Документация Яндекс Объектное хранилище

Условия и стоимость использования программного обеспечения Яндекс Объектное хранилище определяются индивидуально. Чтобы узнать стоимость использования ПО, напишите на почтовый адрес yandex-space@yandex-team.ru.

Название документа	Ссылка на скачивание
Документация, содержащая информацию, необходимую для эксплуатации экземпляра ПО	Скачать документ
Документация, содержащая описание процессов, обеспечивающих поддержание жизненного цикла ПО, в том числе устранение неисправностей и совершенствование, а также информацию о персонале, необходимом для обеспечения такой поддержки	Скачать документ
Документация, содержащая описание технических средств хранения исходного текста и объектного кода программного обеспечения, а также технических средств компиляции исходного текста в объектный код программного обеспечения	Скачать документ
Документация, содержащая описание технических средств, необходимых для активации, выпуска, распространения, управления лицензионными ключами программного обеспечения	Скачать документ
Документация, содержащая описание функциональных характеристик экземпляра ПО	Скачать документ
Инструкция по установке экземпляра ПО	Скачать документ

Создание бакета

Чтобы создать бакет, у вас должна быть *минимальная* роль `EDITOR` на тенант.

AWS CLI

Чтобы создать бакет, назначьте учетной записи S3, через который работает AWS CLI, роль `EDITOR`.

В терминале выполните команду:

```
aws s3api create-bucket \
  --endpoint-url=https://<s3_api_хост> \
  --bucket <имя_бакета>
```

Где:

- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--bucket` — имя бакета.

Результат:

```
{
  "Location": "/<имя_бакета>"
}
```

Будет создан бакет со следующими параметрами:

- без ограничения максимального размера;
- с ограниченным доступом на чтение объектов, к списку объектов и на чтение настроек.

Опциональные параметры

Вы можете применить к бакету предопределенный ACL или настроить разрешения для отдельных учетных записей S3, публичных групп (группа всех пользователей, группа всех аутентифицированных пользователей). Эти настройки несовместимы: у бакета должен быть либо предопределенный ACL, либо набор отдельных разрешений.

> **Примечание**
>
> Чтобы управлять настройками ACL бакета, назначьте учетной записи S3, через который работает AWS CLI, роль `ADMIN`.

Предопределенный ACL

```
aws s3api create-bucket \
  --endpoint-url=https://<s3_api_хост> \
```

```
  --bucket <имя_бакета> \
  --acl <предопределенный_ACL>
```

Где `--acl` — предопределенный ACL.

Отдельные разрешения

```
aws s3api create-bucket \
  --endpoint-url=https://<s3_api_хост> \
  --bucket <имя_бакета> \
  <тип_разрешения> <получатель_разрешения>
```

Где:

- Возможные типы разрешений ACL:
  - `--grant-read` — доступ к списку объектов в бакете, чтению различных настроек бакета (жизненный цикл, CORS), чтению всех объектов в бакете.
  - `--grant-write` — доступ к записи, перезаписи и удалению объектов в бакете. Используется только совместно с `--grant-read`.
  - `--grant-full-control` — полный доступ к бакету и объектам в нем.

  Вы можете задать несколько разрешений в одной команде.
- Возможные получатели разрешений:
  - `id=<идентификатор_получателя>` — идентификатор учетной записи S3, которой нужно дать разрешение.
  - `uri=http://acs.amazonaws.com/groups/global/AuthenticatedUsers` — публичная группа всех аутентифицированных пользователей.
  - `uri=http://acs.amazonaws.com/groups/global/AllUsers` — публичная группа всех пользователей.

По умолчанию для каждого нового бакета создается пустой ACL.

API

Чтобы создать бакет, воспользуйтесь методом S3 API `PutBucket`.

Удаление бакета

> **Важно**
>
> Удалить можно только пустой бакет.

AWS CLI

В терминале выполните команду `aws s3api delete-bucket`:

```
aws s3api delete-bucket \
  --endpoint-url=https://<s3_api_хост> \
  --bucket <имя_бакета>
```

Где:

- `--bucket` — имя бакета, который нужно удалить.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

Также вы можете использовать команду `aws s3 rb`:

```
aws --endpoint-url=https://<s3_api_хост> \
  s3 rb s3://<имя_бакета>
```

API

Чтобы удалить бакет, воспользуйтесь методом S3 API `DeleteBucket`.

Ограничение максимального размера бакета

ПО Яндекс Объектное хранилище позволяет ограничить максимальный размер бакета.

Консоль управления

1. В консоли управления на панели слева выберите **Тенанты**, перейдите в нужный тенант, выберите вкладку **Бакеты** и перейдите в бакет, максимальный размер которого вы хотите ограничить.
2. Нажмите **Редактировать**.
3. Задайте значение поля **Макс. размер**.

   Размер `0` означает отсутствие ограничений и аналогичен включенной опции **Без ограничений**.
4. Нажмите **Сохранить**.

Управление жизненными циклами объектов в бакете

ПО Яндекс Объектное хранилище позволяет управлять жизненными циклами объектов в бакете.

Раз в сутки к жизненным циклам применяются изменения, актуальные на момент 00:00 UTC. Операция выполняется в течение нескольких часов.

AWS CLI

Чтобы загрузить конфигурацию с помощью AWS CLI:

1. Опишите конфигурацию жизненных циклов объектов в формате JSON. Например:

   ```
   {
     "Rules": [
       {
         "ID": "DeleteOldBackups",
         "Filter": {
           "Prefix": "backup/"
         },
         "Status": "Enabled",
         "Expiration": {
           "Days": 180
         }
       }
     ]
   }
   ```

   Возможные параметры конфигурации:

   - `ID` — уникальный идентификатор правила. Должен быть меньше или равен 255 символам. Необязательный параметр.
   - `Filter` — фильтр объектов. Необязательный параметр. Может содержать не более одного элемента каждого типа:
     - `Prefix` — префикс ключа объекта. Под действие правила попадают объекты с указанным префиксом ключа. Необязательный параметр.
     - `ObjectSizeGreaterThan` — минимальный размер объекта в байтах. Под действие правила попадают объекты, размер которых больше или равен указанному. Необязательный параметр.
     - `ObjectSizeLessThan` — максимальный размер объекта в байтах. Под действие правила попадают объекты, размер которых меньше или равен указанному. Необязательный параметр.
     - `Tag` — метка объекта. Необязательный параметр. Под действие правила попадают объекты, которым присвоена указанная метка. Передается в виде записи, содержащей две пары значений, например:

       ```
       "Tag": [{"Key": "some_key", "Value": "some_value"}]
       ```

- `And` — логический оператор **И** (`AND`) для фильтров. Необязательный параметр. Используйте этот фильтр, чтобы сочетать в одном правиле фильтрацию по префиксу, размеру или меткам объекта. С помощью логического оператора `And` вы можете настроить фильтрацию одновременно по нескольким меткам. Для этого в ключе `Tags` передайте метки в виде массива объектов, содержащих по две пары значений.

  Пример фильтра с логическим оператором **И** (`AND`):

  ```
  "Filter": {
    "And": {
      "Prefix": "backup/",
      "Tags": [
        {
          "Key": "key_1",
          "Value": "value_1"
        },
        {
          "Key": "key_2",
          "Value": "value_2"
        }
      ],
      "ObjectSizeGreaterThan": 1,
      "ObjectSizeLessThan": 1024
    }
  }
  ```

Если фильтр объектов не задан, под действие правила попадают все объекты в бакете.

- `Status` — статус правила. Обязательный параметр. Значения:
  - `Enabled` — правило включено.
  - `Disabled` — правило выключено.
- `Expiration` — параметр правила для удаления любых объектов. Необязательный параметр. Может содержать:
  - `Date` — дата, после которой объект будет удален. Формат — ISO 8601, например `YYYY-MM-DD`. Время — всегда 00:00 UTC. Ключ `Date` нельзя использовать одновременно с ключом `Days`. Необязательный параметр.
  - `Days` — количество дней с момента создания объекта, после которого объект будет удален. Минимальное значение — `1`. Ключ `Days` нельзя использовать одновременно с ключом `Date`. Необязательный параметр.
  - `ExpiredObjectDeleteMarker` — удаляет маркер удаления, для которого больше не существует неактуальных версий объекта. Принимает значения `true` или `false`. Необязательный параметр.
- `NoncurrentVersionExpiration` — параметр правила для удаления неактивных версий объектов. Необязательный параметр.

  Правило содержит обязательный параметр `NoncurrentDays` — количество дней до удаления неактивной версий объекта. Минимальное значение — `1`.

- `AbortIncompleteMultipartUpload` — параметр правила для удаления всех частей составных загрузок, не завершенных за указанное количество дней. Необязательный параметр.

  Правило содержит обязательный параметр `DaysAfterInitiation` — количество дней с начала загрузки. Минимальное значение — `1`.

Укажите хотя бы один из параметров `Expiration`, `NoncurrentVersionExpiration` или `AbortIncompleteMultipartUpload`.

Сохраните готовую конфигурацию в файле, например `lifecycles.json`.

2. Загрузите конфигурацию в бакет, например, в `backup-bucket`:

```
aws s3api put-bucket-lifecycle-configuration \
  --bucket backup-bucket \
  --endpoint-url=https://<s3_api_хост> \
  --lifecycle-configuration file://lifecycles.json
```

API

Чтобы управлять жизненными циклами объектов в бакете, воспользуйтесь методом S3 API `PutBucketLifecycle`.

Настройка CORS

ПО Яндекс Объектное хранилище позволяет управлять конфигурацией CORS в бакете.

AWS CLI

Чтобы загрузить конфигурацию с помощью AWS CLI:

1. Опишите конфигурацию CORS объектов в формате JSON. Например:

   ```
   {
     "CORSRules": [
       {
         "AllowedHeaders": ["*"],
         "AllowedMethods": ["GET", "HEAD", "PUT", "DELETE"],
         "MaxAgeSeconds": 3000,
         "AllowedOrigins": ["*"]
       }
     ]
   }
   ```

   Готовую конфигурацию можно поместить в файл, например, `cors.json`.

2. Загрузите конфигурацию в бакет, например, `shared-bucket`:

   ```
   aws s3api put-bucket-cors \
     --bucket shared-bucket \
     --cors-configuration file://cors.json \
     --endpoint-url=https://<s3_api_хост>
   ```

API

Чтобы управлять конфигурацией CORS в бакете, воспользуйтесь методом S3 API `PutBucketCors`.

Редактирование ACL бакета

> **Примечание**
>
> Если ранее для бакета уже был задан ACL, то после применения изменений ACL будет полностью перезаписан.

AWS CLI

> **Примечание**
>
> Чтобы управлять ACL бакета, назначьте учетной записи S3, через который работает AWS CLI, роль `ADMIN`.

Посмотрите текущий ACL бакета:

```
aws s3api get-bucket-acl \
  --endpoint https://<s3_api_хост> \
  --bucket <имя_бакета>
```

Где:

- `--bucket` — имя бакета.
- `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

Вы можете применить к бакету предопределенный ACL или настроить разрешения для отдельных учетных записей S3 и публичных групп (группа всех пользователей, группа всех аутентифицированных пользователей). Эти настройки несовместимы: у бакета должен быть либо предопределенный ACL, либо набор отдельных разрешений.

Предопределенный ACL

Выполните команду:

```
aws s3api put-bucket-acl \
  --endpoint https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --acl <предопределенный_ACL>
```

Где:

- `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--bucket` — имя бакета.

- `--acl` — предопределенный ACL.

Настройка отдельных разрешений

1. Чтобы выдать разрешения ACL для учетной записи S3, получите ее идентификатор.
2. Выполните команду:

   ```
   aws s3api put-bucket-acl \
     --endpoint https://<s3_api_хост> \
     --bucket <имя_бакета> \
     <тип_разрешения> <получатель_разрешения>
   ```

   Где:

   - `--bucket` — имя бакета.
   - `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - Возможные типы разрешений ACL:
     - `--grant-read` — доступ к списку объектов в бакете, чтению различных настроек бакета (жизненный цикл, CORS), чтению всех объектов в бакете.
     - `--grant-write` — доступ к записи, перезаписи и удалению объектов в бакете. Используется только совместно с `--grant-read`.
     - `--grant-full-control` — полный доступ к бакету и объектам в нем.
   - Возможные получатели разрешений:
     - `id=<идентификатор_получателя>` — идентификатор учетной записи S3, которой нужно дать разрешение.
     - `uri=http://acs.amazonaws.com/groups/global/AuthenticatedUsers` — публичная группа всех аутентифицированных пользователей.
     - `uri=http://acs.amazonaws.com/groups/global/AllUsers` — публичная группа всех пользователей.

   Чтобы настроить несколько разрешений, укажите параметры, тип разрешения и получателя разрешения несколько раз. Например, чтобы выдать разрешение на запись в бакет, выполните команду:

   ```
   aws s3api put-bucket-acl \
     --endpoint https://storage.yandexcloud.net \
     --bucket <имя_бакета> \
     --grant-read id=<идентификатор_получателя> \
     --grant-write id=<идентификатор_получателя>
   ```

API

Чтобы редактировать ACL бакета, воспользуйтесь методом S3 API `PutBucketAcl`.

Управление политикой доступа (bucket policy)

Политики доступа (bucket policy) устанавливают права на действия с бакетами, объектами и группами объектов.

Применить или изменить политику

Минимально необходимая роль для применения или изменения политики доступа — `ADMIN`.

> **Примечание**
>
> Если ранее бакету была назначена политика доступа, то после применения изменений она будет полностью перезаписана.

Для применения или изменения политики доступа к бакету:

AWS CLI

> **Примечание**
>
> Для управления политикой доступа с помощью AWS CLI учетной записи S3 должна быть назначена роль `ADMIN`.

1. Опишите конфигурацию политики доступа в виде схемы данных формата JSON:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::<имя_бакета>/*",
    "Condition": {
      "Bool": {
        "aws:SecureTransport": "true"
      }
    }
  }
}
```

Где:

- `Version` — версия описания политик доступа. Необязательный параметр.
- `Statement` — правила политики доступа:

- `Effect` — запрет или разрешение запрошенного действия. Возможные значения: `Allow` и `Deny`.
- `Principal` — идентификатор субъекта запрошенного разрешения. Получателем может быть учетная запись S3. Возможные значения: `*`, `<идентификатор_субъекта>`. Необязательный параметр.
- `Action` — действие, которое будет разрешено при срабатывании политики. Возможные значения: `s3:GetObject`, `s3:PutObject` и `*` если необходимо применять политику ко всем действиям.
- `Resource` — ресурс, к которому будет применяться правило.
- `Condition` — условие, которое будет проверяться. Необязательный параметр.

  Вы можете настроить для правила одновременно несколько условий и задать для каждого условия несколько ключей. При этом такие условия и их ключи будут проверяться с логикой `И`. То есть для выполнения правила запрос должен будет соответствовать сразу всем указанным признакам.

  Для каждого ключа условия вы можете задать одновременно несколько значений. При этом такие значения будут проверяться с логикой `ИЛИ`. То есть для выполнения правила запрос должен будет соответствовать любому из заданных значений ключа условия.

Сохраните готовую конфигурацию в файле `policy.json`.

2. Выполните команду:

```
aws s3api put-bucket-policy \
  --endpoint https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --policy file://policy.json
```

API

Чтобы управлять политикой доступа, воспользуйтесь методом S3 API `PutBucketPolicy`. Если ранее для бакета уже была установлена политика доступа, то после применения новой политики она будет полностью перезаписана.

Примечание

Если к бакету применена политика доступа без правил, то доступ будет запрещен всем пользователям. Чтобы отключить проверки запросов по политике доступа, удалите ее.

Просмотреть политику

Минимально необходимая роль для просмотра политики доступа — `storage.configViewer`.

Чтобы просмотреть примененную к бакету политику доступа:

AWS CLI

Выполните следующую команду:

```
aws --endpoint https://<s3_api_хост> s3api get-bucket-policy \
  --bucket <имя_бакета> \
  --output text
```

Результат:

```
{
  "Policy": "{\"Version\":\"2012-10-17\",\"Statement\":
{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"ar
<имя_бакета>/*\",\"Condition\":{\"Bool\":{\"aws:SecureTransport\":\"true\"}}}}"
}
```

API

Воспользуйтесь методом S3 API `GetBucketPolicy`.

Удалить политику

Минимально необходимая роль для удаления политики доступа — `ADMIN`.

Чтобы удалить политику доступа:

AWS CLI

Выполните следующую команду:

```
aws --endpoint https://<s3_api_хост> s3api delete-bucket-policy \
  --bucket <имя_бакета>
```

API

Воспользуйтесь методом S3 API `DeleteBucketPolicy`.

Управление версионированием бакета

Версионирование бакета — это возможность хранить историю объекта с помощью версий.

> **Примечание**
>
> Операция включения необратима: отключить версионирование нельзя, можно только приостановить создание новых версий. После приостановки версионирования новые объекты будут сохраняться с версией `null`.

Включить версионирование бакета:

AWS CLI

Выполните следующую команду:

```
aws --endpoint https://<s3_api_хост> \
  s3api put-bucket-versioning \
  --bucket <имя_бакета> \
  --versioning-configuration 'Status=Enabled'
```

API

Чтобы управлять версионированием бакета, воспользуйтесь методом S3 API `PutBucketVersioning`.

Управление блокировками версий объектов (object lock) в бакете

В версионируемых бакетах можно настроить механизм *блокировки версий объектов* (object lock). Когда он включен, вы можете установить на версию объекта блокировку, то есть запрет на удаление или перезапись. Также для бакета можно настроить блокировки по умолчанию: они будут устанавливаться на все новые версии объектов.

> **Примечание**
>
> В бакетах с приостановленным версионированием блокировка версий объектов недоступна.

Включить возможность блокировок

Включение механизма блокировок не устанавливает блокировки на уже загруженные версии объектов, а только позволяет устанавливать их.

Минимальная необходимая роль — `ADMIN`.

Чтобы включить возможность блокировок:

AWS CLI

Выполните следующую команду:

```
aws s3api put-object-lock-configuration \
  --bucket <имя_бакета> \
  --object-lock-configuration ObjectLockEnabled=Enabled \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--object-lock-configuration` — настройки блокировок в бакете. Значение `ObjectLockEnabled=Enabled` включает механизм блокировок.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Воспользуйтесь методом S3 API `PutObjectLockConfiguration`.

Настроить блокировки по умолчанию

Блокировки по умолчанию устанавливаются на все новые версии объектов, загружаемые в бакет. Настройки не влияют на уже загруженные версии.

Минимальная необходимая роль — `ADMIN`.

Чтобы настроить блокировки по умолчанию:

AWS CLI

1. Опишите конфигурацию блокировок по умолчанию в формате JSON:

   ```
   {
     "ObjectLockEnabled": "Enabled",
     "Rule": {
       "DefaultRetention": {
         "Mode": "<тип_блокировки>",
         "Days": <срок_блокировки_в_днях>,
         "Years": <срок_блокировки_в_годах>
       }
     }
   }
   ```

   Где:

   - `ObjectLockEnabled` — статус механизма блокировок: `Enabled` — механизм включен.

     > **Внимание**
     >
     > Это обязательное поле. Если не указать его со значением `Enabled`, появится сообщение об ошибке `InvalidRequest`, а механизм блокировок не включится.

   - `Mode` — тип блокировки:
     - `GOVERNANCE` — временная управляемая блокировка.
     - `COMPLIANCE` — временная строгая блокировка.
   - `Days` — срок блокировки в днях от момента загрузки версии объекта. Должен быть положительным целым числом. Нельзя указывать вместе с `Years`.
   - `Years` — срок блокировки в годах от момента загрузки версии объекта. Должен быть положительным целым числом. Нельзя указывать вместе с `Days`.

   Готовую конфигурацию можно поместить в файл, например, `default-object-lock.json`.

2. Загрузите конфигурацию в бакет:

   ```
   aws s3api put-object-lock-configuration \
     --bucket <имя_бакета> \
     --object-lock-configuration file://default-object-lock.json \
     --endpoint-url=https://<s3_api_хост>
   ```

   Где:

- `--bucket` — имя бакета.
- `--object-lock-configuration` — настройки блокировок по умолчанию. В данном случае указаны в файле `default-object-lock.json`.
- `--endpoint-url` — эндпоинт ПО Яндекс Объектное хранилище.

Выключить возможность блокировок

Выключение механизма блокировок не снимает установленные блокировки: они продолжают работать, их нельзя будет снять или изменить.

Минимальная необходимая роль — `ADMIN`.

Чтобы выключить возможность блокировок:

AWS CLI

Выполните следующую команду:

```
aws s3api put-object-lock-configuration \
  --bucket <имя_бакета> \
  --object-lock-configuration ObjectLockEnabled="" \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--object-lock-configuration` — настройки блокировок в бакете. Значение `ObjectLockEnabled=""` отключает механизм блокировок.
- `--endpoint-url` — эндпоинт ПО Яндекс Объектное хранилище.

API

Чтобы выключить возможность блокировок версий объектов в бакете, воспользуйтесь методом S3 API `PutObjectLockConfiguration`.

В теле запроса передайте параметр блокировки версий объектов `ObjectLockConfiguration` с пустым значением.

Пример тела HTTP-запроса:

```
<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/" />
```

Управление метками бакета

Метки бакетов — это пары ключ-значение для логической маркировки бакетов.

Добавить или изменить метки

AWS CLI

В терминале выполните команду:

```
aws s3api put-bucket-tagging \
  --bucket <имя_бакета> \
  --tagging 'TagSet=[{Key=<ключ_метки>,Value=<значение_метки>},{Key=
<ключ_метки>,Value=<значение_метки>}]' \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--tagging` — массив меток бакета, где:
  - `Key` — ключ метки, тип: `string`.
  - `Value` — значение метки, тип: `string`.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Чтобы добавить или изменить метки бакета, воспользуйтесь методом S3 API `PutBucketTagging`.

Посмотреть метки

AWS CLI

В терминале выполните команду:

```
aws s3api get-bucket-tagging \
  --bucket <имя_бакета> \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

Результат:

```
{
    "TagSet": [
```

```
        {
            "Key": "test-key-1",
            "Value": "test-value-1"
        },
        {
            "Key": "test-key-2",
            "Value": "test-value-2"
        }
    ]
}
```

API

Чтобы посмотреть метки бакета, воспользуйтесь методом S3 API `GetBucketTagging`.

Удалить метки

AWS CLI

В терминале выполните команду:

```
aws s3api delete-bucket-tagging \
  --bucket <имя_бакета> \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Чтобы удалить метки бакета, воспользуйтесь методом S3 API `DeleteBucketTagging`.

Загрузка объекта

Пользователь может создавать внутри бакета папки и загружать туда объекты. При этом необходимо помнить, что в SDK и HTTP API ключом объекта будет весь путь к объекту от корня бакета.

Для загрузки объектов в бакет вы можете использовать инструменты, поддерживающие работу с ПО Яндекс Объектное хранилище, а также подписанные URL-ссылки.

Простая загрузка

AWS CLI

1. Чтобы загрузить один объект, выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 cp <путь_к_локальному_файлу> s3://<имя_бакета>/<ключ_объекта>
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 cp` — команда для загрузки объекта. Чтобы загрузить объект, в первой части команды укажите путь к локальному файлу, который нужно загрузить, а во второй — имя вашего бакета и ключ, по которому объект будет храниться в бакете.

   Чтобы загрузить все объекты из локальной директории, используйте команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 cp --recursive <путь_к_локальной_директории>/ s3://<имя_бакета>/<префикс>/
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 cp --recursive` — команда для загрузки всех объектов из локальной директории, включая вложенные. Чтобы загрузить объекты, в первой части команды укажите путь к папке, из которой нужно скопировать файлы в бакет, а во второй — имя вашего бакета и идентификатор папки в хранилище.

Команда `aws s3 cp` — высокоуровневая, ее функциональность ограничена. Подробнее см. в справочнике AWS CLI. Все возможности загрузки, которые поддерживаются в ПО Яндекс Объектное хранилище, можно использовать при выполнении команды aws s3api put-object.

API

Чтобы загрузить объект, воспользуйтесь методом S3 API `PutObject`.

Загрузка версии объекта с блокировкой (object lock)

Если в бакете включены версионирование и блокировки версий объектов, вы можете указать настройки блокировки (запрета на удаление или перезапись) при загрузке версии объекта.

AWS CLI

1. Выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3api put-object \
     --body <путь_к_локальному_файлу> \
     --bucket <имя_бакета> \
     --key <ключ_объекта> \
     --object-lock-mode <тип_временной_блокировки> \
     --object-lock-retain-until-date <дата_и_время_окончания_временной_блокировки> \
     --object-lock-legal-hold-status <статус_бессрочной_блокировки>
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3api put-object` — команда для загрузки версии объекта. Чтобы загрузить версии объекта с блокировкой, укажите следующие параметры:
     - `--body` — путь к файлу, который нужно загрузить в бакет.
     - `--bucket` — имя вашего бакета.
     - `--key` — ключ, по которому объект будет храниться в бакете.
     - `--object-lock-mode` — тип временной блокировки:
       - `GOVERNANCE` — временная управляемая блокировка.
       - `COMPLIANCE` — временная строгая блокировка.
     - `--object-lock-retain-until-date` — дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с параметром `--object-lock-mode`.
     - `--object-lock-legal-hold-status` — статус бессрочной блокировки:
       - `ON` — блокировка установлена.
       - `OFF` — блокировка не установлена.

   Вы можете установить на версию объекта только временную блокировку (параметры `object-lock-mode` и `object-lock-retain-until-date`), только бессрочную блокировку (`object-lock-legal-hold-status`) или обе сразу.

API

Чтобы загрузить версию объекта с блокировкой, воспользуйтесь методом S3 API `PutObject` с заголовками `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date` для временной блокировки и `X-Amz-Object-Lock-Legal-Hold` для бессрочной.

Если для бакета также настроены временные блокировки по умолчанию, все объекты в него нужно загружать с указанием их MD5-хешей:

AWS CLI

1. Вычислите MD5-хэш файла и закодируйте его по схеме Base64:

   ```
   md5=($(md5sum <путь_к_локальному_файлу>))
   md5_base64=$(echo $md5 | base64)
   ```

2. Загрузите объект в бакет:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3api put-object \
     --body <путь_к_локальному_файлу> \
     --bucket <имя_бакета> \
     --key <ключ_объекта> \
     --content-md5 $md5_base64
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3api put-object` — команда для загрузки версии объекта. Чтобы загрузить версии объекта, укажите следующие параметры:
     - `--body` — путь к файлу, который нужно загрузить в бакет.
     - `--bucket` — имя вашего бакета.
     - `--key` — ключ, по которому объект будет храниться в бакете.
     - `--content-md5` — закодированный MD5-хеш объекта.

   Также вы можете добавить к команде параметры:

   - `--object-lock-mode` и `--object-lock-retain-until-date`, чтобы установить на версию объекта временную блокировку, отличную от настроек бакета по умолчанию;
   - `--object-lock-legal-hold-status`, чтобы установить на версию объекта бессрочную блокировку.

API

Чтобы загрузить версию объекта с временной блокировкой по умолчанию, воспользуйтесь методом S3 API `PutObject` с заголовком `Content-MD5`.

Получение информации об объекте

API

Чтобы получить информацию об объекте, воспользуйтесь методом S3 API `GetObjectMeta`.

Скачивание объекта

> **Примечание**
>
> Чтобы скачать группу объектов с определенным префиксом (папку с объектами) или все объекты из бакета, используйте AWS CLI или совместимые с Amazon S3 API файловые браузеры, например CyberDuck и WinSCP.

AWS CLI

Скачать один объект

```
aws s3 cp \
  --endpoint-url=https://<s3_api_хост> \
  s3://<имя_бакета>/<ключ_объекта> \
  <локальный_путь>
```

Где:

- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `<имя_бакета>` — имя бакета, из которого вы хотите скачать объект.
- `<ключ_объекта>` — ключ объекта, который вы хотите скачать.
- `<локальный_путь>` — путь к папке, в которую будет сохранен скачанный объект. Например, `~/downloads/`.

Скачать папку (все объекты с определенным префиксом)

```
aws s3 cp \
  --endpoint-url=https://<s3_api_хост> \
  --recursive \
  s3://<имя_бакета>/<префикс>/ \
  <локальный_путь>
```

Где:

- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--recursive` — параметр для скачивания всех объектов с указанным префиксом.
- `<имя_бакета>` — имя бакета, из которого вы хотите скачать объекты.
- `<префикс>` — префикс (папка) объектов, которые вы хотите скачать, например `test/folder`.
- `<локальный_путь>` — путь к папке, в которую будут сохранены скачанные объекты, например `~/downloads/`.

Скачать все объекты из бакета

```
aws s3 cp \
  --endpoint-url=https://<s3_api_хост> \
  --recursive \
  s3://<имя_бакета> \
  <локальный_путь>
```

Где:

- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--recursive` — параметр для скачивания всех объектов бакета в локальную папку.
- `<имя_бакета>` — имя бакета, из которого вы хотите скачать объекты.
- `<локальный_путь>` — путь к папке, в которую будут сохранены скачанные объекты. Например, `~/downloads/`.

Объекты из бакета можно скачать выборочно с помощью команды `aws s3api` и шаблона запроса в формате JMESPath. Для скачивания объектов по шаблону выполните команду:

- **Bash**:

  ```
  aws s3api list-objects \
      --endpoint-url https://<s3_api_хост> \
      --bucket <имя_бакета> \
      --query '<запрос>' \
      --output text | xargs -I {} aws s3api get-object --endpoint-url
  https://<s3_api_хост> --bucket <имя_бакета> --key {} <локальный_путь>{}
  ```

  Где:

  - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
  - `--bucket` — имя бакета, из которого вы хотите скачать объекты.
  - `--query` — запрос в формате JMESPath.
  - `<локальный_путь>` — путь к папке, в которую будут сохранены скачанные объекты. Например, `~/downloads/`.

  Пример команды для скачивания из бакета `sample-bucket` в локальную папку `~/downloads/` всех объектов, имена файлов которых начинаются с `date-20231002`:

  ```
  aws s3api list-objects \
    --endpoint-url https://<s3_api_хост> \
    --bucket sample-bucket \
    --query 'Contents[?starts_with(Key, `date-20231002`) == `true`].[Key]' \
    --output text | xargs -I {} aws s3api get-object --endpoint-url
  https://<s3_api_хост> --bucket sample-bucket --key {} ~/downloads/{}
  ```

- **PowerShell**:

```
Foreach($x in (aws s3api list-objects `
  --endpoint-url https://<s3_api_хост> `
  --bucket <имя_бакета> `
  --query '<запрос>' `
  --output text)) `
  {aws s3api get-object --endpoint-url https://<s3_api_хост> --bucket
<имя_бакета> --key $x <локальный_путь>$x}
```

Где:

- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--bucket` — имя бакета, из которого вы хотите скачать объекты.
- `--query` — запрос в формате JMESPath.
- `<локальный_путь>` — путь к папке, в которую будут сохранены скачанные объекты. Например, `d:\downloads\`.

Пример команды для скачивания из бакета `sample-bucket` в локальную папку `d:\downloads\` всех объектов, имена файлов которых начинаются с `date-20231002`:

```
Foreach($x in (aws s3api list-objects `
  --endpoint-url https://<s3_api_хост> `
  --bucket sample-bucket `
  --query 'Contents[?starts_with(Key, `date-20231002`) == `true`].[Key]' `
  --output text)) `
  {aws s3api get-object --endpoint-url https://<s3_api_хост> --bucket sample-
bucket --key $x d:\downloads\$x}
```

API

Чтобы скачать объект, воспользуйтесь методом S3 API `GetObject`.

Восстановление версии объекта в версионируемом бакете

Для восстановления версий объектов бакет должен быть версионируемым. Восстановить можно только те версии, которые были загружены при включенном версионировании.

AWS CLI

Чтобы восстановить версию объекта с помощью AWS CLI:

1. Получите идентификатор нужной версии объекта:

   ```
   aws s3api list-object-versions \
     --endpoint-url https://<s3_api_хост> \
     --bucket <имя_бакета> \
     --prefix <префикс_ключа_объекта>
   ```

   В результате отобразится список версий всех объектов, ключи которых начинаются с указанного префикса. Идентификаторы версий содержатся в параметрах `VersionId`.

   Пример получения списка версий

   Команда:

   ```
   aws s3api list-object-versions \
     --endpoint-url https://<s3_api_хост> \
     --bucket my-bucket \
     --prefix index.html
   ```

   Результат:

   ```
   {
      "Versions": [
          {
              "LastModified": "2015-11-10T00:20:11.000Z",
              "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZ********",
              "ETag": "\"0622528de826c0df5db1258a********\"",
              "StorageClass": "STANDARD",
              "Key": "index.html",
              "Owner": {
                  "DisplayName": "my-username",
                  "ID":
   "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460dd********"
              },
              "IsLatest": true,
              "Size": 38
          },
          {
              "LastModified": "2015-11-09T23:26:41.000Z",
              "VersionId": "rasWWGpgk9E4s0LyTJgusGeR********",
   ```

```
            "ETag": "\"06225825b8028de826c0df5d********\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460dd********"
            },
            "IsLatest": false,
            "Size": 38
        }
    ]
}
```

Чтобы выбрать только один объект:

1.1. Установите и инициализируйте jq.

1.2. Отфильтруйте результат:

```
aws s3api list-object-versions \
  --endpoint-url https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --prefix <префикс_ключа_объекта> \
| jq '.Versions | map(select(.Key == "<ключ_объекта>"))'
```

Подробнее о команде читайте в ее описании в документации Amazon.

2. Скопируйте версию объекта в тот же бакет с тем же ключом, чтобы она стала текущей версией объекта:

```
aws s3api copy-object \
  --endpoint-url https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --copy-source <имя_бакета>/<ключ_объекта>?versionId=<идентификатор_версии> \
  --key <ключ_объекта>
```

Где:

- `--bucket` – имя бакета.
- `--copy-source` – исходный объект для копирования с указанием идентификатора нужной версии.
- `--key` – ключ целевого объекта. Чтобы восстановить версию объекта, ключи целевого и исходного объектов должны совпадать.

Результат:

```
{
  "CopyObjectResult": {
```

```
    "LastModified": "<дата_и_время_последнего_изменения_объекта>",
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
  },
  "VersionId": "<идентификатор_восстановленной_версии_объекта>"
}
```

Подробнее о команде читайте в ее описании в документации Amazon.

API

Чтобы восстановить версию объекта:

1. Получите идентификатор нужной версии объекта с помощью метода S3 API `ListObjectVersions`.
2. Скопируйте версию объекта в тот же бакет с тем же ключом с помощью метода S3 API `PutObjectCopy`.

