ALEIYE

让 | 大 | 数 | 据 | 更 | 简 | 单

Aleiye 企业版 API 接口说明

1 概述

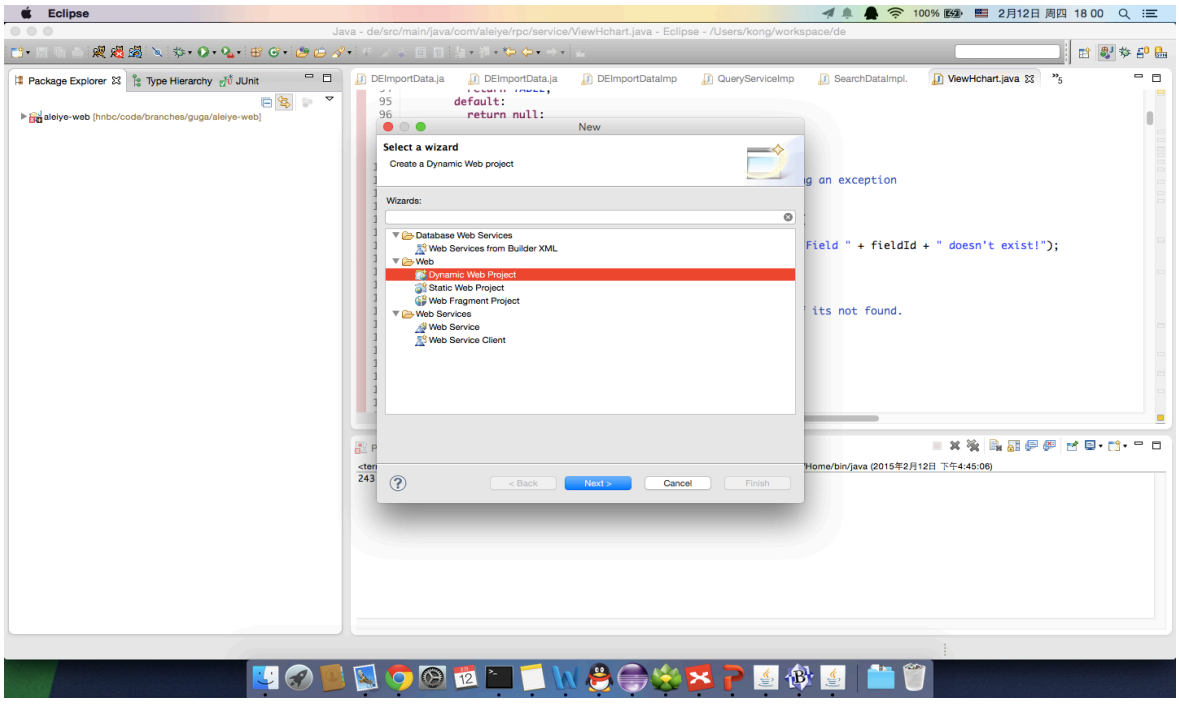
此文档针针对开发人员，能够帮助开发人员将 Aleiye 大数据平台的工具包嵌入到自定义的应用中，并通过产品 API 接口可以对数据完成添加、处理、分析等操作。

2 快速开始

以 Java 语言作为 产品 API 调用为例，各 API 接口具体使用方法如下。

2.1 新建或配置 API 调用项目

使用 Eclipse IDE 为例，创建或者调用 Aleiye 的 API 接口项目。



其中 Aleiye API 的所有类都存放在 com/aleiye/rpc/service 目录下。目录结构为：

2.2 核心依赖包

Aleiye 需要以下依赖包来完成 API 调用，以下为依赖包列表：

- aleiye-client-3.2.0-SNAPSHOT.jar
- aleiye-event-3.2.0-SNAPSHOT.jar
- commons-codec-1.6.jar
- commons-logging-1.1.1.jar
- hamcrest-core-1.1.jar
- httpclient-4.2.5.jar
- httpcore-4.2.4.jar
- jackson-core-lgpl-1.9.13.jar
- jackson-mapper-lgpl-1.9.13.jar
- joda-time-2.9.2.jar
- junit-4.10.jar
- libthrift-0.9.2.jar
- protobuf-java-2.5.0.jar
- slf4j-api-1.5.8.jar
- slf4j-simple-1.7.5.jar

2.3 定义接口调用

- 创建 SearchDataTest 类。
- 粘贴下述代码
- 修改套节字方法 TSocket（String host, int port, int timeout）参数：

参数名称	说明
String host	Aleiey 服务所在 IP 地址或主机名称，需要

	修改
Int port	API 服务调用端口，默认即可
Int timeout	超时时长，可以根据实际情况进行修改

- 在查询用户信息方法 queryUserID()中，修改登录方法 login（String account, String password）中的参数：

参数名称	说明
String account	用户名
String password	密码

- 参数修改完成后，便可以执行测试方法，便可以看到测试输出结果。

```
package com.aleiye.plugins.search;


import com.aleiye.client.service.search.SearchData;
import com.aleiye.client.service.search.model.*;
import com.aleiye.client.service.security.RUserService;
import com.aleiye.client.service.security.model.RUserInfoTO;
import com.aleiye.client.service.sql.RDataCellModel;
import com.aleiye.client.service.sql.REsSqlService;
import com.aleiye.client.service.sql.RQueryResultModel;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TMultiplexedProtocol;
import org.apache.thrift.protocol.TProtocol;
import org.apache.thrift.transport.TFramedTransport;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.transport.TTransport;
import org.joda.time.DateTime;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
```

```
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class SearchDataTest {



    private TTransport transport1;
    private TTransport transport2;
    private SearchData.Client searchData;
    private RUserService.Client userService;
    private REsSqlService.Client essqlService;

    @After
    public void after() throws Exception {
        if (transport1 != null) {
            transport1.close();
        }
        if(transport2 != null){
            transport2.close();
        }
    }

    @Before
    public void before() throws Exception {
        TTransport transport1 = new TSocket("10.0.1.132", 6060, 60 * 1000);
        TTransport transport2 = new TSocket("10.0.1.132", 10003, 60 * 1000);
        transport1.open();
        transport2.open();
        transport1 = new TFramedTransport(transport1);
        transport2 = new TFramedTransport(transport2);
        // tt.open();
```

```
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol1 = new TBinaryProtocol(transport1);
        TProtocol protocol2 = new TBinaryProtocol(transport2);
        // TProtocol protocol = new TCompactProtocol(transport);
        TMultiplexedProtocol rusersp = new TMultiplexedProtocol(protocol1,
RUserService.class.getName());
        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol1,
SearchData.class.getName());
        TMultiplexedProtocol essqlsp = new
TMultiplexedProtocol(protocol2,REsSqlService.class.getName());

        userService = new RUserService.Client(rusersp);
        searchData = new SearchData.Client(serviceProtocol);
        essqlService = new REsSqlService.Client(essqlsp);
    }

    @Test
    // 查询当前用户 ID
    public void testQueryUserID() throws TException {
        RUserInfoTO set = userService.login("admin","123456");
        System.out.println("当前用户的 ID 为:" +set.getAuthCode());
    }


    // 查询当前用户 ID,可指定不同用户查询 ID
    public String queryUserID() throws TException {
        //参数需指定登录用户名及密码
        RUserInfoTO set = userService.login("admin","123456");
        return set.getAuthCode();
    }

    @Test
```

```
    // 查询当前用户（用户 ID 为 1）下所有数据类型（共分为两类：日志数据定义的日志类型和结构化数据的数据库表）
    public void queryAllDataType() throws TException {
        Set<String> set = searchData.queryType(queryUserID());
        System.out.println(set);
    }
    @Test
    // 查询当前用户（用户 ID 为 1）下指定的数据类型下的表结构
    public void queryDataType() throws TException {
        Set<String> set = new HashSet();
        set.add("sql");
//        set.add("dataType2");
        List<TMappingMetaModel> list =
searchData.getFieldMetaDataByUserIdAndType(queryUserID(),set);
        System.out.println(list);

    }

    @Test
    // 查询当前用户（用户 ID 为 1）下所有索引
    public void queryIndecies() throws TException {
        TQueryIndexModel model = new TQueryIndexModel();
//        设置当前用户 ID
        model.setUserId(queryUserID());
//        设置集群名称,默认为"aleiye",无须自定义设置
        model.setDatabase("aleiye");
        model.setIsProcessTime(true);
        model.setFrom(new DateTime("2016-07-01").getMillis());
        model.setTo(new DateTime("2016-07-19").getMillis());
        List<String> list = searchData.queryIndecies(model);
        System.out.println(list);
    }

    @Test
```

```
    // 按搜索语句查询数据结果，以 CSV 表格结构输出
    public void  queryData() throws TException {
        TQueryModel model = new TQueryModel();
        model.setQueryString("A_source:\"sql\"");
        model.setPageSize(1000);

        TCSVResult result = searchData.query2CSV(model, new CSVParameter());
        System.out.println(result.getLines());
    }

    @Test
    // 查询某字段统计，如本字段下所有数据的最大值、最小值、平均值
    public void testFieldStat() throws TException {
        TQueryModel model = new TQueryModel();
        model.setIndecies(Collections.singletonList("aleiye-0-2016-07-12-1"));
        model.setQueryString("A_source:sql");
        FieldStatResult result = searchData.fieldStat(model, "age", "long", 10);
        System.out.println(result);

    }

    @Test
    //通过原句+SQL 查询方式进行数据查询
    public void queryDeSql() throws Exception{

        Long fromDate = new DateTime(2016,07,01,0,0,0).getMillis();
        Long toDate   = new DateTime(2016,07,20,0,0,0).getMillis();
        boolean isProcessTime = false;
        String sql = "select * from test111 limit 10";
        String dequery = "A_source:\"test111\"";
        RQueryResultModel result =
essqlService.query(sql,dequery,fromDate,toDate,isProcessTime,queryUserID());
        List<List<RDataCellModel>> list = result.dataList;
```

```
        for(List<RDataCellModel> ele : list){
            for(RDataCellModel rd : ele){
                System.out.println(rd.data);
            }
        }
    }

}
```

2.4 创建数据入库

- 构建数据对象，AleiyeDataBuilder.builder builder = AleiyeDataBuilder.builder();
- 修改设定属性 builder.setDataType（String datatype）：

参数名称	说明
String datatype	需要创建一个数据类型

- 设定鉴权码 builder.setUserAuth（String setUserAuth）

参数名称	说明
String setUserAuth	用户名密码，默认为 admin

- 必须设定属性业务时间 builder.setBusinessTime（Date BusinessTime）

参数名称	说明
Date BusinessTime	业务时间

- 构建数据发送的客户端；其中 IP 为 aleiye－server 所在及其的 ip；port 为配置的传输端口；一般为 6063，AleiyeDataClient client = new AleiyeDataClient (IP, PORT)；client.start()；

参数名称	说明
String host	Aleiey 服务所在 IP 地址或主机名称，需要

	修改
Int port	配置的传输端口，默认 6063

- 发送数据方式；确定字段类型及字段名称

  方法一

  1） 构建数据结构

  每个字段使用 DataFieldStruct 构建字段信息；

  类名：DataFieldStruct

  属性：filedName：字段名称，fieldType：字段类型

  将构建完毕的字段定义放入 List 中；

  多个字段重复以上操作；

  调用 builder.setDataFieldStruct(dataFieldStructList);将字段定义加载入 builder 中；

  2） 构建数据集合

  每条数据均为一个 List 集合（List<Object>）集合中存储的一条数据的各个字段值；其中字段的顺序必须与 1 中定义的顺序保持一致；

  3） 加载数据

  构建完一条数据后；调用 builder.putData(objectList);将数据加载入 builder 中；

  如果有多条数据则重复执行 2、3 即可；

  4） 传输数据（考虑到网路问题；建议一次传输大小不要超过 50M；）

  List<ByteBuffer> data = builder.build();

  boolean isSuccess = client.tranferData(data);

方法二

1) 构建 Map 对象

```
Map<String, Object> map = new HashMap<>();
```

2) 加载数据

向 map 中放入一条数据的各个字段；其中值的类型只能为（String，Long，Integer，double，java.util.Date,float）

3) 加载数据

调用 builder.putMapData(map);将数据缓存入 builder 中；

如果有多条数据则重复执行 2、3 即可；

4) 发送数据（考虑到网路问题；建议一次传输大小不要超过 50M;）

```
List<ByteBuffer> data = builder.build();
boolean isSuccess = client.tranferData(data);
```

```
import com.aleiye.event.factory.model.ParsedFieldModel;
import com.aleiye.event.protobuf.AleiyeParsedEvent;
import com.aleiye.transfer.data.client.entity.DataFieldStruct;
import com.aleiye.transfer.data.client.enums.FieldType;
import junit.framework.Assert;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.nio.ByteBuffer;
import java.util.*;

public class AleiyeDataClientTest {

    AleiyeDataBuilder.Builder builder;

    AleiyeDataClient client;

    @Before
    public void builder() {
        builder = AleiyeDataBuilder.builder();
        //设置数据类型,必须字段
        builder.setDataType("taiji10");
        //设置业务时间;同一批次数据,只有一个业务时间
        builder.setBusinessTime(new Date());

        //数据必须有用户 Id
        //默认为 admin 用户
        builder.setUserAuth("8B6B254FBE90B2A3");
        //创建 client
        client = new AleiyeDataClient("10.0.1.136", 6060);
        client.start();
    }

    @Test
    public void tranferData() throws Exception {
        //构建数据
        List<Object> objectList = new ArrayList<>();
        objectList.add("751015002016051280022300738858 21");
        objectList.add(11);

        for (int i = 0; i < 1000; i++) {
            builder.putData(objectList);
        }
        List<ByteBuffer> data = builder.build();
        boolean isSuccess = client.tranferData(data);
        Assert.assertTrue(isSuccess);
    }
```

```
    @Test
        public void tranferMapData() throws Exception {
            builder.setDataType("ywt1");
            Map<String,Object> map = new HashMap<>();
            map.put("ywt1_fiedl1",1);
            map.put("ywt1_fiedl2","tsttt");
            map.put("ywt1_fiedl3",1);
            map.put("ywt1_fiedl4",1);
            builder.putMapData(map);
            List<ByteBuffer> data = builder.build();
            boolean isSuccess = client.tranferData(data);
            Assert.assertTrue(isSuccess);
        }


        @Test
        public void tranferPutParsedFieldData() throws Exception {

            builder.setDataType("ywt1");

            List<ParsedFieldModel> dataList = new ArrayList<>();
            ParsedFieldModel dfm = new ParsedFieldModel();
            dfm.setFieldName("ywt1_fiedl3");
            dfm.setFieldValue(123124);
            dfm.setFieldType(AleiyeParsedEvent.FieldType.INTEGER);
            dataList.add(dfm);
            builder.putParsedFieldData(dataList);
            List<ByteBuffer> data = builder.build();
            boolean isSuccess = client.tranferData(data);
            Assert.assertTrue(isSuccess);

            builder.putParsedFieldData(dataList);
            List<ByteBuffer> data1 = builder.build();
            boolean isSuccess1 = client.tranferData(data1);
            Assert.assertTrue(isSuccess1);

        }

        @After
        public void close() {
            client.close();
        }
    }
```

