云备份机制（二期）（三）

设计文档

|  |  |
| --- | --- |
| 文档名称： | 云备份机制（二期）（三）设计文档 |
| 文档编号： | P01-138-2016(01)-2.4.5-01.3-L-D (01) |
| 团队负责人： | 韩伟力 |
| 提交日期： | 2016年7月20号 |

版 本 信 息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **版本号** | **修订人** | **修订日期** | **说明** |
| 1 | 0.1 | 梁蛟 | 20160630 | 框架和初始版本 |
| 2 | 0.2 | 梁蛟 | 20160711 | 修改框架 |
| 3 | 0.3 | 梁蛟 | 20160712 | 加入接口函数设计框架 |
| 4 | 0.4 | 刘武 | 20160713 | 修改接口函数设计模块 |
| 5 | 0.5 | 刘武 | 20160716 | 添加类图、系统总流程图等初版 |
| 6 | 0.6 | 梁蛟 | 20160717 | 添加和修改描述部分及数据结构 |
| 7 | 0.7 | 刘武、梁蛟 | 20160718 | 添加并修改文档中的图表 |
| 8 | 0.8 | 梁蛟 | 20160719 | 修改接口函数设计模块 |
| 9 | 0.9 | 刘武 | 20160720 | 添加和修改图表描述 |
|  |  |  |  |  |

审 批 记 录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **版本号** | **修订人** | **修订日期** | **审批意见** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

注：审批人一般为项目负责人或由项目负责人指定。

**目录**

[1 概述 5](#_Toc469647143)

[2 系统优化介绍 5](#_Toc469647144)

[3 数据结构分析 6](#_Toc469647145)

[3.1 访问令牌 6](#_Toc469647146)

[3.2 文件属性 7](#_Toc469647147)

[3.3 会话信息 8](#_Toc469647148)

[4 系统部署图 9](#_Toc469647149)

[5 系统结构图 10](#_Toc469647150)

[6 系统框架设计 10](#_Toc469647151)

[7 核心功能介绍 11](#_Toc469647152)

[7.1 交互适配管理 11](#_Toc469647153)

[7.2 会话持久化管理 12](#_Toc469647154)

[7.3 本地缓存管理 13](#_Toc469647155)

[7.4 系统优化 13](#_Toc469647156)

[8 类图设计 15](#_Toc469647157)

[9 流程设计 17](#_Toc469647158)

[9.1 系统总体流程设计 17](#_Toc469647159)

[9.2 交互适配管理流程设计 18](#_Toc469647160)

[9.2.1 命令接口转换设计 18](#_Toc469647161)

[9.3 会话持久化管理流程设计 19](#_Toc469647162)

[9.3.1 会话持久化流程设计 19](#_Toc469647163)

[9.3.2 会话加密管理流程设计 19](#_Toc469647164)

[9.4 本地缓存管理流程设计 20](#_Toc469647165)

[9.4.1 数据上传缓存管理流程设计 20](#_Toc469647166)

[9.4.2 数据下载缓存管理流程设计 20](#_Toc469647167)

[9.5 系统优化流程设计 21](#_Toc469647168)

[9.5.1 大文件上传流程设计 21](#_Toc469647169)

[9.5.2 大文件下载流程设计 21](#_Toc469647170)

[9.5.3 并发任务执行流程设计 22](#_Toc469647171)

[9.5.4 中间件授权流程设计 22](#_Toc469647172)

[10 接口函数设计 23](#_Toc469647173)

[10.1 中间件接口 23](#_Toc469647174)

[10.2 中间件接口功能列表 25](#_Toc469647175)

# 概述

云存储与云备份系统可分为云存储服务器，身份认证服务器，密钥管理服务器，云存储与云备份中间件，云存储与云备份客户端这五部分。云存储与云备份中间件，提供了与存储服务器交互的文件传输、加解密和解/压缩等核心机制，有效分离了客户端和与基础设施之间的绑定。在未来，针对客户端的改善，例如用户的界面的改善，更多的使用场景，与客户端桌面系统的更紧密的结合等。同样，中间件可以对数据相关的业务进行持续的优化，而不会影响客户端的稳定性；当然，因为中间件的存在，客户端可以提供更多的形式。在服务器发生一些改变时，也可以通过中间件进行调整，以兼容以前的客户端。此外，云存储与云备份中间件提供给云存储与云备份应用软件与云存储服务器，身份认证服务器，密钥管理服务器之间稳定可靠的通讯和数据传输。因此，云存储与云备份中间件的开发是迫切而重要的。

根据目前桌面计算技术的发展趋势，在安全可靠桌面操作系统中实现云备份机制，以满足某些特定应用的现实需求。桌面操作系统的云备份服务由桌面系统云备份客户端和云备份服务器两部分组成，云备份服务能够给用户提供云存储、文件管理、权限控制、共享、取消共享与在线办公等功能。基于云备份中间件前期任务的成果，在Linux平台上，需要继续完善其平台框架和API支持机制，以支撑更多的桌面OS组件使用云备份与存储服务。

