Руководство пользователя

(разработчика)

Программного обеспечения

«Динамика FileIo»

Оглавление

1 История документа

1.1 Лист сведений о документе

Название	Руководство пользователя (разработчика) Программного обеспечения «Динамика FileIo»
Имя файла	Руководство пользователя (разработчика) Программного обеспечения «Динамика FileIo»
Редакция	1.0
Дата	05.03.2026

1.2 Редакции документа

Дата	Версия	Изменения	Автор
05.03.2026	1.0	Базовая версия документа	

2 Цели и автоматизируемый функции

2.1 Назначение документа

Данное руководство описывает типовые способы встраивания «Динамика FileIo» в различные программные продукты на различных языках.

2.2 Описание системы

О протоколе S3

Протокол S3 (Amazon Simple Storage Service) — это стандарт де-факто для взаимодействия с объектными хранилищами. Он представляет собой RESTful веб-сервис, который позволяет выполнять операции с данными через обычные HTTP/HTTPS запросы .

Dynamika FileIo полностью совместим с S3 API, что означает:

- Вы можете использовать стандартные AWS SDK для большинства популярных языков программирования.
- Инструменты, разработанные для Amazon S3 (например, Cyberduck, WinSCP, AWS CLI), будут работать с Dynamika FileIo "из коробки" .
- Вам не нужно изучать новый проприетарный API — знания, полученные при работе с S3, полностью применимы.

Ключевые понятия

- **Бакет (Контейнер, Bucket)** — логическая емкость для хранения объектов. Аналогична папке верхнего уровня или разделу диска. Имя бакета должно быть уникальным в пределах всего хранилища.
- **Объект (Object)** — единица хранения, состоящая из самих данных файла и метаданных (информации о файле).
- **Ключ (Key)** — уникальный идентификатор объекта внутри бакета. Полный путь к объекту выглядит как bucket_name/key_name. Ключ может включать символы "/", что имитирует иерархическую структуру папок (например, images/2024/photo.jpg).
- **Access Key / Secret Key** — пара ключей для аутентификации. Access Key является идентификатором (логином), Secret Key используется для создания цифровой подписи запроса (паролем) .
- **Endpoint** — URL-адрес, по которому доступен API сервиса (например, https://s3.dynamika.ru или https://storage.example.com:9000).

3 Руководство пользователя

3.1 Подключение к Dynamika FileIo

3.1.1 Получение учетных данных

Для доступа к Dynamika FileIo необходима пара ключей доступа: **Access Key** и **Secret Key**. Эти ключи создаются администратором системы через веб-консоль Dynamika FileIo или с помощью клиента командной строки mc .

3.1.2 Endpoint и модели адресации

При подключении к S3-совместимому сервису необходимо указать endpoint. В зависимости от настроек сервера, поддерживаются две модели адресации бакетов :

1. **Path-style (стиль пути) — используется по умолчанию**: бакет указывается как часть пути в URL.

   text

   https://<endpoint>/<bucket_name>/<object_key>
   Пример: https://s3.dynamika.ru/mydata/photos/logo.png
2. **Virtual-hosted-style (виртуальный хост)**: бакет указывается как поддомен. Эта модель является стандартом для Amazon S3.

   text

   http://<bucket_name>.<endpoint>/<object_key>
   Пример: https://mydata.s3.dynamika.ru/photos/logo.png

При настройке SDK для работы с Dynamika FileIo часто требуется явно включить Path-style адресацию с помощью параметра `s3_force_path_style = true` (или аналогичного), если ваш сервер не поддерживает виртуальные хосты .

3.2 Аутентификация запросов (Signature V4)

3.2.1 Принцип работы

Dynamika FileIo, следуя стандартам AWS, использует механизм подписи запросов **AWS Signature Version 4**. Это обеспечивает :

- **Подтверждение личности**: Запрос подписывается с использованием Secret Key.
- **Защиту данных в пути**: Подпись вычисляется на основе содержимого запроса (headers, тело), что гарантирует его целостность.
- **Защиту от повторного использования (replay attacks)**: Подпись включает временную метку и действительна в течение ограниченного времени (по умолчанию 15 минут) .

