Accessing Cloud Storage

Version 2024.2
2024-09-05

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Accessing Cloud Storage
PDF generated on 2024-09-05
InterSystems IRIS® Version 2024.2

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

1

Introduction to Cloud Storage Adapters

InterSystems IRIS® makes it easy to retrieve, store, and delete data from a cloud storage provider like Amazon Web Services (AWS), Azure Blob Storage (Azure) or Google Cloud Platform (GCP). When using an InterSystems product to access cloud storage, you have two options: use an interoperability production or call low-level APIs.

Interoperability productions are designed to connect external systems, and you can use one to access cloud storage. If you want to bring data from cloud storage into your production, create a business service that uses the inbound adapter. Creating a business operation that uses the outbound adapter allows you to delete or upload data in the cloud.

Low-level APIs allow your code to access cloud storage without using the production framework. They give you programmatic access to cloud storage providers using simple calls within your code.

In AWS, data is stored as *objects*. However, other cloud storage providers use the term *blob* to refer to the same concept. Within InterSystems IRIS, data in cloud storage is referred to as a blob.

2

Inbound Adapter for Cloud Storage

The inbound adapter allows a business service to retrieve data from the cloud so it can be processed within an interoperability production. This adapter takes data from cloud storage and puts it into an InboundInput object, which your business service consumes to work with the data.

The classname for the inbound adapter is EnsLib.CloudStorage.InboundAdapter.

In addition to the properties used to connect to cloud storage, the inbound adapter includes properties that determine which blobs are retrieved and whether the cloud blob is deleted after it is retrieved by the adapter.

2.1 Retrieving Blobs

The inbound adapter includes two properties, **Blob Name Prefix** and **Blob Name Pattern**, that determine which blobs are retrieved from cloud storage. Looking at an example is the easiest way to understand how these properties work together. Consider an AWS S3 bucket that contains the following blobs:

```
foo/bar/baz
foo/bar/bash
foo/bar/bang
foo/boo
```

AWS uses `/` in blob names to create virtual hierarchies, similar to how a file system organizes files into directories. Within this scheme, the **Blob Name Prefix** works like a directory name. For example, `foo/` chooses all blobs, while `foo/bar/` only selects the first three blobs. This selection happens on the AWS server side.

After the client gets a list of blobs from the server, **Blob Name Pattern** is used to filter the list further. For example, if `BlobNamePrefix="/foo/bar/"` and `BlobNamePattern="*ba?"`, the adapter retrieves just the first blob. This filtering happens on the client side. The **Blob Name Pattern** property supports the wildcards `*` and `?`.

If more than one blob meets the criteria set by **Blob Name Prefix** and **Blob Name Pattern**, the adapter forwards each blob to the business service individually in separate `InboundInput` objects.

2.2 Creating a Business Service

As you develop a custom business service to use the inbound adapter, you need to define the `onProcessInput` method so that it accepts an EnsLib.CloudStorage.InboundInput object as its first argument. The inbound adapter places the data from cloud storage into this object type. As your business service works with the `InboundInput` object, it can access its

three properties: `Name` (the name of cloud storage blob), `Meta` (metadata associated with the cloud storage blob), and `Content` (a stream that contains the data from cloud storage).

If you are unfamiliar with developing a custom business service, you can refer to Defining Business Services for details.

2.3 Reprocessing a Blob

Normally, if a blob has been processed, it will be ignored by future processing. If you need to force the production to reprocess a blob, you can remove the global node that tells the system the blob has been processed:

```
^Ens.AppData(..BusinessHost.%ConfigName, BucketName, BlobName)
```

For example, suppose that the business host `MyProduction.MyService` processes blobs and suppose that its configuration name is `BucketProcessor`. Also suppose that this business host has processed a blob named `mysample/data1` in a bucket named `Bucket1`. To force this business host to reprocess this blob, use the ObjectScript shell, go to the namespace where the production is running, and execute the following command:

ObjectScript

```
kill ^Ens.AppData("BucketProcessor", "Bucket1", "mysample/data1")
```

3

Outbound Adapter for Cloud Storage

A business operation in an interoperability production can use the outbound adapter to delete a blob from cloud storage or upload a new blob. Like all business operations, the business operation performs these actions by calling one of the adapter's methods. For information about the properties of the adapter that allow it to connect to the cloud storage provider, see More About Adapters.