# 系统优化介绍

虽然在第一期任务中我们实现了云备份客户端的中间件，但任务主要集中在相关功能的实现上。通过测试分析，已有产品还可从以下几方面进行优化：

（1） 多应用适配支持

在第一期任务中，中间件主要是配合客户端，将任务提交给服务器，从而屏蔽服务器实现细节，简化客户端的设计和实现，也增强了客户端的功能。

由于云存储服务的普及化，提供云存储的服务商越来越多，很多应用可以提供直接访问云上的存储，因此，云备份中间件需要解决不同应用和多种云存储服务之间适配的问题，以支撑更多桌面OS组件使用云备份与云存储服务。

（2） 会话持久化管理

云备份中间件一期任务的解决方案虽然支持用户会话管理，但仍存在一些问题尚待优化：例如，在大量会话数据存在的情况下，会导致服务启动时间过长，而一旦服务突然死亡，可能会丢失会话数据。此外，会话信息以明文形式存储，用户的私密信息很容易被窃取。

（3） 本地缓存优化

云存储是存储和网络I/O密集的服务，大量的I/O会引起机器性能下降，也可能会干扰其它的应用。现代计算机通常有很大的本地存储空间，并且计算能力普遍过剩。因此，可以通过利用本地存储和计算能力，如优化本地缓存的利用，进一步减少I/O，提高服务性能。

（4） 系统优化

为了提高系统的灵活性，中间件具有高度模块化的特征，多个模块协同提供服务。各模块存在数据依赖关系，例如数据从一个模块流向另一个模块，而各模块完成任务的时长也不一样。在二期云备份中间件优化中需要合理组织各模块，以提高系统的效能。此外，云备份中间件应该充分挖掘服务器的功能，包括可用性和性能，为客户端提供更好的服务。

# 数据结构分析

## 访问令牌

访问令牌包含了登录会话需要的安全信息，其构造如图1所示，访问令牌包括令牌号，令牌类型，令牌有效期，刷新令牌参数等。



图1:访问令牌

## 文件属性

在文件上传和下载时需要获取文件属性，其构造如图2所示，文件属性包括，文件状态、文件大小、文件类型、MD5值、文件创建时间、文件权限、文件创建模式（加密、加密压缩、普通）、备注信息、元数据等。



图2:文件属性

## 会话信息

会话信息包含着云备份中间件在客户端和服务器间通信的各种数据，如下图3所示，会话信息包括请求号、请求内容、请求执行状态、请求时间、用户名、请求返回内容、备注信息等。



图3:会话信息

# 系统部署图



图4:系统部署图

云存储与云备份系统的技术架构如图4所示，分为云存储服务器，密钥管理服务器，身份认证服务器，云存储与云备份中间件，云存储与云备份客户端五大部分。其中，云存储与云备份客户端和云存储与云备份中间件被部署在每台终端设备上，终端设备通过网络与三大服务器进行交互。

# 系统结构图

图5:云存储与云备份中间件系统结构图

云存储与云备份中间件优化后的总体结构如图5所示，可以分成三部分：

* 云存储与云备份中间件的底层结构包括操作系统平台、本地库、内存与本地存储三部分，它提供了中间件可以正常运作的基础软硬件。
* 云存储与云备份中间件的功能层由七大核心模块组成，分别是交互适配模块、持久会话模块、缓存管理模块、安全会话模块、数据传输模块、数据加密模块，以及数据压缩模块。
* 云存储云备份中间件的接口层包括了为客户端提供的所有中间件服务，这些服务可以分成三大类，分别是信息处理类、文件操作类，和回收站操作类。

# 系统框架设计

中间件作为客户端和服务器之间的桥梁，减少了客户端和服务器之间适配的差异，降低了系统开发、升级和维护的难度。为了支持不同类型的客户端和更多的服务，我们需要设计更高效的云备份中间件。

从中间件适配的广泛性、服务质量、扩展性出发，我们对云备份中间件进行了重构，对多应用持久会话、交互适配和本地存储管理三个部分进行优化，优化的系统框架如图6所示，多应用持久会话管理，负责管理各种会话数据；交互适配管理，负责对各种客户端和服务器的请求进行交互适配，并提供扩展机制来保障适配；本地存储管理，则负责中间件的临时存储文件的管理。



图6:中间件优化架构示意图

# 核心功能介绍

## 交互适配管理

交互适配管理可以支持多种应用与服务之间的适配。

作为中间件，其主要的目标是完成不同客户端和服务之间的适配。由于设计时考虑的情形和目标不一样，客户端和服务端通常会定义不同的接口，导致服务器和客户端不匹配。

为了减少客户端和服务端之间的差异，可以引入中间件来对双方的调用方式进行转换，也称为适配。适配分为两种情况：1）提供一个标准的接口，客户端调用中间件给出的接口；2）作为一种桥梁，以较小的代价将不同的客户端和服务端整合在一起。

第一种情况适合于规范新开发的客户端，从用户使用的角度来统一定义一个访问接口。第二种情况，实际上是第一种情况的强化形式，需要中间件来设计更灵活的机制来支持更一般化的适配。

由于需求不一致，客户端和服务器端在会在功能上存在差异。通常而言，我们认为客户端使用的是服务器端所提供的功能子集。由此，我们把差异主要分为两类：1）服务器端完全不提供客户端需要的功能；2）客户端的功能可分解为服务端的若干子功能。

为了适配客户端发出的请求（输出形式）和服务端的接收请求（输入形式）之间的差异，系统要先定义一种描述元语言。基于该元语言，我们可以用来描述客户端和服务端的请求，从而实现不同请求之间的比较。对于给定的命令请求规范，构成了多个接口命令描述库 (Interface Command Describe Base)。

