Clone Cloud Store [CCS]

Release 0.7

Frederic Bregier

Jan 07, 2024

CONTENTS:

Version

Date
Jan 07, 2024

CHAPTER

ONE

ARCHITECTURE

1.1 Description

This project uses Quarkus, the Supersonic Subatomic Java Framework.

If you want to learn more about Quarkus, please visit its website: https://quarkus.io/ .

Clone Cloud Store (*CCS*) allows to simplify access to Cloud Storage for major services such as Amazon S3 or S3 like implementations, Azure Blob Storage and Google Cloud Storage.

It provides a simple REST API, much more simpler than usual ones, for Quarkus environments.

One of the goal is to simplify handling big InputStream files, without having to store them physically on disk or in memory, neither in client application neither in front CCS services.

To allow this, specific functionalities of Quarkus Rest services (client and server) are used, such as the possibility to send or receive such InputStream, chunk by chunk, and with back pressure control.

> *It might be possible to use other Http Client, but one has to take care of possible limitations of such Http SDK, such as to not send or receive from client side with all InputStream in memory.*

Clone Cloud Store allows also to clone storage between various Cloud sites, even using different technologies (for instance, one using Amazon S3, another one using Azure Blob Storage):

- It can be used in 1 site only, or multiples sites (no limitations). When a bucket or object is created/deleted on one site, it is automatically asynchronously replicated to other sites. If an object is missing, due to outage or local issues, it can try to reach a copy synchronously on one of the remote sites and proceeds if it exists to its local restoration asynchronously.
- It provides a Reconciliation algorithm which will compare all sites to restore existing Bucket and Objects everywhere. This process is not blocking, meaning the sites can continue to run as usual.
- This reconciliation process allows Disaster Recovery process, without interruption of service during recovery. Note that new creation/deletion of Buckets or Objects is taken into account during reconciliation.
- This reconciliation process allows Cloud migration, without interruption of service during cloning. Note that new creation/deletion of Buckets or Objects is taken into account during reconciliation.

Cloud Clone Store relies on Quarkus and other technologies:

- A database to store the status of Buckets and Objects: MongoDB or PostgreSql
- A topic/queue system to allow asynchronous actions: Kafka or Pulsar
- Optional Prometheus to get observability metrics
- At least 5 JVM processes: (more JVM can be setup to improve reliability and performance) - Accessor (1 or more) - Accessor for Replicator (1 or more) - Replicator (1 or more) - Reconciliator (1) - Administration (1)

A simplest implementation with 1 JVM (1 or more) is available without database, topic or remote sites support. It allows to test the solution with your application or to allow a smooth move to Cloud Clone Store: **Accessor Simple Gateway**

1.1.1 Status logic

Status \ Type	Bucket	Object
UNKNOWN	No status	No status
IN_PROGRESS	Creation in progress	Creation in progress
AVAILABLE	Created and available	Created and available
ERROR	Creation in error	Creation in error
DELETING	Deletion in progress	Deletion in progress
DELETED	Deleted and unavailable	Deleted and unavailable
ERROR_DELETE	Deletion in error	Deletion in error

