

鼎甲迪备

兼容 S3 协议的对象存储 EOBS 用户指南

Release V8.0-9

June, 2025



目录

1	对象存储 EOBS 介绍	1
2	计划和准备	3
3	存储服务器部署	5
3.1	环境要求	5
3.2	安装部署	5
4	快速入手指南	7
4.1	创建存储桶	7
4.1.1	【对象存储 EOBS】创建存储桶	7
4.1.2	【新建存储池】创建存储桶	8
4.2	查看存储桶	9
4.3	创建访问密钥	10
4.4	访问存储桶	11
5	控制台基本操作	13
5.1	存储桶管理	13
5.1.1	存储桶列表	13
5.1.2	创建存储桶	13
5.1.3	删除存储桶	13
5.2	对象管理	14
5.2.1	查看对象	14
5.2.2	删除对象	14
5.2.3	搜索对象	15
5.3	Access Key 管理	16
5.3.1	创建 Access Key	16
5.3.2	禁用 Access Key	16
5.3.3	删除 Access Key	17
6	API 接口	19
6.1	API 概览	19
6.2	Service 操作	20
6.2.1	ListBuckets(ListService) (获取存储桶列表)	20
6.3	Bucket 操作	21
6.3.1	HeadBucket (查询存储桶元数据)	21
6.3.2	ListObjects(ListBucket) (获取对象列表)	21
6.3.3	ListObjectsV2 (获取对象列表)	22
6.4	Object 基础操作	23
6.4.1	PutObject (简单上传对象)	23
6.4.2	GetObject (下载对象)	24
6.4.3	DeleteObject (删除对象)	25
6.4.4	DeleteObjects (删除多个对象)	26
6.4.5	AppendObject (追加上传对象)	27

6.4.6	HeadObject 获取对象元数据	28
6.5	Object 分片上传	29
6.5.1	InitMultipartUploads (创建分片上传对象)	29
6.5.2	ListMultipartUploads (列表分片上传对象)	29
6.5.3	UploadPart (上传分片)	31
6.5.4	ListParts (列表已经上传的分片)	31
6.5.5	AbortMultipartUpload (终止分片上传任务)	32
6.5.6	CompleteMultipartUpload (合并分片)	33
7	功能限制	35
8	术语表	37

1 对象存储 EOBS 介绍

鼎甲科技迪备基于现有的分布式重删存储架构，创新研发了兼容 S3 协议的对象存储服务 EOBS (Extended Object Storage)，通过提供与 S3 一致的 RESTful API 接口，任何支持 S3 协议的备份工具或应用程序，都能够无缝地与存储服务器进行数据交互，满足企业对大规模、高可用和安全存储的需求。

迪备对象存储 EOBS 主要特性包括：

- 平台兼容
存储服务节点支持多种处理器架构，包括英特尔、飞腾、鲲鹏、海光、龙芯等。
- API 接口
 - 存储桶管理：支持查询、删除桶等多种操作。
 - 对象管理：支持上传、下载、追加、删除对象等多种操作。
 - 分片上传：支持创建、上传、合并分片等多种操作。
- 存储桶读写权限
存储桶权限支持私有、公共读、公共读写。
- 存储池
支持基于重删存储池或分布式重删池创建存储桶。
- 签名认证
支持 AWS V2/V4、阿里云 OSS、华为 OBS。
- 数据处理
支持压缩、重删、加密存储。
- 访问方式
迪备控制台、API、AWS SDK、OSS SDK 等。

1. 部署和配置备份服务器

已成功安装和配置备份服务器，具体操作参考《服务端安装用户指南》的备份服务器安装章节。

2. 申请高级功能许可证

申请【重复数据删除】和【对象存储 EOBS】高级功能许可证。

迪备支持对单节点或分布式的重删存储池开启对象存储 EOBS 服务，因此存储集群的所有节点需要参考《分布式重删功能用户指南》完成系统配置和安装部署。

3.1 环境要求

存储集群的所有节点参考《分布式重删功能用户指南》的环境要求章节，完成下列配置：

1. 检查硬件指标。
2. 配置系统环境，包括操作系统、文件系统、网络规划。

3.2 安装部署

存储集群的各节点参考《分布式重删功能用户指南》安装部署章节，完成下列部署：

1. 分布式重删集群各节点的配置时钟同步。
2. 分布式重删集群各节点安装存储服务器模块。
3. 注册存储服务器。
4. 配置网络。
5. 创建重删存储池。

迪备用户需在控制台完成下列基本步骤，即可使用 EOBS 存储桶。

1. 创建存储桶。
2. 查看存储桶。
3. 创建访问密钥（Access Key）。
4. 访问存储桶。

4.1 创建存储桶

存储桶是对象存放的容器。在上传对象之前，您需要在迪备控制台基于重删存储池创建 EOBS 存储桶。

在迪备控制台仅管理员角色的用户可创建存储桶，用户可在【对象存储 EOBS】或【新建存储池】页面创建存储桶。两者的适用场景：

- 【对象存储 EOBS】页面：针对已成功创建的重删存储池，创建 EOBS 存储桶。
- 【新建存储池】页面：在创建存储池的同时，创建读写权限为“私有”的 EOBS 存储桶。

4.1.1 【对象存储 EOBS】创建存储桶

操作步骤如下：

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】，进入【对象存储 EOBS】页面。
2. 点击【新建存储桶】按钮，弹出【新建存储桶】窗口。

新建存储桶

1

2

基本信息

存储池

名称

必填

读写权限

☒ 私有

☐ 公共读

☐ 公共读写

对桶内对象的所有访问操作需要进行 Access Key 身份验证。

- 输入存储桶名称
名称必须唯一，仅支持长度为 3~63 的小写字母、数字和连字符 (-)，但不能以 ds、d2、连字符开头，或以连字符结尾。
- 选择桶的读写权限，默认选择“私有”权限。
 - 私有：必须进行 Access Key/Secret Key 身份验证，才允许对桶内的对象进行所有访问操作。

- 公共读：在不用 Access Key/Secret Key 身份验证直接读桶内的对象，安全风险高。不推荐此配置，建议您选择私有。
- 公共读写：在不用 Access Key/Secret Key 身份验证的情况下直接读/写/删桶内的对象，安全风险高。不推荐此配置，建议您选择私有。
- 点击【下一步】按钮，选择重删存储池，作为 EOBS 存储桶中的对象实际所在的存储池。
- 点击【提交】。

