安全风险智能化管控平台
双重预防机制数据交换规范
（V2.0）

北京伟瑞迪科技有限公司

2025 年 4 月

文件修订历史

序号	日期	修改章节	变更的主要内容	修订版本	修订人
1	2023-12-07	全部	首次发行	V1.0	马思强
2	2025-04-17	全部	优化技术实现方式	V2.0	马思强
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

目录

一、 接入准备

1. 平台分配凭证

accessKey：唯一身份标识（UUID 格式）。

```
00000000-0000-0000-0000-000000000000
```

rsaPublicKey：Base64 编码的 RSA 公钥。

```
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCBNfy8WBPrF/XabvjANs/6NMF0ArqwhfxK
uiUqulLopEa0ncRCwEfKiAuHpN4DA0CLPaom2x0JP28AhwoqrDBN0Ps6Pt0TeomVzZwMC1CD
6YUC0Azdr3EvTH8xy49Fnl/1tCCW1N2gITwDD/iW+jLBqfdTv9wP+l37oCvD7uulmwIDAQAB
```

rsaPrivateKey：Base64 编码的 RSA 私钥。

```
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAIE1/LxYE+sX9dpu+MA2z/o0
wXQCurCF/Eq6JSq6UuikRo6dxELAR8qIC4ek3gMA4Is9qibbE4k/bwCHCiqsME3Q+zo+3RN6
iZXNnAwLUIPphQI4DN2vcS9MfzHLj0WeX/W0IJbU3aAhPAMP+Jb6MsGp910/3A/6XfugK8Pu
66WbAgMBAAECgYBMxeMxlbPgEBCIhT9QFkFlDUuUvrpmgHXzgL7ssGDq0hEKVDt03gLKGCOh
qRgw+Khyk2UySSFGf2ZZDePAjybzfV5alV3DXJMdbxWEyq9Za2v9p91j0ybnKjpQ0RjSqHP6
snv/aAFq6IhAd8rFVLiJDP+rp+KKqOagpBUS/YpZQQJBALyh42b13k3W78x4S2K0duPQt1EI
8Wnh/8nn/C+q/vzH+qJU8b76ZribtTpAhVPOs50psVdvEN6w2QKxPrp4qNkCQQCvW11JtrYl
NpfKKOhW2ggqWlLsiz+ui3ZmJ21+YDbzLO6JN9gTOefm4gole3/3+k20Y3F/qGJoo4bXS0wY
4ZmTAkB9sM2K7KfNbDvrHfj1c1SJJqkP2QqryAid6/ORe7pei45JHpra65UIy9AtWZK9rkzm
m0JtBTfTezmO3rXyoLW5AkBbCPyCMzeWVCQzEE+mHyTUD/Yy1QunasF5wQhH7104m1/unq4v
NQWeSETqNpY1v50tg969XkZQemybQnzsAdq/AkEAr0WPyrCeHp0Jzc09fzQMZlRMu7keSWVm
SSmXBTz1G7KGnhvouuE8PGfk9t6sUDGi1shb/l1xPF/Q/nP8Vz7sqA==
```

2. 本地生成密钥

aesKey：每次上传数据需生成 16 字节随机 AES 密钥。

```
Z6IZRUtII20LB07I
```

3. 注意事项

1. 严禁使用示例密钥进行正式数据交互，所有生产环境密钥必须通过平台重新申请。
2. 调试用 RSA 私钥仅限本地开发使用，正式接入时平台不提供私钥。
3. AES 动态密钥必须满足：16 字节随机生成、单次请求单密钥、生产环境禁用连续可预测序列。

二、 技术实现方式

1. 通过 http/https 接口上传

通过 HTTP/HTTPS 协议传输结构化数据至平台，支持单条提交与批量提交两种模式，采用 RSA+AES 双重加密保障数据传输安全。

单条提交格式为：{...}

批量提交格式为：[{...}, {...}]

其中{...}为数据结构说明中的 JSON 格式示例的内容。

接口地址： https://ip:port/edi/data_upload

请求方式： POST

请求头属性说明：

序号	请求头	描述说明
1	Content-Type	固定值：application/json

请求体示例：


```
{
  "accessKey": "00000000-0000-0000-0000-000000000000",
  "accessSecret": "ZFWhut8R9uHRzkaZfXIQ/1Y61EaTLbXWX4hYVV6cK4SChA44SzYESkT3ClA2RdfTSjn/TXRdpNFT228T3TrkxsrruLvZBymBjLvw2+qMO7BgQfJ4itGvhOlV2CVlOsFqc9uQqUfRyyhQoOD73RAEZJl4gyyTg3jWFAo6EiRCXSg=",
  "dataType": "SafeRiskAnalysisUnit",
  "data": "yuTXJHw9XtsAwDuSOmsUxamkKTMO2qjwALuNnALiunZ3FJnhawTZGpCFUo7rt/L2rxL/yTAsBNbyUNUdzh8ZzA5O6k2gpQXW9eCSU1Dd13vUbf6uBHfXvxUnmIajOwDOCjcrpA5KCLhZxlhQqJwhiuZhogqU7wVJyhq59BCOMoQORiP8eHMCeD2j95CKssZ8jH8W65I+MtHOOHFjxlbOuGv6ifcJsamP8xgwZkgOsIQULsubMOlZNcvBENIsZMPUWOJaeZNLbkTJj3BvK7SI6PVB8+vNSLhxrAmQzuLF6OpgRDOUd4xaI1t7pGuE4vCStdEslbamAys1bqnA+hqUVitNlYYiAOQ37AV2BdIVSq7uHBFKofirOielNROgrRXB4ImBAOm67AeoG5TNEESOeYTDYWM52ag2Fu9z23WpHMn1VqboxjbHTFIhZLOwOqdYEfY72dwYLWweKKpw4VJMNyf/VGO53v2z8MC6vN9qGkbC+L3MR6o670a3u5/WBGf7APdEkNfcrQa76nPYQdr8/Dz8ilTrfnqjWs1WQ/zOFDEoa++7loCNEQv6W21y3u+wm8qcxhrKlnd2O5qE4Lomw9gBstDaWdpwElY6SHq3GWP5vw5dMrJHKKRCVSIQDk+D2Hg8at3tP1yaFyzWdNUq8Cn6oTVvq6c/mAdDM/bzjQ9CamktY7727I/gZgL7ghSl9ffqp9lEoeYLHUUkl7zNiX5tCheaJVp5ZLoecbdraccsB7fYki1CJoJar6XjbHvXSYVPrgeWCuz8wu7TLPTknu9WDJbRJFEkhiICRQdITk3Re9/PiTYuXNt+nUO5KoMzxJbdO6RPuHGGnO4FCskXO
```

```
Q6h39YYi53FNKkN+gKms1HdxQOqFbf6UoZ3sAevnZwaeSyhQmeuAbXcs30x+4YIg9nrrdQfh
OrPEmwSJFww4pLvCM8Yx2xonoHWZ3fvJ/IQ7ENVTiZ1lOHplH1yLsPVNg=="
}
```