The classname of the outbound adapter is EnsLib.CloudStorage.OutboundAdapter.

3.1 Deleting Blobs

A business operation deletes an blob from cloud storage by calling the DeleteBlob method of the outbound adapter. For example, the business operation could call:

ObjectScript

```
Set tSC = ..Adapter.DeleteBlob(..BucketName, request.BlobName)
```

In this example, the business operation uses a property to identify the cloud storage bucket and the production uses a request message to provide the name of the blob being deleted.

3.2 Uploading Blobs

The outbound adapter provides three different methods for uploading blobs, depending on the source or data type of the data:

- UploadBlobFromStream — Accepts data as a `%Stream.Object`
- UploadBlobFromString — Accepts data as a String
- UploadBlobFromFile — Retrieves data from the file system

The first two parameters of each method specify the bucket name and blob name. The third parameter of each method is used to receive the data or location of the data from the business operation. For example, a business operation that receives a message containing data as a stream could call:

ObjectScript

```
Set tSC = ..Adapter.UploadBlobFromStream(..BucketName,
          request.BlobName, request.Content)
```

3.3 Creating a Business Operation

Most properties used to connect to cloud storage are defined in the outbound adapter. However, the actual bucket name that the production wants to work with must be defined in the business operation and passed to the adapter methods. Commonly, the business operation uses a property that appears in the Management Portal to define the bucket name.

If you need general guidance on developing a custom business operation, see Defining Business Operations.

For your convenience, InterSystems provides a simple business operation, EnsLib.CloudStorage.BusinessOperation, that demonstrates how to delete a blob and upload a blob from a stream. This sample business operation uses two simple message classes, EnsLib.CloudStorage.DeleteRequest and EnsLib.CloudStorage.UploadRequest to delete and upload blobs. EnsLib.CloudStorage.BusinessOperation includes the property BucketName as described above. The outbound adapter does not include this property, so custom business operations must include it.

4

Settings and Details for Cloud Storage Adapters

Inbound adapters and outbound adapters both include properties that identify the cloud storage provider and provide the credentials needed to access that provider. Values for these properties can be defined in the Management Portal once the business service or business operation using an adapter has been added to the production. The properties used to connect to the cloud storage provider are:

Note: The setting names use AWS terminology; however, they can be used with any cloud storage provider by substituting the equivalent property. For example, use "Container Name" for the BucketName property when using Microsoft Azure.

BucketName (inbound only) — Identifies the cloud storage bucket that contains the blobs you want to work with. The outbound adapter does not include this setting. Instead, the associated business operation should define a BucketName property. The included business operation, EnsLib.CloudStorage.BusinessOperation, includes this setting. For details, see Creating a Business Operation.

BlobNamePrefix (inbound only) — see Retrieving Blobs.

BlobNamePatter (inbound only) — see Retrieving Blobs.

DeleteAfterDownload (inbound only) — check this box to indicate the blob should be deleted after download.

StorageProvider — Identifies the cloud storage provider.

EndPoint — Identifies a PrivateLink endpoint.

ProviderCredentialsFile— Credentials needed to access the provider. These should be stored securely. Specify the file path for the credentials file.

- AWS — With AWS, you can leave this blank to use the default credential provider chain to obtain the credentials needed to access and S3 bucket. If you prefer to use a credentials file, you can download the credentials file from AWS and then specify its file path. See, Sign Up for AWS and Create an IAM User for more details.
- GCP — Create access credentials by following Create and manage service account keys.
- Azure — Azure doesn't support credentials files. It uses a connection string instead, see Configure Azure Storage connection strings for details. The connection string contains key-value pairs delimited by semicolons. The string should be edited to remove the semicolons and each key-value pair placed on its own line.

  A sample connection string looks like:

```
DefaultEndpointsProtocol=https;AccountName=sampleuser;AccountKey=5X774mvEs41WxQsOw19PB2Y;EndpointSuffix=core.windows.net
```

This needs to be broken down to create a file that looks like:

```
DefaultEndpointsProtocol=https
AccountName=sampleuser
AccountKey=5X774mvEs41WxQsOw19PB2Y
EndpointSuffix=core.windows.net
```

StorageRegion — Identifies the region of your cloud storage.

- AWS — For a list of AWS regions, see Amazon Regions, Availability Zones, and Local Zones.
- GCP — For a list of GCP regions, see Bucket Locations.
- Azure — The region is implied in the connection string. No explicit setting is required.

