

# 数据传输服务 DTS

## SDK文档

## 产品文档



腾讯云

**【版权声明】**

©2013-2019 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

**【商标声明】**

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

**【服务声明】**

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

## 文档目录

SDK文档

数据订阅 SDK

# SDK文档

## 数据订阅 SDK

最近更新时间：2019-08-08 14:35:07

### 数据订阅 SDK 下载

单击下载 [数据订阅 SDK Version 2.8.0](#)。

### 发布日志

#### Version 2.8.0

1. 优化了内部鉴权逻辑。
2. 减少用户参数设置。

#### Version 2.7.2

1. 在 DDL 语句返回值中填充 db 值。

#### Version 2.7.0

1. 数据订阅通道后台的 HA 功能，秒级切换。
2. 添加 SDK 监控指标上报。

#### Version 2.6.0

1. 支持单 SDK 订阅多个通道。
2. 支持订阅 Client 的 stop、start 等操作。
3. 支持 DataMessage.Record 的序列化。
4. 优化 SDK 的性能，降低资源消耗。

#### Version 2.5.0

1. 修复高并发情况下小概率出现的 bug。
2. 支持事务中记录的全局唯一自增的 ID。

#### Version 2.4.0

1. 配合后台优化了订阅的逻辑，可以精确显示 SDK 当前的消费时间点。

- 修复了后台的少数特殊字符的编码问题。
- 修复了多项兼容性问题，建议更老版本用户尽快升级至此版本。

## 运行原理

### 拉取

SDK 的拉取和确认消息是两个异步的线程同时在做的，拉取是按顺序拉取的，两个线程的执行分别独立是严格有序，但这两个线程之间是异步的。

拉取到的消息会按顺序调用用户注册的 notify 函数，SDK 保证每一条消息会推送一次，且只有一次；如果没有调用 m.ackAsConsumed() 函数，消息还是会继续 notify，因为拉取和确认是异步的。

### 确认机制

SDK 采用的是增量确认机制，可以重复确认，但不可以漏确认任何一条消息，包括 BEGIN 和 COMMIT 消息。

- 例如客户端收到了1、2、3、4、5五条消息，但是客户端程序只对1、2、5这三条消息调用了 m.ackAsConsumed() 操作，那么 SDK 只会向服务器确认消费了1、2这两条消息，如果此时客户端程序出现故障，SDK 下次会从3这条消息开始获取。  
由于 SDK 消息拉取和确认是异步的，所以如果中间有消息没确认，SDK 仍然会去拉取新消息并 notify 给客户端，但是超过一定的长度（目前为8000），就不会拉取新的消息。
- 每一个消息记录都有唯一的 record\_id 和 checkpoint，SDK 其实是对消息的 checkpoint 进行确认。

## 运行环境要求

- Java 环境：JRE 1.6 及以上版本。
- SDK 需要运行在腾讯云的 CVM 主机上，与订阅实例在同地域的同一个 VPC 下（如果不在同一个 VPC，需要配置互通）。
- 如果要在公网上访问，可以用 CVM 做端口转发，不过这样**带宽和性能无法得到保证**，强依赖于外网带宽。

## 示例代码

使用腾讯云 Binlog 订阅示例代码如下：

```
import com.qcloud.dts.context.SubscribeContext;
import com.qcloud.dts.message.ClusterMessage;
import com.qcloud.dts.message.DataMessage;
import com.qcloud.dts.subscribe.ClusterListener;
```

```
import com.qcloud.dts.subscribe.DefaultSubscribeClient;
import com.qcloud.dts.subscribe.SubscribeClient;

import java.util.List;

public class Main {

    public static void main(String[] args) throws Exception {
        //创建一个context
        SubscribeContext context=new SubscribeContext();

        //用户secretId、secretKey
        context.setSecretId("AKID-522dabxxxxxxxxxxxxxxxxxxxx");
        context.setSecretKey("AKEY-0ff4cxxxxxxxxxxxxxxxxxxxx");

        //设置channel所在的region，2.8.0以后的SDK推荐设置region参数
        //region值参照：https://cloud.tencent.com/document/product/236/15833#.E5.9C.B0.E5.9F.9F.E5.88.9
        //7.E8.A1.A8
        context.setRegion("ap-beijing");
        //订阅的serviceIp和servicePort
        //注意：2.8.0以前的SDK需要设置Ip和Port两个参数，2.8.0以后的版本如果设置了region参数则可以省略
        //context.setServiceIp("10.108.112.24");
        //context.setServicePort(50120);

        //创建客户端
        SubscribeClient client=new DefaultSubscribeClient(context);
        //创建订阅listener
        ClusterListener listener= new ClusterListener() {
            @Override
            public void notify(List<ClusterMessage> messages) throws Exception {
                //消费订阅到的数据
                for(ClusterMessage m:messages){
                    for(DataMessage.Record.Field f:m.getRecord().getFieldList()){
                        if(f.getFieldname().equals("id")){
                            System.out.println("seq:"+f.getValue());
                        }
                    }
                }
                //消费完之后，确认消费
                m.ackAsConsumed();
            }
            @Override
            public void onException(Exception e){
                System.out.println("listen exception"+e);
            }
        };
    }
}
```

```
//添加监听者
client.addClusterListener(listener);
//设置请求的订阅通道
client.askForGUID("dts-channel-B2eG8xbLvi472wV3");
//启动客户端
client.start();
}
}
```