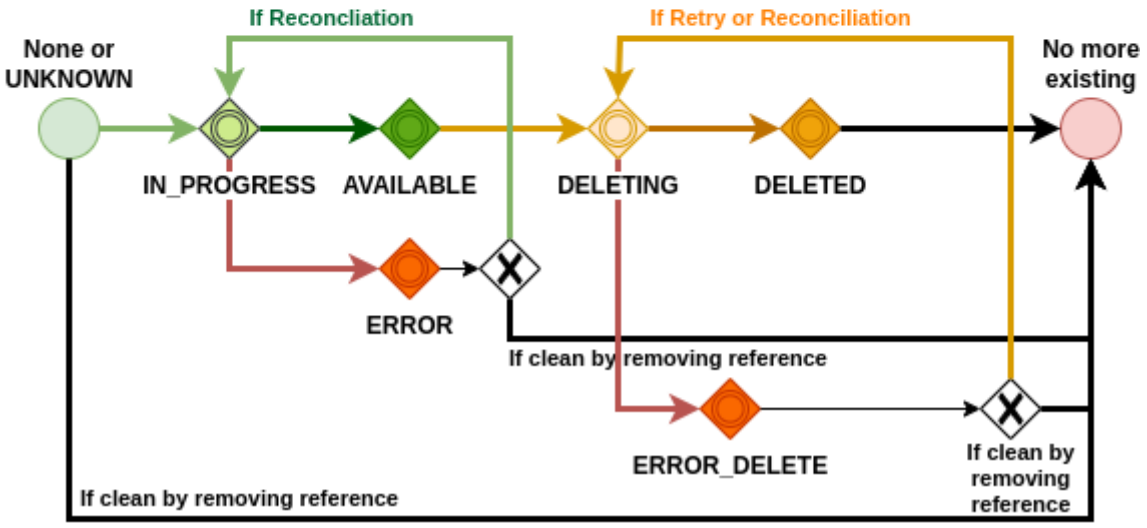

Fig. 1: Status for Objects and Buckets

1.1.2 Architecture

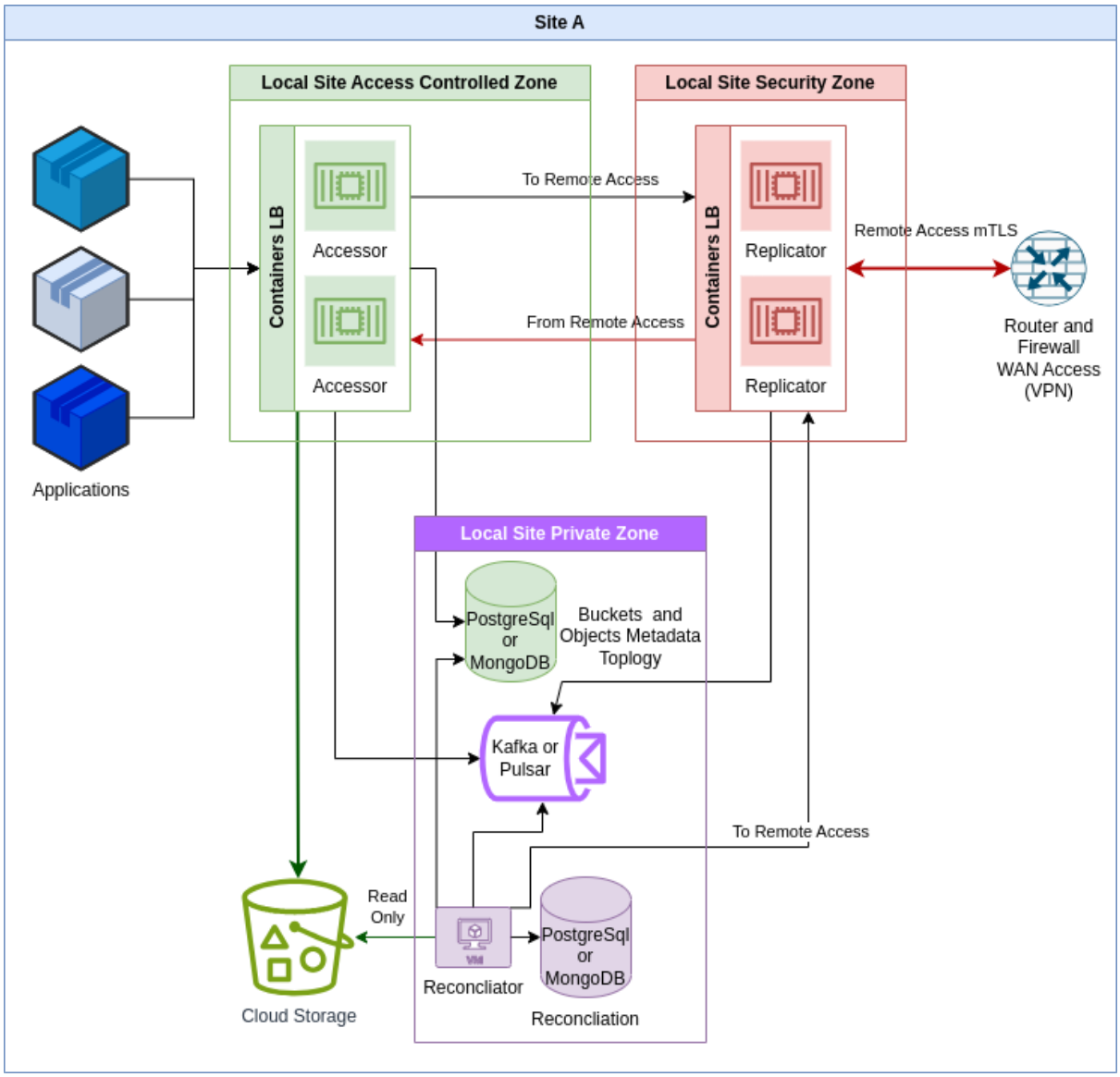

Fig. 2: Architecture on 1 site

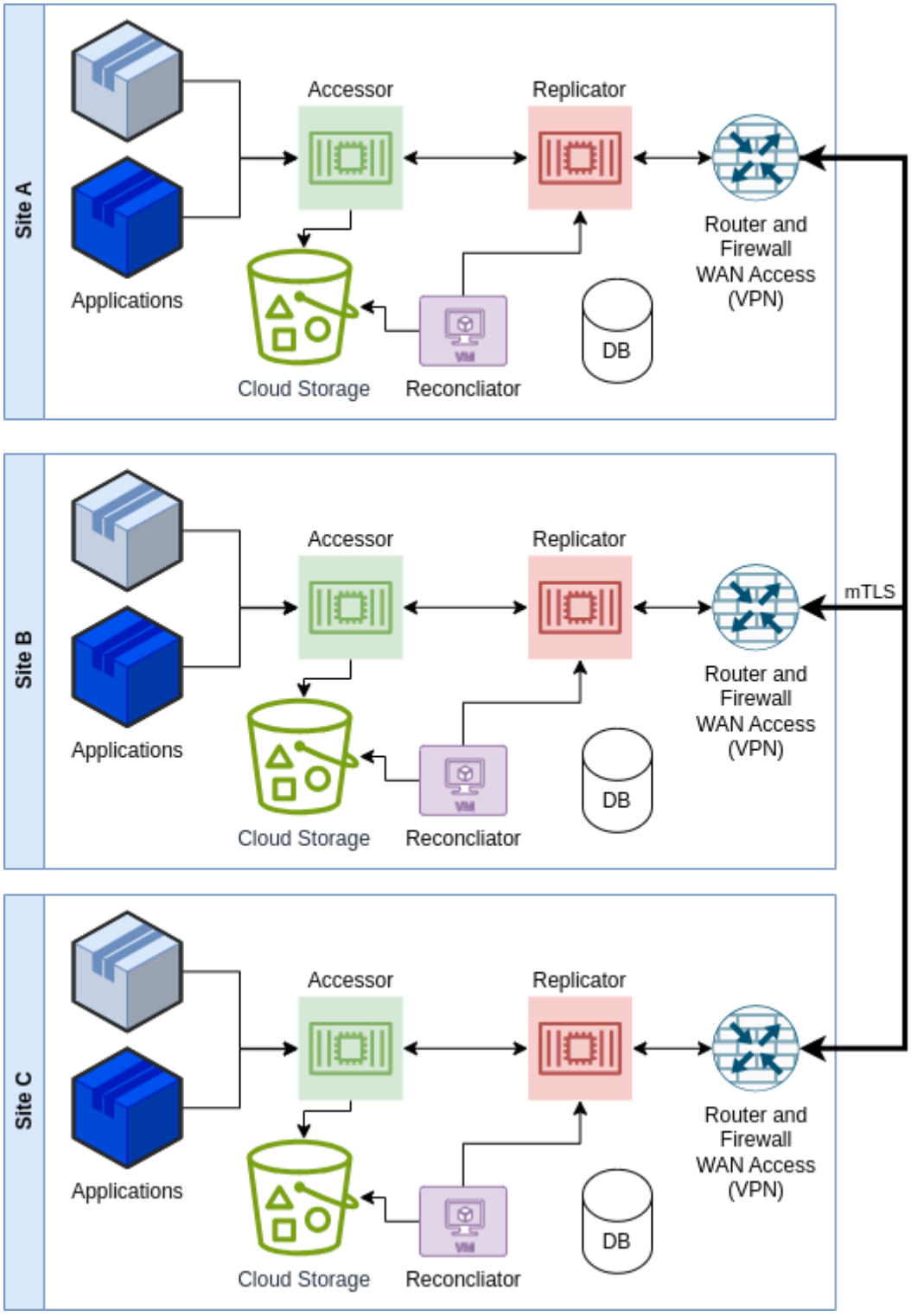

Fig. 3: Architecture on multiple sites

1.1.3 Disaster Recovery or Cloud Migration

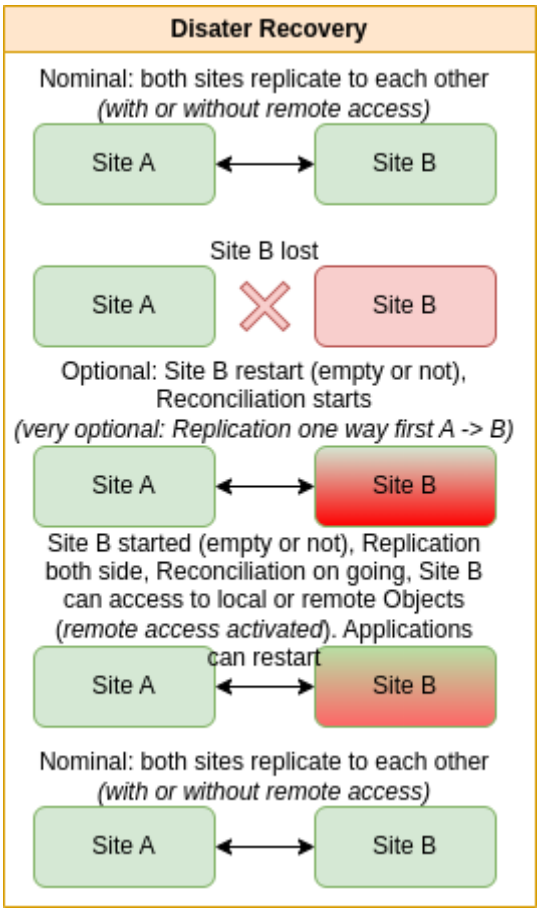

Fig. 4: Disaster Recovery

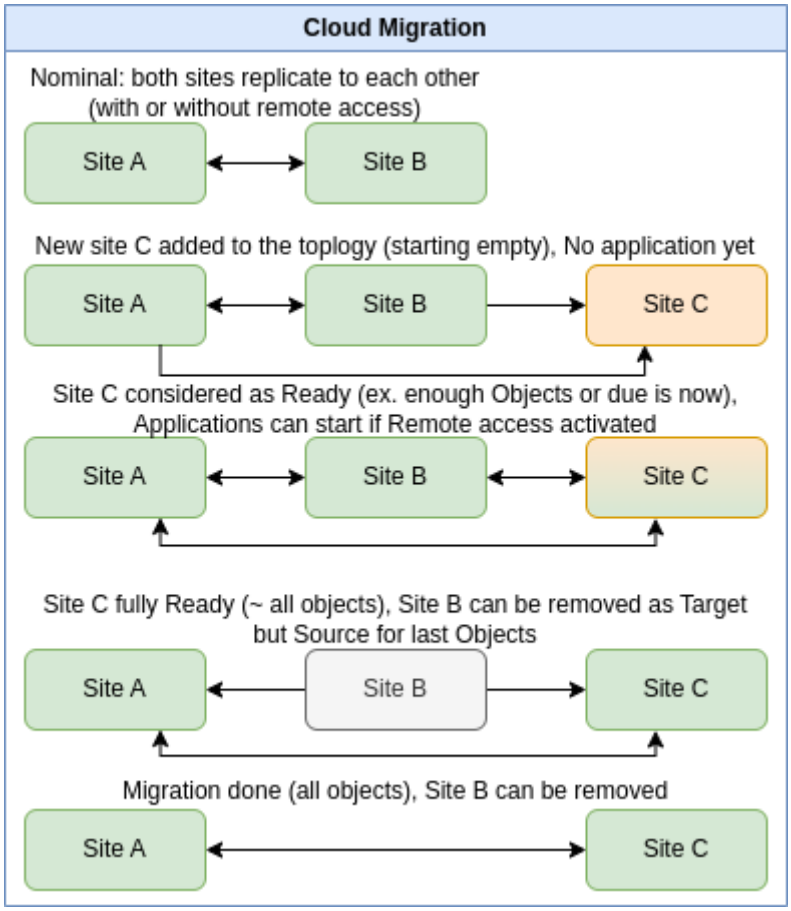

Fig. 5: Cloud Migration

1.2 Missing or In Progress Functionalities

- API could change, in particular Accessor public API (for client application) (see Client Authentication)
- Client without Quarkus is still on going
  - The idea is to allow non Quarkus application in Java to have a ready client SDK.
  - For Quarkus application, the client already exists
- Client authentication
  - Could be done through MTLS or OIDC
  - For CCS interservice authentication, MTLS is the choice but not yet implemented
  - Note that API, in particular public API of Accessor Service, could change due to choice of Authentication; For instance, currently, the clientId is passed as a header but later on could be deduced from authentication
- Reconciliation
  - First steps on Reconciliation computations are still in progress
  - Note that replication is active and remote access if not locally present is possible (through configuration)
- PostgreSQL full support
  - Currently, only MongoDB is fully supported.
  - PostgreSQL shall be available soon.

    - Missing Liquibase configuration for both PostgreSql and MongoDB
- Kafka is the default Topic manager. However, switching to Apache Pulsar should be easy by just applying changes to pom.xml (moving from Kafka to Pulsar) and to application.yaml to ensure correct configuration is done.
- Advanced functionalities such as:
    - Allowing specific access on all or part of CRUD options to a Bucket owned by an application to another one (for instance, to allow producer / consumer of files)
    - Compression of HTTPS link is functional but not yet activated (and will be based on a property)
    - Bandwidth limitation is moved to Quarkus normal configuration (see https://quarkus.io/guides/http-reference#configure-traffic-shaping)
        - It shall be useful only for Replicator and in particular in outbound global mode per site
    - Quarkus Metrics are available but not yet for actions within Clone Cloud Store. The work is on going.
    - Health check service to be done
- Distribution of final jars according to various options is still in debate
    - A choice between Kafka or Pulsar implies 2 different jar due to pom differences
    - However, for PostgreSql or MongoDB, it can be done through configuration so keeping one jar
    - Should it be separate jar (individual per module and per option) or flatten jar (per option)?

CHAPTER

TWO

COMMONS

2.1 Modules

In order to try to keep modular as much as possible, close to an Hexagonal architecture, and to restrict as much as possible the dependencies for external applications using this solution, the following modules are proposed:

- Client side and Server side:
  - **standard**: to hold generic extensions as Guid, Multiple Actions InputStream, Zstd InputStreams, Cipher InputStream, Stream and Iterator utils or ParametersChecker with no Quarkus dependencies
    - Almost all modules depend on this one
