BaaP SDK使用说明文档

我要开发合约

1. maven工程创建pom.xm配置

配置Maven repo

```
<repository>
    <id>pdx-release</id>
    <name>biz.pdxtech</name>
    <url>http://baap.pdx.life:8081/nexus/content/repositories/releases  </url>
</repository>
```

配置PDX依赖包`baap-api`和`baap-setting`(最新版本请咨询开发员)

```
<dependency>
    <groupId>biz.pdxtech.baap</groupId>
    <artifactId>baap-api</artifactId>
    <version>${sdk.version}</version>
</dependency>
```

```
<dependency>
    <groupId>biz.pdxtech.baap</groupId>
    <artifactId>baap-setting</artifactId>
    <version>${sdk.version}</version>
</dependency>
```

2. BaaP合约接口说明

开发合约接口，实现接口`biz.pdxtech.baap.api.contract.IContract`，其中必须实现的方法为

> 交易执行方法
>
> ```
> @POST
> @Path("/exec")
> @Produces({MediaType.APPLICATION_JSON})
> @Consumes({"application/pdx-baap"})
> TransactionResp exec(@Context BaapContext ctx, Transaction tx);
> ```

此外可以选择性实现

> 合约初始化方法，在合约启动之后会被**BaaP**主动调用

```
@POST
@Path("/init")
@Consumes({"application/pdx-baap"})
default void init(@Context BaapContext ctx) {
}
```

合约查询方法

```
@POST
@Path("/query")
@Consumes({"application/pdx-baap"})
@Produces({MediaType.APPLICATION_OCTET_STREAM})
default InpuStream query(@Context BaapContext ctx, byte[] qstr) {
}
```

合约回滚方法

```
@POST
@Path("/undo")
@Produces(MediaType.APPLICATION_JSON)
@Consumes({"application/pdx-baap"})
default TransactionResp undo(@Context BaapContext ctx, String txid) {
}
```

合约终止方法

```
@POST
@Path("/fini")
@Consumes({"application/pdx-baap"})
default void fini(@Context BaapContext ctx) {
}
```

3. 开发一个简单的合约项目

开发合约接口`MyIContract`如下，其中`MyIContract`是符合**JAX-RS(JSR311)**标准的**RESTFULL WebService**；`BaaP`限使用`Jersey`框架

```
@Path("/sample")
public class MyContract implements IContract {
    private static Logger logger = LoggerFactory.getLogger(MyContract.class);

    @Override
    @POST
    @Path("/query")
    @Produces({MediaType.APPLICATION_OCTET_STREAM})
    @Consumes({"application/pdx-baap"})
    public InputStream query(@Context BaapContext ctx, byte[] qstr) {
        return new ByteArrayInputStream(qstr);
    }

    @Override
    @POST
    @Path("/exec")
    @Produces({MediaType.APPLICATION_JSON})
    @Consumes({"application/pdx-baap"})
    public TransactionResp exec(@Context BaapContext ctx, Transaction tx) {

        logger.info(ctx.txid());
        TransactionResp resp = new TransactionResp();
        resp.setTxId(ctx.txid());

        resp.putState("a", "a".getBytes())
                .putState("b", "b".getBytes());

        resp.addNotif(new Event(URI.create("contract:///?filter=" +
URLEncoder.encode("dst=='daap://default_pdx_chain/45874a3c0afc2a4d6cc9dea20245350f2981
d3ea/sample'"
)), "abcdefg".getBytes()));

        if (ctx.stream() != null) {
            ctx.stream().forEach((k, v) -> {
                System.out.println(":::" + k);
                try {
                    System.out.println(":::" + v);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            });
        }

        System.out.println(new String(ctx.ext().state("a", ctx.dst(),
"cf47cd481a483abd8a0202b13509da6c2b32b3973cf04019fe924838ca4f591f"
)));

        return resp;
    }
}
```

设置**Jersey**全局配置

```
register(new BaapContextBinder())
    .register(MultiPartFeature.class).register(JacksonFeature.class)
    .register(TransactionReader.class);
```