3 采集器管理接口说明（RCollectService.Iface）

3.1 方法说明概要

方法名称	说明
获取指定用户管理的所有采集器（listCollectByUserID）	该接口针对某一用户获取其所有采集器信息
通过采集器 ID 获取采集器信息（getCollectorById）	基于采集器 ID 从而获取采集器信息
获取指定 ID 采集器的监控明细（getDetailsById）	通过采集器 ID 获取采集器的监控明细
暂停采集器（pauseCollector）	暂停采集器
重置采集器（resumeCollector）	重置采集器
删除采集器（delCollector）	删除采集器

3.2 获取指定用户管理的所有采集器（listCollectByUserID）

- 方法定义

  List<RCollectorTO> listCollectByUserID(int userId) throws RException, RReturnNullException，TException

- 方法说明

  该接口针对某一用户获取其所有采集器信息

- 参数说明

请求参数	参数说明
UserId	用户 ID

- 返回值：List<RCollectorTO>（采集器信息）

属性	说明

Int id	主键，采集器 ID（系统会自动生成，从 1 开始累积生成）
String collectName	采集器名称
String host	主机地址
String ip	主机 IP
String mac	MAC 地址
int port	端口
int userId	隶属用户
RCollectorRunType sts	状态：RUNNING/运行、SHUTDOWN/停止、PAUSED/暂停
long eventCount	事件总数
long createTime	创建时间
long modifyTime	更新时间

- 异常

返回异常	异常说明
RException	
RReturnNullException	Null
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
```

```
//采集器管理接口服务
       TMultiplexedProtocol collectorSP = new TMultiplexedProtocol(protocol,
RCollectService.class.getName());
     collect = new RCollectService.Client(collectorSP);
}
  @Test
    public void testCollectors() throws Exception {
       //查看用户 ID 为 1 下的所有采集器列表
       List ll=collect.listCollectByUserID(1);
//       System.out.println(ll);
       System.out.println(collect.getCollectorById(2));
    )
```

3.3 通过采集器 ID 获取采集器信息（getCollectorById）

- 方法定义：

  RCollectorTO getCollectorById(int collectorId) throws RException, RRet
  urnNullException, TException, TException

- 方法说明

  基于采集器 ID 从而获取采集器信息

- 参数说明

请求参数	参数说明
collectorId	采集器 ID

- 返回值：RCollectorTO

属性	说明
Int id	主键，采集器 ID（系统会自动生成，从 1 开始累积生成）
String collectName	采集器名称

String host	主机地址
String ip	主机 IP
String mac	MAC 地址
int port	端口
int userId	隶属用户
RCollectorRunType sts	状态：RUNNING(运行)/SHUTDOWN(停止)/PAUSED(暂停)
long eventCount	事件总数
long createTime	创建时间
long modifyTime	更新时间

- 异常

返回异常	异常说明
RException	
RReturnNullException null	
TException	Thrift 异常

- 示例

```
@Test
public void getCollectorById() {
    try {
        RCollectorTO collector = client.getCollectorById(2);
        System.out.println(collector);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

3.4 获取指定 ID 采集器的监控明细（getDetailsById）

- 方法定义

  List<RIntelligenceTO> getDetailsById(int collectorId, String type, final boolean fuzzy), TException

- 方法说明

  通过采集器 ID 获取采集器的监控明细

- 参数说明

请求参数	参数说明
collectorId	采集器 ID
Type	采集类型，其中包括：SYSLOG、TEXT、SNMP、TELNET
fuzzy	针对采集类型，进行模糊匹配

- 返回值 List<RIntelligenceTO>

属性	说明
String courseName	任务名称
RCourseState state	任务状态：NORMAL（正常）/OVER(结束)/ERROR(异常)
String type	采集器类型
long acceptedCount	采集成功事件数
long errorCount	采集错误事件数
long completeCount	消费成功事件数
long failedCount	消费失败事件数
long monitorTime	监控信息时间
Map<String,String> header	头信息

- 异常

返回异常	异常说明
RException	
RReturnNullException null	
TException	Thrift 异常

- 示例

```
@Test
public void getDetailsById() {
    try {
        List<RIntelligenceTO> details =
client.getDetailsById(8, "TEXT", false);
        System.out.println(details);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

3.5 暂停采集器（pauseCollector）

- 方法定义

  boolean pauseCollector(int collectorId) throws RException, TException

- 方法说明

  暂停采集器

- 参数说明

请求参数	参数说明
collectorId	采集器 ID

- 返回值 boolean

变量	说明
boolean	是否成功

- 异常

返回异常	异常说明
RException	
TException	Thrift 异常

- 示例

```
@Test
public void pauseCollector() {
    try {
        boolean is = client.pauseCollector(8);
        System.out.println(is);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

3.6 重置采集器（resumeCollector）

- 方法定义

  boolean resumeCollector(int collectorId) throws RException, TException

- 方法说明

  重置采集器

- 参数说明

请求参数	参数说明
collectorId	采集器 ID

- 返回值 boolean

Boolean	说明
boolean	是否成功

- 异常

返回异常	异常说明
RException	
TException	Thrift 异常

- 示例

```
@Test
public void resumeCollector() {
    try {
        boolean is = client.resumeCollector(8);
        System.out.println(is);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

3.7 删除采集器（delCollector）

- 方法定义

  boolean delCollector(int collectorId) throws RException, TException

- 方法说明

  删除采集器

- 参数说明

请求参数	参数说明
collectorId	采集器 ID

- 返回值 boolean

Boolean	说明
boolean	是否成功

- 异常

返回异常	异常说明
RException	
TException	Thrift 异常

- 示例

```
@Test
public void delCollector() {
    try {
        boolean is = client.delCollector(8);
```

```
            System.out.println(is);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

4 数据库采集管理接口说明（RDatabaseInfoService.Iface）

4.1 方法说明概要

方法名称	说明
添加用户采集数据库信息（insertDataBaseInfo）	添加用户采集数据库信息
查询用户连接数据库信息（getDatabaseInfo）	查询用户连接数据库信息
删除用户采集数据库信息（deleteDataBaseInfo）	删除用户采集数据库信息。
验证创建连接时 URL 别名是否重名（valiDBUniqueName）	验证创建连接时 URL 别名是否重名。
通过采集器 ID 获取所有采集的数据库集合（getDatabaseInfos）	查询用户采集表名。

4.2 添加用户采集数据库信息（insertDataBaseInfo）

- 方法定义

  int insertDataBaseInfo(RDatabaseInfoEntityTO rdatabaseInfoEntity, String mac) throws RException, RReturnNullException , TException

- 方法说明：添加用户采集数据库信息

- 参数说明

参数	参数类型	说明
rdatabaseInfoEntity	RDatabaseInfoEntityTO	数据库实体对象

Mac	String	采集器所在机器的 mac

- 对象说明：rdatabaseInfoEntity

属性	说明
int id	主键，数据库 ID（系统会自动生成，从 1 开始累积生成）
DatabaseEnums.RDatabaseType rDatabaseType	数据库类型：MYSQL/ORACLE/SQLSEVER
String connURL	数据库 URL
String dbUniqueName	数据库 URL 别名
String databaseName	数据库名称
String dbUSerName	用户名
String dbPassword	密码
String dbSchema	schema 信息(oracle 数据库类型)
int userID	用户 ID
long createTime	创建时间
long modifyTime	修改时间
String remark	描述
List<int>tableList	采集表集合
short status	状态
int dbCollect	采集器 ID

- 返回值：int

Integer	说明
int	插入的 id（数据库 id）

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
RDatabaseInfoService.Client client;
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 6
0 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol = new TBinaryProtocol(transport);
        // TProtocol protocol = new TCompactProtocol(transpor
t);
        TMultiplexedProtocol serviceProtocol = new TMultiplexed
Protocol(protocol, RCollectMonitorService.class.getName());

        client = new RDatabaseInfoService.Iface (serviceProtoco
l); }
@Test
public void insertDataBaseInfo(){

    RDatabaseInfoEntityTO rdatabaseInfoEntity = new RDatabaseI
nfoEntityTO();
    rdatabaseInfoEntity.setDbUniqueName("Lee");
    rdatabaseInfoEntity.setDatabaseName("aleiyeX");
    rdatabaseInfoEntity.setRDatabaseType(RDatabaseType.ORACLE);
    rdatabaseInfoEntity.setDbCollectId(1);
    rdatabaseInfoEntity.setConnURL("ainana");
    rdatabaseInfoEntity.setDbPassword("aaa");
    rdatabaseInfoEntity.setUserId(1);
    rdatabaseInfoEntity.setDbUserName("luge");
    try {
        System.out.println(client.insertDataBaseInfo(rdatabase
```

```
InfoEntity));
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

4.3 查询用户连接数据库信息（getDatabaseInfo）

- 方法定义

  RDatabaseInfoEntityTO getDatabaseInfo(int id); throws RException，RReturnNullException，TException

- 方法说明

  查询用户连接数据库信息

- 参数说明

参数	说明
ID	数据库 ID

- 返回值：RDatabaseInfoEntityTO（数据库信息对象）
- 对象说明：

属性	说明
int id	主键，数据库 ID（系统会自动生成，从 1 开始累积生成）
DatabaseEnums.RDatabaseType rDatabaseType	数据库类型：MYSQL/ORACLE/SQLSEVER
String connURL	数据库 URL
String dbUniqueName	数据库 URL 别名
String databaseName	数据库名称
String dbUSerName	用户名
String dbPassword	密码
String dbSchema	schema 信息(oracle 数据库类型)
int userID	用户 ID
long createTime	创建时间
long modifyTime	修改时间
String remark	描述

List<int>tableList	采集表集合
short status	状态
int dbCollect	采集器 ID

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
RDatabaseInfoService.Client client;
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 6
0 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol = new TBinaryProtocol(transport);
        // TProtocol protocol = new TCompactProtocol(transpor
t);
        TMultiplexedProtocol serviceProtocol = new TMultiplexed
Protocol(protocol, RCollectMonitorService.class.getName());

        client = new RDatabaseInfoService.Iface (serviceProtoco
l); }
@Test
public void getDatabaseInfo(){

    try {
        System.out.println(client.getDatabaseInfo(6));
```

```
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

4.4 删除用户采集数据库信息（deleteDataBaseInfo）

- 方法定义

  int deleteDataBaseInfo(int id, String mac, String dbUniqueName); throws RException, RReturnNullException , TException

- 方法说明

  删除用户采集数据库信息。

- 参数说明

参数	参数类型	说明
ID	Int	数据库 ID
Mac	String	采集器所在机器的 mac
dbUniqueName	String	数据库唯一别名

- 返回值：int

Integer	说明
int	0 失败/1 成功

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
RDatabaseInfoService.Client client;
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 6
0 * 1000);
        transport.open();
```

```
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol = new TBinaryProtocol(transport);
        // TProtocol protocol = new TCompactProtocol(transpor
t);
        TMultiplexedProtocol serviceProtocol = new TMultiplexed
Protocol(protocol, RCollectMonitorService.class.getName());
        client = new RDatabaseInfoService.Iface (serviceProtoco
l); }
@Test
public  void deleteDataBaseInfo(){

    try {
        System.out.println(client.deleteDataBaseInfo(7,"
EOACCB66C6EA","test"));
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

4.5 验证创建连接时 URL 别名是否重名（valiDBUniqueName）

- 方法定义

  Boolean valiDBUniqueName(String dbUniqueName); throws RException , RReturnNullException , TException

- 方法说明

  验证创建连接时 URL 别名是否重名。

- 参数说明

参数	参数类型	说明
dbUniqueName	String	数据库 URL 别名

- 返回值：Boolean

Boolean	说明
Boolean	true 未重名/false 重名

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
RDatabaseInfoService.Client client;
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 6
0 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol = new TBinaryProtocol(transport);
        // TProtocol protocol = new TCompactProtocol(transpor
t);
        TMultiplexedProtocol serviceProtocol = new TMultiplexed
Protocol(protocol, RCollectMonitorService.class.getName());
        client = new RDatabaseInfoService.Iface
(serviceProtocol); }
@Test
public  void valiDBUniqueName(){

    try {
        System.out.println(client.valiDBUniqueName("aaa"));
    } catch (TException e) {
```