备注：存储桶创建成功后，用户权限与存储池相同。只有指定用户组下的用户才有权限使用该存储池及其 EOBS 存储桶。

4.1.2 【新建存储池】创建存储桶

操作步骤如下：

1. 在菜单栏中，点击【存储】->【存储池】，进入【存储池】页面。
2. 点击【添加】按钮，弹出【创建存储池】页面。
 - 选择存储池类型、添加存储服务器、选择网络。参考《分布式重删功能用户指南》存储池章节。
 - 勾选【S3 协议】，显示【EOBS 存储桶】设置项。
 - 【EOBS 存储桶】输入一个或多个存储桶名称。
名称必须唯一，仅支持长度为 3~63 的小写字母、数字和连字符 (-)，但不能以 ds、d2、连字符开头，或以连字符结尾。
3. 点击下一步，参考《分布式重删功能用户指南》存储池章节，设置存储池的保留策略、预测容量、加密等选项。
4. 点击下一步，输入存储池的自定义名称，并关联用户组，即完成重删存储池的创建。具体选项说明如下：
 - 名称：自定义。
 - 用户组：只有指定用户组下的用户才有权限使用该存储池及其 EOBS 存储桶。

1
选择

 —

2
设置

 —

3
提交

类型

重删存储池

存储服务器

ubuntu

已用 3.54 GiB, 可用 43.22 GiB, 总共 46.76 GiB

集群私网

块大小

自适应

S3 协议

☒ ?

EOBS 存储桶

s3-1

?

请输入存储桶名称

+

4.2 查看存储桶

存储桶创建成功后，迪备用户权限与其所属存储池的权限相同。如果迪备用户拥有存储池的权限，则有查看和使用 EOBS 存储桶的权限。操作步骤如下：

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【存储桶】。
2. 点击存储桶名称，进入【概览】页面查看存储桶信息，包括基本信息和访问信息。
 - 基本信息
包含存储桶名称、创建时间、所属存储池、读写权限。
 - 访问信息
包含存储桶的访问协议和终端节点。

概览

对象列表

基本信息

存储桶名称test1

创建时间2024-06-22 10:09:56

存储池test1

读写权限私有

访问信息

协议	终端节点
HTTP	192.168.19.50:50306
HTTPS	192.168.19.50:60306

4.3 创建访问密钥

Access Key 和 Secret Key 是访问对象存储 EOBS API 的密钥，相当于常用的用户名和密码。

迪备用户在使用 EOBS 存储桶之前，需创建访问密钥 Access Key，否则无法访问。操作步骤如下：

- 在菜单栏中，点击【存储】->【对象存储 EOBS】->【Access Key】，进入【Access Key】页面。
- 在【Access Key】页面点击【新建 Access Key】按钮。
为降低 Access Key 泄露风险，仅在 Access Key 创建时提供 Secret Key，之后将无法再查询，请妥善保存。
- 点击【复制】或【下载 CSV 文件】，保存新建 Access Key 及其 Secret Key。

新建 Access Key

为降低密钥泄露风险，仅在 Access Key 创建时提供 Secret Key，后续不可再进行查询。
请下载或复制进行保存。

Access Key

0jcp4FhRDJxHT7r0

Secret Key

下载 CSV 文件

复制

☐ 我已知晓并保存 Secret Key

关闭

4. 支持自定义 Access Key 使用。在【Access Key】页面点击【自定义 Access Key】按钮。

自定义 Access Key

为降低密钥泄露风险，Access Key 创建成功后，不可再查询 Secret Key，请妥善保存。

Access Key

Secret Key

10

4. 快速入手指南

备注:

- 如 Secret Key 丢失，可删除该密钥并创建新的 Access Key。
- 为避免影响存储桶访问，请至少保留一个 Access Key。

4.4 访问存储桶

根据[查看存储桶](#)访问信息中可获取协议和终端节点，终端节点默认为存储桶所在存储服务器的 IP。如果是分布式重删池，则是主节点的 IP。

- 默认 IP 访问

EOBS 存储桶支持路径（Path）请求风格和虚拟主机（Virtual Hosted）请求风格。如 HTTPS 协议访问结构如下：

- * 路径风格格式为：`https://ip:60306/<Bucket>`
- * 虚拟主机风格格式为：`https://<Bucket>.ip:60306`

- 自定义域名访问

迪备配置文件支持自定义访问域名。域名配置方式如下：

1. 进入存储池所在的存储服务器的机器终端。
2. 打开 `/etc/opt/scutech/dbackup3/storaged/http.d/base.conf` 配置文件。
3. 为"uri": "/" 的 params 中，增加 `-d <domain.com>` 参数。其中 domain.com 为自定义域名。如自定义域名为 eobs.com：

```
{
    "uri": "/",
    "dll": "s3gw",
    "init": "_make_S3_Service_Handler_Creator",
    "params": "-f /etc/opt/scutech/dbackup3/storaged/s3gw-backend.d -b dedup -d
↵eobs.com "
}
```

4. 在访问 eobs 的客户端上修改 hosts 文件，配置静态解析。

- * 路径风格格式为：`https:// domain.com:60306/<Bucket>`
- * 虚拟主机风格格式为：`https://<Bucket>.domain.com:60306`

如将存储服务器的 IP（格式为 xxx.xxx.xxx.xxx）解析为 eobs.com：

```
xxx.xxx.xxx.xxx eobs.com

xxx.xxx.xxx.xxx Bucket.eobs.com
```


在迪备控制台对象存储 EOBS 基本操作，包括存储桶管理、对象管理、Access Key 管理。

5.1 存储桶管理

5.1.1 存储桶列表

- 1. 在菜单栏中，点击【存储】->【对象存储 EOBS】，进入【对象存储 EOBS】页面。
- 2. 显示存储桶列表，包括存储桶名称、存储池、读写权限、创建时间和操作。

存储桶

Access Key

新建存储桶

名称

存储池

读写权限

创建时间

操作

op1

op

私有

2024-06-26 15:40:51

test3

123

公共读

2024-06-26 15:38:43

123

123

私有

2024-06-26 15:38:27

test2

test1

公共读写

2024-06-22 10:11:47

test1

test1

私有

2024-06-22 10:09:56

存储桶是存储池存放对象的容器。当前用户如需使用 EOBS 存储桶，请先创建对象存储 EOBS 的 Access Key，以免影响存储桶的访问。

5.1.2 创建存储桶

仅具有管理权限的用户允许创建存储桶，其他角色的用户不支持。具体参考[创建存储桶](#)。

5.1.3 删除存储桶

仅具有管理权限的用户允许删除存储桶，其他角色的用户无法进行删除操作。用户在删除存储桶之前，需先清空存储桶中的所有对象。

- 1. 在菜单栏中，点击【存储】->【对象存储 EOBS】，进入【对象存储 EOBS】页面。
- 2. 在【对象存储 EOBS】页面，选择对应的存储桶，点击操作栏下的【删除】按钮。

×

!

