ALIBABA CLOUD

阿里云

全局事务服务 GTS

历史文档

文档版本：20220301

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.续费和更改实例规格

如果您有正在使用的按原有 TPS 规格付费的事务分组，仍然可以通过控制台续费或更改实例规格（变配）。

续费

1. 登录 GTS 控制台。
2. 在**事务总览**页面对所要续费的事务分组的操作列单击**续费**。
3. 在**续费**页面选择合适的时长，然后单击**去支付**。

更改实例规格

1. 登录 GTS 控制台。
2. 在**事务总览**页面对所要变配的事务分组的操作列单击**变配**。
3. 在**配置变更**页面选择实例规格，然后单击**去支付**。
   完成支付后，新的实例规格即时生效。

2.旧版样例简介

如果您之前使用过 GTS 的旧版样例，仍然可以继续使用，不过推荐使用新版样例。

下载样例工程，将压缩包解开，可以看到基于 Maven 环境的样例工程。

基于不同使用场景和方式，划分为：

- AT 模式下，在用户代码中使用注解接入分布式事务
  - sample-txc-simple：最简的 GTS 样例工程，该工程给出了一个最少依赖的 GTS 案例，GTS 使用入门必看。您可以基于 sample-txc-simple 样例工程，分别在阿里云网络及公网中搭建。具体使用步骤，请参见 sample-txc-simple 样例。
  - sample-txc-mq：最简 GTS 样例的基础上，将 MQ 加入分布式事务，保证了数据库操作与消息发送的一致性。数据库操作提交，则消息一定发送成功；数据库操作回滚，则消息一定不会被发送出去。本案例是 GTS 和 MQ 综合实践的入门案例。具体使用步骤，请参见 sample-txc-mq 样例。
  - sample-mq-consumer：该工程主要为上述案例中的 MQ 消息提供者（provider）提供一个消费者（consumer）案例，该工程不依赖 GTS，仅为方便用户理解 MQ 案例。
  - sample-txc-edas：在 EDAS 上使用 GTS 的案例，该案例综合了 GTS、EDAS 及 MQ 的使用，通过 Console 和 Web 两种方式充分展现了用户的使用场景，是 EDAS 上部署 GTS 的进阶案例，该案例可以让用户对 GTS 使用场景有更全面的理解。具体使用步骤，请参见 sample-txc-edas 样例。
  - sample-txc-dubbo：在 Dubbo 框架下使用 GTS 的参考案例。该工程演示了通过 GTS 保证跨 Dubbo 服务分布式事务的一致性。您可以基于 sample-txc-dubbo 样例工程，分别在阿里云网络及公网中搭建。具体使用步骤，请参见sample-txc-dubbo 样例和sample-txc-dubbo 样例（公网） 。
  - sample-txc-springboot：在 Spring Boot 框架下使用 GTS 的简单案例。该工程演示了在 SpringBoot 框架下通过 GTS 保证跨数据转账的分布式事务的一致性。具体使用步骤，请参见 sample-txc-springboot 样例。
  - sample-txc-simple-springfree：在非 Spring 框架下使用 GTS 的简单案例。该工程演示了使用 API 的方式通过 GTS 保证跨数据转账的分布式事务的一致性。具体使用步骤，请参见 sample-txc-simple-springfree 样例。
  - sample-txc-springcloud：在 Spring Cloud 框架下使用 GTS 的简单案例。该工程演示了在 Spring Cloud 框架下通过 GTS 保证跨数据转账的分布式事务的一致性。具体使用步骤，请参见 sample-txc-springcloud 样例。
- TCC 模式下，通过两阶段提交接入分布式事务
  - sample-txc-mt-compensat-simple：在 MT 模式下补偿型事务的简单使用方法。该工程演示了通过 GTS 保证 A、B 两个账户转账事务一致性的应用场景。具体使用步骤，请参见 sample-txc-mt-compensate-simple 样例。
  - sample-txc-mt-reserve-simple：在 MT 模式下预留型事务的简单使用方法。该工程样例演示了如何通过 GTS 保证下订单、扣库存两个操作事务的一致性。具体使用步骤，请参见 sample-txc-mt-reserve-simple 样例。

3.sample-txc-simple 样例

sample-txc-simple 样例是最简单的 GTS 样例，所需依赖最少。您可以分别在阿里云网络及公网中搭建该样例。

前提条件

在使用 sample-txc-simple 样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

样例逻辑说明

从 A 账户转账给 B 账户，其中 A 和 B 分别位于两个数据库中，使用 GTS 事务保证 A 账户和 B 账户的总额始终不变。

搭建样例

1. 初始化数据库。
   i. 在 *txc-yun-sample/sql/* 目录下执行 *txc_undo_log.sql*，在两个 RDS 实例中分别创建 txc_undo_log 表。
   ii. 在 *txc-yun-sample/sql/* 目录下执行 *sample-txc-simple.sql*，在 RDS 1 中创建 user_money_a 表，在 RDS 2 中创建 user_money_b 表。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 修改样例配置。
   i. 找到并打开 sample-txc-simple/src/main/resources 目录下的 txc-client-context.xml 文件。
   ii. 将数据库的 url、username和 password 的值修改为您实际使用的值。
   iii. 将 constructor-arg value 的值替换为申请好的 GTS 事务分组 ID。
   iv. 将 accessKey 和 secretKey 的值修改为您实际账号的值。