适配的最终目标是实现不同命令接口的转换，很多时候功能差异不能通过元描述语言来完成转换，需要引入外部的支持。因此，交互管理需要提供一个更一般的框架，可以支持引入外部的工具，包括命令和函数库等。

## 会话持久化管理

持久会话信息管理可以支持多用户、多应用会话信息高效安全地管理。

云备份中间件在客户端和服务器之间工作，多个客户端的命令通过中间件提交给服务器。在执行备份/恢复任务时，客户端可以批量提交任务给中间件。即使用户注销后，中间件仍要在后台继续执行任务。更进一步，机器在重启后，也需要中间件自动执行未完成的任务。因此，中间件需要持久化保存多个客户端的请求，以可靠地执行客户端提交的任务。

用户在访问存储服务器上的资源时，需要提供身份和授权信息，因此中间件也需要保存用户的相关私密信息。由于中间件要为多个用户提供服务，因此，如何安全地保管用户的信息并防止私密信息被窃取，变得至关重要。

为了防止用户信息的泄露，需要从以下几方面进行优化：

1）不同用户的数据，按照会话进行逻辑隔离；一个用户只能访问自己所属会话的数据。2）明文形式的秘密信息，只能作为会话数据在内存中存在。3）用户的持久私密信息以加密的形式存储，通过会话管理模块只能获取到加密形式的数据。

为了提高中间件的扩展性和效率，客户端的请求在中间件以本地数据库来存储，例如敏捷存储接口和服务。客户端可以查询提交的请求，如果请求没有执行，用户可以撤销该任务。

## 本地缓存管理

缓存管理用于管理本地存储以减少客户端I/O。

在文件上传和下载时，中间件需要先将文件存储在临时区域，该区域只能被中间件访问。通过分析请求和文件内容，可以进一步在客户端和服务器端上降低系统I/O。

由于客户端提供远程打开文件的功能，导致客户端可能会反复访问同一个文件，如果每次都从远程下载文件，则是对资源的浪费。因此，我们需要高效利用临时存储区域，以降低I/O。一般而言，可以考虑LRU的替换策略。

由于服务器端提供文件共享的功能，多个客户端可以请求相同的资源。对于相同URL的请求，如果目标存在，则中间件可直接返回临时存储区域中的内容。

同一个用户，备份数据时会有相同的文件；不同的用户之间，也可能存在相同的文件。

在上传文件时，不同用户间压缩内容可以复用，加密数据由于存在不同的密钥，所以不可以复用；同一用户的相同文件，加密后的数据也可选为相同，可利用服务器提供的复制功能，直接在服务器操作，减少计算和I/O。

在下载文件时，如果不同用户的文件数据相同，则可以避免将内容写入到临时存储区域。通过复用内存中的数据，直接写到用户指定路径，从而避免磁盘I/O。

## 系统优化

系统优化包括优化中间件，充分利用服务器的高级功能，以提高系统性能。

在第一期任务中，中间件实现了对图形化和CLI两种客户端的支持。我们发现，客户端通常只使用了服务端的一部分功能。通过利用服务端的一些高级功能，可以为客户端提供更好的体验。在这期任务中，中间件将通过引入服务器端的一些高级功能，改善客户端和服务器端的用户体验。

1）大文件传输

在很多情况下，客户端和云备份服务器分布在不同的地域，网络传输速度受各种条件制约。一般而言，一个文件如果传输不成功，下一次只能重传，这浪费了带宽和时间。事实上，服务器端支持分块传输的功能，也就是说客户端可以传输指定大小的文件数据。因此，中间件可以根据网络数据传输的质量，确定文件分块传输的大小，提高大文件的传输成功率。

2）并发I/O

现代计算机的处理器通常都有多个处理器或者多个处理核，多任务可以充分发挥系统性能。由于中间件需要对上传数据进行压缩和加密，因此其任务分为计算密集和I/O密集两类。我们要进一步考虑系统中不同类型任务的协调，以切实提高系统效率。

加密/压缩计算是计算密集的，而上传和下载则是I/O密集。由于现代处理器的计算性能很高，而磁盘访问和网络传输相对很慢。当用户的请求很多时，任务间会产生相互影响，从而降低效率。特别地，磁盘I/O往往是制约系统能力的重要因素。

由于文件备份和下载的流程互逆，我们以文件备份为例来说明。中间件读取文件，对文件进行压缩，然后加密，最后上传。在文件访问的过程中，操作系统会把文件内容缓存在系统中，直到内存达到一个阈值。由于上传的是最终形式的数据，中间形式的数据并不需要保存，而在当前的实现中，中间件总是将中间计算结果以文件的形式来保存，这大大浪费了内存，也有可能增大系统I/O。

考虑到上传和下载的连续性，我们可以利用多任务来进行优化。将一个任务分为多个阶段的子任务，构成流水线（Pipeline），以最大化复用内存，降低I/O。由于各个子任务可能会存在不均衡的情况，可进一步根据系统状况进行动态调节。

3）中间件的权限Delegation

在原有系统中，客户端提交任务如果长时间不活跃，则会导致大量任务的失败和重新提交，浪费了资源，用户体验也不够好。结合会话信息管理，我们可以进一步支持用户对中间件的授权，从而解决身份认证和授权信息过期所引发的问题。