  - **quarkus**: to hold extensions for Quarkus until native support comes to Quarkus (in particular efficient InputStream support for both POST and GET using reactive API) and to hold generic extensions as Chunked InputStream, Priority Queue, State Machine or Tea InputStream with Quarkus dependencies or the service identification for CCS
    - Almost all modules depend on this one
    - It relies on current patch Quarkus-patch-client which handles InputStream, until Quarkus fill the gap
  - **quarkus-server**: to hold extensions for Quarkus for Server part
    - It relies on current patch Quarkus-patch-client which handles InputStream, until Quarkus fill the gap
  - **database**: to hold the DB extension using Panache

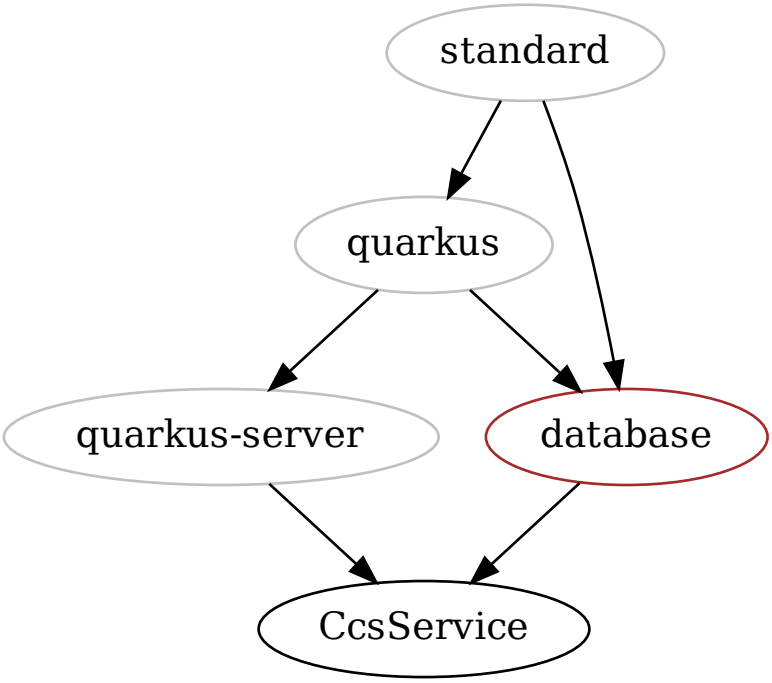

Fig. 1: Dependencies Graph for Cloud Cloud Store Common

2.2 Common Standard

Common Standard is meant for "non Quarkus" usage (does not implied Quarkus libraries), while Common Utils is meant for "Quarkus" context, relying on Common Standard too.

2.2.1 GuidLike and relative Guid

Prefer to use GUID instead of UUID since UUID are not able to be allocated without collision between several independent instances.

One proposed implementation is the **GuidLike**. It uses by default the MAC Address and the PID of the Java process, plus a tenant, the time and an internal counter.

Listing 1: Example code for **GuidLike**

```
// Create Guid simply
String guid = new GuidLike().getId();
String guid = GuidLike.getGuid(); // Equivalent
```

Those Guid could be used in particular when any unique Id is needed, for instance in database model.

For simple Uuid (not Guid), there is also the LongUuid implementation (uniqueness cannot be ensure across several JVM, neither several regions).

Listing 2: Example code for **LongUuid**

```
LongUuid uuid = new LongUuid();
// If hexa form is needed
String hexa = uuid.getId();
// If long form is needed
long id = uuid.getLong();
// or quicker
long id2 = LongUuid.getLongUuid();
```

2.2.2 BaseXx

This is simple encapsulation of other libraries to simplify usage:

Listing 3: Example code for **BaseXx**

```
  final String encoded64 = BaseXx.getBase64("FBTest64P".getBytes());
  final byte[] bytes64 = BaseXx.getFromBase64(encoded64);
  final String encoded32 = BaseXx.getBase32("FBTest32".getBytes());
  final byte[] bytes32 = BaseXx.getFromBase32(encoded32);
  final String encoded16 = BaseXx.getBase16("FBTest16".getBytes());
  final byte[] bytes16 = BaseXx.getFromBase16(encoded16);
```

2.2.3 Various X InputStream

- **ChunkInputStream** : From one InputStream, cut into sub-InputStream with a fix length (useful for multipart support for Object Storage for InputStream greater than 5 GB)
- **MultipleActionsInputStream**: Can compute one Digest from an InputStream, while reading it, and compress or decompress using ZSTD algorithm
- **FakeInputStream**: Only usable in test, create a fake InputStream for a given length with no memory impact (except one buffer) (allows to create 2 TB InputStream for instance). This one is placed in **ccs-test-stream** module since test only related.
- **TeeInputStream**: Used to consume twice (or more) one InputStream. Note that the overall speed will be the slowest consumer speed. If one consumer is closing its own InputStream, it will not affect the others.