```
            e.printStackTrace();
        }
    }
```

4.6 通过采集器 ID 获取所有采集的数据库集合（getDatabaseInfos）

- 方法定义

  List<RDatabaseTableEntiLtyTO> getDatabaseInfos (int collectId); throws RException, RReturnNullException , TException

- 方法说明

  查询用户采集表名。

- 参数说明

参数	参数类型	说明
collectId	int	采集器 ID

- 返回值：RDatabaseInfoEntityTO（数据库信息集合）
- 对象说明：

属性	说明
int id	主键，数据库 ID（系统会自动生成，从 1 开始累积生成）
DatabaseEnums.RDatabaseType rDatabaseType	数据库类型：MYSQL/ORACLE/SQLSEVER
String connURL	数据库 URL
String dbUniqueName	数据库 URL 别名
String databaseName	数据库名称
String dbUSerName	用户名
String dbPassword	密码
String dbSchema	schema 信息(oracle 数据库类型)
int userID	用户 ID
long createTime	创建时间
long modifyTime	修改时间
String remark	描述
List<int>tableList	采集表集合
short status	状态

int dbCollect	采集器 ID

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
RDatabaseInfoService.Client client;
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 6
0 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        // TProtocol protocol = new TJSONProtocol(transport);
        TProtocol protocol = new TBinaryProtocol(transport);
        // TProtocol protocol = new TCompactProtocol(transpor
t);
        TMultiplexedProtocol serviceProtocol = new TMultiplexed
Protocol(protocol, RCollectMonitorService.class.getName());
        client = new RDatabaseInfoService.Iface
(serviceProtocol); }
@Test
public  void getDatabaseInfos(){

    try {
        System.out.println(client.getDatabaseInfos(1));
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

5 采集表信息管理接口说明（RDatabaseTableService.Iface）

5.1 方法说明概要

方法名称	说明
添加用户采集表名（insertCollectTable）	添加用户采集表名
查询用户采集表名（getCollectTable）	查询用户采集表名

5.2 添加用户采集表名（insertCollectTable）

- 方法定义

  int insertCollectTable(RDatabaseTableEntityTO rdatabasetableEntity)；

  throws RException，RReturnNullException ，TException

- 方法说明

  添加用户采集表名

- 参数说明

参数	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int databaseId	对应数据库 ID
String tableName	表名
long createTime	创建时间
long lastCollectTime	最后采集时间
int increPeriod	自增周期
String remark	描述
List<int>fieldList	字段集合
DatabaseEnums.RImportType rImoprtType	是否是增量导入 1:增量,2:全量
String increFieldName	自增字段名称
DatabaseEnums.RIncreFieldType rincreFieldType	字段类型 0:LONG,1:STRING,2:DATATIME
String increFieldValue	自增字段最后的值

short status	状态

- 返回值：int

Integer	说明
int	0 失败/1 成功

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("127.0.0.1", 6060, 60
* 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol,
RDatabaseTableService.class.getName());

    client = new
RDatabaseTableService.Client(serviceProtocol);
}

@Test
```

```
public void insertCollectTable(){

    List rdatabasetableEntity = new ArrayList();
    RDatabaseTableEntityTO rdatabasetableEntity1 = new
RDatabaseTableEntityTO();
    rdatabasetableEntity1.setUserId(1);
    rdatabasetableEntity1.setDatabaseId(9);
    rdatabasetableEntity1.setIncreFieldName("QQ");
    rdatabasetableEntity1.setIncrePeriod("0 0 2/* * * ?");
    rdatabasetableEntity1.setTableName("XX");
    RDatabaseTableEntityTO rdatabasetableEntity2 = new
RDatabaseTableEntityTO();
    rdatabasetableEntity2.setUserId(1);
    rdatabasetableEntity2.setDatabaseId(9);
    rdatabasetableEntity2.setIncreFieldName("MM");
    rdatabasetableEntity2.setIncrePeriod("0 0 0 2/ * * ?");
    rdatabasetableEntity2.setTableName("HH");
    rdatabasetableEntity.add(rdatabasetableEntity1);
    rdatabasetableEntity.add(rdatabasetableEntity2);
    try {

System.out.println(client.insertCollectTable(rdatabasetabl
eEntity));
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

输出结果：

```
1
```

5.3 查询用户采集表名（getCollectTable）

- 方法定义

list<RDatabaseTableEntityTO> getCollectTable(int databaseId)； throws RException，RReturnNullException ，TException

- 方法说明

  查询用户采集表名

- 参数说明

请求参数	参数说明
databaseId	数据库 ID

- 返回值：list<RDatabaseTableEntityTO>

RDatabaseTableEntityTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int databaseId	对应数据库 ID
String tableName	表名
long createTime	创建时间
long lastCollectTime	最后采集时间
int increPeriod	自增周期
String remark	描述
List<int>fieldList	字段集合
DatabaseEnums.RImportType rImoprtType	是否是增量导入 1:增量,2:全量
String increFieldName	自增字段名称
DatabaseEnums.RIncreFieldType rincreFieldType	字段类型 0:LONG,1:STRING,2:DATATIME
String increFieldValue	自增字段最后的值
short status	状态

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Test
public void getCollectTable(){

    try {
        System.out.println(client.getCollectTable(8));
    } catch (TException e) {
        e.printStackTrace();
    }
}
```

输出结果：

```
[RDatabaseTableEntityTO(id:71, userId:1, databaseId:8, tableName:good, createT
ime:0, lastCollectTime:0, increPeriod:0 0 0 2/ * * ?, remark:null, fieldList:nu
ll, rImportType:null, increFieldName:bbb, rincreFieldType:null, increFieldValu
e:null, status:1), RDatabaseTableEntityTO(id:70, userId:1, databaseId:8, table
Name:nice, createTime:0, lastCollectTime:0, increPeriod:0 0 2/* * * ?, remark:n
ull, fieldList:null, rImportType:null, increFieldName:aaa, rincreFieldType:nul
l, increFieldValue:null, status:1)]
```

6 仪表盘接口说明（RDashBoardService.Iface）

6.1 方法说明概要

方法名称	说明
添加仪表盘列表信息（insert）	添加新仪表盘列表信息，参数 entity 中属性 ID 必须为 null，name 和 userId 必须不为空
修改仪表盘列表信息（modify）	修改已有仪表盘列表信息，更新非 null 值的属性

	值信息
删除仪表盘列表信息（deleteById）	删除一条仪表盘列表信息。参数 entity 只有 ID 属性有效，不能为空，删除该 ID 对应的数据。
查询仪表盘列表信息（queryList）	查询仪表盘列表信息，按创建时间逆序排列。
查询仪表盘单条列表信息（queryBean）	查询一条仪表盘列表信息
查询是否有重名仪表盘列表信息（queryBeanByName）	查询一条仪表盘列表信息
查询是否有重名仪表盘列表信息（queryBeanByName）	添加仪表盘图表配置信息
添加仪表盘图表配置信息（insertConf）	根据用户 ID 和位置下标，更新相关图表配置信息中的查询参数和刷新时间
更新仪表盘图表配置信息（updateByCurridAndGridIndex）	更新仪表盘图表配置信息
查询仪表盘图表配置信息（queryListByCurrid）	查询仪表盘图表配置信息
查询首页仪表盘图表配置信息（queryIndexListByCurrid）	查询首页仪表盘图表配置信息
删除仪表盘图表配置信息（deleteByCurridAndGridIndex）	根据用户 ID 和位置下标删除相关图表配置信息
查询首页仪表盘图表配置信息（queryBeanByGridIndex）	根据列表信息 ID、用户 ID、位置下标查询相关图表配置信息
添加或更新首页仪表盘列表信息配置信息（updateDashIndexCfg）	添加或者更新首页仪表盘展示配置信息
查询首页仪表盘列表信息配置信息（queryIndexCfg）	查询首页仪表盘展示配置信息

6.2 添加仪表盘列表信息（insert）

- 方法定义

  boolean insert (RUserDashboardEntity entity) throws TException

- 方法说明：添加新仪表盘列表信息，参数 entity 中属性 ID 必须为 null，name

和 userId 必须不为空

- 参数说明

请求参数 RUserDashboardEntity 类	参数说明
Int id	主键 ID，插入时必须为空
String name	仪表盘列表信息名称
String description	仪表盘列表信息描述
Int userId	所属用户 ID，不能为空
long createTime	创建时间（为空时默认为系统当前时间）
long modifyTime	修改时间

- 返回值 boolean

boolean	说明
boolean	True：添加成功 False：添加失败

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
```

```
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol,
RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testInsertDashBoard() throws TException {
    RUserDashboardEntity rd = new RUserDashboardEntity();
    rd.setUserId(1);
    rd.setName("testDemo1");
    rd.setDiscription("test_demo_1");
    boolean flag = dashBoardService.insert(rd);
    System.out.println(flag);
}
```

6.3 修改仪表盘列表信息（modify）

- 方法定义：boolean modify (RUserDashboardEntity entity) throws TException e
- 方法说明：修改已有仪表盘列表信息，更新非 null 值的属性值信息
- 参数说明：

请求参数 RUserDashboardEntity 类	参数说明
Int id	数据主键 ID，不能为空
String name	仪表盘列表信息名称
String description	仪表盘列表信息描述
Int userId	所属用户 ID，不能为空
long createTime	创建时间（为空时默认为系统当前时间）

long modifyTime	修改时间

- 返回值 boolean

boolean	说明
boolean	True：成功 False：失败

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testModifyDashBoard() throws TException {
    RUserDashboardEntity rd = new RUserDashboardEntity();
    rd.setId(1);
    rd.setUserId(1);
    rd.setName("testDemo1");
    rd.setDiscription("test_demo_2223");
    boolean flag = dashBoardService.modify(rd);
```

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

6.4 删除仪表盘列表信息（deleteById）

- 方法定义：boolean deleteById (RUserDashboardEntity entity) throws TException e
- 方法说明：删除一条仪表盘列表信息。参数 entity 只有 ID 属性有效，不能为空，删除该 ID 对应的数据。
- 参数说明：

请求参数 RUserDashboardEntity 类	参数说明
Int id	数据主键 ID，不能为空

- 返回值 boolean

boolean	说明
boolean	True：成功 False：失败

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
```

```
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}


@Test
public void testDelDashBoard() throws TException {
    RUserDashboardEntity rd = new RUserDashboardEntity();
    rd.setId(1);
    rd.setUserId(1);
    boolean flag = dashBoardService.deleteById(rd);
    System.out.println(flag);
}
```

6.5 查询仪表盘列表信息（queryList）

- 方法定义：List<RUserDashboardEntity> queryList(RUserDashboardEntity entity) throws TException e
- 方法说明：查询仪表盘列表信息，按创建时间逆序排列。参数 entity 中只有 userId 属性有效，userId 为 null 时结果集为所有仪表盘信息，userId 不为 null 时结果集为该用户创建的仪表盘信息。若结果为空则抛出 TException 异常。需对异常捕获并做对应处理。
- 参数说明：

请求参数 RUserDashboardEntity 类	参数说明
Int userId	所属用户 ID

- 返回值 List<RUserDashboardEntity>

类型	说明
RUserDashboardEntity	仪表盘列表数据

RUserDashboardEntity	说明
Intid	主键 ID
Stringname	仪表盘列表信息名称
String description	仪表盘列表信息描述
intuserId	所属用户 ID
Long createTime	创建时间（为空时默认为系统当前时间）
Long modifyTime	修改时间

返回异常	异常说明
TException	结果为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryListDashBoard() {
    try {
```

```
        RUserDashboardEntity rd = new RUserDashboardEntity();
        rd.setUserId(1);
        List<RUserDashboardEntity> list = dashBoardService.queryList(rd);
        System.out.println(JSON.toJSONString(list));
    } catch (TException e) {
        System.out.println("异常");
    }
}
```

6.6 查询仪表盘单条列表信息（queryBean）

- 方法定义：RUserDashboardEntity queryBean (RUserDashboardEntity entity) throws TException e
- 方法说明：查询一条仪表盘列表信息。参数 entity 中 id、userId 和 name 属性为可选过滤条件，返回结果集为与属性值相同的数据。若结果集包含多条数据，则返回第一条数据。若结果为空则抛出 TException 异常。需对异常捕获并做对应处理。
- 参数说明

请求参数 RUserDashboardEntity 类	参数说明
Int id	数据主键 ID
String name	仪表盘列表信息名称
Int userId	所属用户 ID

返回值

类型	说明
RUserDashboardEntity	仪表盘列表数据

返回异常	异常说明
TException	结果为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryBeanDashBoard() {
    try {
        RUserDashboardEntity rd = new RUserDashboardEntity();
        rd.setName("testDemo1");
        rd.setUserId(1);
        RUserDashboardEntity entity = dashBoardService.queryBean(rd);
        System.out.println(JSON.toJSONString(entity));
    } catch (TException e) {
        System.out.println("无结果或者其他异常");
    }
}
```

6.7 查询是否有重名仪表盘列表信息（queryBeanByName）

- 方法定义：RUserDashboardEntity queryBeanByName (RUserDashboardEntity entity) throws TException e

- 方法说明：查询一条仪表盘列表信息。参数 entity 中 userId 和 name 属性为必选过滤条件，返回结果集为与属性值相同的数据。若结果集包含多条数据，则返回其中第一条数据。若结果为空则抛出 TException 异常。需对异常捕获并做对应处理。

- 参数说明

请求参数 RUserDashboardEntity 类	参数说明
String name	仪表盘列表信息名称，必须为有效值
Int userId	所属用户 ID，必须为有效值

- 返回值：RUserDashboardEntity

类型	说明
RUserDashboardEntity	一条仪表盘列表数据

RUserDashboardEntity	说明
Intid	主键 ID
Stringname	仪表盘列表信息名称
String description	仪表盘列表信息描述
intuserId	所属用户 ID
Long createTime	创建时间（为空时默认为系统当前时间）
Long modifyTime	修改时间

返回异常	异常说明

TException	结果为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryBeanByNameDashBoard () {
    try {
        RUserDashboardEntity rd = new RUserDashboardEntity();
        rd.setName("testDemo2");
        rd.setUserId(1);
        RUserDashboardEntity entity = dashBoardService.queryBean(rd);
        System.out.println(JSON.toJSONString(entity));
    } catch (TException e) {
        System.out.println("无结果或者其他异常");
    }
}
```

6.8 添加仪表盘图表配置信息（insertConf）

- 方法定义：boolean insertConf (RuserDashBoardConfEntity entity) throws TException e

- 方法说明：添加仪表盘图表配置信息，参数 entity 中属性 ID 必须为 null，其他值必须为有效值。
- 参数说明：

请求参数 RUserDashboardEntity 类	参数说明
Int did	数据主键 ID
Int dashId	所属仪表盘列表信息数据 ID
Int currid	所属用户 ID
Int gridIndex	九宫格位置下标(取值 1-9 的整数)
Int reportInfoId	报表查询参数数据主键 ID
Int refreshTime	图表刷新时间（秒）

- 返回值：boolean

类型	说明
boolean	True 添加成功 False 添加失败

返回异常	异常说明
TException	

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
```

```
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
           new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}