Переименование и перемещение объекта

Переименование

AWS CLI

1. Чтобы переименовать объект, выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 mv s3://<имя_бакета>/<ключ_объекта> s3://<имя_бакета>/<новый_ключ_объекта>
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 mv` — команда для переименования или перемещения объекта. Чтобы переименовать объект, в первой части команды укажите имя бакета и актуальный ключ объекта, который нужно переименовать, а во второй — имя бакета и новый ключ объекта.

   Результат:

   ```
   move: s3://<имя_бакета>/<ключ_объекта> to s3://<имя_бакета>/<новый_ключ_объекта>
   ```

Подробнее о команде `aws s3 mv` см. в документации AWS CLI Command Reference.

Перемещение

AWS CLI

1. Чтобы переместить объект, например, из одного бакета в другой, выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 mv s3://<имя_бакета-источника>/<ключ_объекта>
   s3://<имя_целевого_бакета>/<ключ_объекта>
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 mv` — команда для переименования или перемещения объекта. Чтобы переместить объект, в первой части команды укажите имя бакета-источника и ключ объекта, который нужно переместить, а во второй — имя целевого бакета и ключ объекта.

   Результат:

   ```
   move: s3://<имя_бакета-источника>/<ключ_объекта> to
   s3://<имя_целевого_бакета>/<ключ_объекта>
   ```

Подробнее о команде `aws s3 mv` см. в документации AWS CLI Command Reference.

В ПО Яндекс Объектное хранилище папки имитируются с помощью префиксов ключей. Чтобы переместить объект из одной папки в другую, переименуйте префикс его ключа.

Копирование объектов

В ПО Яндекс Объектное хранилище поддерживается копирование объектов на стороне сервера (*Server Side Copy*).

Большие объекты, загруженные с помощью составной загрузки, хранятся в бакете частями. Копирование таких объектов происходит путем вызова `UploadPartCopy` для каждой его части.

Вы можете скопировать как все содержимое бакета, так и отдельный его объект.

Копирование одного объекта

AWS CLI

1. Выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 cp s3://<бакет_источник>/<ключ_объекта> s3://<бакет_приемник>/<ключ_объекта>
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 cp` — команда для копирования объектов.

   Результат:

   ```
   copy: s3://<бакет_источник>/<ключ_объекта> to
   s3://<бакет_приемник>/<ключ_объекта>
   ```

   Подробнее о команде `aws s3 cp` см. в документации AWS CLI Command Reference.

Копирование всех объектов бакета в другой бакет

AWS CLI

1. Выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3 cp --recursive s3:<бакет_источник>/ s3:<бакет_приемник>/
   ```

   Где:

   - `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `s3 cp` — команда для копирования объектов.
   - `--recursive` — параметр для копирования всех объектов бакета-источника.

   Подробнее о команде `aws s3 cp` см. в документации AWS CLI Command Reference.

Все объекты будут скопированы из бакета-источника в бакет-приемник.

Примечание

Вы можете скопировать объекты между бакетами, как в рамках одного тенанта, так и между разными тенантами. Для этого убедитесь, что у используемой учетной записи есть права на запись в оба бакета.

Получение ссылки на скачивание

ПО Яндекс Объектное хранилище позволяет сгенерировать подписанную ссылку на объект. Любой человек, получивший эту ссылку, сможет скачать объект из бакета.

Примечание

Продлить срок действия подписанной ссылки на объект в бакете нельзя. Если срок действия ссылки истек, сформируйте новую ссылку.

Настройка блокировок версии объекта (object lock)

Если в бакете включены версионирование и блокировки версий объектов, вы можете настроить блокировку версии, уже загруженной в бакет.

Установить или настроить временную блокировку (governance или compliance)

Минимальные необходимые роли:

- для установки блокировки — `EDITOR`;
- для изменения существующей блокировки — `ADMIN`.

Временную строгую блокировку (compliance-mode retention) можно только продлить. Сократить ее или заменить на управляемую блокировку (governance-mode retention) нельзя.

Чтобы установить или настроить блокировку:

AWS CLI

1. Выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3api put-object-retention \
     --bucket <имя_бакета> \
     --key <ключ_объекта> \
     --version-id <идентификатор_версии> \
     --retention Mode=<тип_блокировки>,RetainUntilDate="<срок_блокировки>" \
     --bypass-governance-retention
   ```

   Где:

   - `--bucket` — имя вашего бакета.
   - `--key` — ключ объекта.
   - `--version-id` — идентификатор версии объекта.
   - `--retention` — настройки временной блокировки (оба параметра обязательны):
     - `Mode` — тип блокировки:
       - `GOVERNANCE` — временная управляемая блокировка. Этот тип нельзя указать, если на версию объекта уже установлена строгая блокировка.
       - `COMPLIANCE` — временная строгая блокировка.
     - `RetainUntilDate` — дата и время окончания блокировки в формате RFC3339. Например, `2025-01-01T00:00:00Z`. Конец блокировки указывается в часовом поясе UTC±00:00. Чтобы указать другой часовой пояс, добавьте к концу записи `+` или `-` и смещение от UTC±00:00. Если на версию объекта уже установлена строгая блокировка, ее можно только продлить, то есть новые дата и время должны быть позже текущих.

- `--bypass-governance-retention` — флаг, подтверждающий обход блокировки. Его нужно установить, если на версию объекта уже установлена управляемая блокировка.

API

Воспользуйтесь методом S3 API `PutObjectRetention`.

Снять временную управляемую блокировку (governance)

Минимально необходимая роль — `ADMIN`.

Чтобы снять блокировку:

AWS CLI

1. Выполните команду:

   ```
   aws --endpoint-url=https://<s3_api_хост>/ \
     s3api put-object-retention \
     --bucket <имя_бакета> \
     --key <ключ_объекта> \
     --version-id <идентификатор_версии> \
     --retention "{}" \
     --bypass-governance-retention
   ```

   Где:

   - `--bucket` — имя вашего бакета.
   - `--key` — ключ объекта.
   - `--version-id` — идентификатор версии объекта.
   - `--retention` — настройки временной блокировки. В обоих параметрах указаны пустые строки, чтобы снять блокировку.
   - `--bypass-governance-retention` — флаг, подтверждающий обход блокировки.

API

Воспользуйтесь методом S3 API `PutObjectRetention` с заголовком `X-Amz-Bypass-Governance-Retention: true` и пустым элементом `Retention`.

Установить или удалить бессрочную блокировку (legal hold)

Минимально необходимая роль — `ADMIN`.

Чтобы установить или удалить блокировку:

AWS CLI

1. Выполните команду:

```
aws --endpoint-url=https://<s3_api_хост>/ \
  s3api put-object-legal-hold \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --version-id <идентификатор_версии> \
  --legal-hold Status=<статус_блокировки>
```

Где:

- `--bucket` — имя вашего бакета.
- `--key` — ключ объекта.
- `--version-id` — идентификатор версии объекта.
- `--legal-hold` — настройки бессрочной блокировки:
  - `Status` — статус блокировки:
    - `ON` — блокировка установлена.
    - `OFF` — блокировка не установлена.

API

Воспользуйтесь методом S3 API `PutObjectLegalHold`.

Примеры

Установка управляемой блокировки со смещением по московскому времени (UTC+3)

AWS CLI

```
aws --endpoint-url=https://<s3_api_хост>/ \
  s3api put-object-retention \
  --bucket test-bucket \
  --key object-key/ \
  --version-id 0005FA15******** \
  --retention Mode=GOVERNANCE,RetainUntilDate="2025-01-01T00:00:00+03:00" \
```

Удаление объекта

Удалить объект или версию объекта без блокировки

Любой объект или версию объекта, на которую не установлена блокировка (в том числе если в бакете не включены блокировки), можно удалить без дополнительных подтверждений.

Минимальная необходимая роль — `EDITOR`.

Чтобы удалить объект:

AWS CLI

В терминале выполните команду `aws s3api delete-object`:

```
aws s3api delete-object \
  --endpoint-url https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --key <ключ_объекта>
```

Где:

- `--bucket` — имя вашего бакета.
- `--key` — ключ объекта.

Чтобы одновременно удалить список объектов, укажите ключи этих объектов в параметре `--delete`:

- **Bash**:

  ```
  aws s3api delete-objects \
    --endpoint-url=https://<s3_api_хост> \
    --bucket <имя_бакета> \
    --delete '{"Objects":[{"Key":"<ключ_объекта_1>"},{"Key":"
  <ключ_объекта_2>"},...,{"Key":"<ключ_объекта_n>"}]}'
  ```

- **PowerShell**:

  ```
  aws s3api delete-objects `
    --endpoint-url=https://<s3_api_хост> `
    --bucket <имя_бакета> `
    --delete '{\"Objects\":[{\"Key\":\"<ключ_объекта_1>\"},{\"Key\":\"
  <ключ_объекта_2>\"},...,{\"Key\":\"<ключ_объекта_n>\"}]}'
  ```

Где:

- `--bucket` — имя бакета.

- `<ключ_объекта_1>`, `<ключ_объекта_2>`, `<ключ_объекта_n>` — ключи объектов, которые нужно удалить.

Результат:

```
{
  "Deleted": [
      {
          "Key": "<ключ_объекта_1>",
          "VersionId": "null"
      },
      {
          "Key": "<ключ_объекта_2>",
          "VersionId": "null"
      }
      ...
      {
          "Key": "<ключ_объекта_n>",
          "VersionId": "null"
      }
  ]
}
```

Указать объекты для удаления можно с помощью шаблона запроса в формате JMESPath. Для удаления объектов по шаблону выполните команду:

- **Bash**:

  ```
  aws s3api list-objects \
    --endpoint-url https://<s3_api_хост> \
    --bucket <имя_бакета> \
    --query '<запрос>' \
    --output text | xargs -I {} aws s3api delete-object --endpoint-url
  https://<s3_api_хост> --bucket <имя_бакета> --key {}
  ```

  Где:

  - `--bucket` — имя бакета.
  - `--query` — запрос в формате JMESPath.

  Пример команды для удаления из бакета `sample-bucket` всех объектов, расположенных в папке `screenshots`, имена файлов которых начинаются с даты `20231002`:

  ```
  aws s3api list-objects \
    --endpoint-url https://<s3_api_хост> \
    --bucket sample-bucket \
    --query 'Contents[?starts_with(Key, `screenshots/20231002`) == `true`].[Key]' \
  ```

```
  --output text | xargs -I {} aws s3api delete-object --endpoint-url
https://<s3_api_хост> --bucket sample-bucket --key {}
```

- **PowerShell**:

```
Foreach($x in (aws s3api list-objects `
  --endpoint-url https://<s3_api_хост> `
  --bucket <имя_бакета> `
  --query '<запрос>' `
  --output text)) `
  {aws s3api delete-object --endpoint-url https://<s3_api_хост> --bucket
<имя_бакета> --key $x}
```

Где:

- `--bucket` — имя бакета.
- `--query` — запрос в формате JMESPath.

Пример команды для удаления из бакета `sample-bucket` всех объектов, расположенных в папке `screenshots`, имена файлов которых начинаются с даты `20231002`:

```
Foreach($x in (aws s3api list-objects `
  --endpoint-url https://<s3_api_хост> `
  --bucket sample-bucket `
  --query 'Contents[?starts_with(Key, `screenshots/20231002`) == `true`].[Key]' `
  --output text)) `
  {aws s3api delete-object --endpoint-url https://<s3_api_хост> --bucket sample-
bucket --key $x}
```

API

Воспользуйтесь методом S3 API `DeleteObject`.

Удалить версию объекта с блокировкой (object lock)

Если в бакете включены блокировки версий объектов, некоторым или всем пользователям может быть запрещено удалять версию объекта.

Чтобы проверить, установлена ли блокировка, и удалить версию объекта при возможности:

AWS CLI

1. Получите информацию о блокировке версии объекта:

```
aws --endpoint-url=https://<s3_api_хост> \
  s3api head-object \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --version-id <идентификатор_версии>
```

Где:

- `--bucket` — имя вашего бакета.
- `--key` — ключ объекта.
- `--version-id` — идентификатор версии объекта.

Если на версию установлена блокировка, информация о ней отобразится в результате выполнения команды:

```
{
  ...
  "ObjectLockMode": "<тип_временной_блокировки>",
  "ObjectLockRetainUntilDate": "<дата_и_время>",
  "ObjectLockLegalHoldStatus": "<статус_бессрочной_блокировки>",
  ...
}
```

Где:

- `ObjectLockMode` — тип временной блокировки:
  - `GOVERNANCE` — временная управляемая блокировка. Удалить версию объекта может пользователь с ролью `ADMIN`.
  - `COMPLIANCE` — временная строгая блокировка. Удалить версию объекта нельзя.
- `ObjectLockRetainUntilDate` — дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`.
- `ObjectLockLegalHoldStatus` — статус бессрочной блокировки:
  - `ON` — включена. Удалить версию объекта нельзя. Снять блокировку может пользователь с ролью `ADMIN`.
  - `OFF` — выключена.

2. Если установлена временная управляемая блокировка (`"ObjectLockMode": "GOVERNANCE"`) и у вас есть роль `ADMIN`, удалите версию объекта:

```
aws --endpoint-url=https://<s3_api_хост> \
  s3api delete-object \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --version-id <идентификатор_версии> \
  --bypass-governance-retention
```

Где:

- `--bucket` — имя вашего бакета.
- `--key` — ключ объекта.
- `--version-id` — идентификатор версии объекта.
- `--bypass-governance-retention` — флаг, подтверждающий обход блокировки.

API

1. Чтобы получить информацию о блокировке версии объекта, воспользуйтесь методами S3 API `GetObjectRetention` (временная блокировка) и `GetObjectLegalHold` (бессрочная блокировка).
2. Если установлена только временная управляемая блокировка (`GOVERNANCE`) и у вас есть роль `ADMIN`, для удаления версии объекта воспользуйтесь методом S3 API `DeleteObject`. Укажите в запросе идентификатор версии и заголовок `X-Amz-Bypass-Governance-Retention`, чтобы подтвердить обход блокировки.

Удаление всех объектов из бакета

Чтобы очистить бакет:

AWS CLI

1. Создайте переменную с именем бакета:

   ```
   BUCKET_NAME=<имя_бакета>
   ```

2. Удалите все версии объектов из бакета:

   ```
   aws s3api delete-objects \
     --endpoint-url https://<s3_api_хост> \
     --bucket $BUCKET_NAME \
     --delete \
       "$(aws s3api list-object-versions \
         --endpoint-url https://<s3_api_хост> \
         --bucket $BUCKET_NAME \
         --query '{Objects: Versions[].{Key: Key, VersionId: VersionId}}' \
         --max-items 1000)"
   ```

   Также вместо параметра `--max-items` вы можете использовать `--max-keys`.

   Результат:

   ```
   {
       "Deleted": [
           {
               "Key": "object_000",
               "VersionId": "0005CDD3********"
           },
           {
               "Key": "object_001",
               "VersionId": "0005CDD3********"
           },
           ...
       ]
   }
   ```

   С помощью этой команды можно удалить до 1000 версий объектов: это связано с ограничением операции `aws s3api delete-objects`. Если в бакете больше версий, повторите команду несколько раз.

3. Если для бакета включено версионирование, удалите все delete-маркеры:

   ```
   aws s3api delete-objects \
     --endpoint-url https://<s3_api_хост> \
   ```

```
  --bucket $BUCKET_NAME \
  --delete \
    "$(aws s3api list-object-versions \
      --endpoint-url https://<s3_api_хост> \
      --bucket $BUCKET_NAME \
      --query '{Objects: DeleteMarkers[].{Key: Key, VersionId: VersionId}}' \
      --max-items 1000)"
```

Также вместо параметра `--max-items` вы можете использовать `--max-keys`.

Результат:

```
{
    "Deleted": [
        {
            "Key": "object_034",
            "DeleteMarker": true,
            "DeleteMarkerVersionId": "0005CDD3********"
        },
        {
            "Key": "object_057",
            "DeleteMarker": true,
            "DeleteMarkerVersionId": "0005CDD3********"
        },
        ...
    ]
}
```

С помощью этой команды можно удалить до 1000 delete-маркеров: это связано с ограничением операции `aws s3api delete-objects`. Если в бакете больше версий, повторите команду несколько раз.

4. Удалите частично загруженные объекты:

```
aws s3api list-multipart-uploads \
  --endpoint-url https://<s3_api_хост> \
  --bucket $BUCKET_NAME \
| jq -r '.Uploads[] | "--key \"\(.Key)\" --upload-id \(.UploadId)"' \
| while read -r line; do
  eval
    "aws s3api abort-multipart-upload \
      --endpoint-url https://<s3_api_хост> \
      --bucket $BUCKET_NAME \
      $line";
done
```

5. Получите список частей объектов, оставшихся в бакете:

```
aws s3api list-multipart-uploads \
  --endpoint-url https://<s3_api_хост> \
```

```
  --bucket $BUCKET_NAME \
| jq -r '.Uploads[] | "--key \"\(.Key)\" --upload-id \(.UploadId)"' \
| while read -r line; do
  eval
    "aws s3api list-parts \
      --endpoint-url https://<s3_api_хост> \
      --bucket $BUCKET_NAME \
      $line";
done
```

В списке могут оказаться части объектов, загрузка которых началась до выполнения предыдущего шага и закончилась после него. Если список не пуст, повторите шаги 4–5.

Python (boto3)

Выполните следующий код:

```
import boto3

bucket_name = '<имя_бакета>'
s3 = boto3.resource('s3',
    endpoint_url='https://<s3_api_хост>',
    aws_access_key_id='<идентификатор_ключа>',
    aws_secret_access_key='<секретный_ключ>')
bucket = s3.Bucket(bucket_name)

# Deleting all versions (works for non-versioned buckets too).
bucket.object_versions.delete()

# Aborting all multipart uploads, which also deletes all parts.
for multipart_upload in bucket.multipart_uploads.iterator():
    # Part uploads that are currently in progress may or may not succeed,
    # so it might be necessary to abort a multipart upload multiple times.
    while len(list(multipart_upload.parts.all())) > 0:
        multipart_upload.abort()
```

API

Воспользуйтесь методом S3 API `DeleteMultipleObjects`.

В консоли управления информация о занятом месте обновляется с задержкой в несколько минут.

Удаление частично загруженного объекта

Объект может быть загружен не полностью при использовании составной загрузки.

Чтобы удалить частично загруженный объект вручную:

AWS CLI

Выполните следующую команду:

```
aws --endpoint-url=https://<s3_api_хост> s3api abort-multipart-upload \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --upload-id <идентификатор_загрузки>
```

Если вы не знаете идентификатор загрузки (`number`), найдите его в списке загрузок.

```
aws --endpoint-url=https://<s3_api_хост> s3api list-multipart-uploads \
  --bucket <имя_бакета>
```

API

Воспользуйтесь методом S3 API `AbortMultipartUpload` .

Редактирование ACL объекта

Для управления доступом к объекту в бакете можно воспользоваться ACL.

> **Примечание**
>
> Если ранее для объекта уже был задан ACL, то после применения изменений ACL будет полностью перезаписан.

AWS CLI

> **Примечание**
>
> Чтобы управлять ACL объекта, назначьте учетной записи S3, через которую работает AWS CLI, роль `ADMIN`.

Посмотрите текущий ACL объекта:

```
aws s3api get-object-acl \
  --endpoint https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --key <ключ_объекта>
```

Где:

- `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--bucket` — имя бакета.
- `--key` — ключ объекта.

Вы можете применить к объекта предопределенный ACL или настроить разрешения для отдельных учетных записей S3 или публичных групп (группа всех пользователей, группа всех аутентифицированных пользователей). Эти настройки несовместимы: у объекта должен быть либо предопределенный ACL, либо набор отдельных разрешений.

Предопределенный ACL

Выполните команду:

```
aws s3api put-object-acl \
  --endpoint https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --acl <предопределенный_ACL>
```

Где:

- `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
- `--bucket` — имя бакета.
- `--key` — ключ объекта.
- `--acl` — предопределенный ACL.

Настройка отдельных разрешений

1. Чтобы выдать разрешения ACL для учетной записи S3, получите ее идентификатор.
2. Выполните команду:

   ```
   aws s3api put-object-acl \
     --endpoint https://<s3_api_хост> \
     --bucket <имя_бакета> \
     --key <ключ_объекта> \
     <тип_разрешения> <получатель_разрешения>
   ```

   Где:

   - `--endpoint` — эндпоинт S3 API ПО Яндекс Объектное хранилище.
   - `--bucket` — имя бакета.
   - `--key` — ключ объекта.
   - Возможные типы разрешений ACL:
     - `--grant-read` — доступ к чтению объекта.
     - `--grant-full-control` — полный доступ к объекту.
     - `--grant-read-acp` — доступ к чтению ACL объекта.
     - `--grant-write-acp` — доступ к редактированию ACL объекта.

     Вы можете задать несколько разрешений в одной команде.
   - Возможные получатели разрешений:
     - `id=<идентификатор_получателя>` — идентификатор учетной записи S3, которой нужно дать разрешение.
     - `uri=http://acs.amazonaws.com/groups/global/AuthenticatedUsers` — публичная группа всех аутентифицированных пользователей.
     - `uri=http://acs.amazonaws.com/groups/global/AllUsers` — публичная группа всех пользователей.

API

Чтобы отредактировать ACL объекта, воспользуйтесь методом S3 API `PutObjectAcl`.

Чтобы загрузить объект с установкой ACL, воспользуйтесь методом S3 API `PutObject` с заголовками `X-Amz-Acl`, `X-Amz-Grant-Read`, `X-Amz-Grant-Read-Acp`, `X-Amz-Grant-Write-Acp` и `X-Amz-Grant-Full-Control`.

Управление метками объекта

Метки объектов — это пары ключ-значение для логической маркировки объектов.

Добавить или изменить метки

AWS CLI

В терминале выполните команду, которая перезаписывает все имеющиеся у объекта метки:

```
aws s3api put-object-tagging \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --tagging 'TagSet=[{Key=<ключ_метки>,Value=<значение_метки>},{Key=
<ключ_метки>,Value=<значение_метки>}]' \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--key` — ключ объекта в бакете.
- `--tagging` — массив меток объекта, где:
  - `Key` — ключ метки, тип: `string`.
  - `Value` — значение метки, тип: `string`.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Чтобы добавить или изменить метки объекта, воспользуйтесь методом S3 API `PutObjectTagging`.

Посмотреть метки

AWS CLI

В терминале выполните команду:

```
aws s3api get-object-tagging \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--key` — ключ объекта в бакете.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

Результат:

```
{
    "TagSet": [
        {
            "Key": "test-key-1",
            "Value": "test-value-1"
        },
        {
            "Key": "test-key-2",
            "Value": "test-value-2"
        }
    ]
}
```

API

Чтобы посмотреть метки объекта, воспользуйтесь методом S3 API `GetObjectTagging`.

Удалить метки

AWS CLI

В терминале выполните команду:

```
aws s3api delete-object-tagging \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета.
- `--key` — ключ объекта в бакете.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Чтобы удалить метки объекта, воспользуйтесь методом S3 API `DeleteObjectTagging`.

Управление пользовательскими метаданными объекта

При загрузке объекта в бакет вместе с ним вы можете передавать набор пользовательских метаданных в виде пар `ключ-значение`.

Загрузить объект с метаданными

AWS CLI

В терминале выполните команду:

```
aws s3api put-object \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --body <путь_к_локальному_файлу> \
  --metadata "<ключ_1>"="<значение_1>","<ключ_2>"="<значение_2>" \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета, в который вы хотите загрузить объект.
- `--key` — ключ объекта в бакете.
- `--body` — путь к локальному файлу, который вы хотите загрузить в бакет.
- `--metadata` — набор пользовательских метаданных в виде пар `ключ-значение`, указанных через запятую. Ключи должны состоять только из ASCII-символов. Размер значения не должен превышать 2 КБ.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Воспользуйтесь методом S3 API `PutObject`, например, с помощью утилиты `curl`.

Используйте `curl` версии 8.3.0 и выше.

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
LOCAL_FILE="<путь_к_локальному_файлу>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"
META_KEY_1="<ключ_1>"
META_VALUE_1="<значение_1>"
META_KEY_2="<ключ_2>"
META_VALUE_2="<значение_2>"

curl \
  --request PUT \
  --user "${AWS_KEY_ID}:${AWS_SECRET_KEY}" \
  --aws-sigv4 "aws:amz:<region-id>:s3" \
  --upload-file "${LOCAL_FILE}" \
```

```
  --header "X-Amz-Meta-${META_KEY_1}: ${META_VALUE_1}" \
  --header "X-Amz-Meta-${META_KEY_2}: ${META_VALUE_2}" \
  --verbose \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где:

- `AWS_KEY_ID` — идентификатор статического ключа доступа.
- `AWS_SECRET_KEY` — секретный ключ.
- `LOCAL_FILE` — путь к локальному файлу, который вы хотите загрузить в бакет.
- `BUCKET_NAME` — имя бакета, в который вы хотите загрузить объект.
- `OBJECT_PATH` — ключ объекта в бакете.
- `META_KEY_1` и `META_KEY_1` — ключи пользовательских метаданных. Ключи должны состоять только из ASCII-символов.
- `META_VALUE_1` и `META_VALUE_2` — значения пользовательских метаданных. Размер значения не должен превышать 2 КБ.

Получить метаданные объекта

AWS CLI

В терминале выполните команду:

```
aws s3api head-object \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета, в котором размещен объект.
- `--key` — ключ объекта в бакете.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Воспользуйтесь методом S3 API `GetObjectMeta`, например, с помощью утилиты `curl`.

Используйте `curl` версии 8.3.0 и выше.

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"

curl \
  --head \
  --user "${AWS_KEY_ID}:${AWS_SECRET_KEY}" \
```

```
  --aws-sigv4 "aws:amz:<region-id>:s3" \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где:

- `AWS_KEY_ID` — идентификатор статического ключа доступа.
- `AWS_SECRET_KEY` — секретный ключ.
- `BUCKET_NAME` — имя бакета, в котором размещен объект.
- `OBJECT_PATH` — ключ объекта в бакете.

Изменить метаданные объекта

> **Важно**
>
> Существующий набор пользовательских метаданных будет полностью перезаписан новым.

AWS CLI

В терминале выполните команду:

```
aws s3api copy-object \
  --bucket <имя_бакета> \
  --key <ключ_объекта> \
  --copy-source <имя_бакета>/<ключ_объекта> \
  --metadata "<ключ_1>"="<значение_1>","<ключ_2>"="<значение_2>" \
  --metadata-directive REPLACE \
  --endpoint-url=https://<s3_api_хост>
```

Где:

- `--bucket` — имя бакета, в котором вы хотите изменить метаданные объекта.
- `--key` — ключ объекта в бакете.
- `--copy-source` — путь к объекту в формате `<имя_бакета>/<ключ_объекта>`.
- `--metadata` — набор новых пользовательских метаданных в виде пар `ключ-значение`, указанных через запятую. Ключи должны состоять только из ASCII-символов. Размер значения не должен превышать 2 КБ.
- `--metadata-directive` — параметр, который указывает, что нужно заменить метаданные объекта на новые.
- `--endpoint-url` — эндпоинт S3 API ПО Яндекс Объектное хранилище.

API

Воспользуйтесь методом S3 API `PutObjectCopy`, например, с помощью утилиты `curl`.

Используйте `curl` версии 8.3.0 и выше.

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"
META_KEY_1="<ключ_1>"
META_VALUE_1="<значение_1>"
META_KEY_2="<ключ_2>"
META_VALUE_2="<значение_2>"

curl \
  --request PUT \
  --user "${AWS_KEY_ID}:${AWS_SECRET_KEY}" \
  --aws-sigv4 "aws:amz:<region-id>:s3" \
  --header "X-Amz-Copy-Source: /${BUCKET_NAME}/${OBJECT_PATH}" \
  --header "X-Amz-Metadata-Directive: REPLACE" \
  --header "X-Amz-Meta-${META_KEY_1}: ${META_VALUE_1}" \
  --header "X-Amz-Meta-${META_KEY_2}: ${META_VALUE_2}" \
  --verbose \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где:

- `AWS_KEY_ID` — идентификатор статического ключа доступа.
- `AWS_SECRET_KEY` — секретный ключ.
- `BUCKET_NAME` — имя бакета, в котором вы хотите изменить метаданные объекта.
- `OBJECT_PATH` — ключ объекта в бакете.
- `META_KEY_1` и `META_KEY_1` — новые ключи пользовательских метаданных. Ключи должны состоять только из ASCII-символов.
- `META_VALUE_1` и `META_VALUE_2` — новые значения пользовательских метаданных. Размер значения не должен превышать 2 КБ.

Концепции ПО Яндекс Объектное хранилище

В ПО Яндекс Объектное хранилище используются следующие основные понятия:

Понятие	Описание
Бакет	Выделенная часть хранилища для пользовательских данных.
Объект	Файл с данными в произвольном формате.
Версионирование бакета	Средство хранения нескольких копий объекта в бакете.
Блокировка версии объекта (object lock)	Защита версии объекта от удаления или перезаписи.
Частичное изменение объекта	Механизм частичного изменения и дозаписи объектов.
Жизненные циклы объектов в бакете	Правила автоматического удаления объектов и изменения их класса хранилища.
Подписанный (pre-signed) URL	Способ, с помощью которого анонимному пользователю можно выдать доступ к операции с хранилищем.
Составная (multipart) загрузка	Способ загрузки больших объектов.
Список управления доступом	Механизм предоставления доступа к объектам и бакетам.
Политика доступа	Механизм управления правами на действия с объектами и бакетами.
Загрузка данных через HTML-форму	Способ загрузки объектов в хранилище из браузера. Позволяет анонимным пользователям загружать объекты в бакет.
Метки	Пары ключ-значение для логической маркировки бакетов.
Квоты и лимиты	Ограничения на использование ПО Яндекс Объектное хранилище.

Бакет

Бакет — это выделенная часть хранилища для пользовательских данных. Каждый бакет в инсталляции ПО Яндекс Объектное хранилище имеет уникальное название, по которому выполняются запросы к ПО Яндекс Объектное хранилище.

Данные в бакетах хранятся в виде объектов. Вы можете логически структурировать данные с помощью нескольких бакетов или с помощью папок (префиксов) внутри одного бакета.

Создать бакет можно с помощью API, а также популярных инструментов для работы с HTTP API Amazon S3.

Именование бакетов

Название бакета используется как часть URL для доступа к данным и его будут видеть ваши пользователи. Например, `https://<s3_api_хост>/bucket-name`.

Правила именования:

- Имена бакетов уникальны для всей инсталляции ПО Яндекс Объектное хранилище, т.е. нельзя создать два бакета с одинаковыми именами даже в разных тенантах. Помните об этом, если планируете создавать бакеты автоматически через API.
- На имя бакета накладываются следующие ограничения:
  - Длина имени должна быть от 3 до 63 символов.
  - Имя может содержать строчные буквы латинского алфавита, цифры, дефисы и точки.
  - Первый и последний символы должны быть буквами или цифрами.
  - Символы справа и слева от точки должны быть буквами или цифрами.
  - Имя не должно иметь вид IP-адреса (например `10.1.3.9`).

URL бакета

Для обращения к бакету вы можете использовать следующие формы URL:

- `http://<s3_api_хост>/<имя_бакета>?<параметры>`
- `http://<имя_бакета>.<s3_api_хост>?<параметры>`

Настройки бакетов

Вы можете:

- Ограничить максимальный размер бакета.

  ПО Яндекс Объектное хранилище не позволит загрузить объект, если при его добавлении размер бакета превысит максимальное значение.

Особенности использования

- Обновление статистики бакета может занимать до 20 минут. Поэтому возможны ситуации, при которых будет превышен заданный максимальный объем бакета, например, при быстрой последовательной загрузке объектов в бакет.

- В консоли управления информация о занятом месте обновляется с задержкой.
- Бакет нельзя переименовать.
- Производительность ПО Яндекс Объектное хранилище не зависит от количества бакетов. Вы можете хранить все свои данные в одном бакете или в нескольких.
- Бакеты не могут быть вложенными.
- Удалить можно только пустой бакет.
- После удаления объектов из бакета, удаленный объем еще некоторое время считается занятым.
- После удаления бакета может пройти некоторое время прежде, чем вы сможете создать новый бакет с этим же именем.

> **Примечание**
>
> Если вы ограничили максимальный объем бакета, то бакет может некоторое время оставаться недоступным для записи, даже если вы освободили достаточно места для новых объектов.

Объект

Объекты размещаются в бакетах и содержат пользовательские данные произвольного формата в том виде, в котором они были загружены.

Идентификатор объекта — строковый ключ.

Вместе с объектом ПО Яндекс Объектное хранилище хранит пользовательские и системные метаданные.

ПО Яндекс Объектное хранилище поддерживает следующие операции с объектами:

- Загрузка объекта в хранилище.
- Скачивание объекта из хранилища.
- Копирование объекта внутри хранилища, например, из бакета в бакет.
- Удаление объекта.
- Частичное изменение объекта в хранилище.

Все остальные операции, которые можно выполнить с помощью инструментов, являются комбинациями из указанных выше.

Для объектов в бакете можно настраивать жизненные циклы.

Ключ

Ключ — это идентификатор объекта в бакете.

Структура хранения объектов плоская, хотя инструменты с графическим интерфейсом предлагают работать с ПО Яндекс Объектное хранилище как с иерархической файловой системой. Видимость иерархического хранилища достигается за счет того, что ключи можно записывать как пути в файловой системе, например, `top_level_prefix/subprefix/text_data.txt`.

Ключ должен:

- иметь кодировку UTF-8;
- быть меньше 1024 байт;
- не содержать символов `: * ? " < > | !`.

Для использования в ключе безопасны следующие символы: `[a-zA-Z0-9]`, `-`, `_`, `/`, `\`. Другие символы могут вызвать различные проблемы при использовании ПО Яндекс Объектное хранилище.

Папка

В ПО Яндекс Объектное хранилище нет папок, однако графические файловые менеджеры, например, CyberDuck имитируют папки. В роли папки выступает объект с нулевым размером, ключ которого входит в ключи других объектов как префикс.

Каждый из инструментов управляет объектами и папками согласно своей логике, которая описана в документации на каждый из них.

URL объекта

Ссылку на объект в бакете можно указывать в одном из форматов:

- `http(s)://<бакет>.<s3_api_хост>/<ключ>?<параметры>`
- `https://<s3_api_хост>/<бакет>/<ключ>?<параметры>`

Где:

- `<бакет>` — имя бакета.
- `<ключ>` — ключ (путь к файлу).
- `<параметры>` — дополнительные параметры, необходимые для доступа к бакету. Например подпись и срок действия.

Метаданные

С объектом хранятся метаданные в виде пар `ключ-значение`.

Метаданные могут быть системными и пользовательскими.

Системные метаданные

Системные метаданные определяются ПО Яндекс Объектное хранилище.

Ключ	Описание
`Date`	Дата и время отправки запроса на загрузку объекта.
`Content-Length`	Размер объекта в байтах.
`Last-Modified`	Дата создания или последнего изменения объекта.
`Content-MD5`	MD5 хэш объекта, закодированный в base64.
`Cache-Control`	Значение HTTP заголовка `Cache-Control`, который клиент передает при сохранении объекта в бакет. В дальнейшем, ПО Яндекс Объектное хранилище возвращает этот заголовок клиентам при ответе на запрос объекта или его метаданных. Например, заголовок `Cache-Control: max-age=200` обозначает, что объект устаревает через 200 секунд после того, как клиент получил его. Подробнее о заголовке читайте в RFC 7234.

`Expires`	Значение HTTP заголовка `Expires`, который клиент передает при сохранении объекта в бакет. В дальнейшем, ПО Яндекс Объектное хранилище возвращает этот заголовок клиентам при ответе на запрос объекта или его метаданных. Например, заголовок `Expires: Thu, 15 Apr 2020 20:00:00 GMT` обозначает, что объект устаревает 15 апреля 2020 года в 20:00:00 по Гринвичу. Подробнее о заголовке читайте в RFC 7234.

заголовка должно начинаться с `X-Amz-Meta-`. При запросе объекта через HTTP API ПО Яндекс Объектное хранилище возвращает метаданные в виде HTTP-заголовков с этим же префиксом.

Ключи метаданных должны состоять только из ASCII-символов. Передаваемые заголовки будут преобразованы по правилу: `X-Amz-Meta-foo-bar_baz` → `X-Amz-Meta-Foo-Bar_baz`. Здесь `Foo-Bar_baz` – ключ метаданных, которые будут сохранены вместе с объектом.

Примечание

Заголовок PUT-запроса не должен превышать 8 КБ. Размер пользовательских метаданных в этом заголовке не должен превышать 2 КБ.

Версионирование бакета

Версионирование бакета — это возможность хранить историю объекта с помощью версий. Каждая версия является полной копией объекта и занимает соответствующий объем в хранилище. С помощью управления версиями вы можете защитить ваши данные как от непреднамеренных действий пользователя, так и от сбоев приложений.

Версионирование включается для бакета и применяется ко всем объектам внутри бакета.

- После включения этой функции к каждому загруженному объекту добавляется параметр `version_id`, который позволяет работать с конкретной версией объекта.
- До включения версионирования каждому объекту бакета присваивается идентификатор версии (`version_id`) равный `null`.

  После приостановки версионирования `version_id` существующих объектов не меняется. Каждому новому объекту будет присваиваться идентификатор версии `null`. Если версия `null` уже есть, она будет перезаписана.
- При перезаписи версии объекта создается новый объект с тем же идентификатором и случайно сгенерированным значением `version_id`.

  Для обращения к предыдущей версии объекта используется идентификатор объекта и необходимый `version_id`.
- При включенном версионировании предыдущие версии объектов можно восстанавливать.

> **Примечание**
>
> Операция включения необратима: отключить версионирование нельзя, можно только приостановить создание новых версий. После приостановки версионирования новые объекты будут сохраняться с версией `null`.

При удалении версия объекта помечается delete-маркером и не занимает места.

Очистить бакет от ненужных или удаленных версий объектов можно вручную или настроив жизненный цикл объектов.

Блокировка версии объекта (object lock)

Механизм *блокировки версии объекта* (object lock) в версионируемых бакетах позволяет запретить удаление или перезапись версии объекта. Блокировка обеспечивает хранение версии по принципу WORM (write once read many), но при этом не запрещает загружать новые версии объекта.

Включение механизма блокировок не устанавливает блокировки на уже загруженные версии объектов, а только позволяет их устанавливать. Выключение блокировок также не снимает установленные блокировки: они продолжают работать, их нельзя снять или изменить.

Блокировки делятся на типы в зависимости от сроков и строгости.

Устанавливать блокировки можно для конкретных версий объектов (при загрузке или после нее) и по умолчанию для всех новых версий, загружаемых в бакет.

Типы блокировок

Два типа блокировок устанавливаются временно — до определенной даты и времени:

Временная управляемая блокировка (governance-mode retention)

Пользователь с правами на загрузку объектов (роль `EDITOR`) может установить блокировку.

Пользователь с ролью `ADMIN` может обойти блокировку (удалить или перезаписать версию объекта), изменить срок блокировки или снять ее. Эти действия пользователь должен явно подтверждать: например, при запросе через REST API, совместимый с Amazon S3, — с помощью заголовка `X-Amz-Bypass-Governance-Retention: true`.

Временная строгая блокировка (compliance-mode retention)

Пользователь с правами на загрузку объектов (роль `EDITOR`) может установить блокировку.

Пользователь с ролью `ADMIN` может только продлить блокировку.

Обойти, сократить или снять блокировку до ее окончания нельзя.

Еще один тип блокировки устанавливается бессрочно:

Бессрочная блокировка (legal hold)

Пользователь с правами на загрузку объектов (роль `EDITOR`) может установить и снять блокировку.

Обойти блокировку нельзя.

Временные и бессрочная блокировка управляются независимо друг от друга. На версию объекта одновременно могут быть установлены одна временная и одна бессрочная блокировка. Пока они установлены вместе, бессрочная блокировка имеет приоритет: удалить или перезаписать версию нельзя, даже если временная блокировка разрешает это некоторым пользователям.

Таблица действий и ролей

Тип блокировки	Управляемая	Строгая (compliance)	♾ Бессрочная

	(governance)		(legal hold)
Кто может...			
...установить блокировку	`EDITOR`	`EDITOR`	`EDITOR`
...удалить или перезаписать версию объекта	`ADMIN`	Никто	Никто
...сократить блокировку	`ADMIN`	Никто	—
...продлить блокировку	`ADMIN`	`ADMIN`	—
...заменить одну временную блокировку на другую	`ADMIN`	Никто	—
...снять блокировку	`ADMIN`	Никто	`EDITOR`

Блокировки по умолчанию

Для бакета можно настроить *блокировки по умолчанию*: они будут устанавливаться на все новые версии объектов, загружаемые в бакет.

Для блокировок по умолчанию нужно указать:

- тип: временный управляемый или временный строгий;
- срок в днях или годах с момента загрузки версии объекта. Дата и время окончания блокировки рассчитываются для каждой версии автоматически.

Если для бакета настроены блокировки по умолчанию, для каждой загружаемой версии объекта нужно вычислить MD5-хеш, закодировать его по схеме Base64 и указать получившееся значение в запросе: например, при запросе через REST API — в заголовке `Content-MD5`.

Даже если для бакета настроены блокировки по умолчанию, при загрузке или после загрузки конкретной версии объекта можно указать другие настройки временной блокировки — они будут иметь приоритет. При этом загрузить версию без временной блокировки или снять ее после загрузки нельзя.

Версии объектов, уже загруженные в бакет, не затрагиваются изменениями настроек блокировок по умолчанию.

Частичное изменение объекта

В ПО Яндекс Объектное хранилище реализован механизм *частичного изменения и дозаписи объектов* в бакете.

> **Примечание**
>
> Механизм частичного изменения объекта не входит в стандартную функциональность S3 API и доступен для бакетов с выключенным версионированием.

С помощью частичного изменения вы можете хранить в бакете данные, например логи, в виде единого файла, периодически дописывая в него информацию.

Также упрощается работа с большими файлами. Например, чтобы изменить один байт информации в большом файле вы можете использовать:

- стандартные методы `GetObject` и `PutObject` S3 API, полностью скачав и загрузив объект обратно в хранилище;
- метод `PatchObject`, загрузив в хранилище только изменившуюся или новую часть файла.

Таким образом, частичная перезапись позволяет увеличить скорость работы с ПО Яндекс Объектное хранилище и уменьшить накладные расходы.

На стороне сервера при частичном изменении будут перезаписаны:

- объект целиком, если он изначально был загружен с помощью метода `PutObject`.
- составные части объекта, попадающие в диапазон изменения, если объект изначально был загружен по частям.

Указанная функциональность реализована в виде метода `PatchObject` и поддержана в GeeseFS.

GeeseFS является рекомендованным инструментом для частичного изменения объектов в бакете.

Также вы можете напрямую отправлять запросы `PatchObject` к API, либо расширить набор методов AWS SDK, воспользовавшись спецификацией метода `PATCH`.

Одновременное изменение объектов

В ПО Яндекс Объектное хранилище поддерживается одновременное частичное изменение объектов.

При частичном изменении каждый параллельный запрос выполняется на собственном снимке объекта, таким образом, разные запросы на изменение одного и того же объекта обрабатываются независимо.

Все изменения при параллельных запросах применяются атомарно.

Разрешение конфликтов

Для одновременного частичного изменения реализован механизм разрешения конфликтов.

При успешном разрешении конфликта применяется то изменение, которое было произведено позднее.

Количество попыток перезаписи ограничено на стороне сервера. Если серверу не удается разрешить конфликты, пользователю возвращается ответ с ошибкой `ConcurrentUpdatesPatchConflict`.

Спецификация

```
"PatchObject":{
  "name":"PatchObject",
  "http":{
    "method":"PATCH",
    "requestUri":"/{Bucket}/{Key+}"
  },
  "input":{"shape":"PatchObjectRequest"},
  "output":{"shape":"PatchObjectOutput"},
  "httpChecksum":{
    "requestAlgorithmMember":"ChecksumAlgorithm",
    "requestChecksumRequired":false
  }
},
"PatchAppendPartSize":{"type": "integer"},
"PatchedObjectInfo":{
  "type":"structure",
  "members":{
    "ETag":{"shape":"ETag"},
    "LastModified":{"shape":"LastModified"}
  }
},
"PatchObjectOutput":{
  "type":"structure",
  "members":{
    "Object":{"shape":"PatchedObjectInfo"}
  }
},
"PatchObjectRequest":{
  "type":"structure",
  "required":[
    "Bucket",
    "Key",
    "ContentRange"
  ],
  "members":{
    "Body":{
      "shape":"Body",
      "streaming":true
    },
    "Bucket":{
      "shape":"BucketName",
      "location":"uri",
```

```
      "locationName":"Bucket"
    },
    "ContentLength":{
      "shape":"ContentLength",
      "location":"header",
      "locationName":"Content-Length"
    },
    "ContentMD5":{
      "shape":"ContentMD5",
      "location":"header",
      "locationName":"Content-MD5"
    },
    "ContentRange":{
      "shape":"ContentRange",
      "location":"header",
      "locationName":"Content-Range"
    },
    "IfMatch":{
      "shape":"IfMatch",
      "location":"header",
      "locationName":"If-Match"
    },
    "IfUnmodifiedSince":{
      "shape":"IfUnmodifiedSince",
      "location":"header",
      "locationName":"If-Unmodified-Since"
    },
    "Key":{
      "shape":"ObjectKey",
      "location":"uri",
      "locationName":"Key"
    },
    "PatchAppendPartSize":{
      "shape":"PatchAppendPartSize",
      "location":"header",
      "locationName":"X-Yc-S3-Patch-Append-Part-Size"
    }
  },
  "payload":"Body"
},
```

Жизненные циклы объектов в бакете

С помощью *жизненных циклов* вы можете настроить действия, которые автоматически будут выполняться с отдельными объектами или группами объектов в бакете в заданные моменты времени.

Виды действий:

- Удаление объектов или их неактивных версий.
- Удаление незавершенных составных загрузок.

Фильтры для группировки объектов:

- Префикс ключа объекта.
- Минимальный или максимальный размер объекта.
- Метка объекта.
- Логический оператор **И** (`AND`). Позволяет группировать объекты с помощью сочетания нескольких фильтров.

В одном правиле жизненного цикла можно задать только один фильтр. Чтобы одновременно задать для жизненного цикла более одного типа фильтра, используйте логический оператор **И** (`AND`).

Если вы используете S3 API, задайте конфигурацию жизненного цикла в формате XML. Для различных инструментов с поддержкой S3 API могут понадобиться другие форматы конфигурации, например для AWS CLI применяется формат JSON.

Задать конфигурацию жизненных циклов объектов можно только для каждого отдельного бакета. Не получится задать конфигурацию жизненных циклов для группы бакетов, тенанта.

Раз в сутки к жизненным циклам применяются изменения, актуальные на момент 00:00 UTC. Операция выполняется в течение нескольких часов.

CORS

ПО Яндекс Объектное хранилище поддерживает кросс-доменные запросы к объектам в бакете.

Для управления конфигурацией CORS для каждого бакета можно применять:

- HTTP API, совместимый с Amazon S3.

  Таким образом, конфигурацией CORS можно управлять при помощи инструментов, поддерживающих HTTP API Amazon S3.

При управлении конфигурацией CORS с помощью HTTP API, совместимого с Amazon S3, задайте ее в формате XML.

Подписанные (pre-signed) URL

С помощью подписанных URL произвольный пользователь может выполнять в ПО Яндекс Объектное хранилище различные операции, например:

- Скачать объект.
- Загрузить объект.
- Создать бакет.

Подписанный URL — это URL, содержащий в своих параметрах данные для авторизации запроса. Составить подписанный URL может пользователь, обладающий статическими ключами доступа.

Раздел содержит общие принципы построения подписанного URL с использованием AWS Signature V4.

> **Примечание**
>
> SDK для различных языков программирования и другие инструменты для работы с AWS S3 содержат готовые методы генерирования подписанного URL, которые можно использовать и для ПО Яндекс Объектное хранилище.

Общий вид подписанного URL

```
https://<имя_бакета>.<s3_api_хост>/<ключ_объекта>?
     X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=<access_key-id>%2F<YYYYMMDD>%2F<region-id>%2Fs3%2Faws4_request
    &X-Amz-Date=<время_в_формате_ISO_8601>
    &X-Amz-Expires=<время_жизни_ссылки>
    &X-Amz-SignedHeaders=<список_подписанных_заголовков>
    &X-Amz-Signature=<подпись>
```

Параметры подписанного URL:

Параметр	Описание
`X-Amz-Algorithm`	Идентифицирует версию подписи и алгоритм ее вычисления. Значение — `AWS4-HMAC-SHA256`.
`X-Amz-Credential`	Идентификатор для подписи. Строка формата `<access-key-id>/<YYYYMMDD>/<region-id>/s3/aws4_request`, где `<YYYYMMDD>` должна совпадать с датой, установленной в заголовке `X-Amz-Date`.

`X-Amz-Date`	Время в формате ISO8601, например, `20180719T000000Z`. Указанная дата должна по значению (не по формату) совпадать с датой в параметре `X-Amz-Credential`.
`X-Amz-Expires`	Время в секундах, в течение которого ссылка действительна. Начало отсчета — момент, указанный в `X-Amz-Date`. Максимальное значение — 2592000 секунд (30 дней).
`X-Amz-SignedHeaders`	Заголовки запроса, которые вы хотите подписать, разделенные точкой с запятой (`;`). Обязательно подписывайте заголовок `Host` и все заголовки `X-Amz-*`, которые используются в запросе. Другие заголовки подписывать не обязательно, однако чем больше вы подпишете заголовков, тем безопаснее будет запрос.
`X-Amz-Signature`	Подпись запроса.

Составление подписанного URL

Чтобы получить подписанный URL:

1. Составьте канонический запрос.
2. Составьте строку для подписи.
3. Сформируйте подписывающий ключ.
4. Вычислите подпись с помощью ключа.
5. Сформируйте подписанный URL.

Для составления подписанного URL необходимо обладать статическими ключами доступа.

Канонический запрос

Общий вид канонического запроса:

```
<HTTPVerb>\n
<CanonicalURL>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
UNSIGNED-PAYLOAD
```

HTTPVerb

HTTP метод, которым будет отправлен запрос: `GET`, `PUT`, `HEAD` или `DELETE`.

CanonicalURL

URL-кодированный ключ объекта. Например, `/folder/object.ext`.

Примечание

Не нормализуйте путь. Например, объект может иметь ключ `some//strange//key//example` и нормализация пути `/<bucket-name>/some/strange/key/example` сделает ключ некорректным.

CanonicalQueryString

Каноническая строка запроса должна включать все query параметры конечного URL, кроме `X-Amz-Signature`. Параметры в строке должны быть URL-кодированы и отсортированы по алфавиту.

Пример:

```
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=JK38EXAMPLEAKDID8%2F20190801%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=20190801T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
```

CanonicalHeaders

Список заголовков запроса со значениями.

Требования:

- Каждый заголовок отделяется символом новой строки `\n`.
- Имена заголовков должны быть в нижнем регистре.
- Заголовки должны быть отсортированы по алфавиту.
- Не должно быть лишних пробелов.
- Список должен содержать заголовок `host` и все заголовки `x-amz-*`, которые используются в запросе.

Дополнительно в список можно добавить любой из заголовков запроса. Чем больше вы подпишете заголовков, тем безопаснее будет запрос.

Пример:

```
host:sample-bucket.<s3_api_хост>
x-amz-date:20190801T000000Z
```

SignedHeaders

Список имен заголовков запроса в нижнем регистре, отсортированный по алфавиту и разделенный точками с запятыми.

Пример:

```
host;x-amz-date
```

Завершение канонического запроса

Завершать канонический запрос всегда должна строка `UNSIGNED-PAYLOAD`.

Строка для подписи

Общий вид строки для подписи:

```
"AWS4-HMAC-SHA256" + "\n" +
<timestamp> + "\n" +
<scope> + "\n" +
Hex(Hash-SHA256(<CanonicalRequest>))
```

Где:

- `AWS4-HMAC-SHA256` — алгоритм хэширования.
- `timestamp` — текущее время в формате ISO 8601, например, `20190801T000000Z`. Указанная дата должна по значению (не по формату) совпадать с датой в `scope`.
- `scope` — `<YYYYMMDD>/<region-id>/s3/aws4_request`.
- `CanonicalRequest` — сформированный ранее канонический запрос. В строку для подписи помещается SHA256-хэш канонического запроса в шестнадцатеричном представлении.

Подписывающий ключ

Чтобы сгенерировать подписывающий ключ:

1. Закодируйте дату с помощью секретного ключа:

   ```
   DateKey = sign("AWS4" + "SecretKey", "yyyymmdd")
   ```

2. Закодируйте регион с помощью полученного на предыдущем шаге ключа `DateKey`:

   ```
   RegionKey = sign(DateKey, "<region-id>")
   ```

3. Закодируйте сервис с помощью полученного на предыдущем шаге ключа `RegionKey`:

   ```
   ServiceKey = sign(RegionKey, "s3")
   ```

4. Получите подписывающий ключ:

   ```
   SigningKey = sign(ServiceKey, "aws4_request")
   ```

Подпись строки с помощью ключа

Чтобы получить подпись строки, необходимо использовать механизм `HMAC` с хэширующей функцией `SHA256`, а полученный результат преобразовать в шестнадцатеричное представление.

```
signature = Hex(sign(SigningKey, StringToSign))
```

Подписанный URL

Чтобы составить подписанный URL, к URL ресурса добавьте параметры, необходимые для авторизации запроса, в том числе параметр `X-Amz-Signature` с вычисленной подписью в значении.

Значения остальных параметров должны совпадать с аналогичными значениями, указанными ранее в каноническом запросе и в строке для подписи.

Пример составления подписанного URL для скачивания объекта

Составим подписанный URL для скачивания объекта `object-for-share.txt` из бакета в течение часа:

- Статический ключ:

  ```
  access_key_id = 'JK38EXAMP********'
  secret_access_key = 'ExamP1eSecReTKeykdo********'
  ```

- Канонический запрос:

  ```
  GET
  /object-for-share.txt
  X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
  Credential=YCAJEK0Iv6x********eLTAdg%2F20231208%2F<region-
  id>%2Fs3%2Faws4_request&X-Amz-Date=20231208T184504Z&X-Amz-Expires=3600&X-Amz-
  SignedHeaders=host
  host:<имя_бакета>.<s3_api_хост>

  host
  UNSIGNED-PAYLOAD
  ```

- Строка для подписи:

  ```
  AWS4-HMAC-SHA256
  20231208T184504Z
  20231208/<region-id>/s3/aws4_request
  e823d75aad02c1317589bd5373fe9e20d5ef44499237703ff23e5600********
  ```

- Подписывающий ключ:

```
sign(sign(sign(sign("AWS4" + "ExamP1eSecReTKeykdokKK38800","20190801"),"<region-id>"),"s3"),"aws4_request")
```

Функция `sign` введена для обозначения способа вычисления ключа с помощью механизма HMAC с хэширующей функцией SHA256.

- Подпись:

```
b10c16a1997bb524bf59974512f1a6561cf2953c29dc3efbdb920790********
```

- Подписанный URL:

```
https://<имя_бакета>.<s3_api_хост>/object-for-share.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=YCAJEK0Iv6xqy-pEQcueLTAdg%2F20231208%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=20231208T195434Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=b10c16a1997bb524bf59974512f1a6561cf2953c29dc3efbdb920790********
```

Примеры кода для генерации подписанных URL

В подразделе приведены примеры кода для генерации подписанных URL.

Python

```
import datetime
import hashlib
import hmac

access_key = '<идентификатор_статического_ключа>'
secret_key = '<содержимое_статического_ключа>'
object_key = '<ключ_объекта>'
bucket = '<имя_бакета>'
host = '<s3_api_хост>'
now = datetime.datetime.now(datetime.UTC)
datestamp = now.strftime('%Y%m%d')
timestamp = now.strftime('%Y%m%dT%H%M%SZ')

def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

canonical_request = """GET
/{bucket}/{object_key}
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=
{access_key}%2F{datestamp}%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=
{timestamp}&X-Amz-Expires=3600&X-Amz-SignedHeaders=host
host:{host}

host
```

```
UNSIGNED-PAYLOAD""".format(
    bucket=bucket,
    object_key=object_key,
    access_key=access_key,
    datestamp=datestamp,
    timestamp=timestamp,
    host=host)

print()
print("Canonical request:\n" + canonical_request)
print()

string_to_sign = """AWS4-HMAC-SHA256
{timestamp}
{datestamp}/<region-id>/s3/aws4_request
{request_hash}""".format(
    timestamp=timestamp,
    datestamp=datestamp,
    request_hash=hashlib.sha256(canonical_request.encode('utf-8')).hexdigest())

print()
print("String to be signed:\n" + string_to_sign)
print()

signing_key = sign(sign(sign(sign(('AWS4' + secret_key).encode('utf-8'), datestamp),
'<region-id>'), 's3'), 'aws4_request')
signature = hmac.new(signing_key, string_to_sign.encode('utf-8'),
hashlib.sha256).hexdigest()

print()
print("Signature: " + signature)
print()

signed_link = "https://" + host + '/' + bucket + '/' + object_key + "?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=" + access_key + "%2F" + datestamp +
"%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=" + timestamp + "&X-Amz-
Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=" + signature + "\n"

print()
print("Signed Link:\n" + signed_link)
```

PHP

```
<?php

  date_default_timezone_set('UTC');
  $keyid = "<идентификатор_статического_ключа>";
  $secretkey = "<содержимое_статического_ключа>";
  $path = "<ключ_объекта>";
  $objectname = "/".implode("/", array_map("rawurlencode", explode("/", $path)));
  $host = "<имя_бакета>.<s3_api_хост>";
```

```
  $region = "<region-id>";
  $timestamp = time();
  $dater = strval(date('Ymd', $timestamp));
  $dateValue = strval(date('Ymd', $timestamp))."T".strval(date('His',
$timestamp))."Z";

  // Generate the canonical request
  $canonical_request = "GET\n".$objectname."\nX-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=".$keyid."%2F".$dater."%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=".$dateValue."&X-Amz-Expires=3600&X-Amz-
SignedHeaders=host\nhost:".$host."\n\nhost\nUNSIGNED-PAYLOAD";

  echo "<b>Canonical request: </b><br>".$canonical_request."<br><br>";

  // Generate the string to be signed
  $string_to_sign = "AWS4-HMAC-
SHA256\n".$dateValue."\n".$dater."/".$region."/s3/aws4_request\n".openssl_digest($canon
"sha256", $binary = false);

  echo "<b>String to be signed: </b><br>".$string_to_sign."<br><br>";

  // Generate the signing key
  $signing_key = hash_hmac('sha256', 'aws4_request', hash_hmac('sha256', 's3',
hash_hmac('sha256', '<region-id>', hash_hmac('sha256', $dater, 'AWS4'.$secretkey,
true), true), true), true);

  echo "<b>Signing key: </b><br>".$signing_key."<br><br>";

  // Generate the signature
  $signature = hash_hmac('sha256', $string_to_sign, $signing_key);

  echo "<b>Signature: </b><br>".$signature."<br><br>";

  // Generate the pre-signed link
  $signed_link = "https://".$host.$objectname."?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-
Amz-Credential=".$keyid."%2F".$dater."%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=".$dateValue."&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-
Signature=".$signature."\n";

  echo '<b>Signed link: </b><br>'.'<a href = "'.$signed_link.'" target =
"_blank">'.$signed_link.'</a>';

?>
```

Примеры получения подписанной ссылки в инструментах ПО Яндекс Объектное хранилище

В подразделе приведены примеры генерации подписанных URL с помощь различных инструментов ПО Яндекс Объектное хранилище.

AWS CLI

Ссылку на скачивание объекта можно сгенерировать с помощью AWS CLI. Для этого выполните команду:

```
aws s3 presign s3://<имя_бакета>/<ключ_объекта> \
  --expires-in <время_жизни_ссылки> \
  --endpoint-url "https://<s3_api_хост>/"
```

Чтобы ссылка сформировалась корректно, обязательно укажите параметр `--endpoint-url` с указанием на доменное имя S3 API ПО Яндекс Объектное хранилище.

Python (boto3)

Пример генерирует подписанный URL для скачивания объекта `object-for-share` из бакета `bucket-with-objects`. URL действителен в течение 100 секунд.

```
# coding=utf-8

import boto3
from botocore.client import Config


ENDPOINT = "https://<s3_api_хост>"

ACCESS_KEY = "JK38EXAMP********"
SECRET_KEY = "ExamP1eSecReTKeykdo********"

session = boto3.Session(
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,
    region_name="<region-id>",
)
s3 = session.client(
    "s3", endpoint_url=ENDPOINT, config=Config(signature_version="s3v4")
)

presigned_url = s3.generate_presigned_url(
    "get_object",
    Params={"Bucket": "bucket-with-objects", "Key": "object-for-share"},
    ExpiresIn=100,
)

print(presigned_url)
```

Составная (multipart) загрузка

ПО Яндекс Объектное хранилище позволяет сохранять объекты размером в несколько терабайт. Чем больше объект, тем больше вероятность, что загрузка прервется ошибкой сети. Поэтому большие объекты эффективнее загружать частями меньшего размера. Такой способ сохранения объектов в ПО Яндекс Объектное хранилище называется составная загрузка.

Удалить незавершенную загрузку можно вручную методом `AbortMultipartUpload` S3 API, также вы можете настроить для бакета автоматическое удаление незавершенных загрузок в определенный момент времени — для этого создайте жизненный цикл (действие `AbortIncompleteMultipartUpload`).

Список управления доступом (ACL)

ACL в ПО Яндекс Объектное хранилище — список разрешений для каждого объекта и бакета.

Разрешения, выданные на бакет, распространяются на все находящиеся в нем объекты. Также ACL позволяет расширить доступы к отдельному объекту.

По умолчанию для каждого нового объекта или бакета создается пустой ACL. Пользователь, обладающий соответствующими правами, может отредактировать или загрузить ACL для бакетов и объектов.

С помощью ACL можно выдать разрешения учетной записи S3 или публичной группе (группе всех пользователей, группе всех аутентифицированных пользователей), при этом необходимо знать идентификатор получателя разрешений. При выдаче разрешений вы можете использовать предопределенные ACL, которые содержат популярные наборы доступов.

Описание структуры ACL смотрите в разделе XML-схема ACL. В одном ACL вы можете задать не более 100 правил.

Идентификатор получателя разрешений

- Учетная запись S3

  Для получения идентификатора в консоли управления на панели слева выберите **Тенанты**, перейдите в нужный тенант, выберите вкладку **Учетные данные S3** и выберите нужную учетную запись.
- Публичная группа

  Для выдачи разрешений используйте URI публичной группы.

Операции с ACL

- С помощью API, совместимого с Amazon S3, вы можете загрузить или скачать ACL для бакета или объекта.

  Удалить ACL невозможно. Чтобы удалить все разрешения, загрузите пустой ACL.

Виды разрешений

Разрешения соответствуют ролям учетных записей S3 в тенанте.

Название разрешения	Роль	Описание
`READ`	`VIEWER`	Для бакета: разрешение на получение списка объектов в бакете, чтение различных настроек бакета (жизненный цикл, CORS), чтение всех объектов в бакете. Для объекта: разрешение на чтение.

`WRITE`	`EDITOR`	Для бакета: запись, перезапись и удаление объектов в бакете. Используется обязательно вместе с `READ`, отдельно указать разрешение `WRITE` нельзя. Для объекта: разрешение не имеет смысла, при записи объекта проверяются разрешения для бакета.
`FULL_CONTROL`	`ADMIN`	Полный доступ к объектам и бакетам.
`READ_ACP`	`VIEWER`	Разрешение на чтение ACL. Только для объектов.
`WRITE_ACP`	`EDITOR`	Разрешение на запись ACL. Только для объектов.

Если при оформлении ACL указать доступ `WRITE`, но при этом не указать `READ`, то ПО Яндекс Объектное хранилище ответит с кодом `501 Not Implemented`.

Предопределенные ACL

ACL	Описание
`private` `bucket-owner-full-control`	Учетные записи S3 получают разрешения в соответствии со своими ролями в тенанте.
`public-read`	Публичная группа `AllUsers` получает разрешение `READ`.
`public-read-write`	Публичная группа `AllUsers` получает разрешения `READ` и `WRITE`.
`authenticated-read`	Публичная группа `AuthenticatedUsers` получает разрешение `READ`.

Предопределенные ACL могут применяться как к объектам, так и к бакетам. ACL `public-read-write`, примененный к объекту, эквивалентен `public-read`.

Загрузить предопределенный ACL можно только с помощью HTTP API, совместимого с Amazon S3. При загрузке ACL используйте HTTP-заголовок `X-Amz-Acl`.

Публичные группы

AllUsers

Включает в себя всех пользователей.

Разрешение для `AllUsers` выглядит следующим образом:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
</Grantee>
```

AuthenticatedUsers

Включает в себя всех аутентифицированных пользователей: как из вашего тенанта, так и из других тенантов.

Разрешение для `AuthenticatedUsers` выглядит следующим образом:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers</URI>
</Grantee>
```

Политика доступа (bucket policy)

Политики доступа устанавливают права на действия с бакетами, объектами и группами объектов.

Политика срабатывает, когда пользователь делает запрос к какому-либо ресурсу. В результате срабатывания политики запрос либо выполняется, либо отклоняется.

> **Примечание**
>
> Если к бакету применена политика доступа без правил, то доступ будет запрещен всем пользователям. Чтобы отключить проверки запросов по политике доступа, удалите ее.

Политику доступа можно описать в формате JSON по специальной схеме, чтобы передать через один из программных инструментов — AWS CLI или API.

Из чего состоит политика

Политика доступа состоит из правил, а правило — из следующих базовых элементов:

Ресурс

Бакет, объект в бакете (`<имя_бакета>/some/key`) или префикс (`<имя_бакета>/some/path/*`), в том числе пустой префикс для обозначения всех объектов в бакете (`<имя_бакета>/*`). В правиле можно указать несколько ресурсов.

> **Примечание**
>
> Ресурс бакета не включает в себя ресурсы всех его объектов. Чтобы правило в политике доступа относилось к бакету и всем объектам, их нужно указать как отдельные ресурсы: например, `samplebucket` и `samplebucket/*`.

Если вы описываете политику в формате JSON, у ресурса должен быть префикс `arn:aws:s3:::`, например `arn:aws:s3:::<имя_бакета>`.

Если имя ресурса содержит символ `?`, `*` или `$`, поставьте перед таким символом знак `$`, а сам символ заключите в фигурные скобки `{}`. Например, имени бакета `my?bucket` будет соответствовать запись `my${?}bucket`.

Действие

Набор операций над ресурсом, который будет запрещен или разрешен правилом.

Результат

Запрет или разрешение запрошенного действия. Сначала проверяется попадание запроса в фильтр с действием `Deny`, при совпадении запрос отклоняется и дальнейшие проверки не проводятся. При попадании в фильтр с действием `Allow` запрос разрешается. Если запрос не попал ни в один из фильтров, то запрос отклоняется.

Принципал

Получатель запрошенного разрешения. Это может быть учетная запись S3 или анонимный пользователь.

Условие

Определение случаев, когда действует правило.

Если для правила задано одновременно несколько условий или внутри одного условия задано одновременно несколько ключей, то такие условия и ключи применяются с логикой `и`.

Если для одного ключа условия задано одновременно несколько значений, то такие значения применяются с логикой `или`.

Примеры конфигурации

- Правило, которое разрешает скачивать объекты только из указанного диапазона IP-адресов:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<имя_бакета>/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "100.101.102.128/30"
        }
      }
    }
  ]
}
```

- Правило, которое запрещает скачивать объекты с указанного IP-адреса:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "*",
      "Resource": "arn:aws:s3:::<имя_бакета>/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<имя_бакета>/*",
```

```
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "100.101.102.103"
        }
      }
    }
  ]
}
```

- Правило, которое дает разным пользователям полный доступ только к определенным папкам, каждому пользователю — к своей:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"User1PermissionsResource",
      "Effect":"Allow",
      "Principal": {
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "*",
      "Resource":["arn:aws:s3:::<имя_бакета>/user1path/*"]
    },
    {
      "Sid":"User1PermissionsPrefix",
      "Effect":"Allow",
      "Principal": {
          "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "s3:ListBucket",
      "Resource":["arn:aws:s3:::<имя_бакета>"],
      "Condition": {
        "StringLike": {
          "s3:prefix": "user1path/*"
        }
      }
    },
    {
      "Sid":"User2PermissionsResource",
      "Effect":"Allow",
      "Principal": {
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "*",
      "Resource":["arn:aws:s3:::<имя_бакета>/user2path/*"]
    },
    {
      "Sid":"User2PermissionsPrefix",
      "Effect":"Allow",
      "Principal": {
```

```
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "s3:ListBucket",
      "Resource":["arn:aws:s3:::<имя_бакета>"],
      "Condition": {
        "StringLike": {
          "s3:prefix": "user2path/*"
        }
      }
    }
  ]
}
```

Загрузка файла с помощью HTML формы

Раздел содержит информацию о том, как из браузера загрузить файлы в ПО Яндекс Объектное хранилище с помощью HTML формы.