Listing 4: Example code for **TeeInputStream**

```
int nbTee = x;
try (FakeInputStream fakeInputStream = new FakeInputStream(INPUTSTREAM_SIZE, (byte) 'a');
     TeeInputStream teeInputStream = new TeeInputStream(fakeInputStream, nbTee)) {
  InputStream is;
  final Future<Integer>[] total = new Future[nbTee];
  final ExecutorService executor = Executors.newFixedThreadPool(nbTee);
  for (int i = 0; i < nbTee; i++) {
    is = teeInputStream.getInputStream(i);
    final ThreadReader threadReader = new ThreadReader(i, is, size);
    total[i] = executor.submit(threadReader);
  }
  executor.shutdown();
  while (!executor.awaitTermination(10000, TimeUnit.MILLISECONDS)) {
    // Empty
  }
  for (int i = 0; i < nbTee; i++) {
    assertEquals(INPUTSTREAM_SIZE, (int) total[i].get());
  }
  // If one wants to know if any of the underlying threads raised an exception on their own InputStream
  teeInputStream.throwLastException();
  // teeInputStream.close() implicit since in Try resource
} catch (final InterruptedException | ExecutionException | IOException e) {
  LOGGER.error(e);
  fail("Should not raised an exception: " + e.getMessage());
}
```

2.2.3.1 ZstdCompressInputStream and ZstdDecompressInputStream

ZSTD (Zstandard) is a modern and efficient compression (both in time, memory and compression).

Those InputStreams allows to compress or decompress on the flY.

General usages should be that those compression / decompression

Listing 5: Example code for **ZstdCompressInputStream and ZstdDecompressInputStream**

```
final ZstdCompressInputStream zstdCompressInputStream = new ZstdCompressInputStream(inputStream);
// Here TrafficShaping is applied once compression is done, and before decompression, as if there were a
// trafficShaping between sending InputStream and receiving InputStream (wire handling)
final var trafficShapingInputStream = new TrafficShapingInputStream(zstdCompressInputStream, trafficShaping);
// Supposedly here: wire transfer
final ZstdDecompressInputStream zstdDecompressInputStream =
    new ZstdDecompressInputStream(trafficShapingInputStream);
int read;
while ((read = zstdDecompressInputStream.read(bytes, 0, bytes.length)) >= 0) {
  // Do something with the decompressed InpuStream
}
zstdCompressInputStream.close();
zstdDecompressInputStream.close();
```

2.2.4 ParametersChecker

Can be used for String (testing also emptiness) and for general Object. For null String only, use the special method.

It allows also some general sanity check to prevent wrong data in entry (such as CDATA or ENTITY in xml, SCRIPTS in Javascript, `;` in sql parameters...). 2 special methods `checkSanityBucketName(name)` and `checkSanityObjectName(name)` are intended to ensure correctness of such names when using Object Storage.

This could be later on extended to use external library specialized in sanity check (such as the Owasp library).

I also includes a special function to fix Instant to milliseconds, instead of 1000 nanoseconds, since most of the database cannot handle more than millisecond.

2.2.5 Various Random

It could be useful (and in particular for Guid) to get random values in an efficient way or in a secure way (a bit less efficient but still efficient).

- **RandomUtil** helps to get efficient Random values
- **SystemRandomSecure** helps to get efficient and secure Random values.

2.2.6 Singleton

Utility class to get standard Singleton (empty and unmodifiable object), such as:

- Empty byte array
- Empty List
- Empty Set
- Empty Map
- Empty InputStream
- Empty OutputStream (moved to **ccs-test-stream** module since test only related)

2.2.7 SysErrLogger

In some rare case, we cannot have a Logger due to the fact the initialization is not done.

In some other case, for quality code reasons, while we do not need to log anything in a catched exception, it is useful to set a log (but we do not want an output).

This is where the SysErrLogger comes.

Listing 6: Example code for **SysErrLogger**

```
try {
  something raising an exception
} catch (final Exception ignore) {
  // This exception shall be ignored
  SysErrLogger.FAKE_LOGGER.ignoreLog(ignore);
}
// Output to SysErr without Logger
SysErrLogger.FAKE_LOGGER.syserr(NOT_EMPTY, new Exception("Fake exception"));
// Output to SysOut without Logger
SysErrLogger.FAKE_LOGGER.sysout(NOT_EMPTY);
```

2.2.8 System Properties and Quarkus Configuration

We need sometimes to get configuration (Quarkus) or System Properties statically and not through injection.

Listing 7: Example code for **SystemPropertyUtil**

```
SystemPropertyUtil.get(KEY, defaultValue);
SystemPropertyUtil.getAndSet(KEY, defaultValue);
SystemPropertyUtil.set(KEY, defaultValue);
// Quarkus Configuration
SystemPropertyUtil.getBooleanConfig(KEY)
SystemPropertyUtil.getStringConfig(KEY);
SystemPropertyUtil.getLongConfig(KEY);
SystemPropertyUtil.getBooleanConfig(KEY);
```

2.3 Common Quarkus

This module contains some class to help handling InputStream within Quarkus efficiently.

Using *Uni* was not possible for InputStream since Quarkus does not support yet correctly InputStream. A patch is submitted to enable this (see https://github.com/quarkusio/quarkus/pull/37308) "@Blocking" mode must be declared imperatively, which means that a new thread is used.

Two cases occur:

- Sending an InputStream to a remote REST API
- Receiving an InputStream from a remote REST API

2.3.1 Client and Server Abstract implementation for InputStream

In order to make it easier to integrate the InputStream management with back-pressure in all APIs, an abstract implementation is provide both for Client ans Server.

The full example is located in the test part of the **ccs-common-quarkus-server**.

- `io.clonecloudstore.common.quarkus.example.model` contains the definition of the model of data (In and Out).
- `io.clonecloudstore.common.quarkus.example.client` contains the **ApiClient** and its factory and the extension of different abstract needed for the client.

The abstract **ClientAbstract** defines some abstract methods that must be specified within the final client implementation, in order to include business implementation.

2.3.1.1 Client sending InputStream

Note that if several API are intended for this client to send InputStream (various usages), one shall specialized the answer of those abstract methods through more general BusinessIn and BusinessOut types (for instance, using multiple sub elements or using instanceOf check).

Listing 8: Zoom on **ClientAbstract** POST way (sending InputStream to server)

```
/**
 * @param context 1 for sending InputStream, -1 for receiving InputStream, or anything else
 * @return the headers map
 */
protected abstract Map<String, String> getHeadersFor(I businessIn, int context);

/**
 * @return the BusinessOut from the response content and/or headers
 */
protected abstract O getApiBusinessOutFromResponse(final Response response);
```

2.3.1.2 Client receiving InputStream

Note that if several API are intended for this client to receive InputStream (various usages), one shall specialized the answer of those abstract methods through more general BusinessIn and BusinessOut types (for instance, using multiple sub elements or using instanceOf check).

