server 对指定信号进行消息推送时, 就会执行 {@code consumer}
     *
     * @param consumer 当 Server 端有返回结果时的消费者
     * @param signal   需要进行"信号订阅"的信号名
     * @param args     传递给 Server 的参数, 可能在首次建立订阅时需要
     */
    public UdsResult<UdsConnection> subSignal(
        Consumer<UdsResult<Object>> consumer,
        String signal, Object... args) {
    }

    /**
     * 获取当前命名空间下的所有可被方法调用的方法
     *
     * @return 所有可被方法调用的方法
     */
    public UdsResult<NamespaceResultData> getAllMethods() {
    }

    /**
     * 获取当前命名空间下的所有可被触发或订阅的信号
     *
     * @return 所有可被触发或订阅的信号
     */
    public UdsResult<NamespaceResultData> getAllSignals() {
    }

    /**
     * 获取当前命名空间下的所有可被调用的方法, 及可被触发或订阅的信号
     *
     * @return 所有可被调用的方法, 及可被触发或订阅的信号
     */
    public UdsResult<NamespaceResultData> getAllMethodsAndSignals() {
    }

    /**
     * 方法调用
     * <p>
     * 同步调用某个方法
     *
     * @param method 方法名
     * @param args   可变长方法参数
     * @return 执行指定方法后的返回结果
     */
    public UdsResult<Object> callMethod(String method, Object... args) {
    }

    /**
     * 信号触发
     * <p>
     * 异步调用某个方法, 且不存在返回值
     *
     * @param signal 信号名
     * @param args   可变长信号参数
     * @return 信号是否触发成功
     */
    public UdsResult<Void> triggerSignal(String signal, Object... args) {
    }
```

`com.qgschina.udssdk.client.model`

```
    package com.qgschina.udssdk.client.model;

    import com.qgschina.udssdk.client.constant.UdsCode;
    import lombok.Data;

    /**
```

```
 * UDS 调用返回结果
 *
 * @param <T> 最终返回值类型
 */
@Data
public class UdsResult<T> {

  /**
   * 响应码
   */
  private UdsCode code;

  /**
   * 提示消息
   */
  private String message;

  /**
   * 返回值
   */
  private T data;
}
```

`com.qgschina.udssdk.client.Client.UdsConnection`

```
/**
 * 关闭建立的信号订阅长连接
 */
public void disconnectSub() {
}
```

`com.qgschina.udssdk.common.model.NamespaceResultData`

```
package com.qgschina.udssdk.common.model;

import java.util.List;
import lombok.Data;

/**
 * 获取地址空间返回时的数据结果包装对象
 */
@Data
public class NamespaceResultData {

  /**
   * 可调用方法列表
   */
  private List<NamespaceResultDataItem> method;

  /**
   * 可调用方法数量
   */
  private Integer methodNum;

  /**
   * 可触发信号列表
   */
  private List<NamespaceResultDataItem> signal;
}
```

`com.qgschina.udssdk.common.model.NamespaceResultDataItem`

```
package com.qgschina.udssdk.common.model;

import java.util.List;
```

```java
import lombok.Data;

/**
 * 获取地址空间返回时的具体数据项
 */
@Data
public class NamespaceResultDataItem {

  /**
   * 方法名或信号号
   */
  private String name;

  /**
   * 参数名列表
   */
  private List<String> parameterNames;

  /**
   * 参数类型列表
   */
  private List<String> parameterTypes;
}
```

数据类型说明

因 SDK 需要兼容多语言, 故方法参数及返回值仅允许以下几种数据类型:

- 字符串：`String`
- 数字：`Integer`（`int`）及 `Long`（`long`）
- 浮点数：`Double`（double）及 Float（float`）
- 布尔值：`Boolean`（`boolean`）
- 字典：`Map` 及 POJO
- 列表：`List`，Java 中数组用作可变长参数，故 SDK 不允许除了 `byte[]` 之外的数组类型

对于 `byte[]` 会有两种处理方式（参考示例代码 `TestService#echoMsg(byte[], List<Map<String, Object>>)`）：

- 作为方法参数：`byte[]` 仍然还是 `byte[]`
- 作为方法返回值：`byte[]` 会被转为 Base64 再给 Client，即 Client 得到的是 Base64 字符串

安装方式

SDK 压缩包（`uds-sdk-<version>.tar.gz`）解压后的目录结构:

```
.
├── lib # Java UDS SDK 的依赖包
│   ├── jackson-annotations-2.12.2.jar
│   ├── jackson-core-2.12.2.jar
│   ├── jackson-databind-2.12.2.jar
│   ├── netty-buffer-4.1.60.Final.jar
│   ├── netty-codec-4.1.60.Final.jar
│   ├── netty-codec-dns-4.1.60.Final.jar
│   ├── netty-codec-http-4.1.60.Final.jar
│   ├── netty-codec-socks-4.1.60.Final.jar
│   ├── netty-common-4.1.60.Final.jar
│   ├── netty-handler-4.1.60.Final.jar
│   ├── netty-handler-proxy-4.1.60.Final.jar
│   ├── netty-resolver-4.1.60.Final.jar
│   ├── netty-resolver-dns-4.1.60.Final.jar
│   ├── netty-resolver-dns-native-macos-4.1.60.Final-osx-x86_64.jar
│   ├── netty-transport-4.1.60.Final.jar
│   ├── netty-transport-native-epoll-4.1.60.Final-linux-x86_64.jar
│   ├── netty-transport-native-kqueue-4.1.60.Final-osx-x86_64.jar
```

```
│   ├── netty-transport-native-unix-common-4.1.60.Final.jar
│   ├── reactive-streams-1.0.3.jar
│   ├── reactor-core-3.4.4.jar
│   ├── reactor-netty-core-1.0.5.jar
│   └── slf4j-api-1.7.30.jar
├── pom.xml # Maven pom 文件
├── uds-sdk-<version>.jar # 本文档所描述的 Java UDS SDK
└── 跨语言微服务_Java_SDK.pdf # 本文档
```

Maven

假设已位于 SDK 压缩包解压后的路径中，执行以下命令：

```
mvn install:install-file -Dfile=uds-sdk-<version>.jar -DpomFile=pom.xml
```

原始方式

手动将 SDK 压缩包解压后 `lib` 中的所有 `.jar` 及 `uds-sdk-<version>.jar` 加入到你项目的 classpath 中即可。