请求体属性说明：

序号	属性名称	描述说明
1	accessKey	平台分配的 UUID。
2	accessSecret	RSA 加密后的 AES 密钥(Base64)。
3	dataType	业务数据类型，参考数据结构说明标题小括号里的内容。
4	data	AES 加密后的业务数据(Base64)。

响应体示例：

```
{
  "status": "SUCCESS",
  "info": "成功"
}
```

响应体属性说明：

序号	属性名称	描述说明
1	status	执行结果： SUCCESS：成功； FAIL：失败。
2	info	执行结果描述。

2. 通过 mqtt 消息队列上传

通过 MQTT 协议传输结构化数据至平台，支持单条提交与批量提交两种模式，采用 RSA+AES 双重加密保障数据传输安全。

单条提交格式为：{...}

批量提交格式为：[{...}, {...}]

其中{...}为数据结构说明中的 JSON 格式示例的内容。

地址： tcp://ip:port

主题：edi_data_upload

账号：test

密码：EdiDataUpload@2025

消息体示例：


```
{
  "accessKey": "00000000-0000-0000-0000-000000000000",
  "accessSecret": "ZFWhut8R9uHRzkaZfXIQ/1Y61EaTLbXWX4hYVV6cK4SChA44SzYESkT3ClA2RdfTSjn/TXRdpNFT228T3TrkxsrruLvZBymBjLvw2+qMO7BgQfJ4itGvhOlV2CVlOsFqc9uQqUfRyyhQoOD73RAEZJl4gyyTg3jWFAo6EiRCXSg=",
  "dataType": "SafeRiskAnalysisUnit",
  "data": "yuTXJHw9XtsAwDuSOmsUxamkKTMO2qjwALuNnALiunZ3FJnhawTZGpCFUo7rt/L2rxL/yTAsBNbyUNUdzh8ZzA5O6k2gpQXW9eCSU1Ddl3vUbf6uBHfXvxUnmIajOwDOCjcrpA5KCLhZxlhQqJwhiuZhogqU7wVJyhq59BCOMoQORiP8eHMCeD2j95CKssZ8jH8W65I+MtHOOHFjxlbOuGv6ifcJsamP8xgwZkgOsIQULsubMOlZNcvBENIsZMPUWOJaeZNLbkTJj3BvK7SI6PVB8+vNSLhxrAmQzuLF6OpgRDOUd4xaI1t7pGuE4vCStdEslbamAys1bqnA+hqUVitNlYYiAOQ37AV2BdIVSq7uHBFKofirOielNROgrRXB4ImBAOm67AeoG5TNEESOeYTDYWM52ag2Fu9z23WpHMn1VqboxjbHTFIhZLOwOqdYEfY72dwYLWweKKpw4VJMNyf/VGO53v2z8MC6vN9qGkbC+L3MR6o67Oa3u5/WBGf7APdEkNfcrQa76nPYQdr8/Dz8ilTrfnqjWs1WQ/zOFDEoa++7loCNEQv6W21y3u+wm8qcxhrKlnd2O5qE4Lomw9gBstDaWdpwElY6SHq3GWP5vw5dMrJHKKRCVSIQDk+D2Hg8at3tP1yaFyzWdNUq8Cn6oTVvq6c/mAdDM/bzjQ9CamktY7727I/gZgL7ghSl9ffqp9lEoeYLHUUkl7zNiX5tCheaJVp5ZLoecbdraccsB7fYki1CJoJar6XjbHvXSYVPrgeWCuz8wu7TLPTknu9WDJbRJFEkhiICRQdITk3Re9/PiTYuXNt+nUO5KoMzxJbdO6RPuHGGnO4FCskXOQ6h39YYi53FNKkN+gKms1HdxQOqFbf6UoZ3sAevnZwaeSyhQmeuAbXcs30x+4YIg9nrrdQfhOrPEmwSJFww4pLvCM8Yx2xonoHWZ3fvJ/IQ7ENVTiZ1lOHplH1yLsPVNg=="
}
```


消息体属性说明：

序号	属性名称	描述说明
1	accessKey	平台分配的 UUID。
2	accessSecret	RSA 加密后的 AES 密钥(Base64)。
3	dataType	业务数据类型，参考数据结构说明标题小括号里的内容。
4	data	AES 加密后的业务数据(Base64)。

三、 数据结构说明

1. 安全风险分析单元（SafeRiskAnalysisUnit)