   > **说明** 请使用主账户的Access Key ID和Access Key Secret。如果使用RAM子账户的Access Key ID和Access Key Secret，需要先使用主账户对子账户进行授权。

4. 在 sample-txc-simple 目录下执行 *build.sh* 命令，编译样例工程。
5. 在 sample-txc-simple 目录下执行 *run.sh* 命令，启动样例。

在公网环境中搭建样例

> **说明**
> - mysql-connector-java 的版本需要和 MySQL 数据库版本匹配。样例的 *pom.xml* 中推荐添加 5.1.38 版本的依赖，该版本已经在 5.0.55、5.6.16、5.6.21 三个 MySQL 数据库版本上验证过。
> - MySQL 数据库的库名、表名和字段名需要设置为大小写不敏感。

1. 初始化数据库。
   i. 在两个 RDS 实例上分别安装 MySQL，并创建数据库 db1 和 db2。

ii. 在 *txc-yun-sample/sql/* 目录下执行 *txc_undo_log.sql*，在两个 RDS 实例中分别创建 txc_undo_log 表。

iii. 在 *txc-yun-sample/sql/* 目录下执行 *sample-txc-simple.sql*，在 RDS 1 中创建 user_money_a 表，在 RDS 2 中创建 user_money_b 表。

2. 下载样例 txc-yun-sample并上传到 ECS 上。

3. 修改样例配置。

i. 找到并打开 *sample-txc-simple/src/main/resources* 目录下的 *txc-client-context.xml* 文件。

ii. 将数据库的 url、username和 password 的值修改为您实际使用的值。

iii. 将 accessKey 和 secretKey 的值修改为您实际账号的值。

> **说明** 请使用主账户的Access Key ID和Access Key Secret。如果使用RAM子账户的Access Key ID和Access Key Secret，需要先使用主账户对子账户进行授权。

iv. 将 Scaner 修改为如下配置。

```
<bean class="com.taobao.txc.client.aop.TxcTransactionScaner">
    <constructor-arg value="myapp"/>
    <constructor-arg value="txc_test_public.1129****3855****.QD"/>
    <constructor-arg value="1" />
    <constructor-arg value="https://test-cs-gts.aliyuncs.com" />
</bean>
```

4. 在 sample-txc-simple 目录下执行 *build.sh* 命令，编译样例工程。

5. 在 *txc-yun-sample/sample-txc-simple/client/bin* 目录下执行 *run.sh* 命令，启动样例。

结果验证

启动样例后，查看运行结果。

4.sample-txc-mq 样例

本样例介绍如何把 MQ 事务消息的发送纳入 GTS 管理的全局事务。

前提条件

在使用 sample-txc-mq 样例前，请先完成以下工作：

- 开通 GTS（创建事务分组）。
- 开通 MQ。具体操作，请参见快速入门概述。
- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

样例逻辑说明

A 账户向 B 账户转账 10 次，前 5 次成功，后 5 次失败。在转账成功后，通过 MQ 增加了一个通知，告知转账成功。使用 GTS 事务保证了账户 A 和 B 总金额始终不变，同时保证了只有转账成功的情况下才能收到 MQ 的消息通知。

搭建样例

1. 下载样例 txc-yun-sample并上传到 ECS 上。
2. 在两个 RDS 实例中分别执行样例中 sql 目录下的 *txc_sample1.sql* 、 *txc_sample2.sql* 和 *txc_undo_log.sql*，创建数据表。
3. 修改样例配置。
   i. 打开 *sample-txc-mq/src/main/resources* 目录下的 *txc-mq-client-context.xml* 文件。
   ii. 将 constructor-arg value 的值替换为申请好的 GTS 事务分组 ID。
   iii. 将 <bean id="txc_mq_producer" class="com.taobao.txc.rm.mq.TxcMQProducerImpl"> 下的参数替换为申请的 MQ 相关配置。
   iv. 将 accessKey 和 secretKey 的值修改为您实际账号的值。

   > **说明** 请使用主账户的Access Key ID和Access Key Secret。如果使用RAM子账户的Access Key ID和Access Key Secret，需要先使用主账户对子账户进行授权。