整个流程是个典型的生产者消费者模型，SDK 作为消费者不断地从服务器拉取订阅的 Binlog 数据，消费数据，消费完确认消费完数据，比较直观：

1. 首先配置参数，创建消费客户端 `SubscribeClient`。
2. 然后创建一个监听器 `ClusterListener`，消费收到的 Binlog 订阅数据，消费完之后返回确认消息。
3. 最后启动客户端，开始流程。

在监听器 `ClusterListener` 中，可以根据用户自身的需求，对收到的数据进行操作，还可以对收到 Binlog 数据根据类型进行过滤，例如过滤掉所有 `drop` 语句等。

示例代码中，用户需要提供五个参数。

- 其中 `secretId` 和 `secretKey` 是跟用户腾讯云账号关联的密钥值，可以在控制台的【访问管理】>【访问密钥】>【API密钥管理】查看，SDK 用这两个参数来对用户操作进行鉴权。

#### ⚠ 注意：

数据订阅 SDK 已经接入了 CAM 权限控制，根账号默认有所有的权限，可以直接用根账号的云 API 密钥访问；子账号默认没有任何权限，需要根账号给予子账号赋予 `name/dts:AuthenticateSubscribeSDK` 操作的权限，或者赋予 DTS 所有操作的权限 `QcloudDTSFullAccess`。

- 另外三个参数 `servicelp`、`servicePort`、`channelId` 都是与用户 Binlog 订阅相关的，在云数据库 MySQL 相应页面配置好订阅内容后，可在 [DTS 控制台](#) 的【数据订阅】页查看。