JSON 格式示例：


```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-20 15:30:00",
  "recordCreateBy": "张伟",
  "recordUpdateTime": "2023-10-20 15:30:00",
  "recordUpdateBy": "王芳",
  "companyCode": "A12345678",
  "hazardCode": "d942b8fe-4f0d-4e7b-bd5e-5e5b5f5e5e5e",
  "riskUnitName": "化工原料储罐区安全风险评估单元",
  "hazardDep": "安全生产管理部",
  "hazardLiablePerson": "李明",
  "isHazard": "1",
  "hazardName": "2 号甲苯储罐",
  "hazardType": "3",
  "hazardPoint": {
    "lng": "116.40769",
    "lat": "39.89945"
  },
  "establishDate": "2018-05-15",
  "hazardRank": "2",
  "rValue": "3.45",
  "hazardDesc": "容积 50m³的常压立式储罐，存储介质为甲苯",
  "hiddenDanger": "法兰连接处存在微量渗漏可能，需定期紧固维护",
```

```
  "emerDeal": "1.立即启动泄漏应急预案\n2.使用防爆工具进行堵漏\n3.启动喷淋稀释系统\n4.疏散周边作业人员"
}
```


安全风险分析单元属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordUpdateTime	最后修改时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
6	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
7	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。
8	hazardCode	风险分析对象编码，安全风险分析对象编码即危险化学品登记综合服务系统中的危险源编码。 字符串（36），必传。
9	riskUnitName	安全风险分析单元名称。 字符串（200），必传。
10	hazardDep	责任部门，安全风险分析对象所属部门名称。

<table>
<tr><td></td><td></td><td>字符串（200），必传。</td></tr>
<tr><td>11</td><td>hazardLiablePerson</td><td>责任人，安全风险分析对象所属部门负责人姓名。
字符串（20），必传。</td></tr>
<tr><td>12</td><td>isHazard</td><td>是否为风险分析对象。
字符串，必传，填写数字枚举值：
0：否；
1：是。</td></tr>
<tr><td>13</td><td>hazardName</td><td>风险分析对象名称。
字符串（200），必传。</td></tr>
<tr><td>14</td><td>hazardType</td><td>风险分析单元分类。
字符串，非必传，填写数字枚举值：
1：作业活动；
2：作业场所；
3：设备；
4：其他。</td></tr>
<tr><td>15</td><td>hazardPoint</td><td>风险点位置，WGS-84 坐标系。
对象，非必传，属性说明：
{
"lng": "116.40769",
"lat": "39.89945"
}
<table>
<tr><th>参数名称</th><th>描述说明</th></tr>
<tr><td>lng</td><td>经度。
字符串（20），填写数字值。</td></tr>
<tr><td>lat</td><td>纬度。
字符串（20），填写数字值。</td></tr>
</table></td></tr>
<tr><td>16</td><td>establishDate</td><td>风险分析单元投用日期，风险点或设备的投用日期。
字符串，非必传，时间格式：yyyy-MM-dd</td></tr>
<tr><td>17</td><td>hazardRank</td><td>风险分析单元等级。</td></tr>
</table>

		字符串，非必传，填写数字枚举值： 1：Ⅰ级最高； 2：Ⅱ级； 3：Ⅲ级； 4：Ⅳ级最低。
18	rValue	风险分析单元 R 值。 字符串（50），非必传。
19	hazardDesc	风险分析单元描述。 字符串（200），非必传。
20	hiddenDanger	潜在隐患情况。 字符串（10000），非必传。
21	emerDeal	应急处置措施。 字符串（10000），非必传。

2. 安全风险事件（SafeRiskEvent)

JSON 格式示例：

```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-20 08:30:00",
  "recordCreateBy": "张伟",
  "recordUpdateTime": "2023-10-20 08:45:00",
  "recordUpdateBy": "王芳",
  "companyCode": "A12345678",
  "riskUnitId": "1",
  "riskEventName": "储罐区静电火花引发火灾事件",
```

```
  "riskEventContent": "物料输送过程中因静电接地失效引发可燃蒸气闪燃",
  "riskEventFactor": "1. 静电跨接器损坏\n2. 空气湿度低于 30%\n3. 甲苯流速超过安全阈值",
  "possibleAccident": "B 类火灾（液体火灾）伴蒸气云闪爆",
  "affectRange": "储罐区及周边 15 米范围内作业区域",
  "affectResult": "1. 可能造成 3-5 人烧伤\n2. 储罐压力连锁反应\n3. 厂区紧急停车损失约 500 万元",
  "riskEventDep": "生产运营部",
  "riskEventPerson": "李国强"
}
```


安全风险事件属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordUpdateTime	最后修改时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
6	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
7	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。

8	riskUnitId	安全风险分析单元 ID，所属安全风险单元主键。 字符串（36），必传。
9	riskEventName	安全风险事件名称。 字符串（100），必传
10	riskEventContent	风险事件内容。 字符串（1000），非必传。
11	riskEventFactor	风险事件因素。 字符串（1000），非必传。
12	possibleAccident	可能发生的事故。 字符串（1000），非必传。
13	affectRange	影响范围。 字符串（1000），非必传。
14	affectResult	潜在的后果。 字符串（1000），非必传。
15	riskEventDep	责任部门。 字符串（200），非必传。
16	riskEventPerson	责任人姓名。 字符串（20），非必传。