    @Test
    public void testInsertConf() throws TException {
        RUserDashBoardConfEntity cf = new RUserDashBoardConfEntity();
//        cf.setDid();
        cf.setDashId(2);
        cf.setCurrid(1);
        cf.setGridIndex(1);
        cf.setReportInfoId(130);
        cf.setRefreshTime(300);
        boolean flag = dashBoardService.insertConf(cf);
        System.out.println(flag);
    }
```

6.9 更新仪表盘图表配置信息（updateByCurridAndGridIndex）

- 方法定义：boolean updateByCurridAndGridIndex (RuserDashBoardConfEntity entity) throws TException e
- 方法说明：根据用户 ID 和位置下标，更新相关图表配置信息中的查询参数和刷新时间。查询参数数据主键 ID 和刷新时间必须至少有一个为有效值。
- 参数说明：

请求参数 RUserDashboardEntity 类	参数说明
Int dashId	所属仪表盘列表信息项 ID，必须为有效值

Int currid	所属用户 ID，必须为有效值
Int gridIndex	九宫格位置下标(取值 1-9 的整数) ，必须为有效值
Int reportInfoId	报表查询参数数据主键 ID，可选
Int refreshTime	图表刷新时间（秒），可选

- 返回值：boolean

类型	说明
boolean	True：成功 False：失败

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testUpdateConf() throws TException {
```

```
    RUserDashBoardConfEntity cf = new RUserDashBoardConfEntity();
    cf.setDashId(2);
    cf.setCurrid(1);
    cf.setGridIndex(1);
    cf.setReportInfoId(130);
    cf.setRefreshTime(600);
    boolean flag = dashBoardService.updateByCurridAndGridIndex(cf);
    System.out.println(flag);
}
```

6.10 查询仪表盘图表配置信息（queryListByCurrid）

- 方法定义：

List<RuserDashBoardConfEntity> queryListByCurrid (long currId, int dashId) throws TException e

- 方法说明：查询仪表盘图表配置信息。参数 entity 中 currId 和 dashId 必须为有效值。
- 参数说明:

请求参数 RUserDashboardEntity 类	参数说明
long currId	所属用户 ID
Int dashId	所属仪表盘列表信息 ID

返回值:list

类型	说明
List	仪表盘图表配置数据

返回异常	异常说明

TException	结果为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryListByCurrid() throws TException {
    List<RUserDashBoardConfEntity> list = dashBoardService.queryListByCurrid(1L, 2);
    System.out.println(JSON.toJSONString(list));
}
```

6.11 查询首页仪表盘图表配置信息（queryIndexListByCurrid）

- 方法定义:List<RuserDashBoardConfEntity> queryIndexListByCurrid (long userId) throws TException e
- 方法说明：查询仪表盘图表配置信息。参数 userId 必须为有效值。

请求参数	参数说明
long userId	用户 ID

- 返回值：list

类型	说明

List	仪表盘图表配置数据

返回异常	异常说明
TException	结果为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryIndexListByCurrid() throws TException {
    List<RUserDashBoardConfEntity> list = dashBoardService.queryIndexListByCurrid(1L);
    System.out.println(JSON.toJSONString(list));
}
```

6.12 删除仪表盘图表配置信息（deleteByCurridAndGridIndex）

- 接口定义：boolean deleteByCurridAndGridIndex (RuserDashBoardConfEntity entity) throws TException e
- 接口说明：根据用户 ID 和位置下标删除相关图表配置信息。查询参数 ID 和刷新时间都必须为有效值。

请求参数 RUserDashboardEntity 类	参数说明
Int dashId	所属仪表盘列表信息项 ID，必须为有效值
Int currid	所属用户 ID，必须为有效值
Int gridIndex	九宫格位置下标(取值 1-9 的整数) ，必须为有效值

- 返回值：boolean

类型	说明
boolean	True：成功 False：失败

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}
```

```
@Test
public void testDelConf() throws TException {
    RUserDashBoardConfEntity cf = new RUserDashBoardConfEntity();
    cf.setDashId(2);
    cf.setCurrid(1);
    cf.setGridIndex(1);
    boolean flag = dashBoardService.deleteByCurridAndGridIndex(cf);
    System.out.println(flag);
}
```

6.13 查询首页仪表盘图表配置信息（queryBeanByGridIndex）

- 接口定义：RuserDashBoardConfEntity queryBeanByGridIndex (RuserDashBoardConfEntity entity) throws TException e
- 方法说明：根据列表信息 ID、用户 ID、位置下标查询相关图表配置信息。参数均必须为有效值，返回结果集为与属性值相同的一条数据。若结果集匹配多条数据，则返回第一条数据。若结果集为空则抛出 TException 异常。需对异常捕获并做对应处理。

请求参数 RUserDashboardEntity 类	参数说明
Int dashId	所属仪表盘列表信息 ID，必须为有效值
Int currid	所属用户 ID，必须为有效值
Int gridIndex	九宫格位置下标(取值 1-9 的整数) ，必须为有效值

- 返回值：RuserDashBoardConfEntity

类型	说明
RuserDashBoardConfEntity	配置信息

RuserDashBoardConfEntity	说明
Intdid	数据主键 ID
Int dashId	所属仪表盘列表信息数据 ID
Int currid	所属用户 ID
Int gridIndex	九宫格位置下标(取值 1-9 的整数)
Int reportInfoId	报表查询参数数据主键 ID
Int refreshTime	图表刷新时间（秒）

返回异常	异常说明
TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryBeanByGridIndex() {
    try {
        RUserDashBoardConfEntity cf = new RUserDashBoardConfEntity();
        cf.setDashId(2);
```

```
        cf.setCurrid(1);
        cf.setGridIndex(2);
        RUserDashBoardConfEntity entity = dashBoardService.queryBeanByGridIndex(cf);
        System.out.println(JSON.toJSONString(entity));
    } catch (TException e) {
        e.printStackTrace();
        System.out.println("无结果或者其他异常");
    }
}
```

6.14 添加或更新首页仪表盘列表信息配置信息（updateDashIndexCfg）

- 方法定义：boolean updateDashIndexCfg (RuserDashIndexCfg entity) throws TException e
- 方法说明：添加或者更新首页仪表盘展示配置信息。参数 entity 的 user_id 和 dash_id 均必须为有效值。若原配置信息不存在时则进行添加，否则进行更新。

请求参数 RUserDashIndexCfg 类	参数说明
Int user_id	所属用户 ID，必须为有效值
Int dash_id	仪表盘列表信息 ID，必须为有效值

- 返回值：boolean

类型	说明
boolean	True：成功 False：失败

返回异常	异常说明

TException	其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testUpdateDashIndexCfg() throws TException {
    RUserDashIndexCfg cf = new RUserDashIndexCfg();
    cf.setDash_id(2);
    cf.setUser_id(1);
    boolean flag = dashBoardService.updateDashIndexCfg(cf);
    System.out.println(flag);
}
```

6.15 查询首页仪表盘列表信息配置信息（queryIndexCfg）

- 方法定义：RuserDashIndexCfg queryIndexCfg (long userId) throws TException e
- 方法说明：查询首页仪表盘展示配置信息。参数 user_id 必须为有效值。若结果为空则抛出 TException 异常。需对异常捕获并做对应处理。
- 参数说明：

请求参数 RUserDashIndexCfg 类	参数说明

Int userId	所属用户 ID，必须为有效值

- 返回值：RuserDashIndexCfg

类型	说明
RuserDashIndexCfg	首页仪表盘配置信息

RuserDashIndexCfg	说明
Int user_id	主键 ID
Int dash_id	仪表盘列表信息 ID

返回异常	异常说明
TException	结果集为空或者其他异常

- 示例

```
@Before
public void before() throws Exception {
    transport = new TSocket("localhost", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDashBoardService.class.getName());
    dashBoardService = new RDashBoardService.Client(serviceProtocol);
}

@Test
public void testQueryIndexCfg() throws TException {
```

```
    RUserDashIndexCfg entity = dashBoardService.queryIndexCfg(1);
    System.out.println(JSON.toJSONString(entity));
}
```

7 SNMP 采集信息管理接口说明（RSnmpCollectService.Iface）

7.1 方法说明概要

方法名称	说明
添加 SNMP 配置信息（insertSnmpConfList）	添加 SNMP 配置信息，此方法是传入要添加的列表后多条添加 SNMP 采集信息。
删除 SNMP 配置信息（delSnmpConfList）	删除 SNMP 配置信息，此方法是传入要删除的配置 id 列表后多条删除 SNMP 采集信息。
修改 SNMP 配置信息（updateSnmpConfList	）修改 SNMP 配置信息，此方法是传入要修改的列表后多条修改 SNMP 采集信息。
根据 userId 查询 SNMP 配置信息（selectSnmpConfListByUserId）	查询 SNMP 配置信息，根据用户 id 查询出所有 snmp 配置信息
根据 collectId 查询 SNMP 配置信息（selectSnmpConfListByCollectId）	查询 SNMP 配置信息，根据采集器 id 查询出所有 snmp 配置信息
添加和修改 SNMP 配置信息（insertOrUpdateSnmpConfList）	添加和修改 SNMP 配置信息，此方法是传入要添加或者修改的列表后根据是否有 id 多条添加或者修改 SNMP 采集信息

7.2 添加 SNMP 配置信息（insertSnmpConfList）

- 方法定义：**int** insertSnmpConfList (List<RSnmpConfModelTO> snmpConfModelList) t**hrows** RException，RReturnNullException ，TException;
- 方法说明：添加 SNMP 配置信息，此方法是传入要添加的列表后多条添加 SNMP 采集信息。
- 参数说明：

请求参数	参数说明
List<RSnmpConfModelTO>	SNMP 采集配置信息列表

RSnmpConfModelTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int collectId	采集器 ID
String dataType	采集数据类型
String collectType	设备采集类型（cpu,内存,温度）
String driverIpAddress	被采集的网络设备 ip
int port	Snmp 端口
String remark	描述
int brandDeviceId	设备厂商关联 id
List<String> oids	采集 oid 列表
String snmpVersion	Snmp 版本
long period	采集间隔时间

String community	V2 版本 Community 字符串
String userName	V3 版本用户名
String password	V3 版本密码
String encoding	编码格式
String mac	采集器 mac 地址
Sstring oid	页面 oid
int brandId	厂商 id
String brandName	厂商名称
int deviceId	设备 id
String deviceName	设备名
String status	是否删除
long createTime	创建时间
long modifyTime	修改时间
String timeZone	时区
List<Map<String,String>> couseList	相同设备 ip 采集配置列表

- 返回值：int

Integer	说明
int	0 失败/n 成功

返回异常	异常说明
RException	

RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        TProtocol protocol = new TBinaryProtocol(transport);

        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
@Test
public void insertSnmpConfListService() throws TException {

    List<RSnmpConfModelTO> snmpConfList = new ArrayList<RSnmpConfModelTO>();

    RSnmpConfModelTO snmpConf = new RSnmpConfModelTO();

    snmpConf.setId(85);

    snmpConfList.add(snmpConf);

    int num = client.insertSnmpConfList(snmpConfList);
```

```
    System.out.println(num);

}
```

7.3 删除 SNMP 配置信息（delSnmpConfList）

- 方法定义：int delSnmpConfList (List<Integer> id,String mac) throws RException, RReturnNullException , TException;
- 方法说明：删除 SNMP 配置信息，此方法是传入要删除的配置 id 列表后多条删除 SNMP 采集信息。
- 参数说明：

请求参数	参数说明
List<Integer> id	SNMP 采集配置 id 列表
String mac	采集器 mac 地址

- 返回值：int

Integer	说明
int	0 失败/n 成功

- 示例

```
    @Before
        public void before() throws Exception {
            TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
            transport.open();
            transport = new TFramedTransport(transport);
            // tt.open();
            // 协议要和服务端一致
            TProtocol protocol = new TBinaryProtocol(transport);
```

```
        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
@Test
public void delSnmpConfListService() throws TException {

    List<Integer> idList = new ArrayList<Integer>();
    idList.add(85);

    int num = client.delSnmpConfList(idList,"E0ACCB66C6E6");

    System.out.println(num);

}
```

7.4 修改 SNMP 配置信息（updateSnmpConfList）

- 方法定义：int updateSnmpConfList (List<RSnmpConfModelTO> snmpConfModelList) throws RException, RReturnNullException , TException;
- 方法说明：修改 SNMP 配置信息，此方法是传入要修改的列表后多条修改 SNMP 采集信息。
- 参数说明：

请求参数	参数说明
List<RSnmpConfModelTO>	SNMP 采集配置信息列表

RSnmpConfModelTO	说明

int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int collectId	采集器 ID
String dataType	采集数据类型
String collectType	设备采集类型（cpu,内存,温度）
String driverIpAddress	被采集的网络设备 ip
int port	Snmp 端口
String remark	描述
int brandDeviceId	设备厂商关联 id
List<String> oids	采集 oid 列表
String snmpVersion	Snmp 版本
long period	采集间隔时间
String community	V2 版本 Community 字符串
String userName	V3 版本用户名
String password	V3 版本密码
String encoding	编码格式
String mac	采集器 mac 地址
Sstring oid	页面 oid
int brandId	厂商 id
String brandName	厂商名称

int deviceId	设备 id
String deviceName	设备名
String status	是否删除
long createTime	创建时间
long modifyTime	修改时间
String timeZone	时区
List<Map<String,String>> couseList	相同设备 ip 采集配置列表

- 返回值：int

Integer	说明
int	0 失败/n 成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        TProtocol protocol = new TBinaryProtocol(transport);
```

```
        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
    RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
    @Test
    public void updateSnmpConfListService() throws TException{

        List<RSnmpConfModelTO> snmpConfList = new ArrayList<RSnmpConfModelTO>();
        RSnmpConfModelTO snmpConf = new RSnmpConfModelTO();
        snmpConf.setId(85);
        snmpConfList.add(snmpConf);

        int num = client.updateSnmpConfList(snmpConfList);

        System.out.println(num);


    }

```

7.5 根据 userId 查询 SNMP 配置信息（selectSnmpConfListByUserId）

- 方法定义：List<RSnmpConfModelTO> selectSnmpConfListByUserId (int userId) throws RException, RReturnNullException , TException;
- 方法说明：查询 SNMP 配置信息，根据用户 id 查询出所有 snmp 配置信息。
- 参数说明：

请求参数	参数说明

int userId	用户 id

- 返回值：List<RSnmpConfModelTO>

List	参数说明
List<RSnmpConfModelTO>	SNMP 采集配置信息列表

RSnmpConfModelTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int collectId	采集器 ID
String dataType	采集数据类型
String collectType	设备采集类型（cpu,内存,温度）
String driverIpAddress	被采集的网络设备 ip
int port	Snmp 端口
String remark	描述
int brandDeviceId	设备厂商关联 id
List<String> oids	采集 oid 列表
String snmpVersion	Snmp 版本
long period	采集间隔时间
String community	V2 版本 Community 字符串
String userName	V3 版本用户名

String password	V3 版本密码
String encoding	编码格式
String mac	采集器 mac 地址
Sstring oid	页面 oid
int brandId	厂商 id
String brandName	厂商名称
int deviceId	设备 id
String deviceName	设备名
String status	是否删除
long createTime	创建时间
long modifyTime	修改时间
String timeZone	时区
List<Map<String,String>> couseList	相同设备 ip 采集配置列表

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
        transport.open();
```

```
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        TProtocol protocol = new TBinaryProtocol(transport);

        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
    RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
    @Test
    public void selectSnmpConfListByUserIdService() throws TException{

        List<RSnmpConfModelTO> snmpConfList = client.selectSnmpConfListByUserId(1);

        System.out.println(snmpConfList);

    }
```