#### 📘 说明：

`servicelp` 即数据订阅控制台【服务地址】里的 IP、`servicePort` 即【服务地址】里的端口号、`channelId` 即【通道ID】。

## SDK说明

### SubscribeContext 类

## 类说明

主要用于设置用户 SDK 的配置信息，其中包括安全凭证 secretId、secretKey、订阅服务的 IP 和端口。

## 构造方法

```
public SubscribeContext()
```

## 类方法

### 设置安全凭证 secretId

#### 函数原型

```
public void setSecretId(String secretId)
```

#### 输入参数

参数名	类型	参数含义
secretId	String	安全凭证secretId，可以在控制台的【访问管理】>【访问密钥】>【API密钥管理】查看

#### 返回结果

无

#### 抛出异常

无

### 设置安全凭证 secretKey

#### 函数原型

```
public void setSecretKey(String secretKey)
```

#### 输入参数

参数名	类型	参数含义
secretKey	String	安全凭证 secretKey，可以在控制台的【访问管理】>【访问密钥】>【API密钥管理】查看

#### 返回结果

无

#### 抛出异常

无

### 设置订阅服务的 IP 地址



### 函数原型

```
public void setServiceIp(String serviceIp)
```

### 输入参数

参数名	类型	参数含义
serviceIp	String	订阅服务的 IP 地址，可以在控制台的订阅通道配置页面查看

### 返回结果

无

### 抛出异常

无

### 设置订阅服务的端口

### 函数原型

```
public void setServicePort(String servicePort)
```

### 输入参数

参数名	类型	参数含义
servicePort	String	订阅服务的端口号，可以在控制台的订阅通道配置页面查看

### 返回结果

无

### 抛出异常

无

## SubscribeClient 接口和 DefaultSubscribeClient 接口

### 类说明

DefaultSubscribeClient 类实现了 SubscribeClient 接口。

用于构建订阅 SDK 的客户端程序，也就是 Binlog 消息的消费者。

DefaultSubscribeClient 根据用户的需求提供了同步确认和异步确认的两种实现。在同步的情况下，在每次客户端消费完 Binlog 消息之后，同步地确认已经收到的消息，这样确保了确认完的消息服务器可以尽快收到，此种模式下 SDK 整体性能不如异步确认的方式；在异步的情况下，消费者程序异步确认消息，也就是拉取消息和确认消息异步执行，相互不影响，性能比同步确认好。用户可以根据需求选择不同的确认模式。

### 构造方法

### 构造 DefaultSubscribeClient

#### 函数原型

```
public DefaultSubscribeClient(SubscribeContext context, boolean isSync) throws Exception
```

#### 输入参数

参数名	类型	参数含义
context	SubscribeContext	用户 SDK 的配置信息
isSynce	boolean	表示 SDK 是否使用同步消费模式

#### 返回结果

DefaultSubscribeClient 实例

#### 抛出异常

- `IllegalArgumentException` : 如果用户提交的参数 context, 有参数不合理将抛此异常。不合理包括: 没有安全凭证或者格式出错, 没有服务的 IP 和端口, 或者格式出错。
- `Excetion` : 如果 SDK 内部初始化出错, 将抛 `Exception` 异常。

### 构造 DefaultSubscribeClient

#### 函数原型

```
public DefaultSubscribeClient(SubscribeContext context) throws Exception
```

#### 输入参数

参数名	类型	参数含义
context	SubscribeContext	用户 SDK 的配置信息

#### 返回结果

DefaultSubscribeClient 实例, 默认为异步确认消息

#### 抛出异常

- `IllegalArgumentException` : 如果用户提交的参数 context, 有参数不合理将抛此异常。不合理包括: 没有安全凭证或者格式出错, 没有服务的 IP 和端口, 或者格式出错。
- `Excetion` : 如果 SDK 内部初始化出错, 将抛 `Exception` 异常。

#### 类方法

添加 SDK 消费客户端的监听者

### 函数说明

将监听者 ClusterListener 加入到一个 SubscribeClient 中，才可以订阅通道中的增量数据。

### 函数原型

```
public void addClusterListener(ClusterListener listener) throws Exception
```

### 输入参数

参数名	类型	参数含义
listener	ClusterListener	消费客户端需要使用的监听者，消费 Binlog 消息的主流程应该实现在 ClusterListener 这里面

### 返回结果

无

### 抛出异常

- IllegalArgumentException：如果用户提交的参数 listener 为空，将抛 IllegalArgumentException 异常。
- Exception：当前 SDK 暂时仅支持一个 Listener，如果加入多个监听者，将抛 Exception 异常。

### 请求某个订阅通道的增量数据

### 函数原型

```
public void askForGUID(String channelId)
```

### 输入参数

参数名	类型	参数含义
channelId	String	订阅通道的 ID，可以在控制台的订阅通道配置页面查看

### 返回结果

无

### 抛出异常

无

### 启动 SDK 客户端

### 函数原型

```
public void start() throws Exception
```

### 输入参数

无

## 返回结果

无

## 抛出异常

Exception : 如果 SDK 内部启动出错, 将抛 Exception 异常。

## 停止 SDK 客户端

### 函数原型

```
public void stop(int waitSeconds) throws Exception
```

```
public void stop() throws Exception
```

### 输入参数

参数名	类型	参数含义
waitSeconds	int	等待时间, 单位为秒, 表示等待多久开始强制停止 SDK 的运行

其中, 不带参数的 stop 函数会耐心等待线程停止, 可能等待的时间较长, 具体时间由系统的调度决定; 建议对重启时间有要求的场景, 始终使用带超时时间的 stop 函数。