4. 在 *sample-txc-mq* 目录下执行 *build.sh*，编译样例工程。
5. 编译完成后，在 *sample-txc-mq/client/bin* 目录下执行 *run.sh*，启动样例。
   查看 MQ 的 Provider 的运行结果。
6. 将 sample-mq-consumer 样例工程上传到 ECS。
7. 在 *sample-mq-consumer/src/main/java/com/taobao/txc/tests* 路径下打开 *SMSListener.java*，修改其中的 xxxxx 为申请的 MQ 配置。
8. 在 *sample-mq-consumer* 目录下执行 *build.sh*，编译样例工程。
9. 编译完成后，在 *sample-mq-consumer/client/bin* 目录下执行 *run.sh*，启动样例。
   可以消费掉刚刚 sample-txc-mq 工程生产出来的 MQ 消息。

5.sample-txc-edas 样例

该样例综合了 GTS、EDAS 及 MQ 的使用，通过 Console 和 Web 两种方式充分展现了用户的使用场景，是 EDAS 上部署 GTS 的进阶样例，帮助您更全面的理解 GTS 的使用场景。

前提条件

在使用 sample-txc-edas 样例工程，请先完成以下工作：

- 开通EDAS

  > 说明 需要开通专业版或铂金版。

- 开通 GTS（创建事务分组）
- 在 EDAS 控制台创建或导入 3 个 ECS 实例，详情请参见创建ECS实例，用于部署样例中的 3 个服务。

样例逻辑说明

DecMoney 账号向 IncMoney 账号转账，IncMoney 会根据当前账号余额更新用户相应的 Level。业务为了模拟应用异常产生，故意先给 IncMoney 打款，然后从 DecMoney 扣款后检查 DecMoney 余额，如果为负值则抛出应用异常导致全局事务回滚。

该业务的展现方式有两种，分别是 Web 方式和 Console 方式。此外，还提供了一种带有 MQ 操作的 Console 方式客户端。

样例模块说明

本样例主要包含以下几个子工程：

- txc-client-web：EDAS+GTS 的 MVC 客户端 Spring Mvc，用 Web 方式展现业务逻辑。
- txc-client-console：EDAS+GTS 的 Console 客户端，用 Console 的方式展现业务逻辑。
- txc-client-mq：EDAS+GTS+MQ 的 Console 客户端，用 Console 的方式展现综合了 EDAS 和 MQ 的业务逻辑。
- txc-edas-api：DecMoney、 IncMoney 、UpdateStarLevel 服务接口定义。
- txc-level-service：UpdateStarLevel 的 EDAS 服务端应用（WAR）。
- txc-money-service：IncMoney，DecMoney 的 EDAS 服务端应用（WAR）。

搭建样例

1. 下载样例 txc-yun-sample。
2. 根据数据源的使用情况，修改数据源配置。
   - 同时拥有 DRDS 数据源和 RDS 数据源，并在上述数据源配置文件中配置了它们。则在服务发布后，您将同时有两个分组，一个是 DRDS 的，一个是 RDS 的。
   - 使用 RDS 数据源
     a. 请在下列文件中删除 DRDS 配置：
     - *sample-txc-edas/txc-money-service/src/main/resources/txc-datasource.xml*
     - *sample-txc-edas/txc-level-service/src/main/resources/txc-datasource.xml*
     - *sample-txc-edas/txc-money-service/src/main/resources/hsf-provider-beans.xml*
     - *sample-txc-edas/txc-level-service/src/main/resources/hsf-provider-beans.xml*

b. 在 *sample-txc-edas/txc-client-web/src/main/resources/hsf-consumer-beans-drds.xml* 文件中，将 version 由 *1.1.1* 改为 *1.1.2*，将所有 *DRDS* 替换为 *RDS* 。

sample-txc-edas/txc-client-web/src/main/resources/hsf-consumer-beans-drds.xml

- 使用 DRDS 数据源
  a. 请在下列文件中删除 RDS 配置：
     - *sample-txc-edas/txc-money-service/src/main/resources/txc-datasource.xml*
     - *sample-txc-edas/txc-level-service/src/main/resources/txc-datasource.xml*
     - *sample-txc-edas/txc-money-service/src/main/resources/hsf-provider-beans.xml*
     - *sample-txc-edas/txc-level-service/src/main/resources/hsf-provider-beans.xml*
  b. 在下列文件中，将 version 由 *1.1.2* 改为 *1.1.1*，将所有 *RDS* 替换为 *DRDS*。
     - *sample-txc-edas/txc-client-mq/src/main/resources/hsf-consumer-beans-rds.xml*
     - *sample-txc-edas/txc-client-console/src/main/resources/hsf-consumer-beans-rds.xml*

3. 在 *sample-txc-edas* 目录下执行 *build.sh*，编译样例工程。
4. 将下列编译完成的 WAR 包部署到 EDAS 中。部署操作请参见在ECS集群中创建并部署应用。
   - *sample-txc-edas/txc-level-service/target/txc-level-service.war*
   - *sample-txc-edas/txc-money-service/target/txc-money-service.war*
   - *sample-txc-edas/txc-client-web/target/txc-client-web.war*

