InterSystems™
IRIS Data Platform

Accessing Cloud Storage

Version 2025.1
2025-06-03

InterSystems Corporation   1 Memorial Drive   Cambridge MA 02142   www.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.

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

2
Using the Inbound Adapter for Cloud Storage

Within a production, you can include custom business services that use the inbound cloud adapter to retrieve data from cloud storage. To do so, create one or more business service classes as described here. Then add them to your production and configure them.

2.1 Overall Behavior of the Adapter

The cloud inbound adapter, EnsLib.CloudStorage.InboundAdapter, provides settings that you can use to specify the location of the cloud storage to examine, as well as settings to authenticate to the service provider. When included in a business service, the adapter periodically polls for available resources. Then at each polling interval:

1. If the adapter finds input from its configured data source, it constructs an input object to hold the data, and it calls the internal **ProcessInput()** method of the business service, passing the object to it. The input object depends upon the adapter.
2. The internal **ProcessInput()** method of the business service receives the input object. It then performs basic production tasks such as maintaining internal information as needed by all business services. You do not customize or override this method, which your business service class inherits.
3. The **ProcessInput()** method then calls your custom **OnProcessInput()** method, passing the input object. The requirements for this method are described in Implementing the OnProcessInput() Method.
4. Your custom **OnProcessInput()** method examines the input object and sends it to other hosts in the production (or creates and sends new messages based on the input object).

2.2 Creating a Business Service

To use the cloud inbound adapter, create a business service class as follows:

- Your business service class should extend Ens.BusinessService.
- In your class, the *ADAPTER* parameter should equal EnsLib.CloudStorage.InboundAdapter.

- Your class should implement the **OnProcessInput()** method, as described in Implementing the OnProcessInput() Method.
- For other options and general information, see Defining a Business Service Class.

2.3 Implementing the OnProcessInput() Method

Within your business service class, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As EnsLib.CloudStorage.InboundInput,
                      Output pOutput As %RegisteredObject) As %Status {
}
```

Where:

- *pInput* is the input object created by the adapter, using data retrieved from cloud storage.
- *pOutput* is the generic output argument required in the method signature. You can use a more specific message class in the method signature.

The **OnProcessInput()** method should do the following:

1. Examine the input object (*pInput*) and decide how to use it. The Content property of this object is a stream that contains the data retrieved from the cloud storage.
2. Create an instance of the request message, which will be the message that your business service sends.

   For information on creating message classes, see Defining Messages.
3. For the request message, set its properties as appropriate, using values in the input.
4. Call a suitable method of the business service to send the request to some destination within the production. Specifically, call **SendRequestSync()**, **SendRequestAsync()**, or (less common) **SendDeferredResponse()**. For details, see Sending Request Messages.

   Each of these methods returns a status (specifically, an instance of %Status).
5. Make sure that you set the output argument (*pOutput*). Typically you set this equal to the response message that you have received. This step is required.
6. Return an appropriate status. This step is required.

2.4 Properties of the Input Object

The input object is an instance of EnsLib.CloudStorage.InboundInput, which has the following properties:

- Name is the name of cloud storage blob.
- Meta contains the metadata associated with the cloud storage blob.
- Content is a stream that contains the data from cloud storage.

Your **OnProcessInput()** method should examine these properties and use them as needed for your business case.

2.5 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
Using the Outbound Adapter for Cloud Storage

Within a production, you can use the cloud outbound adapter to access data in cloud storage. To do so, create a custom business operation as described here. After creating the business operation, add it to your production and configure it.

3.1 Creating a Business Operation to Use the Adapter

To use the cloud outbound adapter, create a business operation class as follows:

- Your business operation class should extend Ens.BusinessOperation.
- In your class, the *ADAPTER* parameter should equal EnsLib.CloudStorage.OutboundAdapter.
- In your class, the *INVOCATION* parameter should specify the invocation style you want to use, which must be one of the following.
  - **Queue** means the message is created within one background job and placed on a queue, at which time the original job is released. Later, when the message is processed, a different background job is allocated for the task. This is the most common setting.
  - **InProc** means the message will be formulated, sent, and delivered in the same job in which it was created. The job will not be released to the sender's pool until the message is delivered to the target. This is only suitable for special cases.
- Your class should define a *message map* that includes at least one entry. A message map is an XData block entry that has the following structure:

```
XData MessageMap
{
<MapItems>
  <MapItem MessageType="messageclass">
    <Method>methodname</Method>
  </MapItem>
  ...
</MapItems>
}
```

- Your class should define all the methods named in the message map. These methods are known as *message handlers*. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

Here *Sample* is the name of the method, *RequestClass* is the name of a request message class, and *ResponseClass* is the name of a response message class. See Defining Messages.

To use the adapter, the method code must call methods of the Adapter property of your business operation. See Deleting Blobs and Uploading Blobs.

- For other options and general information, see Defining a Business Operation Class.

The following example shows the general structure that you need:

Class Definition