3.2.2 Методы передачи подписи

Существует три основных способа передать подпись серверу :

1. **HTTP-заголовок Authorization** (самый распространенный):
   Запрос включает заголовок `Authorization`, содержащий всю необходимую информацию (алгоритм, дата, область действия, подписанные заголовки, сама подпись).
2. **Query-параметры (Presigned URL)**:
   Параметры аутентификации добавляются прямо в URL. Это позволяет создать временную ссылку на объект, которую можно передать третьему лицу или использовать для загрузки из браузера без передачи Secret Key.

   text

   ```
   https://<endpoint>/bucket/file.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=...&X-Amz-Date=...&X-Amz-Expires=3600&X-Amz-Signature=...
   ```
3. **POST-формы (Browser-based uploads)**:
   Используется для загрузки файлов непосредственно из браузера. Поля формы содержат политику и подпись.

3.3 Базовые операции S3 API

3.3.1 Управление бакетами

- CreateBucket — создание нового бакета .
- ListBuckets — получение списка всех бакетов.
- DeleteBucket — удаление пустого бакета.
- HeadBucket — проверка существования бакета и наличия прав доступа к нему.

3.3.2 Управление объектами

- PutObject — загрузка объекта в бакет. Если объект с таким ключом уже существует, он будет перезаписан (если не включено версионирование) .
- GetObject — скачивание объекта .
- HeadObject — получение только метаданных объекта без его тела. Полезно для проверки существования или получения Content-Type, ETag и т.д. .
- CopyObject — копирование объекта внутри бакета или между бакетами .
- DeleteObject — удаление объекта.
- ListObjects / ListObjectsV2 — получение списка объектов в бакете (до 1000 за один запрос). Поддерживает фильтрацию по префиксу (папке) .

3.4 Примеры использования с AWS SDK

3.4.1 Python (boto3)

Установка: pip install boto3

python

```
import boto3
from botocore.client import Config

# Конфигурация подключения
endpoint = 'https://s3.dynamika.ru'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

# Создание клиента с поддержкой path-style адресации
s3_client = boto3.client(
    's3',
    endpoint_url=endpoint,
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key,
```

```
    config=Config(signature_version='s3v4', s3={'addressing_style': 'path'})
)

# 1. Создание бакета
bucket_name = 'my-python-bucket'
s3_client.create_bucket(Bucket=bucket_name)

# 2. Загрузка файла
s3_client.upload_file('local_image.jpg', bucket_name, 'images/remote_image.jpg')
print(f"Файл загружен в s3://{bucket_name}/images/remote_image.jpg")

# 3. Скачивание файла
s3_client.download_file(bucket_name, 'images/remote_image.jpg', 'downloaded_image.jpg')

# 4. Получение списка объектов
response = s3_client.list_objects_v2(Bucket=bucket_name, Prefix='images/')
if 'Contents' in response:
    for obj in response['Contents']:
        print(f" - {obj['Key']} ({obj['Size']} bytes)")
```

3.4.2 Node.js (AWS SDK v3)

Установка: `npm install @aws-sdk/client-s3`

javascript

```
import { S3Client, CreateBucketCommand, PutObjectCommand, ListObjectsV2Command } from "@aws-sdk/client-s3";
import { readFileSync } from 'fs';

const endpoint = 'https://s3.dynamika.ru';
const accessKey = 'YOUR_ACCESS_KEY';
const secretKey = 'YOUR_SECRET_KEY';

const client = new S3Client({
  endpoint: endpoint,
  region: 'us-east-1', // или любая другая, если не принципиально
  credentials: {
    accessKeyId: accessKey,
    secretAccessKey: secretKey,
  },
  forcePathStyle: true, // Включаем path-style адресацию
});
```

```
const bucketName = 'my-nodejs-bucket';

async function run() {
  try {
    // Создание бакета
    await client.send(new CreateBucketCommand({ Bucket: bucketName }));
    console.log(`Bucket ${bucketName} created.`);

    // Загрузка объекта из строки
    await client.send(new PutObjectCommand({
      Bucket: bucketName,
      Key: 'hello.txt',
      Body: 'Hello, Dynamika FileIo!',
    }));
    console.log('Object uploaded.');

    // Список объектов
    const list = await client.send(new ListObjectsV2Command({ Bucket: bucketNam
e }));
    console.log('Objects in bucket:', list.Contents);

  } catch (error) {
    console.error('Error:', error);
  }
}

run();
```