Note: The cloud storage adapters were developed using the InterSystems PEX framework, so the ObjectScript source code for the adapter looks different than other adapters. For example, the outbound adapter methods are actually wrappers for methods written in a Java PEX component.

5

Cloud Storage APIs

Your ObjectScript code can upload, download, and delete data from a cloud storage provider by calling a set of low-level APIs, allowing you to access cloud storage without using an interoperability production. Your code interacts with the cloud storage provider by creating a client, then calling the client's methods to perform actions like uploading a blob or deleting a blob. The class for this cloud storage client is %Net.Cloud.Storage.Client. It is the same class for each cloud storage provider.

The cloud storage APIs are simple to use. For example, the following code is all you need to upload a file to an Amazon Web Services S3 bucket:

ObjectScript

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
// Create Cloud Storage Client for S3
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,
               "/home/AWSCredentials", "us-east-1", .tSC)

// Upload file to S3
If myClient.BucketExists(bucketName){
   Do myClient.UploadBlobFromFile(bucketName, blobName, "/usr/file.jpg")
}
// Close client
Do myClient.Close()
```

5.1 Creating a Client

Before working with a cloud storage provider's buckets and blobs, your code must create a cloud storage client using the following syntax:

ObjectScript

```
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(javaServer,
   provider,credentialsFile,region,.tSC,endPoint)
```

Where:

- *javaServer* is the name of an InterSystems external server for Java (also known as a Java gateway). To use the default Java external server rather than creating a custom one, simply leave this argument empty.
- *provider* is an integer that indicates which cloud storage provider is being accessed with the client. For S3 buckets, use `0`.

- *credentialsFile* is a file that contains the credentials used to access the cloud storage provider. The file must be formatted according to the provider's specifications. If you are accessing an S3 bucket, you can leave this argument empty to use the default credential provider chain.
- *region* is the region containing the buckets you want to work with. For a list of AWS regions, see Amazon Regions, Availability Zones, and Local Zones.
- *tSC*, which is returned by reference, is the status code returned by the method call.
- *endPoint* is an optional endpoint for AWS PrivateLink.

5.1.1 Closing a Client

Once you are done working with a provider's buckets and blobs, be sure to use the `Close()` method to close the client that you created. For example:

ObjectScript

```
Do myClient.Close()
```

5.2 Working with Buckets

The cloud storage client includes a set of methods designed to work with a provider's buckets, which are the storage containers for blobs. The signatures of these methods are:

```
Method BucketExists(bucketName As %String) As %Boolean
Method GetBucketInfo(bucketName As %String) As BucketInfo
Method ListBuckets() As %ListOfObjects
Method CreateBucket(bucketName As %String)
Method DeleteBucket(bucketName As %String)
```

For example, to create a cloud storage client in order to retrieve details about a bucket, enter:

ObjectScript

```
Set bucketName = "s3-bucket"
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,
               "/home/AWSCredentials", "us-east-1", .tSC)
Set bucketDetails = myClient.GetBucketInfo(bucketName)
Do myClient.Close()
```

5.2.1 Bucket Details

The cloud storage client uses a %Net.Cloud.Storage.BucketInfo object to represent the details about a bucket. When you call **GetBucketInfo()**, the details about the specified bucket are returned in an instance of %Net.Cloud.Storage.BucketInfo object. Likewise, a call to **ListBuckets()** returns all of the available buckets in a collection of these objects, allowing you to access details about each bucket. To learn more about what bucket details are available, see the properties of %Net.Cloud.Storage.BucketInfo.

For convenience, the %Net.Cloud.Storage.BucketInfo class includes a method that allows you to put the details of a bucket into JSON format; the method is **toJSON()**.

5.3 Retrieving Blob Information

The cloud storage client uses the following methods to retrieve information about blobs in a particular bucket:

```
Method BlobExists(bucketName As %String, blobName As %String) As %Boolean
Method GetBlobInfo(bucketName As %String, blobName As %String) As BlobInfo
Method ListBlobs(bucketName As %String) As %ListOfObjects
```

The client provides separate methods to download the content of a blob.

As an example, if you wanted to retrieve details about a particular blob, like its size, you could enter:

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,"/home/AWSCredentials", "us-east-1",
 .tSC)
Set blobDetails = myClient.GetBlobInfo(bucketName, blobName)
Do myClient.Close()
```

5.3.1 Blob Details

The cloud storage client uses a %Net.Cloud.Storage.BlobInfo object to represent the details about a blob. When you call **GetBlobInfo()**, the details about the specified blob are returned in an instance of %Net.Cloud.Storage.BlobInfo object. Likewise, a call to **ListBlobs()** returns all of the available blobs in a collection of these objects, allowing you to access details about each blob. To learn more about what blob details are available, see the properties of %Net.Cloud.Storage.BlobInfo.