Listing 9: Zoom on **ClientAbstract** GET way (receiving InputStream from server)

```
/**
 * @param context 1 for sending InputStream, -1 for receiving InputStream, or anything else
 * @return the headers map
 */
protected abstract Map<String, String> getHeadersFor(I businessIn, int context);

/**
 * @return the BusinessOut from the response content and/or headers
 */
protected abstract O getApiBusinessOutFromResponse(final Response response);
```

2.3.1.3 Client definition of Service

Note that **ApiServiceInterface** is the API of the server, with specific attention on InputStream, using a different Java Interface than the server's one. This is due to the need to access to low level injected values such as `HttpServerRequest` and `Closer`.

Note: these declarations are not useful since the client service will never be used for those end points.

Listing 10: Example test code for **ApiServiceInterface** (client side)

```
@Path(ApiConstants.API_COLLECTIONS)
@POST
@Consumes(MediaType.APPLICATION_OCTET_STREAM)
@Produces(MediaType.APPLICATION_JSON)
Uni<Response> createObject(InputStream content,
    @DefaultValue("name") @RestHeader(ApiConstants.X_NAME) String name,
    @DefaultValue("0") @RestHeader(ApiConstants.X_LEN) long len);

@Path(ApiConstants.API_COLLECTIONS + "/{business}")
@GET
@Produces(MediaType.APPLICATION_OCTET_STREAM)
Uni<InputStream> readObject(@RestPath String business);
```

2.3.1.4 Server definition of Service

Be careful that API using InputStream (push or pull) are defined with the annotation `@Blocking` on server side.

Listing 11: Example test code for **ApiService** (server side)

```
@Path(API_COLLECTIONS)
@POST
@Consumes(MediaType.APPLICATION_OCTET_STREAM)
@Produces(MediaType.APPLICATION_JSON)
@Blocking
public Uni<Response> createObject(HttpServerRequest request, @Context final Closer closer,
    final InputStream inputStream,
    @DefaultValue("name") @RestHeader(X_NAME) String name,
    @DefaultValue("0") @RestHeader(X_LEN) long len) {
  ApiBusinessIn businessIn = new ApiBusinessIn();
  businessIn.name = name;
  businessIn.len = len;
  return createObject(request, closer, businessIn, businessIn.len, null, keepCompressed, inputStream);
}

@Path(API_COLLECTIONS + "/{business}")
@GET
@Produces(MediaType.APPLICATION_OCTET_STREAM)
@Blocking
public Uni<Response> readObject(@RestPath final String business,
    final HttpServerRequest request, @Context final Closer closer) {
  ApiBusinessIn businessIn = new ApiBusinessIn();
  businessIn.name = business;
  String xlen = request.getHeader(X_LEN);
  long len = LEN;
  if (ParametersChecker.isNotEmpty(xlen)) {
    len = Long.parse(xlen);
  }
  businessIn.len = len;
  return readObject(request, closer, businessIn, futureAlreadyCompressed);
}
```

- `keepInputStreamCompressed` specifies for each end point if the InputStream shall be kept compressed if already compressed, or uncompressed if compressed.

The Client Factory should be used as ``@ApplicationScoped`` in order to ensure it is always the unique one.

2.3.1.5 Client implementation

Listing 12: Example test code for **ApiClient**

```
public ApiBusinessOut postInputStream(final String name, final InputStream content,
    final long len, final boolean shallCompress, final boolean alreadyCompressed) throws CcsWithStatusException {
  ApiBusinessIn businessIn = new ApiBusinessIn();
  businessIn.name = name;
  businessIn.len = len;
  final var inputStream = prepareInputStreamToSend(content, shallCompress, alreadyCompressed, businessIn);
  final var uni = getService().createObject(name, len, inputStream);
  return getResultFromPostInputStreamUni(uni, inputStream);
}

public InputStreamBusinessOut<ApiBusinessOut> getInputStream(final String name, final long len,
    final boolean acceptCompressed, final boolean shallDecompress)
```

(continues on next page)

(continued from previous page)

```
    throws CcsWithStatusException {
  ApiBusinessIn businessIn = new ApiBusinessIn();
  businessIn.name = name;
  businessIn.len = len;
  prepareInputStreamToReceive(acceptCompressed, businessIn);
  final var uni = getService().readObject(name);
  return getInputStreamBusinessOutFromUni(acceptCompressed, shallDecompress, uni);
}
```

- `shallCompress` and `acceptCompressed` specify if the InputStream must be compressed (either in POST or GET).
- `alreadyCompressed` specifies if the InputStream is already compressed or not in POST.
- `shallDecompress` specifies if the InputStream shall be decompressed if received compressed.

2.3.1.6 Client implementation using Quarkus Service

It is possible to use native Quarkus client. (service is the injected ApiService with correct URL from `quarkus.rest-client."org.acme.rest.client.ExtensionsService".url=yourUrl`).

Listing 13: Example test code for **ApiClient** using service

```
public class ApiClient extends ClientAbstract<ApiBusinessIn, ApiBusinessOut, ApiServiceInterface> {
  public boolean checkName(final String name) {
    final Uni<Response> uni = getService().checkName(name);
    ApiBusinessIn businessIn = new ApiBusinessIn();
    businessIn.name = name;
    try (final Response response = exceptionMapper.handleUniResponse(uni)) {
      return name.equals(response.getHeaderString(X_NAME));
    } catch (final CcsClientGenericException | CcsServerGenericException | CcsWithStatusException e) {
      return false;
    }
  }
  ...
}
```

Some helpers are created to make it easier to handle the return status.

Listing 14: Example test code for **ExceptionMapper** helper

```
  // Response format
  final Uni<Response> uni = getService().checkName(name);
  try (final Response response = exceptionMapper.handleUniResponse(uni)) {
    // OK
  } catch (final CcsClientGenericException | CcsServerGenericException | CcsWithStatusException e) {
    // Handle exception
  }

  // DTO format
  final var uni = getService().getObjectMetadata(name);
  return (ApiBusinessOut) exceptionMapper.handleUniObject(this, uni);
```

Note that if a Factory is going to be used for several targets, the factory is then not correctly initialized with the right URI. Therefore the following example shall be followed:

Listing 15: Example code for **ApiClientFactory and ApiClient** with multiple targets

```
// Still get the Factory using @Inject
@Inject
ApiClientFactory factory;

// Then in method where the client is needed for a particular URI
try (final ApiClient apiClient = factory.newClient(uri)) {
  // This method is synchronized on Factory to prevent wrong setup
  // (getUri() will return the right URI at construction but not guaranteed later on)
}
```

2.3.1.7 Server implementation

- `io.clonecloudstore.common.quarkus.server` contains the NativeStreamHandlerAbstract, the StreamServiceAbstract and some filters implementations for the server.

With those abstracts, the code needed is shortest and allow to be extended to any API and usages.

The abstract **StreamServiceAbstract** defines abstract methods, as **NativeStreamHandlerAbstract**, that must be specified within the final client implementation, in order to include business implementation.