> **Примечание**
>
> Через форму нельзя загрузить объекты размером более 5 ГБ.
> Поля в форме после “file” игнорируются.

Общее описание

Если вы хотите, чтобы пользователи вашего сервиса могли загружать файлы в ваш бакет напрямую из браузера, то:

1. Вы разрабатываете HTML-форму, в которой есть все необходимое для отправки запроса в ПО Яндекс Объектное хранилище и помещаете ее на страницу вашего сервиса.
2. Пользователь открывает в браузере страницу вашего сервиса и с помощью формы загружает файл в хранилище.

Чтобы устанавливать правила и ограничения на загрузку файлов, к форме необходимо приложить политику безопасности.

Для создания формы выполните следующие действия:

1. Разработайте политику безопасности, которая описывает параметры запроса к ПО Яндекс Объектное хранилище. Например, политика может ограничивать размер загружаемого объекта.
2. На основании политики безопасности сгенерируйте подпись.
3. Создайте HTML-форму с подписанной политикой безопасности, которую вы будете предлагать пользователям для загрузки файлов.

HTML-форма

Общий вид HTML страницы с формой для загрузки файла:

```
<html>
    <head>
        ...
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        ...
    </head>
    <body>
        ...
        <form action="https://<s3_api_хост>/{bucket-name}" method="post"
enctype="multipart/form-data">
            ...
            <input .../>
```

```
            ...
        </form>
        ...
    </body>
</html>
```

HTML-форма описывается тегом `<form>` и состоит из объявления и полей.

Объявление формы содержит атрибуты:

- `action` — URL бакета, в который необходимо загрузить объект.
- `method` — HTTP метод. Значение — `POST`.
- `enctype` — Тип содержимого запроса. Значение — `multipart/form-data`.

Поля формы содержат подробное описание запроса к ПО Яндекс Объектное хранилище и ограничений, которые применяются к этому запросу.

Форма и ее поля должны быть в кодировке UTF-8. Установите атрибут `charset` тега `<meta>` страницы в значение `UTF-8`.

```
<html>
    <head>
        ...
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        ...
```

Поля формы

ПО Яндекс Объектное хранилище поддерживает механизмы подписи формы AWS Signature V2 и V4. От механизма подписи зависят названия и состав полей формы. AWS Signature V2 поддержана только для совместимости, по возможности не используйте ее.

Общий вид формы:

AWS Signature V4

```
<form action="https://<s3_api_хост>/{bucket-name}" method="post"
enctype="multipart/form-data">
   Ключ в хранилище:
   <input type="input" name="key" value="object_key" /><br />
   <!-- Свойства запроса -->
   <input type="hidden" name="X-Amz-Credential" value="access_key_id/date/<region-
id>/s3/aws4_request" />
   <input type="hidden" name="acl" value="predefined-acl-name" />
   <input type="hidden" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
   <input type="hidden" name="X-Amz-Date" value="date" />
   <input type="hidden" name="success_action_redirect" value="some-URL" />
   <input type="hidden" name="policy" value="base64-encoded-policy-document" />
   <input type="hidden" name="X-Amz-Signature" value="signature-string" />
   <!-- Прочие необходимые поля -->
```

```
    Файл для загрузки:
    <input type="file" name="file" /> <br />
    <!-- Поля после “file” игнорируются -->
    <input type="submit" name="submit" value="Загрузить" />
</form>
```

AWS Signature V2

```
 <form action="https://<s3_api_хост>/{bucket-name}" method="post"
enctype="multipart/form-data">
      Ключ в хранилище:
      <input type="input" name="key" value="object_key" />
      <!-- Свойства запроса -->
      <input type="hidden" name="AWSAccessKeyId" value="access_key_id" />
      <input type="hidden" name="acl" value="access_type" />
      <input type="hidden" name="success_action_redirect" value="url" />
      <input type="hidden" name="policy" value="base64-encoded-policy-document" />
      <input type="hidden" name="signature" value="signature_string" />
      <input type="hidden" name="Content-Type" value="content/type" />
      <!-- Прочие необходимые поля -->
      Файл для загрузки:
      <input type="file" name="file" /> <br />
      <!-- Поля после “file” игнорируются -->
      <br />
      <input type="submit" value="Загрузить файл" />
</form>
```

Примечание

Дальнейшее описание актуально только при использовании AWS Signature V4.

Описание полей формы:

Поле	Описание	Обязательное
`acl`	ACL для объекта. Можно установить один из предопределенных ACL. Например, если вы хотите сделать объект публичным, используйте значение `public-read`.	Нет
`Cache-Control`	Набор директив для кэширования данных согласно RFC 2616.	Нет
`Content-Disposition`	Имя, под которым ПО Яндекс Объектное хранилище предложит сохранить объект как файл при выгрузке. Соответствует RFC 2616.	Нет

`Content-Encoding`	Определяет кодировку содержимого согласно RFC 2616.	Нет
`Content-Type`	MIME тип загружаемого файла. Если не указать `Content-Type`, то ПО Яндекс Объектное хранилище сохраняет объект с типом `application/octet-stream`. В дальнейшем это может создать сложности для пользовательских программ, поскольку они не будут понимать формат файла, например браузер не сможет отобразить изображение.	Нет
`Expires`	Срок устаревания ответа. Соответствует RFC 2616.	Нет
`key`	Ключ объекта. Можно указать ключ целиком или шаблон вида `prefix/${filename}`, т.е. если вы загружаете файл `some_file.jpg`, то итоговый ключ объекта будет `prefix/some_file.jpg`.	Да
`policy`	Политика безопасности, описывающая разрешения запроса.	Условно
`X-Amz-Signature`	Подпись политики, которую необходимо сгенерировать с помощью секретного ключа. Необходимо, если в форме есть политика безопасности.	Условно
`success_action_redirect`	URL, на который перенаправляется пользователь после успешной загрузки файла. Если значение не установлено, то ПО Яндекс Объектное хранилище возвращает ответ, установленный в поле `success_action_status`.	Нет
`success_action_status`	Статус ответа при успешной загрузке. Если не указан `success_action_redirect`, то ПО Яндекс Объектное хранилище возвращает значение `success_action_status`. Тело ответа пустое. Допустимые значения: 200, 204 (по умолчанию).	Нет
`X-Amz-Algorithm`	Алгоритм для подписи политики безопасности. Значение — `AWS4-HMAC-SHA256`. Необходимо, если в форме есть политика безопасности.	Условно

`X-Amz-Credential`	Идентификатор для подписи. Строка формата `<access-key-id>/<date>/<region-id>/s3/aws4_request`, где `<date>` должна совпадать со значением поля `X-Amz-Date` и той датой, которая используется для подписи политики. Необходимо, если в форме есть политика безопасности.	Условно
`X-Amz-Date`	Дата в формате ISO8601, например, `20180719T000000Z`. Должна по значению (не по формату) совпадать с датой в поле `X-Amz-Credential`, а также с датой, которая используется для подписи политики. Необходимо, если в форме есть политика безопасности.	Условно
`X-Amz-Meta-*`	Пользовательские метаданные объекта. Все заголовки, начинающиеся с `X-Amz-Meta-` ПО Яндекс Объектное хранилище воспринимает как пользовательские, не обрабатывает их и сохраняет в том виде, в котором они переданы. Общий размер пользовательских заголовков не должен превышать 2KB. Размер пользовательских данных определяется как длина строки в кодировке UTF-8. В размере учитываются и названия заголовков и их значения.	Нет
`file`	Поле ввода, позволяющее пользователю выбрать файл для загрузки. Поле должно быть последним в форме. Все поля, указанные после `file`, игнорируются. Нельзя загрузить более одного файла в одном запросе.	Да

```
        ["starts-with", "$key", "users-uploads/"],
        {"acl": "public-read"},
        {"success_action_redirect": "http://localhost/"},
        ["starts-with", "$Content-Type", ""],
        ["content-length-range", 0, 1048576]
    ]
}
```

Поле `expiration` содержит срок действия политики в формате ISO8601, например, `2019-07-22T15:39:36Z`. По истечению срока действия политики ПО Яндекс Объектное хранилище перестает принимать файлы, загружаемые с помощью формы.

Поле `conditions` содержит набор правил для полей формы. Хотя бы одно правило должно быть указано для каждого поля формы.

Правила политики безопасности могут быть следующих типов:

Тип правила	Описание

Точное совпадение	Поле в форме должно иметь в точности то значение, которое указано в политике. Например, `{"acl": "public-read"}`. Также возможна альтернативная форма записи: `[ "eq", "$acl", "public-read" ]`.
Частичное совпадение	Значение поля в форме должно начинаться с указанной в политике строки. Например, `["starts-with", "$key", "key_prefix"]`. Если в качестве значения указана пустая строка, то соответствующее поле может принимать любое значение. Например, `["starts-with", "$Content-Type", ""]`.
`content-length-range`	Ограничение размера загружаемого объекта. Например, `["content-length-range", 0, 1048576]`.

Возможные ограничения:

Элемент	Тип ограничения	Область ограничения
`acl`	Полное и частичное совпадение.	Поле формы `acl`.
`bucket`	Полное и частичное совпадение.	Имя бакета.
`content-length-range`	`content-length-range`	`content-length-range`
`key`	Полное и частичное совпадение.	Поле формы `key`. Позволяет задать ключ объекта или префикс.
`success_action_redirect`	Полное и частичное совпадение.	Поле формы `success_action_redirect`.
`success_action_status`	Полное и частичное совпадение.	Поле формы `success_action_status`.
`X-Amz-*`	Полное совпадение.	Поля формы `X-Amz-*`, кроме `X-Amz-Meta-*`.
`X-Amz-Meta-*`	Полное и частичное совпадение.	Поля формы `X-Amz-Meta-*`.

`Cache-Control`
`Content-Disposition`
`Content-Encoding`
`Content-Type`
`Expires`
	Полное и частичное совпадение.	Одноименные поля формы.

Подпись политики безопасности

Общий алгоритм подписи политики:

1. Закодировать JSON-документ политики в base64.
2. Сгенерировать подписывающий ключ.
3. Сгенерировать подпись политики.

Пример генерирования формы с помощью boto3

Входные условия:

- Файлы должны сохраняться в бакет `user-data` с префиксом `/users/upload/`.
- Загруженные объекты открыты для публичного чтения.
- В случае успешной загрузки происходит перенаправление на страницу `https://example.com`.

Для генерирования полей формы воспользуемся boto3 Python SDK:

```
aws_access_key_id = 'JK38EXAMP********'
aws_secret_access_key = 'ExamP1eSecReTKeykdo********'
endpoint = 'https://<s3_api_хост>'

s3 = boto3.client('s3',
                  aws_access_key_id=aws_access_key_id,
                  aws_secret_access_key=aws_secret_access_key,
                  region_name='<region-id>',
                  endpoint_url=endpoint,
                  config=botocore.client.Config(signature_version='s3v4'),
                  )

key = 'users/uploads/${filename}'
bucket = 'user-data'
conditions = [{"acl":"public-read"}, ["starts-with", "$key", "users/uploads"],
{'success_action_redirect': 'https://example.com'}]
fields = {'success_action_redirect': 'https://example.com'}

prepared_form_fields = s3.generate_presigned_post(Bucket=bucket,
                                                  Key=key,
                                                  Conditions=conditions,
                                                  Fields=fields,
                                                  ExpiresIn=60 * 60)
```

```
print(prepared_form_fields)
```

Скрипт вернет документ JSON следующего вида:

```
{
    'url': u'https://<s3_api_хост>/user-data',
    'fields': {
        'X-Amz-Algorithm': 'AWS4-HMAC-SHA256',
        'X-Amz-Date': '20190722T153936Z',
        'success_action_redirect': 'https://example.com',
        'X-Amz-Signature':
'4bdfb2209fc30744458be10bc3b99361f2f50add20f2ca2425587a27********',
        'key': 'users/uploads/${filename}',
        'policy': u'eyJjb25kaXRpb25zIj...M5OjM2WiJ9',
        'X-Amz-Credential': u'JK38EXAMP********/20190722/<region-
id>/s3/aws4_request'}
}
```

Используя значения из выданного документа, можно построить HTML-страницу с формой для отправки файла:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    </head>
    <body>
        <form action="https://<s3_api_хост>/user-data" method="post"
enctype="multipart/form-data">
            Ключ в хранилище:
            <input type="input"    name="key" value="users/uploads/${filename}" /><br
/>
            <input type="hidden"   name="X-Amz-Credential"
value="JK38EXAMP********/20190722/<region-id>/s3/aws4_request" />
            <input type="hidden"   name="acl" value="public-read" />
            <input type="hidden"   name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
            <input type="hidden"   name="X-Amz-Date" value="20190722T153936Z" />
            <input type="hidden"   name="success_action_redirect"
value="https://example.com" />
            <input type="hidden"   name="policy"
value="eyJjb25kaXRpb25zIj...M5OjM2WiJ9" />
            <input type="hidden" name="X-Amz-Signature"
value="4bdfb2209fc30744458be10bc3b99361f2f50add20f2ca2425587a2722859f96" />
            Файл для загрузки:
            <input type="file"   name="file" /> <br />
            <input type="submit" name="submit" value="Загрузить" />
        </form>
    </body>
</html>
```

Класс хранилища

ПО Яндекс Объектное хранилище позволяет хранить объекты в Стандартном классе хранилища.

Класс хранилища опционально указывается при загрузке каждого отдельного объекта.

Идентификаторы классов хранилища

При работе с ПО Яндекс Объектное хранилище через API, совместимый с Amazon S3, используйте для классов хранилища следующие идентификаторы:

- Стандартное хранилище — `STANDARD`.

Метки

Метка — это пара ключ-значение для логической маркировки бакетов и объектов.

Метки бакетов

Ограничения при работе с метками:

- Максимальное количество меток на один ресурс: 64.
- Параметры ключа метки:
  - Длина — от 1 до 63 символов.
  - Может содержать строчные буквы латинского алфавита, цифры, дефисы и подчеркивания.
  - Первый символ — буква.
- Параметры значения метки:
  - Длина — до 63 символов.
  - Может содержать строчные буквы латинского алфавита, цифры, дефисы и подчеркивания.
- Метки с префиксом `aws:` в ключе зарезервированы для работы ПО Яндекс Объектное хранилище. Такие метки не учитываются в общем количестве меток бакета, их нельзя удалить или изменить.

Метки объектов

Ограничения при работе с метками объектов:

- Максимальное количество меток на один объект: 10.
- Параметры ключа метки:
  - Длина — до 128 символов.
  - Содержание — буквы латинского алфавита, цифры, пробел и символы `+` `-` `=` `.` `_` `:` `/` `@`.
- Параметры значения метки:
  - Длина — до 256 символов.
  - Содержание — буквы латинского алфавита, цифры, пробел и символы `+` `-` `=` `.` `_` `:` `/` `@`.
- Метки с префиксом `aws:` в ключе зарезервированы для работы ПО Яндекс Объектное хранилище. Такие метки не учитываются в общем количестве меток объекта, их нельзя удалить или изменить.

Квоты и лимиты

В ПО Яндекс Объектное хранилище действуют следующие ограничения:

- *Квоты* — организационные ограничения, которые можно изменить в консоли управления.
- *Лимиты* — технические ограничения, обусловленные особенностями архитектуры ПО Яндекс Объектное хранилище. Изменить лимиты невозможно.

Квоты

Вид ограничения	Значение
Объем хранилища в одном тенанте	Без ограничений

Лимиты

Вид ограничения	Значение
Максимальный размер объекта	5 ТБ
Общий размер заголовков для 1 запроса к HTTP API	8 КБ
Размер пользовательских метаданных объекта	2 КБ
Максимальный размер данных для загрузки за 1 запрос	5 ГБ
Минимальный размер части данных для составной загрузки, кроме последнего	5 МБ
Максимальное количество частей в составной загрузке	10000
Максимальный размер политики доступа к бакету	20 КБ

Обзор способов управления доступом в ПО Яндекс Объектное хранилище

В ПО Яндекс Объектное хранилище реализовано несколько взаимосвязанных механизмов для управления доступом:

- Список управления доступом (ACL).
- Политика доступа (bucket policy).
- Подписанные (pre-signed) URL.

Алгоритм проверок:

1. *ACL бакета*:
    - Если запрос прошел проверку *ACL бакета*, проверяется включена ли *политика доступа бакета*.
    - Если запрос не прошел проверку *ACL бакета*, то применяется проверка доступа через *ACL объекта*.
2. *Политика доступа бакета*:
    - Если политика доступа включена:
        1. Если запрос подошел хотя бы под одно из правил `Deny` политики бакета, то применяется проверка доступа через *ACL объекта*.
        2. Если запрос не подошел ни под одно из правил политики бакета, то применяется проверка доступа через *ACL объекта*.
3. *ACL объекта*:
    - Если запрос прошел проверку *ACL объекта*, доступ будет разрешен.
    - Если запрос не прошел проверку *ACL объекта*, доступ будет запрещен.

Список управления доступом (ACL)

Список управления доступом (ACL) — перечень разрешений на выполнение действий. Позволяет базово разграничить доступы.

Получатели доступа:

- учетная запись S3;
- публичная группа.

Доступ выдается на бакет или объект.

Политика доступа (bucket policy)

Политика доступа (bucket policy) — перечень правил, запрещающих или разрешающих действия при выполнении определенных условий. Позволяет гранулярно разграничить доступы к бакетам, объектам и группам объектов.

Получатели доступа:

- учетная запись S3;
- анонимный пользователь.

Доступ выдается на бакет, объект или группу объектов.

Подписанные (pre-signed) URL

Подписанные (pre-signed) URL — способ предоставления анонимным пользователям временного доступа к определенным действиям в ПО Яндекс Объектное хранилище с помощью URL, содержащих в своих параметрах данные для авторизации запроса.

Доступ выдается на бакет или объект.

Список управления доступом (ACL)

ACL в ПО Яндекс Объектное хранилище — список разрешений для каждого объекта и бакета.

Разрешения, выданные на бакет, распространяются на все находящиеся в нем объекты. Также ACL позволяет расширить доступы к отдельному объекту.

По умолчанию для каждого нового объекта или бакета создается пустой ACL. Пользователь, обладающий соответствующими правами, может отредактировать или загрузить ACL для бакетов и объектов.

С помощью ACL можно выдать разрешения учетной записи S3 или публичной группе (группе всех пользователей, группе всех аутентифицированных пользователей), при этом необходимо знать идентификатор получателя разрешений. При выдаче разрешений вы можете использовать предопределенные ACL, которые содержат популярные наборы доступов.

Описание структуры ACL смотрите в разделе XML-схема ACL. В одном ACL вы можете задать не более 100 правил.

Идентификатор получателя разрешений

- Учетная запись S3

  Для получения идентификатора в консоли управления на панели слева выберите **Тенанты**, перейдите в нужный тенант, выберите вкладку **Учетные данные S3** и выберите нужную учетную запись.
- Публичная группа

  Для выдачи разрешений используйте URI публичной группы.

Операции с ACL

- С помощью API, совместимого с Amazon S3, вы можете загрузить или скачать ACL для бакета или объекта.

  Удалить ACL невозможно. Чтобы удалить все разрешения, загрузите пустой ACL.

Виды разрешений

Разрешения соответствуют ролям учетных записей S3 в тенанте.

Название разрешения	Роль	Описание
`READ`	`VIEWER`	Для бакета: разрешение на получение списка объектов в бакете, чтение различных настроек бакета (жизненный цикл, CORS), чтение всех объектов в бакете. Для объекта: разрешение на чтение.

`WRITE`	`EDITOR`	Для бакета: запись, перезапись и удаление объектов в бакете. Используется обязательно вместе с `READ`, отдельно указать разрешение `WRITE` нельзя. Для объекта: разрешение не имеет смысла, при записи объекта проверяются разрешения для бакета.
`FULL_CONTROL`	`ADMIN`	Полный доступ к объектам и бакетам.
`READ_ACP`	`VIEWER`	Разрешение на чтение ACL. Только для объектов.
`WRITE_ACP`	`EDITOR`	Разрешение на запись ACL. Только для объектов.

Если при оформлении ACL указать доступ `WRITE`, но при этом не указать `READ`, то ПО Яндекс Объектное хранилище ответит с кодом `501 Not Implemented`.

Предопределенные ACL

ACL	Описание
`private` `bucket-owner-full-control`	Учетные записи S3 получают разрешения в соответствии со своими ролями в тенанте.
`public-read`	Публичная группа `AllUsers` получает разрешение `READ`.
`public-read-write`	Публичная группа `AllUsers` получает разрешения `READ` и `WRITE`.
`authenticated-read`	Публичная группа `AuthenticatedUsers` получает разрешение `READ`.

Предопределенные ACL могут применяться как к объектам, так и к бакетам. ACL `public-read-write`, примененный к объекту, эквивалентен `public-read`.

Загрузить предопределенный ACL можно только с помощью HTTP API, совместимого с Amazon S3. При загрузке ACL используйте HTTP-заголовок `X-Amz-Acl`.

Публичные группы

AllUsers

Включает в себя всех пользователей.

Разрешение для `AllUsers` выглядит следующим образом:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
</Grantee>
```

AuthenticatedUsers

Включает в себя всех аутентифицированных пользователей: как из вашего тенанта, так и из других тенантов.

Разрешение для `AuthenticatedUsers` выглядит следующим образом:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers</URI>
</Grantee>
```

Политика доступа (bucket policy)

Политики доступа устанавливают права на действия с бакетами, объектами и группами объектов.

Политика срабатывает, когда пользователь делает запрос к какому-либо ресурсу. В результате срабатывания политики запрос либо выполняется, либо отклоняется.

> **Примечание**
>
> Если к бакету применена политика доступа без правил, то доступ будет запрещен всем пользователям. Чтобы отключить проверки запросов по политике доступа, удалите ее.

Политику доступа можно описать в формате JSON по специальной схеме, чтобы передать через один из программных инструментов — AWS CLI или API.

Из чего состоит политика

Политика доступа состоит из правил, а правило — из следующих базовых элементов:

Ресурс

Бакет, объект в бакете (`<имя_бакета>/some/key`) или префикс (`<имя_бакета>/some/path/*`), в том числе пустой префикс для обозначения всех объектов в бакете (`<имя_бакета>/*`). В правиле можно указать несколько ресурсов.

> **Примечание**
>
> Ресурс бакета не включает в себя ресурсы всех его объектов. Чтобы правило в политике доступа относилось к бакету и всем объектам, их нужно указать как отдельные ресурсы: например, `samplebucket` и `samplebucket/*`.

Если вы описываете политику в формате JSON, у ресурса должен быть префикс `arn:aws:s3:::`, например `arn:aws:s3:::<имя_бакета>`.

Если имя ресурса содержит символ `?`, `*` или `$`, поставьте перед таким символом знак `$`, а сам символ заключите в фигурные скобки `{}`. Например, имени бакета `my?bucket` будет соответствовать запись `my${?}bucket`.

Действие

Набор операций над ресурсом, который будет запрещен или разрешен правилом.

Результат

Запрет или разрешение запрошенного действия. Сначала проверяется попадание запроса в фильтр с действием `Deny`, при совпадении запрос отклоняется и дальнейшие проверки не проводятся. При попадании в фильтр с действием `Allow` запрос разрешается. Если запрос не попал ни в один из фильтров, то запрос отклоняется.

Принципал

Получатель запрошенного разрешения. Это может быть учетная запись S3 или анонимный пользователь.

Условие

Определение случаев, когда действует правило.

Если для правила задано одновременно несколько условий или внутри одного условия задано одновременно несколько ключей, то такие условия и ключи применяются с логикой `и`.

Если для одного ключа условия задано одновременно несколько значений, то такие значения применяются с логикой `или`.

Примеры конфигурации

- Правило, которое разрешает скачивать объекты только из указанного диапазона IP-адресов:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<имя_бакета>/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "100.101.102.128/30"
        }
      }
    }
  ]
}
```

- Правило, которое запрещает скачивать объекты с указанного IP-адреса:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "*",
      "Resource": "arn:aws:s3:::<имя_бакета>/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<имя_бакета>/*",
```

```
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "100.101.102.103"
        }
      }
    }
  ]
}
```

- Правило, которое дает разным пользователям полный доступ только к определенным папкам, каждому пользователю — к своей:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"User1PermissionsResource",
      "Effect":"Allow",
      "Principal": {
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "*",
      "Resource":["arn:aws:s3:::<имя_бакета>/user1path/*"]
    },
    {
      "Sid":"User1PermissionsPrefix",
      "Effect":"Allow",
      "Principal": {
          "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "s3:ListBucket",
      "Resource":["arn:aws:s3:::<имя_бакета>"],
      "Condition": {
        "StringLike": {
          "s3:prefix": "user1path/*"
        }
      }
    },
    {
      "Sid":"User2PermissionsResource",
      "Effect":"Allow",
      "Principal": {
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "*",
      "Resource":["arn:aws:s3:::<имя_бакета>/user2path/*"]
    },
    {
      "Sid":"User2PermissionsPrefix",
      "Effect":"Allow",
      "Principal": {
```

```
        "CanonicalUser": "<идентификатор_пользователя>"
      },
      "Action": "s3:ListBucket",
      "Resource":["arn:aws:s3:::<имя_бакета>"],
      "Condition": {
        "StringLike": {
          "s3:prefix": "user2path/*"
        }
      }
    }
  ]
}
```

Подписанные (pre-signed) URL

С помощью подписанных URL произвольный пользователь может выполнять в ПО Яндекс Объектное хранилище различные операции, например:

- Скачать объект.
- Загрузить объект.
- Создать бакет.

Подписанный URL — это URL, содержащий в своих параметрах данные для авторизации запроса. Составить подписанный URL может пользователь, обладающий статическими ключами доступа.

Раздел содержит общие принципы построения подписанного URL с использованием AWS Signature V4.

> **Примечание**
>
> SDK для различных языков программирования и другие инструменты для работы с AWS S3 содержат готовые методы генерирования подписанного URL, которые можно использовать и для ПО Яндекс Объектное хранилище.

Общий вид подписанного URL

```
https://<имя_бакета>.<s3_api_хост>/<ключ_объекта>?
     X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=<access_key-id>%2F<YYYYMMDD>%2F<region-id>%2Fs3%2Faws4_request
    &X-Amz-Date=<время_в_формате_ISO_8601>
    &X-Amz-Expires=<время_жизни_ссылки>
    &X-Amz-SignedHeaders=<список_подписанных_заголовков>
    &X-Amz-Signature=<подпись>
```

Параметры подписанного URL:

Параметр	Описание
`X-Amz-Algorithm`	Идентифицирует версию подписи и алгоритм ее вычисления. Значение — `AWS4-HMAC-SHA256`.
`X-Amz-Credential`	Идентификатор для подписи. Строка формата `<access-key-id>/<YYYYMMDD>/<region-id>/s3/aws4_request`, где `<YYYYMMDD>` должна совпадать с датой, установленной в заголовке `X-Amz-Date`.

`X-Amz-Date`	Время в формате ISO8601, например, `20180719T000000Z`. Указанная дата должна по значению (не по формату) совпадать с датой в параметре `X-Amz-Credential`.
`X-Amz-Expires`	Время в секундах, в течение которого ссылка действительна. Начало отсчета — момент, указанный в `X-Amz-Date`. Максимальное значение — 2592000 секунд (30 дней).
`X-Amz-SignedHeaders`	Заголовки запроса, которые вы хотите подписать, разделенные точкой с запятой (`;`). Обязательно подписывайте заголовок `Host` и все заголовки `X-Amz-*`, которые используются в запросе. Другие заголовки подписывать не обязательно, однако чем больше вы подпишете заголовков, тем безопаснее будет запрос.
`X-Amz-Signature`	Подпись запроса.

Составление подписанного URL

Чтобы получить подписанный URL:

1. Составьте канонический запрос.
2. Составьте строку для подписи.
3. Сформируйте подписывающий ключ.
4. Вычислите подпись с помощью ключа.
5. Сформируйте подписанный URL.

Для составления подписанного URL необходимо обладать статическими ключами доступа.

Канонический запрос

Общий вид канонического запроса:

```
<HTTPVerb>\n
<CanonicalURL>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
UNSIGNED-PAYLOAD
```

HTTPVerb

HTTP метод, которым будет отправлен запрос: `GET`, `PUT`, `HEAD` или `DELETE`.

CanonicalURL

URL-кодированный ключ объекта. Например, `/folder/object.ext`.

Примечание

Не нормализуйте путь. Например, объект может иметь ключ `some//strange//key//example` и нормализация пути `/<bucket-name>/some/strange/key/example` сделает ключ некорректным.

CanonicalQueryString

Каноническая строка запроса должна включать все query параметры конечного URL, кроме `X-Amz-Signature`. Параметры в строке должны быть URL-кодированы и отсортированы по алфавиту.

Пример:

```
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=JK38EXAMPLEAKDID8%2F20190801%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=20190801T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
```

CanonicalHeaders

Список заголовков запроса со значениями.

Требования:

- Каждый заголовок отделяется символом новой строки `\n`.
- Имена заголовков должны быть в нижнем регистре.
- Заголовки должны быть отсортированы по алфавиту.
- Не должно быть лишних пробелов.
- Список должен содержать заголовок `host` и все заголовки `x-amz-*`, которые используются в запросе.

Дополнительно в список можно добавить любой из заголовков запроса. Чем больше вы подпишете заголовков, тем безопаснее будет запрос.

Пример:

```
host:sample-bucket.<s3_api_хост>
x-amz-date:20190801T000000Z
```

SignedHeaders

Список имен заголовков запроса в нижнем регистре, отсортированный по алфавиту и разделенный точками с запятыми.

Пример:

```
host;x-amz-date
```

Завершение канонического запроса

Завершать канонический запрос всегда должна строка `UNSIGNED-PAYLOAD`.

Строка для подписи

Общий вид строки для подписи:

```
"AWS4-HMAC-SHA256" + "\n" +
<timestamp> + "\n" +
<scope> + "\n" +
Hex(Hash-SHA256(<CanonicalRequest>))
```

Где:

- `AWS4-HMAC-SHA256` — алгоритм хэширования.
- `timestamp` — текущее время в формате ISO 8601, например, `20190801T000000Z`. Указанная дата должна по значению (не по формату) совпадать с датой в `scope`.
- `scope` — `<YYYYMMDD>/<region-id>/s3/aws4_request`.
- `CanonicalRequest` — сформированный ранее канонический запрос. В строку для подписи помещается SHA256-хэш канонического запроса в шестнадцатеричном представлении.

Подписывающий ключ

Чтобы сгенерировать подписывающий ключ:

1. Закодируйте дату с помощью секретного ключа:

   ```
   DateKey = sign("AWS4" + "SecretKey", "yyyymmdd")
   ```

2. Закодируйте регион с помощью полученного на предыдущем шаге ключа `DateKey`:

   ```
   RegionKey = sign(DateKey, "<region-id>")
   ```

3. Закодируйте сервис с помощью полученного на предыдущем шаге ключа `RegionKey`:

   ```
   ServiceKey = sign(RegionKey, "s3")
   ```

4. Получите подписывающий ключ:

   ```
   SigningKey = sign(ServiceKey, "aws4_request")
   ```

Подпись строки с помощью ключа

Чтобы получить подпись строки, необходимо использовать механизм `HMAC` с хэширующей функцией `SHA256`, а полученный результат преобразовать в шестнадцатеричное представление.

```
signature = Hex(sign(SigningKey, StringToSign))
```

Подписанный URL

Чтобы составить подписанный URL, к URL ресурса добавьте параметры, необходимые для авторизации запроса, в том числе параметр `X-Amz-Signature` с вычисленной подписью в значении.

Значения остальных параметров должны совпадать с аналогичными значениями, указанными ранее в каноническом запросе и в строке для подписи.

Пример составления подписанного URL для скачивания объекта

Составим подписанный URL для скачивания объекта `object-for-share.txt` из бакета в течение часа:

- Статический ключ:

  ```
  access_key_id = 'JK38EXAMP********'
  secret_access_key = 'ExamP1eSecReTKeykdo********'
  ```

- Канонический запрос:

  ```
  GET
  /object-for-share.txt
  X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
  Credential=YCAJEK0Iv6x********eLTAdg%2F20231208%2F<region-
  id>%2Fs3%2Faws4_request&X-Amz-Date=20231208T184504Z&X-Amz-Expires=3600&X-Amz-
  SignedHeaders=host
  host:<имя_бакета>.<s3_api_хост>

  host
  UNSIGNED-PAYLOAD
  ```

- Строка для подписи:

  ```
  AWS4-HMAC-SHA256
  20231208T184504Z
  20231208/<region-id>/s3/aws4_request
  e823d75aad02c1317589bd5373fe9e20d5ef44499237703ff23e5600********
  ```

- Подписывающий ключ:

```
sign(sign(sign(sign("AWS4" + "ExamP1eSecReTKeykdokKK38800","20190801"),"<region-id>"),"s3"),"aws4_request")
```

Функция `sign` введена для обозначения способа вычисления ключа с помощью механизма HMAC с хэширующей функцией SHA256.

- Подпись:

```
b10c16a1997bb524bf59974512f1a6561cf2953c29dc3efbdb920790********
```

- Подписанный URL:

```
https://<имя_бакета>.<s3_api_хост>/object-for-share.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=YCAJEK0Iv6xqy-pEQcueLTAdg%2F20231208%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=20231208T195434Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=b10c16a1997bb524bf59974512f1a6561cf2953c29dc3efbdb920790********
```

Примеры кода для генерации подписанных URL

В подразделе приведены примеры кода для генерации подписанных URL.