7.6 根据 collectId 查询 SNMP 配置信息（selectSnmpConfListByCollectId）

- 方法定义：List<RSnmpConfModelTO> selectSnmpConfListByCollectId (int collectId) throws RException, RReturnNullException , TException;
- 方法说明：查询 SNMP 配置信息，根据采集器 id 查询出所有 snmp 配置信息。
- 参数说明：

请求参数	参数说明
int collectId	采集器 id

- 返回值：List<RSnmpConfModelTO>

List	参数说明
List<RSnmpConfModelTO>	SNMP 采集配置信息列表

RSnmpConfModelTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int collectId	采集器 ID
String dataType	采集数据类型
String collectType	设备采集类型（cpu,内存,温度）
String driverIpAddress	被采集的网络设备 ip
int port	Snmp 端口
String remark	描述
int brandDeviceId	设备厂商关联 id
List<String> oids	采集 oid 列表
String snmpVersion	Snmp 版本
long period	采集间隔时间
String community	V2 版本 Community 字符串
String userName	V3 版本用户名
String password	V3 版本密码
String encoding	编码格式

String mac	采集器 mac 地址
Sstring oid	页面 oid
int brandId	厂商 id
String brandName	厂商名称
int deviceId	设备 id
String deviceName	设备名
String status	是否删除
long createTime	创建时间
long modifyTime	修改时间
String timeZone	时区
List<Map<String,String>> couseList	相同设备 ip 采集配置列表

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
```

```
        TProtocol protocol = new TBinaryProtocol(transport);

        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
    RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
    @Test
    public void selectSnmpConfListByCollectIdService() throws TException{

        List<RSnmpConfModelTO> snmpConfList = client.selectSnmpConfListByCollectId(7);

        System.out.println(snmpConfList);

    }

```

7.7 添加和修改 SNMP 配置信息（insertOrUpdateSnmpConfList）

- 方法定义：int insertOrUpdateSnmpConfList (List<RSnmpConfModelTO> snmpConfModelList) throws RException, RReturnNullException , TException;
- 方法说明：添加和修改 SNMP 配置信息，此方法是传入要添加或者修改的列表后根据是否有 id 多条添加或者修改 SNMP 采集信息。
- 参数说明：

请求参数	参数说明
List<RSnmpConfModelTO>	SNMP 采集配置信息列表

RSnmpConfModelTO	说明

int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int userID	用户 ID
int collectId	采集器 ID
String dataType	采集数据类型
String collectType	设备采集类型（cpu,内存,温度）
String driverIpAddress	被采集的网络设备 ip
int port	Snmp 端口
String remark	描述
int brandDeviceId	设备厂商关联 id
List<String> oids	采集 oid 列表
String snmpVersion	Snmp 版本
long period	采集间隔时间
String community	V2 版本 Community 字符串
String userName	V3 版本用户名
String password	V3 版本密码
String encoding	编码格式
String mac	采集器 mac 地址
Sstring oid	页面 oid
int brandId	厂商 id
String brandName	厂商名称

int deviceId	设备 id
String deviceName	设备名
String status	是否删除
long createTime	创建时间
long modifyTime	修改时间
String timeZone	时区
List<Map<String,String>> couseList	相同设备 ip 采集配置列表

- 返回值：int

Integer	说明
int	0 失败/n 成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
    public void before() throws Exception {
        TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
        transport.open();
        transport = new TFramedTransport(transport);
        // tt.open();
        // 协议要和服务端一致
        TProtocol protocol = new TBinaryProtocol(transport);
```

```
        TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol,
RSnmpCollectServer.class.getName());

        client = new RSnmpCollectServer.Client(serviceProtocol);

    }
@Test
public void insertOrUpdateSnmpConfListService() throws TException{

    List<RSnmpConfModelTO> snmpConfList = new ArrayList<RSnmpConfModelTO>();

    RSnmpConfModelTO snmpConf = new RSnmpConfModelTO();
    snmpConf.setUserId(100);
    snmpConfList.add(snmpConf);

    int num = client.insertOrUpdateSnmpConfList(snmpConfList);

    System.out.println(num);

}
```

8 Syslog/SNMP 采集监控管理接口说明

（RCollectMonitorService.Iface）

8.1 方法说明概要

方法名称	说明
根据 collectId 查询 sylog 监控信息（getAllMonitot）	查询 syslog 监控信息，根

	据采集器 id 查询出所有 syslog 配置信息
修改 syslog 监控信息（updateCollectMonitor）	修改 syslog 监控信息，用于启动和停止 syslog 采集器
修改 snmp 监控信息（updateSnmpMonitor）	修改 snmp 监控信息，用于启动和停止 snmp 采集器
查询 snmp 监控信息（getSnmpMonitor）	修改 snmp 监控信息，用于启动和停止 snmp 采集器

8.2 根据 collectId 查询 sylog 监控信息（getAllMonitot）

- 方法定义：RCollectMonitorEntityTO getAllMonitot (int collectId) throws RException，RReturnNullException ，TException;
- 方法说明：查询 syslog 监控信息，根据采集器 id 查询出所有 syslog 配置信息。
- 参数说明：

请求参数	参数说明
int collectId	采集器 id

- 返回值：RCollectMonitorEntityTO

RCollectMonitorEntityTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int collectId	采集器 id
String syslogStatus	Syslog 采集状态(启动，停止)
String syslogProtocol	Syslog 采集协议

long createTime	创建时间
long modifyTime	修改时间
String mark	描述
String mac	采集器 mac 地址
int port	端口
int userId	用户 id

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol, RCollectMonitorService.class.getName());

    clientMonitor = new RCollectMonitorService.Client(serviceProtocol);
}

@Test
public void getAllMonitorService() throws TException{
    RCollectMonitorEntityTO rCollectMonitorEntityTO = clientMonitor.getAllMonitor(1);
    System.out.println(rCollectMonitorEntityTO);
}
```

8.3 修改 syslog 监控信息（updateCollectMonitor）

- 方法定义：int updateCollectMonitor (RCollectMonitorEntityTO collectMonitorEntity) throws RException，RReturnNullException ，TException;

- 方法说明：修改 syslog 监控信息，用于启动和停止 syslog 采集器。
- 参数说明：

请求参数	参数说明
RCollectMonitorEntityTO collectMonitorEntity	Syslog 监控实体

RCollectMonitorEntityTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int collectId	采集器 id
String syslogStatus	Syslog 采集状态(启动，停止)
String syslogProtocol	Syslog 采集协议
long createTime	创建时间
long modifyTime	修改时间
String remark	描述
String mac	采集器 mac 地址
int port	端口
int userId	用户 id

- 返回值：int

Integer	说明
int	0 失败/1 成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol, RCollectMonitorService.class.getName());

    clientMonitor = new RCollectMonitorService.Client(serviceProtocol);
}
```

```
@Test
public void updateCollectMonitorService() throws TException {
    RCollectMonitorEntityTO rCollectMonitorEntityTO = new RCollectMonitorEntityTO();
    rCollectMonitorEntityTO.setId(3);
    rCollectMonitorEntityTO.setSyslogProtocol("UDP");
    rCollectMonitorEntityTO.setSyslogStatus("停止");
    rCollectMonitorEntityTO.setMac("E0ACCB66C6E6");

    int num = clientMonitor.updateCollectMonitor(rCollectMonitorEntityTO);
    System.out.println(num);
}
```

8.4 修改 snmp 监控信息（updateSnmpMonitor）

- 方法定义：int updateCollectMonitor (RCollectMonitorEntityTO collectMonitorEntity) **throws** RException, RReturnNullException , TException;
- 方法说明：修改 snmp 监控信息，用于启动和停止 snmp 采集器。
- 参数说明：

请求参数	参数说明
RCollectSnmpMonitorEntityTO collectSnmpMonitorEntity	snmp 监控实体

RCollectSnmpMonitorEntityTO	说明
int id	主键，表 ID（系统会自动生成，从 1 开始累积生成）
int collectId	采集器 id
String snmpStatus	snmp 采集状态(启动，停止)
String snmpProtocol	snmp 采集协议
long createTime	创建时间
long modifyTime	修改时间
String remark	描述
String mac	采集器 mac 地址
int port	端口
int userId	用户 id

- 返回值：int

Integer	说明
int	0 失败/1 成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```java
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol, RCollectMonitorService.class.getName());

    clientMonitor = new RCollectMonitorService.Client(serviceProtocol);
}

@Test
public void updateSnmpMonitorService() throws TException {
    RCollectSnmpMonitorEntityTO rCollectSnmpMonitorEntityTO = new RCollectSnmpMonitorEntityTO();
    rCollectSnmpMonitorEntityTO.setId(1);
    rCollectSnmpMonitorEntityTO.setCollectId(2);
    rCollectSnmpMonitorEntityTO.setSnmpProtocol("UDP");
    rCollectSnmpMonitorEntityTO.setSnmpStatus("停止");
    int num = clientMonitor.updateSnmpCollectMonitor(rCollectSnmpMonitorEntityTO);
    System.out.println(num);
}
```

8.5 查询 snmp 监控信息（getSnmpMonitor）

- 方法定义：RCollectSnmpMonitorEntityTO getSnmpMonitor (int collectId) throws R Exception, RReturnNullException , TException;
- 方法说明：修改 snmp 监控信息，用于启动和停止 snmp 采集器。
- 参数说明：

请求参数	参数说明
int collectId	采集器 id

- 返回值：RCollectSnmpMonitorEntityTO

• RCollectSnmpMonitorEntityTO	说明
RCollectSnmpMonitorEntityTO collectMonitorEntity	Snmp 监控实体

RCollectSnmpMonitorEntityTO	说明
int id	主键，表 ID（系统会自动生成，从 1

	开始累积生成）
int collectId	采集器 id
String snmpStatus	snmp 采集状态(启动，停止)
String snmpProtocol	snmp 采集协议
long createTime	创建时间
long modifyTime	修改时间
String remark	描述
String mac	采集器 mac 地址
int port	端口
int userId	用户 id

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("127.0.0.1", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new TMultiplexedProtocol(protocol, RCollectMonitorService.class.getName());

    clientMonitor = new RCollectMonitorService.Client(serviceProtocol);
}

@Test
public void getSnmpMonitorService() throws TException {
    RCollectSnmpMonitorEntityTO rCollectSnmpMonitorEntityTO = clientMonitor.getSnmpMonitor(2);
    System.out.println(rCollectSnmpMonitorEntityTO);
}
```

9 数据类型接口说明（service RDataTypeService）

9.1 方法说明概要

方法名称	说明
保存分隔符切分的数据类型和字段（insertSeparatorDataType）	基于分隔符进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行保存。
保存正则表达式数据类型和字段 （insertRegexDataType）	基于正则表达式进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行保存。
修改分隔符切分的数据类型和字段（updateSeparatorDataType）	基于分隔符方式进行数据解析的数据类型和该数据类型字段进行修改。
修改正则表达式数据类型和字段（updateRegexDataType）	基于正则表达式进行数据解析，将解析后的数据类型和该数据类型下的所有字段进行修改。
通过用户 id 查询所有数据类型（getDataTypeByUserId）	通过用户 id 查询所有数据类型
查询分隔符切分数据类型和字段（getDataTypeByUserId）	基于分隔符进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行查询。
查询正则表达式数据类型和字段（getRegexDataType）	基于正则表达式进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行查询。
验证数据类型名称是否重复（checkName）	验证数据类型名称是否重复。

获取全部数据类型及字段（getAllDataType）	获取全部数据类型及字段。
按数据类型 id 删除（delById）	按数据类型 id 删除。

9.2 保存分隔符切分的数据类型和字段（insertSeparatorDataType）

- 方法定义：insertSeparatorDataType(1:DataTypeSeparatorModel.RDataTypeSeparatorModelTO dataTypeSeparatorModel) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：基于分隔符进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行保存。
- 参数说明：

请求参数	参数说明
RDataTypeSeparatorModelTO	分隔符切分配置实体

RDataTypeSeparatorModelTO	说明
string typeName	数据类型名称
string demoData	样例数据
short status	状态
int userId	用户 id
short parserType	解析类型 1/分隔符、2/正则表达式
string separator	切分解析操作链
list <RDataFieldEntityTO>	数据字段集合

RDataFieldEntityTO	说明

int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对
boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否
string connector	多 value 连接符
string separator	单 value 拆分符

- 返回值：int

Integer	说明
int	0/失败、大于 0/成功、小于 0/字段不合法

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void insertSeparatorDataType() throws TException {

    List<RDataFieldEntityTO> dataFields = new ArrayList();
    RDataFieldEntityTO rDataFieldEntityTO = new RDataFieldEntityTO();
    rDataFieldEntityTO.setFieldName("field_0");
    rDataFieldEntityTO.setFieldType((short) 0);