3.4.3 Java (io.minio 8.5.7)

Добавьте в ваш pom.xml следующие зависимости:

xml

```
<dependencies>
    <!-- Spring Boot Starter Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- MinIO Java SDK -->
    <dependency>
```

```
        <groupId>io.minio</groupId>
        <artifactId>minio</artifactId>
        <version>8.5.7</version>
    </dependency>

    <!-- Для работы с аннотациями ConfigurationProperties -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-configuration-processor</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Конфигурация приложения (application.properties)

properties

```
# Dynamika FileIo / MinIO Configuration
minio.endpoint=http://s3.dynamika.ru:9000
minio.access-key=YOUR_ACCESS_KEY
minio.secret-key=YOUR_SECRET_KEY
minio.secure=false
minio.default-bucket=my-spring-bucket
```

Класс свойств конфигурации (Properties Class)

java

```
package com.dynamika.fileio.config;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
@ConfigurationProperties(prefix = "minio")
public class MinioProperties {

    private String endpoint;
    private String accessKey;
    private String secretKey;
    private boolean secure;
    private String defaultBucket;

    // Геттеры и сеттеры
```

```
    public String getEndpoint() {
        return endpoint;
    }

    public void setEndpoint(String endpoint) {
        this.endpoint = endpoint;
    }

    public String getAccessKey() {
        return accessKey;
    }

    public void setAccessKey(String accessKey) {
        this.accessKey = accessKey;
    }

    public String getSecretKey() {
        return secretKey;
    }

    public void setSecretKey(String secretKey) {
        this.secretKey = secretKey;
    }

    public boolean isSecure() {
        return secure;
    }

    public void setSecure(boolean secure) {
        this.secure = secure;
    }

    public String getDefaultBucket() {
        return defaultBucket;
    }

    public void setDefaultBucket(String defaultBucket) {
        this.defaultBucket = defaultBucket;
    }
}
```

Класс конфигурации MinIO Client

java

```
package com.dynamika.fileio.config;

import io.minio.MinioClient;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MinioClientConfig {

    private final MinioProperties minioProperties;

    public MinioClientConfig(MinioProperties minioProperties) {
        this.minioProperties = minioProperties;
    }

    @Bean
    public MinioClient minioClient() {
        return MinioClient.builder()
                .endpoint(minioProperties.getEndpoint())
                .credentials(minioProperties.getAccessKey(), minioProperties.getSecretK
ey())
                .build();
    }
}
```

DTO для ответов

java

```
package com.dynamika.fileio.dto;

import java.time.ZonedDateTime;
import java.util.Map;

public class FileInfoDto {
    private String fileName;
    private String bucketName;
    private Long size;
    private ZonedDateTime lastModified;
    private String etag;
```

```
    private String contentType;
    private String url;
    private Map<String, String> metadata;

    // Конструкторы
    public FileInfoDto() {}

    public FileInfoDto(String fileName, String bucketName, Long size,
```

3.4.4 PHP (AWS SDK v3)

Установка: `composer require aws/aws-sdk-php`

php

```
<?php
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\Exception\AwsException;

$endpoint = 'https://s3.dynamika.ru';
$accessKey = 'YOUR_ACCESS_KEY';
$secretKey = 'YOUR_SECRET_KEY';

$options = [
    'version' => 'latest',
    'region' => 'us-east-1',
    'endpoint' => $endpoint,
    'use_path_style_endpoint' => true, // Включаем path-style
    'credentials' => [
        'key' => $accessKey,
        'secret' => $secretKey,
    ],
];

try {
    $s3Client = new S3Client($options);

    // Создание бакета
    $bucketName = 'my-php-bucket';
    $s3Client->createBucket(['Bucket' => $bucketName]);
    echo "Bucket '{$bucketName}' created.\n";
```

```
    // Загрузка объекта
    $s3Client->putObject([
        'Bucket' => $bucketName,
        'Key' => 'hello.txt',
        'Body' => 'Hello from PHP!',
    ]);
    echo "Object uploaded.\n";

    // Получение объекта
    $result = $s3Client->getObject([
        'Bucket' => $bucketName,
        'Key' => 'hello.txt',
    ]);
    echo "Object content: " . $result['Body'] . "\n";

