**Token跨区优化方案设计**

**云平台**

*TP-LINK TECHNOLOGIES CO., LTD*

[*http://www.tp-link.com.cn*](http://www.tp-link.com.cn)

*SouthBuilding, No.5 Keyuan Road, Central Zone*

*Science & TechnologyPark, NanShan*

*ShenZhen, P. R. China*

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本/状态** | **作者** | **起止日期** | **描述** |
| V0.9.0 | 莫阳 | 2017-8-8 | 根据禅道系统中的评审意见进行修改：   1. 调整时序图的位置 2. 合并4.3小节（Token跨区业务中可能出现的异常情况及后果）至3.4小节（Token跨区流程优化） |
| V0.8.0 | 莫阳 | 2017-8-1 | 根据评审意见修改部分跨区操作的优化流程；  补充新旧方案在常见业务场景下的时序图；  更新优化方案中新token生成方式的内容等 |
| V0.1.3 | 郭伟 | 2017-5-2 | 添加最大在线数为1时可能的异常情况 |
| V0.1.2 | 郭伟 | 2017-4-28 | 添加各步骤异常情况 |
| V0.1.1 | 郭伟 | 2017-4-26 | 添加最大在线数异常情况 |
| V0.1.0 | 郭伟 | 2017-4-25 | 根据评审意见修改   1. token添加生成区域信息 |
| V0.0.3 | 郭伟  吴瑨强 | 2017-4-24 | 修改删除token流程 |
| V0.0.2 | 郭伟  吴瑨强 | 2017-4-21 | 添加初版优化方案 |
| V0.0.1 | 郭伟 | 2017-4-19 | 完成初稿 |

**目录**

[1 概述 1](#_Toc489986348)

[2 现有设计 1](#_Toc489986349)

[2.1 Token在缓存中的存储形式 1](#_Toc489986350)

[2.1.1 Token（所有区） 1](#_Toc489986351)

[2.1.2 AccountToken（主区） 1](#_Toc489986352)

[2.2 消息队列（主区RabbitMQ） 2](#_Toc489986353)

[2.2.1 写token.fanout 2](#_Toc489986354)

[2.2.2 读token.%s.queue 2](#_Toc489986355)

[2.2.3 事件与消息体 2](#_Toc489986356)

[2.3 跨区相关业务流程 3](#_Toc489986357)

[3 优化方案 3](#_Toc489986358)

[3.1 Token在缓存中的存储形式 3](#_Toc489986359)

[3.1.1 Token（所有区） 3](#_Toc489986360)

[3.1.2 AccountToken（所有区） 3](#_Toc489986361)

[3.2 Token的生成规则 3](#_Toc489986362)

[3.2.1 现有的token生成方式 3](#_Toc489986363)

[3.2.2 优化后的token生成方式 3](#_Toc489986364)

[3.3 消息队列（主区RabbitMQ） 4](#_Toc489986365)

[3.3.1 写token.all.fanout 4](#_Toc489986366)

[3.3.2 读token.%s.queue 5](#_Toc489986367)

[3.3.3 事件与消息体 5](#_Toc489986368)

[3.4 Token跨区流程优化 6](#_Toc489986369)

[3.4.1 生成token 6](#_Toc489986370)

[3.4.1.1 原有流程 6](#_Toc489986371)

[3.4.1.2 优化流程 8](#_Toc489986372)

[3.4.2 校验token 9](#_Toc489986373)

[3.4.2.1 原有流程 9](#_Toc489986374)

[3.4.2.2 优化流程 10](#_Toc489986375)

[3.4.3 删除token 12](#_Toc489986376)

[3.4.3.1 原有流程 12](#_Toc489986377)

[3.4.3.2 优化流程 13](#_Toc489986378)

[3.4.4 批量删除指定accountId下的多个token 15](#_Toc489986379)

[3.4.4.1 原有流程 15](#_Toc489986380)

[3.4.4.2 优化流程 16](#_Toc489986381)

[3.4.5 批量删除指定accountId和appType下的多个token 17](#_Toc489986382)

[3.4.5.1 原有流程 17](#_Toc489986383)

[3.4.5.2 优化流程 18](#_Toc489986384)