Listing 16: Zoom on abstract methods in **NativeStreamHandlerAbstract** helper for InputStream received by the server

```
/**
 * @return True if the digest is to be computed on the fly
 */
protected abstract boolean checkDigestToCumpute(I businessIn);

/**
 * Check if the request for POST is valid, and if so, adapt the given MultipleActionsInputStream that will
 * be used to consume the original InputStream.
 * The implementation shall use the business logic to check the validity for this InputStream reception
 * (from client to server) and, if valid, use the MultipleActionsInputStream, either as is or as a standard InputStream.
 * (example: check through Object Storage that object does not exist yet, and if so
 * add the consumption of the stream for the Object Storage object creation).
 * Note that the stream might be kept compressed if keepInputStreamCompressed was specified at construction.
 */
protected abstract void checkPushAble(I businessIn, MultipleActionsInputStream inputStream)
    throws CcsClientGenericException, CcsServerGenericException;

/**
 * Returns a BusinessOut in case of POST (receiving InputStream on server side).
 * The implementation shall use the business logic to get the right
 * BusinessOut object to return.
 * (example: getting the StorageObject object, including the computed or given Hash)
 *
 * @param businessIn businessIn as passed in constructor
 * @param finalHash  the final Hash if computed on the fly, or the original given one
 * @param size       the real size read (from received stream, could be compressed size if decompression is off at
 *                   construction)
 */
protected abstract O getAnswerPushInputStream(I businessIn, String finalHash, long size)
    throws CcsClientGenericException, CcsServerGenericException;

/**
 * Returns a Map for Headers response in case of POST (receiving InputStream on server side).
 * (example: headers for object name, object size, ...)
 *
 * @param businessIn  businessIn as passed in constructor
 * @param finalHash   the final Hash if computed on the fly, or the original given one
 * @param size        the real size read
 * @param businessOut previously constructed from getAnswerPushInputStream
 */
protected abstract Map<String, String> getHeaderPushInputStream(I businessIn, String finalHash, long size,
                                                                O businessOut)
    throws CcsClientGenericException, CcsServerGenericException;
```

Fig. 2: Illustration of network steps in receiving InputStream within server

Listing 17: Zoom on abstract methods in **NativeStreamHandler** helper for InputStream sent by the server

```
/**
 * The implementation must check using business object that get inputStream request (server sending InputStream as
 * result) is valid according to the businessIn from te Rest API and the headers.
 * (example: ObjectStorage check of existence of object)
 *
 * @return True if the read action is valid for this businessIn object and headers
 */
protected abstract boolean checkPullAble(I businessIn, MultiMap headers)
    throws CcsClientGenericException, CcsServerGenericException;

/**
 * Returns the InputStream required for GET (server is sending the InputStream back to the client).
 * The implementation shall use the business logic and controls to get the InputStream to return.
 * (example: getting the Object Storage object stream)
 *
 * @param businessIn businessIn as passed in constructor
 */
protected abstract InputStream getPullInputStream(I businessIn)
    throws CcsClientGenericException, CcsServerGenericException;

/**
```

(continues on next page)

(continued from previous page)

```
  * Returns a Map for Headers response in case of GET, added to InputStream get above  (server is sending the
  * InputStream back to the client)
  * (example: headers for object name, object size...)
  *
  * @param businessIn businessIn as passed in constructor
  */
protected abstract Map<String, String> getHeaderPullInputStream(I businessIn)
    throws CcsClientGenericException, CcsServerGenericException;
```

Fig. 3: Illustration of network steps in sending InputStream within server

Listing 18: Zoom on abstract methods in **NativeStreamHandler** helper for error message (in or out)

```
/**
 * Return headers for error message.
 * (example: get headers in case of error as Object name, Bucket name...)
 */
protected abstract Map<String, String> getHeaderError(I businessIn, int status);
```

Note that if several API are intended for this server to send or receive InputStream (various usages), one shall specialized the answer of those abstract methods through more general BusinessIn and BusinessOut types (for instance, using multiple sub elements or using instanceOf check).

Listing 19: Example test code for **ApiService** (Class definition and REST service definition)

```
@ApplicationScoped
@Path(API_ROOT)
public class ApiService
    extends StreamServiceAbstract<ApiBusinessIn, ApiBusinessOut, NativeStreamHandler> {
```

The interaction with a Driver is done through the extension of **NativeStreamHandlerAbstract**.

Listing 20: Example test code for **NativeStreamHandler**

```
@RequestScoped
public class NativeStreamHandler
    extends NativeStreamHandlerAbstract<ApiBusinessIn, ApiBusinessOut> {
  public NativeStreamHandler() {
  }
  // Implement abstract methods
}
```

2.3.2 PriorityQueue

It might be necessary to handle a PriorityQueue locally with the ability to manage re-prioritization (as done in the Linux scheduler).

Several functions (lambda) shall be passed at construction time:

Listing 21: Example code for **PriorityQueue** creation

```
private static final Comparator<ElementTest> comparator = Comparator.comparingLong(o -> o.rank);
private static final Comparator<ElementTest> findEquals = Comparator.comparing(o -> o.uniqueId);
private static final UnaryOperator<ElementTest> reprioritize = e -> {
  e.rank /= 2;
  return e;
};
private static final int MAX_RUNNABLE = 10;
ManagedRunnable<ElementTest> managedRunnable =
    new ManagedRunnable<>(MAX_RUNNABLE, comparator, findEquals, reprioritize);
```

- `MAX_RUNNABLE` is the number of priority elements that will be managed in a round-robin way equally.
- `comparator` is the function to compare 2 items to find which one is priority (lowest will be placed at first position).
- `findEquals` is the function to find an item equals to the one passed as argument (equality can be based on something different than objects are really the same).
- `reprioritize` is the function that traverse all items (not already in short list live) to reorder them accordingly to the new priority.

The main point is that the queue is split in 2, in order to not have too much items running (or active) at the same time. This is fixed by the maxRunnable parameter. Once items are ordered according to priority, this parameter allows to consume those number of items in a round robin way.

Listing 22: Example code for **PriorityQueue** usage

```
// One can add items 1 by 1
managedRunnable.add(new ElementTest(10));
// This one will have a lower priority
managedRunnable.add(new ElementTest(20));
// One can add items from a collection
List<ElementTest> list = new ArrayList<>();
// This one will be latest one
list.add(new ElementTest(50));
// This one will be the first
list.add(new ElementTest(5));
managedRunnable.addAll(list);
// Now we can start to consume
while (!managedRunnable.isEmpty()) {
  e = managedRunnable.poll();
  if (e != null) { // Queue might be empty when poll is called
```

(continues on next page)

(continued from previous page)

```
    // Do something with e
    ...
    // If e needs to regain Queue as active runner
    if (myTaskNeedsToContinue(e)) {
      // Item will return to active items (round-robin)
      managedRunnable.addContinue(e);
    } else if (myTaskNeedsToBeReprioritize(e)) {
      // Here the item will be add at the end (out of round-robin)
      managedRunnable.add(e);
    }
  }
}
```

One usage could be to select among a lot of actions to be done the top 10 to apply, and then poll out next ones when able to do so.

To prevent that an old entry is never planned, the `reprioritize` is called each time one element will be taken out of the round robin sub-queue, such that it has a chance to become the most priority one.

2.3.3 TrafficShaping

Limiting traffic on network (or any other resource) could be difficult natively. This aims to propose a simple solution.

Since Quarkus implements natively trafficShaping, the project will use this default one.

2.3.4 JsonUtil

Since ObjectMapper from Jackson library is often needed for manual integration, this helper returns an ObjectMapper:

- If Quarkus has initialized it, the one from Quarkus
- If not, a default one, almost equivalent

2.3.5 StateMachine

StateMachine package allows to handle simple State Machine with various steps.

Listing 23: Example code for **StateMachine**

```
// States definition
public enum ExampleEnumState {
  NONE, START, RUNNING, PAUSE, END, ERROR;

  static final List<StateTransition<ExampleEnumState>> configuration;

  static {
    ExampleStateTransition[] stateTransitions = ExampleStateTransition.values();
    configuration = new ArrayList<>(stateTransitions.length);
    for (final ExampleStateTransition stateTransition : stateTransitions) {
      configuration.add(stateTransitions.elt);
    }
  }

  public static StateMachine<ExampleEnumState> newStateMachine() {
    return new StateMachine<>(NONE, configuration);
  }
}
// StateTransition definition
public enum ExampleStateTransition {
  tNONE(NONE, START, EnumSet.of(START)),
  tSTART(START, RUNNING, EnumSet.of(RUNNING, PAUSE, ERROR)),
  tRUNNING(RUNNING, END, EnumSet.of(PAUSE, END, ERROR)),
  tPAUSE(PAUSE, RUNNING), tEND(END), tERROR(ERROR);

  public final StateTransition<ExampleEnumState> elt;

  ExampleStateTransition(final ExampleEnumState state) {
```

(continues on next page)

(continued from previous page)

```
    elt = new StateTransition<>(state);
  }

  ExampleStateTransition(final ExampleEnumState state,
                         final ExampleEnumState stateNext) {
    elt = new StateTransition<>(state, stateNext);
  }

  ExampleStateTransition(final ExampleEnumState state,
                         final ExampleEnumState stateNext,
                         final EnumSet<ExampleEnumState> set) {
    elt = new StateTransition<>(state, stateNext, set);
  }
}
// Example of usages
final StateMachine<ExampleEnumState> stateMachine =
      ExampleEnumState.newStateMachine();
stateMachine.getState(); // NONE
stateMachine.isReachable(END); // False
stateMachine.setDryState(END); // NONE
stateMachine.isReachable(START); // True
stateMachine.setState(START); // START
stateMachine.isTerminal(); // False
```

2.4 Common DB

2.4.1 DB Utils

2.4.1.1 RestQuery, DbQuery and DbUpdate

RestQuery allows to define "standard" query in a Object model, in order to be able to serialize into a Json. This Json can then be sent through REST API.