} catch (AwsException $e) {
    echo "Error: " . $e->getMessage() . "\n";
}
```

3.5 Расширенные возможности

3.5.1 Multipart-загрузка больших файлов

Для файлов размером более 100 МБ рекомендуется использовать multipart-загрузку. Файл разбивается на части, которые загружаются параллельно (или последовательно), а затем собираются на сервере .

Процесс состоит из трех шагов:

1. `CreateMultipartUpload` — инициализация процесса, получение `UploadId`.
2. `UploadPart` — загрузка каждой части с указанием её номера и `UploadId`.
3. `CompleteMultipartUpload` — завершение загрузки и сборка объекта.

AWS SDK автоматически используют multipart-загрузку для методов `upload_file` (Python) или `fPutObject` (MinIO JS) при превышении определенного порога .

3.5.2 Presigned URLs (временный доступ)

Presigned URL позволяет предоставить временный доступ к объекту (на чтение или запись) без раскрытия Secret Key. Это идеально для организации прямых загрузок из браузера .

Пример на Python (генерация ссылки для скачивания):

python

```python
# Генерация ссылки, действительной 1 час (3600 секунд)
url = s3_client.generate_presigned_url(
    ClientMethod='get_object',
    Params={'Bucket': bucket_name, 'Key': 'images/remote_image.jpg'},
    ExpiresIn=3600
)
print(f"Временная ссылка: {url}")
```

Пример на Node.js (генерация ссылки для загрузки через POST форму) :

javascript

```javascript
// Генерация данных для POST-формы (загрузка)
const post = await s3.createPresignedPost({
  Bucket: bucketName,
  Key: 'upload/${filename}', // ${filename} будет заменено на имя файла
  Expires: 60, // 60 секунд
  Conditions: [["content-length-range", 0, 1048576]], // максимум 1 МБ
});

console.log("URL формы:", post.url);
console.log("Поля формы:", post.fields);
// Эти данные можно использовать в HTML-форме: <form action={post.url} meth
od="post" enctype="multipart/form-data"> ... </form>
```

3.5.3 Версионирование объектов

Версионирование позволяет хранить несколько версий одного объекта под одним ключом. Это защищает от случайного удаления или перезаписи .

- Включить версионирование можно через веб-консоль или команду `mc version enable`.
- При включенном версионировании, операция `PUT` создает новую версию, а `DELETE` добавляет маркер удаления (сам объект не удаляется).
- Каждая версия имеет уникальный `VersionId`. Операции с объектом позволяют указать `VersionId` для доступа к конкретной версии.

3.5.4 Политики жизненного цикла (ILM)

Правила жизненного цикла позволяют автоматизировать управление объектами. Например :

- Перемещать объекты на более дешевый класс хранения через 30 дней.
- Автоматически удалять старые версии объектов или завершенные multipart-загрузки.

- Удалять объекты с истекшим сроком хранения.

Политики настраиваются через веб-консоль или S3 API.

3.6 Часто задаваемые вопросы (FAQ)

В: Какой порт используется для подключения к Dynamika FileIo?
О: По умолчанию API доступен на порту 9000, а веб-консоль на порту 9001. При использовании HTTPS, обычно это порт 443 .

В: В чем разница между Access Key и Secret Key?
О: Access Key — это открытый идентификатор пользователя (логин). Secret Key — это секретный ключ, используемый для подписи запросов. Secret Key **никогда** не должен передаваться в открытом виде .

В: Почему я получаю ошибку "SignatureDoesNotMatch"?
О: Эта ошибка означает, что сервер не смог воспроизвести вашу подпись. Возможные причины: неверный Secret Key, рассинхронизация времени на клиенте и сервере (проверьте NTP), или неправильно сформированный запрос (например, не те заголовки включены в подпись).

В: Как мне дать доступ к файлу другому человеку, не создавая ему учетную запись?
О: Используйте presigned URL. Сгенерируйте временную ссылку на объект и отправьте ее .

В: Есть ли ограничение на размер загружаемого файла?
О: При использовании стандартного PutObject размер ограничен 5 ГБ. Для файлов большего размера необходимо использовать Multipart Upload, который позволяет загружать объекты до нескольких терабайт .

В: Как настроить автоматическое резервное копирование бакета?
О: Используйте команду mc mirror для периодической синхронизации содержимого бакета с другим бакетом или локальной директорией . Также можно использовать инструменты для резервного копирования, поддерживающие S3 (Veeam, Bacula) .