[4 异常情况 19](#_Toc489986385)

[4.1 同时在线数设为1，不同区同时登录的问题 19](#_Toc489986386)

[4.2 同时在线的token数可能超过预设的最大在线数 19](#_Toc489986387)

[5 变更总结 1](#_Toc489986388)

[6 线上更新 1](#_Toc489986389)

# 概述

针对目前token跨区操作的不合理之处提出此优化方案。现有方案中将美国区设立为主区，并且将AccountToken只存储于主区，这就导致了各从区对token的业务处理在很多情况下都会触发对主区的读写操作，使得系统的平均响应时延因为相对小概率的跨区访问请求而增加。本优化方案的主要思路是取消“主区”的概念，各区都维护自己的AccountToken记录，并通过消息队列来同步各区之间的数据，已达到减少跨区操作提高响应效率的目的。

# 现有设计

## Token在缓存中的存储形式

### Token（所有区）

Key: {token.%s}:%s

Value: Map

过期时间：APP 30天，第三方应用15天

举例：

127.0.0.1:6380> HGETALL {token.23e2fc1d}:73c3abf949eb4994b81c9f8

1) "createdAt"

2) "2017-04-19T10:46:50.013+08:00"

3) "ip"

4) "172.29.76.64"

5) "terminalUUID"

6) "termUUID"

7) "appType"

8) "postman"

9) "accountId"

10) "546499"

### AccountToken（主区）

Key: account:%s:id

Value: Map

过期时间：不过期

举例：

127.0.0.1:6380> HGETALL account:546499:id

1) "tplink.cloud.test"

2) "23e2fc1d-179b7d5a0e8f4d84acee710,23e2fc1d-b4102d36319d47f3992fc2c"

3) "ifttt.router.staging"

4) "23e2fc1d-192eea989cd24e7fa7b1279"

5) "postman"

6) "23e2fc1d-7e3b49580a3345718e44d34,23e2fc1d-b9e86b0797d9468ca08781a,23e2fc1d-f1ab2bef398c4f2ebe59b6e"

## 消息队列（主区RabbitMQ）

主要用于通知从区删除单个或多个token。

### 写token.fanout

应用场景举例：

1. 用户logout，触发removeDalToken事件，用于删除单个token
2. 用户修改密码、重置密码（或OAuthServer解除授权），触发removeDalAccountTokens事件，用于批量删除tokens



### 读token.%s.queue

token.%s.queue消息队列分别对应每个从区，没有对应主区的队列。

%s可取ie和sg

### 事件与消息体

1. 删除单个token

{

"event": "removeDalToken",

"params": {

"token": "ABCDE12345"

}

}

1. 批量删除tokens

{

"event": "removeDalAccountTokens",

"params": {

"accountTokens": ["123ABC", "234BCD"]

}

}

## 跨区相关业务流程

为方便与优化方案进行对比，token跨区相关业务的现有流程请参考3.4章节下的各“原有流程”小节内容。

# 优化方案

## Token在缓存中的存储形式

### Token（所有区）

Token存储形式与现有方案保持一致。但是优化后的token采用1位字符来表示生成区，并且由于已经取消了“主区”的概念，所以新生成的token自然也不会再写入“主区”。

### AccountToken（所有区）

AccountToken的存储形式由现有方案的只存主区改为存所有区。

## Token的生成规则

### 现有的token生成方式

现有token由以下几个部分组成：

|  |  |  |
| --- | --- | --- |
| **hashCode-8** | **Separator-1** | **UUID-sub-23** |

Token长度=8+1+23=32 字符，生成代码如下：

TOKEN\_SEPARATOR = "-";

hashCode = DigestUtils.md5DigestAsHex(accountId.getBytes("UTF-8")).substring(0, 8);

token = hashCode + TOKEN\_SEPARATOR + CommonUtils.getUUID().substring(0, 23);

其中，CommonUtils.getUUID()的结果是基于16个字符（"0"~"9","a"~"f"）表示的32位字符串长度的UUID，而UUID-sub-23指的是该UUID字符串的前23位。

### 优化后的token生成方式

优化后的token由以下几个部分组成：

|  |  |  |  |
| --- | --- | --- | --- |
| **hashCode-8** | **Separator-1** | **Region-1** | **UUID-in-22** |