Python

```
import datetime
import hashlib
import hmac

access_key = '<идентификатор_статического_ключа>'
secret_key = '<содержимое_статического_ключа>'
object_key = '<ключ_объекта>'
bucket = '<имя_бакета>'
host = '<s3_api_хост>'
now = datetime.datetime.now(datetime.UTC)
datestamp = now.strftime('%Y%m%d')
timestamp = now.strftime('%Y%m%dT%H%M%SZ')

def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

canonical_request = """GET
/{bucket}/{object_key}
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=
{access_key}%2F{datestamp}%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=
{timestamp}&X-Amz-Expires=3600&X-Amz-SignedHeaders=host
host:{host}

host
```

```
UNSIGNED-PAYLOAD""".format(
    bucket=bucket,
    object_key=object_key,
    access_key=access_key,
    datestamp=datestamp,
    timestamp=timestamp,
    host=host)

print()
print("Canonical request:\n" + canonical_request)
print()

string_to_sign = """AWS4-HMAC-SHA256
{timestamp}
{datestamp}/<region-id>/s3/aws4_request
{request_hash}""".format(
    timestamp=timestamp,
    datestamp=datestamp,
    request_hash=hashlib.sha256(canonical_request.encode('utf-8')).hexdigest())

print()
print("String to be signed:\n" + string_to_sign)
print()

signing_key = sign(sign(sign(sign(('AWS4' + secret_key).encode('utf-8'), datestamp),
'<region-id>'), 's3'), 'aws4_request')
signature = hmac.new(signing_key, string_to_sign.encode('utf-8'),
hashlib.sha256).hexdigest()

print()
print("Signature: " + signature)
print()

signed_link = "https://" + host + '/' + bucket + '/' + object_key + "?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=" + access_key + "%2F" + datestamp +
"%2F<region-id>%2Fs3%2Faws4_request&X-Amz-Date=" + timestamp + "&X-Amz-
Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=" + signature + "\n"

print()
print("Signed Link:\n" + signed_link)
```

PHP

```
<?php

  date_default_timezone_set('UTC');
  $keyid = "<идентификатор_статического_ключа>";
  $secretkey = "<содержимое_статического_ключа>";
  $path = "<ключ_объекта>";
  $objectname = "/".implode("/", array_map("rawurlencode", explode("/", $path)));
  $host = "<имя_бакета>.<s3_api_хост>";
```

```
  $region = "<region-id>";
  $timestamp = time();
  $dater = strval(date('Ymd', $timestamp));
  $dateValue = strval(date('Ymd', $timestamp))."T".strval(date('His',
$timestamp))."Z";

  // Generate the canonical request
  $canonical_request = "GET\n".$objectname."\nX-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=".$keyid."%2F".$dater."%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=".$dateValue."&X-Amz-Expires=3600&X-Amz-
SignedHeaders=host\nhost:".$host."\n\nhost\nUNSIGNED-PAYLOAD";

  echo "<b>Canonical request: </b><br>".$canonical_request."<br><br>";

  // Generate the string to be signed
  $string_to_sign = "AWS4-HMAC-
SHA256\n".$dateValue."\n".$dater."/".$region."/s3/aws4_request\n".openssl_digest($canon
"sha256", $binary = false);

  echo "<b>String to be signed: </b><br>".$string_to_sign."<br><br>";

  // Generate the signing key
  $signing_key = hash_hmac('sha256', 'aws4_request', hash_hmac('sha256', 's3',
hash_hmac('sha256', '<region-id>', hash_hmac('sha256', $dater, 'AWS4'.$secretkey,
true), true), true), true);

  echo "<b>Signing key: </b><br>".$signing_key."<br><br>";

  // Generate the signature
  $signature = hash_hmac('sha256', $string_to_sign, $signing_key);

  echo "<b>Signature: </b><br>".$signature."<br><br>";

  // Generate the pre-signed link
  $signed_link = "https://".$host.$objectname."?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-
Amz-Credential=".$keyid."%2F".$dater."%2F<region-id>%2Fs3%2Faws4_request&X-Amz-
Date=".$dateValue."&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-
Signature=".$signature."\n";

  echo '<b>Signed link: </b><br>'.'<a href = "'.$signed_link.'" target =
"_blank">'.$signed_link.'</a>';

?>
```

Примеры получения подписанной ссылки в инструментах ПО Яндекс Объектное хранилище

В подразделе приведены примеры генерации подписанных URL с помощь различных инструментов ПО Яндекс Объектное хранилище.

AWS CLI

Ссылку на скачивание объекта можно сгенерировать с помощью AWS CLI. Для этого выполните команду:

```
aws s3 presign s3://<имя_бакета>/<ключ_объекта> \
  --expires-in <время_жизни_ссылки> \
  --endpoint-url "https://<s3_api_хост>/"
```

Чтобы ссылка сформировалась корректно, обязательно укажите параметр `--endpoint-url` с указанием на доменное имя S3 API ПО Яндекс Объектное хранилище.

Python (boto3)

Пример генерирует подписанный URL для скачивания объекта `object-for-share` из бакета `bucket-with-objects`. URL действителен в течение 100 секунд.

```
# coding=utf-8

import boto3
from botocore.client import Config


ENDPOINT = "https://<s3_api_хост>"

ACCESS_KEY = "JK38EXAMP********"
SECRET_KEY = "ExamP1eSecReTKeykdo********"

session = boto3.Session(
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,
    region_name="<region-id>",
)
s3 = session.client(
    "s3", endpoint_url=ENDPOINT, config=Config(signature_version="s3v4")
)

presigned_url = s3.generate_presigned_url(
    "get_object",
    Params={"Bucket": "bucket-with-objects", "Key": "object-for-share"},
    ExpiresIn=100,
)

print(presigned_url)
```

Аутентификация в API ПО Яндекс Объектное хранилище

Вы можете работать с ПО Яндекс Объектное хранилище с помощью AWS S3 API.

Для аутентификации в AWS S3 API используйте статический ключ доступа. Статический ключ доступа выпускается на учетную запись S3, и все действия с использованием этого ключа выполняются от имени этой учетной записи.

> **Примечание**
>
> Учетная запись S3 может просматривать список бакетов только в том тенанте, в котором она была создана.
>
> Учетная запись S3 может выполнять действия с объектами в бакетах, которые созданы в тенантах, отличных от тенанта учетной записи. Для этого назначьте учетной записи права на нужный бакет с помощью ACL или политики доступа.

Если вы хотите использовать AWS S3 API напрямую, без SDK и приложений, вам придется самостоятельно подписывать запросы.

Пример использования AWS S3 API

Начиная с версии 8.3.0 утилита `curl` поддерживает автоматическое формирование строки для подписи, подпись запроса и подстановку необходимых заголовков при работе с AWS S3 API.

Также вы можете вручную сформировать указанные заголовки и подписать запрос.

> **Примечание**
>
> Убедитесь, что учетная запись S3, от имени которой вы выполняете запрос, имеет необходимые права для выполнения запрашиваемого действия. Например, для загрузки объекта в бакет назначьте учетной записи роль `EDITOR` или `ADMIN` в тенанте.

Ниже приведены примеры запросов для загрузки объекта в бакет.

Версия curl 8.3.0 и выше

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
LOCAL_FILE="<путь_к_локальному_файлу>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"

curl \
  --request PUT \
  --user "${AWS_KEY_ID}:${AWS_SECRET_KEY}" \
```

```
  --aws-sigv4 "aws:amz:<region-id>:s3" \
  --upload-file "${LOCAL_FILE}" \
  --verbose \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где:

- `AWS_KEY_ID` — идентификатор статического ключа доступа.
- `AWS_SECRET_KEY` — секретный ключ.
- `LOCAL_FILE` — путь к локальному файлу, который вы хотите загрузить, например `./sample.txt`.
- `BUCKET_NAME` — имя бакета, в который загружается файл.
- `OBJECT_PATH` — ключ, который будет присвоен объекту в бакете, например `new-prefix/sample-object.txt`.

Аналогично вы можете загрузить файл в бакет, не сохраняя его локально. Например, заархивируйте директорию и отправьте архив в бакет:

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"
DIRECTORY_PATH="<путь_к_директории>"

tar -cvzf - "${DIRECTORY_PATH}" | curl \
  --request PUT \
  --user "${AWS_KEY_ID}:${AWS_SECRET_KEY}" \
  --aws-sigv4 "aws:amz:<region-id>:s3" \
  --upload-file - \
  --verbose \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где `DIRECTORY_PATH` — путь к директории, которую вы хотите заархивировать.

Версия curl 8.2.1 и ниже

```
AWS_KEY_ID="<идентификатор_статического_ключа>"
AWS_SECRET_KEY="<секретный_ключ>"
LOCAL_FILE="<путь_к_локальному_файлу>"
BUCKET_NAME="<имя_бакета>"
OBJECT_PATH="<ключ_объекта>"
CONTENT_TYPE="<MIME-тип_объекта>"
DATE_VALUE=`date -R`
STRING_TO_SIGN="PUT\n\n${CONTENT_TYPE}\n${DATE_VALUE}\n/${BUCKET_NAME}/${OBJECT_PATH}"
SIGNATURE=`echo -en ${STRING_TO_SIGN} | openssl sha1 -hmac ${AWS_SECRET_KEY} -binary
| base64`

curl \
  --request PUT \
  --upload-file "${LOCAL_FILE}" \
```

```
  --verbose \
  --header "Host: <s3_api_хост>" \
  --header "Date: ${DATE_VALUE}" \
  --header "Content-Type: ${CONTENT_TYPE}" \
  --header "Authorization: AWS ${AWS_KEY_ID}:${SIGNATURE}" \
  "https://<s3_api_хост>/${BUCKET_NAME}/${OBJECT_PATH}"
```

Где:

- `AWS_KEY_ID` — идентификатор статического ключа доступа.
- `AWS_SECRET_KEY` — секретный ключ.
- `LOCAL_FILE` — путь к локальному файлу, который вы хотите загрузить, например `./sample.txt`.
- `BUCKET_NAME` — имя бакета, в который загружается файл.
- `OBJECT_PATH` — ключ, который будет присвоен объекту в бакете, например `new-prefix/sample-object.txt`.
- `CONTENT_TYPE` — MIME-тип загружаемого объекта, например `text/plain`.

Как пользоваться S3 API

Подготовка к работе

Чтобы воспользоваться API:

1. Создайте учетную запись S3.
2. Назначьте учетной записи роль, которая нужна для вашего проекта.
3. Создайте статический ключ доступа.

Авторизация статическими ключами необходима для обращения напрямую к HTTP API.

Общий вид запроса к API

> **Примечание**
>
> Для работы с S3 API в ПО Яндекс Объектное хранилище лучше использовать AWS CLI или AWS SDK, подходящий для вашей среды разработки.

Общий вид запроса к API:

```
{GET|HEAD|PUT|DELETE} /<имя_бакета>/<ключ_объекта> HTTP/2
Host: <s3_api_хост>
Content-Length: length
Date: date
Authorization: authorization string (AWS Signature Version 4)

Request_body
```

Запрос содержит HTTP-метод, имя бакета и ключ объекта.

Имя бакета можно указать как часть имени хоста. В этом случае запрос примет вид:

```
{GET|HEAD|PUT|DELETE} /<ключ_объекта> HTTP/2
Host: <имя_бакета>.<s3_api_хост>
...
```

Набор заголовков зависит от конкретного запроса и описан в документации на соответствующий запрос.

Если вы используете API напрямую (без SDK и приложений), то для подписи запросов вам придется самостоятельно генерировать заголовок `Authorization`. О том, как это сделать, читайте в разделе Authenticating Requests (AWS Signature Version 4) документации Amazon S3.

URL для запроса

URL может иметь одну из следующих форм:

- `http(s)://<s3_api_хост>/<имя_бакета>/<ключ_объекта>?<query-параметры>`
- `http(s)://<имя_бакета>.<s3_api_хост>/<ключ_объекта>?<query-параметры>`

> **Примечание**
>
> Для бакетов с точками в имени, например `example.ru`, протокол HTTPS доступен только с URL в формате `https://<s3_api_хост>/<имя_бакета>/<ключ_объекта>?<query-параметры>`.

URL содержит имя бакета, ключ объекта и query-параметры. Пример возможных query-параметров см. в описании метода `get` для получения объекта.

CORS запросы

Кросс-доменные запросы доступны для всех методов API, которые управляют объектами.

Для проверки разрешений CORS подразумевает предварительный запрос `Options` к ресурсу. ПО Яндекс Объектное хранилище позволяет отправлять кросс-доменные запросы к ресурсам без предварительного запроса, при этом в запросе должны быть те же заголовки, что и у предварительного запроса.

Подписывание запросов

Многие запросы к ПО Яндекс Объектное хранилище аутентифицируются на стороне сервиса и пользователь, отправляющий запрос, должен его подписать.

ПО Яндекс Объектное хранилище поддерживает подпись AWS Signature V4.

Процесс подписывания состоит из этапов:

1. Генерирование строки для подписи.
2. Генерирование подписывающего ключа.
3. Подпись строки с помощью ключа.

Для подписи необходимо использовать механизм HMAC с хэширующей функцией SHA256. Поддержка соответствующих методов есть во многих языках программирования. В примерах предполагается, что существует функция `sign(KEY, STRING)`, которая выполняет кодирование входной строки по заданному ключу.

Генерирование строки для подписи

Строка для подписи (`StringToSign`) зависит от сценария использования ПО Яндекс Объектное хранилище:

- Обращение к API, совместимому с Amazon S3, без помощи SDK или специализированных утилит.
- Загрузка объектов с помощью HTML-формы.
- Подписывание URL с помощью query-параметров.

Генерирование подписывающего ключа

Чтобы сгенерировать подписывающий ключ, вам необходимо иметь статические ключи доступа к ПО Яндекс Объектное хранилище.

Чтобы сгенерировать подписывающий ключ:

1. Закодируйте дату с помощью секретного ключа:

   ```
   DateKey = sign("AWS4" + "SecretKey", "yyyymmdd")
   ```

2. Закодируйте регион с помощью полученного на предыдущем шаге ключа `DateKey`:

   ```
   RegionKey = sign(DateKey, "<region-id>")
   ```

3. Закодируйте сервис с помощью полученного на предыдущем шаге ключа `RegionKey`:

   ```
   ServiceKey = sign(RegionKey, "s3")
   ```

4. Получите подписывающий ключ:

```
SigningKey = sign(ServiceKey, "aws4_request")
```

Подпись строки с помощью ключа

Чтобы получить подпись строки, необходимо использовать механизм `HMAC` с хэширующей функцией `SHA256`, а полученный результат преобразовать в шестнадцатеричное представление.

```
signature = Hex(sign(SigningKey, StringToSign))
```

Отладка с помощью AWS CLI

Чтобы отладить процесс формирования канонического запроса, строки для подписи и подписывающего ключа, используйте утилиту AWS CLI с параметром `--debug`.

AWS CLI

В терминале выполните команду для создания бакета и посмотрите, как генерируются параметры для запроса:

```
aws s3api create-bucket \
  --endpoint-url=https://<s3_api_хост> \
  --bucket <имя_бакета> \
  --debug
```

Результат:

```
2024-06-03 13:02:36,238 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
PUT
/<имя_бакета>

host:<s3_api_хост>
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b********
x-amz-date:20240603T100236Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b********

2024-06-03 13:02:36,238 - MainThread - botocore.auth - DEBUG - StringToSign:
AWS4-HMAC-SHA256
20240603T100236Z
20240603/<region-id>/s3/aws4_request
7877a13bafaa45f9751e7f345b64a63acc6de279ff927736e906d7c5********

2024-06-03 13:02:36,238 - MainThread - botocore.auth - DEBUG - Signature:
90545034742d1e057c8eeb2cca3c23a38a3ced5ef847f61ac80cb8e1********
```

Общие заголовки запросов

Заголовок	Описание
`Authorization`	Любой запрос к ПО Яндекс Объектное хранилище должен быть авторизован. Вместе с этим заголовком обязательно надо использовать либо заголовок `Date`, либо заголовок `X-Amz-Date`. О способах авторизации читайте в соответствующих разделах руководства.
`Cache-Control`	Набор директив для кэширования данных согласно RFC 2616.
`Content-Disposition`	Имя, под которым ПО Яндекс Объектное хранилище предложит сохранить объект как файл при выгрузке. Соответствует RFC 2616.
`Content-Encoding`	Определяет кодировку содержимого согласно RFC 2616.
`Content-Length`	Длина тела запроса (без заголовков) в соответствии с RFC 2616. Заголовок обязателен для всех запросов, которые передают в ПО Яндекс Объектное хранилище какие-либо данные, например, при загрузке объекта.
`Content-Type`	Тип данных в запросе. Например, `text/html`. Подробнее про типы данных читайте в статье Википедии Список MIME-типов. По умолчанию выставляется `binary/octet-stream`.
`Content-MD5`	128-битный MD5-хэш тела запроса, закодированный `base64`. Соответствует спецификации RFC 1864. ПО Яндекс Объектное хранилище использует заголовок, чтобы убедиться, что отправленные данные соответствуют полученным. Если в бакете настроены блокировки версий объектов по умолчанию, использовать заголовок `Content-MD5` при загрузке или копировании версии объекта обязательно.

`Date`	Дата и время отправки запроса. Формат: `Thu, 18 Jan 2018 09:57:35 GMT`. Когда установлен `X-Amz-Date`, ПО Яндекс Объектное хранилище игнорирует заголовок `Date`.
`Expect`	Ожидаемый код `100-continue`. Приложение при загрузке данных в ПО Яндекс Объектное хранилище может использовать следующую логику: - Отправлять запрос без тела, но с выставленным заголовком Expect: 100-continue. - Отправлять запрос с телом после получения ответа `100-continue`. В этом запросе заголовка `Expect` быть не должно.
`Expires`	Срок устаревания ответа. Соответствует RFC 2616.
`Host`	Хост-получатель запроса. Заголовок необходим для HTTP/1.1, но необязателен для HTTP/1.0 запросов.
`X-Amz-Date`	Дата и время на источнике запроса. Формат: `20211102T145822Z`. Когда установлен `X-Amz-Date`, ПО Яндекс Объектное хранилище игнорирует заголовок `Date`.

Общие заголовки ответов

Помимо стандартных HTTP-заголовков ПО Яндекс Объектное хранилище использует дополнительные заголовки, которые могут присутствовать в ответах.

В таблице ниже описаны дополнительные заголовки, а также стандартные заголовки, к которым необходимы дополнительные пояснения.

Заголовок	Описание
`ETag`	MD5-хэш объекта, если объект загружен единым файлом. Если объект загружен с помощью составной загрузки, то для вычисления `ETag` необходимо получить MD5-хэш от суммы MD5-хэшей загруженных частей (назовем ее `MD5_sum`) и присоединить к полученной строке количество частей (`N`): `"MD5_sum-N"`. Не включает метаданные объекта.
`X-Amz-Request-Id`	Уникальный идентификатор запроса.

Если запрос был отправлен с CORS-заголовками, то ответ может содержать заголовки ответа на предварительный запрос `Options`.

Ответы

Успешный ответ

При отсутствии ошибок ПО Яндекс Объектное хранилище отвечает HTTP-кодами 2xx. Код и тело ответа зависят от запроса и рассматриваются в описаниях запросов.

Ответ с ошибкой

При возникновении ошибки ПО Яндекс Объектное хранилище отвечает сообщением с соответствующим HTTP-кодом и дополнительным описанием в формате XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

Тег	Описание
`Code`	Код ошибки. Перечень кодов смотрите ниже по тексту.
`Message`	Описание ошибки на английском языке.
`RequestId`	Идентификатор запроса вызвавшего ошибку. Равен значению заголовка `X-Amz-Request-Id`.
`Resource`	Бакет или объект, при работе с которым возникла ошибка.

Коды ошибок

HTTP	Код ошибки	Описание

301	`PermanentRedirect`	К указанному бакету всегда следует обращаться по указанному в ответе адресу.
307	`Redirect`	К указанному бакету временно следует обращаться по указанному в ответе адресу.
400	`BadDigest`	Хэш переданный в заголовке `Content-MD5` не совпадает с вычисленным на стороне сервиса.
400	`CloudTotalAliveSizeQuotaExceed`	После загрузки объекта будет превышена соответствующая квота. Увеличьте квоту или удалите ненужные объекты, а после этого загрузите объект заново.
400	`CredentialsNotSupported`	Учетные данные не поддерживаются.
400	`EntityTooSmall`	Загружаемый объект меньше минимально допустимого размера.

400	`EntityTooLarge`	Загружаемый объект больше максимально допустимого.
400	`IncompleteBody`	Размер отправленных данных меньше, чем указано в заголовке `Content-Length`.
400	`IncorrectNumberOfFilesInPostRequest`	Метод POST требует передачи строго одного файла.
400	`InlineDataTooLarge`	Данные запроса превышают максимально допустимый размер.
400	`InvalidDigest`	Хэш, переданный в заголовке Content-MD5, не верен.
400	`InvalidArgument`	Недопустимый аргумент.
400	`InvalidBucketName`	Недопустимое имя бакета.

400	`InvalidPart`	Одна и более частей составной загрузки не найдены. Проверьте корректность перечня. Возможно отсутствующие части не были загружены.
400	`InvalidPartOrder`	Перечень частей составной загрузки указан некорректно. Части должны быть отсортированы по возрастанию номера.
400	`InvalidRequest`	Используйте AWS4-HMAC-SHA256.
400	`BucketMaxSizeExceeded`	Попытка превысить максимальный размер бакета. Описание ошибки в ответе: "You have attempted to exceed the max size configured for the bucket."
400	`InvalidStorageClass`	Некорректный класс хранилища.
`InvalidURI`		
`KeyTooLongError`		

`MalformedACLError`		
`MalformedPOSTRequest`		
`MalformedXML`		
`MaxMessageLengthExceeded`		
`MaxPostPreDataLengthExceededError`		
`MetadataTooLarge`		
`MissingRequestBodyError`		
`MissingSecurityHeader`		
400	`RequestIsNotMultiPartContent`	Запрос должен содержать данные типа `multipart/form-data`.
400	`RequestTimeout`	Таймаут на чтение/ запись.

400	`TooManyBuckets`	После создания бакета будет превышена соответствующая квота. Удалите ненужные бакеты, а после этого создайте бакет заново.
400	`UnexpectedContent`	В запросе не должно быть контента.
400	`UnresolvableGrantByEmailAddress`	Не зарегистрированный e-mail.
400	`UserKeyMustBeSpecified`	Запрос должен содержать указанный в описании ошибки заголовок.
403	`AccessDenied`	Доступ запрещен.
403	`InvalidAccessKeyId`	Неизвестный ключ.
403	`InvalidObjectState`	Запрос не может быть выполнен для текущего состояния объекта.
403	`InvalidSecurity`	Предоставленные секретные ключи не валидны.

403	`NotSignedUp`	Для аккаунта не разрешено использование ПО Яндекс Объектное хранилище.
403	`RequestTimeTooSkewed`	Слишком большая разница между временем отправки запроса и временем на сервере.
403	`SignatureDoesNotMatch`	Предоставленная подпись запроса не соответствует вычисленной ПО Яндекс Объектное хранилище.
404	`NoSuchBucket`	Указанного бакета не существует.
404	`NoSuchKey`	Указанный ключ не существует.
404	`NoSuchUpload`	Указанной составной загрузки не существует. Ошибка возникает в следующих случаях: - Указан неправильный идентификатор загрузки. - Загрузка прервана. - Загрузка завершена.

404	`NoSuchTagSet`	Указанному бакету не назначены метки.

Метод	Описание
PutBucket	Создает бакет.
GetBucket	Проверяет существование бакета и доступ к нему.
ListObjects	Возвращает список объектов в бакете.
ListBuckets	Возвращает список бакетов.
DeleteBucket	Удаляет бакет.
PutBucketVersioning	Включает или приостанавливает версионирование бакета.
GetBucketVersioning	Возвращает состояние опции версионирования.

416	`InvalidRange`	Неправильный диапазон в заголовке `Range`.
429	`TooManyRequests`	Слишком много запросов к ПО Яндекс Объектное хранилище. Снизьте частоту обращений.
	`InternalError`	Внутренняя ошибка ПО Яндекс Объектное хранилище. Повторите запрос.
501	`NotImplemented`	Переданный заголовок не обрабатывается ПО Яндекс Объектное хранилище.

Параметр	Описание
`bucket`	Имя бакета.

Заголовок	Описание
`X-Amz-Acl`	Устанавливает виды разрешений для бакета.
`X-Amz-Grant-Read`	Устанавливает получателю доступа разрешение на просмотр содержимого бакета и разрешение на чтение объектов в бакете.
`X-Amz-Grant-Write`	Устанавливает получателю доступа разрешение на запись объектов. Используйте обязательно вместе с `X-Amz-Grant-Read`, иначе ПО Яндекс Объектное хранилище ответит с кодом `501 Not Implemented`.
`X-Amz-Grant-Read-Acp`	Устанавливает получателю доступа разрешение на чтение ACL бакета.
`X-Amz-Grant-Write-Acp`	Устанавливает получателю доступа разрешение на запись ACL бакета.
`X-Amz-Grant-Full-Control`	Устанавливает получателю доступа разрешения: `READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` на бакет.

Значение для заголовков `X-Amz-Grant-*` представляет собой список получателей доступа, разделенный запятыми. Каждый получатель доступа идентифицируется структурой вида

`<тип_получателя_доступа>:<идентификатор_получателя_доступа>`. ПО Яндекс Объектное хранилище поддерживает следующие типы получателей:

- `id` — получатель доступа — учетная запись S3.
- `uri` — получатель доступа — публичная группа.

Пример:

```
X-Amz-Grant-Read: uri="http://acs.amazonaws.com/groups/s3/AuthenticatedUsers"
```

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Метод GetBucket

Возвращает метаданные бакета или ошибку.

Позволяет проверить:

- Существует ли бакет.
- Достаточно ли у пользователя прав для доступа к бакету.

Запрос

```
HEAD /{bucket} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных и означает, что бакет существует и у пользователя достаточно прав для доступа к нему.

Метод ListObjects

Возвращает список объектов в бакете.

При выдаче используется пагинация, за один запрос можно получить список не длиннее 1000 объектов. Если объектов больше, то необходимо выполнить несколько запросов подряд.

Примечание

У этого метода есть две версии:

- `listObjectsV2` — актуальная версия, более удобная в использовании.
- `listObjectsV1` — предыдущая версия.

Для вызова обоих методов используется один и тот же `URL`, но он отличается query-параметром. Чтобы вызвать `listObjectsV2`, используйте параметр `list-type=2`.

listObjectsV2

Запрос

```
GET /{bucket}?list-type=2&continuation-token=ContinuationToken&delimiter=Delimiter&encoding-type=EncodingType&max-keys=MaxKeys&prefix=Prefix&start-after=StartAfter HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Все перечисленные в таблице параметры необязательные.

Параметр	Описание
`continuation-token`	Используется для получения следующей части списка, если все результаты не помещаются в один ответ. Чтобы получить следующую часть списка, используйте значение `NextContinuationToken` из предыдущего ответа.

`delimiter`	Символ-разделитель. Если параметр указан, то ПО Яндекс Объектное хранилище рассматривает ключ как путь к файлу, где каталоги разделяются символом `delimiter`. В ответе на запрос пользователь увидит перечень “файлов” и “каталогов” в бакете. “Файлы” будут выведены в элементах `Contents`, а “каталоги” в элементах `CommonPrefixes`. Если в запросе указан еще и параметр `prefix`, то ПО Яндекс Объектное хранилище вернет перечень “файлов” и “каталогов” в “каталоге” `prefix`.
`encoding-type`	Кодировка ответа от сервера. ПО Яндекс Объектное хранилище по требованию клиента может закодировать ответ в требуемом виде. Возможные значения: `url`.
`max-keys`	Максимальное количество элементов в ответе. По умолчанию ПО Яндекс Объектное хранилище выдает не более 1000 элементов `Contents` и `CommonPrefixes`. Параметр следует использовать, если вам нужно получать менее 1000 элементов в одном ответе. Если под критерии отбора попадает больше ключей, чем поместилось в выдаче, то ответ содержит `<IsTruncated>true</IsTruncated>`. Чтобы получить все элементы выдачи, если их больше `max-keys`, необходимо выполнить несколько последовательных запросов к ПО Яндекс Объектное хранилище с параметром `continuation-token`, где для каждого запроса `continuation-token` равен значению элемента `NextContinuationToken` из предыдущего ответа.
`prefix`	Строка, с которой должен начинаться ключ. ПО Яндекс Объектное хранилище выберет только те ключи, которые начинаются с `prefix`. Может использоваться одновременно с параметром `delimiter`. В этом случае логика выдачи определяется параметром `delimiter`.
`start-after`	Ключ, с которого нужно начать листинг.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
   <IsTruncated>boolean</IsTruncated>
   <Contents>
      <ETag>string</ETag>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Size>integer</Size>
      <StorageClass>string</StorageClass>
   </Contents>
   ...
   <Name>string</Name>
   <Prefix>string</Prefix>
   <Delimiter>string</Delimiter>
   <MaxKeys>integer</MaxKeys>
   <CommonPrefixes>
      <Prefix>string</Prefix>
   </CommonPrefixes>
   ...
   <EncodingType>string</EncodingType>
   <KeyCount>integer</KeyCount>
   <ContinuationToken>string</ContinuationToken>
   <NextContinuationToken>string</NextContinuationToken>
   <StartAfter>string</StartAfter>
</ListBucketResult>
```

Элемент	Описание
`ListBucketResult`	Корневой элемент.

`IsTruncated`	Флаг, показывающий все ли результаты возвращены в этом ответе. `True` — не все. `False` — все. Путь: `/ListBucketResult/IsTruncated`.
`Contents`	Описание объекта. Ответ будет содержать столько элементов `Contents`, сколько ключей попало под условия запроса. Путь: `/ListBucketResult/Contents`.
`ETag`	MD5-хэш объекта. Метаданные в расчете хэша не участвуют. Путь: `/ListBucketResult/Contents/ETag`.
`Key`	Ключ объекта. Путь: `/ListBucketResult/Contents/Key`.
`LastModified`	Дата и время последнего изменения объекта. Путь: `/ListBucketResult/Contents/LastModified`.
`Size`	Размер объекта в байтах. Путь: `/ListBucketResult/Contents/Size`.
`StorageClass`	Класс хранилища объекта: `STANDARD`. Путь: `/ListBucketResult/Contents/StorageClass`.
`Name`	Имя бакета. Путь: `/ListBucketResult/Name`.
`Prefix`	Значение query-параметра `prefix`. Путь: `/ListBucketResult/Prefix`.
`Delimiter`	Значение query-параметра `delimiter`. Путь: `/ListBucketResult/Delimiter`.

`MaxKeys`	Значение query-параметра `max-keys`. Путь: `/ListBucketResult/MaxKeys`.
`CommonPrefixes`	Часть имени ключа, которая определяется при обработке query-параметров `delimiter` и `prefix`. Путь: `/ListBucketResult/CommonPrefixes`.
`EncodingType`	Кодировка, в которой ПО Яндекс Объектное хранилище представляет ключ в XML-ответе. Появляется, если клиент при запросе передал параметр `encoding-type`. Путь: `/ListBucketResult/EncodingType`.
`KeyCount`	Количество ключей, возвращенных запросом. Количество ключей всегда будет меньше или равно `MaxKeys`. Путь: `/ContinuationToken/KeyCount`.
`ContinuationToken`	Значение query-параметра `continuation-token`. Путь: `/ContinuationToken/ContinuationToken`.
`NextContinuationToken`	Значение, которое надо подставить в query-параметр `continuation-token` для получения следующей части списка, если весь список не поместился в текущий ответ. Возвращается только в том случае, если `IsTruncated = true`. Путь: `/ListBucketResult/NextContinuationToken`.
`StartAfter`	Значение query-параметра `start-after`. Путь: `/ListBucketResult/StartAfter`.

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Все перечисленные в таблице параметры необязательные.

Параметр	Описание
`delimiter`	Символ-разделитель. Если параметр указан, то ПО Яндекс Объектное хранилище рассматривает ключ как путь к файлу, где каталоги разделяются символом `delimiter`. В ответе на запрос пользователь увидит перечень “файлов” и “каталогов” в бакете. “Файлы” будут выведены в элементах `Contents`, а “каталоги” в элементах `CommonPrefixes`. Если в запросе указан еще и параметр `prefix`, то ПО Яндекс Объектное хранилище вернет перечень “файлов” и “каталогов” в “каталоге” `prefix`.
`encoding-type`	Кодировка ответа от сервера. ПО Яндекс Объектное хранилище по требованию клиента может закодировать ответ в требуемом виде. Возможные значения: `url`.
`marker`	Ключ, с которого начнется выдача. В результирующей выдаче ПО Яндекс Объектное хранилище оставит ключи, начиная со следующего за `marker`.
`max-keys`	Максимальное количество элементов в ответе. По умолчанию ПО Яндекс Объектное хранилище выдает не более 1000 элементов `Contents` и `CommonPrefixes`. Параметр следует использовать, если вам нужно получать менее 1000 элементов в одном ответе. Если под критерии отбора попадает больше ключей, чем поместилось в выдаче, то ответ содержит `<IsTruncated>true</IsTruncated>`. Чтобы получить все элементы выдачи, если их больше `max-keys`, необходимо выполнить несколько последовательных запросов к ПО Яндекс Объектное хранилище с параметром `marker`, где для каждого запроса `marker` равен значению элемента `NextMarker` из предыдущего ответа.
`prefix`	Строка, с которой должен начинаться ключ. ПО Яндекс Объектное хранилище выберет только те ключи, которые начинаются с `prefix`. Может использоваться одновременно с параметром `delimiter`. В этом случае логика выдачи определяется параметром `delimiter`.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в разделе {#T}.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
   <IsTruncated>boolean</IsTruncated>
   <Marker>string</Marker>
   <NextMarker>string</NextMarker>
   <Contents>
      <ETag>string</ETag>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Size>integer</Size>
      <StorageClass>string</StorageClass>
   </Contents>
   ...
   <Name>string</Name>
   <Prefix>string</Prefix>
   <Delimiter>string</Delimiter>
   <MaxKeys>integer</MaxKeys>
   <CommonPrefixes>
      <Prefix>string</Prefix>
   </CommonPrefixes>
   ...
   <EncodingType>string</EncodingType>
</ListBucketResult>
```

Элемент	Описание
`ListBucketResult`	Корневой элемент.

`IsTruncated`	Флаг, показывающий все ли результаты возвращены в этом ответе. `True` — не все. `False` — все. Путь: `/ListBucketResult/IsTruncated`.
`Marker`	Значение query-параметра `marker`. Путь: `/ListBucketResult/Marker`.
`NextMarker`	Значение, которое надо подставить в query-параметр `marker` для получения следующей части списка, если весь список не поместился в текущий ответ. Путь: `/ListBucketResult/NextMarker`.
`Contents`	Описание объекта. Ответ будет содержать столько элементов `Contents`, сколько ключей попало под условия запроса. Путь: `/ListBucketResult/Contents`.
`ETag`	MD5-хэш объекта. Метаданные в расчете хэша не участвуют. Путь: `/ListBucketResult/Contents/ETag`.
`Key`	Ключ объекта. Путь: `/ListBucketResult/Contents/Key`.
`LastModified`	Дата и время последнего изменения объекта. Путь: `/ListBucketResult/Contents/LastModified`.
`Size`	Размер объекта в байтах. Путь: `/ListBucketResult/Contents/Size`.
`StorageClass`	Класс хранилища объекта: `STANDARD`. Путь: `/ListBucketResult/Contents/StorageClass`.
`Name`	Имя бакета. Путь: `/ListBucketResult/Name`.