For convenience, the %Net.Cloud.Storage.BlobInfo class includes a method that allows you to put the details of a blob into JSON format: **toJSON()**.

5.4 Uploading Blobs

The cloud storage APIs allow you to upload data and files to cloud storage from InterSystems IRIS®. You can use any of the following methods to upload blobs to a cloud storage provider, depending on the source of the blob's data:

```
Method UploadBlobFromString(bucketName As %String, blobName As %String, content As %String)
Method UploadBlobFromFile(bucketName As %String, blobName As %String, filePath As %String)
Method UploadBlobFromStream(bucketName As %String, blobName As %String, stream As %GlobalBinaryStream)
```

For example, to upload a file to an S3 bucket, you could include:

ObjectScript

```
 Set bucketName = "s3-bucket"
 Set blobName = "s3-object-blob"
 Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,
              "/home/AWSCredentials", "us-east-1", .tSC)
 Do myClient.UploadBlobFromFile(bucketName, blobName, "/usr/file.jpg")
 Do myClient.Close()
```

5.5 Downloading Blobs

You can use the cloud storage APIs to download data from a cloud storage provider in order to work with it in InterSystems IRIS. Various methods are available, allowing you to choose the target format of the data:

```
Method DownloadBlobToString(bucketName As %String, blobName As %String) As %String
Method DownloadBlobToFile(bucketName As %String, blobName As %String, filePath As %String)
Method DownloadBlobToStream(bucketName As %String, blobName As %String) As %GlobalBinaryStream
```

For example, to download a blob from an S3 bucket and store it in a stream, enter:

ObjectScript

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,
               "/home/AWSCredentials", "us-east-1", .tSC)
Set IRISStream = myClient.DownloadBlobToStream(bucketName, blobName)
Do myClient.Close()
```

5.6 Single Method for Uploading Blobs

The cloud storage APIs allow you to upload data and files to cloud storage without using an interoperability production. These class methods allow you to make a single call which will create a client, upload the blob, then close the client:

- `SingleUploadBlobFromFile`
- `SingleUploadBlobFromStream`
- `SingleUploadBlobFromString`

For example, to upload a file to Amazon S3, you could include:

ObjectScript

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
Set credentials = "/home/AWSCredentials"
Set region = "us-east-1"
Set filePath = "/usr/file.jpg"
Set status = ##class(%Net.Cloud.Storage.Client).SingleUploadBlobFromFile(, 0,
             credentials, region, bucketName, blobName, filePath)
```

5.7 Single Method for Downloading Blobs

You can use the cloud storage APIs to download data from a cloud storage provider in order to work with it in InterSystems IRIS. These class methods allow you to make a single call which will create a client, download the blob, then close the client:

- `SingleDownloadBlobToFile`
- `SingleDownloadBlobToStream`
- `SingleDownloadBlobToString`

ObjectScript

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
Set credentials = "/home/AWSCredentials"
Set region = "us-east-1"
Set status = ##class(%Net.Cloud.Storage.Client).SingleDownloadBlobToStream(, 0,
             credentials, region, bucketName, blobName)
```

5.8 Deleting Blobs

Like the other cloud storage APIs, the method to delete a blob from cloud storage is straightforward. All it requires is the name of the blob you want to delete, including the bucket where it is stored.

```
Method DeleteBlob(bucketName As %String, blobName As %String)
```

For example, to delete a blob from an S3 bucket, enter:

ObjectScript

```
Set bucketName = "s3-bucket"
Set blobName = "s3-object-blob"
Set myClient = ##class(%Net.Cloud.Storage.Client).CreateClient(,0,
               "/home/AWSCredentials", "us-east-1", .tSC)
Do myClient.DeleteBlob(bucketName, blobName)
Do myClient.Close()
```