Client Data Sharing Specification

08/14/2026 11:43 am EDT

Exchanging data with ID5

ID5 uses RClone to send and receive data, which determines the supported transfer protocols.

Our main delivery mechanism is to exchange data directly via an AWS S3 bucket or an S3-compatible object storage (e.g., GCS) controlled by the client.

There are several different ways to share data:

1. Client-Hosted Bucket (push method)

The preferred method is for clients to host their own AWS S3 or S3-compatible storage. To help with this, ID5 will share our AWS account ID.

The required permissions and configuration examples for select cloud providers are detailed below.

2. ID5-Hosted Buckets (pull method)

For clients unable to host their own S3-compatible storage, or for specific use cases where pull is the better fit, we offer an alternative where ID5 hosts the bucket. In this scenario, we support two methods for client authentication and access, both of which require the client to have an AWS account:

Direct Cross-Account Access via Bucket Policy

We attach a bucket policy to the hosted bucket that explicitly grants read (and/or write) permissions to principals in the client's AWS account. Once this is in place, clients are free to manage access on their side — for example, delegating permissions to specific IAM users, roles, or applications through their own IAM policies. This approach is straightforward and works well when your team already has established IAM governance.

IAM Role-Based Access (AssumeRole)

We create a dedicated IAM role in our AWS account with the appropriate permissions on the bucket and configure its trust policy to allow the client's account to assume it. Client applications then call `sts:AssumeRole` to obtain short-lived, scoped credentials. This is the recommended approach for programmatic or automated workloads, as it avoids the need for long-lived credentials and provides a clear audit trail through CloudTrail.

In either case, we require the client's AWS account ID(s) or canonical user ID(s). The choice between these options will depend on your security preferences and existing AWS setup.



For security reasons, ID5 does not support creating buckets or providing client access via Access and Secret keys.

Data Retention

For all storage hosted by ID5, our data retention policy expires data after 90 days.

Supported Data Formats

We prefer to deliver and receive data in Parquet format with `zstd` compression. We also support CSV and JSON, compressed with `gzip` or `zstd`, as well as `snappy` compression for Parquet.

Permissions Details

Amazon Web Services (AWS)

S3 Bucket Policy Requirements

- Account ARN is: **arn:aws:iam::243105029713:root**
- Dedicated IAM role: **arn:aws:iam::243105029713:role/id5-cross-account-data-access**
- We can handle either an entire bucket or a specific prefix.

Permissions are split into read-only and read-write depending on your use case.

Read-only:
Required for retrieving data and verification purposes:

Permission	Scope	Why
`s3:ListBucket`	Bucket	List objects in a bucket or prefix
`s3:GetObject`	Object	Download objects, get metadata

Read-write:
Required when ID5 also uploads data to your bucket:

Permission	Scope	Why
`s3:ListBucket`	Bucket	List objects in a bucket or prefix
`s3:GetBucketLocation`	Bucket	Required by most S3 clients for bucket checks
`s3:PutObject`	Object	Uploads and overwrites
`s3:GetObject`	Object	Metadata/integrity checks
`s3:DeleteObject`	Object	Needed in case of re-delivery of the same dataset
`s3:PutObjectAcl`	Object	Uploads and overwrites with the correct ACL
`s3:CreateMultipartUpload`	Object (multipart)	Initiates multipart uploads for large files *(optional)*
`s3:UploadPart`	Object (multipart)	Uploads each chunk in a multipart upload *(optional)*
`s3:CompleteMultipartUpload`	Object (multipart)	Finalizes the assembled object *(optional)*
`s3:AbortMultipartUpload`	Object (multipart)	Cleans up failed upload fragments *(optional)*

Policy Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::243105029713:root"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/some/prefix/*"
      ]
    },
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::243105029713:root"
      },
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "some/prefix/*"
          ]
        }
      }
    }
  ]
}
```

Other S3-compatible services

For S3-compatible services, we require that the client provide us with:

- The endpoint URL
- An Access Key ID
- A Secret Access Key

The bucket should have a policy equivalent to the AWS S3 permissions described above.

Google Cloud Platform (GCP)

We can interact with GCP buckets in two ways:

S3 Compatibility Layer (HMAC keys)

We require that you generate HMAC keys with the appropriate access to the storage. This leverages the interoperability layer provided by GCP.

Native GCP via Service Account

We require a JSON credentials file for a service account with the permissions listed below.

> **ATTENTION**
> `ObjectAdmin` is **not** required. The built-in Storage Object User or Storage Object Viewer roles cover the necessary permissions. You can also use a Managed Folder for fine-grained access control without granting permissions at the bucket level.

Read-only:

Permission	Scope	Why
`storage.objects.get`	Object	Download objects, get metadata
`storage.objects.list`	Object	List objects in a bucket or prefix
`storage.buckets.get`	Bucket	Bucket existence checks required by most S3 clients

Read-write:

Permission	Scope	Why
`storage.objects.get`	Object	Download / HEAD object
`storage.objects.list`	Object	List objects
`storage.objects.create`	Object	Upload objects
`storage.objects.delete`	Object	Delete objects; also required when overwriting an existing object
`storage.buckets.get`	Bucket	Bucket existence checks
`storage.multipartUploads.create`	Object (multipart)	Initiate multipart uploads
`storage.multipartUploads.abort`	Object (multipart)	Abort incomplete multipart uploads
`storage.multipartUploads.list`	Object (multipart)	List in-progress multipart uploads
`storage.multipartUploads.listParts`	Object (multipart)	List parts of a multipart upload

Azure

Azure is not natively supported for data transfers. However, since ID5 uses RClone, Azure integration can be arranged for clients who cannot use S3 or GCS alternatives — reach out to your ID5 contact to discuss.