3. 安全风险管控措施（SafeRiskControlMeasure）

JSON 格式示例：

```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-20 10:20:00",
  "recordCreateBy": "张伟",
```

```
  "recordUpdateTime": "2023-10-20 10:35:00",
  "recordUpdateBy": "王芳",
  "companyCode": "A12345678",
  "riskEventId": "1",
  "dataSrc": "1",
  "riskMeasureDesc": "1. 安装静电在线监测系统（阈值<10Ω）\n2. 工艺参数控制：甲苯流速≤3m/s\n3. 强制通风系统（≥12 次/小时换气）",
  "classify1": "1",
  "classify2": "101",
  "classify3": "工艺安全联锁优化",
  "troubleshootContent": "1. 每日检查静电接地电阻值（标准≤4Ω）\n2. 每周校验流速报警装置\n3. 每月测试紧急切断阀响应时间（要求≤5s）",
  "riskMeasureLevel": "6",
  "riskMeasureDep": "安全生产管理部",
  "riskMeasurer": "李明"
}
```


安全风险管控措施属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss

4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordUpdateTime	最后修改时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
6	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
7	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。
8	riskEventId	安全风险事件 ID，所属风险事件主键。 字符串（36），必传。
9	dataSrc	管控方式。 字符串，非必传，填写数字枚举值： 1：自动化监控； 2：隐患排查。
10	riskMeasureDesc	安全风险管控措施描述。 字符串（4000），必传。
11	classify1	管控措施分类 1。 字符串，必传，填写数字枚举值： 1：工程技术； 2：维护保养； 3：操作行为； 4：应急措施。
12	classify2	管控措施分类 2。 字符串，必传，填写数字枚举值： 101：工艺控制； 102：关键设备/部件； 103：安全部件； 104：安全仪表； 105：其他； 201：动设备；

		202：静设备； 203：其他； 301：人员资质； 302：操作记录； 303：交接班； 304：其他； 401：应急设施； 402：个体防护； 403：消防设施； 404：应急预案； 405：其他。
13	classify3	管控措施分类 3，企业自定义。 字符串（100），非必传。
14	troubleshootContent	隐患排查内容。 字符串（10000），必传。
15	riskMeasureLevel	管控级别。 字符串，非必传，填写数字枚举值： 1：企业级； 2：部门级； 3：车间级； 4：班组级； 5：岗位级； 6：主要负责人； 7：技术负责人； 8：操作负责人。
16	riskMeasureDep	管控责任部门。 字符串（200），非必传。
17	riskMeasurer	管控责任人。 字符串（20），非必传。

4. 隐患排查任务（DangerCheckTask)

JSON 格式示例：


```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-20 11:00:00",
  "recordCreateBy": "张伟",
  "recordUpdateTime": "2023-10-20 11:15:00",
  "recordUpdateBy": "王芳",
  "companyCode": "A12345678",
  "riskMeasureId": "1",
  "troubleshootContent": "1. 静电接地电阻每日检测（标准≤4Ω）\n2. 流速报警装置功能测试\n3. 紧急切断阀响应时间校验（≤5秒）",
  "checkCycle": "1",
  "checkCycleUnit": "2",
  "workStartTime": "08:00:00",
  "workEndTime": "17:00:00",
  "workDayType": "2",
  "workType": "1",
  "taskNum": "5",
  "checkLevel": "5",
  "checkTimesDay": "1",
  "checkTimesNum": "3",
  "checkTaskType": "1",
  "checkStartDate": "2023-10-21 08:00:00",
```

```
  "checkEndDate": "2023-12-31 17:00:00"
}
```

隐患排查任务属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordUpdateTime	最后修改时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
6	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
7	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。
8	riskMeasureId	安全风险管控措施 ID，所属风险管控措施主键。 字符串（36），必传。
9	troubleshootContent	隐患排查内容。 字符串（1000），必传。
10	checkCycle	巡检周期，本次生成排查任务到下次生成排查任务的时间跨度。 字符串（4），必传，填写数字值。

11	checkCycleUnit	巡检周期单位，本次生成排查任务到下次生成排查任务的时间跨度的单位。 字符串，必传，填写数字枚举值： 1：小时:; 2：天； 3：月； 4：年。
12	workStartTime	工作开始时间，当排查周期是小时的时候必传。 字符串，格式：HH:mm:ss
13	workEndTime	工作结束时间，当排查周期是小时的时候必传。 字符串，格式：HH:mm:ss
14	workDayType	工作日类型。 字符串，非必传，填写枚举值数字： 1：每天； 2：法定工作日； 3：非法定工作日。
15	workType	任务类型。 字符串，必传，填写数字枚举值： 1：日常任务； 2：主要负责人任务； 3：技术负责人任务； 4：操作负责任任务。
16	taskNum	包保任务对应项，当任务类型为 2(主要负责人任务)、3(技术负责人任务)、4(操作负责人任务)`时此项必填。数字应与《危险化学品企业重大危险源安全包保责任人隐患排查任务清单》各项标号对应。 字符串，填写数字枚举值： **主要负责人：** 1：核查技术负责人、操作负责人是否按规定时间、规定内容履行职责。 2：确认重大危险源安全管理制度、操作规程是否实用有效，操作人

		员是否按制度和操作规程执行。 3：核查是否存在重大安全隐患，确认各类安全隐患是否及时整改。 4：核查重大危险源的管理和操作岗位人员数量、学历和资格是否满足要求，是否进行安全培训，是否具备安全管理、操作和应急方面的能力。 5：确认有关重大危险源的安全投入是否到位，是否合理有效使用安全费用。 6：确认重大危险源安全监测监控有关数据是否接入危险化学品安全生产风险监测预警系统。 7：确认重大危险源现场安全设施是否完好。 8：确认重大危险源专项应急预案是否每半年演练一次，是否达到演练效果。 9：核查双重预防机制数字化运行效果是否达到优良等级。 **技术负责人：** 1：现场确认重大危险源温度、压力、液位、流量、组份等信息的不间断采集和监测系统以及可燃气体和有毒有害气体泄漏检测报警装置是否具备信息远传、连续记录、事故预警、信息存储等功能。 2：现场核查重大危险源安全阀、压力表、液位计、可燃有毒气体报警仪、视频监控等是否存在故障、报警等信息，有关设备是否存在超期未检问题。 3：确认重大危险源设备设施的设计、制造、安装、使用、检测、维修、改造和报废，是否符合国家标准或者行业标准。 4：确认重大危险源与周边安全间距是否符合安全要求。对于超过个人和社会可容许风险值限值标准的重大危险源，组织采取相应的降低风险措施，直至风险满足可容许风险标准要求。 5：组织审查涉及重大危险源的外来施工单位及人员的相关资质、安全管理等情况。 6：重大活动、重点时段和节假日前组织进行重大危险源安全风险隐患排查。 7：现场审查涉及重大危险源的工艺、设备、人员变更方案，确保变