Token长度=8+1+1+22=32 字符，生成代码如下：

REGION\_SIGN = "A";

TOKEN\_SEPARATOR = "-";

hashCode = DigestUtils.md5DigestAsHex(accountId.getBytes("UTF-8")).substring(0, 8);

token = hashCode + TOKEN\_SEPARATOR + REGION\_SIGN + CommonUtils.getUUID22();

其中，REGION\_SIGN表示生成token的区域，CommonUtils.getUUID22()的结果是采用Base58编码后的22位字符串长度的UUID，UUID-in-22即该UUID字符串的全部。

1. **REGION\_SIGN说明**

一般情况下，REGION\_SIGN使用1位字符表示即可满足需求，还能做到和现存token兼容。

原有token在TOKEN\_SEPARATOR后面的部分采用UUID的前23位字符，由于采用的是16进制编码所以字符取值必为"0"~"9"或"a"~"f"。因此我们可以使用大写字母（"A"~"Z"）来表示REGION\_SIGN，从而区分新旧token。 例如：

* us-east-1: A
* eu-west-1: B
* ap-southeast-1: C

1. **CommonUtils.getUUID22()说明**

引入REGION\_SIGN后为保持token的32位字符长度不变，UUID部分的长度由原来的23位减少为22位，如果直接采用UUID（32位字符串表示）的前22位字符势必会降低新token的随机性。因此我们考虑采用Base58编码来解决这个问题。

Base58编码是一种基于58个可打印字符来表示二进制数据的方法。假设需要X位字符来表示一个128bit的UUID，那么需要满足X^58 > 2^128，经过计算可知X > 21.8，即采用Base58编码恰好需要22位字符来表示一个UUID。本方案中选取58个可打印字符如下：

"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz"

注：去除了几个看起来会产生歧义的字符，如 0 (零), O (大写字母O), I (大写的字母i) 和l (小写的字母L)

1. **新老token随机性比较**

由于新token的UUID-in-22部分代表的是一个完整的UUID，而老token的UUID-sub-23只是32位UUID字符串表示的一部分（前23位子串），所以新token的随机性要好于老token。

## 消息队列（主区RabbitMQ）

### 写token.all.fanout

应用场景举例：

1. login添加token
2. logout删除token
3. 用户修改密码批量删除tokens
4. OAuthServer解除授权批量删除tokens



### 读token.%s.queue

%s取use1, aps1, euw1等

### 事件与消息体

1. login

更新本区AccountToken

{

"header": {

"sourceRegion": "use1",

"timestamp": 13572468

},

"event": "login",

"params": {

"accountId": 1000,

"appType": "testAppType",

"token": "ABCDE12345"

}

}

1. logout

删除本区token并更新AccountToken

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "logout",

"params": {

"accountId": 1000,

"appType": "testAppType",

"token": "ABCDE12345"

}

}

1. 批量删除token（appType为可选参数）
   1. 批量删除指定accountId和appType下的所有token

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "removeTokens",

"params": {

"accountId": 1000,

"appType": "testAppType"

}

}

* 1. 批量删除指定accountId下的所有token

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "removeTokens",

"params": {

"accountId": 1000

}

}

## Token跨区流程优化

### 生成token

主要业务场景为用户以密码方式登陆（loginByPassword）、刷新token（refreshToken）操作等。

#### 原有流程

该业务的原有流程时序图如下，



图 3‑1 login原有流程时序图

主要操作步骤有：

1. 写本区Token
   1. 若非主区，再***写主区***Token
2. ***读主区***AccountToken，获取当前accountId和appType下的tokenList  
   将新token加入该tokenList，判断tokenList长度是否超过最大在线数限制
   1. 若超过，从tokenList中删除最老的Token0
   2. ***读主区***，判断Token0是否存在 （假设存在）
      1. ***写主区***，为Token0添加logoff标记
      2. ***写主区***，重设Token0的过期时间
      3. ***写消息队列***，用于删除从区Token0
3. ***写主区***AccountToken，将新的tokenList写回

涉及的事件与消息体如下（用于通知各区删除token0）

{

"event": "removeDalToken",

"params": {

"token": "ABCDE12345"