结果验证

本样例提供了 WebService 和 Console 两种验证方式，Console 又可以使用 txc-client-console 和 txc-client-mq 两个子工程验证。

- 使用 WebService 验证部署结果
  i. 使用浏览器访问地址 http://ip:8080/txc-client-web/client。

  显示 inc_money、dec_money 和 level 三个表的当前值。

  ii. 通过 reset 可以重置 dec_money 的初始值并清空 inc_money 和 level 表的值。
  iii. 输入每次转款的金额数测试成功的全局事务和余额不足产生异常并回滚的全局事务。
- 使用 Console 验证部署结果
  - 使用 txc-client-console 子工程验证：
    a. 将 sample-txc-edas 工程上传到 ECS 上。
    b. 修改数据源配置。
    c. 在 sample-txc-edas 目录下执行 *build.sh*， 编译样例工程。
    d. 设置运行时环境变量，将下面的各参数的值（xxxx）替换为实际配置。如果需要，可以咨询 EDAS 技术支持人员。

```
export JAVA_OPTS="-Dproject.name=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dtenant.id=xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dspas.identity=/home/admin/.spas_key/default -Daddress.server.domain=xxxx -Daddress.server.port=xxxx -Dconfigserver.client.port=xxxx -DJM.LOG.RETAIN.COUNT=7 -DJM.LOG.FILE.SIZE=300MB"
```

    e. 在 *sample-txc-edas/txc-client-console/client/bin* 目录下执行 *run.sh*，查看执行结果。

- 使用 txc-client-mq 子工程验证：
  a. 将 sample-txc-edas 工程上传到 ECS 上。
  b. 修改数据源配置。
  c. 在 sample-txc-edas 目录下执行 *build.sh*，编译样例工程。
  d. 设置运行时环境变量，将下面的各参数的值（xxxx）替换为实际配置。如果需要，可以咨询 EDAS 技术支持人员。

```
export JAVA_OPTS="-Dproject.name=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dtenant.id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dspas.identity=/home/admin/.spas_key/default -Daddress.server.domain=xxxx -Daddress.server.port=xxxx -Dconfigserver.client.port=xxxx -DJM.LOG.RETAIN.COUNT=7 -DJM.LOG.FILE.SIZE=300MB"
```

  e. 在 *sample-txc-edas/txc-client-mq/client/bin* 目录下执行 *run.sh*，查看 MQ 的 Provider 执行结果。
  f. 将 sample-mq-consumer 工程上传到 ECS。
  g. 在 *sample-mq-consumer/src/main/java/com/taobao/txc/tests* 路径打开 *SMSListener.java*，修改其中的 xxxxx 为申请的 MQ 配置。
  h. 在 sample-mq-consumer 目录下执行 *build.sh*，编译该样例工程。
  i. 编译完成后，在 *sample-mq-consumer/client/bin* 目录下执行 *run.sh*，可以消费掉刚刚 sample-txc-edas/txc-client-mq 工程生产出来的 MQ 消息。

执行结果

1. 使用 webService 方式验证部署结果。
   i. 访问 txc-client-web 客户端应用的地址：http://ip:8080/txc-client-web/client
   ii. 首先显示的是 inc_money，dec_money，level 三个表的当前值。
   iii. 通过 reset 可以重置 dec_money 的初始值并清空 inc_money 和 level 表的值。
   iv. 输入每次转款的金额数测试成功的全局事务和余额不足产生异常并回滚的全局事务。
2. 使用 txc-client-console 子工程的 console 方式验证服务部署结果。
   i. 将 sample-txc-edas 整个工程拷贝到 ECS 服务器上，按步骤（4）修改参数配置，然后在 sample-txc-edas 目录下执行 build.sh 完成工程编译。
   ii. 设置运行时环境变量，将下面的 xxxx 替换为自己的配置，可以咨询 EDAS 管理人员询问。

```
export JAVA_OPTS="-Dproject.name=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dtenant.id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dspas.identity=/home/admin/.spas_key/default -Daddress.server.domain=xxxx -Daddress.server.port=xxxx -Dconfigserver.client.port=xxxx -DJM.LOG.RETAIN.COUNT=7 -DJM.LOG.FILE.SIZE=300MB"
```

1. 在 sample-txc-edas/txc-client-console/client/bin 目录下执行 run.sh，可以看到执行结果。
2. 使用 txc-client-mq 子工程的 console 方式验证综合了 MQ 业务的服务部署结果。
3. 将 sample-txc-edas 整个工程拷贝到 ECS 服务器上，按步骤（4）修改参数配置，然后在 sample-txc-edas 目录下执行 build.sh 完成工程编译。
4. 设置运行时环境变量，将下面的 xxxx 替换为自己的配置，可以咨询 EDAS 管理人员询问。

```
export JAVA_OPTS="-Dproject.name=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dtenant.id=xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx -Dspas.identity=/home/admin/.spas_key/default -Daddress.server.
domain=xxxx -Daddress.server.port=xxxx -Dconfigserver.client.port=xxxx -DJM.LOG.RETAIN.COUN
T=7 -DJM.LOG.FILE.SIZE=300MB"
```