		更过程风险受控。 8：针对重大危险源安全风险隐患排查情况，组织制定管控措施和治理方案并监督落实。 9：组织演练重大危险源专项应急预案和现场处置方案。 **操作负责人：** 1：检查岗位操作人员是否严格执行重大危险源安全生产规章制度和操作规程，是否严格遵守劳动纪律。 2：检查涉及重大危险源的特殊作业、检维修作业是否按规定办理作业票，监护人是否在场，作业过程有无违章，安全风险是否受控。 3：检查重大危险源安全隐患是否整改到位，装置设备是否存在带“病”运行情形。 4：检查涉及重大危险源的外来施工单位及人员有无违章行为。 5：检查重大危险源的设备设施 (包括动静设备、自控系统、安全设施等)是否完好。 6：检查应急设施、应急装备、应急器材、消防设施是否完好。 7：确认现场监控设施是否完好，是否有效覆盖重大危险源区域。 8：确认现场可燃、有毒气体报警器和火灾报警器是否处于正常状态，报警信息是否及时处置。 9：检查危险化学品安全生产风险监测预警系统，警示信息是否及时处置，系统是否正常运行。 10：检查现场隐患排查人员是否熟悉排查流程，是否运用移动终端开展隐患排查，并形成闭环管理。
17	checkLevel	巡检级别。 字符串，必传，填写数字枚举值： 1：企业级； 2：部门级； 3：车间级； 4：班组级； 5：岗位级。
18	checkTimesDay	巡检频次(天数/班数)。

		字符串（3），非必传，填写数字值。
19	checkTimesNum	巡检频次(次数)。 字符串（3），非必传，填写数字值。
20	checkTaskType	巡检任务类型。 字符串，非必传，填写数字枚举值： 1：个人任务； 2：部门任务； 3：群组任务； 4：全员任务。
21	checkStartDate	巡检有效开始时间。 字符串，非必传，格式：yyyy-MM-dd HH:mm:ss
22	checkEndDate	巡检有效结束时间。 字符串，非必传，格式：yyyy-MM-dd HH:mm:ss

5. 隐患排查记录（DangerCheckRecord）

JSON 格式示例：


```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-21 14:00:00",
  "recordCreateBy": "张伟",
  "recordCreatorPhoneNo": "13900000000",
  "recordUpdateTime": "2023-10-21 14:30:00",
  "recordUpdateBy": "王芳",
  "recordUpdaterPhoneNo": "13911111111",
  "companyCode": "A12345678",
```

```
  "checkTaskId": "1",
  "checkTime": "2023-10-21 10:25:00",
  "mobileMe": "356879104523678",
  "isDefend": "0",
  "checkStatus": "2",
  "checkRiskName": "静电接地电阻值超标",
  "checkRiskLevel": "2",
  "checkDep": "安全生产管理部",
  "checker": "李明",
  "checkerPhoneNo": "13812345678"
}
```


隐患排查记录属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordCreatorPhoneNo	创建人手机号 字符串（20），必传。
6	recordUpdateTime	最后修改时间。

		字符串，必传，格式：yyyy-MM-dd HH:mm:ss
7	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
8	recordUpdaterPhoneNo	最后修改人手机号。 字符串（20），必传。
9	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。
10	checkTaskId	隐患排查任务 ID，所属隐患排查任务主键。 字符串（36），必传。
11	checkTime	排查时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
12	mobileMe	IMEI 码，国际移动设备识别码 IMEI。当设备存在多个 IMEI 码时，只上传第一个。 字符串（15），必传。
13	isDefend	是否包保责任人任务，当排查任务中任务类型为主要负责人任务、技术负责人任务、操作负责人任务时，此项为 1。 字符串，必传，填写数字枚举值： 0：否； 1：是。
14	checkStatus	排查风险名称。 字符串，必传，填写数字枚举值： 1：正常； 2：存在隐患； 3：未排查； 4：其他。
15	checkRiskName	排查风险名称。， 字符串（200），非必传。
16	checkRiskLevel	排查风险等级。 字符串，必传，填写数字枚举值： 1：重大风险；

		2：较大风险； 3：一般风险； 4：低风险。
17	checkDep	排查部门。 字符串（200），非必传。
18	checker	排查人。 字符串（20），必传。
19	checkerPhoneNo	排查人联系电话。 字符串（20），必传。

6. 隐患信息（DangerInfo）

JSON 格式示例：


```
{
  "recordId": "1",
  "recordDeleted": "0",
  "recordCreateTime": "2023-10-21 15:00:00",
  "recordCreateBy": "张伟",
  "recordUpdateTime": "2023-10-21 15:30:00",
  "recordUpdateBy": "王芳",
  "companyCode": "A12345678",
  "hazardCode": "d942b8fe-4f0d-4e7b-bd5e-5e5b5f5e5e5e",
  "riskMeasureId": "1",
  "checkRecordId": "1",
  "dangerName": "静电接地系统失效隐患",
  "dangerLevel": "2",
  "registerTime": "2023-10-21 14:45:00",
```

```
  "registrant": "李明",
  "dangerSrc": "1",
  "dangerManageType": "2",
  "hazardDangerType": "7",
  "hazardCategory": "3",
  "dangerDesc": "储罐区 3 号静电接地桩电阻值 5.8Ω（标准≤4Ω），存在静电积聚引发燃爆风险",
  "dangerReason": "1. 接地铜缆氧化导致接触不良\n2. 雨季土壤电阻率升高\n3. 未执行月度专项检测",
  "controlMeasures": "1. 更换 16mm²镀锡铜接地线\n2. 添加降阻剂改善接地极\n3. 建立双周检测制度",
  "cost": "2.5",
  "liablePerson": "李明",
  "dangerManageDeadline": "2023-10-25 17:00:00",
  "dangerState": "1"
}
```