存储桶 test2 目前为空，删除后无法恢复！

您确定要删除吗？

请输入验证码

UFJh

5.2 对象管理

迪备用户拥有 EOBS 存储桶的权限，则可对存储桶内的文件进行管理。

5.2.1 查看对象

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【存储桶】。
2. 点击存储桶名称。进入存储桶详情。
3. 点击【对象列表】，进入对象列表页面，列表出存储桶中所有的文件及其信息，包括文件名称、大小、最后修改时间。

概览

对象列表

删除

按对象搜索 输入前缀，在当前目录搜索

test1/17.57/ShenTong/doc/Help/_images

☐	名称	大小	上次修改时间	操作
☐	 BCLOB.png	4 KiB	2 小时前	
☐	 DATETIME.png	12 KiB	2 小时前	
☐	 LOBtest.png	14 KiB	2 小时前	
☐	 NUMBER.png	10 KiB	2 小时前	
☐	 STCursor.png	5 KiB	2 小时前	
☐	 STRING.png	9 KiB	2 小时前	
☐	 TIMESTAMP.png	11 KiB	2 小时前	
☐	 arraysize.png	5 KiB	2 小时前	
☐	 bianchenliucheng.bmp	348 KiB	2 小时前	
☐	 bindnames.png	2 KiB	2 小时前	
☐	 bindvars.png	3 KiB	2 小时前	

显示第 1 到第 15 条记录，总共 150 + 条记录 每页显示 15 条记录

< 1 2 3 4 5 ... 10 >

1 跳转

5.2.2 删除对象

仅具有管理权限的用户允许删除对象，其他角色的用户无法进行删除对象的操作。

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【存储桶】，选择对应存储桶。
2. 进入【概览】页面，点击【对象列表】，进入对象列表页面。
3. 选择需要删除的对象，点击操作栏下的【删除】按钮。或点击左上方的【删除】按钮进行批量删除。

概览

删除

按对象搜索

输入前缀，在当前目录搜索

列表

对象列表

test1/17.57/ShenTong/doc/Help/_images

☒	名称	大小	上次修改时间	操作
☒	 BCLOB.png	4 KiB	2 小时前	
☒	 DATETIME.png	12 KiB	2 小时前	
☒	 LOBtest.png	14 KiB	2 小时前	
☒	 NUMBER.png	10 KiB	2 小时前	
☒	 STCursor.png	5 KiB	2 小时前	
☒	 STRING.png	9 KiB	2 小时前	
☒	 TIMESTAMP.png	11 KiB	2 小时前	
☒	 arraysize.png	5 KiB	2 小时前	
☒	 bianchenliucheng.bmp	348 KiB	2 小时前	
☒	 bindnames.png	2 KiB	2 小时前	
☒	 bindvars.png	3 KiB	2 小时前	

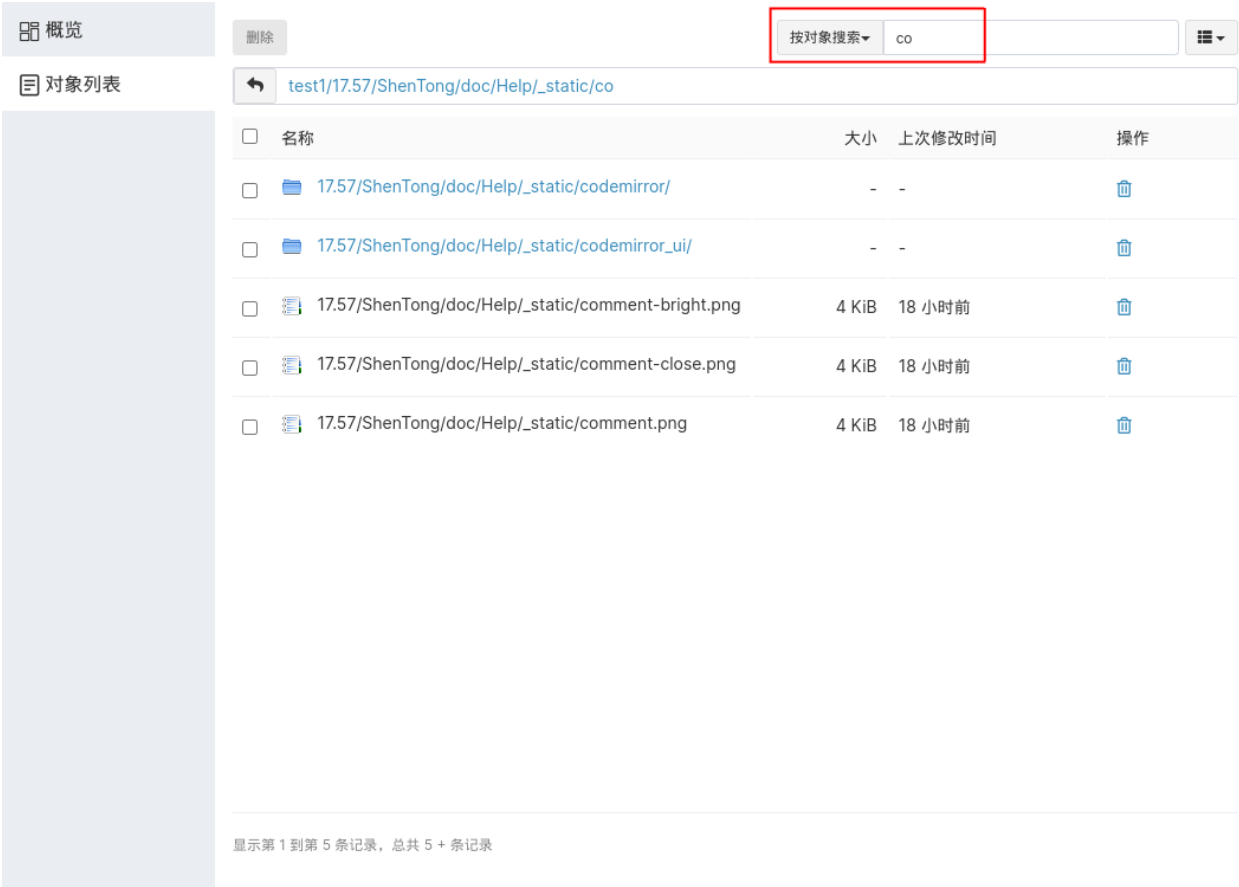
显示第 1 到第 15 条记录，总共 150 + 条记录 每页显示 15 条记录