1. 在 sample-txc-edas/txc-client-mq/client/bin 目录下执行 run.sh，可以看到 MQ 的provider执行结果。
2. 将 sample-mq-consumer 工程拷贝到 ECS 服务器中，在 sample-mq-consumer/src/main/java/com/taobao/txc/tests 中找到 SMSListener.java，修改其中的 xxxxx 为申请的 MQ 配置。在 sample-mq-consumer 目录下执行 build.sh 编译该工程，编译完成后在 sample-mq-consumer/client/bin 目录下执行 run.sh 可以消费掉刚刚 sample-txc-edas/txc-client-mq 工程生产出来的 MQ 消息。

6.sample-txc-dubbo 样例（公网）

GTS 从 2.8.18 版本开始支持 Dubbo。您无需再关心全局事务上下文 XID 在服务调用链路上的传播，进一步简化了编程模型。

前提条件

在使用样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

实现原理

基于 Dubbo 的 Filter 机制，GTS 的 SDK 内置了专门用于在调用链路上传播事务上下文的 `TransactionPropagationFilter`，有需要有用户可以对这个类进行 DEBUG 来了解事务传播的机制。

> **说明**
> - Java 运行环境的最低要求是 1.8。
> - 只支持 2.7.0 及以上版本的 Dubbo，即 `org.apache.dubbo` Group 的 Dubbo。

样例逻辑说明

该样例模拟了用户下订单、减库存的业务逻辑。客户端（Client）通过调用订单服务（OrderService）创建订单，之后通过调用库存服务（StockService）扣库存。其中在订单服务读写订单数据库、库存服务读写库存数据库过程中，GTS 将保证跨服务事务的一致性。

搭建样例

1. 初始化数据库。
   i. 进入 *txc-yun-sample/sql* 目录中，执行 *txc_undo_log.sql*，在两个 RDS 实例中分别创建 txc_undo_log 表。
   ii. 在 RDS1 中创建 orders 表，在 RDS2 中创建 stock 表。
   建表语句位于 *txc-yun-sample/sql* 的 *sample-txc-dubbo.sql* 文件中。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 在 IDE 中修改样例配置。
   i. 将
   ii. 将 *resource* 目录下每个 xml 文件中 constructor-arg value 的值替换为申请好的 GTS 事务分组 ID。
   iii. 将 *resource* 目录下每个 xml 文件中 accessKey 和 secretKey 的值修改为您实际账号的值。

   > **说明** 请使用主账户的 Access Key ID / Access Key Secret。如果使用 RAM 子账户的 Access Key ID / Access Key Secret，需要先使用主账户对子账户进行授权。

   iv. 将 url、username和 password 的值修改为您实际使用的值。
4. 在 *txc-yun-sample/sample-txc-dubbo* 目录下执行 `build.sh` 脚本，编译样例。
   编译后会在 *sample-txc-dubbo/client/bin* 目录下生成 *stock_run.sh*、*order_run.sh* 和 *client_run.sh* 三个运行脚本，分别对应库存服务、订单服务以及客户端。

5. 在 *txc-yun-sample/sample-txc-dubbo* 目录下执行 *run.sh* 脚本，启动样例。

   该脚本会依次启动 *order_run.sh*（订单服务）、*stock_run.sh*（库存服务）和 *client_run.sh*（客户端程序）。

7.sample-txc-dubbo 样例

GTS 从 2.8.18 版本开始支持 Dubbo。您无需再关心全局事务上下文 XID 在服务调用链路上的传播，进一步简化了编程模型。

前提条件

在使用样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

实现原理

基于 Dubbo 的 Filter 机制，GTS 的 SDK 内置了专门用于在调用链路上传播事务上下文的 `TransactionPropagationFilter`，有需要有用户可以对这个类进行 DEBUG 来了解事务传播的机制。

> **说明**
> - Java 运行环境的最低要求是 1.8。
> - 只支持 2.7.0 及以上版本的 Dubbo，即 `org.apache.dubbo` Group 的 Dubbo。

样例逻辑说明

该样例模拟了用户下订单、减库存的业务逻辑。客户端（Client）通过调用订单服务（OrderService）创建订单，之后通过调用库存服务（StockService）扣库存。其中在订单服务读写订单数据库、库存服务读写库存数据库过程中，GTS 将保证跨服务事务的一致性。