}

}

#### 优化流程

该业务的优化流程时序图如下，



图 3‑2 login优化流程时序图

主要操作步骤有：

1. 写本区Token
2. 更新本区AccountToken（添加Token）
3. ***写消息队列（token.all.fanout）***，通知各区更新AccountToken（添加）

（注：若写消息队列时出现异常，则可能会导致实际在线数超过设定的最大在线数，详见4.2小节举例）

各区收到消息后更新本区AccountToken，更新AccountToken（添加Token）通用流程如下：

1. 读本区AccountToken，获取当前accountId和appType下的tokenList
2. 将新token加入该tokenList，判断tokenList长度是否超过最大在线数限制
   1. 若超过，从tokenList中删除最老的Token0，读本区（假设本区存在Token0）
      1. 若Token0是本区生成，
         1. 写本区，为Token0添加logoff标记
         2. 写本区，重设Token0的过期时间
      2. 若Token0非本区生成
         1. 直接删除本区的Token0
3. 写本区AccountToken，将新的tokenList写回

涉及的事件与消息体如下（用于通知各区更新AccountToken）

{

"header": {

"sourceRegion": "use1",

"timestamp": 13572468

},

"event": "login",

"params": {

"accountId": 1000,

"appType": "testAppType",

"token": "ABCDE12345"

}

}

### 校验token

主要业务场景为用户发起verifyToken操作等。

#### 原有流程

该业务的原有流程时序图如下，



图 3‑3 verifyToken原有流程时序图

主要操作步骤有：

1. 读本区Token
   1. 若Token不存在且非主区，***读主区***Token （假设主区Token存在）
      1. ***读主区***Token TTL
      2. 写本区Token
2. 返回校验结果

#### 优化流程

该业务的优化流程时序图如下，



图 3‑4 verifyToken优化流程时序图

主要操作步骤有：

1. 读本区Token
   1. 若Token不存在且非本区生成，***读生成区***Token
      1. 如果生成区Token存在，写本区Token，设定TTL为1小时
      2. 如果生成区Token不存在，写本区Token并标记过期，设定TTL为1小时

### 删除token

主要业务场景为用户执行登出（logout）操作等。

#### 原有流程

该业务的原有流程时序图如下，



图 3‑5 logout原有流程时序图

主要操作步骤有：

1. 读本区Token
   1. 若Token不存在且非主区，***读主区***Token
   2. 若本区Token存在，删本区Token
2. 假设主区Token存在
   1. ***删主区***Token
   2. ***读主区***AccountToken，获取当前accountId和appType下的tokenList
   3. 从tokenList中移除Token，并***写回主区***AccountToken
3. ***写消息队列***，用于删除从区Token

涉及的事件与消息体如下（用于通知各区删除token）

{

"event": "removeDalToken",

"params": {

"token": "ABCDE12345"

}

}

#### 优化流程

该业务的优化流程时序图如下，



图 3‑6 logout优化流程时序图

主要操作步骤有：

1. 读本区Token（校验cloudUserName）
   1. 若Token不存在且非生成区，***读生成区Token***

注：校验成功才能发消息队列通知其他各区删除Token，因此有必要进行这一步跨区操作，但一般情况下不会走到这一步，因为logout之前通常会有校验token的操作。

1. 删本区Token （若本区Token存在）
2. 更新本区AccountToken（删除Token）
3. ***写消息队列（token.all.fanout）***，通知各区（包括生成区）删除Token以及更新AccountToken

（注：若写消息队列时出现异常，则可能会导致该Token仍然可用）

其他各区收到消息后：

1. 删除本区token
2. 更新本区AccountToken（删除token）

更新AccountToken（删除token）的通用流程如下：

1. 读本区AccountToken，获取当前accountId和appType下的tokenList
2. 从tokenList中移除token，并写回本区AccountToken

涉及的事件与消息体如下（用于通知各区删除Token，并更新AccountToken）

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "logout",

"params": {

"accountId": 1000,

"appType": "testAppType",

"token": "ABCDE12345"