`Prefix`	Значение query-параметра `prefix`. Путь: `/ListBucketResult/Prefix`.
`Delimiter`	Значение query-параметра `delimiter`. Путь: `/ListBucketResult/Delimiter`.
`MaxKeys`	Значение query-параметра `max-keys`. Путь: `/ListBucketResult/MaxKeys`.
`CommonPrefixes`	Часть имени ключа, которая определяется при обработке query-параметров `delimiter` и `prefix`. Путь: `/ListBucketResult/CommonPrefixes`.
`EncodingType`	Кодировка, в которой ПО Яндекс Объектное хранилище представляет ключ в XML-ответе. Появляется, если клиент при запросе передал параметр `encoding-type`. Путь: `/ListBucketResult/EncodingType`.

Метод ListBuckets

Возвращает перечень бакетов, доступных пользователю.

Запрос

```
GET / HTTP/2
```

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<ListAllMyBucketsResult>
  <Buckets>
    <Bucket>
      <Name>bucket-name</Name>
      <CreationDate>date_time</CreationDate>
    </Bucket>
    ...
  </Buckets>
</ListAllMyBucketsResult>
```

Элемент	Описание
`Bucket`	Содержит описание бакета. Путь: `/ListAllMyBucketsResult/Buckets/Bucket`.
`Buckets`	Содержит перечень бакетов. Путь: `/ListAllMyBucketsResult/Buckets`.

`CreationDate`	Время создания бакета в формате `yyyy-mm-ddThh:mm:ss.timezone`. Путь: `/ListAllMyBucketsResult/Buckets/Bucket/CreationDate`.
`ListAllMyBucketsResult`	Корневой элемент ответа. Путь: `/ListAllMyBucketsResult`.
`Name`	Имя бакета. Путь: `/ListAllMyBucketsResult/Buckets/Bucket/Name`.

Метод DeleteBucket

Удаляет пустой бакет. Если бакет не пустой, то сначала нужно удалить все находящиеся в ней объекты.

Запрос

```
DELETE /{bucket} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод PutBucketVersioning

Включает или приостанавливает версионирование бакета.

Версионирование можно установить в одно из двух состояний:

- `Enabled` — включить управление версиями для объектов в бакете. Все новые объекты, добавляемые в бакет, будут получать уникальный идентификатор версии.
- `Suspended` — приостанавливает управление версиями для объектов в бакете. Все новые объекты, добавляемые в бакет, будут получать идентификатор версии = `null`.

Запрос

```
PUT /{bucket}?versioning HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`versioning`	Обязательный параметр для обозначения типа операции.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<VersioningConfiguration>
   <Status>string</Status>
</VersioningConfiguration>
```

Элемент	Описание
`Status`	Состояние опции версионирования бакета. Тип: Строка Возможные значения: `Enabled | Suspended`

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Метод GetBucketVersioning

Возвращает состояние опции версионирования бакета.

Запрос

```
GET /{bucket}?versioning HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`versioning`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<VersioningConfiguration>
   <Status>string</Status>
</VersioningConfiguration>
```

Элемент	Описание
`VersioningConfiguration`	Корневой элемент.
`Status`	Состояние опции версионирования бакета. Путь: `/VersioningConfiguration/Status` Тип: Строка Возможные значения: `Enabled

Метод ListObjectVersions

Возвращает метаданные обо всех версиях объектов в бакете.
Также можно использовать параметры запроса в качестве критерия выбора для возврата метаданных о подмножестве версий объекта.

Запрос

```
GET /{bucket}?versions&delimiter=Delimiter&encoding-type=EncodingType&key-marker=KeyMarker&max-keys=MaxKeys&prefix=Prefix&version-id-marker=VersionIdMarker
HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Все перечисленные в таблице параметры необязательные.

Параметр	Описание
`delimiter`	Символ-разделитель. Если параметр указан, то ПО Яндекс Объектное хранилище рассматривает ключ как путь к файлу, где каталоги разделяются символом `delimiter`. В ответе на запрос пользователь увидит перечень “файлов” и “каталогов” в бакете. “Файлы” будут выведены в элементах `Contents`, а “каталоги” в элементах `CommonPrefixes`. Если в запросе указан еще и параметр `prefix`, то ПО Яндекс Объектное хранилище вернет перечень “файлов” и “каталогов” в “каталоге” `prefix`.
`encoding-type`	Кодировка ответа от сервера. ПО Яндекс Объектное хранилище по требованию клиента может закодировать ответ в требуемом виде. Возможные значения: `url`.

`key-marker`	Ключ, с которого начнется выдача. В результирующей выдаче ПО Яндекс Объектное хранилище оставит ключи, начиная со следующего за `key-marker`.
`max-keys`	Максимальное количество элементов в ответе. По умолчанию ПО Яндекс Объектное хранилище выдает не более 1000 элементов `Contents` и `CommonPrefixes`. Параметр следует использовать, если вам нужно получать менее 1000 элементов в одном ответе. Если под критерии отбора попадает больше ключей, чем поместилось в выдаче, то ответ содержит `<IsTruncated>true</IsTruncated>`. Чтобы получить все элементы выдачи, если их больше `max-keys`, необходимо выполнить несколько последовательных запросов к ПО Яндекс Объектное хранилище с параметром `key-marker`, где для каждого запроса `key-marker` и `version-id-marker` равны значениям элементов `NextKeyMarker` и `NextVersionIdMarker` из предыдущего ответа.
`prefix`	Строка, с которой должен начинаться ключ. ПО Яндекс Объектное хранилище выберет только те ключи, которые начинаются с `prefix`. Может использоваться одновременно с параметром `delimiter`. В этом случае логика выдачи становится той, что указана в описании параметра `delimiter`.
`version-id-marker`	Версия объекта, с которой начинается выдача. В результирующей выдаче ПО Яндекс Объектное хранилище оставит версии, начиная со следующей за `version-id-marker`.

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult>
   <IsTruncated>boolean</IsTruncated>
   <KeyMarker>string</KeyMarker>
   <VersionIdMarker>string</VersionIdMarker>
   <NextKeyMarker>string</NextKeyMarker>
   <NextVersionIdMarker>string</NextVersionIdMarker>
   <Version>
      <ETag>string</ETag>
      <IsLatest>boolean</IsLatest>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Owner>
         <DisplayName>string</DisplayName>
         <ID>string</ID>
      </Owner>
      <Size>integer</Size>
      <StorageClass>string</StorageClass>
      <VersionId>string</VersionId>
   </Version>
   ...
   <DeleteMarker>
      <IsLatest>boolean</IsLatest>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Owner>
         <DisplayName>string</DisplayName>
         <ID>string</ID>
      </Owner>
      <VersionId>string</VersionId>
   </DeleteMarker>
   ...
   <Name>string</Name>
   <Prefix>string</Prefix>
   <Delimiter>string</Delimiter>
   <MaxKeys>integer</MaxKeys>
   <CommonPrefixes>
      <Prefix>string</Prefix>
   </CommonPrefixes>
   ...
   <EncodingType>string</EncodingType>
</ListVersionsResult>
```

Элемент	Описание
`ListVersionsResult`	Корневой элемент

`CommonPrefixes`	Часть имени ключа, которая определяется при обработке query параметров `delimiter` и `prefix`. Путь: `/ListVersionsResult/CommonPrefixes`.
`DeleteMarker`	Контейнер для объекта, который является маркером удаления. Путь: `/ListVersionsResult/DeleteMarker`.
`Delimiter`	Значение query параметра `delimiter`. Путь: `/ListVersionsResult/Delimiter`.
`EncodingType`	Кодировка, в которой ПО Яндекс Объектное хранилище представляет ключ в XML-ответе. Появляется, если клиент при запросе передал параметр `encoding-type`. Путь: `/ListVersionsResult/EncodingType`.
`IsTruncated`	Признак неполноты списка. Если `IsTruncated` — `true`, то это означает, что ПО Яндекс Объектное хранилище вернул не полный список частей. Путь: `/ListVersionsResult/IsTruncated`.
`KeyMarker`	Последний ключ, возвращенный в неполном ответе. Путь: `/ListVersionsResult/KeyMarker`.
`MaxKeys`	Значение query параметра `max-keys`. Путь: `/ListBucketResult/MaxKeys`.
`Name`	Название бакета. Путь: `/ListBucketResult/Name`.
`NextKeyMarker`	Значение, которое надо подставить в query параметр `key-marker` для получения следующей части списка, если весь список не поместился в текущий ответ. Путь: `/ListBucketResult/NextMarker`.

`NextVersionIdMarker`	Значение, которое надо подставить в query параметр `version-id-marker` для получения следующей части списка, если весь список не поместился в текущий ответ. Путь: `/ListBucketResult/NextVersionIdMarker`.
`Prefix`	Значение query параметра `prefix`. Путь: `/ListBucketResult/Prefix`.
`Version`	Версия объекта. Путь: `/ListBucketResult/Version`.
`VersionIdMarker`	Отмечает последнюю версию ключа, возвращенную в усеченном ответе. Путь: `/ListBucketResult/VersionIdMarker`.

Метод PutObjectLockConfiguration

Настраивает механизм блокировок версий объектов в версионируемом бакете: включает или выключает механизм и настраивает блокировки по умолчанию.

Когда механизм блокировок включен, вы можете установить на версию объекта блокировку, то есть запрет на удаление или перезапись:

- при загрузке объекта (метод `PutObject`);
- после загрузки объекта (методы `PutObjectRetention` и `PutObjectLegalHold`).

Запрос

```
PUT /{bucket}?object-lock HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Заголовки

Используйте в запросе только общие заголовки.

Схема данных

```
<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <ObjectLockEnabled>string</ObjectLockEnabled>
   <Rule>
      <DefaultRetention>
         <Mode>string</Mode>
         <Days>integer</Days>
         <Years>integer</Years>
      </DefaultRetention>
   </Rule>
</ObjectLockConfiguration>
```

Элемент	Описание

`ObjectLockConfiguration`	Корневой элемент. Чтобы выключить механизм блокировок, передайте этот параметр с пустым значением, например `<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/" />`. Путь: `ObjectLockConfiguration`.
`ObjectLockEnabled`	Статус механизма блокировок: • `Enabled` — механизм блокировок включен. Если элемент не указан, будет выведено сообщение об ошибке `InvalidRequest`, а механизм блокировок не включится. Путь: `ObjectLockConfiguration\ObjectLockEnabled`.
`Rule`	Настройки блокировок. Путь: `ObjectLockConfiguration\Rule`.
`DefaultRetention`	Настройки блокировок по умолчанию. Путь: `ObjectLockConfiguration\Rule\DefaultRetention`.
`Mode`	Тип блокировки по умолчанию: • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Mode`.
`Days`	Срок блокировки по умолчанию в днях от момента загрузки версии объекта. Должен быть положительным целым числом. Нельзя указывать вместе с `Years`. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Days`.
`Years`	Срок блокировки по умолчанию в годах от момента загрузки версии объекта. Должен быть положительным целым числом. Нельзя указывать вместе с `Days`. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Years`.

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetObjectLockConfiguration

Возвращает настройки механизма блокировок версий объектов в версионируемом бакете.

Запрос

```
GET /{bucket}?object-lock HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Схема данных

```
<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <ObjectLockEnabled>string</ObjectLockEnabled>
   <Rule>
      <DefaultRetention>
         <Mode>string</Mode>
         <Days>integer</Days>
         <Years>integer</Years>
      </DefaultRetention>
   </Rule>
</ObjectLockConfiguration>
```

Элемент	Описание

`ObjectLockEnabled`	Статус механизма блокировок: • `Enabled` — механизм блокировок включен. Если элемент не указан, механизм блокировок выключен. Путь: `ObjectLockConfiguration\ObjectLockEnabled`.
`Rule`	Настройки блокировок. Путь: `ObjectLockConfiguration\Rule`.
`DefaultRetention`	Настройки блокировок по умолчанию. Путь: `ObjectLockConfiguration\Rule\DefaultRetention`.
`Mode`	Тип блокировки по умолчанию: • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Mode`.
`Days`	Срок блокировки по умолчанию в днях от момента загрузки версии объекта. Должен быть положительным целым числом. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Days`.
`Years`	Срок блокировки по умолчанию в годах от момента загрузки версии объекта. Должен быть положительным целым числом. Путь: `ObjectLockConfiguration\Rule\DefaultRetention\Years`.

Метод PutBucketTagging

Добавляет метки бакету. При этом перезаписываются все имеющиеся у бакета метки.

Запрос

```
PUT /{bucket}?tagging HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query-параметры

Параметр	Описание
`tagging`	Обязательный параметр для обозначения типа операции.

Схема данных

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <TagSet>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Элемент	Описание
`Tagging`	Корневой элемент.
`TagSet`	Массив меток.
`Tag`	Контейнер для метки.

`Key`	Ключ метки. Тип: string.
`Value`	Значение метки. Тип: string.

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Метод GetBucketTagging

Возвращает метки бакета.

Запрос

```
GET /{bucket}?tagging HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.

Query-параметры

Параметр	Описание
`tagging`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которых описана ниже.

Схема данных

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <TagSet>
      <Tag>
         <Key>string</Key>
```

```
        <Value>string</Value>
      </Tag>
    </TagSet>
</Tagging>
```

Элемент	Описание
`Tagging`	Корневой элемент.
`TagSet`	Массив меток.
`Tag`	Контейнер для метки.
`Key`	Ключ метки. Тип: string.
`Value`	Значение метки. Тип: string.

Метод DeleteBucketTagging

Удаляет метки бакета.

Запрос

```
DELETE /{bucket}?tagging HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.

Query-параметры

Параметр	Описание
`tagging`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Все методы Object

Метод	Описание
PutObject	Загружает объект.
GetObject	Выгружает объект.
PatchObject	Частично изменяет объект.
CopyObject	Копирует объект, хранящийся в бакете.
GetObjectMeta	Выгружает метаданные объекта.
DeleteObject	Удаляет объект.
DeleteMultipleObjects	Удаляет объекты по списку.
OptionsObject	Проверяет возможность CORS-запроса к объекту.
PutObjectTagging	Добавляет метки объекту.
GetObjectTagging	Возвращает метки объекта.
DeleteObjectTagging	Удаляет метки объекта.

Метод PutObject

Загружает объект и его метаданные.

> **Примечание**
>
> ПО Яндекс Объектное хранилище не блокирует объект на запись и может принимать одновременно несколько запросов на запись одного объекта, однако по умолчанию пользователь сможет получить только последний записанный объект. Чтобы при перезаписи или удалении объектов сохранялась история, включите версионирование.

Чтобы убедиться, что объект передан по сети без повреждений, используйте заголовок `Content-MD5`. ПО Яндекс Объектное хранилище вычислит `MD5` для сохраненного объекта и если вычисленная `MD5` не совпадет с переданной в заголовке, вернет ошибку. Эту проверку можно выполнить и на стороне клиента, сравнив `ETag` из ответа ПО Яндекс Объектное хранилище с предварительно вычисленной `MD5`.

> **Важно**
>
> Если в бакете настроены блокировки версий объектов по умолчанию, использовать заголовок `Content-MD5` обязательно.

Запрос

```
PUT /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Заголовки

Используйте в запросе необходимые общие заголовки.

Дополнительно можно использовать заголовки, перечисленные в таблице ниже.

Заголовок	Описание

`X-Amz-Meta-*`	Пользовательские метаданные объекта. Все заголовки, начинающиеся с `X-Amz-Meta-`, ПО Яндекс Объектное хранилище преобразует по правилу: `X-Amz-Meta-foo-bar_baz` → `X-Amz-Meta-Foo-Bar_baz`. Общий размер пользовательских заголовков не должен превышать 2KB. Размер пользовательских данных определяется как длина строки в кодировке UTF-8. В размере учитываются и названия заголовков и их значения.
`X-Amz-Storage-Class`	Класс хранилища объекта. Может иметь любое из значений: • `STANDARD` — стандартное хранилище.
`X-Amz-Object-Lock-Mode`	Тип временной блокировки, устанавливаемой на объект (если бакет версионируемый и в нем включен механизм блокировок): • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Вы можете установить на версию объекта только временную блокировку (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочную блокировку (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.
`X-Amz-Object-Lock-Retain-Until-Date`	Дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с заголовком `X-Amz-Object-Lock-Mode`.

`X-Amz-Object-Lock-Legal-Hold`	Статус бессрочной блокировки, устанавливаемой на объект (если бакет версионируемый и в нем включен механизм блокировок): • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. Вы можете установить на версию объекта только временную блокировку (заголовки `X-Amz-Object-Lock-Mode` и

Заголовок	Описание
`X-Amz-Acl`	Устанавливает предопределенный ACL для объекта.
`X-Amz-Grant-Read`	Устанавливает получателю доступа разрешение на чтение объекта.
`X-Amz-Grant-Read-Acp`	Устанавливает получателю доступа разрешение на чтение ACL объекта.
`X-Amz-Grant-Write-Acp`	Устанавливает получателю доступа разрешение на запись ACL объекта.
`X-Amz-Grant-Full-Control`	Устанавливает получателю доступа разрешения: `READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` на объект.

Значение для заголовков `X-Amz-Grant-*` представляет собой разделенный запятыми список получателей доступа. Каждый получатель доступа идентифицируется структурой вида `<тип_получателя_доступа>:<идентификатор_получателя_доступа>`. ПО Яндекс Объектное хранилище поддерживает следующие типы получателей:

- `id` — получатель доступа — учетная запись S3.
- `uri` — получатель доступа — публичная группа.

Пример:

```
X-Amz-Grant-Read: uri="http://acs.amazonaws.com/groups/s3/AuthenticatedUsers"
```

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetObject

Возвращает объект.

Запрос

```
GET /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`response-content-type`	Устанавливает заголовок ответа `Content-Type`.
`response-content-language`	Устанавливает заголовок ответа `Content-Language`.
`response-expires`	Устанавливает заголовок ответа `Expires`.
`response-cache-control`	Устанавливает заголовок ответа `Cache-Control`.
`response-content-disposition`	Устанавливает заголовок ответа `Content-Disposition`.
`response-content-encoding`	Устанавливает заголовок ответа `Content-Encoding`.
`versionId`	Ссылка на конкретную версию объекта.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также в запросе можно использовать следующие заголовки:

Заголовок	Описание

`Range`	Определяет диапазон байт для загрузки из объекта. Подробнее про заголовок Range читайте в спецификации HTTP http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35.
`If-Modified-Since`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если он изменялся после указанного времени. - Код 304. Если объект не изменялся после указанного времени. Если в запросе одновременно присутствуют заголовки `If-Modified-Since` и `If-None-Match` и проверки по ним разрешаются как `If-Modified-Since -> true` и `If-None-Match -> false`, то ПО Яндекс Объектное хранилище возвращает код 304. Подробности смотрите в RFC 7232.
`If-Unmodified-Since`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если он не изменялся с указанного времени. - Код 412. Если объект не изменялся с указанного времени. Если в запросе одновременно присутствуют заголовки `If-Unmodified-Since` и `If-Match` и проверки по ним разрешаются как `If-Unmodified-Since -> false` и `If-Match -> true`, то ПО Яндекс Объектное хранилище возвращает код 200 и запрошенные данные. Подробности смотрите в RFC 7232.
`If-Match`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если его `ETag` совпадает с переданным. - Код 412. Если его `ETag` не совпадает с переданным. Если в запросе одновременно присутствуют заголовки `If-Unmodified-Since` и `If-Match` и проверки по ним разрешаются как `If-Unmodified-Since -> false` и `If-Match -> true`, то ПО Яндекс Объектное хранилище возвращает код 200 и запрошенные данные. Подробности смотрите в RFC 7232.
`If-None-Match`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если его `ETag` не совпадает с переданным. - Код 304. Если его `ETag` совпадает с переданным. Если в запросе одновременно присутствуют заголовки `If-Modified-Since` и `If-None-Match` и проверки по ним разрешаются как `If-Modified-Since -> true` и `If-None-Match -> false`, то ПО Яндекс Объектное хранилище возвращает код 304. Подробности смотрите в RFC 7232.

Ответ

Заголовки

Помимо общих заголовков вы можете увидеть в ответе заголовки, перечисленные в таблице ниже.

Заголовок	Описание
`X-Amz-Meta-*`	Пользовательские метаданные объекта.
`X-Amz-Object-Lock-Mode`	Тип временной блокировки, установленной на объект (если бакет версионируемый и в нем включен механизм блокировок): • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. На версию объекта может быть установлена только временная блокировка (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочная блокировка (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.
`X-Amz-Object-Lock-Retain-Until-Date`	Дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с заголовком `X-Amz-Object-Lock-Mode`.
`X-Amz-Object-Lock-Legal-Hold`	Статус бессрочной блокировки, установленной на объект (если бакет версионируемый и в нем включен механизм блокировок): • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. На версию объекта может быть установлена только временная блокировка (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочная блокировка (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод PatchObject

Частично изменяет и дозаписывает данные объекта.

> **Примечание**
>
> Механизм частичного изменения объекта не входит в стандартную функциональность S3 API и доступен для бакетов с выключенным версионированием.

В запросе передаются изменяемый диапазон объекта и новые данные.

Допускается одновременное изменение объекта несколькими запросами.

Запрос

```
PATCH /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Заголовки

Используйте в запросе общие заголовки, а также заголовки, представленные ниже:

Заголовок	Описание

`Content-Range`	Обязательный параметр. Значение: `bytes {<начальный_байт>}-{<конечный_байт>}/*`. Границы диапазона включены. Максимальная длина диапазона — 5 ГБ. Заголовок `Content-Length` должен быть равен длине `Content-Range`. Чтобы дозаписать данные в объект, укажите значение конечного байта большее, чем размер объекта. Значение начального байта не может быть больше, чем размер объекта. Формат заголовка соответствует спецификации RFC 9110 со следующими исключениями: • Параметр заголовка `complete-length` игнорируется. • Параметр заголовка `last-pos` опциональный.
`X-Yc-S3-Patch-Append-Part-Size`	Опциональный параметр. Значение: `{размер_новой_составной_части}`. Размер новых составных частей объекта, загруженного по частям, при дозаписи в конец объекта. В байтах. Если последняя составная часть объекта достигла указанного размера, следующая дозапись будет сохранена как новая составная часть. Значение по умолчанию — 25 МБ.
`If-Match`	Опциональный параметр. Условие для частичного изменения объекта. Если `ETag` объекта равен заданному в заголовке, то операция выполняется. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412 `Precondition Failed`. Можно использовать вместе с заголовком `If-Unmodified-Since`.
`If-Unmodified-Since`	Опциональный параметр. Условие для частичного изменения объекта. Операция выполняется, если объект не менялся с указанного времени. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412 `Precondition Failed`. Можно использовать вместе с заголовком

Ошибка	Описание

`MissingContentRange`	В запросе отсутствует заголовок `Content-Range`.
`MalformedPatchAppendPartSize`	В запросе неверно указано значение заголовка `X-Yc-S3-Patch-Append-Part-Size`.
`ObjectVersionPatchConflict`	Во время частичного изменения объект был перезаписан новой версией.
`ConcurrentUpdatesPatchConflict`	Из-за большого количества одновременных запросов не удалось разрешить конфликт.

Схема данных

```
<?xml version="1.0" encoding="UTF-8"?>
<PatchObjectResult>
    <Object>
        <LastModified>2023-07-21T11:46:16.357Z</LastModified>
        <ETag>&#34;daebcb5e72f14b690c18018c9f92af05-2&#34;</ETag>
    </Object>
</PatchObjectResult>
```

Тег	Описание
`PatchObjectResult`	Корневой элемент.
`Object`	Родительский тег для результатов изменения.
`LastModified`	Дата и время последнего изменения объекта. При частичном изменении объекта тег не изменяется.
`ETag`	ETag обновленного объекта.

Метод CopyObject

Создает копию объекта, хранящегося в бакете. Объекты, размером до 5 ГБ можно скопировать одной операцией `CopyObject`, если объект больше, чем 5 ГБ, то необходимо применять операцию `CopyObjectPart`.

Чтобы указать источник для копирования, используйте заголовок `X-Amz-Copy-Source`. Запрос на копирование не должен передавать каких-либо данных, кроме заголовков.

> **Примечание**
>
> ПО Яндекс Объектное хранилище не блокирует объект на запись и может одновременно принять несколько запросов, копирующих объекты в один и тот же результирующий объект. После завершения всех запросов результирующим объектом окажется тот, чья операция копирования запустилась последней.

Метаданные объекта копируются вместе с объектом. При необходимости их можно изменить, задав в явном виде соответствующие заголовки.

Также с помощью заголовков можно:

- Добавить условия для копирования объекта.
- Установить блокировку на объект (если бакет версионируемый и механизм блокировок включен).

Пользователь должен иметь разрешение на чтение исходного объекта и разрешение на запись в результирующий бакет.

Запрос

```
PUT /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя результирующего бакета.
`key`	Ключ результирующего объекта.

Заголовки

Обязательные заголовки перечислены в таблице ниже.

Заголовок	Описание
`X-Amz-Copy-Source`	Имя бакета и ключ объекта, который будет копироваться, разделенные символом `/`. Например, `X-Amz-Copy-Source: /source_bucket/sourceObject`. Если в бакете включено версионирование, то вы можете скопировать определенную версию объекта. Для этого укажите в заголовке идентификатор версии объекта — добавьте `?versionId=<version-id>` к значению заголовка, например: `/mybucket/image.png?versionId=0005E4A66AD990A4`. Если не указывать идентификатор версии, то будет скопирована последняя версия объекта.

Также используйте необходимые общие заголовки.

Заголовки, описанные в таблице ниже используйте, если вам необходимо изменить поведение метода `copy` по умолчанию.

Заголовок	Описание
`X-Amz-Metadata-Directive`	Режим копирования метаданных. Если значение заголовка `COPY`, то метаданные объекта копируются, а все заголовки `X-Amz-Meta-*` игнорируются. Это поведение метода `copy` по умолчанию. Если значение заголовка `REPLACE`, то метаданные объекта подменяются указанными в запросе.
`X-Amz-Copy-Source-If-Match`	Условие для копирования объекта. Если `ETag` объекта равен заданному в заголовке, то объект копируется. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Unmodified-Since`.

`X-Amz-Copy-Source-If-None-Match`	Условие для копирования объекта. Если `ETag` объекта не равен заданному в заголовке, то объект копируется. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Modified-Since`.
`X-Amz-Copy-Source-If-Unmodified-Since`	Условие для копирования объекта. Объект копируется, если он не изменялся с указанного времени. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Match`.
`X-Amz-Copy-Source-If-Modified-Since`	Условие для копирования объекта. Объект копируется, если он изменился с указанного времени. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-None-Match`.
`X-Amz-Storage-Class`	Класс хранилища объекта. Может иметь любое из значений: • `STANDARD` — стандартное хранилище.

`X-Amz-Object-Lock-Mode`	Тип временной блокировки, устанавливаемой на объект (если бакет версионируемый и в нем включен механизм блокировок): • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Вы можете установить на версию объекта только временную блокировку (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочную блокировку (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.
`X-Amz-Object-Lock-Retain-Until-Date`	Дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с заголовком `X-Amz-Object-Lock-Mode`.
`X-Amz-Object-Lock-Legal-Hold`	Статус бессрочной блокировки, устанавливаемой на объект (если бакет версионируемый и в нем включен механизм блокировок): • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. Вы можете установить на версию объекта только временную блокировку (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочную блокировку (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.

Заголовок	Описание

`X-Amz-Storage-Class`	Класс хранилища объекта. Может иметь значения: • `STANDARD` — стандартное хранилище.

нные объекта.

меся с
Объектное
о правилу:
`X-Amz-Meta-foo-bar_baz` → `X-Amz-Meta-Foo-Bar_baz`.

Общий размер пользовательских заголовков не должен превышать 2KB. Размер пользовательских данных определяется как длина строки в кодировке UTF-8. В размере учитываются и названия заголовков и их значения.

Если `X-Amz-Metadata-Directive: COPY` то

Элемент	Описание
`CopyObjectResult`	Содержит элементы ответа. Путь: `/CopyObjectResult`.
`ETag`	`ETag` результирующего объекта. Поскольку при вычислении `ETag` не учитываются метаданные, то `ETag` исходного и результирующего объекта должны быть равны. Путь: `/CopyObjectResult/ETag`.
`LastModified`	Дата последнего изменения объекта. Путь: `/CopyObjectResult/LastModified`.

Метод GetObjectMeta

Возвращает метаданные объекта.

Метод эквивалентен методу `GetObject`, но в ответе отсутствует сам объект.

Запрос

```
HEAD /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также в запросе можно использовать следующие заголовки:

Заголовок	Описание
`Range`	Определяет диапазон байт для загрузки из объекта. Подробнее про заголовок Range читайте в спецификации HTTP http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35.
`If-Modified-Since`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если он изменялся после указанного времени. - Код 304. Если объект не изменялся после указанного времени. Если в запросе одновременно присутствуют заголовки `If-Modified-Since` и `If-None-Match` и проверки по ним разрешаются как `If-Modified-Since -> true` и `If-None-Match -> false`, то ПО Яндекс Объектное хранилище возвращает код 304. Подробности смотрите в RFC 7232.

`If-Unmodified-Since`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если он не изменялся с указанного времени. - Код 412. Если объект не изменялся с указанного времени. Если в запросе одновременно присутствуют заголовки `If-Unmodified-Since` и `If-Match` и проверки по ним разрешаются как `If-Unmodified-Since -> false` и `If-Match -> true`, то ПО Яндекс Объектное хранилище возвращает код 200 и запрошенные данные. Подробности смотрите в RFC 7232.
`If-Match`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если его `ETag` совпадает с переданным. - Код 412. Если его `ETag` не совпадает с переданным. Если в запросе одновременно присутствуют заголовки `If-Unmodified-Since` и `If-Match` и проверки по ним разрешаются как `If-Unmodified-Since -> false` и `If-Match -> true`, то ПО Яндекс Объектное хранилище возвращает код 200 и запрошенные данные. Подробности смотрите в RFC 7232.
`If-None-Match`	Если указан, то ПО Яндекс Объектное хранилище возвращает: - Объект. Если его `ETag` не совпадает с переданным. - Код 304. Если его `ETag` совпадает с переданным. Если в запросе одновременно присутствуют заголовки `If-Modified-Since` и `If-None-Match` и проверки по ним разрешаются как `If-Modified-Since -> true` и `If-None-Match -> false`, то ПО Яндекс Объектное хранилище возвращает код 304. Подробности смотрите в RFC 7232.

Помимо общих заголовков вы можете увидеть в ответе заголовки, перечисленные в таблице ниже.

Заголовок	Описание
`X-Amz-Meta-*`	Пользовательские метаданные объекта.

`X-Amz-Object-Lock-Mode`	Тип временной блокировки, установленной на объект (если бакет версионируемый и в нем включен механизм блокировок): • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. На версию объекта может быть установлена только временная блокировка (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочная блокировка (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.
`X-Amz-Object-Lock-Retain-Until-Date`	Дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с заголовком `X-Amz-Object-Lock-Mode`.
`X-Amz-Object-Lock-Legal-Hold`	Статус бессрочной блокировки, установленной на объект (если бакет версионируемый и в нем включен механизм блокировок): • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. На версию объекта может быть установлена только временная блокировка (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочная блокировка (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.

Метод DeleteObject

Удаляет объект.

Запрос

```
DELETE /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`versionId`	Ссылка на конкретную версию объекта.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также, если на версию объекта в версионируемом бакете установлена временная управляемая блокировка (governance-mode retention), обязательно используйте заголовок, описанный ниже, чтобы обойти блокировку и подтвердить удаление. Удалить версию объекта с блокировкой может только учетная запись S3 с ролью `storage.admin`. Проверить блокировку можно с помощью метода `GetObjectRetention`.

Заголовок	Описание
`X-Amz-Bypass-Governance-Retention`	Заголовок, подтверждающий обход временной управляемой блокировки. Укажите значение `true`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод DeleteMultipleObjects

Удаляет объекты по списку ключей, переданному в запросе.

Выполняется быстрее, чем удаление тех же объектов по одному отдельными запросами.

Список на удаление может содержать не более 1000 ключей.

Если объекта не существует, то ПО Яндекс Объектное хранилище в ответе отметит его как удаленный.

Ответ можно настроить так, что ПО Яндекс Объектное хранилище выдаст одну из выборок:

- Статусы всех операций удаления.
- Только статусы с ошибкой удаления. В этом случае, если не произошло ни одной ошибки, ответ будет пустым.

Запрос

```
POST /{bucket}?delete HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`delete`	Флаг, обозначающий операцию удаления.

Заголовки

Используйте в запросе необходимые общие заголовки.

Для данного запроса заголовки `Content-MD5` и `Content-Length` обязательны.

Также, если на версии объектов в версионируемом бакете установлены временные управляемые блокировки (governance-mode retention), обязательно используйте заголовок, описанный ниже, чтобы обойти блокировки и подтвердить удаление. Удалить версию объекта с блокировкой может только учетная запись S3 с ролью `storage.admin`. Проверить блокировку можно с помощью метода `GetObjectRetention`.

Заголовок	Описание
`X-Amz-Bypass-Governance-Retention`	Заголовок, подтверждающий обход временной управляемой блокировки. Укажите значение `true`.

Схема данных

Перечень ключей на удаление передается в XML формате.

```
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
    <Quiet>true</Quiet>
    <Object>
         <Key>Key</Key>
    </Object>
    ...
</Delete>
```

Тег	Описание
`Delete`	Содержит тело ответа. Путь: `/Delete`.
`Quiet`	`<Quiet>true</Quiet>` устанавливает “тихий” режим. ПО Яндекс Объектное хранилище запишет в ответ только ошибки удаления. Если ошибок нет, запрос не возвращает тело ответа. Если на момент запроса указанный в нем объект не существует, вернется результат `Deleted`. Если не указан, то значение по умолчанию — `false`. Путь: `/Delete/Quiet`.
`Object`	Содержит параметры удаления объекта. Путь: `/Delete/Object`.
`Key`	Ключ объекта. Путь: `/Delete/Object/Key`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Структура данных

```
<DeleteResult>
  <Deleted>
    <Key>some/key.txt</Key>
  </Deleted>
  <Error>
    <Key>some/another/key.txt</Key>
    <Code>TextErrorCode</Code>
    <Message>Describing message</Message>
  </Error>
</DeleteResult>
```

Тег	Описание
`DeleteResult`	Тело ответа. Путь: `/DeleteResult`.
`Deleted`	Успешно удаленный объект. Отсутствует, если в запросе был выставлен `<Quiet>true</Quiet>`. Путь: `/DeleteResult/Deleted`.
`Key`	Ключ объекта. Путь: `/DeleteResult/Deleted/Key` или `/DeleteResult/Error/Key`
`Error`	Ошибка удаления объекта. Путь: `/DeleteResult/Error`.
`Code`	Код ошибки. Путь: `/DeleteResult/Error/Code`.
`Message`	Описание ошибки. Путь: `/DeleteResult/Error/Message`.

Метод OptionsObject

Проверяет возможность CORS-запроса к объекту.

Запрос

```
OPTIONS /{bucket}/{key} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Заголовки

Заголовок	Описание
`Origin`	Исходный домен запроса. Например, `http://www.example.com`. Обязательный.
`Access-Control-Request-Method`	HTTP метод, которым будет отправлен запрос к ресурсу. Обязательный.
`Access-Control-Request-Headers`	Список заголовков, которые будут отправлены в последующем запросе к объекту. Заголовки отделяются запятыми. Необязательный.