< 1 2 3 4 5 ... 10 >

1 跳转

5.2.3 搜索对象

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【存储桶】，选择对应存储桶。
2. 进入【概览】页面，点击【对象列表】，进入对象列表页面。
3. 在右上角搜索栏中，输入需要搜索的对象名称。



备注：

- 仅支持对象名称的前缀搜索。
- 仅支持搜索当前目录下的文件或文件夹。

5.3 Access Key 管理

Access Key 和 Secret Key 是访问对象存储 EOBS API 的密钥，相当于常用的用户名和密码。

5.3.1 创建 Access Key

迪备用户在使用 EOBS 存储桶之前，需创建访问密钥 Access Key，否则无法访问。具体参考[创建访问密钥](#)。

5.3.2 禁用 Access Key

当暂时无需访问 Access Key 时，可选择禁用对应 Access Key，禁用后可再次启用。

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【Access Key】，进入【Access Key】页面。
2. 在【Access Key】页面，选择需要禁用的 Access Key，点击操作栏下的【禁用】按钮。

备注：为避免影响存储桶访问，请至少保留一个 Access Key。

5.3.3 删除 Access Key

当需要永久删除访问 Access Key 时，可选择删除对应 Access Key，删除后无法恢复。

1. 在菜单栏中，点击【存储】->【对象存储 EOBS】->【Access Key】，进入【Access Key】页面。
2. 在【Access Key】页面，选择需要删除的 Access Key，点击操作栏下的【删除】按钮。

存储桶

Access Key

新建 Access Key

自定义 Access Key

Access Key

状态

创建时间

操作

e5LP054jJI0hQndV

✓

2024-06-22 10:11:02

🔑 🗑️

test1

✓

2024-06-22 10:11:16

🔑 🗑️

0jcp4FhRDJxHT7r0

✓

2024-06-26 15:30:06

🔑 🗑️

v4TBgqk2lzUuWih2

✓

2024-06-26 15:39:12

🔑 🗑️

Access Key 和 Secret Key 是访问对象存储 EOBS API 的密钥，请妥善保管。如 Secret Key 丢失，可删除该密钥并创建新的 Access Key。

EOBS 兼容部分 S3 接口，满足业务备份功能所需要的 API。

6.1 API 概览

表 1: Service 操作

名称	功能
ListBuckets(ListService)	获取存储桶列表

表 2: Bucket 操作

名称	功能
HeadBucket	查询存储桶元数据
ListObjects(ListBucket)	获取对象列表
ListObjectsV2	获取对象列表

表 3: Object 基础操作

名称	功能
PutObject	简单上传对象，单个 PUT 操作中上传最大为 5GB 的单个对象
HeadObject	获取对象元数据
GetObject	下载对象
DeleteObject	删除对象
AppendObject	追加上传对象
DeleteObjects	删除同一 Bucket 中的多个对象

表 4: Object 分片上传

名称	功能
InitMultipartUpload	创建分片上传对象
ListMultipartUploads	列表分片上传对象任务
UploadPart	上传分片

续下页

表 4 – 接上页

名称	功能
ListParts	列表已经上传的分片
CompleteMultipartUpload	完成分片上传，将分片合并成对象
AbortMultipartUpload	终止分片上传任务

6.2 Service 操作

6.2.1 ListBuckets(ListService)（获取存储桶列表）

返回经过身份验证的请求发送方所拥有的所有存储桶的列表。

- 请求语法

```
GET / HTTPS
Host: IP:60306
```

- URI 请求参数
无
- 请求体
无
- 示例
获取请求者拥有的所有 Bucket。
- 请求示例

```
GET / HTTPS
Authorization: authorization string
Host: IP:60306
```

- 返回示例

```
HTTPS 200 OK
<ListAllMyBucketsResult>
  <Buckets>
    <Bucket>
      <CreationDate>2024-3-20T23:32:47+00:00</CreationDate>
      <String>DOC-EXAMPLE-BUCKET</String>
    </Bucket>
    <Bucket>
      <CreationDate>2024-3-20T23:32:50+00:00</CreationDate>
      <String>DOC-EXAMPLE-BUCKET2</String>
    </Bucket>
  </Buckets>
  <Owner>
    <DisplayName>admin</DisplayName>
    <ID>AIDACKCEVSQ6C2EXAMPLE</ID>
  </Owner>
</ListAllMyBucketsResult>
```

6.3 Bucket 操作

6.3.1 HeadBucket（查询存储桶元数据）

使用此操作来确定 Bucket 是否存在以及您是否有权访问它。如果 Bucket 存在并且您有权访问，此操作将返回 200 OK。

如果 Bucket 不存在或您没有访问它的权限，HEAD 请求将返回一个通用的 400 Bad request、403 Forbidden 或 404 not Found 代码。

- 请求语法

```
HEAD /Bucket HTTPS
Host: IP:60306
```

- URI 请求参数
 - Bucket: 存储桶名称。必选。
- 请求体
 - 无
- 示例
 - 获取已存在且有权限的存储桶 test-bucket 的元数据。
- 请求示例

```
HEAD /test-bucket HTTPS
Authorization: authorization string
Host: IP:60306
Connection: Keep-Alive
```

- 返回示例

```
HTTPS 200 OK
Accept-Ranges: bytes
Connection: Keep-Alive
Date: Mon, 25 Mar 2024 08:46:09 GMT
x-amz-request-id: 140180275062528-1711356369-5408C5D0
```

6.3.2 ListObjects(ListBucket)（获取对象列表）

返回 Bucket 中的部分或全部（最多 1000 个）对象。您可以使用请求参数作为选择标准来返回 Bucket 中对象的子集。

- 请求语法

```
GET /Bucket?delimiter=Delimiter&encoding-type=EncodingType&marker=Marker&max-keys=MaxKeys&
prefix=Prefix HTTPS
Host: IP:60306
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. delimiter: 对 Object 名字进行分组的字符。所有 Object 名字包含指定的前缀，第一次出现 delimiter 字符之间的 Object 作为一组元素（即 CommonPrefixes）。默认值：无。
 3. encoding-type: 对返回的内容进行编码并指定编码的类型。默认值：无。可选值：URL。
 4. marker: 设定从 marker 之后按字母排序开始返回 Object。marker 用来实现分页显示效果，参数的长度必须小于 1024 字节。做条件查询时，即使 marker 在列表中不存在，也会从符合 marker 字母排序的下一个开始打印。默认值：无。

- 5. **max-keys**: 指定返回 Object 的最大数。如果因为 **max-keys** 的设定无法一次完成列举, 返回结果会附加 **NextMarker** 元素作为下一次列举的 **marker**。取值: 大于 0 小于等于 1000。默认值: 1000。
- 6. **prefix**: 限定返回文件的 **Key** 必须以 **prefix** 作为前缀。如果把 **prefix** 设为某个文件夹名, 则列举以此 **prefix** 开头的文件, 即该文件夹下递归的所有文件和子文件夹。默认值: 无。

- 请求体
无
- 示例
获取请求者拥有 **test-bucket** 的所有 Object。
- 请求示例