隐患信息属性说明：

序号	参数名称	描述说明
1	recordId	主键。 字符串（36），必传。
2	recordDeleted	删除标志。 字符串，必传，填写数字枚举值： 0：正常； 1：已删除。
3	recordCreateTime	创建时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
4	recordCreateBy	创建人姓名。 字符串（20），必传。
5	recordUpdateTime	最后修改时间。

		字符串，必传，格式：yyyy-MM-dd HH:mm:ss
6	recordUpdateBy	最后修改人姓名。 字符串（20），必传。
7	companyCode	企业编码，危险化学品登记综合服务系统中的企业编码。 字符串（9），必传。
8	hazardCode	安全风险分析对象编码，安全风险分析对象编码即危险化学品登记综合服务系统中的危险源编码。 字符串（36），必传。
9	riskMeasureId	安全风险管控措施 ID，管控措施主键，所有隐患排查任务产生的隐患必须绑定管控措施。 字符串（36），必传。
10	checkRecordId	隐患排查记录 ID，隐患排查记录主键，所有由隐患排查时产生的隐患必须提供关联的隐患排查记录 ID。 字符串（36），必传。
11	dangerName	隐患名称。 字符串（200），必传。
12	dangerLevel	隐患等级。 字符串，必传，填写数字枚举值： 1：一般隐患； 2：重大隐患。
13	registerTime	登记时间。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
14	registrant	登记人姓名。 字符串（20），必传。
15	dangerSrc	隐患来源。 字符串，必传，填写数字枚举值： 1：日常排查； 2：综合性排查； 3：专业性排查； 4：季节性排查；

		5：重点时段及节假日前排查； 6：事故类比排查； 7：复产复工前排查； 8：外聘专家诊断式排查； 9：管控措施失效； 10：其他； 11：执法检查。
16	enforcementId	执法编号，当隐患来源类型为 11(执法检查)时此项必填。 字符串（36）。
17	dangerManageType	治理类型。 字符串，必传，填写数字枚举值： 1：即查即改； 2：限期整改。
18	hazardDangerType	隐患类型。 字符串，必传，填写数字枚举值： 1：安全； 2：工艺； 3：电气； 4：仪表； 5：消防； 6：总图； 7：设备； 8：其他。
19	hazardCategory	隐患类别。 字符串，必传，填写数字枚举值： 1：其他隐患； 2：主要负责人登记隐患； 3：技术负责人登记隐患； 4：操作负责人登记隐患。
20	dangerDesc	隐患描述。

		字符串（10000），必传。
21	dangerReason	原因分析。 字符串（10000），必传。
22	controlMeasures	控制措施。 字符串（1000），必传。
23	cost	资金，单位（万元）。 字符串（100），非必传，填写数字值。
24	liablePerson	整改责任人姓名。 字符串（20），必传。
25	dangerManageDeadline	隐患治理期限。 字符串，必传，格式：yyyy-MM-dd HH:mm:ss
26	checkAcceptPerson	验收人姓名，当隐患状态为 3(已验收)时此项必填。 字符串（20）。
27	checkAcceptTime	验收时间，当隐患状态为 3(已验收)时此项必填。 字符串，格式：yyyy-MM-dd HH:mm:ss
28	checkAcceptComment	验收情况。 字符串（1000），非必传。
29	dangerState	隐患状态。 字符串，必传，填写数字枚举值： 1：整改中； 2：待验收； 3：已验收。

四、 附录

1. AES 加解密工具

```
import lombok.SneakyThrows;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.stream.Collectors;
import java.util.stream.Stream;

/**
 * AES 加解密工具
 */
public final class AESEncryptionUtils {
    private static final String ALGORITHM = "AES";
    private static final String CIPHER_PADDING = "AES/ECB/PKCS5Padding";
    private static final char[] CHAR_BASE = "Aa1Bb2Cc3Dd4Ee5Ff6Gg7Hh8Ii9JjOKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz".toCharA
rray();

    /**
     * 生成秘钥
     *
     * @return AES 秘钥
     */
    public static String getKey() {
        return Stream.generate(() -> Character.toString(CHAR_BASE[(int) (Math.random() * CHAR_BASE.length)]))
                .limit(16).collect(Collectors.joining());
    }

    /**
     * 加密
```

```
     *
     * @param key     AES 秘钥
     * @param content 要加密的内容
     * @return 加密后的内容（Base64 字符串格式）
     */
    @SneakyThrows
    public static String encrypt(String key, String content) {
        Cipher cipher = Cipher.getInstance(CIPHER_PADDING);
        cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(key.getBytes(StandardCharsets.UTF_8), ALGORITHM));
        return Base64.getEncoder().encodeToString(
                cipher.doFinal(content.getBytes(StandardCharsets.UTF_8)));
    }

    /**
     * 解密
     *
     * @param key     AES 秘钥
     * @param content 要解密的内容（Base64 字符串格式）
     * @return 解密后的内容
     */
    @SneakyThrows
    public static String decrypt(String key, String content) {
        Cipher cipher = Cipher.getInstance(CIPHER_PADDING);
        cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(key.getBytes(StandardCharsets.UTF_8), ALGORITHM));
        return new String(
                cipher.doFinal(Base64.getDecoder().decode(content)),
                StandardCharsets.UTF_8);
    }
}
```