}

}

### 批量删除指定accountId下的多个token

主要业务场景为用户执行修改密码或重置密码等操作。

#### 原有流程

该业务的原有流程时序图如下，



图 3‑7批量删除指定accountId下的多个token原有流程时序图

主要操作步骤有：

1. ***读主区***AccountToken，获取当前accountId下的所有tokenList
2. ***删除主区***accountId对应的AccountToken
3. ***删除主区所有***tokenList里的tokens
4. ***写消息队列***，用于删除从区tokenList里的tokens

涉及的事件与消息体如下（用于通知各区删除该accountId下的所有tokens）

{

"event": "removeDalAccountTokens",

"params": {

"accountTokens": ["123ABC", "234BCD"]

}

}

#### 优化流程

该业务的优化流程时序图如下，



图 3‑8 批量删除指定accountId下的多个token优化流程时序图

主要操作步骤有：

1. 读本区AccountToken，获取当前accountId下的所有tokenList
2. 删除本区accountId对应的AccountToken记录
3. 删除本区tokenList里的tokens

（注：若删除tokenList出现异常，返回接口调用失败，该tokenList里的tokens将只能通过logout或自动过期失效）

1. ***写消息队列（token.all.fanout）***，通知其他区批量删除该accountId下的tokens

涉及的事件与消息体如下（用于通知各区删除该accountId下的所有tokens，并更新AccountToken）

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "removeTokens",

"params": {

"accountId": 1000

}

}

### 批量删除指定accountId和appType下的多个token

典型业务场景为OAuthServer解除授权的操作。

#### 原有流程

该业务的原有流程时序图如下，



图 3‑9批量删除指定accountId和appType下的多个token原有流程时序图

主要操作步骤有：

1. ***读主区***AccountToken，获取当前accountId和appType下的tokenList
2. ***删除主区***AccountToken中的appType键值对
3. ***删除主区***tokenList里的tokens
4. ***写消息队列***，用于删除从区tokenList里的tokens

涉及的事件与消息体如下（用于通知各区删除该accountId和appType下的所有tokens）

{

"event": "removeDalAccountTokens",

"params": {

"accountTokens": ["123ABC", "234BCD"]

}

}

#### 优化流程

该业务的优化流程时序图如下，



图 3‑10 批量删除指定accountId和appType下的多个token优化流程时序图

主要操作步骤有：

1. 读本区AccountToken，获取当前accountId和appType下的tokenList
2. 删除本区AccountToken中的appType键值对
3. 删除本区tokenList里的tokens

（注：若删除tokenList出现异常，返回接口调用失败，该tokenList里的tokens将只能通过logout或自动过期失效）

1. ***写消息队列（token.all.fanout）***，通知其他区批量删除该tokenList下的tokens

涉及的事件与消息体如下（用于通知各区删除该accountId和appType下的所有tokens，并更新AccountToken）

{

"header": {

"sourceRegion": "use1",

"timestamp": 123456

},

"event": "removeTokens",

"params": {

"accountId": 1000,

"appType": "testAppType"

}

}

# 异常情况

## 同时在线数设为1，不同区同时登录的问题

若要求同时在线数为1，那么在不同区同时登录，可能会被同时挤下线。因为各区均会先更新本区的AccountToken，然后发消息队列通知其他区。

## 同时在线的token数可能超过预设的最大在线数

若要求最大同时在线数为N，那么最坏情况下（只写入本区），实际的同时在线数最大为

N \* Region数

例如，当要求最大同时在线数为1时，区域数为3时：

1. 在aps1登录成功且同步AccountToken成功，此时可以用T0-aps1登录



1. 在use1登录成功但同步AccountToken到aps1失败（此时可以用T1-use1和T0-aps1登录）



1. 在eu登录成功且同步AccountToken均失败（此时可以用T2-eu、T1-use1和T0-aps1登录）



1. 本次登录的区域和同步成功的区域将全部被挤下线，例如在aps1登录且同步AccountToken成功，此时可以用T3-aps1登录



# 变更总结

* 借助消息队列减少跨区操作，节省响应时间
* Auth模块根据优化方案进行开发
* AppServer模块的logout接口对接Auth模块
* Account模块的修改密码和重置密码接口对接Auth模块

# 线上更新

**（上线计划待更新）**