    RDataTypeSeparatorModelTO rDataTypeSeparatorModelTO = new RDataTypeSeparatorModelTO();
    rDataTypeSeparatorModelTO.setUserId(1);
    rDataTypeSeparatorModelTO.setStatus((short) 1);
    rDataTypeSeparatorModelTO.setParserType((short) 1);
    rDataTypeSeparatorModelTO.setTypeName("dataTypeDemo");
    rDataTypeSeparatorModelTO.setDemoData("aaa\nbbb\nccc");
    rDataTypeSeparatorModelTO.setSeparator(
            "[{\"optFields\":null,\"symbol\":null,\"parserType\":" +
            "\"relation\",\"relatFields\":{\"field_0\":\"field_0\"}}]");
    rDataTypeSeparatorModelTO.setDataField(dataFields);

    int successId = client.insertSeparatorDataType(rDataTypeSeparatorModelTO);
```

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

9.3 保存正则表达式数据类型和字段 （insertRegexDataType）

- 方法定义：insertRegexDataType(1:DataTypeRegexModel.RDataTypeRegexModelTO dataTypeRegexModel) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：基于正则表达式进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行保存。
- 参数说明：

请求参数	参数说明
RDataTypeRegexModelTO	正则表达式数据类型配置实体

RDataTypeRegexModelTO	说明
int id	字段 id
long createrTime	创建时间
string typeName	数据类型名称
string demoData	样例数据
short status	是否有效
int userId	用户 id
short parserType	解析类型
string regexs	正则解析字符串

list <RDataFieldEntityTO>	数据字段集合

RDataFieldEntityTO	说明
int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对
boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否
string connector	多 value 连接符
string separator	单 value 拆分符

- 返回值：int

Integer	说明
int	0/失败、大于 0/成功、小于 0/字段不合法

返回异常	异常说明
RException	
RReturnNullException	返回异常

TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void insertRegexDataType() throws TException {

    List<RDataFieldEntityTO> dataFields = new ArrayList();
    RDataFieldEntityTO rDataFieldEntityTO = new RDataFieldEntityTO();
    rDataFieldEntityTO.setFieldName("field_0");
    rDataFieldEntityTO.setFieldType((short) 0);

    RDataTypeRegexModelTO rDataTypeRegexModelTO =
            new RDataTypeRegexModelTO();

    rDataTypeRegexModelTO.setUserId(1);
    rDataTypeRegexModelTO.setStatus((short) 1);
    rDataTypeRegexModelTO.setParserType((short) 2);
    rDataTypeRegexModelTO.setTypeName("dataTypeDemo");
    rDataTypeRegexModelTO.setDemoData("aaa\nbbb\nccc");
    rDataTypeRegexModelTO.setRegexs("^(?<clientip>\\S+)");
    rDataTypeRegexModelTO.setDataField(dataFields);
```

```
    int successId = client.insertRegexDataType(rDataTypeRegexModelTO);
    System.out.println(successId);
}
```

9.4 修改分隔符切分的数据类型和字段（updateSeparatorDataType）

- 方法定义：updateSeparatorDataType(1:DataTypeSeparatorModel.RDataTypeSeparatorModelTO dataTypeSeparatorModel) throws (1:Exception.RException e,2:Exception.RReturnNullException
- 方法说明：基于分隔符方式进行数据解析的数据类型和该数据类型字段进行修改。
- 参数说明：

请求参数	参数说明
RDataTypeSeparatorModelTO	分隔符切分配置实体

RDataTypeSeparatorModelTO	说明
string typeName	数据类型名称
string demoData	样例数据
short status	状态
int userId	用户 id
short parserType	解析类型 1/分隔符、2/正则表达式
string separator	切分解析操作链
list <RDataFieldEntityTO>	数据字段集合

RDataFieldEntityTO	说明
int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对
boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否
string connector	多 value 连接符
string separator	单 value 拆分符

- 返回值：int

Integer	说明
int	0/失败、大于 0/成功、小于 0/字段不合法

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void updateSeparatorDataType() throws TException {

    List<RDataFieldEntityTO> dataFields = new ArrayList();
    RDataFieldEntityTO rDataFieldEntityTO = new RDataFieldEntityTO();
    rDataFieldEntityTO.setFieldName("field_0");
    rDataFieldEntityTO.setFieldType((short) 0);

    RDataTypeSeparatorModelTO rDataTypeSeparatorModelTO = new RDataTypeSeparatorModelTO();
    rDataTypeSeparatorModelTO.setUserId(1);
    rDataTypeSeparatorModelTO.setStatus((short) 1);
    rDataTypeSeparatorModelTO.setParserType((short) 1);
    rDataTypeSeparatorModelTO.setTypeName("dataTypeDemo");
    rDataTypeSeparatorModelTO.setDemoData("aaa\nbbb\nccc");
    rDataTypeSeparatorModelTO.setSeparator(
            "[{\"optFields\":null,\"symbol\":null,\"parserType\":" +
                    "\"relation\",\"relatFields\":{\"field_0\":\"field_0\"}}]");
    rDataTypeSeparatorModelTO.setDataField(dataFields);
    rDataTypeSeparatorModelTO.setId(1);

    int successId = client.updateSeparatorDataType(rDataTypeSeparatorModelTO);
```

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

9.5 修改正则表达式数据类型和字段（updateRegexDataType）

- 方法定义：updateRegexDataType(1:DataTypeRegexModel.RDataTypeRegexModelTO dataTypeRegexModel) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：基于正则表达式进行数据解析，将解析后的数据类型和该数据类型下的所有字段进行修改。
- 参数说明：

请求参数	参数说明
RDataTypeRegexModelTO	正则表达式数据类型配置实体

RDataTypeRegexModelTO	说明
int id	字段 id
long createrTime	创建时间
string typeName	数据类型名称
string demoData	样例数据
short status	是否有效
int userId	用户 id
short parserType	解析类型
string regexs	正则解析字符串
list <RDataFieldEntityTO>	数据字段集合

RDataFieldEntityTO	说明
int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对
boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否
string connector	多 value 连接符
string separator	单 value 拆分符

- 返回值：int

Integer	说明
int	0/失败、大于 0/成功、小于 0/字段不合法

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void updateRegexDataType() throws TException {

    List<RDataFieldEntityTO> dataFields = new ArrayList();
    RDataFieldEntityTO rDataFieldEntityTO = new RDataFieldEntityTO();
    rDataFieldEntityTO.setFieldName("clientip");
    rDataFieldEntityTO.setFieldType((short) 0);

    RDataTypeRegexModelTO rDataTypeRegexModelTO =
            new RDataTypeRegexModelTO();

    rDataTypeRegexModelTO.setUserId(1);
    rDataTypeRegexModelTO.setStatus((short) 1);
    rDataTypeRegexModelTO.setParserType((short) 2);
    rDataTypeRegexModelTO.setTypeName("dataTypeDemo");
    rDataTypeRegexModelTO.setDemoData("aaa\nbbb\nccc");
    rDataTypeRegexModelTO.setRegexs("^(?<clientip>\\S+)");
    rDataTypeRegexModelTO.setDataField(dataFields);
    rDataTypeRegexModelTO.setId(2);
```

```
    int successId = client.updateRegexDataType(rDataTypeRegexModelTO);
    System.out.println(successId);
}
```

9.6 通过用户 id 查询所有数据类型（getDataTypeByUserId）

- 方法定义：getDataTypeByUserId(1:int userId) throws (1:Exception.RException e, 2:Exception.RReturnNullException rne);
- 方法说明：通过用户 id 查询所有数据类型
- 参数说明：

请求参数	参数说明
int userId	用户 id

- 返回值：list<RDataTypeEntityTO>

RDataFieldEntityTO	说明
int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对
boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否

string connector	多 value 连接符
string separator	单 value 拆分符

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void getDataTypeByUserId() throws TException {

    System.out.println(client.getDataTypeByUserId(2));
}
```

9.7 查询分隔符切分数据类型和字段（getDataTypeByUserId）

- 方法定义：getSeparatorDataType(1:int dataTypeId) throws (1:Exception.RExcepti on e,2:Exception.RReturnNullException rne);
- 方法说明：基于分隔符进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行查询。
- 参数说明：

请求参数	参数说明
int dataTypeId	数据类型 id

- 返回值：RDataTypeSeparatorModelTO

RDataTypeSeparatorModelTO	说明
string typeName	数据类型名称
string demoData	样例数据
short status	状态
int userId	用户 id
short parserType	解析类型 1/分隔符、2/正则表达式
string separator	切分解析操作链
list <RDataFieldEntityTO>	数据字段集合

返回异常	异常说明
RException	
RReturnNullException	返回异常

TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void getSeparatorDataType() throws TException {

    System.out.println(client.getSeparatorDataType(3));
}
```

9.8 查询正则表达式数据类型和字段（getRegexDataType）

- 方法定义：getRegexDataType(1:int dataTypeId) throws (1:Exception.RException e, 2:Exception.RReturnNullException rne);
- 方法说明：基于正则表达式进行数据解析，并将解析后的数据类型和该数据类型下的所有字段进行查询。
- 参数说明：

请求参数	参数说明
int dataTypeId	数据类型 id

- 返回值：RDataTypeRegexModelTO

RDataTypeRegexModelTO	说明
int id	字段 id
long createrTime	创建时间
string typeName	数据类型名称
string demoData	样例数据
short status	是否有效
int userId	用户 id
short parserType	解析类型
string regexs	正则解析字符串
list <RDataFieldEntityTO>	数据字段集合

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
```

```
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void getSeparatorDataType() throws TException {

    System.out.println(client.getSeparatorDataType(4));
}
```

9.9 验证数据类型名称是否重复（checkName）

- 方法定义：checkName(1:int userId,2:string typeName) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：验证数据类型名称是否重复。
- 参数说明：

请求参数	参数说明
int userId	用户 id
string typeName	数据类型名称

- 返回值：int

Integer	说明
int	0/不存在数据类型名、1/存在数据类型名

返回异常	异常说明

RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void checkName() throws TException {

    System.out.println(client.checkName(1,"dataTypeDemo"));
}
```

9.10 获取全部数据类型及字段（getAllDataType）

- 方法定义：getAllDataType(1:int userId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne)
- 方法说明：获取全部数据类型及字段。

- 参数说明：

请求参数	参数说明
int userId	用户 id

- 返回值：list<RDataTypeRegexModelTO>

RDataTypeRegexModelTO	说明
int id	字段 id
long createrTime	创建时间
string typeName	数据类型名称
string demoData	样例数据
short status	是否有效
int userId	用户 id
short parserType	解析类型
string regexs	正则解析字符串
list <RDataFieldEntityTO>	数据字段集合

RDataFieldEntityTO	说明
int id	主键字段 id
int dataTypeId	数据类型 id
string fieldName	字段名称
short fieldType	字段类型 0/string、1/数字、2/日期、3/ip、4/键值对

boolean isBusinessTime	是否是业务时间 1/是、0/否
string dateFormatter	日期格式
string timeZone	时区
boolean isMap	是否是键值对 1/是、0/否
string connector	多 value 连接符
string separator	单 value 拆分符

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void getAllDataType() throws TException {
```

```
    System.out.println(client.getAllDataType(1));
}
```

9.11 按数据类型 id 删除（delById）

- 方法定义：delById(1:int id) throws (1:Exception.RException e,2:Exception.RReturnNullException rne)
- 方法说明：按数据类型 id 删除。
- 参数说明：

请求参数	参数说明
int id	数据类型 id

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
```

```
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RDataTypeService.class.getName());

    client = new RDataTypeService.Client(serviceProtocol);
}

@Test
public void delById() throws TException {

    System.out.println(client.delById(4));
}
```

10 文本采集接口说明（service RTextCollectService）

10.1 方法说明概要

方法名称	说明
添加文本采集配置（insertCollectConf）	添加文本采集方式的配置。
通过采集器 id 查询文本采集配置（selectCollectConfByCollectId）	通过采集器 id 查询文本采集配置。
通过用户 id 查询文本采集配置（selectCollectConfByUserId）	通过用户 id 查询文本采集配置。
删除文本采集配置（delCollectConf）	删除文本采集配置。
修改文本采集配置（updateCollectConf）	修改文本采集配置。
通过采集路径查询采集器（getByPath）	通过采集路径查询采集器。

10.2 添加文本采集配置（insertCollectConf）

- 方法定义：insertCollectConf(1:TextCollectConfEntity.RTextCollectConfEntityTO collectConfEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);

- 方法说明：添加文本采集方式的配置。
- 参数说明：

请求参数	参数说明
RTextCollectConfEntityTO	文本采集实体

RTextCollectConfEntityTO	说明
Int id	字段 id
Int userid	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string collectPath	采集路径
short status	是否已删除 0/已删除、1/没删除
long status	创建时间
Long modifyTime	修改时间
string timeZone	时区
string dateFormatter	日期格式
string remark	备注
string encoding	编码
string mac	mac 地址

- 返回值：int

Integer	说明

int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}


@Test
public void insertCollectConf() throws TException {

    RTextCollectConfEntityTO rTextCollectConfEntityTO = new RTextCollectConfEntityTO();
    rTextCollectConfEntityTO.setUserId(1);
    rTextCollectConfEntityTO.setCollectId(1);
    rTextCollectConfEntityTO.setDataTypeId(1);
    rTextCollectConfEntityTO.setCollectPath("F:\\data.txt");
    rTextCollectConfEntityTO.setStatus((short) 1);
    rTextCollectConfEntityTO.setEncoding("UTF-8");
```

```
    System.out.println(client.insertCollectConf(rTextCollectConfEntityTO));
}
```

10.3 通过采集器 id 查询文本采集配置（selectCollectConfByCollectId）

- 方法定义：list<TextCollectConfEntity.RTextCollectConfModelTO> selectCollectConfByCollectId(1:i32 collectId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过采集器 id 查询文本采集配置。
- 参数说明：

请求参数	参数说明
int collectId	采集器 id

- 返回值：list<RTextCollectConfModelTO>

RTextCollectConfModelTO	说明
int id	字段 id
int userId	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string dataTypeName	采集路径
string collectPath	备注
string remark	编码

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}
@Test
public void selectCollectConfByCollectId() throws TException {

    System.out.println(client.selectCollectConfByCollectId(1));
}
```

10.4 通过用户 id 查询文本采集配置（selectCollectConfByUserId）

- 方法定义：list<TextCollectConfEntity.RTextCollectConfModelTO> selectCollectConfByUserId(1:i32 userId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过用户 id 查询文本采集配置。

- 参数说明：

请求参数	参数说明
int userId	用户 id

- 返回值：list<RTextCollectConfModelTO>

RTextCollectConfModelTO	说明
int id	字段 id
int userId	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string dataTypeName	采集路径
string collectPath	备注
string remark	编码

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
```

```
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}
@Test
public void selectCollectConfByUserId() throws TException {

    System.out.println(client.selectCollectConfByUserId(1));
}
```

10.5 删除文本采集配置（delCollectConf）

- 方法定义：int delCollectConf(1:i32 id,2:string mac) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：删除文本采集配置。
- 参数说明：

请求参数	参数说明
int id	文本采集 id
string mac	采集器 mac

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	

RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}
@Test
public void delCollectConf() throws TException {

    System.out.println(client.delCollectConf(18,"005056C00001"));
}
```

10.6 修改文本采集配置（updateCollectConf）

- 方法定义：int updateCollectConf(1:TextCollectConfEntity.RTextCollectConfEntityTO collectConfEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：修改文本采集配置。
- 参数说明：