## 返回结果

无

## 抛出异常

Exception : 如果 SDK 内部关闭出错, 将抛 Exception 异常。

## ClusterListener 接口

### 接口说明

这是一个回调接口, SDK 用户需要实现这个接口的 notify 函数消费订阅数据, 另外通过实现 onException 函数来处理消费过程中可能出现的异常。

### 接口函数

#### 通知 SDK 消费客户端订阅消息

##### 函数说明

主要用来实现对增量数据的消费, 但 SDK 接收到数据时, 会通过 notify 通知 ClusterListener 消费数据。

##### 函数原型

```
public abstract void notify(List<ClusterMessage> messages) throws Exception
```

### 输入参数

参数名	类型	参数含义
-----	----	------

参数名	类型	参数含义
messages	List<ClusterMessage>	订阅数据数组，ClusterMessage 具体实现详见其定义

### 返回结果

无

### 抛出异常

消费订阅数据时，如果有异常会抛到用户实现的 onException 函数中，用户可以根据需求自行处理。

### 消费订阅数据时的异常处理

#### 函数说明

主要用来实现对消费订阅数据时的异常处理，用户在 onException 可以自己实现自己的安全退出策略。

#### 函数原型

```
public abstract void onException(Exception exception)
```

#### 输入参数

参数名	类型	参数含义
exception	Exception	Java标准库中的 Exception 类

### 返回结果

无

### 抛出异常

无

## ClusterMessage 类

### 类说明

类 ClusterMessage 将通过 notify 函数传递消费的订阅数据。每个 ClusterMessage 保存 TencentDB for MySQL 中的一个事务的数据记录，事务中的每条记录通过 Record 保存。

### 类方法

#### 从 ClusterMessage 中获取记录

#### 函数原型

```
public Record getRecord()
```

#### 输入参数

无

## 返回结果

类型	参数含义
Record	变更记录，对应某个事务中某条具体的记录，例如 begin，commit，update，insert 等

## 抛出异常

无

## 确认消费完的数据

## 函数说明

用于向订阅服务器确认已消费完的数据，根据在 SubscribeClient 中设置的值，此函数执行同步或者异步的消息确认。用户消费完之后一定要调用这个数据，否则可能影响正常的逻辑，SDK 会收到重复的数据。

### ⚠ 注意：

这里 SDK 收到的所有消息都必须调用 ackAsConsumed，包括业务逻辑可能不关心的数据，否则 SDK 在获取一定的数据之后不会再拉取新的数据。

## 函数原型

```
public void ackAsConsumed() throws Exception
```

## 输入参数

无

## 返回结果

无

## 抛出异常

Exception：如果确认过程出现内部错误，将抛出异常。

## Record 类

### 类说明

表示订阅的 Binlog 数据中的某一条记录，通常是某个事务 ClusterMessage 的成员，记录可能是 begin，commit，update 语句等。

### 类方法

#### 获取 Record 的属性值

### 函数原型

```
public String getAttribute(String key)
```

## 输入参数

参数名	类型	参数含义
key	String	属性值的名称

可能的属性键值为：

属性键值 Key	说明
record_id	Record 的 ID，这个 ID 在通道内按字符串比较自增有序，不保证连续
source_type	Record 对应数据库实例的引擎类型，目前取值为：mysql
source_category	Record 的类型，目前取值为：full_recorded
timestamp	Record 落 binlog 的时间，这个时间同时也是这条 SQL 在 TencentDB 中执行的时间
sdkInfo	Record 对应的 binlog 文件的位点，格式为：fileoffset@filename，file_name 为 binlog 文件的数字后缀
record_type	Record 对应的操作类型，主要取值包括： insert/update/delete/replace/ddl/begin/commit/heartbeat
db	Record 更新表，对应的数据库名；DDL 类型的记录，此字段为空
table_name	Record 更新表的表名；DDL 类型的记录，此字段为空
record_encoding	Record 对应的编码
primary	Record 更新表的主键列名
fields_enc	Record 每个字段值的编码，各个字段之间用逗号隔开，如果非字符类型那么取值为空
gtid	Record 所在事务的 GTID 值

## 返回结果

类型	参数含义
String	属性值

## 抛出异常

无

## 获取记录的变更类型

### 函数原型

```
public DataMessage.Record.Type getOpt()
```

### 输入参数

无

### 返回结果

类型	参数含义
DataMessage.Record.Type	记录类型 DataMessage.Record.Type 可能的取值包括 : insert、delete、update、replace、ddl、begin、commit、heartbeat。其中 heartbeat 为数据传输内部定义的心跳表，主要用于检查订阅通道是否健康，理论上每秒都会产生一条 heartbeat。

### 抛出异常

无

### 获取记录的在 Binlog 中的 Checkpoint

#### 函数原型

```
public String getCheckpoint()
```

### 输入参数

无

### 返回结果

类型	参数含义
String	记录在 Binlog 中的 Checkpoint，格式为 : binlog_offset@binlogfid。其中 binlog_offset 为变更记录在 binlog 文件中的偏移量，binlog_fid 为 binlog 文件名。