设置**J2EE Web**项目的**web.xml**

```
<filter>
   <filter-name>BaapFilter</filter-name>
   <filter-class>biz.pdxtech.baap.setting.filter.BaapFilter </filter-class>
</filter>

<filter-mapping>
  <filter-name>BaapFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<servlet>
  <servlet-name>BaapServlet</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer </servlet-class>
  <init-param>
    <param-name>javax.ws.rs.Application</param-name>
    <param-value>biz.pdxtech.baap.sample.icontract.MyApplication </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>BaapServlet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

4. BaaP API使用说明

`BaapContext`:

> *BaapContext* 封装了BaaP对用户开发的工具支持;
>
> ```
> String txid(); // 获取当前执行的交易ID
> String oobmid();// 获取oobm数据流id
> Map<String, String> stream();// 获取当前执行交易的数据流 k为流名称 v为路径
> String dst();// 获取当前合约
> String spuk(); // 获取发送者的公钥
> String meta(String key);// 获取其他属性
> BaapContextExt ext();// 获取扩展上下文工具，详见下扩展方法
> byte[] state(@Nonnull String key, @Nullable String txid); // 获取参数执行的Key最新状
> 态或者某个txid执行后的最新状态
> Map<String, InputStream> stream();
> String node(); // 获取当前执行节点的节点id
> String chain();// 获取当前链
> ```

`BaapContextExt`:

> *BaapContextExt*封装了BaaP对用户开发的扩展工具支持；

```
String chain();// 获取链信息
String node();// 获取节点信息
String npuk();// 获取节点公钥
byte[] state(@Nonnull String key, @Nonnull String dst, @Nullable String txid);// 获取指定合约或者指定合约指定交易的最新状态
Txstatus txstatus(@Nonnull String txid);// 获取交易执行状态
Nodedesc nodedesc();// 获取当前节点信息(含公钥、块信息、链、名称、ip、类型)
String stx(@Nonnull Transaction tx);// 向同一Bapp发送交易
Pair<Boolean, String> vtx(@Nonnull Transaction tx);// 验证是否链上发来的交易，一般与stx一起使用
byte[] pull(int slice, String node, String desc);// 从某节点上的同一Bapp拉去数据
int push(byte[] slice, String node, String desc);// 向某节点上的同一Bapp推送数据
```

`TransactionResp`：

> *TransactionResp* 封装了用户对BaaP回复的支持；其中重要属性

```
private String status;// 交易执行用户定义状态
private String reason;// 交易执行用户定义原因
private Map<String, byte[]> state;// 用户合约执行状态的封装，将会存储到区块链，且决定交易执行的一致性
private List<Event> notif;// 合约执行通知的封装，BaaP将针对用户指定的Notify进行消息通知
```

我要发送交易

1. maven工程创建pom.xm配置

配置Maven repo

```
<repository>
    <id>pdx-release</id>
    <name>biz.pdxtech</name>
    <url>http://baap.pdx.life:8081/nexus/content/repositories/releases  </url>
</repository>
```

配置PDX依赖包`baap-api`和`baap-driver-ethereum`

```
<dependency>
    <groupId>biz.pdxtech.baap</groupId>
    <artifactId>baap-api</artifactId>
    <version>${sdk.version}</version>
</dependency>
```

```
<dependency>
    <groupId>biz.pdxtech.baap</groupId>
    <artifactId>baap-driver-ethereum</artifactId>
    <version>${sdk.version}</version>
</dependency>
```

2. BaaP驱动接口说明

`biz.pdxtech.baap.driver.BlockChainDriver` 包含以下方法：

> 查询合约方法，对应IContract的query方法

```
byte[] query(@Nonnull String dst, @Nonnull byte[] qstr);
```

> 执行交易方法，对应IContract的exec方法

```
String apply(@Nonnull Transaction tx) throws BlockChainDriverException ;
String apply(@Nonnull Transaction tx, @Nonnull Map<String, InputStream> stream)
throws BlockChainDriverException ;
```

> 订阅**通知**方法