为了保证授权不被滥用，获取的秘密信息只能用于用户授权前提交的任务。

# 类图设计

图7:云存储与云备份系统中间件UML类图

该图描述了云存储与云备份系统中间件的类结构。主要由安全会话管理模块、数据传输模块、数据加密模块、数据压缩模块构成、持久会话模块、交互适配模块以及缓存管理模块。当客户端交互模块接收到客户端传来的请求后，通过交互适配模块对用户请求进行适配或分解，并将适配或分解之后的请求发送给控制模块，控制模块首先将请求交给解析模块进行指令类别判断，然后根据解析模块返回的指令类别对以下模块进行调用。

* 安全会话管理模块

由令牌类组成，通过记录与令牌操作有关的日志，以帮助管理员在需要时采取防御措施。该类完成操作日志的记录后，调用服务器交互模块完成指令操作。此外，通过调用令牌转换方法，中间件可以获得与云存储服务器交互所需的信息。

* 数据传输模块

由数据上传、数据下载以及查询状态类组成。在数据上传类中首先需要获取待上传文件的属性以判断是否是大文件，如果上传的是大文件，则调用文件分片方法后再进行上传。在数据下载类中，如果指令中的参数含有加密和压缩信息，则下载文件后控制模块需要调用相应的解密和解压缩方法。查询状态类的功能是要针对已上传和已下载的数据量进行查询，并返回客户端传输的进度。

* 数据加密模块

由数据加密类组成。如果文件上传前需要进行文件加密操作，控制模块会调用数据加密类的加密方法对指定文件进行加密。如果下载文件后需要进行解密操作，控制模块会调用数据加密类的解密方法对指定文件进行解密。

* 数据压缩模块

由数据压缩交互类和数据压缩类组成。如果需要进行文件的压缩以及解压缩操作，控制模块会调用数据压缩交互类的方法，数据压缩交互类通过识别布尔值调用数据压缩类的压缩或者解压缩方法。

* 持久会话模块

由会话备份、会话恢复、会话加密、会话逻辑隔离、会话任务查询和会话任务撤销组成。中间件在启动时自动检测数据库中是否有未完成的任务，并恢复和执行未完成的任务。当会话任务由客户端传入到中间件时，首先对会话进行加密并备份到数据库中，中间件按照会话进行逻辑隔离，确保一个用户只能访问自己所属会话的数据。此外，持久会话模块提供给用户查询会话任务和撤销会话任务的功能。

* 交互适配模块

由接口适配和功能分解组成。不同的客户端向服务器发送请求时，可能与服务器接口不匹配。交互适配模块将来自不同客户端的请求转化为与服务器接口相匹配的标准格式。此外，当客户端的请求可分解为服务端的若干子请求时，交互适配模块先将功能分解之后再交给控制模块逐一执行。

* 缓存管理模块

由临时文件存储、临时文件访问以及临时文件替换组成。客户端反复访问的文件将被保存在临时区域，不同用户或不同客户端访问相同文件时，也可以从临时区域获取，无需从服务器远程下载。同时，加密和压缩的文件也能够复用以减少计算和I/O。

# 流程设计

描述引出

## 系统总体流程设计

图8:云存储与云备份系统中间件总体流程图

客户端向中间件发出指令后，客户端交互模块将指令和参数交给交互适配模块，通过交互适配模块对用户请求进行适配。如果客户端请求的功能可以分解为服务器端的若干个子功能，交互适配模块首先将功能分解然后再进行适配，将适配和分解之后的请求发送给控制模块，并由控制模块调用解析器进行指令解析。如果是与安全会话有关的指令，控制模块调用安全会话模块。如果是与身份认证有关的指令，安全会话模块调用服务器交互模块，再由服务器交互模块调用身份认证服务器API得到返回结果；如果是与令牌转换有关的指令，安全会话模块调用动态链接库获得返回结果。如果是与数据压缩有关的指令，那么控制模块调用数据压缩模块进行处理。如果是与数据加密有关的指令，那么控制模块调用数据加密模块进行处理。如果是与数据传输有关的指令，控制模块调用数据传输模块：数据传输模块根据得到的指令，若判断需要查询状态，则返回当前传输状态；若判断是上传或者是下载操作，首先查看缓存管理模块中的临时文件区域是否存在上传或者下载的文件，如果存在，则用临时区域的文件替代用户请求的文件。若判断是文件上传操作，则根据文件的大小判断是否需要进行分片，然后进行文件上传操作；若判断是文件下载操作，则调用服务器API下载进行指定文件的下载。若判断是与会话任务相关的操作，控制模块调用持久会话模块，若判断需要查看当前任务，则返回当前任务状态；若判断需要撤销已经提交的任务，则撤销任务并返回结果。如果是其他指令，那么控制模块将指令直接交给服务器交互模块，由服务器交互模块调用云存储服务器API获得返回结果。最后，客户端交互模块再将结果返回给客户端。