```
GET /test-bucket HTTPS
Authorization: authorization string
Host: IP:60306
```

- 返回示例

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <Name>test-bucket</Name>
  <Prefix></Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>test4.txt</Key>
    <LastModified>2024-03-25T03:06:49Z</LastModified>
    <ETag>"46d7a9532282332f023c08fe25ff7105"</ETag>
    <Size>14</Size>
    <StorageClass>STANDARD</StorageClass>
    <Type>Normal</Type>
  </Contents>
</ListBucketResult>
```

6.3.3 ListObjectsV2（获取对象列表）

在每个请求中返回 Bucket 中的部分或全部（最多 1000 个）对象。您可以使用请求参数作为选择标准来返回 Bucket 中对象的子集。

- 请求语法

```
GET /?list-type=2&continuation-token=ContinuationToken&delimiter=Delimiter&encoding-
→type=EncodingType&fetch-owner=FetchOwner&max-keys=MaxKeys&prefix=Prefix&start-
→after=StartAfter HTTPS
Host: IP:60306
```

- URI 请求参数
 - 1. **Bucket**: 存储桶名称。必选。
 - 2. **list-type**: 取值只能为 2, 用于标识是 **ListObjectsV2** 接口。必选。
 - 3. **continuation-token**: 指定 List 操作需要从此 object 开始。您可从 **ListObjectsV2** 结果中的 **NextContinuationToken** 获取此 object。默认值: 无。
 - 4. **delimiter**: 对 Object 名字进行分组的字符。所有 Object 名字包含指定的前缀, 第一次出现 **delimiter** 字符之间的 Object 作为一组元素 (即 **CommonPrefixes**)。默认值: 无。
 - 5. **encoding-type**: 对返回的内容进行编码并指定编码的类型。默认值: 无。可选值: URL。
 - 6. **fetch-owner**: 指定是否在返回结果中包含 **owner** 信息。默认值: 无。
 - 7. **max-keys**: 指定返回 Object 的最大数。取值: 大于 0 小于等于 1000。默认值: 1000。

8. **prefix**: 限定返回文件的 **Key** 必须以 **prefix** 作为前缀。如果把 **prefix** 设为某个文件夹名, 则列举以此 **prefix** 开头的文件, 即该文件夹下递归的所有文件和子文件夹。默认值: 无。

9. **start-after**: 设定从 **start-after** 之后按字母排序开始返回 **Object**。默认值: 无。

- 请求体
无
- 示例
获取请求者拥有 **test-bucket** 的前 2 个 **Object**。
- 请求示例

```
GET /test-bucket/?list-type=2&max-keys=2 HTTPS
Authorization: authorization string
Host: IP:60306
```

- 返回示例

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <Name>test-bucket</Name>
  <Prefix></Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>2</MaxKeys>
  <IsTruncated>true</IsTruncated>
  <KeyCount>2</KeyCount>
  <NextContinuationToken>abc</NextContinuationToken>
  <Contents>
    <Key>aaa</Key>
    <LastModified>2024-03-27T05:38:53Z</LastModified>
    <ETag>"5eb63bbbe01eeed093cb22bb8f5acdc3"</ETag>
    <Size>11</Size>
    <StorageClass>STANDARD</StorageClass>
    <Type>Normal</Type>
  </Contents>
  <Contents>
    <Key>abc</Key>
    <LastModified>2024-03-27T05:38:46Z</LastModified>
    <ETag>"d41d8cd98f00b204e9800998ecf8427e"</ETag>
    <Size>0</Size>
    <StorageClass>STANDARD</StorageClass>
    <Type>Normal</Type>
  </Contents>
</ListBucketResult>
```

6.4 Object 基础操作

6.4.1 PutObject（简单上传对象）

上传对象。支持对象覆盖。最大上传 5G。

- 请求语法

```
PUT /Bucket/Object HTTPS
Host: IP:60306
Content-MD5: ContentMD5
Content-Type: ContentType
```

(续下页)

(接上页)

```
x-amz-meta-*: VALUE
Body
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. Object: 对象名称。必选。
 3. Content-MD5: 用于检查消息内容是否与发送时一致。Content-MD5 是由 MD5 算法生成的值。上传了 Content-MD5 请求头后, EOBS 会计算消息体的 Content-MD5 并检查一致性。如果需要通过 Content-MD5 进行 MD5 验证, 可将 Content-MD5 加入到请求头中。
 4. Content-Type: HTTP 请求内容类型。
 5. x-amz-meta-: 配置以 *x-amz-meta-* 为前缀的参数, 则该参数视为元数据。
- 请求体
自定义
- 示例
上传对象 test.txt 到存储桶 test-bucket。
- 请求示例

```
PUT /test-bucket/test.txt HTTPS
Host: IP:60306
Authorization: authorization string
[10 bytes of object data]
```

- 返回示例

```
HTTPS 200 OK
Server: nginx
Date: Mon, 25 Mar 2024 11:35:57 GMT
Content-Length: 10
ETag: "1b2cf535f27731c974343645a3985328"
x-amz-request-id: 140180233099008-1711366557-2806B850
```

6.4.2 GetObject (下载对象)

从 EOBS 服务器下载对象。

- 请求语法

```
GET /Bucket/Object HTTPS
Host: IP:60306
Range: Range
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. Object: 对象名称。必选。
 3. Range: 下载对象的指定字节范围。
- 请求体
空
- 错误码

表 5: GetObject 错误码

Code	HTTP status code	说明
NoSuchKey	404	指定的对象不存在

- 示例
从存储桶 test-bucket 下载对象 test.txt。
- 请求示例

```
GET /test-bucket/test.txt HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 200 OK
x-amz-request-id: 140180275062528-1711436458-34015170
Last-Modified: Tue, 26 Mar 2024 07:00:54 GMT
Date: Tue, 26 Mar 2024 07:00:58 GMT
Content-Length: 10
ETag: "1b2cf535f27731c974343645a3985328"
x-oss-object-type: Normal
[10 bytes of object data]
```

6.4.3 DeleteObject（删除对象）

从存储桶中删除对象。

- 请求语法