```
default BlockChainDriver subscribe(@Nonnull BlockChainListener<Event> listener,
@Nonnull URI... subject) throws BlockChainDriverException {
        return this;
}// 订阅
default BlockChainDriver unsubscribe(@Nonnull URI... subject) throws
BlockChainDriverException {
        return this;
}// 取消订阅
```

> 释放**driver**资源

```
void fini();
```

3. 开发一个简单的BaaP驱动程序

驱动程序`MyDriver`如下：

```
public static void main(String[] args) throws BlockChainDriverException ,
InvalidAlgorithmParameterException , NoSuchAlgorithmException , IOException {
        // init BlockChainDriver
        Properties properties = new Properties();
        properties.setProperty("baap-chain-id", Constants.BAAP_CHAIN_ID_DEFAULT );
        properties.setProperty("baap-chain-type", Constants.BAAP_CHAIN_TYPE_ETHEREUM );
        properties.setProperty("baap-url", "http://127.0.0.1:8080/" );
        properties.setProperty("baap-private-key",
"510c37d6ed45d8fd179276bfd785b610dac329ee578b245e9f693a1e1bd34065"  );
        BlockChainDriver driver = BlockChainDriverFactory.get(properties);

        // subscribe demo
        subscribe(driver);
        // apply tx
        apply(driver);
        // apply tx with stream
        applyWithStream(driver);
        // query
        query(driver);

        driver.fini();

    }

    private static void subscribe(BlockChainDriver driver) throws
BlockChainDriverException {
        driver.subscribe(msg -> {
            System.out.println("###############" + msg.getTopic());
            System.out.println("###############" + new String(msg.getData()));
        }, URI.create("contract:///?dst=" +
URLEncoder.encode("daap://default_pdx_chain/45874a3c0afc2a4d6cc9dea20245350f2981d3ea/s
ample")));
    }

    private static void apply(BlockChainDriver driver) throws
BlockChainDriverException {
        long t = System.currentTimeMillis();
        String txid = driver.apply(createTx());
        System.out.println(txid);
        System.out.println(System.currentTimeMillis() - t);
    }

    private static void applyWithStream(BlockChainDriver driver) throws
BlockChainDriverException , FileNotFoundException {
        Map<String, InputStream> stream = new HashMap<>();
        stream.put("a", new FileInputStream("/home/bochenlong/a.txt" ));
        stream.put("b", new FileInputStream("/home/bochenlong/b.txt" ));
        String txid = driver.apply(createTx(), stream);
        System.out.println(txid);
    }

    private static void query(BlockChainDriver driver) throws IOException {
        String qstr = UUID.randomUUID().toString();
```

```
        System.out.println(qstr);
        InputStream qr =
driver.query("daap://default_pdx_chain/eee8da2aeed816335181a4a15db7a49f3f2b7975/life.p
dx/ledger", qstr.getBytes());
        System.out.println(IOUtils.toString(qr, "utf-8"));
    }

    private static Transaction createTx() {
        Transaction tx = new Transaction();

 tx.setDst(URI.create("daap://default_pdx_chain/45874a3c0afc2a4d6cc9dea20245350f2981d3
ea/pdx.dapp/sample/db" ));

        tx.putMeta("name", "liudaqi".getBytes())
                .putMeta("age", "18".getBytes())
                .putMeta("sex", "man".getBytes());
        tx.setBody("DESC: this is a smart boy" .getBytes());
        return tx;
    }
```

注意项：

- *1 我们推荐一个项目里实例化一个BaaP Driver用于BaaP平台交互*
- *2 我们设置了一系列的属性用户BaaP Driver实例化，但并不是都是必要的，见下：*
    - `baap-chain-id` 默认不设置，我们将设置为BaaP 默认链
    - `baap-chain-type` 默认不设置，我们将设置为BaaP 默认实现的Ethereum链
    - `baap-url` 默认不设置，我们将随机选择最优节点连接
    - `baap-private-key` 默认不设置，我们将随机生成，但它对于您的加解密情景不友好，建议自行配置