It focuses on the "Where" condition only and therefore can be used for any SELECT, INSERT or UPDATE command.

DbQuery allows to generate a SQL (PostgreSQL) or NoSQL (MongoDB) query. It can be used to express a request and using the Repository model, it will be taken into account natively, for both model (SQL or NoSQL).

It focuses on the "Where" condition only and therefore can be used for any SELECT, INSERT or UPDATE command.

DbUpdate alows to general e SQL (PostgreSQL) or NoSQL (MongoDB) Update part query. It can be used to express the Update part and using the Repository model, it will be taken into account natively, for both model (SQL or NoSQL).

It focuses on the "Update" part (SET) condition only and therefore can be used for UPDATE command only, in conjunction of a *DbQuery*.

2.4.1.2 StreamHelperAbstract

StreamHelperAbstract allows to handle easily Stream (real Stream) on SELECT. It allows to limit memory usage.

2.4.1.3 RepositoryBaseInterface

RepositoryBaseInterface allows to specify common methods for all repositories, whatever Sql or NoSQL.

Listing 24: Example code for **Global Model definition**

```
@MappedSuperclass
public abstract class DbDtoExample {
  // No Id nor BsonId
  // Here come other fields

  @Transient
  public void fromTransferRequest(DtoExample dto) {
    setGuid(dto.getGuid()); // and other setters
  }

  @Transient
  public DtoExample getTransferRequest() {
    DtoExample transferRequest = new DtoExample();
    transferRequest.setGuid(getGuid()); // and other setters
    return transferRequest;
  }

  public abstract String getGuid();

  public abstract DbDtoExample setGuid(String guid);
}
public interface DbDtoExampleRepository extends RepositoryBaseInterface<DbDtoExample> {
  String TABLE_NAME = "dto_example";
  // Here other field names
}
```

Listing 25: Example code for **Global DTO definition**

```
@RegisterForReflection
public class DtoExample extends DbDtoExample {
  private String guid;

  public String getGuid() {
    return guid;
  }

  public DtoExample setGuid(final String guid) {
    this.guid = guid;
    return this;
  }
}
```

2.4.2 MongoDb

It provides the implementations for all DB-Utils package for NoSQL MongoDB.

Listing 26: Example code for **MongoDB Model Implementation definition**

```
@MongoEntity(collection = TABLE_NAME)
public class MgDbDtoExample extends DbDtoExample {
  @BsonId
  private String guid;

  public MgDbDtoExample() {
    // Empty
  }

  public MgDbDtoExample(final DtoExample dto) {
    fromTransferRequest(dto);
  }

  @Override
  public String getGuid() {
    return guid;
  }

  @Override
  public MgDbDtoExample setGuid(final String guid) {
    this.guid = guid;
    return this;
```

(continues on next page)

(continued from previous page)

```
  }
}

@ApplicationScoped
public class MgDbDtoExampleRepository extends ExtendedPanacheMongoRepositoryBase<DbDtoExample, MgDbDtoExample>
    implements DbDtoExampleRepository {
  @Override
  public String getTable() {
    return TABLE_NAME;
  }
}
```

In addition, it provides **MongoSqlHelper** to help to build SQL request from DbQuery and DbUpdate.

It provides also an abstraction **AbstractCodec** to make easier the declaration of Codec for DTO (see example).

Listing 27: Example code for **AbstractCodec**

```
public class MgDbDtoExampleCodec extends AbstractCodec<MgDbDtoExample> {
  public MgDbDtoExampleCodec() {
    super();
  }

  @Override
  protected void setGuid(final MgDbDtoExample mgDbDtoExample, final String guid) {
    mgDbDtoExample.setGuid(guid);
  }

  @Override
  protected String getGuid(final MgDbDtoExample mgDbDtoExample) {
    return mgDbDtoExample.getGuid();
  }

  @Override
  protected MgDbDtoExample fromDocument(final Document document) {
    MgDbDtoExample mgDbDtoExample = new MgDbDtoExample();
    mgDbDtoExample.setField1(document.getString(FIELD1));
    mgDbDtoExample.setField2(document.getString(FIELD2));
    mgDbDtoExample.setTimeField(document.get(TIME_FIELD, Date.class).toInstant());
    return mgDbDtoExample;
  }

  @Override
  protected void toDocument(final MgDbDtoExample mgDbDtoExample, final Document document) {
    document.put(FIELD1, mgDbDtoExample.getField1());
    document.put(FIELD2, mgDbDtoExample.getField2());
    document.put(TIME_FIELD, mgDbDtoExample.getTimeField());
  }

  @Override
  public Class<MgDbDtoExample> getEncoderClass() {
    return MgDbDtoExample.class;
  }
}
```

2.4.2.1 MongoBulkInsertHelper

MongoBulkInsertHelper allows to handle easily bulk operation on INSERT or UPDATE for MongoDB.

2.4.3 PostgreSQL

It provides the implementations for all DB-Utils package for SQL PostgreSQL.

Listing 28: Example code for **PostgreSQL Model Implementation definition**

```
@Entity
@Table(name = TABLE_NAME,
    indexes = {@Index(name = TABLE_NAME + "_filter_idx", columnList = FIELD1 + ", " + TIME_FIELD)})
public class PgDbDtoExample extends DbDtoExample {
  @Id
  @Column(name = ID, nullable = false, length = 40)
  private String guid;
```

(continues on next page)

(continued from previous page)

```
  public PgDbDtoExample() {
    // Empty
  }

  public PgDbDtoExample(final DtoExample dto) {
    fromDto(dto);
  }

  @Override
  public String getGuid() {
    return guid;
  }

  @Override
  public PgDbDtoExample setGuid(final String guid) {
    this.guid = guid;
    return this;
  }
}

@ApplicationScoped
@Transactional
public class PgDbDtoExampleRepository extends ExtendedPanacheRepositoryBase<DbDtoExample, PgDbDtoExample>
    implements DbDtoExampleRepository {
  public PgDbDtoExampleRepository() {
    super(new PgDbDtoExample());
  }

  @Override
  public String getTable() {
    return TABLE_NAME;
  }
}
```

In addition, it provides **PostgreSqlHelper** to help to build SQL request from DbQuery and DbUpdate.

It provides also 2 extra Types supported by PostgreSQL:

- Set Type as an Array implementation (**PostgreStringArrayType**)
  - Set to Array shall be implemented carefully within the DTO class (see example)
- Map Type (String, String) as a Jsonb implementation (**PostgreStringMapAsJsonbType**)

Listing 29: Example code for **PostgreStringArrayType** and **PostgreStringMapAsJsonbType**

```
@Column(columnDefinition = "text[]", name = ARRAY1)
@Type(type = ARRAY_TYPE_CLASS)
private String[] array1;
/**
 * To get a Set internally instead of an array
 */
@IgnoreProperty
@Transient
private final Set<String> set1 = new HashSet<>();
/**
 * To ensure array ans set are correctly initialized
 */
@IgnoreProperty
@Transient
private boolean checked;
@Column(name = MAP1, columnDefinition = JSON_TYPE)
@Type(type = MAP_TYPE_CLASS)
private final Map<String, String> map1 = new HashMap<>();
```

2.5 Common Configuration

Several parts are concerned by the configuration.