```
DELETE /Bucket/Object HTTPS
Host: IP:60306
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. Object: 对象名称。必选。
- 请求体
空
- 示例
删除存储桶 test-bucket 对象 test.txt。
- 请求示例

```
DELETE /test-bucket/test.txt HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 204
Date: Tue, 26 Mar 2024 07:08:00 GMT
x-amz-request-id: 140180275062528-1711436880-34015170
```

6.4.4 DeleteObjects（删除多个对象）

删除同一个存储桶中的多个对象。

- 请求语法

```
POST /Bucket?delete HTTPS
Host: IP:60306
Content-MD5: MD5Value
Encoding-type: EncodingType
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>key</Key>
  </Object>
  ...
</Delete>
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. delete: 用于指定 DeleteObjects 操作。必选。
 3. Content-MD5: 一串由 MD5 算法生成的值，用于检查消息内容是否与发送时一致。Content-MD5 是由 MD5 算法生成的值。上传了 Content-MD5 请求头后，EOBS 会计算消息体的 Content-MD5 并检查一致性。如果需要通过 Content-MD5 进行 MD5 验证，可将 Content-MD5 加入到请求头中。必选。
 4. Encoding-type: 对返回的内容进行编码并指定编码的类型。默认值：无。可选值：URL。
- 请求体
 1. Delete: 保存 DeleteMultipleObjects 请求的容器。必选。子节点：一个或多个 Object 元素，Quiet 元素；父节点：None。
 2. Quiet: 打开简单响应模式的开关。提供以下两种消息返回模式：简单模式（quiet）：不返回消息体。详细模式（verbose）：返回的消息体中会包含所有删除 Object 的结果。默认采用详细模式。有效值：true（开启简单模式）、false（开启详细模式）。默认值：false。父节点：Delete。
 3. Object: 保存一个 Object 信息的容器。必选。子节点：Key；父节点：Delete。
 4. Key: 被删除 Object 的名字。必选。父节点：Object。
- 示例

删除存储桶 test-bucket 对象 test1.txt, test2.txt。

 - 请求示例

```
POST /test-bucket?delete HTTPS
Host: IP:60306
Authorization: authorization string
Content-MD5: E5h2ePW99nHpNbqx6s33jg==
<Delete><Quiet>false</Quiet><Object><Key>test1.txt</Key></Object><Object><Key>
→ test2.txt</Key></Object></Delete>
```

* 返回示例

```
HTTPS 200 OK
Date: Wed, 27 Mar 2024 03:27:21 GMT
Content-Type: application/xml
x-amz-request-id: 140180275062528-1711510041-5424FF00

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult>
  <Deleted>
```

(续下页)

(接上页)

```
<Key>test1.txt</Key>
</Deleted>
<Deleted>
  <Key>test2.txt</Key>
</Deleted>
</DeleteResult>
```

6.4.5 AppendObject（追加上传对象）

以追加写的方式上传对象。

- 请求语法

```
POST /Bucket/Object?append&position=Position HTTPS
Host: IP:60306
Content-MD5: ContentMD5
x-amz-meta-*: VALUE
Body
```

- URI 请求参数
 - Bucket: 存储桶名称。必选。
 - Object: 对象名称。必选。
 - append: 用于指定 AppendObject 操作。必选。
 - position: 用于指定从何处进行追加。每次操作成功后，响应消息头 x-amz-next-append-position 会标明下一次追加的 position。必选。
 - Content-MD5: 用于检查消息内容是否与发送时一致。Content-MD5 是由 MD5 算法生成的值。上传了 Content-MD5 请求头后，EOBS 会计算消息体的 Content-MD5 并检查一致性。如果需要通过 Content-MD5 进行 MD5 验证，可将 Content-MD5 加入到请求头中。
 - x-amz-meta-: 配置以 x-amz-meta- 为前缀的参数，则该参数视为元数据。
- 请求体
- 自定义
- 错误码

表 6: AppendObject 错误码

Code	HTTP status code	说明
ObjectNotAppendable	409	对一个非 Appendable Object 进行 AppendObject 操作。
InvalidAppendPosition	400	错误的追加位置。

- 示例

以追加方式上传对象 append.txt 到存储桶 test-bucket。

 - 请求示例

```
POST /test-bucket/append.txt?append&position=0 HTTPS
Host: IP:60306
Authorization: authorization string
[3 bytes of object data]
```

- 返回示例

```
HTTPS 200 OK
Date: Tue, 26 Mar 2024 07:22:18 GMT
Content-Length: 0
ETag: "1b2cf535f27731c974343645a3985328"
x-amz-request-id: 140180233099008-1711366557-2806B850
x-amz-next-append-position: 3
x-obs-next-append-position: 3
x-oss-next-append-position: 3
```

6.4.6 HeadObject 获取对象元数据

获取某个对象的元数据。使用此接口不会返回对象内容。

- 请求语法

```
HEAD /Bucket/Object HTTPS
Host: IP:60306
```

- URI 请求参数
 - Bucket：存储桶名称。必选。
 - Object：对象名称。必选。
- 请求体
空
- 错误码

表 7: HeadObject 错误码

Code	HTTP status code	说明
NoSuchKey	404	指定的对象不存在

- 示例
获取存储桶 test-bucket 对象 test.txt 元数据。
 - 请求示例

```
HEAD /test-bucket/test.txt HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 200 OK
x-amz-request-id: 140180275062528-1711436458-34015170
Last-Modified: Tue, 26 Mar 2024 07:00:54 GMT
Date: Tue, 26 Mar 2024 07:00:58 GMT
Content-Length: 3
ETag: "1b2cf535f27731c974343645a3985328"
Content-Type: text/plain
```

6.5 Object 分片上传

6.5.1 InitMultipartUploads（创建分片上传对象）

使用 Multipart Upload 模式传输数据前，您必须先调用 InitiateMultipartUpload 接口来初始化一个 Multipart Upload 事件。

- 请求语法

```
POST /Bucket/Object?uploads HTTPS
Host: IP:60306
Content-Type: ContentType
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. Object: 对象名称。必选。
 3. uploads: 用于指定 InitMultipartUploads 操作。必选。
 4. Content-Type: HTTP 请求内容类型。

- 请求体
空

- 示例
在存储桶 test-bucket 创建分片对象 example-object。
 - 请求示例

```
POST /test-bucket/example-object?uploads HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 200 OK
x-amz-request-id: 140180233099008-1711366557-2806B850
Date: Mon, 25 Mar 2024 11:35:57 GMT
Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult>
  <Bucket>test-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>8836f462-eb45-11ee-8000-fa163e24d887</UploadId>
</InitiateMultipartUploadResult>
```