2. RSA 加解密工具

```
import lombok.Getter;
import lombok.RequiredArgsConstructor;
import lombok.SneakyThrows;

import javax.crypto.Cipher;
import java.nio.charset.StandardCharsets;
import java.security.*;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;

/**
 * RSA 加解密工具
 */
public final class RSAEncryptionUtils {
    public static final String ALGORITHM = "RSA";

    /**
     * 生成秘钥对
     *
     * @param keySize 秘钥大小
     * @return 秘钥对（含公钥和私钥）
     */
    @SneakyThrows
    public static RSAKeyPair getKeyPair(int keySize) {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(ALGORITHM);
        keyPairGenerator.initialize(keySize, new SecureRandom());
        KeyPair keyPair = keyPairGenerator.generateKeyPair();
```

```
        Base64.Encoder encoder = Base64.getEncoder();
        return new RSAKeyPair(
                encoder.encodeToString(keyPair.getPublic().getEncoded()),
                encoder.encodeToString(keyPair.getPrivate().getEncoded()));
    }

    /**
     * 公钥加密
     *
     * @param publicKey 公钥（Base64 字符串格式）
     * @param content   要加密的内容
     * @return 加密后的内容（Base64 字符串格式）
     */
    @SneakyThrows
    public static String encrypt(String publicKey, String content) {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, KeyFactory.getInstance(ALGORITHM)
                .generatePublic(new X509EncodedKeySpec(Base64.getDecoder().decode(publicKey))));
        return Base64.getEncoder().encodeToString(
                cipher.doFinal(content.getBytes(StandardCharsets.UTF_8)));
    }

    /**
     * 私钥解密
     *
     * @param privateKey 公钥（Base64 字符串格式）
     * @param content    要解密的内容（Base64 字符串格式）
     * @return 解密后的内容
     */
    @SneakyThrows
```

```
    public static String decrypt(String privateKey, String content) {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, KeyFactory.getInstance(ALGORITHM)
                .generatePrivate(new PKCS8EncodedKeySpec(Base64.getDecoder().decode(privateKey))));
        return new String(
                cipher.doFinal(Base64.getDecoder().decode(content)),
                StandardCharsets.UTF_8);
    }

    @Getter
    @RequiredArgsConstructor
    public static final class RSAKeyPair {
        private final String publicKey;
        private final String privateKey;
    }
}
```

3. 图像格式化工具

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Base64;

/**
 * 图像格式化工具
 */
public final class ImageFormatUtils {
```

```
    /**
     * 格式化图像
     *
     * @param image 图片
     * @return 图像格式化文本
     */
    public static String formatImage(BufferedImage image) throws IOException {
        try (ByteArrayOutputStream outputStream = new ByteArrayOutputStream()) {
            ImageIO.write(image, "png", outputStream);
            return "" + Base64.getEncoder().encodeToString(outputStream.toByteArray());
        }
    }
}
```

4. http/https 请求工具

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;
import java.util.stream.Collectors;
```

```
/**
 * HTTP 请求工具
 */
public final class HttpRequestUtils {

    /**
     * 做 POST 请求
     *
     * @param url         接口地址
     * @param requestBody 请求体
     */
    public static void doPost(String url, String requestBody) throws IOException {
        HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
        connection.setRequestProperty("Accept", "application/json");
        connection.setConnectTimeout(3000);
        connection.setReadTimeout(5000);
        connection.setDoOutput(true);
        try (OutputStream outputStream = connection.getOutputStream()) {
            outputStream.write(requestBody.getBytes(StandardCharsets.UTF_8));
        }
        int responseCode = connection.getResponseCode();
        String responseBody;
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream(), StandardC
harsets.UTF_8))) {
            responseBody = reader.lines().collect(Collectors.joining("\n"));
        }
        System.out.println(responseCode);
```

```
        System.out.println(responseBody);
    }

    static {
        try {
            X509TrustManager trustManager = new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    return null;
                }

                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                }

                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                }
            };

            SSLContext sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[]{trustManager}, new SecureRandom());
            HttpsURLConnection.setDefaultSSLSocketFactory(sslContext.getSocketFactory());
            HttpsURLConnection.setDefaultHostnameVerifier((hostname, session) -> true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5. mqtt 客户端工具

```
import org.eclipse.paho.client.mqttv3.*;
```

```
import java.nio.charset.StandardCharsets;
import java.util.Objects;

/**
 * MQTT 客户端
 * <dependency>
 * <groupId>org.eclipse.paho</groupId>
 * <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
 * <version>1.2.5</version>
 * </dependency>
 */
public final class MQTTClient implements MqttCallback {
    private final String serverURI;
    private MqttConnectOptions options;
    private MqttClient client;

    /**
     * @param serverURI 服务器连接地址
     * @param username  登录用户名
     * @param password  登录密码
     */
    public MQTTClient(String serverURI, String username, String password) {
        this.serverURI = serverURI;
        new Thread(() -> {
            try {
                options = new MqttConnectOptions();
                options.setUserName(username);
                options.setPassword(password.toCharArray());
                options.setConnectionTimeout(10);
```

```
                reconnect();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }).start();
    }

    /**
     * @param topic   主题
     * @param message 消息体
     */
    public void push(String topic, String message) throws MqttException {
        client.publish(topic, new MqttMessage(message.getBytes(StandardCharsets.UTF_8)));
    }

    private void reconnect() throws MqttException {
        close();
        client = new org.eclipse.paho.client.mqttv3.MqttClient(serverURI, MqttClient.generateClientId());
        client.setCallback(this);
        client.connect(options);
    }

    private void close() {
        if (Objects.nonNull(client)) {
            try {
                client.disconnect();
                client.close();
            } catch (MqttException e) {
                e.printStackTrace();
            } finally {
```