## 交互适配管理流程设计

描述引出，这里重点不在于设计那两个插件，那两个插件只不过是我们测试用的客户端，重点是设计一个统一接口来实现交互适配管理。所以，我个人认为不需要写插件的设计。

### 命令接口转换设计



## 会话持久化管理流程设计

描述引出

### 会话持久化流程设计



### 会话加密管理流程设计



## 本地缓存管理流程设计

### 数据上传缓存管理流程设计



### 数据下载缓存管理流程设计



## 系统优化流程设计

### 大文件上传流程设计



### 大文件下载流程设计



### 并发任务执行流程设计



# 接口函数设计

## 中间件接口

云存储与云备份系统客户端和中间件部署在客户端；客户端与服务器端的通信通过中间件进行。中间件以守护进程的模式工作，在指定的端口号 10000接收客户端的请求。

客户端请求中间件服务的流程如下：1）客户端将用户的操作转化为对应的指令，指令的格式参照10.2节中的描述；2）将指令封装为报文后，发送给中间件监听的端口；3）中间件接收到指令报文后，对指令进行解析，并进行相应的会话备份、文件加密、压缩、文件分片等处理；4）中间件执行命令，将适配处理后的指令发送给服务器，根据服务器返回的结果，把客户端请求指令的唯一uuid和给客户端的返回结果作为response（字符串格式的json数据）返回给客户端，方便客户端做出对应的指令响应。

接口如下表所示：

|  |  |  |
| --- | --- | --- |
| 接口编号 | 接口名称 | 中间件接口 |
| MW-IF-1 | 接口定义 | 中间件在设定端口（10000）提供服务 |
| 传入参数 | 请求报文 //不同请求对应不同字符串，具体格式见10.2节功能列表中传入参数定义. |
| 返回结果 | 返回结果是指令请求的唯一uuid和服务器响应的json数据：  uuid = 客户端请求的md5值  mwresult = 服务器的响应结果 |
| 通信 | 网络连接（socket） |
| 传入参数示例  （上传文件） | userid = kaeyika163com  time = 1469004031.3127  request = '-k -X PUT -T /home/sari/mwtest/1.JPG "https://192.168.7.62:443/v1/AUTH\_kaeyika163com/con1/ec2.JPG?op=CREATE&overwrite=true&metadata=\{"ln":4,"imp":"no"\}&mode=ENCRYPT\_COMPRESS" -H "X-Auth-Token: jZNLlcC7mlupK2vBikw90IhenY903hmgvgprgoKP"' |
| 返回结果示例  （上传文件） | 示例：  uuid = 3e14e3a96dc84c1292bcc7a383c5f61a  mwresult = {"status": "0", "ctime": "1443594726.64746", "X-Object-Permisson": "700", "mtime": "1443594726.64746", "msg": "", "path": "/con1/ec2.JPG", "md5": "21763cd504d6234573794903644a1405", "size": "5447680"}  其中，各参数的含义如下：  "uuid":"客户端指令的md5值",  "status": "指令执行状态，0表示执行成功",  "path":"该文件的用户空间相对路径",  "size":"文件字节大小",  "ctime":"文件创建时间",  "mtime":"文件修改时间",  "X-Object-Permisson": "文件权限值",  "md5":"被上传文件的md5签名"  } |
| 说明 | 1. 中间件接口会将客户端请求的md5值作为唯一的uuid与服务器响应结果一起返回给客户端，需要客户端在请求时自行记录相应请求的md5值，以便标记对应指令的返回结果。 2. 10.2章节定义了传入参数的格式，也可参考以下文档： |