请求参数	参数说明

RTextCollectConfEntityTO	文本采集实体

RTextCollectConfEntityTO	说明
Int id	字段 id
Int userid	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string collectPath	采集路径
short status	是否已删除 0/已删除、1/没删除
long status	创建时间
Long modifyTime	修改时间
string timeZone	时区
string dateFormatter	日期格式
string remark	备注
string encoding	编码
string mac	mac 地址

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明

RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}
@Test
public void updateCollectConf() throws TException {

    RTextCollectConfEntityTO rTextCollectConfEntityTO =
            new RTextCollectConfEntityTO();
    rTextCollectConfEntityTO.setId(16);
    rTextCollectConfEntityTO.setUserId(1);
    rTextCollectConfEntityTO.setCollectId(1);
    rTextCollectConfEntityTO.setDataTypeId(1);
    rTextCollectConfEntityTO.setCollectPath("F:\\data.txt");
    rTextCollectConfEntityTO.setStatus((short) 1);
    rTextCollectConfEntityTO.setEncoding("UTF-8");

    System.out.println(client.updateCollectConf(rTextCollectConfEntityTO));
}
```

10.7 通过采集路径查询采集器（getByPath）

- 方法定义：TextCollectConfEntity.RTextCollectConfModelTO getByPath(1:TextCollectConfEntity.RTextCollectConfModelTO collectConfEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过采集路径查询采集器。
- 参数说明：

请求参数	参数说明
RTextCollectConfModelTO	文本采集 VO 实体

RTextCollectConfModelTO	说明
Int id	字段 id
Int userid	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string dataTypeName	数据类型名称
string collectPath	采集路径
string remark	备注
string encoding	编码

- 返回值：RTextCollectConfModelTO

RTextCollectConfModelTO	说明
Int id	字段 id

Int userid	用户 id
Int collectId	采集器 id
Int dataTypeId	数据类型 id
string dataTypeName	数据类型名称
string collectPath	采集路径
string remark	备注
string encoding	编码

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RTextCollectService.class.getName());

    client = new RTextCollectService.Client(serviceProtocol);
}
@Test
public void getByPath() throws TException {
```

```
    RTextCollectConfModelTO rTextCollectConfModelTO =
            new RTextCollectConfModelTO();
    rTextCollectConfModelTO.setCollectPath("F:\\data.txt");
    rTextCollectConfModelTO.setCollectId(1);
    System.out.println(client.getByPath(rTextCollectConfModelTO));
}
```

11 文件上传接口说明（service RFileUploadService）

11.1 方法说明概要

方法名称	说明
添加上传文件信息（insertFileInfo）	添加上传文件信息。
通过 userId 查询上传文件信息（selectFileInfoByUserId）	通过 userid 查询上传文件信息。
通过 dataTypeId 查询上传文件信息（selectFileInfoByDataTypeId）	通过 datatypeid 查询上传文件信息。
删除上传文件（delUploadFile）	通过 datatypeid 查询上传文件信息。

11.2 添加上传文件信息（insertFileInfo）

- 方法定义：int insertFileInfo(1:FileInfoEntity.RFileInfoEntityTO fileInfoEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：添加上传文件信息。
- 参数说明：

请求参数	参数说明
RFileInfoEntityTO	文件信息实体

RFileInfoEntityTO	说明
Int id	字段 id
Int userid	用户 id
int dataTypeId	数据类型 id
string filePath	文件路径
int size	文件大小
int rows	文件行数
short status	是否已删除
string typeName	数据类型名称

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
```

```
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RFileUploadService.class.getName());

    client = new RFileUploadService.Client(serviceProtocol);
}
@Test
public void insertCollectConf() throws TException {

    RFileInfoEntityTO rFileInfoEntityTO = new RFileInfoEntityTO();
    rFileInfoEntityTO.setUserId(1);
    rFileInfoEntityTO.setDataTypeId(1);
    rFileInfoEntityTO.setFilePath(
            "/Application/nirvana/apache-tomcat-8.0.30/" +
            "webapps/de/fileupload/1/" +
            "4e53b42a-e160-45c8-93b8-ae9d6b5f9e01/nginx-demolog-1447309072.log");
    rFileInfoEntityTO.setRows(2555);
    rFileInfoEntityTO.setSize(359000);
    rFileInfoEntityTO.setStatus((short) 1);

    System.out.println(client.insertFileInfo(rFileInfoEntityTO));
}
```

11.3 通过 userId 查询上传文件信息（selectFileInfoByUserId）

- 方法定义：list<FileInfoEntity.RFileInfoEntityTO> selectFileInfoByUserId(1:i32 userId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过 userid 查询上传文件信息。
- 参数说明：

请求参数	参数说明

Int userid	用户 id

- 返回值：list<RFileInfoEntityTO>

RFileInfoEntityTO	说明
RFileInfoEntityTO	文件信息实体

RFileInfoEntityTO	说明
Int id	字段 id
Int userid	用户 id
int dataTypeId	数据类型 id
string filePath	文件路径
int size	文件大小
int rows	文件行数
short status	是否已删除
string typeName	数据类型名称

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
```

```
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RFileUploadService.class.getName());

    client = new RFileUploadService.Client(serviceProtocol);
}
@Test
public void selectFileInfoByUserId() throws TException {

    System.out.println(client.selectFileInfoByUserId(1));
}
```

11.4 通过 dataTypeId 查询上传文件信息（selectFileInfoByDataTypeId）

- 方法定义：list<FileInfoEntity.RFileInfoEntityTO> selectFileInfoByDataTypeId(1:i32 dataTypeId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过 datatypeid 查询上传文件信息。
- 参数说明：

请求参数	参数说明
int dataTypeId	数据类型 id

- 返回值：list<RFileInfoEntityTO>

RFileInfoEntityTO	说明
RFileInfoEntityTO	文件信息实体

RFileInfoEntityTO	说明
Int id	字段 id
Int userid	用户 id
int dataTypeId	数据类型 id
string filePath	文件路径
int size	文件大小
int rows	文件行数
short status	是否已删除
string typeName	数据类型名称

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RFileUploadService.class.getName());

    client = new RFileUploadService.Client(serviceProtocol);
}
```

```
@Test
public void selectFileInfoByDataTypeId() throws TException {

    System.out.println(client.selectFileInfoByDataTypeId(1));
}
```

11.5 删除上传文件（delUploadFile）

- 方法定义：int delUploadFile(1:int id) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过 datatypeid 查询上传文件信息。
- 参数说明：

请求参数	参数说明
int id	文件 id

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
```

```
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RFileUploadService.class.getName());

    client = new RFileUploadService.Client(serviceProtocol);
}
@Test
public void delUploadFile() throws TException {

    System.out.println(client.delUploadFile(17));
}
```

12 syslog 采集配置说明（service RSyslogCollectService）

12.1 方法说明概要

方法名称	说明
添加 syslog 采集配置参数（insertSyslogConf）	添加 syslog 采集配置参数。
通过采集器 id 查询 syslog 采集配置（selectSyslogConfByCollectId）	通过采集器 id 查询 syslog 采集配置。
删除 syslog 采集配置（delSyslogConf）	删除 syslog 采集配置。
修改 syslog 采集配置（updateSyslogConf）	修改 syslog 采集配置。

12.2 添加 syslog 采集配置参数（insertSyslogConf）

- 方法定义：int insertSyslogConf(1:SyslogConfEntity.RSyslogConfEntityTO syslogConfEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：添加 syslog 采集配置参数。

- 参数说明：

请求参数	参数说明
RSyslogConfEntityTO	syslog 采集实体

RSyslogConfEntityTO	说明
Int id	字段 id
Int userid	用户 id
int collectId	采集器 id
int dataTypeId	数据类型 id
string ipAddress	ip 地址
short status	是否已删除
long creatTime	创建时间
long modifyTime	修改时间
string timeZone	时区
string dateFormatter	日期格式
string remark	备注
int brandDeviceId	厂商设备类型 id
string encoding	编码

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RSyslogCollectService.class.getName());

    client = new RSyslogCollectService.Client(serviceProtocol);
}
@Test
public void insertSyslogConf() throws TException {

    RSyslogConfEntityTO rSyslogConfEntityTO = new RSyslogConfEntityTO();
    rSyslogConfEntityTO.setUserId(1);
    rSyslogConfEntityTO.setCollectId(1);
    rSyslogConfEntityTO.setDataTypeId(1);
    rSyslogConfEntityTO.setIpAddress("10.0.1.139");
    rSyslogConfEntityTO.setStatus((short) 1);
    rSyslogConfEntityTO.setBrandDeviceId(16);
    rSyslogConfEntityTO.setEncoding("UTF-8");

    System.out.println(client.insertSyslogConf(rSyslogConfEntityTO));
}
```

12.3 通过采集器 id 查询 syslog 采集配置（selectSyslogConfByCollectId）

- 方法定义：list<SyslogConfEntity.RSyslogConfModelTO> selectSyslogConfByCollectId(1:i32 collectId) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：通过采集器 id 查询 syslog 采集配置。
- 参数说明：

请求参数	参数说明
int collectId	采集器 id

- 返回值：list<RSyslogConfModelTO>

RSyslogConfModelTO（syslog 采集 vo 实体）	说明
Int id	字段 id
Int userid	用户 id
int collectId	采集器 id
int dataTypeId	数据类型 id
string dataTypeName	数据类型名称
string ipAddress	ip 地址
string remark	备注
string brandName	设备厂商名
string deviceName	设备类型名
int brandId	设备厂商 id

int deviceId	设备类型 id
string encoding	编码

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RSyslogCollectService.class.getName());

    client = new RSyslogCollectService.Client(serviceProtocol);
}
@Test
public void selectSyslogConfByCollectId() throws TException {

    System.out.println(client.selectSyslogConfByCollectId(1));
}
```

12.4 删除 syslog 采集配置（delSyslogConf）

- 方法定义：int delSyslogConf(1:i32 id) throws (1:Exception.RException e,2:Exce

ption.RReturnNullException rne);

- 方法说明：删除 syslog 采集配置。
- 参数说明：

请求参数	参数说明
int id	syslog 采集配置 id

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RSyslogCollectService.class.getName());

    client = new RSyslogCollectService.Client(serviceProtocol);
}
```

```
@Test
public void delSyslogConf() throws TException {

    System.out.println(client.delSyslogConf(17));
}
```

12.5 修改 syslog 采集配置（updateSyslogConf）

- 方法定义：int updateSyslogConf(1:SyslogConfEntity.RSyslogConfEntityTO syslogConfEntity) throws (1:Exception.RException e,2:Exception.RReturnNullException rne);
- 方法说明：修改 syslog 采集配置。
- 参数说明：

请求参数	参数说明
RSyslogConfEntityTO	syslog 采集实体

RSyslogConfEntityTO	说明
Int id	字段 id
Int userid	用户 id
int collectId	采集器 id
int dataTypeId	数据类型 id
string ipAddress	ip 地址
short status	是否已删除
long creatTime	创建时间

long modifyTime	修改时间
string timeZone	时区
string dateFormatter	日期格式
string remark	备注
int brandDeviceId	厂商设备类型 id
string encoding	编码

- 返回值：int

Integer	说明
int	0/失败、1/成功

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.69", 6060, 60 * 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    TMultiplexedProtocol serviceProtocol =
            new TMultiplexedProtocol(protocol, RSyslogCollectService.class.getName());
```

```
    client = new RSyslogCollectService.Client(serviceProtocol);
}
@Test
public void updateSyslogConf() throws TException {

    RSyslogConfEntityTO rSyslogConfEntityTO = new RSyslogConfEntityTO();
    rSyslogConfEntityTO.setId(17);
    rSyslogConfEntityTO.setUserId(1);
    rSyslogConfEntityTO.setCollectId(1);
    rSyslogConfEntityTO.setDataTypeId(1);
    rSyslogConfEntityTO.setIpAddress("10.0.1.138");
    rSyslogConfEntityTO.setBrandDeviceId(16);
    rSyslogConfEntityTO.setEncoding("UTF-8");

    System.out.println(client.updateSyslogConf(rSyslogConfEntityTO));
}
```

13 原语搜索接口说明

13.1 方法说明概要

方法名称	说明
基础检索返回表格类型数据（TResultModel query）	基于检索返回表格类型的数据。
基础检索返回 CSV 格式数据（TQueryModel model）	基于基础检索返回 CSV 格式数据。
报表检索表格返回表格格式（queryReport）	基于报表搜索命令进行查询。
报表检索结果 CSV 格式（queryReport2CSV）	基于报表搜索命令进行查询并以 CSV 格式返回。
导出文档（scanData）	导出文档

根据查询删除文档（queryByDelete）	根据查询删除文档
根据查询修改文档（queryByUpdate）	根据查询修改文档

13.2 基础检索返回表格类型数据（TResultModel query）

- 方法定义：TResultModel query(TQueryModel queryModel) throws RException
- 方法说明：基于检索返回表格类型的数据。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象

- 对象说明：TQueryModel queryModel

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

- 返回值：TResultModel（检索结果实体对象）
- 对象说明：

属性	说明

List <TRecordModel> hits	记录集
String tooktime	检索耗时
Int pageSize	每页记录数量
Int pageNum	当前页数
Long count	匹配记录总数

TRecordModel（记录集）	说明
String type	业务类型标识(A_source 字段值)
Map<String,String> auxiliaryFields	辅助字段集，以 A_为前缀
Map<String,String> fields	业务字段集
Int pageNum	当前页数
Long count	匹配记录总数

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}

@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60 *
1000);
```

```
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}

@Test
public void testQuery() throws TException {
    TQueryModel qm = new TQueryModel();
    qm.setUserId(DEStringUtils.encodeUserId("13"));
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0, 0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 20, 19, 8, 46).getMillis());
    qm.setQueryString("*:*");
    qm.setPageSize(1);
    TResultModel returnV = searchData.query(qm);
    System.out.println(JSON.toJSONString(returnV));
}
```

13.3 基础检索返回 CSV 格式数据（TQueryModel model）

- 方法定义：TCSVResult query2CSV(TQueryModel queryModel, CSVParameter csvParameter) throws RException
- 方法说明：基于基础检索返回 CSV 格式数据。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象

CSVParameter	CSV 参数实体

- 对象说明

TQueryModel model（检索过滤对象）	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

CSVParameter(CSV 参数实体对象）	说明
Boolean flat	是否打平，打平时，嵌套字段会以.联合成一个字段，默认值为 TRUE
String separator	字段分隔符

- 返回值：TCSVResult
- 对象说明

TCSVResult（CSV 结果实体对象）	说明
List<String>headers	列名
List<String>lines	列内容

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}

@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60
* 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}
```

```
@Test
public void testQuery2CSV() throws TException {
    TQueryModel qm = new TQueryModel();
    qm.setUserId("xxxxxxxxxxxxxxx");
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0,
0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 20, 19, 8,
46).getMillis());
    qm.setQueryString("*:*");
    qm.setPageSize(1);
    CSVParameter csvParameter = new CSVParameter();
    TCSVResult returnV = searchData.query2CSV(qm,
csvParameter);
    System.out.println(JSON.toJSONString(returnV));
}
```