Также, используйте необходимые общие заголовки

Ответ

Заголовки

Кроме общих заголовков, ответ может содержать:

Заголовок	Описание
`Access-Control-Allow-Origin`	Домен, который был передан в заголовке `Origin` запроса. Если в конфигурации CORS в элементе `AllowedOrigin` задано `*`, то значение заголовка `Access-Control-Allow-Origin` также будет `*`. Если доступ с домена запрещен, то ПО Яндекс Объектное хранилище возвращает ошибку 403 и отсутствуют все заголовки `Access-Control-*`.
`Access-Control-Max-Age`	Допустимый период (в секундах) кэширования ответа.
`Access-Control-Allow-Methods`	Допустимые для использования в запросе методы. Если допустимых методов нет, то ПО Яндекс Объектное хранилище возвращает ошибку 403 и отсутствуют все заголовки `Access-Control-*`.
`Access-Control-Allow-Headers`	Список HTTP-заголовков, которые можно использовать в последующем запросе к объекту. Если все заголовки запрещены, то этот заголовок не входит в ответ.
`Access-Control-Expose-Headers`	Список HTTP-заголовков, которые получит JavaScript-клиент.

Коды ответов

Метод возвращает:

- 200 — если запросы к объекту разрешены.
- 403 — если запросы к объекту запрещены.

Подробные описания кодов ответов смотрите в соответствующем разделе.

Метод PutObjectRetention

Устанавливает на версию объекта временную блокировку либо настраивает или снимает уже установленную блокировку.

Чтобы на версии объектов можно было устанавливать блокировки, в бакете должен быть включен механизм блокировок. Включить его можно с помощью метода `PutObjectLockConfiguration`.

Изменить установленную блокировку может только пользователь с ролью `storage.admin`. При изменении управляемой блокировки нужно использовать заголовок, подтверждающий обход блокировки. Строгую блокировку можно только продлить. Проверить блокировку можно с помощью метода `GetObjectRetention`.

Запрос

```
PUT /{bucket}/{key}?retention&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`retention`	Обязательный параметр для обозначения типа операции.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также, если на версию объекта установлена временная управляемая блокировка (governance-mode retention), обязательно используйте заголовок, описанный ниже, чтобы обойти блокировку и подтвердить ее изменение.

Заголовок	Описание

`X-Amz-Bypass-Governance-Retention`	Заголовок, подтверждающий обход временной управляемой блокировки. Укажите значение `true`.

Схема данных

```
<Retention xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Mode>string</Mode>
    <RetainUntilDate>timestamp</RetainUntilDate>
</Retention>
```

Элемент	Описание
`Retention`	Корневой элемент с настройками блокировки. Чтобы снять управляемую блокировку (если у вас есть роль `storage.admin`), оставьте элемент пустым. Путь: `Retention`.
`Mode`	Тип блокировки: • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Путь: `Retention\Mode`.
`RetainUntilDate`	Дата и время окончания блокировки в формате RFC3339. Например, `2025-01-01T00:00:00`. Конец блокировки указывается в часовом поясе UTC±00:00. Чтобы указать другой часовой пояс, добавьте к концу записи `+` или `-` и смещение от UTC±00:00. Путь: `Retention\RetainUntilDate`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод PutObjectLegalHold

Устанавливает на версию объекта бессрочную блокировку или снимает ее.

Чтобы на версии объектов можно было устанавливать блокировки, в бакете должен быть включен механизм блокировок. Включить его можно с помощью метода `PutObjectLockConfiguration`.

Установить или снять блокировку может учетная запись S3 с ролью `storage.uploader`. Проверить блокировку можно с помощью метода `GetObjectLegalHold`.

Запрос

```
PUT /{bucket}/{key}?legal-hold&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`legal-hold`	Обязательный параметр для обозначения типа операции.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе необходимые общие заголовки.

Схема данных

```
<LegalHold xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>string</Status>
</LegalHold>
```

Элемент	Описание

`Status`	Статус бессрочной блокировки: • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. Путь: `LegalHold\Status`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetObjectRetention

Возвращает настройки временной блокировки, установленной на версию объекта.

Запрос

```
GET /{bucket}/{key}?retention&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`retention`	Обязательный параметр для обозначения типа операции.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Схема данных

```
<Retention xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Mode>string</Mode>
```

```
    <RetainUntilDate>timestamp</RetainUntilDate>
</Retention>
```

Элемент	Описание
`Mode`	Тип блокировки: • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Путь: `Retention\Mode`.
`RetainUntilDate`	Дата и время окончания блокировки в формате RFC3339. Например, `2025-01-01T00:00:00`. Конец блокировки указывается в часовом поясе UTC±00:00. Чтобы указать другой часовой пояс, добавьте к концу записи `+` или `-` и смещение от UTC±00:00. Путь: `Retention\RetainUntilDate`.

Метод GetObjectLegalHold

Возвращает настройки бессрочной блокировки, установленной на версию объекта.

Запрос

```
GET /{bucket}/{key}?legal-hold&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`legal-hold`	Обязательный параметр для обозначения типа операции.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Схема данных

```
<LegalHold xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>string</Status>
```

```
</LegalHold>
```

Элемент	Описание
`Status`	Статус бессрочной блокировки: • `ON` — блокировка установлена. • `OFF` — блокировка не установлена. Путь: `LegalHold\Status`.

Метод PutObjectTagging

Добавляет метки объекту в бакете. При этом перезаписываются все имеющиеся у объекта метки.

Запрос

```
PUT /{bucket}/{key}?tagging&versionId={versionId} HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`tagging`	Тип операции. Обязательный параметр.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Схема данных

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <TagSet>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Элемент	Описание
`Tagging`	Корневой элемент.
`TagSet`	Массив меток.

`Tag`	Контейнер для метки.
`Key`	Ключ метки. Тип: string.
`Value`	Значение метки. Тип: string.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Кроме общих заголовков, ответ может содержать:

Заголовок	Описание
`x-amz-version-id`	Идентификатор версии объекта.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Метод GetObjectTagging

Возвращает метки объекта в бакете.

Запрос

```
GET /{bucket}/{key}?tagging&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`tagging`	Тип операции. Обязательный параметр.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе общие заголовки.

Ответ

Заголовки

Кроме общих заголовков, ответ может содержать:

Заголовок	Описание
`x-amz-version-id`	Идентификатор версии объекта.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML.

Схема данных

```
<?xml version="1.0" encoding="UTF-8" ?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <TagSet>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </TagSet>
</Tagging>
```

Элемент	Описание
`Tagging`	Корневой элемент.
`TagSet`	Массив меток.
`Tag`	Контейнер для метки.
`Key`	Ключ метки. Тип: string.
`Value`	Значение метки. Тип: string.

Метод DeleteObjectTagging

Удаляет все метки объекта в бакете.

Запрос

```
DELETE /{bucket}/{key}?tagging&versionId={versionId} HTTP/2
```

Path-параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query-параметры

Параметр	Описание
`tagging`	Тип операции. Обязательный параметр.
`versionId`	Идентификатор версии объекта. Обязательный параметр.

Заголовки

Используйте в запросе общие заголовки.

Ответ

Заголовки

Кроме общих заголовков, ответ может содержать:

Заголовок	Описание
`x-amz-version-id`	Идентификатор версии объекта.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ не содержит дополнительных данных.

Общий порядок составной (multipart) загрузки

Составная загрузка позволяет сохранять объекты по частям. Это может пригодиться при загрузке или копировании больших объектов. Рекомендуем использовать составную загрузку для объектов от 100 МБ.

Составная загрузка состоит из следующих шагов:

1. Инициализация загрузки.
   Пользователь отправляет запрос на начало составной загрузки, а ПО Яндекс Объектное хранилище возвращает идентификатор, который следует использовать для всех последующих операций с загрузкой.
   Пользовательские метаданные объекта следует передавать на этом этапе загрузки.
2. Загрузка объекта по частям.
   Каждая часть объекта отправляется отдельным запросом и должна иметь порядковый номер, который используется для сборки объекта на стороне ПО Яндекс Объектное хранилище. Если ПО Яндекс Объектное хранилище получит две части объекта с одинаковыми номерами, то сохранит последнюю пришедшую.
   Для каждой загруженной части ПО Яндекс Объектное хранилище возвращает заголовок `ETag` в ответе. Пользователь должен сохранить номера и соответствующие им `ETag` для всех загруженных частей. Это необходимо для операции завершения загрузки.
   В процессе загрузки можно получить от ПО Яндекс Объектное хранилище список уже загруженных частей объекта.
3. Завершение загрузки.
   При получении запроса на завершение загрузки ПО Яндекс Объектное хранилище собирает все загруженные части в единый объект и присоединяет к объекту метаданные, которые были переданы при инициализации загрузки.

   > **Примечание**
   >
   > Пока загрузка не завершена, части объекта сохраняются по отдельности и занимают место, при этом их нельзя получить из ПО Яндекс Объектное хранилище. Незавершенные загрузки учитываются при расчете занятого места.

   Помимо запроса на завершение загрузки пользователь может отправить запрос о прерывании загрузки. В этом случае ПО Яндекс Объектное хранилище удалит все полученные части объекта для заданной загрузки и удалит саму загрузку.
   После завершения или прерывания загрузки пользователь не сможет более использовать идентификатор загрузки в запросах.

Пользователь может одновременно запустить несколько составных загрузок.

Методы составной загрузки:

Метод	Описание
CreateMultipartUpload	Инициализирует составную загрузку.

UploadPart	Загружает часть объекта.
UploadPartCopy	Копирует часть объекта.
ListParts	Выдает список загруженных частей.
AbortMultipartUpload	Прерывает составную загрузку.
CompleteMultipartUpload	Завершает составную загрузку.
ListMultipartUploads	Выдает список незавершенных загрузок.

Метод CreateMultipartUpload

Возвращает идентификатор, который следует использовать во всех дальнейших операциях по загрузке объекта.

Если вместе с объектом необходимо хранить пользовательские метаданные, то передавать их следует в этом запросе.

Запрос

```
POST /{bucket}/{key}?uploads HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`uploads`	Флаг, обозначающий операцию составной загрузки.

Заголовки

Используйте в запросе необходимые общие заголовки.

Дополнительно можно использовать заголовки, перечисленные в таблице ниже.

Заголовок	Описание

`X-Amz-Meta-*`	Пользовательские метаданные объекта. Все заголовки, начинающиеся с `X-Amz-Meta-`, ПО Яндекс Объектное хранилище преобразует по правилу: `X-Amz-Meta-foo-bar_baz` → `X-Amz-Meta-Foo-Bar_baz`. Общий размер пользовательских заголовков не должен превышать 2 КБ. Размер пользовательских данных определяется как длина строки в кодировке UTF-8. В размере учитываются и названия заголовков и их значения.
`X-Amz-Storage-Class`	Класс хранилища объекта. Может иметь любое из значений: • `STANDARD` — стандартное хранилище.
`X-Amz-Object-Lock-Mode`	Тип временной блокировки, устанавливаемой на объект (если бакет версионируемый и в нем включен механизм блокировок): • `GOVERNANCE` — временная управляемая блокировка. • `COMPLIANCE` — временная строгая блокировка. Вы можете установить на версию объекта только временную блокировку (заголовки `X-Amz-Object-Lock-Mode` и `X-Amz-Object-Lock-Retain-Until-Date`), только бессрочную блокировку (`X-Amz-Object-Lock-Legal-Hold`) или обе сразу.
`X-Amz-Object-Lock-Retain-Until-Date`	Дата и время окончания временной блокировки в любом из форматов, описанных в стандарте HTTP. Например, `Mon, 12 Dec 2022 09:00:00 GMT`. Указывается только вместе с заголовком `X-Amz-Object-Lock-Mode`.

Заголовок	Описание
`X-Amz-Acl`	Устанавливает предопределенный ACL для объекта.
`X-Amz-Grant-Read`	Устанавливает получателю доступа разрешение на чтение объекта.
`X-Amz-Grant-Read-Acp`	Устанавливает получателю доступа разрешение на чтение ACL объекта.
`X-Amz-Grant-Write-Acp`	Устанавливает получателю доступа разрешение на запись ACL объекта.
`X-Amz-Grant-Full-Control`	Устанавливает получателю доступа разрешения: `READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` на объект.

Значение для заголовков `X-Amz-Grant-*` представляет собой разделенный запятыми список получателей доступа. Каждый получатель доступа идентифицируется структурой вида `<тип_получателя_доступа>:<идентификатор_получателя_доступа>`. ПО Яндекс Объектное хранилище поддерживает следующие типы получателей:

- `id` — получатель доступа — учетная запись S3.
- `uri` — получатель доступа — публичная группа.

Пример:

```
X-Amz-Grant-Read: uri="http://acs.amazonaws.com/groups/s3/AuthenticatedUsers"
```

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<InitiateMultipartUploadResult>
  <Bucket>bucket-name</Bucket>
  <Key>object-key</Key>
  <UploadId>upload-id</UploadId>
</InitiateMultipartUploadResult>
```

Тег	Описание
`InitiateMultipartUploadResult`	Корневой тег ответа. Путь: `/InitiateMultipartUploadResult`.
`Bucket`	Имя бакета в который загружается объект. Путь: `/InitiateMultipartUploadResult/Bucket`.
`Key`	Ключ, который ассоциируется с объектом, после окончания загрузки. Путь: `/InitiateMultipartUploadResult/Key`.
`UploadId`	Идентификатор загрузки. Все последующие операции с загрузкой должны передавать в ПО Яндекс Объектное хранилище этот идентификатор. Путь: `/InitiateMultipartUploadResult/UploadId`.

Метод UploadPart

Сохраняет часть объекта.

Пользователь самостоятельно нумерует части объекта и передает номера в ПО Яндекс Объектное хранилище. Номер однозначно идентифицирует часть и определяет ее порядок в общей последовательности. Номер — это целое число в промежутке от 1 до 10000 включительно.

Если загружаются несколько частей с одинаковым номером, ПО Яндекс Объектное хранилище сохраняет последнюю поступившую.

Размер каждой части, кроме последней, должен быть не менее 5MB.

Запрос

```
PUT /{bucket}/{key}?partNumber=PartNumber&uploadId=UploadId HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`partNumber`	Идентификатор, который вы присвоили загружаемой части.
`uploadId`	Идентификатор составной загрузки, который ПО Яндекс Объектное хранилище вернуло при инициализации.

Заголовки

Используйте в запросе необходимые общие заголовки.

Заголовок `Content-Length` обязателен.

Заголовок `Content-MD5` обязателен, если в бакете настроены блокировки версий объектов по умолчанию.

Ответ

Заголовки

Ответ может содержать общие заголовки и заголовки, перечисленные в таблице ниже.

Заголовок	Описание

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Дополнительно, ПО Яндекс Объектное хранилище может вернуть ошибки, описанные в таблице ниже.

Ошибка	Описание	HTTP-код
`NoSuchUpload`	Указанная загрузка не существует. Возможно указан неверный идентификатор загрузки или загрузка была завершена или удалена.	404 Not Found
`EntityTooSmall`	Размер части слишком мал. Загружаемая часть должна быть не менее 5MB.	400 Bad Request

Метод UploadPartCopy

Копирует часть объекта.

Имеет такую же функциональность, как и метод `UploadPart`, только данные передаются не в теле запроса, а копируются из существующего объекта.

Запрос

```
PUT /{bucket}/{key}?partNumber=PartNumber&uploadId=UploadId HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя результирующего бакета.
`key`	Ключ результирующего объекта.

Query параметры

Параметр	Описание
`partNumber`	Идентификатор, который вы присвоили загружаемой части.
`uploadId`	Идентификатор составной загрузки, который ПО Яндекс Объектное хранилище вернуло при инициализации.

Заголовки

Используйте в запросе необходимые общие заголовки.

Заголовок `Content-Length` обязателен. Также обязательны заголовки, перечисленные в таблице ниже.

Заголовок	Описание
`X-Amz-Copy-Source`	Имя бакета и ключ объекта, данные которого будут копироваться, разделенные символом `/`. Например, `X-Amz-Copy-Source: /source_bucket/sourceObject`.

`X-Amz-Copy-Source-Range`	Диапазон байт для копирования из исходного объекта. Например, если указать `X-Amz-Copy-Source-Range:bytes=10-36`, то ПО Яндекс Объектное хранилище скопирует с 10-го по 36-й байт исходного объекта.

умолчанию.

Если вы хотите добавить условия на копирование, то используйте заголовки, перечисленные в таблице ниже.

Заголовки, описанные в таблице ниже используйте, если вам необходимо изменить поведение метода `copy` по умолчанию.

Заголовок	Описание
`X-Amz-Copy-Source-If-Match`	Условие для копирования объекта. Если `ETag` объекта равен заданному в заголовке, то объект копируется. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Unmodified-Since`.
`X-Amz-Copy-Source-If-None-Match`	Условие для копирования объекта. Если `ETag` объекта не равен заданному в заголовке, то объект копируется. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Modified-Since`.
`X-Amz-Copy-Source-If-Unmodified-Since`	Условие для копирования объекта. Объект копируется, если он не изменялся с указанного времени. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-Match`.

`X-Amz-Copy-Source-If-Modified-Since`	Условие для копирования объекта. Объект копируется, если он изменился с указанного времени. Если условие не выполнено, то ПО Яндекс Объектное хранилище вернет ошибку 412. Можно использовать вместе с заголовком `X-Amz-Copy-Source-If-None-Match`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Дополнительно, ПО Яндекс Объектное хранилище может вернуть ошибки, описанные в таблице ниже.

Ошибка	Описание	HTTP-код
`NoSuchUpload`	Указанная загрузка не существует. Возможно указан неверный идентификатор загрузки или загрузка была завершена или удалена.	404 Not Found
`EntityTooSmall`	Размер части слишком мал. Загружаемая часть должна быть не менее 5MB.	400 Bad Request

Схема данных

```
<CopyObjectResult>
   <LastModified>2019-02-15T14:32:00</LastModified>
   <ETag>"9bgh7535f2734ec974343yuc93985328"</ETag>
</CopyObjectResult>
```

Элемент	Описание
`CopyObjectResult`	Содержит элементы ответа. Путь: `/CopyObjectResult`.

`ETag`	`ETag` результирующей части составной загрузки. Путь: `/CopyObjectResult/ETag`.
`LastModified`	Дата последнего изменения части составной загрузки. Путь: `/CopyObjectResult/LastModified`.

Метод ListParts

Возвращает список уже загруженных частей для указанной составной загрузки.

Ответ не может содержать более 1000 элементов. Если в составной загрузке больше частей, то ПО Яндекс Объектное хранилище возвращает маркер `IsTruncated` и элемент `NextPartNumberMarker`. Оставшиеся элементы можно получить последовательными запросами, в которых параметр `part-number-marker` равен `NextPartNumberMarker` из предыдущего запроса.

Запрос

```
GET /{bucket}/{key}?uploadId=UploadId HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Изменить ответ от ПО Яндекс Объектное хранилище можно параметрами, описанными в таблице ниже.

Параметр	Описание
`encoding-type`	Кодировка ответа от сервера. ПО Яндекс Объектное хранилище по требованию клиента может закодировать ответ в требуемом виде.
`max-parts`	Максимальное количество элементов в ответе за один запрос. По умолчанию 1000.
`part-number-marker`	Номер части, с которого должен начинаться ответ. ПО Яндекс Объектное хранилище включит в ответ только части, номера которых больше указанного.
`uploadId`	Идентификатор составной загрузки.

Обязателен только параметр `uploadId`.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<ListPartsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>example-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>0005B465********</UploadId>
  <Initiator>
      <ID>ajeanexa********</ID>
      <DisplayName>ajeanexampleuser</DisplayName>
  </Initiator>
  <Owner>
      <ID>ajeanexa********</ID>
      <DisplayName>ajeanexampleuser</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
  <PartNumberMarker>0</PartNumberMarker>
  <NextPartNumberMarker>2</NextPartNumberMarker>
  <MaxParts>2</MaxParts>
  <IsTruncated>true</IsTruncated>
  <Part>
    <PartNumber>1</PartNumber>
    <LastModified>2020-11-18T17:33:46.787Z</LastModified>
    <ETag>"1430884b802ee5206cbe0ca0cf9b73d4"</ETag>
    <Size>8388608</Size>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2020-11-18T17:33:46.812Z</LastModified>
    <ETag>"449aa7eaff2e841923b0da9ff2be5946"</ETag>
    <Size>8388608</Size>
  </Part>
</ListPartsResult>
```

Возможные теги ответа описаны в таблице ниже.

Тег	Описание
`ListPartsResult`	Корневой элемент ответа. Путь: `/ListPartsResult`.
`Bucket`	Бакет, к которому относится составная загрузка. Путь: `/ListPartsResult/Bucket`.
`Encoding-Type`	Кодировка, в которой ПО Яндекс Объектное хранилище представляет ключ в XML-ответе. Появляется, если клиент при запросе передал параметр `encoding-type`. Путь: `/ListPartsResult/Encoding-Type`.
`Key`	Ключ, для которого производится составная загрузка. Путь: `/ListPartsResult/Key`.
`UploadId`	Идентификатор составной загрузки. Путь: `/ListPartsResult/UploadId`.
`Initiator`	Информация о пользователе, инициировавшем загрузку. Путь: `/ListPartsResult/Initiator`.
`ID`	Идентификатор пользователя. Путь: `/ListPartsResult/Initiator/ID`.
`DisplayName`	Отображаемое имя пользователя. Путь: `/ListPartsResult/Initiator/DisplayName`.
`Owner`	Информация о владельце объекта, совпадает с `Initiator`. Путь: `/ListPartsResult/Owner`.
`StorageClass`	Класс хранилища объекта: `STANDARD`. Путь: `/ListPartsResult/StorageClass`.

`PartNumberMarker`	Номер части, после которого начинается список. Первый элемент списка имеет номер, следующий за `PartNumberMarker`. Путь: `/ListPartsResult/PartNumberMarker`.
`NextPartNumberMarker`	Номер части, которым заканчивается текущий список. Присутствует в случае, когда в ответ не поместился весь перечень частей. Путь: `/ListPartsResult/NextPartNumberMarker`.
`MaxParts`	Максимальная длина списка для одного ответа. Путь: `/ListPartsResult/MaxParts`.
`IsTruncated`	Признак неполноты списка. Если `IsTruncated` — `true`, то это означает, что ПО Яндекс Объектное хранилище вернул не полный список частей. Путь: `/ListPartsResult/IsTruncated`.
`Part`	Описание части загрузки. Путь: `/ListPartsResult/Part`.
`PartNumber`	Номер части. Уникальный целочисленный идентификатор, определяющий положение части в загрузке. Путь: `/ListPartsResult/Part/PartNumber`.
`LastModified`	Дата и время загрузки части. Путь: `/ListPartsResult/Part/LastModified`.
`ETag`	`ETag` загруженной части. Путь: `/ListPartsResult/Part/ETag`.
`Size`	Размер загруженной части. Путь: `/ListPartsResult/Part/Size`.

Метод AbortMultipartUpload

Прерывает загрузку и удаляет все уже сохраненные части объекта. Если запрос на прерывание загрузки поступил в процессе загрузки какой-либо из частей, то результат не гарантирован.

Рекомендуем после прерывания загрузки получить список частей, и если он не пуст, то следует еще раз отправить запрос на прерывание загрузки. Запросы на прерывание следует отправлять до тех пор, пока список частей не станет пустым.

Запрос

```
DELETE /{bucket}/{key}?uploadId=UploadId HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`uploadId`	Идентификатор составной загрузки, который ПО Яндекс Объектное хранилище вернуло при инициализации.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Дополнительно, ПО Яндекс Объектное хранилище может вернуть ошибки, описанные в таблице ниже.

Ошибка	Описание	HTTP-код
`NoSuchUpload`	Указанная загрузка не существует. Возможно указан неверный идентификатор загрузки или загрузка была завершена или удалена.	404 Not Found

Метод CompleteMultipartUpload

Запрос завершает составную загрузку.

При получении запроса ПО Яндекс Объектное хранилище:

- Собирает конечный объект из полученных в процессе загрузки частей в порядке их номеров
- Удаляет идентификатор загрузки, так что все последующие запросы с идентификатором загрузки вернут ошибку `NoSuchUpload`.

При завершении загрузки клиент должен предоставить список частей, которые он отправлял. Описание каждой части должно содержать `ETag`, который клиент получает в ответ на каждую загруженную часть.

В зависимости от размера объекта и количества частей операция может занять несколько минут.

Если запрос завершился с ошибкой, то клиентское приложение должно быть готово повторить запрос.

Запрос

```
POST /{bucket}/{key}?uploadId=UploadId HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`uploadId`	Идентификатор составной загрузки, который ПО Яндекс Объектное хранилище вернуло при инициализации.

Заголовки

Используйте в запросе необходимые общие заголовки.

Схема данных

Список частей составной загрузки передается в виде XML-файла следующего формата:

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>PartNumber</PartNumber>
    <ETag>ETag</ETag>
  </Part>
  ...
</CompleteMultipartUpload>
```

Тег	Описание
`CompleteMultipartUpload`	Данные запроса. Путь: `/CompleteMultipartUpload`.
`Part`	Данные о загруженной части объекта. Путь: `/CompleteMultipartUpload/Part`.
`PartNumber`	Номер части. Уникальный идентификатор, определяющий положение части среди других частей в загрузке. Путь: `/CompleteMultipartUpload/Part/PartNumber`.
`ETag`	Идентификатор, который клиент получил от ПО Яндекс Объектное хранилище в ответ на загрузку части. Путь: `/CompleteMultipartUpload/Part/ETag`.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Дополнительно, ПО Яндекс Объектное хранилище может вернуть ошибки, описанные в таблице ниже.

Ошибка	Описание	HTTP-код

`NoSuchUpload`	Указанная загрузка не существует. Возможно указан неверный идентификатор загрузки или загрузка была завершена или удалена.	404 Not Found
`InvalidPart`	Некоторые из указанных частей не найдены. Возможные причины: - Части не загружены. - Переданный `ETag` не совпадает с сохраненным.	400 Bad Request
`InvalidPartOrder`	Список частей передан не в упорядоченном по возрастанию виде. Список должен быть отсортирован по возрастанию по номерам частей.	400 Bad Request

Схема данных

```
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Location>http://Example-Bucket.<s3_api_хост>/Example-Object</Location>
  <Bucket>Example-Bucket</Bucket>
  <Key>Example-Object</Key>
  <ETag>"3858f62230ac3c915f300c664312c11f-9"</ETag>
</CompleteMultipartUploadResult>
```

Тег	Описание
`CompleteMultipartUploadResult`	Данные ответа. Путь: `/CompleteMultipartUploadResult`.
`Location`	URI созданного в результате загрузки объекта. Путь: `/CompleteMultipartUploadResult/Location`.
`Bucket`	Имя бакета, в котором находится объект. Путь: `/CompleteMultipartUploadResult/Bucket`.
`Key`	Ключ созданного объекта. Путь: `/CompleteMultipartUploadResult/Key`.
`ETag`	Хэш объекта. ETag может быть, а может и не быть MD5. Путь: `/CompleteMultipartUploadResult/ETag`.

Метод ListMultipartUploads

Возвращает список текущих составных загрузок.

Ответ не может содержать более 1000 элементов. Если загрузок больше, то ПО Яндекс Объектное хранилище возвращает элемент `IsTruncated`, а также элементы `NextKeyMarker` и `NextUploadIdMarker`, которые необходимо использовать для параметров `key-marker` и `upload-id-marker` последующего запроса.

Запрос

```
GET /{bucket}?uploads HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`delimiter`	Символ-разделитель. Если параметр указан, то ПО Яндекс Объектное хранилище рассматривает ключ как путь к файлу, где имена каталогов разделяются символом `delimiter`. На выходе пользователь увидит перечень “файлов” и “каталогов” в “корне” бакета. “Файлы” будут выведены в элементах `Uploads`, а “каталоги” в элементах `CommonPrefixes`. Если в запросе указан еще и параметр `prefix`, то ПО Яндекс Объектное хранилище вернет перечень “файлов” и “каталогов” в “каталоге” `prefix`.

`max-uploads`	Максимальное количество загрузок в ответе. По умолчанию ПО Яндекс Объектное хранилище выдает не более 1000 ключей. Этот параметр следует использовать, если вам нужно получать менее 1000 ключей в одном ответе. Если под критерии отбора попадает больше ключей, чем поместилось в выдаче, то ответ содержит `<IsTruncated>true</IsTruncated>`. Чтобы получить все объекты выдачи, если их больше `max-keys`, необходимо выполнить несколько последовательных запросов к ПО Яндекс Объектное хранилище с параметром `key-marker`, где для каждого запроса `key-marker` равен значению элемента `NextKeyMarker` предыдущего ответа.
`key-marker`	Ключ. Выдача начнется с ключа, следующего за указанным в значении параметра. Используется вместе с `upload-id-marker` для фильтрации выдачи. Если `upload-id-marker` указан, то в выдачу попадет и `key-marker` тоже.
`prefix`	Строка, с которой должен начинаться ключ. ПО Яндекс Объектное хранилище выберет только те ключи, которые начинаются с `prefix`.
`upload-id-marker`	Идентификатор загрузки. Выдача начнется с той загрузки, идентификатор которой следует за указанной в параметре. При обработке учитывается `key-marker`, т.е. в выдачу попадут те загрузки для которых пересекутся фильтр по `upload-id-marker` и фильтр по `key-marker`. Если `key-marker` не указан, то `upload-id-marker` игнорируется.
`uploads`	Флаг, обозначающий операцию составной загрузки.

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Успешный ответ содержит дополнительные данные в формате XML, схема которого описана ниже.

Схема данных

```
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>bucket</Bucket>
  <KeyMarker></KeyMarker>
  <UploadIdMarker></UploadIdMarker>
  <NextKeyMarker>my-movie.m2ts</NextKeyMarker>
  <NextUploadIdMarker>0005B466********</NextUploadIdMarker>
  <MaxUploads>3</MaxUploads>
  <IsTruncated>true</IsTruncated>
  <Upload>
    <Key>my-divisor</Key>
    <UploadId>0005B465********</UploadId>
    <Initiator>
      <ID>ajeanexa********</ID>
      <DisplayName>ajeanexa********</DisplayName>
    </Initiator>
    <Owner>
      <ID>aje2v5og9qpl********</ID>
      <DisplayName>aje2v5og9qplr6pe0c59</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2020-11-18T17:33:46.007Z</Initiated>
  </Upload>
  <Upload>
    <Key>my-movie.m2ts</Key>
    <UploadId>0005B465********</UploadId>
    <Initiator>
      <ID>ajeanexa********</ID>
      <DisplayName>ajeanexampleuser</DisplayName>
    </Initiator>
    <Owner>
      <ID>aje2v5og9qpl********</ID>
      <DisplayName>aje2v5og9qplr6pe0c59</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2020-11-18T18:34:47.017Z</Initiated>
  </Upload>
  <Upload>
    <Key>my-movie.m2ts</Key>
    <UploadId>0005B466********</UploadId>
    <Initiator>
      <ID>ajeanexa********</ID>
      <DisplayName>ajeanexa********</DisplayName>
    </Initiator>
    <Owner>
      <ID>aje2v5og9qpl********</ID>
      <DisplayName>aje2v5og9qplr6pe0c59</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2020-11-18T18:35:41.231Z</Initiated>
```

```
  </Upload>
</ListMultipartUploadsResult>
```

Тег	Описание
`ListMultipartUploadsResult`	Корневой тег ответа. Путь: `/ListMultipartUploadsResult`.
`Bucket`	Бакет, к которому относится составная загрузка. Путь: `/ListMultipartUploadsResult/Bucket`.
`KeyMarker`	Ключ. Выдача начинается с ключа, следующего за указанным в значении элемента. Смотрите описание параметра запроса `key-marker`. Путь: `/ListMultipartUploadsResult/KeyMarker`.
`UploadIdMarker`	Идентификатор загрузки. Выдача начнется с той загрузки, идентификатор которой следует за указанной в параметре. Путь: `/ListMultipartUploadsResult/UploadIdMarker`.
`NextKeyMarker`	Ключ. Если выдача не вместила все элементы, которые должен получить пользователь, то это значение необходимо использовать в параметре `key-marker` для последующих запросов. Присутствует в случае, когда в ответ не поместились все элементы. Путь: `/ListMultipartUploadsResult/NextKeyMarker`.

`NextUploadIdMarker`	Идентификатор загрузки. Если выдача не вместила все элементы, которые должен получить пользователь, то это значение необходимо использовать в параметре `upload-id-marker` для последующих запросов. Присутствует в случае, когда в ответ не поместились все элементы. Путь: `/ListMultipartUploadsResult/NextUploadMarker`.
`Encoding-Type`	Кодировка, в которой ПО Яндекс Объектное хранилище представляет ключ в XML-ответе. Смотрите описание параметра запроса `encoding-type`. Путь: `/ListMultipartUploadsResult/Encoding-Type`.
`MaxUploads`	Максимальная длина списка для одного ответа. Смотрите параметр запроса `max-uploads`. Путь: `/ListMultipartUploadsResult/MaxUploads`.
`IsTruncated`	Признак неполноты списка. Если `IsTruncated` — `true`, то это означает, что ПО Яндекс Объектное хранилище вернуло не полный список загрузок. Путь: `/ListMultipartUploadsResult/IsTruncated`.
`Upload`	Описание загрузки. Путь: `/ListMultipartUploadsResult/Upload`.
`Key`	Ключ конечного объекта загрузки. Путь: `/ListMultipartUploadsResult/Upload/Key`.
`UploadId`	Идентификатор составной загрузки. Путь: `/ListMultipartUploadsResult/Upload/UploadId`.
`Initiator`	Инициатор составной загрузки. Путь: `/ListMultipartUploadsResult/Upload/Initiator`.