## 中间件接口功能列表

下表所示是中间件接口功能列表,对于中间件接口各功能的传入参数进行了定义。

注：userid与Tenant\_id相同，即用户id，传入参数中userid应是具体的用户id值；

time参数给出了具体获取方法，指系统时间；

request参数中IP:Port对应服务器的ip地址和端口，Tenant\_id与userid相同，具体可详见10.1中的示例。

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 传入参数定义 | 返回数据 | 参数说明 |
| (1)  验证token | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST "https://IP:Port/v1/verify\_token" -H "X-Auth-Token: {Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": "",  "access\_token": "token值",  "verify\_flag": "true",  "email": [邮箱](mailto:gouqi1@163.com)  } | Token\_id：租户对应的token的id |
| (2)  用户注册初始化 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT -H "X-Auth-Token:{Token\_id}" https://IP:Port/v1/AUTH\_{Tenant\_id}/register' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0"  "Errors": [],  "msg": ""  "Number successed": 14,  "Number failed": “0”  } | Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (3)  获取配额信息 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X GET "https://IP:Port/v1/AUTH\_{Tenant\_id}/quota?op=info" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "X-Container-Count": 20,  "X-PUT-Timestamp": 1,  "X-Account-Meta-Quota-Bytes": "2000000000",  "X-Account-Bytes-Used": 1435086605,  "status": "0",  "X-Account-Container-Count": 20,  "X-Account-Object-Count": 0,  "X-Bytes-Used": 1435086605,  "X-Type": "Account",  "msg": "",  "X-Timestamp": "1439555730.91294",  "X-Object-Count": 0  } | op：固定值info  Tenant\_id：租户id  Token\_id：租户对应token的id |
| (4)  查询文件传输状态 | userid = userid  time = '%.4f' %time.time()  request = 'get\_transfer\_status' | uuid = 客户端请求的md5值  mwresult =  {  "TotalPercent": "当前下载文件的百分比",  "Current Speed":"当前下载文件的速度",  "Total":"下载文件的大小",  "Received":"接收到文件的大小",  "AverageDload":"平均下载速度",  "SpeedUpload":"平均上传速度",  "Time Spent":"总的时间花费",  "Time Left":"下载剩余时间"  } | 1. 查询文件上传、下载的状态信息 2. 此功能是中间件提供给客户端的，不涉及服务器，在上传或者下载过程中，客户端需要获取到文件传输进度。 |
| (5)  增加存储空间配额 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST "https://IP:Port/v1/AUTH\_{Tenant\_id}/quota?op=createstorage" -H "X-Auth-Token: { Token\_id}" -H "X-Account-Meta-Quota-Bytes: {quota\_byte}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | quota\_byte: 增加空间大小（字节数）  Tenant\_id：租户id  Token\_id：租户对应的token的id  注：quota数值为示例，具体空间待定。  此接口在应用程序界面上当发生新用户注册完成事件时调用，用于通知云端创建默认配额。 |
| (6)  上传文件 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT -T {LOCAL\_FILE} "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=CREATE\ [&overwrite=<true|false>][&metadata=<STRING>] \ [&mode=<NORMAL|ENCRYPT|COMPRESS|ENCRYPT\_COMPRESS>]" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "X-Object-Permisson": "\*\*\*权限",  "path":"string 该文件的用户空间相对路径",  "size":"uint64 文件字节大小",  "mtime":"uint64文件修改时间",  "ctime":"uint64文件创建时间",  "md5":"string被上传文件的md5签名",  "msg": "　string 返回消息",  } | LOCAL\_FILE:上传的文件  Container:容器名称  PATH：上传至路径  overwrite：true覆盖名称相同的文件;false不覆盖名称相同的文件  metadata:用户自定义元数据  mode：指定用户上传文件时的属性。其中，NORMAL默认值-原文传输,  ENCTYPE加密传输,  COMPRESS压缩传输, ENCRYPT\_COMPRESS加密压缩传输  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (7)  上传临时文件（支持分片上传功能） | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT -T <LOCAL\_FILE> "https://IP:Port/v1/AUTH\_{Tenant\_id}/<segments\_path>?op=CREATE  [&overwrite=<true|false>][&metadata=<STRING>][&mode=<NORMAL|ENCRYPT|COMPRESS|ENCRYPT\_COMPRESS>]" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "X-Object-Permisson": "\*\*\* 权限",  "status": "0"，  "path":"string该文件的用户空间相对路径",  "size":"uint64文件字节大小",  "mtime":"uint64文件修改时间",  "ctime":"uint64文件创建时间",  "md5":"string被上传文件的md5签名",  "msg": "string 返回消息",  } | LOCAL\_FILE:上传的文件  segments\_path：上传至segments容器下的路径  Overwrite：true覆盖名称相同的文件;false不覆盖名称相同的文件  metadata:用户自定义元数据  mode：指定用户上传文件时的属性, NORMAL默认值-原文传输, ENCTYPE加密传输, COMPRESS压缩传输, ENCRYPT\_COMPRESS加密压缩传输(该参数只有使用云存储SDK时起作用；只用云存储Restful API该参数的固定值为NORMAL)  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (8)  合并文件(支持分片上传功能) | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT -d  '[  {  "path": "文件分片路径",  "etag":"文件分片md5值",  "size\_bytes":文件分片大小  },  {  "path":"文件分片路径",  "etag":"文件分片md5值",  "size\_bytes": 文件分片大小  }  ]'  "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?multipart-manifest=put[&overwrite=<true|false>]" -H "X-Auth-Token: { Token\_id}" -H "X-Static-Large-Object: true"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0"，  "X-Object-Permisson": "\*\*\* 权限",  "path":" string 该文件的用户空间相对路径",  "size":"uint64文件字节大小",  "mtime":"uint64文件修改时间",  "ctime":" uint64文件创建时间",  "md5":" string被上传文件的md5签名"  "msg": "　string 返回消息",  } | container:容器名称  PATH: 合并文件的目标路径  overwrite：true覆盖名称相同的文件;false不覆盖名称相同的文件  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (9)  创建目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=MKDIRS" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | container:容器名称  PATH：目录路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (10)  删除文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X DELETE "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=DELETE[&ftype=<f|d>][&cover=<false|true>]" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | PATH：目录路径  Container:容器名称  PATH：对象路径  ftype:对象类型；f表示文件，d表示目录  cover：默认值为false，历史文件不自动覆盖;可选true,用历史文件覆盖删除的文件  Tenant\_id：租户id  Token\_id：租户对应的token的id  注: 删除的文件会生成一个唯一的标识uuid，用于说明相同文件名文件的删除。uuid被作为元数据信息进行存储 |