13.4 报表检索表格返回表格格式（queryReport）

- 方法定义：TResultTable queryReport(TQueryModel queryModel) throws RException
- 方法说明：基于报表搜索命令进行查询。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象

- 对象说明：TQueryModel model（检索过滤对象）

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0

string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

- 返回值：TResultTable
- 对象说明：TResultTable（表格结果实体）

属性	说明
List <String> header	表头
List <List <String>>	行记录集

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}
```

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60
* 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}

@Test
public void testQueryReport() throws TException {
    TQueryModel qm = new TQueryModel();
    qm.setUserId(DEStringUtils.encodeUserId("13"));
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0,
0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 20, 19, 8,
46).getMillis());
    qm.setQueryString("*:*|report count(A_source) on
A_source");
    TResultTable returnV = searchData.queryReport(qm);
    System.out.println(JSON.toJSONString(returnV));
}
```

13.5 报表检索结果 CSV 格式（queryReport2CSV）

- 方法定义：TCSVResult queryReport2CSV(TQueryModel querymodel, CSVParameter csvParameter) throws RException
- 方法说明：基于报表搜索命令进行查询并以 CSV 格式返回。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象
CSVParameter	CSV 参数实体

- 对象说明：TQueryModel model（检索过滤对象）

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

- 对象说明：CSVParameter（CSV 参数实体）

属性	参数说明
Boolean flat	是否打平，打平时，嵌套字段会以.联合成一个字段，默认值为 TRUE
String separator	字段分隔符

- 返回值：TCSVResult

TCSVResult（CSV 结果实体）	说明
List<String>headers	列名
List<String>lines	列内容

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}

@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60
* 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());
```

```
        searchData = new SearchData.Client(serviceProtocol);
    }

    @Test
    public void testQueryReportCsv() throws TException {
        TQueryModel qm = new TQueryModel();
        qm.setUserId(DEStringUtils.encodeUserId("13"));
        qm.setDatabase("aleiye");
        qm.setA_from(new DateTime(2016, 6, 14, 0, 0,
    0).getMillis());
        qm.setA_to(new DateTime(2016, 6, 20, 19, 8,
    46).getMillis());
        qm.setQueryString("*:*|report count(A_source) on
    A_source");
        CSVParameter csvParameter = new CSVParameter();
        TCSVResult returnV = searchData.queryReport2CSV(qm,
    csvParameter);
        System.out.println(JSON.toJSONString(returnV));
    }
```

13.6 导出文档（scanData）

- 方法定义：ScanResult scanData(TQueryModel queryModel, TScanModel scanModel)throws Rexception
- 方法说明：导出文档。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象
TScanModel scanModel	统计字段名称

- 对象说明：TQueryModel queryModel（检索过滤对象）

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，

	默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

- 对象说明：TScanModel scanModel（统计字段名称对象）

TScanModel	说明
List <String> scanfields	导出字段集，为空时表示导出所有字段
int scanSize	每次导出的最大文档数量，默认值 1000
String scrollID	滚动 ID

- 返回值：ScanResult
- 对象说明：ScanResult（导出文档结果实体）

属性	说明
List <Map<String,String>> lines	文档集
long totalCount	要导出的文档总数
long scanCount	本次导出文档数量
String scrollID	滚动 ID，用于下一批次导出时使用

- 异常

返回异常	异常说明
RException	

RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}

@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60
* 1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}


@Test
public void testScanData() throws Exception {
```

```
    TQueryModel qm = new TQueryModel();
    qm.setQueryString("*:*");
    qm.setUserId(DEStringUtils.encodeUserId("13"));
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0,
0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 17, 19, 8,
46).getMillis());
    TScanModel scanModel = new TScanModel();
    List<String> list = new ArrayList<>();
    list.add("clientip");
    list.add("bytes");
    scanModel.setScanfields(list);
    ScanResult result = searchData.scanData(qm, scanModel);
    String scollId = result.getScrollID();
    long totalCount = result.totalCount;
    long scanCount = 0L;
    while (true) {
        scanModel.setScrollID(scollId);
        result = searchData.scanData(qm, scanModel);
        scollId = result.getScrollID();
        scanCount += result.getScanCount();
        for (Map<String,String> lineMap : result.getLines())
{
            System.out.println(JSON.toJSONString(lineMap));
        }
        if (scanCount >= totalCount)
            break;
    }

}
```

13.7 根据查询删除文档（queryByDelete）

- 方法定义：DELResult queryByDelete(TQueryModel model) throws Rexception
- 方法说明：根据查询删除文档。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象

- 对象说明：TQueryModel queryModel（检索过滤对象）

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Int pageSize	每页记录数，默认值为 100
Int pageNum	查询的起始页(0 相当于第一页)，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户
Set<String> type	业务类型，为空时，表示不过滤业务类型
String database	库(索引前缀），为空时，表示不过滤库，默认值为 aliye。
Boolean isHighlighter	Boolean，默认值为 true
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE
Boolean timeIsAsc	返回数据是否是按时间正序排序，默认值为 FALSE

- 返回值：DELResult

- 对象说明：DELResult（删除结果实体）

属性	说明
long deleted	删除成功的记录数
long failed	删除失败的记录数
long found	匹配的记录数

long missing	漏掉的记录数

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}

@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60 *
1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}
```

```
@Test
public void testQueryByDel() throws Exception {
    TQueryModel qm = new TQueryModel();
    qm.setQueryString("*:*");
    qm.setUserId(DEStringUtils.encodeUserId("13"));
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0, 0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 25, 19, 8, 46).getMillis());
    DELResult delResult = searchData.queryByDelete(qm);
    System.out.printf(JSON.toJSONString(delResult));

}
```

13.8 根据查询修改文档（queryByUpdate）

- 方法定义：DELResult queryByDelete(TQueryModel model) throws Rexception
- 方法说明：根据查询修改文档。
- 参数说明：

请求参数	参数说明
TQueryModel queryModel	检索过滤对象

- 对象说明：TQueryModel（检索过滤对象）

属性	说明
Long a_from	起始时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
Long a_to	截止时间，a_from 和 a_to 都为 0 时，则表示所有时间段，默认值为 0
string queryString	检索语句，不能为空
String userId	用户标识，需加密，为空时，表示不过滤用户

String database	库(索引前缀)，为空时，表示不过滤库，默认值为 aliye。
List <String> indecies	索引列表
Boolean includeLower	是否包含起始时间，默认值为 FALSE
Boolean includeUpper	是否包含截止时间，默认值为 TRUE
Boolean isProcessTime	起止时间是否是入库时间，默认值为 FALSE

- 返回值：UpdateResult
- 对象说明：UpdateResult（更新结果实体）

属性	说明
long batches	更新批次数
long searchFailures	search 错误数
long tookTime	update 时间
long updated	更新记录数
long versionConflicts	版本冲突数
long indexingFailures	索引错误数

- 异常

返回异常	异常说明
RException	
RReturnNullException	返回异常
TException	Thrift 异常

- 示例

```
private TTransport transport;
private SearchData.Client searchData;

@After
public void after() throws Exception {
    if (transport != null)
        transport.close();
}
```

```
@Before
public void before() throws Exception {
    TTransport transport = new TSocket("10.0.1.136", 6060, 60 *
1000);
    transport.open();
    transport = new TFramedTransport(transport);
    // tt.open();
    // 协议要和服务端一致
    // TProtocol protocol = new TJSONProtocol(transport);
    TProtocol protocol = new TBinaryProtocol(transport);
    // TProtocol protocol = new TCompactProtocol(transport);
    TMultiplexedProtocol serviceProtocol = new
TMultiplexedProtocol(protocol, SearchData.class.getName());

    searchData = new SearchData.Client(serviceProtocol);
}


@Test
public void testQueryUpdate() throws TException {
    TQueryModel qm = new TQueryModel();
    qm.setQueryString("A_source: \"test123\"");
    qm.setUserId(DEStringUtils.encodeUserId("12"));
    qm.setDatabase("aleiye");
    qm.setA_from(new DateTime(2016, 6, 14, 0, 0, 0).getMillis());
    qm.setA_to(new DateTime(2016, 6, 17, 23, 0, 0).getMillis());
    TUpdateModel tUpdateModel = new TUpdateModel();
    List<ESStructTypeModel> listStruct = Lists.newArrayList();
    ESStructTypeModel ele = new ESStructTypeModel();
    ele.setFieldName("clientip");
    ele.setFieldType(ESFieldTypeEnum.STRING);
    ele.setData("127.0.0.1");
    ESStructTypeModel ele1 = new ESStructTypeModel();
```

```
    ele1.setFieldName("bytes1");
    ele1.setFieldType(ESFieldTypeEnum.LONG);
    ele1.setData("100");
    tUpdateModel.addToFieldTypeList(ele);
    tUpdateModel.addToFieldTypeList(ele1);
    UpdateResult updateResult = searchData.queryByUpdate(qm,
tUpdateModel);
    System.out.printf(JSON.toJSONString(updateResult));
}
```

14 SQL 检索

14.1 方法说明概要

方法名称	说明
检索	基于 sql 标准查询指定用户的历史数据
获取指定表的结构信息	查询某张表的结构信息

14.2 检索

- 方法定义：RSqlService query(String userId, String sql, long fromTime, long toTime) throws RException, RNoResourceException
- 方法说明:基于 sql 标准查询指定用户的历史数据
- 参数说明：

请求参数	参数说明
String userId	检索的数据所属的用户 Id,该 Id 必须为 aleiye 平台加密之后的 Id。
String sql	查询的 sql

Long fromTime	Sql 所要搜索的数据的起始业务时间，如果为 0，表示没有起始时间。格式为格林威治时间的毫秒值
Long toTime	Sql 所要搜索的数据的截止业务时间， 如果为 0，表示没有起始时间。格式为格林威治时间的毫秒值

- 返回值：RQueryResultModel model

返回结果	参数说明
RQueryResultModel model	检索返回的结果

RQueryResultModel model	说明
List<List<RDataCellModel>> dataList	返回的数据集合.(所有返回的值，为指定类型的 toString 值。其中 TIMESTAMP 及 DATE 均通过获取毫秒数取 String)
List<RStructTypeModel> fieldTypeList	返回数据的字段及其类型说明

RStructTypeModel	说明
String fieldName	字段名称
RFieldTypeEnum fieldType	字段类型
Boolean allowNull	字段是否允许为 null

RFieldTypeEnum	说明
STRING	字符串类型
SHORT	Short 型
INTEGER	INTEGER
LONG	LONG
FLOAT	FLOAT
DOUBLE	DOUBLE
BINARY	BINARY
BYTE	BYTE
BOOLEAN	BOOLEAN
TIMESTAMP	TIMESTAMP

DATE	DATE
DECIMAL	DECIMAL

RDataCellModel	说明
String data	字段值

- 异常

返回异常	异常说明
RException	返回异常
RNoResourceException	没有资源（如表不存在）
TException	Thrift 异常

- 示例

```
@Test
def testSQL(): Unit = {
  var transport: TTransport = new TSocket("10.0.1.136", 10003,
60 * 60 * 1000)
  transport.open
  transport = new TFramedTransport(transport)
  // 协议要和服务端一致
  val protocol: TProtocol = new TBinaryProtocol(transport)
  val serviceProtocol: TMultiplexedProtocol = new
TMultiplexedProtocol(protocol, classOf[RSqlService].getName)

  val client = new RSqlService.Client(serviceProtocol)
  val userId = "8B6B254FBE90B2A3"

  val result = client.query(userId, "select count(1) from test1
where A_timestamp between
strDateToLong('2015060314','yyyyMMddHH') and
strDateToLong('2016060314','yyyyMMddHH')", 0, 0)
  transport.close()
  val iterator = result.dataList.iterator()
```

```
    while (iterator.hasNext) {
      val a = iterator.next()
      val it1 = a.iterator()
      println()
      while (it1.hasNext) {
        val r1 = it1.next()
        print(",")
        print(r1.data)
      }
    }
  }
```

14.3 获取指定表的结构信息

- 方法定义：List< RStructTypeModel> getTableSchema (String userId, String tableName) throws RException, RNoResourceException
- 方法说明:查询某张表的结构信息
- 参数说明:

请求参数	参数说明
String userId	检索的数据所属的用户 Id,该 Id 必须为 aleiye 平台加密之后的 Id。
String tableName	需要查询的表名

- 返回值：RStructTypeModel model

RStructTypeModel	说明
String fieldName	字段名称
RFieldTypeEnum fieldType	字段类型
Boolean allowNull	字段是否允许为 null

RFieldTypeEnum	说明

STRING	字符串类型
SHORT	Short 型
INTEGER	INTEGER
LONG	LONG
FLOAT	FLOAT
DOUBLE	DOUBLE
BINARY	BINARY
BYTE	BYTE
BOOLEAN	BOOLEAN
TIMESTAMP	TIMESTAMP
DATE	DATE
DECIMAL	DECIMAL

- 示例

```
@Test
def testSchema(): Unit = {
  var transport: TTransport = new TSocket("10.0.1.136", 10003,
60 * 1000)
  transport.open
  transport = new TFramedTransport(transport)
  // 协议要和服务端一致
  val protocol: TProtocol = new TBinaryProtocol(transport)
  val serviceProtocol: TMultiplexedProtocol = new
TMultiplexedProtocol(protocol, classOf[RSqlService].getName)

  val client = new RSqlService.Client(serviceProtocol)
  val list = client.getTableSchema("8B6B254FBE90B2A3","test1")
  val t = list.iterator()
  while(t.hasNext){
    val t1 = t.next()
    println(t1.fieldName)
  }
```

```
    transport.close()
}
```

附录

1. 如何获取用户 API key

通过用户登录 Aleiye 产品后，在用户信息内查看用户鉴权码。

2. 如何获取应用 ID

通过用户名登录 Aleiye 产品后，在应用中心每个应用名称后，会有本应用的 ID 显示。