### 抛出异常

无

### 获取记录的在 Binlog 中的时间戳

#### 函数原型

```
public String getTimestamp()
```

### 输入参数

无

### 返回结果



类型	参数含义
String	时间戳字符串

### 抛出异常

无

获取记录的对应的数据库名称

### 函数原型

```
public String getDbname()
```

### 输入参数

无

### 返回结果

类型	参数含义
String	数据库名称字符串

### 抛出异常

无

获取记录的对应的数据表名称

### 函数原型

```
public String getTableName()
```

### 输入参数

无

### 返回结果

类型	参数含义
String	数据表名称字符串

### 抛出异常

无

获取记录的对应的主键列名

### 函数原型

```
public String getPrimaryKeys()
```

**输入参数**

无

**返回结果**

类型	参数含义
String	主键列名，如果是联合主键，这些列名之间用分号分隔

**抛出异常**

无

获取订阅实例的数据库类型

**函数原型**

```
public DBType getDbType()
```

**输入参数**

无

**返回结果**

类型	参数含义
DBType	目前数据传输仅支持 TencentDB for MySQL，为 DBType.MYSQL

**抛出异常**

无

获取 Record 的字段个数

**函数原型**

```
public int getFieldCount()
```

**输入参数**

无

**返回结果**

类型	参数含义
int	Record字段的个数，与该字段对应的表的列数相等，或者是列数的两倍（对于更新操作的 Record）

**抛出异常**

无

判断 Record 是否是事务中的第一条记录

#### 函数原型

```
public Boolean isFirstInLogevent()
```

#### 输入参数

无

#### 返回结果

类型	参数含义
Boolean	如果是事务中的第一条日志，返回 True，否则返回 False

#### 抛出异常

无

获取记录对应表的字段定义列表

#### 函数原型

```
public List<Field> getFieldList()
```

#### 输入参数

无

#### 返回结果

类型	参数含义
List<Field>	Field 数组，具体见 Field 类定义

#### ⚠ 注意：

- 对于 INSERT 类型的记录，List 中 Field 是按订阅表的定义顺序按序对应，Field 中记录的值是插入的值，也即是后镜像。
- 对于 DELETE 类型的记录，List 中 Field 是按订阅表的定义顺序按序对应，Field 中记录的值是删除前的值，也即是前镜像。
- 对于 UPDATE 类型的记录，List 中包含修改前后的值，也即是同时包含前镜像、后镜像；其中前镜像（修改前的值）在list的偶数位，后镜像在list的奇数位；前镜像、后镜像的列表也与订阅表的定义顺序按序对应，因而此时 List 中 Field 的数量是对应订阅表列数的两倍。

#### 抛出异常

无

## Field 类

### 类说明

Field 类定义了每个字段的编码、类型、字段名、字段值以及是否为主键等属性。

### 类方法

#### 获取字段的编码格式

##### 函数原型

```
public String getFieldEnc()
```

##### 输入参数

无

##### 返回结果

类型	参数含义
String	String 类型的字段编码

### 抛出异常

无

#### 获取字段名称

##### 函数原型

```
public String getFieldname()
```

##### 输入参数

无

##### 返回结果

类型	参数含义
String	String 类型的字段名称

### 抛出异常

无

#### 获取字段的数据类型

##### 函数原型

```
public Field.Type getType()
```

### 输入参数

无

### 返回结果

类型	参数含义
Field.Type	Field.Type 是一个枚举类型，对应 MySQL 支持的数据类型，包括：INT8, INT16, INT24, INT32, INT64, DECIMAL, FLOAT, DOUBLE, NULL, TIMESTAMP, DATE, TIME, DATETIME, YEAR, BIT, ENUM, SET, BLOB, GEOMETRY, STRING, UNKOWN

### 抛出异常

无

### 获取字段的值

#### 函数原型

```
public ByteString getFieldname()
```

### 输入参数

无

### 返回结果

类型	参数含义
ByteString	字段的值，当值为空时，值为NULL

### 抛出异常

无

### 判断字段是否为主键

#### 函数原型

```
public Boolean isPrimary()
```

### 输入参数

无

### 返回结果

类型	参数含义
Boolean	如果字段为主键，返回 True，否则返回 False

---

抛出异常

无