```
                client = null;
            }
        }
    }

    @Override
    public void connectionLost(Throwable e) {
        e.printStackTrace();
        try {
            client.reconnect();
        } catch (MqttException ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public void messageArrived(String topic, MqttMessage message) throws Exception {
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken token) {
    }
}
```

6. 通过 https 接口批量上传数据的示例

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
```

```
import java.util.Map;

public class TestHttpRequest {

    public static void main(String[] args) throws IOException {
        Map<String, String> location = new LinkedHashMap<>();
        location.put("lng", "116.40769");
        location.put("lat", "39.89945");

        Map<String, Object> data = new LinkedHashMap<>();
        data.put("recordId", "1");
        data.put("recordDeleted", "0");
        data.put("recordCreateTime", "2023-10-20 15:30:00");
        data.put("recordCreateBy", "张伟");
        data.put("recordUpdateTime", "2023-10-20 15:30:00");
        data.put("recordUpdateBy", "王芳");
        data.put("companyCode", "A12345678");
        data.put("hazardCode", "d942b8fe-4f0d-4e7b-bd5e-5e5b5f5e5e5e");
        data.put("riskUnitName", "化工原料储罐区安全风险评估单元");
        data.put("hazardDep", "安全生产管理部");
        data.put("hazardLiablePerson", "李明");
        data.put("isHazard", "1");
        data.put("hazardName", "2 号甲苯储罐");
        data.put("hazardType", "3");
        data.put("hazardPoint", location);
        data.put("establishDate", "2018-05-15");
        data.put("hazardRank", "2");
        data.put("rValue", "3.45");
        data.put("hazardDesc", "容积 50m³的常压立式储罐，存储介质为甲苯");
        data.put("hiddenDanger", "法兰连接处存在微量渗漏可能，需定期紧固维护");
```

```
        data.put("emerDeal", "1.立即启动泄漏应急预案\n2.使用防爆工具进行堵漏\n3.启动喷淋稀释系统\n4.疏散周边作业人员
");

        List<Object> dataList = new ArrayList<>();
        dataList.add(data);

        String accessKey = "00000000-0000-0000-0000-000000000000";
        String publicKey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCBNfy8WBPrF/XabvjANs/6NMF0ArqwhfxKuiUqulLopEa0ncRC
wEfKiAuHpN4DA0CLPaom2x0JP28AhwoqrDBN0Ps6Pt0TeomVzZwMC1CD6YUC0Azdr3EvTH8xy49Fnl/1tCCW1N2gITwDD/iW+jLBqfdTv9wP+l37oCvD
7uulmwIDAQAB";
        String requestBody = ServiceDataProcessUtils.processServiceData(accessKey, publicKey, "SafeRiskAnalysisUnit
", dataList);
        HttpRequestUtils.doPost("https://ip:port/edi/data_upload", requestBody);
    }
}
```

7. 通过 mqtt 接口批量推送数据的示例

```
import org.eclipse.paho.client.mqttv3.MqttException;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

public class TestMqttPush {
```

```
public static void main(String[] args) throws IOException {
    Map<String, String> location = new LinkedHashMap<>();
    location.put("lng", "116.40769");
    location.put("lat", "39.89945");

    Map<String, Object> data = new LinkedHashMap<>();
    data.put("recordId", "1");
    data.put("recordDeleted", "0");
    data.put("recordCreateTime", "2023-10-20 15:30:00");
    data.put("recordCreateBy", "张伟");
    data.put("recordUpdateTime", "2023-10-20 15:30:00");
    data.put("recordUpdateBy", "王芳");
    data.put("companyCode", "A12345678");
    data.put("hazardCode", "d942b8fe-4f0d-4e7b-bd5e-5e5b5f5e5e5e");
    data.put("riskUnitName", "化工原料储罐区安全风险评估单元");
    data.put("hazardDep", "安全生产管理部");
    data.put("hazardLiablePerson", "李明");
    data.put("isHazard", "1");
    data.put("hazardName", "2 号甲苯储罐");
    data.put("hazardType", "3");
    data.put("hazardPoint", location);
    data.put("establishDate", "2018-05-15");
    data.put("hazardRank", "2");
    data.put("rValue", "3.45");
    data.put("hazardDesc", "容积 50m³的常压立式储罐，存储介质为甲苯");
    data.put("hiddenDanger", "法兰连接处存在微量渗漏可能，需定期紧固维护");
    data.put("emerDeal", "1.立即启动泄漏应急预案\n2.使用防爆工具进行堵漏\n3.启动喷淋稀释系统\n4.疏散周边作业人员");

    List<Object> dataList = new ArrayList<>();
```

```
            dataList.add(data);

            String accessKey = "00000000-0000-0000-0000-000000000000";
            String publicKey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCBNfy8WBPrF/XabvjANs/6NMF0ArqwhfxKuiUqulLopEaOncRCwEfKiAu
HpN4DAOCLPaom2xOJP28AhwoqrDBN0Ps6Pt0TeomVzZwMC1CD6YUCOAzdr3EvTH8xy49Fnl/1tCCW1N2gITwDD/iW+jLBqfdTv9wP+l37oCvD7uulmwIDA
QAB";
            String message = ServiceDataProcessUtils.processServiceData(accessKey, publicKey, "SpecialWorkCheckAccept", dataList);

            try (MQTTClient mqttClient = new MQTTClient("tcp://ip:port", "test", "EdiDataUpload@2025")) {
                mqttClient.push("edi_data_upload", message);
            } catch (MqttException e) {
                e.printStackTrace();
            }
        }
}
```