搭建样例

1. 初始化数据库。
   i. 进入 *txc-yun-sample/sql* 目录中，执行 *txc_undo_log.sql*，在两个 RDS 实例中分别创建 txc_undo_log 表。
   ii. 在 RDS1 中创建 orders 表，在 RDS2 中创建 stock 表。
   建表语句位于 *txc-yun-sample/sql* 的 *sample-txc-dubbo.sql* 文件中。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 在 IDE 中修改样例配置。
   i. 将 *resource* 目录下每个 xml 文件中 constructor-arg value 的值替换为申请好的 GTS 事务分组 ID。
   ii. 将 *resource* 目录下每个 xml 文件中 accessKey 和 secretKey 的值修改为您实际账号的值。
   > **说明** 请使用主账户的 Access Key ID / Access Key Secret。如果使用 RAM 子账户的 Access Key ID / Access Key Secret，需要先使用主账户对子账户进行授权。
   iii. 找到并打开 *txc-yun-sample/sample-txc-dubbo/src/main/resources* 目录下的 *dubbo-order-service.xml* 和 *dubbo-stock-service.xml* 文件，将 url、username和 password 的值修改为您实际使用的值。
4. 在 *txc-yun-sample/sample-txc-dubbo* 目录下执行 `build.sh` 脚本，编译样例。
   编译后会在 *sample-txc-dubbo/client/bin* 目录下生成 *stock_run.sh*、*order_run.sh* 和 *client_run.sh* 三个运行脚本，分别对应库存服务、订单服务以及客户端。

5. 在 *txc-yun-sample/sample-txc-dubbo* 目录下执行 *run.sh* 脚本，启动样例。

   该脚本会依次启动 *order_run.sh*（订单服务）、*stock_run.sh*（库存服务）和 *client_run.sh*（客户端程序）。

8.sample-txc-simple-springfree 样例

该样例工程介绍 AT 模式下，非 Spring 框架下如何通过 GTS 保证跨数据库转账的分布式事务的一致性。

前提条件

在使用样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

样例逻辑说明

从 A 账户向 B 账户转账，如果为负值则抛出应用异常，导致全局事务回滚。

搭建样例

1. 在两个数据库分别执行 *txc_sample1.sql*、*txc_sample2.sql* 和 *txc_undo_log.sql*，创建 *txc_undo_log* 表。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 在 IDE 中修改样例配置。
   i. 找到并打开 *SpringFreeClient.java* 文件，在 txcTransactionScaner 方法中修改 GTS 的逻辑组名。

      事务逻辑组名需要使用包含 ID 和 Region 信息的全名，例如 *gtstest.432242345522.HZ*。
   ii. 找到并打开 *SpringFreeClient.java* 文件，将 accessKey 和 secretKey 的值修改为您实际账号的值。

      > **说明** 请使用主账户的Access Key ID和Access Key Secret。如果使用RAM子账户的Access Key ID和Access Key Secret，需要先使用主账户对子账户进行授权。

   iii. 找到并打开 *SpringFreeClient.java* 文件，将两个数据库的 url、username和 password 的值修改为您实际使用的值。

   Java 源代码在 */sample-txc-simple-springfree/src/main/java/com/taobao/txc/tests* 目录下，可以根据业务需求修改。
4. 运行 `mvn package -Dmaven.test.skip=true -Ptest` 命令，生成 JAR 包 *pay.jar*。
5. 在 *sample-txc-simple-springfree* 目录下执行 *build.sh* 编译本工程。

结果验证

编译完成后，在 *sample-txc-simple-springfree/client/bin* 目录下执行 *run.sh*，可以看到运行结果。

9.sample-txc-springboot 样例

该样例工程介绍 AT 模式下，Spring Boot 框架如何通过 GTS 保证跨数据库转账的分布式事务的一致性。

前提条件

在使用样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

样例逻辑说明

从 A 账户向 B 账户转账。为了触发应用异常，样例故意先给 B 账户打款，然后从 A 账户扣款后检查 A 账户余额。如果为负值则抛出应用异常，导致全局事务回滚。