6.5.2 ListMultipartUploads（列表分片上传对象）

列举所有执行中的 Multipart Upload 事件，即已经初始化但还未完成（Complete）或者还未中止（Abort）的 Multipart Upload 事件。

- 请求语法

```
GET /Bucket?uploads&delimiter=Delimiter&encoding-type=EncodingType&key-
  ↪marker=KeyMarker&max-uploads=MaxUploads&prefix=Prefix&upload-id-
  ↪marker=UploadIdMarker HTTPS
Host: IP:60306
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. uploads: 用于指定 ListMultipartUploads 操作。必选。

3. **delimiter**: 用于对 Object 名称进行分组的字符。所有名称包含指定的前缀且首次出现 **delimiter** 字符之间的 Object 作为一组元素 **CommonPrefixes**。默认值: 无。
4. **encoding-type**: 对返回的内容进行编码并指定编码的类型。默认值: 无。可选值: URL。
5. **key-marker**: 与 **upload-id-marker** 参数配合使用, 用于指定返回结果的起始位置。
 - 1) 如果未设置 **upload-id-marker** 参数, 查询结果中包含所有 Object 名称的字典序大于 **key-marker** 参数值的 **Multipart Upload** 事件。
 - 2) 如果设置了 **upload-id-marker** 参数, 查询结果中包含: 所有 Object 名称的字典序大于 **key-marker** 参数值的 **Multipart Upload** 事件。Object 名称等于 **key-marker** 参数值, 但是 **UploadId** 比 **upload-id-marker** 参数值大的 **Multipart Upload** 事件。
6. **max-uploads**: 限定此次返回 **Multipart Upload** 事件的最大个数。最大值为 1000。
7. **prefix**: 限定返回的 Object Key 必须以 **prefix** 作为前缀。注意使用 **prefix** 查询时, 返回的 Key 中仍会包含 **prefix**。默认值: 无。
8. **upload-id-marker**: 与 **key-marker** 参数配合使用, 用于指定返回结果的起始位置。如果未设置 **key-marker** 参数, 则忽略 **upload-id-marker** 参数。如果设置了 **key-marker** 参数, 查询结果中包含: 所有 Object 名称的字典序大于 **key-marker** 参数值的 **Multipart Upload** 事件。Object 名称等于 **key-marker** 参数值, 但是 **UploadId** 比 **upload-id-marker** 参数值大的 **Multipart Upload** 事件。

- 请求体

无

- 示例

列举存储桶 **test-bucket** 所有执行中的 **Multipart Upload** 事件。

- 请求示例

```
GET /test-bucket?uploads HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 200 OK
Date: Tue, 26 Mar 2024 07:22:18 GMT
Content-Length: 459
x-amz-request-id: 140180233099008-1711366557-2806B850

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult>
  <Bucket>test-bucket</Bucket>
  <Prefix></Prefix>
  <Delimiter></Delimiter>
  <KeyMarker></KeyMarker>
  <UploadIdMarker></UploadIdMarker>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <Upload>
    <Key>example-object</Key>
    <Initiated>2024-03-26T07:50:36Z</Initiated>
    <StorageClass>STANDARD</StorageClass>
    <UploadId>8836f462-eb45-11ee-8000-fa163e24d887</UploadId>
  </Upload>
</ListMultipartUploadsResult>
```

6.5.3 UploadPart（上传分片）

根据指定的对象名和 uploadId 来分块（Part）上传数据。

- 请求语法

```
PUT /Bucket/Object?partNumber=PartNumber&uploadId=UploadId HTTPS
Host: IP:60306
Content-MD5: ContentMD5
Body
```

- URI 请求参数

1. Bucket: 存储桶名称。必选。
2. Object: 对象名称。必选。
3. partNumber: 每一个上传的 Part 都有一个标识它的号码（partNumber）。取值：1~10000。必选。
4. uploadId: 唯一标识上传的 Part 属于哪个 Object。必选。
5. Content-MD5: 用于检查消息内容是否与发送时一致。Content-MD5 是由 MD5 算法生成的值。上传了 Content-MD5 请求头后，EOBS 会计算消息体的 Content-MD5 并检查一致性。如果需要通过 Content-MD5 进行 MD5 验证，可将 Content-MD5 加入到请求头中。

- 请求体

自定义

- 示例

上传存储桶 test-bucket 分片对象 example-object。

- 请求示例

```
PUT /test-bucket/example-object?partNumber=1&uploadId=8836f462-eb45-11ee-8000-
fa163e24d887 HTTPS
Host: IP:60306
Authorization: authorization string
[5242880 bytes of object data]
```

- 返回示例

```
HTTPS 200 OK
Date: Tue, 26 Mar 2024 07:22:18 GMT
Content-Length: 0
ETag: "79b281060d337b9b2b84ccf390adcf74"
x-amz-request-id: 140180233099008-1711366557-2806B850
```

6.5.4 ListParts（列表已经上传的分片）

列举指定 Upload ID 所属的所有已经上传成功 Part。

- 请求语法

```
GET /Bucket/Object?max-parts=MaxParts&part-number-marker=PartNumberMarker&
uploadId=UploadId&encoding-type=EncodingType HTTPS
Host: IP:60306
```

- URI 请求参数

1. Bucket: 存储桶名称。必选。
2. Object: 对象名称。必选。
3. uploadId: MultipartUpload 事件的 ID。必选。
4. max-parts: 规定在 OSS 响应中的最大 Part 数目。最大值：1000。
5. part-number-marker: 指定 List 的起始位置，只有 Part Number 数目大于该参数的 Part 会被列出。默认值：无。
6. encoding-type: 对返回的内容进行编码并指定编码的类型。默认值：无。可选值：URL。

- 请求体
无
- 示例
列举存储桶 test-bucket 分片对象 example-object 已成功上传的 Part。
 - 请求示例

```
GET /test-bucket/example-object?uploadId=8836f462-eb45-11ee-8000-fa163e24d887 HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 200 OK
x-amz-request-id: 140180233099008-1711366557-2806B850
Date: Mon, 25 Mar 2024 11:35:57 GMT
Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult>
  <Bucket>test-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>8836f462-eb45-11ee-8000-fa163e24d887</UploadId>
  <MaxParts>1000</MaxParts>
  <IsTruncated>false</IsTruncated>
  <StorageClass></StorageClass>
  <ChecksumAlgorithm></ChecksumAlgorithm>
  <Part>
    <PartNumber>1</PartNumber>
    <Size>5242880</Size>
    <ETag>"79b281060d337b9b2b84ccf390adcf74"</ETag>
    <LastModified>2024-03-26 09:40:19</LastModified>
  </Part>
  <Part>
    <PartNumber>3</PartNumber>
    <Size>5242880</Size>
    <ETag>"79b281060d337b9b2b84ccf390adcf74"</ETag>
    <LastModified>2024-03-26 10:07:45</LastModified>
  </Part>
</ListPartsResult>
```

6.5.5 AbortMultipartUpload（终止分片上传任务）

用于取消 MultipartUpload 事件并删除对应的 Part 数据。

- 请求语法