| (11)  批量删除文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST -d '{  "list":[{  "path":"/{container}/{obj\_path}",  "ftype":"<f|d>"  },  {  " path ":"/{container}/{obj\_path}",  "ftype":"<f|d>"  }]}'  "https://IP:Port/v1/AUTH\_{Tenant\_id}/batch?op=DELETE" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | container:容器名称  Obj\_path:对象路径  ftype:对象类型；f表示文件，d表示目录  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (12)  移动文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{src\_path}?op=MOVE[&ftype=<f|d>]" -H "X-Auth-Token: { Token\_id}"  -H "Destination:/{container}/{dst\_path}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | container:容器名称、  src\_path:源对象路径  dst\_path：目标对象路径  ftype：对象类型；f表示文件，d表示目录  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (13)  批量移动文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST -d  '{  "list":[  {  "from":"/{container}/{obj\_path}",  "to":"/{container}/{obj\_path}",  "ftype":"<f|d>"  },  {  "from":"/{container}/{obj\_path}",  "to":"/{container}/{obj\_path}",  "ftype":"<f|d>"  }  ]  }'  "https://IP:Port/v1/AUTH\_{Tenant\_id}/batch?op=MOVE" -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | form:源对象路径（Container:容器名称，Obj\_path:对象路径）  To ：目标对象路径  ftype：对象类型；f表示文件，d表示目录  Tenant\_id：租户id  Token\_id：租户对应的token的id  注：此处的batch不是path，是命令固定值 |
| (14)  复制文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{src\_path}?op=COPY[&ftype=<f|d>][&async=true]"  -H "X-Auth-Token: { Token\_id}"  -H "Destination:/{container}/ {dst\_path}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  }  异步复制：  {  "status":"0",  "msg": "tx3b4623ad80954aa4a7df2e6613d811f9"  } | Container：容器名称  src\_path：复制源路径  dst\_path: 复制目的路径  Tenant\_id：租户id  async：true为异步复制方式，false为同步复制方式  ftype：对象类型；f表示文件，d表示目录  Token\_id：租户对应的token的id |
| (15)  批量复制文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST -d  '{  "list":[  {  "from":"/{container}/{obj\_path}",  "to":"/{container}/{obj\_path}",  "ftype":"<f|d>"  },  {  "from":"/{container}/{obj\_path}",  "to":"/{container}/{obj\_path}",  "ftype":"<f|d>"  }  ]  }'  "https://IP:Port/v1/AUTH\_{Tenant\_id}/batch?op=COPY"  -H "X-Auth-Token: { Token\_id}"' | uuid = 客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | From：源对象路径（Container:容器名称，Obj\_path:对象路径）  To：目标对象路径  Container:容器名称  Obj\_path:对象路径  ftype:对象类型；f表示文件，d表示目录  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (16)  获取文件 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -L "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{path}?op=OPEN[&offset=<LONG>][&length=<LONG>][&version=<LATEST|指定版本>][&mode=<NORMAL|ENCRYPT|COMPRESS|ENCRYPT\_COMPRESS>]"  -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult = [dest\_path] download success!  同时文件下载到指定路径  注：下载文件的前提条件是客户端需要先调用(22)获得待下载文件属性，即获得mode参数后，客户端才能发出指令进行下载文件操作。 | Container：容器名称  PATH：文件下载的源路径  dest\_path: 文件下载的目标路径  offset：从第几个字节开始读取  length：读取文件内容长度(字节)  Tenant\_id：租户id  Token\_id：租户对应的token的id  version：文件版本(表示最新文件 | 历史文件版本号)  mode:指定上传文件时的属性, NORMAL默认值-原文传输, ENCTYPE加密传输, COMPRESS压缩传输, ENCRYPT\_COMPRESS加密压缩传输(该参数只有使用云存储SDK时起作用；只用云存储Restful API该参数的固定值为NORMAL)  注：1. mode参数是必选的，四方第一次会议时提到文件传输前必须加密。即必须指明ENCRYPT | ENCRYPT\_COMPRESS  2.mode中的参数需要由客户端先调用(22)获取文件属性来获得。 |
| (17)  获取文件历史 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -L "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{path}?op=GETHISTORY"  -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  “status”:”0”,  ”msg”:” ”,  ”info”:  [  {  "ftype": "f",  "name": "Jimmy.txt/1432724740.39658",  "modificationTime": "1432724740.41272"  "metadata": "somethingnew"  "md5": "ddd9892de1df7b3e749cd48e2948f787",  "status": "0",  "ctime": "1439565709.55347",  "X-Object-Permisson": "700",  "mtime": "1439565709.55347",  "msg": "", "path": "/\u5907\u4efd/seg\_test",  "size": "393"  },  ]  } | Container：容器名称  PATH：文件路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (18)  获取用户操作历史 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -L "https://IP:Port/v1/AUTH\_{Tenant\_id}?op=GET\_OP\_HISTORY[&recent=<INT>]"  -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": "",  "info": [["seg\_test", "object", "GET", "AUTH\_lisiqqcom", "1439890176.36", "", "get file versions"]]  } | Recent：获取用户最近多少天的操作历史，默认值为0，表示获取用户全部操作历史，recent=1表示获取最近一天操作历史，recent=2表示获取最近两天操作历史，以此类推  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (19)  清除用户操作历史 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -L "https://IP:Port/v1/AUTH\_{Tenant\_id}?op=DELETE\_HISTORY[&recent=<INT>]"  -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Recent：清除用户最近多少天的操作历史，默认值为0，表示清除用户全部操作历史，recent=1表示删除最近一天操作历史，recent=2表示删除最近两天操作历史，以此类推  