搭建样例

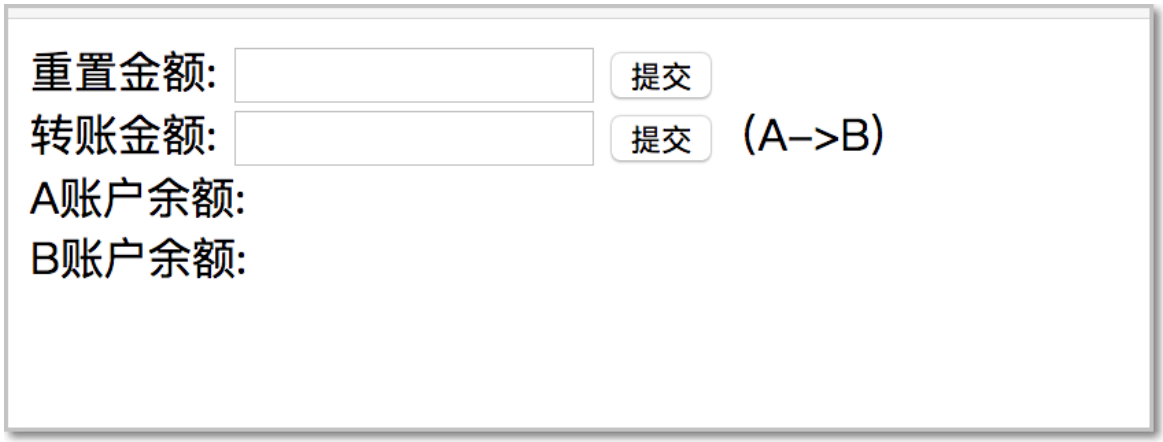
1. 初始化数据库。
   i. 在两个数据库分别执行 *sample-txc-springboot.sql* 和 *txc_undo_log.sql*，创建 *txc_undo_log* 表。
   ii. 分别在两个数据库执行 *txc_sample_springcloud.sql* 中的相关命令，创建 *account* 表。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 修改样例配置。
   i. 找到并打开 *AccConfig.java* 文件，在 txcTransactionScaner 方法中修改 GTS 的逻辑组名。
   事务逻辑组名需要使用包含 ID 和 Region 信息的全名，例如 *gtstest.432242345522.HZ*。
   ii. 找到并打开 *application.properties* 文件，将 accessKey 和 secretKey 的值修改为您实际账号的值。

   > **说明** 请使用主账户的 Access Key ID / Access Key Secret。如果使用 RAM 子账户的 Access Key ID / Access Key Secret，需要先使用主账户对子账户进行授权。

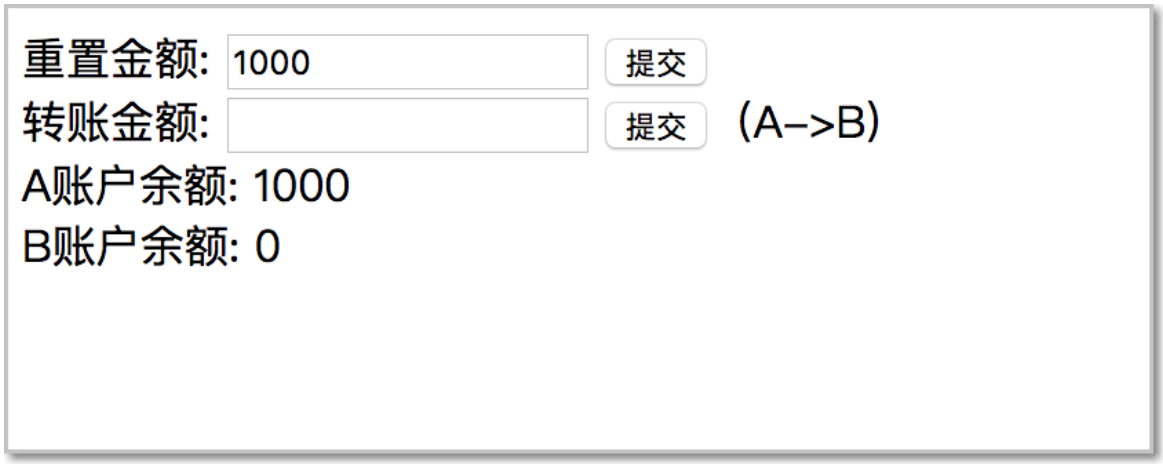
   iii. 找到并打开 *application.properties* 文件，将两个数据库的 url、username和 password 的值修改为您实际使用的值。
4. 运行 *txc-yun-sample/sample-txc-springcloud* 目录下的 *sh build.sh* 脚本，编译工程。
   编译后会在相应文件中生产 *client* 目录。
5. 在 *sample-txc-springboot/client/bin* 目录下运行 *sh pay_run.sh*，启动样例。