`ID`	Идентификатор пользователя. Возможные пути: - `/ListMultipartUploadsResult/Upload/Initiator/ID`
`DisplayName`	Отображаемое имя пользователя. Возможные пути: - `/ListMultipartUploadsResult/Upload/Initiator/DisplayName`
`Owner`	Информация о владельце объекта, совпадает с `Initiator`. Путь: `/ListMultipartUploadsResult/Owner`.
`StorageClass`	Класс хранилища объекта: `STANDARD`. Путь: `/ListMultipartUploadsResult/Upload/StorageClass`.
`Initiated`	Дата и время запроса на начало составной загрузки.
`/ListMultipartUploadsResult/Prefix`	Префикс ключа. Смотрите параметр запроса `prefix`. Путь: `/ListMultipartUploadsResult/Prefix`.
`Delimiter`	Символ-разделитель, который использовался при формировании выдачи. Путь: `/ListMultipartUploadsResult/Delimiter`.
`CommonPrefixes`	Содержит элемент `Prefix`. Путь: `/ListMultipartUploadsResult/CommonPrefixes`.
`CommonPrefixes/Prefix`	Часть имени ключа, которая определяется при обработке параметров запроса `delimiter` и `prefix`. Путь: `/ListMultipartUploadsResult/CommonPrefixes/Prefix`.

Все методы CORS

Метод	Описание
PutBucketCors	Загружает конфигурацию бакета для CORS.
GetBucketCors	Возвращает конфигурацию бакета для CORS.
DeleteBucketCors	Удаляет конфигурацию бакета для CORS.

CORS-конфигурация бакетов

ПО Яндекс Объектное хранилище позволяет управлять конфигурацией CORS бакета. Для загрузки конфигурации CORS необходимо сформировать XML-документ, описанный в этом разделе. Скачивая существующую конфигурацию, вы получаете документ такого же формата.

Общий вид конфигурации:

```
<CORSConfiguration>
    <CORSRule>
        <AllowedOrigin>URL</AllowedOrigin>
        <AllowedMethod>HTTP_Method</AllowedMethod>
        <AllowedHeader>Header_Name</AllowedHeader>
        ...
    </CORSRule>
    ...
</CORSConfiguration>
```

Конфигурация может содержать не более 100 правил.

Элементы

Элемент	Описание
`CORSConfiguration`	Корневой элемент конфигурации CORS. Не может содержать более 100 элементов `CORSRule` Путь: `/CORSConfiguration`.
`CORSRule`	Правило для фильтрации входящих запросов к ресурсу. Каждое правило должно содержать хотя бы по одному элементу `AllowedMethod` и `AllowedOrigin`. Путь: `/CORSConfiguration/CORSRule`.
`ID`	Уникальный идентификатор правила (не более 255 символов). Необязательный. Можно использовать для поиска правила в файле. Путь: `/CORSConfiguration/CORSRule/ID`.
`AllowedMethod`	HTTP метод (`PUT`, `GET`, `HEAD`, `POST`, `DELETE`), разрешенный для использования при кросс-доменном запросе. Каждый метод следует указать в отдельном элементе. Обязательно указать хотя бы один метод. Путь: `/CORSConfiguration/CORSRule/AllowedMethod`.

`AllowedOrigin`	Сайт, с которого разрешены кросс-доменные запросы к бакету. Должен быть указан хотя бы один элемент `AllowedOrigin`. Может содержать не более одного символа `*`. Примеры: `http://*.example.com`, `*`. Путь: `/CORSConfiguration/CORSRule/AllowedOrigin`.
`AllowedHeader`	Разрешенный заголовок в запросе к объекту. Если разрешенных заголовков несколько, то каждый следует указать в отдельном `AllowedHeader`. В имени заголовка можно использовать один символ `*` для определения шаблона, например `<AllowedHeader>*</AllowedHeader>` означает, что разрешены все заголовки. Запрос методом options содержит заголовок `Access-Control-Request-Headers`. ПО Яндекс Объектное хранилище сопоставляет заголовки, переданные в `Access-Control-Request-Headers`, с набором `AllowedHeader` и отвечает на `options` списком разрешенных. Путь: `/CORSConfiguration/CORSRule/AllowedHeader`.
`MaxAgeSeconds`	Время в секундах, в течение которого браузер сохраняет в кэше результат запроса к объекту методом options. Путь: `/CORSConfiguration/CORSRule/MaxAgeSeconds`.
`ExposeHeader`	Заголовок, разрешенный к показу в JavaScript-приложении в браузере. Если допустимы несколько заголовков, то укажите каждый из них в отдельном элементе. В запросе к объекту JavaScript-клиент может оперировать только заголовками, определенными в элементах `ExposeHeader`. Путь: `/CORSConfiguration/CORSRule/ExposeHeader`.

Пример

Приведенная ниже конфигурация позволяет отправлять кросс-доменные запросы с сайта `http://www.example.com` методами `PUT`, `POST`, `DELETE` без ограничений по заголовкам.

```
<CORSConfiguration>
    <CORSRule>
        <AllowedOrigin>http://www.example.com</AllowedOrigin>
        <AllowedMethod>PUT</AllowedMethod>
        <AllowedMethod>POST</AllowedMethod>
        <AllowedMethod>DELETE</AllowedMethod>
        <AllowedHeader>*</AllowedHeader>
```

```
    </CORSRule>
</CORSConfiguration>
```

Метод PutBucketCors

Загружает конфигурацию CORS для бакета. Если конфигурация существует, то заменяет ее.

Конфигурация CORS — это XML-файл размером не более 64КБ. Конфигурация не может содержать более 100 правил.

Запрос

```
PUT /{bucket}?cors HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`cors`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Заголовок `Content-MD5` обязателен.

Схема данных

Конфигурация CORS передается в виде XML-документа. Описание схемы смотрите в соответствующем разделе

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetBucketCors

Возвращает конфигурацию CORS для бакета.

Запрос

```
GET /{bucket}?cors HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`cors`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Схема данных

Возвращаемые данные имеют ту же структуру, которую имеют данные, передаваемые методом upload.

Метод DeleteBucketCors

Удаляет конфигурацию CORS для бакета.

Запрос

```
DELETE /{bucket}?cors HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`cors`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Если конфигурация CORS не существует, то ПО Яндекс Объектное хранилище ответит кодом `204 No Content`.

Все методы

Метод	Описание
PutBucketLifecycle	Загружает конфигурацию жизненного цикла объектов.
GetBucketLifecycle	Возвращает конфигурацию жизненного цикла объектов.
DeleteBucketLifecycle	Удаляет конфигурацию жизненного цикла объектов.

Конфигурация жизненных циклов объектов в бакете

ПО Яндекс Объектное хранилище позволяет управлять жизненными циклами объектов в бакете. Для загрузки конфигурации жизненных циклов в ПО Яндекс Объектное хранилище необходимо сформировать XML-документ, описанный в этом разделе. При скачивании существующей конфигурации, вы получите документ такого же формата.

Общий вид конфигурации:

```
<LifecycleConfiguration>
    <Rule>
        <ID>Описание правила</ID>
        <Status>{Enabled|Disabled}</Status>
        <Filter>
            <And>
                <ObjectSizeGreaterThan>минимальный размер
объекта</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>максимальный размер объекта</ObjectSizeLessThan>
                <Prefix>префикс ключа</Prefix>
                <Tag>
                    <Key>ключ метки</Key>
                    <Value>значение метки</Value>
                </Tag>
                ...
            </And>
            <ObjectSizeGreaterThan>минимальный размер объекта</ObjectSizeGreaterThan>
            <ObjectSizeLessThan>максимальный размер объекта</ObjectSizeLessThan>
            <Prefix>префикс ключа</Prefix>
            <Tag>
                <Key>ключ метки</Key>
                <Value>значение метки</Value>
            </Tag>
        </Filter>

        <Transition>
            <StorageClass>Идентификатор класса хранилища</StorageClass>
            <!-- <Date> или <Days> -->
        </Transition>

        <Expiration>
            <!-- <Date> или <Days> -->
            <!-- <ExpiredObjectDeleteMarker> -->
        </Expiration>

        <NoncurrentVersionTransition>
            <StorageClass>Идентификатор класса хранилища</StorageClass>
            <NoncurrentDays>Перенос версий, которые старше указанного количества
дней</NoncurrentDays>
        </NoncurrentVersionTransition>
```

```
        <NoncurrentVersionExpiration>
            <NoncurrentDays>Удаление версий, которые старше указанного количества
дней</NoncurrentDays>
        </NoncurrentVersionExpiration>

        <AbortIncompleteMultipartUpload>
            <DaysAfterInitiation>Удаление загрузок, не завершенных за указанное
количество дней</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
        ...
    </Rule>
    <Rule>
      ...
    </Rule>
    ...
</LifecycleConfiguration>
```

Конфигурация может содержать до 1000 правил.

Элементы

Элемент	Описание
`LifecycleConfiguration`	Корневой элемент XML-документа. Может содержать до 1000 элементов `Rule`. Путь: `LifecycleConfiguration`.
`Rule`	Описание правила. Объекты, попадающие под действие правила задаются элементом `Filter`. Действия над объектами определяются элементами `Transition` и `Expiration`. Действий каждого типа может быть несколько. Путь: `LifecycleConfiguration\Rule`.
`ID`	Уникальный идентификатор правила. Произвольный текст длиной до 255 символов, например "Удалить через 20 дней". Необязательный параметр, который можно использовать для поиска правила в конфигурации. Если идентификатор не указан, то ПО Яндекс Объектное хранилище генерирует его автоматически. Путь: `LifecycleConfiguration\Rule\ID`.

`Status`	Статус правила. Правило можно активировать, установив `<Status>Enabled</Status>`, или отключить, установив `<Status>Disabled</Status>`. Путь: `LifecycleConfiguration\Rule\Status`.
`Filter`	Фильтр объектов. Содержит не более одного элемента каждого типа: `And`, `Prefix`, `ObjectSizeGreaterThan`, `ObjectSizeLessThan`, `Tag`. Если установить пустой фильтр `<Filter></Filter>`, то правило применяется ко всем объектам в бакете. Путь: `LifecycleConfiguration\Rule\Filter`.
`ObjectSizeGreaterThan`	Минимальный размер объекта в байтах. Под действие правила попадают объекты, размер которых больше или равен указанному. Фильтр может содержать только один минимальный размер объекта. Путь: `LifecycleConfiguration\Rule\Filter\ObjectSizeGreaterThan`.
`ObjectSizeLessThan`	Максимальный размер объекта в байтах. Под действие правила попадают объекты, размер которых меньше или равен указанному. Фильтр может содержать только один максимальный размер объекта. Путь: `LifecycleConfiguration\Rule\Filter\ObjectSizeLessThan`.

`Prefix`	Префикс ключа. Под действие правила попадают объекты с указанным префиксом ключа. Примеры префиксов для ключа `some/long/object/key`: `some`, `some/`, `some/lo`. Фильтр может содержать только один префикс. Путь: `LifecycleConfiguration\Rule\Filter\Prefix`.
`Tag`	Метка объекта Под действие правила попадают объекты, которым присвоена указанная метка. Фильтр может содержать только одну метку объекта. Путь: `LifecycleConfiguration\Rule\Filter\Tag`.
`And`	Логический оператор **И** (`AND`) для фильтров. Может содержать любое сочетание следующих элементов: `Prefix`, `ObjectSizeGreaterThan`, `ObjectSizeLessThan`, `Tag`. Путь: `LifecycleConfiguration\Rule\Filter\And`.
`Key`	Ключ метки объекта Путь: `LifecycleConfiguration\Rule\Filter\Tag\Key`.
`Value`	Значение метки объекта Путь: `LifecycleConfiguration\Rule\Filter\Tag\Value`.
`Expiration`	Правило для удаления объекта из ПО Яндекс Объектное хранилище. Содержит элемент `Days` или `Date`, который определяет сроки исполнения действия. Дополнительно может содержать `ExpiredObjectDeleteMarker` — маркер удаления объекта с истекшим сроком действия, который указывает, удалит ли ПО Яндекс Объектное хранилище маркер удаления при отсутствии неактивных версий. Путь: `LifecycleConfiguration\Rule\Expiration`.

`Date`	Дата исполнения правила. Формат — ISO 8601, например, `YYYY-MM-DD`. Время — всегда 00:00 UTC. Путь: `LifecycleConfiguration\Rule\Expiration\Date`.
`Days`	Интервал исполнения правила. Задается количеством дней после загрузки объекта. Минимальное значение — 1. Путь: `LifecycleConfiguration\Rule\Expiration\Days`.
`NoncurrentVersionExpiration`	Правило для удаления неактивных версий объекта из ПО Яндекс Объектное хранилище. Это правило применяется не ко всему объекту, а только к его неактивным версиям. Содержит элемент `NoncurrentDays`, который определяет сроки исполнения действия. Путь: `LifecycleConfiguration\Rule\NoncurrentVersionExpiration`.
`AbortIncompleteMultipartUpload`	Правило для удаления загрузок, не завершенных за указанное количество дней. Содержит элемент `DaysAfterInitiation`, который определяет срок исполнения правила. Путь: `LifecycleConfiguration\Rule\AbortIncompleteMultipartUpload\DaysAfterInitiation`.

Метод PutBucketLifecycle

Загружает конфигурацию жизненных циклов объектов в бакете.

Запрос

```
PUT /{bucket}?lifecycle HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`lifecycle`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Заголовок `Content-MD5` обязателен.

Схема данных

Вид конфигурации описан в соответствующем разделе.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetBucketLifecycle

Возвращает конфигурацию жизненных циклов объектов в бакете.

Запрос

```
GET /{bucket}?lifecycle HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`lifecycle`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Если конфигурации не существует, то ПО Яндекс Объектное хранилище возвращает ошибку 404 с кодом `NoSuchLifecycleConfiguration`.

Перечень других возможных ответов смотрите в соответствующем разделе.

Схема данных

Возвращаемые данные имеют ту же структуру, которую имеют данные, передаваемые методом `PutBucketLifecycle`. Структура описана в соответствующем разделе.

Метод DeleteBucketLifecycle

Удаляет конфигурацию жизненного цикла объектов в бакете.

Запрос

```
DELETE /{bucket}?lifecycle HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`lifecycle`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе только общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Если конфигурация жизненных циклов бакета не существует, то ПО Яндекс Объектное хранилище ответит кодом 200.

Все методы ACL

Метод	Описание
GetObjectAcl	Возвращает список управления доступом для объекта.
PutObjectAcl	Загружает список управления доступом для объекта.
GetBucketAcl	Возвращает список управления доступом для бакета.
PutBucketAcl	Загружает список управления доступом для бакета.

XML-структура конфигурации ACL

Общий вид ACL:

```
<AccessControlPolicy>
  <Owner>
    <ID>8caede4d8w78r43d14f2e7fagrbf45c78ejc7c6cde********</ID>
    <DisplayName>CustomersName@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xsi:type="CanonicalUser">
        <ID>8caede4d8w78r43d14f2e7fagrbf45c78ejc7c6cde********</ID>
        <DisplayName>UserName</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Элементы

Элемент	Описание
`AccessControlPolicy`	Корневой элемент. Путь: `/AccessControlPolicy`.
`Owner`	Информация о пользователе. Пользователь может указать этот элемент для запросов методами `objectPutAcl` и `bucketPutAcl`. Если элемент указан, то при загрузке ACL ПО Яндекс Объектное хранилище проверяет соответствие переданного идентификатора фактическому и если они не совпадают, то отвечает с кодом 403. Путь: `/AccessControlPolicy/Owner`.
`AccessControlList`	Список управления доступом. Не может содержать более 100 доступов. Путь: `/AccessControlPolicy/AccessControlList`.
`Grant`	Описание доступа. Путь: `/AccessControlPolicy/AccessControlList/Grant`.

`Grantee`	Тип получателя разрешений. Возможные значения для `type`: • `CanonicalUser` — для учетной записи S3. • `Group` — для публичной группы. Путь: `/AccessControlPolicy/AccessControlList/Grant/Grantee`.
`ID`	Идентификатор учетной записи S3. Используется с типом получателя разрешений `CanonicalUser`. Ответ на запрос `bucketGetAcl` содержит идентификатор тенанта, в котором находится бакет. Пути: `/AccessControlPolicy/Owner/ID`, `/AccessControlPolicy/AccessControlList/Grant/Grantee/ID`.
`DisplayName`	Имя пользователя. Игнорируется для запросов `objectPutAcl` и `bucketPutAcl` Пути: `/AccessControlPolicy/Owner/DisplayName`, `/AccessControlPolicy/AccessControlList/Grant/Grantee/DisplayName`.
`URI`	Идентификатор публичной группы. Используется с типом получателя разрешений `Group`. Возможные значения: • `http://acs.amazonaws.com/groups/global/AllUsers` — все пользователи ПО Яндекс Объектное хранилище. • `http://acs.amazonaws.com/groups/global/AuthenticatedUsers` — все аутентифицированные пользователи ПО Яндекс Объектное хранилище. Путь: `/AccessControlPolicy/AccessControlList/Grant/Grantee/URI`.
`Permission`	Разрешения пользователя. Можно указать: `READ`, `WRITE` и `FULL_CONTROL`, при выдаче разрешений для объекта можно также указать `READ_ACP`, `WRITE_ACP`. Путь: `/AccessControlPolicy/AccessControlList/Grant/Grantee/DisplayName`.

Метод GetObjectAcl

Возвращает список управления доступом для объекта.

Запрос

```
GET /{bucket}/{key}?acl HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`acl`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Схема данных

Возвращаемые данные имеют структуру, описанную в разделе XML-схема ACL.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод PutObjectAcl

Загружает список управления доступом для объекта.

> **Примечание**
>
> ACL можно загрузить в виде XML-документа или с помощью специальных заголовков `X-Amz-Grant*`. Не используйте XML-документ и заголовки `X-Amz-Grant*` в одном запросе.

Запрос

```
PUT /{bucket}/{key}?acl HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.
`key`	Ключ объекта.

Query параметры

Параметр	Описание
`acl`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также можно использовать заголовки, перечисленные ниже.

Заголовок	Описание
`X-Amz-Acl`	Устанавливает предопределенный ACL для объекта.
`X-Amz-Grant-Read`	Устанавливает получателю доступа разрешение на чтение объекта.

`X-Amz-Grant-Read-Acp`	Устанавливает получателю доступа разрешение на чтение ACL объекта.
`X-Amz-Grant-Write-Acp`	Устанавливает получателю доступа разрешение на запись ACL объекта.
`X-Amz-Grant-Full-Control`	Устанавливает получателю доступа разрешения: `READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` на объект.

Значение для заголовков `X-Amz-Grant-` представляет собой разделенный запятыми список получателей доступа. Каждый получатель доступа идентифицируется структурой вида `<тип_получателя_доступа>:<идентификатор_получателя_доступа>`. ПО Яндекс Объектное хранилище поддерживает следующие типы получателей:

- `id` — получатель доступа — учетная запись S3.
- `uri` — получатель доступа — публичная группа.

Пример:

```
X-Amz-Grant-Read: uri="http://acs.amazonaws.com/groups/s3/AuthenticatedUsers"
```

Схема данных

ACL передается в виде XML-документа. Описание схемы смотрите в разделе XML-схема ACL.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод GetBucketAcl

Возвращает список управления доступом для бакета.

Запрос

```
GET /{bucket}?acl HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`acl`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Схема данных

Возвращаемые данные имеют структуру, описанную в разделе XML-схема ACL.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод PutBucketAcl

Загружает список управления доступом для бакета.

> **Примечание**
>
> ACL можно загрузить в виде XML-документа или с помощью специальных заголовков `X-Amz-Grant*`. Не используйте XML-документ и заголовки `X-Amz-Grant*` в одном запросе.

Запрос

```
PUT /{bucket}?acl HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`acl`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Также можно использовать заголовки, перечисленные ниже.

Заголовок	Описание
`X-Amz-Acl`	Устанавливает виды разрешений для бакета.
`X-Amz-Grant-Read`	Устанавливает получателю доступа разрешение на просмотр содержимого бакета и разрешение на чтение объектов в бакете.

`X-Amz-Grant-Write`	Устанавливает получателю доступа разрешение на запись объектов. Используйте обязательно вместе с `X-Amz-Grant-Read`, иначе ПО Яндекс Объектное хранилище ответит с кодом `501 Not Implemented`.
`X-Amz-Grant-Read-Acp`	Устанавливает получателю доступа разрешение на чтение ACL бакета.
`X-Amz-Grant-Write-Acp`	Устанавливает получателю доступа разрешение на запись ACL бакета.
`X-Amz-Grant-Full-Control`	Устанавливает получателю доступа разрешения: `READ`, `WRITE`, `READ_ACP`, `WRITE_ACP` на бакет.

разделенным запятыми. Каждый получатель доступа идентифицируется структурой вида `<тип_получателя_доступа>:<идентификатор_получателя_доступа>`. ПО Яндекс Объектное хранилище поддерживает следующие типы получателей:

- `id` — получатель доступа — учетная запись S3.
- `uri` — получатель доступа — публичная группа.

Пример:

```
X-Amz-Grant-Read: uri="http://acs.amazonaws.com/groups/s3/AuthenticatedUsers"
```

Схема данных

ACL передается в виде XML-документа. Описание схемы смотрите в разделе XML-схема ACL.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Все методы политик доступа

Метод	Описание
GetBucketPolicy	Возвращает политику доступа для заданного бакета.
PutBucketPolicy	Применяет политику доступа к заданному бакету.
DeleteBucketPolicy	Удаляет политику доступа заданного бакета.

Метод GetBucketPolicy

Возвращает политику доступа для заданного бакета.

Запрос

```
GET /{bucket}?policy HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`policy`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Схема данных

Данные передаются в формате JSON, подробнее читайте в соответствующем разделе.

Коды ответов

Перечень возможных ответов смотрите в разделе соответствующем разделе.

Метод PutBucketPolicy

Применяет политику доступа к заданному бакету.

Запрос

```
PUT /{bucket}?policy HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`policy`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Схема данных

Данные передаются в формате JSON, подробнее читайте в разделе соответствующем разделе.

Ответ

Заголовки

Ответ может содержать только общие заголовки.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Метод DeleteBucketPolicy

Удаляет политику доступа заданного бакета.

Запрос

```
DELETE /{bucket}?policy HTTP/2
```

Path параметры

Параметр	Описание
`bucket`	Имя бакета.

Query параметры

Параметр	Описание
`policy`	Обязательный параметр для обозначения типа операции.

Заголовки

Используйте в запросе необходимые общие заголовки.

Ответ

При успешном выполнении запроса ПО Яндекс Объектное хранилище возвращает ответ с кодом `HTTP 204` и пустым телом.

Коды ответов

Перечень возможных ответов смотрите в соответствующем разделе.

Схема данных

Схема данных в формате JSON:

```
{
  "Version" : "string",
  "Id" : "string",
  "Statement" : [
    {
      "Sid" : "string",
      ("Principal" | "NotPrincipal") : ("*" | "CanonicalUser" : [
        "string",
        ...
      ]),
      "Effect" : ("Allow" | "Deny"),
      "Action" : ("*" | [
        "string",
        ...
      ]),
      "Resource" : ("*" | [
        "string",
        ...
      ]),
      "Condition" : {
        "condition_type_string" : {
          "condition_key_string" : [[("string" | "number" | "Boolean"),...]...]
        },
        ...
      }
    },
    ...
  ]
}
```

Схема может включать до 10 240 символов.

Описание параметров схемы:

Параметр	Описание
`Version`	(Опционально) **string** Версия описания политик доступа. Примеры значений: `2012-10-17`.

`Id`	(Опционально) **string** Общая информация о политике. Параметр задается пользователем. Примеры значений: `test-policy`, `Anonymous access policy`, `hrtk43sau2s8gqkaje06`.
`Statement[].`	**array** Правила политики доступа. Если к бакету применена политика доступа без правил, то доступ будет запрещен всем пользователям. Чтобы отключить проверки запросов по политике доступа, удалите ее.
`Statement[].Sid`	**string** (Опционально) Идентификатор правила. Параметр задается пользователем. Примеры значений: `test-rule`, `Statement Allow`, `Statement Deny`.
`Statement[].Principal`	**string** (Опционально) Идентификатор субъекта запрошенного разрешения. Получателем может быть учетная запись S3. Возможные значения: `*`, `<идентификатор_субъекта>`.
`Statement[].NotPrincipal`	**string** (Опционально) Идентификатор субъекта, который не получит запрошенного разрешения. В качестве субъекта может выступать учетная запись S3. Возможные значения: `<идентификатор_субъекта>`.
`Statement[].Effect`	**string** Определяет запрет или разрешение запрошенного действия. Возможные значения: `Allow`, `Deny`.
`Statement[].Action`	**string** Определяет действие, которое выполнится при срабатывании политики. Возможные значения: `s3:GetObject`, `s3:PutObject`.

`Statement[].Resource`	**string** Определяет ресурс, в отношении которого будет произведено действие. Возможные значения: • `arn:aws:s3:::<имя_бакета>` — бакет. • `arn:aws:s3:::<имя_бакета>/<ключ_объекта>` — объект в бакете. • `arn:aws:s3:::<имя_бакета>/<префикс_ключей_объектов>*` — все объекты в бакете, ключи которых начинаются с префикса, например `arn:aws:s3:::samplebucket/some/path/*`. Префикс может быть пустым: `arn:aws:s3:::samplebucket/*` — тогда правило будет относиться ко всем объектам в бакете. Ресурс бакета не включает в себя ресурсы всех его объектов. Чтобы правило в политике доступа относилось к бакету и всем объектам, их нужно указать как отдельные ресурсы, например `arn:aws:s3:::samplebucket` и `arn:aws:s3:::samplebucket/*`.
`Statement[].Condition{}.`	**string** (Опционально) Условие, которое будет проверяться. Если для правила задано одновременно несколько условий, то они применяются с логикой `И`, то есть для выполнения правила оно должно будет соответствовать одновременно всем заданным условиям.
`Statement[].Condition{}.` `condition_type_string{}.`	**string** Тип условия. Возможные значения: `StringEquals`, `Bool`.

`Statement[].Condition{}.`
`condition_type_string{}.`
`condition_key_string`
	string
	Ключ условия.
	
	Определяет условие, значение которого будет проверяться.
	Возможные значения: `aws:PrincipalType`, `true`.
	
	Если для одного условия задано сразу несколько ключей, эти ключи будут проверяться с логикой `И`, то есть для выполнения правила оно должно будет соответствовать одновременно всем указанным признакам.
	
	Если для одного ключа условия задано одновременно несколько значений, эти значения будут проверяться с логикой `ИЛИ`, то есть для выполнения правила ключ условия может соответствовать любому из заданных значений.

Действия

Действие	Описание
`s3:AbortMultipartUpload`	Дает право на прерывание составной загрузки.
`s3:DeleteObject`	Дает право на удаление null-версии и добавляет delete-маркер на последнюю версию объекта.
`s3:DeleteObjectVersion`	Дает право на удаление указанной версии объекта.
`s3:GetBucketAcl`	Дает право на чтение ACL бакета.
`s3:GetBucketCORS`	Дает право на чтение информации о конфигурации CORS.
`s3:GetBucketLocation`	Дает право на чтение информации о зоне, в которой находится бакет.
`s3:GetBucketVersioning`	Дает право на чтение информации о состоянии управления версиями бакета.
`s3:GetLifecycleConfiguration`	Дает право на чтение конфигурации жизненного цикла бакета.
`s3:GetObject`	Дает право на чтение объектов.
`s3:GetObjectAcl`	Дает право на чтение ACL объекта.
`s3:GetObjectVersion`	Дает право на чтение определенной версии объекта.
`s3:GetObjectVersionAcl`	Дает право на чтение ACL определенной версии объекта.
`s3:ListBucket`	Дает право на чтение списка некоторых или всех объектов в бакете.
`s3:ListBucketMultipartUploads`	Дает право на чтение списка незавершенных составных загрузок.
`s3:ListBucketVersions`	Дает право на чтение метаданных обо всех версиях объектов в бакете.
`s3:ListMultipartUploadParts`	Дает право на чтение списка загруженных частей определенной составной загрузки.
`s3:PutBucketAcl`	Дает право на установку разрешений ACL на бакет.

`s3:PutBucketCORS`	Дает право на установку конфигурации CORS бакета.
`s3:PutBucketVersioning`	Дает право на установку состояния управления версиями бакета.
`s3:PutLifecycleConfiguration`	Дает право на установку конфигурации жизненного цикла для бакета или замены существующей.
`s3:PutObject`	Дает право на добавление объекта в бакет.
`s3:PutObjectAcl`	Дает право на установку ACL для указанного объекта.
`s3:PutObjectVersionAcl`	Дает право на установку ACL для указанной версии объекта.

Условия

Условия определяют случаи, в которых действует правило.

Ключ условия	Описание
`aws:principaltype`	Задает тип сущности, к которой делается запрос.
`aws:referer`	Сравнивает Referer запроса с заданным в политике.
`aws:securetransport`	Определяет, был ли запрос отравлен с использованием SSL-шифрования.
`aws:sourceip`	Сравнивает IP-адрес, с которого пришел запрос, а также IP-адреса обратных прокси-серверов, например переданных в заголовке X-Forwarded-For, с заданными в политике. Если хотя бы один из IP-адресов совпадает с заданными в политике, условие считается выполненным.
`aws:useragent`	Сравнивает UserAgent запроса с заданными в политике.
`aws:userid`	Сравнивает идентификатор учетной записи S3 с заданным в политике.
`s3:authtype`	Ограничивает входящие запросы заданным методом аутентификации.
`s3:delimiter`	Задает разделитель, который должны включать запросы пользователей.
`s3:max-keys`	Задает максимальное количество ключей, возвращаемых на запрос ListBucket.
`s3:prefix`	Ограничивает доступ по префиксу имени ключа.
`s3:signatureage`	Определяет продолжительность времени, в течение которого действительна подпись запроса аутентификации.
`s3:signatureversion`	Задает версию подписки AWS для запросов аутентификации.
`s3:versionid`	Задает доступ к определенной версии объекта.

`s3:x-amz-acl`	Требует заголовок `X-Amz-Acl` с заданным ACL в запросе.
`s3:x-amz-content-sha256`	Запрещает неподписанное содержимое в запросе.
`s3:x-amz-copy-source`	Ограничивает источник копирования определенным бакетом, префиксом или объектом.
`s3:x-amz-grant-full-control`	Требует заголовок `X-Amz-Grant-Full-Control` (полный доступ) в запросе.
`s3:x-amz-grant-read`	Требует заголовок `X-Amz-Grant-Read` (доступ на чтение) в запросе.
`s3:x-amz-grant-read-acp`	Требует заголовок `X-Amz-Grant-Read` (доступ на чтение ACL) в запросе.
`s3:x-amz-grant-write`	Требует заголовок `X-Amz-Grant-Write` (доступ на запись) в запросе.
`s3:x-amz-grant-write-acp`	Требует заголовок `X-Amz-Grant-Write` (доступ на запись ACL) в запросе.
`s3:x-amz-metadata-directive`	Задает принудительный выбор копирования или замены при копировании объектов.

Примеры политик, в которых условия проверяются с логикой `И`

Несколько условий в одном правиле

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::sample-bucket/*",
    "Condition": {
      "Bool": {
        "aws:sourceip": "192.168.1.1"
      }
    },
    "Condition": {
      "Bool": {
        "aws:userid": "ajelcjkv67ar********"
      }
    }
  }
}
```

Несколько ключей в одном условии

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::sample-bucket/*",
    "Condition": {
      "Bool": {
        "aws:sourceip": "192.168.1.1",
        "aws:userid": "ajelcjkv67ar********"
      }
    }
  }
}
```

Для каждого ключа условия вы можете задать одновременно несколько значений. При этом такие значения будут проверяться с логикой `или`. То есть для выполнения правила запрос должен будет соответствовать любому из заданных значений ключа условия.

Пример политики, в которой условия проверяются с логикой `или`

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::sample-bucket/*",
    "Condition": {
      "Bool": {
        "aws:sourceip": [
          "192.168.1.1",
          "192.168.1.2"
        ]
      }
    }
  }
}
```

Операторы условия

Операторы условия применяются для проверки соответствия значения ключа в условии политики со значением в контексте запроса.

Логические операторы

Логические операторы позволяют создавать условия для сравнения ключа со логическим значением `true` или `false`.

Оператор условия	Описание
`Bool`	Соответствует заданному логическому значению.

Операторы даты и времени

Операторы даты и времени позволяют создавать условия для сравнения ключа со значением даты и времени.

Оператор условия	Описание
`DateEquals`	Соответствует заданной дате.
`DateGreaterThan`	Дата и время больше указанных.
`DateGreaterThanEquals`	Дата и время соответствует или больше указанных.
`DateLessThan`	Дата и время меньше указанных.
`DateLessThanEquals`	Дата и время соответствует или меньше указанных.
`DateNotEquals`	Не соответствует заданной дате.

Операторы IP-адресов

Операторы IP-адресов позволяют создавать условия для сравнения ключа с IP-адресом хоста или диапазоном IP-адресов в формате CIDR.

Оператор условия	Описание
`IPAddress`	Определенный IP-адрес или диапазон IP-адресов.
`NotIPAddress`	Все IP-адреса, кроме указанного IP-адреса или диапазона IP-адресов.

Числовые операторы

Числовые операторы позволяют создавать условия для сравнения ключа с числовым значением типа integer или decimal.

Оператор условия	Описание
`NumericEquals`	Значение равно заданному.
`NumericGreaterThan`	Значение больше заданного.
`NumericGreaterThanEquals`	Значение больше либо равно заданного.
`NumericLessThan`	Значение меньше заданного.
`NumericLessThanEquals`	Значение меньше либо равно заданного.
`NumericNotEquals`	Значение не равно заданному.

Строковые операторы

Строковые операторы позволяют создавать условия для сравнения ключа со строковым значением.

Оператор условия	Описание
`StringEquals`	Значение соответствует, учитывается регистр символов.
`StringEqualsIgnoreCase`	Значение соответствует, не учитывается регистр символов.
`StringLike`	Значение соответствует. В значениях можно использовать знаки подстановки: - `*` – несколько символов. - `?` – один символ.
`StringNotEquals`	Значение не соответствует, учитывается регистр символов.
`StringNotEqualsIgnoreCase`	Значение не соответствует, не учитывается регистр символов.
`StringNotLike`	Значение не соответствует. В значениях можно использовать знаки подстановки: - `*` – несколько символов. - `?` – один символ.

Оператор IfExists

К имени любого оператора (кроме `Null`) можно добавить `IfExists` — например, `BoolIfExists`. Элемент условия с подобным оператором означает:

- Если ключ политики присутствует в контексте запроса, ключ будет обработан, как указано в политике.

- Если ключ отсутствует, элемент примет значение `true`.

Оператор Null

Результат оператора `Null` принимает значение `true`, если ключ условия отсутствует в запросе в момент аутентификации. Если ключ присутствует и его значение задано, результат равен `false`.