```
Class ECLOUD.NewOperation1 Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.CloudStorage.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
{
  Quit $$$ERROR($$$NotImplemented)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="RequestClass">
    <Method>Sample</Method>
  </MapItem>
</MapItems>
}
}
```

3.2 Deleting Blobs

A business operation deletes an blob from cloud storage by calling the **DeleteBlob()** method of the outbound adapter. The signature of this method is as follows:

Class Member

```
method DeleteBlob(bucketName as %String, blobName as %String) as %Status {
}
```

Where:

- *bucketName* is the name of the bucket where the blob is stored.
- *blobName* is the name of the blob.

For example, one of your custom methods could include this:

ObjectScript

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

This code uses the BucketName property to refer to the cloud storage bucket. The name of the blob to delete is taken from the request message.

3.3 Uploading Blobs

The outbound adapter provides three different methods for uploading blobs, depending on the source or data type of the data. The signatures of these methods are as follows:

Class Member

```
method UploadBlobFromFile(bucketName as %String, blobName as %String, filePath as %String) as %Status
 {
}
```

Class Member

```
method UploadBlobFromStream(bucketName as %String, blobName as %String, content as %Stream.Object) as
 %Status {
}
```

Class Member

```
method UploadBlobFromString(bucketName as %String, blobName as %String, content as %String) as %Status
 {
}
```

Where:

- *bucketName* is the name of the bucket where the blob is stored.
- *blobName* is the name of the blob.
- *filePath*, used by **UploadBlobFromFile()**, is the full pathname of the file to be read and uploaded.
- *content*, used by the other methods, is the data to be uploaded. For **UploadBlobFromStream()**, this must be an instance of %Stream.Object. For **UploadBlobFromString()**, this is a string.

For example, one of your custom methods could include this:

ObjectScript

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

3.4 Prebuilt Cloud Business Operation

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.

Cloud Adapter Settings

This section provides reference information for the cloud adapters.

Also see Settings in All Productions.

Settings for the Cloud Inbound Adapter

Provides reference information for settings of EnsLib.CloudStorage.InboundAdapter.

Summary

The cloud inbound adapter has the following settings:

Group	Settings
Cloud Storage	BucketName, BlobNamePrefix, BlobNamePattern, DeleteAfterDownload, AppendTimeStamp, ArchiveBucket, ArchiveFolder, ErrorBucket, ErrorFolder, StorageProvider, EndPoint, ProviderCredentialFile, StorageRegion

The remaining settings are common to all business services. For information, see Settings for All Business Services.

Note: The setting names use AWS terminology; however, they can be used with any cloud storage provider. For example, when using Microsoft Azure, to specify the BucketName setting, use the Microsoft Azure container name.

AppendTimeStamp

Flag to indicate whether appending a timestamp when archiving a blob .

ArchiveBucket

Name of the bucket to use when archiving a blob. This bucket must already exist and must be writable.

ArchiveFolder

Name of the folder (in the ArchiveBucket) to use when archiving a blob.

BucketName

Identifies the cloud storage bucket that contains the blobs you want to work with. This bucket must already exist and must be writable.

BlobNamePrefix

`BlobNamePrefix` and `BlobNamePattern` 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 `BlobNamePrefix` 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, `BlobNamePattern` 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 `BlobNamePattern` property supports the wildcards `*` and `?`.

If more than one blob meets the criteria set by `BlobNamePrefix` and `BlobNamePattern`, the adapter forwards each blob to the business service individually in separate `InboundInput` objects.

BlobNamePattern

See BlobNamePrefix.

DeleteAfterDownload

If this setting is enabled, the adapter deletes the blob after downloading it.

ErrorBucket

Name of the bucket to use when an error occurs when archiving a blob. This bucket must already exist and must be writable.

ErrorFolder

Name of the folder (in the ErrorBucket) to use when an error occurs when archiving a blob.

StorageProvider

Identifies the cloud storage provider.

EndPoint

PrivateLink endpoint.

ProviderCredentialFile

Credentials needed to access the provider. These should be stored securely.

- 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.

Settings for the Cloud Outbound Adapter

Provides reference information for settings of EnsLib.CloudStorage.OutboundAdapter.

Summary

The cloud outbound adapter has the following settings:

Group	Settings
Cloud Storage	StorageProvider, EndPoint, ProviderCredentialFile, StorageRegion

The remaining settings are common to all business operations. For information, see Settings for All Business Operations.

StorageProvider

Identifies the cloud storage provider.

EndPoint

PrivateLink endpoint.

ProviderCredentialFile

- 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
  ```

If you are working with AWS, leave blank to use the default credential provider chain to obtain the credentials needed to access an S3 bucket. If you prefer to use a credential file, enter its pathname.

StorageRegion

Identifies the region of your cloud storage. For a list of AWS regions, see Amazon Regions, Availability Zones, and Local Zones.

- 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.

4

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()
```

4.1 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.

4.1.1 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()
```

4.2 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()
```

4.2.1 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()**.

4.3 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()
```

4.3.1 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()**.

4.4 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()
```

4.5 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()
```

4.6 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)
```

4.7 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)
```

4.8 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()
```