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (20)  创建符号链接 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=CREATESYMLINK&destination=/{container}/{PATH}"  -H "X-Auth-Token: {Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Container：容器名称  PATH：源和目标路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (21)  重命名文件、目录 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=RENAME&destination=/{container}/{PATH}[&ftype=<f|d>]"  -H "X-Auth-Token: {Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Container：容器名称  ftype:对象类型；f表示文件，d表示目录  PATH：源和目标路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (22)  获取文件属性 | userid = userid  time = '%.4f' %time.time()  request = '-k -i "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=GETFILEATTR[&version=<LATEST|指定版本>]"  -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "Content-Length": "200",  "X-File-Type": "f",  "ETag": "7add118c851212736b362105e6235b84",  "X-Timestamp": "1438679161.39779",  "X-Object-Permisson": "500",  "msg":"",  "mode": "COMPRESS"  "metadata": "size:4,clolor:red"  } | Container：容器名称  version : 文件版本，默认值为LATEST，表示当前版本；指定版本为可选的历史版本  PATH: 目标路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (23)  获取回收站中的文件列表 | userid = userid  time = '%.4f' %time.time()  request = '-k -i "https://IP:Port/v1/AUTH\_{Tenant\_id}/recycle/user?op=GETRECYCLER[&start=<LONG>][&limit=<LONG>] " -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": "",  "info":  [  {  "hash": "d34e3d609811b44e56e07e04c1fe7ec3",  "uuid": "vfgUc2hO-OkmCeg-l4OC",  "bytes": 116007,  "ftype": "f",  "time": "1432653385.36927",  "path": "/normal/myDir2.log",  "name": "vfgUc2hO-OkmCeg-l4OC"  },  {  "hash": "dir",  "uuid": "wXQUs5DY-i0lrOH-2Bh8",  "bytes": 0,  "ftype": "d",  "time": "1432653333.77452",  "path": "/normal/myDir1",  "name": "wXQUs5DY-i0lrOH-2Bh8"  }  ]  } | Start:获取回收站文件的起始条目( 默认为0, 从第一条数据开始读取 )  Limit:获取回收站文件的文件数量 (默认为0,读取全部数据)  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (24)  从回收站恢复 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST -d  '{  "list":[  {  "uuid":"uuid值",  "path":"/{container}/{obj\_path}",  "ftype":"<f|d>"  },  {  "uuid":"uuid值",  "path":"/{container}/{obj\_path}",  "ftype":"<f|d>"  }  ]  }'  "https://IP:Port/v1/AUTH\_{Tenant\_id}/batch?op=MOVERECYCLE" -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Uuid值：获取回收站文件裂变的返回值中的uuid字段值  Path：恢复的目标路径（Container:容器名称，Obj\_path:对象路径）  ftype：对象类型；f表示文件，d表示目录  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (25)  清空回收站 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST  "https://IP:Port/v1/AUTH\_{Tenant\_id}/clearrecycle?op=RECYCLER" -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (26)  设置权限 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X PUT "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=SETPERMISSION[&permission=<OCTAL>]" -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Container：容器名称  permission:设置权限值  PATH: 目标路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (27)  获取用户文件列表 | userid = userid  time = '%.4f' %time.time()  request = '-k -i "https://IP:Port/v1/AUTH\_{Tenant\_id}/{container}/{PATH}?op=LISTDIR&ftype=d[&recursive=<true|false>]" -H "X-Auth-Token: { Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": "",  "info":  [  {  "ftype": "f",  "name": "vt.log",  "modificationTime": "1432656458.06808"  "bytes": 116007,  "metadata": "'size':6,'color':'red'",  "md5": "d34e3d609811b44e56e07e04c1fe7ec3",  },  {  "ftype": "d",  "name": "myDir1",  "modificationTime": "1432661311.58144",  "bytes": 0,  "md5": "d41d8cd98f00b204e9800998ecf8427e"  }  ] | Container:容器名称  recursive:为true时显示当前目录以及子目录下的对象，为false时列出当前目录下的首层对象  PATH: 目标路径  Tenant\_id：租户id  Token\_id：租户对应的token的id |
| (28)  设置文件历史存储容器 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X POST https://IP:Port/v1/AUTH\_{Tenant\_id}/{Container\_name} -H "X-Auth-Token: {Token\_id}" -H "X-Versions-Location: {Container\_name}\_versions"' | uuid =客户端请求的md5值  mwresult =  {  "status": "0",  "msg": ""  } | Tenant\_id：租户id  Container\_name:自定义的已存在的容器名称（两个container\_name的值必须一致）  Token\_id：租户对应的token的id |
| (29)  查看设置文件历史容器 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X HEAD https://IP:Port/v1/AUTH\_{Tenant\_id}/{Container\_name} -H "X-Auth-Token: {Token\_id}"' | uuid =客户端请求的md5值  mwresult =  {  "status":"0"  "msg":""  "info":  {  'X-Container-Bytes-Used': '',  'X-Timestamp': '1',  'X-Container-Object-Count': '',  'X-Versions-Location': 'Code\_versions',  'X-PUT-Timestamp': '1439556237.77993'  }  } | Tenant\_id：租户id  Container\_name:自定义的已存在的容器的名称  Token\_id：租户对应的token的id |
| (30)  查询异步复制文件任务执行结果 | userid = userid  time = '%.4f' %time.time()  request = '-k -i -X GET "https://IP:Port/v1/AUTH\_{Tenant\_id}?op=GET\_OP\_TASK&tx\_id={task\_id}" -H "X-Auth-Token: {Token\_id}"' | uuid =客户端请求的md5值  mwresult =  复制中:  [["1441008734.92", "running", ""]]  复制成功:  [["1441008734.92", "success", ""]] | Tenant\_id：租户id  task\_id:异步复制文件的任务id(从异步复制文件的返回值中msg字段的值)  Token\_id：租户对应的token的id |