结果验证

1. 使用浏览器访问地址 http://<ECS IP>:8080/pay。

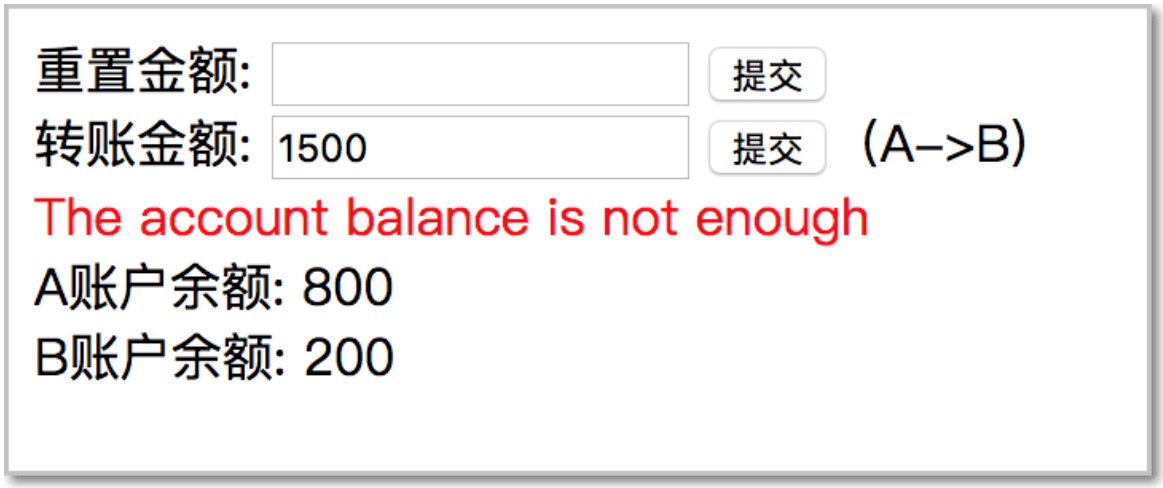


2. 输入重置金额，然后单击**提交**。



3. 输入转账金额，然后单击**提交**。

4. 余额不足，转账失败，全局事务回滚。


重置金额:
提交
转账金额:
1500
提交
(A->B)
The account balance is not enough
A账户余额: 800
B账户余额: 200

10.sample-txc-springcloud 样例

该样例工程介绍 AT 模式下，Spring Cloud 框架如何通过 GTS 保证跨数据库转账的分布式事务的一致性。

前提条件

在使用样例前，请先完成以下工作：

- 准备两个 RDS 实例，用户创建数据库。
- 准备一个 ECS 实例，用于部署本样例。

样例逻辑说明

模拟转账业务，从 A 账户扣款给 B 账户打款。应用程序通过调用扣款微服务和存款微服务完成转账业务。

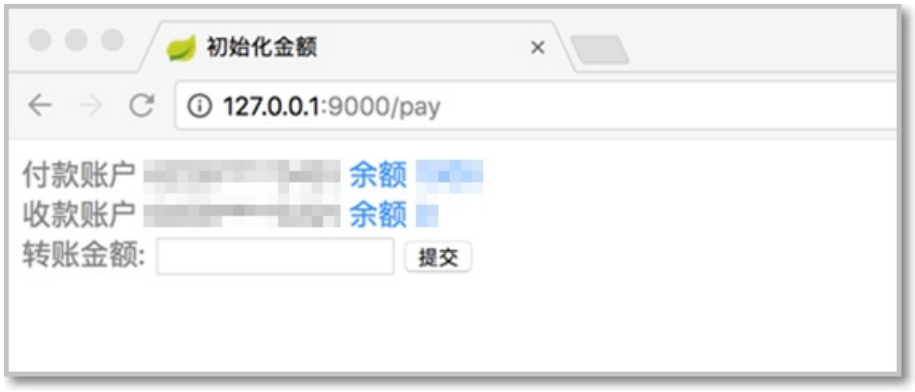
- eureka-locator：微服务的注册中心
- eureka-consumer：应用程序
- eureka-provider：扣款服务
- eureka-provider2：存款服务

搭建样例

1. 初始化数据库。
   i. 分别在两个数据库执行 *txc_undo_log.sql*，创建 *txc_undo_log* 表。
   ii. 分别在两个数据库执行 *txc_sample_springcloud.sql* 中的相关命令，创建 *account* 表。
2. 下载样例 txc-yun-sample并上传到 ECS 上。
3. 修改样例配置。
   i. 分别找到并打开 sample-txc-springcloud 样例下 *eureka-consumer*、*eureka-provider* 和 *eureka-provider2* 三个文件中的 *config.java* 文件，在 txcTransactionScaner 方法中修改 GTS 的事务分组名称。
   ii. 分别找到并打开 *eureka-consumer*、*eureka-provider* 和 *eureka-provider2* 三个文件中的 *application.properties* 文件，将 accessKey 和 secretKey 的值修改为您实际账号的值。
   iii. 分别找到并打开 *eureka-consumer*、*eureka-provider* 和 *eureka-provider2* 三个文件中的 *application.properties* 文件，将数据库的 url、username和 password 的值修改为您实际使用的值。
4. 运行 *txc-yun-sample/sample-txc-springcloud* 目录下的 *sh build.sh* 脚本，编译工程。
   编译后会在相应文件中生产 *client* 目录。
5. 运行样例。
   i. 运行 *eureka-locator/client/bin* 目录下的 *locator.sh* 脚本，启动注册中心。
   ii. 运行 *eureka-provider/client/bin* 目录下的 *provider1_run.sh* 脚本，启动扣款服务。
   iii. 运行 *eureka-provider2/client/bin* 目录下的 *provider2_run.sh* 脚本，启动存款服务。
   iv. 运行 *eureka-consumer/client/bin* 目录下的 *consumer_run.sh* 脚本，启动样例。

结果验证

1. 使用浏览器访问地址 http://<ECS IP>:9000。



2. 在页面中输入转账金额，然后单击**提交**。

3. 余额不足，转账失败，全局事务回滚。