```
DELETE /Bucket/Object?uploadId=UploadId HTTPS
Host: IP:60306
```

- URI 请求参数

1. Bucket: 存储桶名称。必选。
2. Object: 对象名称。必选。
3. uploadId: MultipartUpload 事件的 ID。必选。

- 请求体

空

- 错误码

表 8: AbortMultipartUpload 错误码

Code	HTTP status code	说明
NoSuchUpload	404	指定的 uploadId 不存在

- 示例
删除存储桶 test-bucket 分片对象 example-object 的指定 uploadId。
 - 请求示例

```
DELETE /test-bucket/example-object?uploadId=ca83035e-eb5b-11ee-8000-fa163e24d887
→HTTPS
Host: IP:60306
Authorization: authorization string
```

- 返回示例

```
HTTPS 204 No Content
x-amz-request-id: 140180233099008-1711366557-2806B850
Date: Mon, 25 Mar 2024 11:35:57 GMT
Connection: Keep-Alive
```

6.5.6 CompleteMultipartUpload（合并分片）

在将所有数据 Part 都上传完成后，您必须调用此接口来完成整个文件的分片上传。

- 请求语法

```
POST /Bucket/Object?uploadId=UploadId HTTPS
Host: IP:60306
<CompleteMultipartUpload>
  <Part>
    <ETag>string</ETag>
    <PartNumber>integer</PartNumber>
  </Part>
  ...
</CompleteMultipartUpload>
```

- URI 请求参数
 1. Bucket: 存储桶名称。必选。
 2. Object: 对象名称。必选。
 3. uploadId: MultipartUpload 事件的 ID。必选。
- 请求体
请求接收以下 XML 格式的数据：
 1. CompleteMultipartUpload: 保存 CompleteMultipartUpload 请求内容的容器。必选。子节点：一个或多个 Part 元素；父节点：无。
 2. Part: 保存已上传 Part 信息的容器。必选。子节点：ETag、PartNumber；父节点：CompleteMultipartUpload。
 3. ETag: Part 成功上传后，返回的 ETag 值。必选。父节点：Part。
 4. PartNumber: 第几个 Part。必选。父节点：Part。
- 错误码

表 9: CompleteMultipartUpload 错误码

Code	HTTP status code	说明
NoSuchUpload	404	指定的 uploadId 不存在

- 示例
 - 合并存储桶 test-bucket 对象 example-object 的分片。
 - 请求示例

```
POST /test-bucket/example-object?uploadId=8836f462-eb45-11ee-8000-fa163e24d887
→HTTPS
Host: IP:60306
Authorization: authorization string
<CompleteMultipartUpload>
  <Part>
    <ETag>"79b281060d337b9b2b84ccf390adcf74"</ETag>
    <PartNumber>1</PartNumber>
  </Part>
  <Part>
    <ETag>"79b281060d337b9b2b84ccf390adcf74"</ETag>
    <PartNumber>3</PartNumber>
  </Part>
</CompleteMultipartUpload>
```

- 返回示例

```
HTTPS 200 OK
x-amz-request-id: 140180275062528-1711436458-34015170
Date: Tue, 26 Mar 2024 07:00:58 GMT
Content-Length: 278
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult>
  <Location>http://192.168.18.229:60306/test-bucket/example-object</Location>
  <Bucket>test-bucket</Bucket>
  <Key>example-object</Key>
  <ETag>"a2781d22990923995b6d103873b1ab2d-2"</ETag>
</CompleteMultipartUploadResult>
```

表 10: EOBS 功能限制列表

功能	限制描述
S3 协议开启与关闭	1. 当存储池内不存在存储桶时，S3 协议自动关闭。 2. 当存储池内存在存储桶时，不支持关闭 S3 协议。
存储桶	1. 存储桶名称全局唯一。 2. 存储桶名称仅支持长度为 3~63 的小写字母、数字和连字符 (-)，但不能以 ds、d2、连字符开头，或以连字符结尾。 3. 存储桶一旦创建成功，名称、读写权限、所属的存储池均不支持修改。
对象搜索	1. 仅支持对象名称的前缀搜索。 2. 仅支持搜索当前目录下的文件或文件夹。
对象限制	1. 对象名称限制： (1) 使用 UTF-8 编码。 (2) 存在大小写区分。 (3) 长度必须在 1~1023 字符之间。 (4) 不能以正斜线 (/) 或者反斜线 (\) 开头。 2. PutObject 方式上传普通对象，限制大小为 5G；UploadPart 方式上传分片对象，限制大小为 48.8T。 3. 上传对象数量取决于磁盘空间大小。
防篡改重删池	1. 不支持开启防篡改的重删池创建存储桶。
分布式重删池	1. 访问信息终端节点为主节点 IP。 2. 分布式重删池创建完成后，不支持切换主节点。
存储桶复制	1. 不支持存储桶的复制。
备份集保留策略	1. 存储桶内对象不受备份集保留策略管理。
存储桶访问权限	1. 操作员在没有存储池权限的情况下，创建的 Access Key 无法访问存储桶。 2. admin、普通管理员、操作员在对其 Access Key 禁用或删除后，无法访问存储桶。（存在一两分钟延迟） 3. 普通操作员的 Access Key 仅能访问有权限的存储池上创建的存储桶。 4. admin、普通管理员的 Access Key 可以访问所有存储桶。

表 11：术语

术语	说明
S3	AWS 对象存储服务的名称，是 Simple Storage Service 的缩写。
对象存储	基于对象的存储组织技术，数据通过唯一的对象名称进行索引。
存储桶	对象存储的逻辑存储空间，一个存储桶中的对象具有相同的访问接入点、管理策略等。
对象	对象存储中数据存储的单位。
认证凭据（credential）	S3 API 使用一对配套的 Access Key 和 Secret Key，对每个请求按照特定的算法计算请求的签名，实现接口的认证和鉴权。



全国销售热线：400-650-0081

电话：+86 20 32053160

总部地址：广州市科学城科学大道243号总部经济区A5栋9楼

全国服务热线：400-003-3191

网址：www.scutech.com