Here are the global parameters, whatever the service.

2.5.1 `application.yaml` configuration

The following parameters are for optimization.

Table 1: Common Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.http.so-reuse-port`	`true`	Optimization on Linux/MacOs
`quarkus.http.tcp-cork`	`true`	Optimization on Linux
`quarkus.http.tcp-quick-ack`	`true`	Optimization on Linux
`quarkus.http.tcp-fast-open`	`true`	Optimization on Linux
`quarkus.vertx.prefer-native-transport`	`true`	Optimization for Various platforms
`quarkus.console` related		To control if the UI console should be activated or not

The following parameters are for Http service and client.

Table 2: Http Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.http.limits.max-body-size`	`5T`	Current limit of Cloud Storage providers
`quarkus.http.limits.max-chunk-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.http.limits.max-frame-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.resteasy-reactive.output-buffer-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.resteasy-reactive.input-buffer-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.rest-client.multipart.max-chunk-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.rest-client.max-chunk-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.vertx.eventbus.receive-buffer-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.vertx.eventbus.send-buffer-size`	`98304`	Best choice between 64K, 98K and 128K; See `ccs.bufferSize`
`quarkus.vertx.warning-exception-time`	`30S`	Extending from `2S`
`quarkus.vertx.max-event-loop-execute-time`	`30S`	Extending from `2S`

The following parameters are for TlS support.

Table 3: TLS Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.ssl.native`	`true`	Allow Native SSL support (OpenSSL)
`quarkus.http.ssl` related		To handle MTLS
`quarkus.rest-client.trust-store` `quarkus.rest-client.key-store`		To handle MTLS
`quarkus.http.host` and `quarkus.http.port/ssl-port`		To specify which host and port

The following parameters are for Log and Observability configuration.

Table 4: Log Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.http.access-log` related		To handle Access-log as usual http service
`quarkus.log.console.format`	`%d{HH:mm:ss,SSS} %-5p [%c{2.}] [%l] (%t) (%X) %s%e%n`	To adapt if necessary
`quarkus.log.console.json` and related		To activate with `quarkus-logging-json` module to get log in Json format
`quarkus.log.level`	`INFO`	To adapt as needed
`quarkus.otel` related		To configure OpenTelemetry for Metrics

Listing 30: Example of http access log configuration

```
quarkus.http.access-log.enabled=false
quarkus.http.record-request-start-time=true
quarkus.http.access-log.log-to-file=true
quarkus.http.access-log.base-file-name=quarkus-access-log
quarkus.http.access-log.pattern=%{REMOTE_HOST} %l %{REMOTE_USER} %{DATE_TIME} "%{REQUEST_LINE}" %{RESPONSE_CODE} %b (%
↪{RESPONSE_TIME} ms) [XOpIdIn: %{i,x-clonecloudstore-op-id} Client: "%{i,user-agent}"] [XOpIdOut: %{o,x-clonecloudstore-
↪op-id} Server: "%{o,server}"] [%{LOCAL_SERVER_NAME}]
```

The following parameters are for Traffic Shaping (bandwidth control) for Http service.

Table 5: Traffic Shaping Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.http.traffic-shaping` related		To enable traffic-shaping if needed (in particular with Replicator)

Listing 31: Example of http traffic-shaping configuration

```
quarkus.http.traffic-shaping.enabled=true
quarkus.http.traffic-shaping.inbound-global-bandwidth=1G
quarkus.http.traffic-shaping.outbound-global-bandwidth=1G
quarkus.http.traffic-shaping.max-delay=10s
quarkus.http.traffic-shaping.check-interval=10s
```

The following parameters are for Database configuration. Many options exist, and first, one should decide if MongoDB or PostgreSql is used (see `ccs.db.type`).

Table 6: Database Quarkus Configuration

Property/Yaml property	Default Value	Comment
`quarkus.hibernate-orm` related		For PostgreSql configuration
`quarkus.hibernate-orm.jdbc.statement-batch-size`	`50`	For bulk operation
`quarkus.hibernate-orm.jdbc.statement-fetch-size`	`1000`	For bulk operation
`quarkus.hibernate-orm.fetch.batch-size`	`1000`	For bulk operation
`quarkus.mongodb` related		For MongoDB configuration

Here are the specific global Clouod Clone Store parameters.

Table 7: Common Cloud Clone Store Configuration

Property/Yaml property	Possible Values	Default Value	Definition
`ccs.machineId`	Hexadecimal format of 6 bytes	Empty	Internal Machine Id used if specified (not null or empty) using 6 bytes in Hexadecimal format. Should be used in special case where MacAddress is not reliable
`ccs.bufferSize`	Any number of bytes > 8192	96 KB	Buffer Size ; Optimal is between 64KB, 96KB and 128KB. Note: Quarkus seems to limit to 64KB but setting the same value gives smaller chunk size
`ccs.maxWaitMs`	Any number of milliseconds (> 100 ms)	1 second	Property to define Max waiting time in milliseconds before Time Out within packets (in particular unknown size)
`ccs.driverMaxChu`	Any number > 5M in bytes	512 MB	Property to define Buffer Size for a Driver Chunk (may be override by driver specific configuration)
`ccs.server.computeSha2!`	Boolean	`false`	Property to define if Server will compute SHA 256 on the fly (should be true for Accessor)
`ccs.client.response.timeout`	Any number of milliseconds	6 minutes	Property to define Max transferring time in milliseconds before Time Out (must take into account large file and bandwidth)
`ccs.db.type`	mongo or postgre	Empty, so Mongo by default	Property to define which implementations to use between MongoDB or PostgreSQL

CHAPTER

THREE

ACCESSOR

3.1 BPMN for Accessor

3.1.1 Bucket

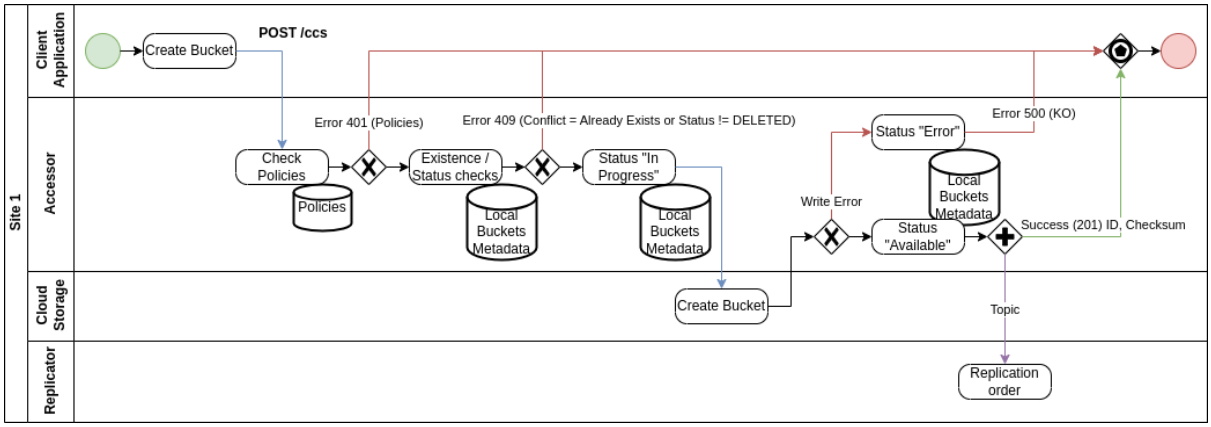

Fig. 1: Create Bucket

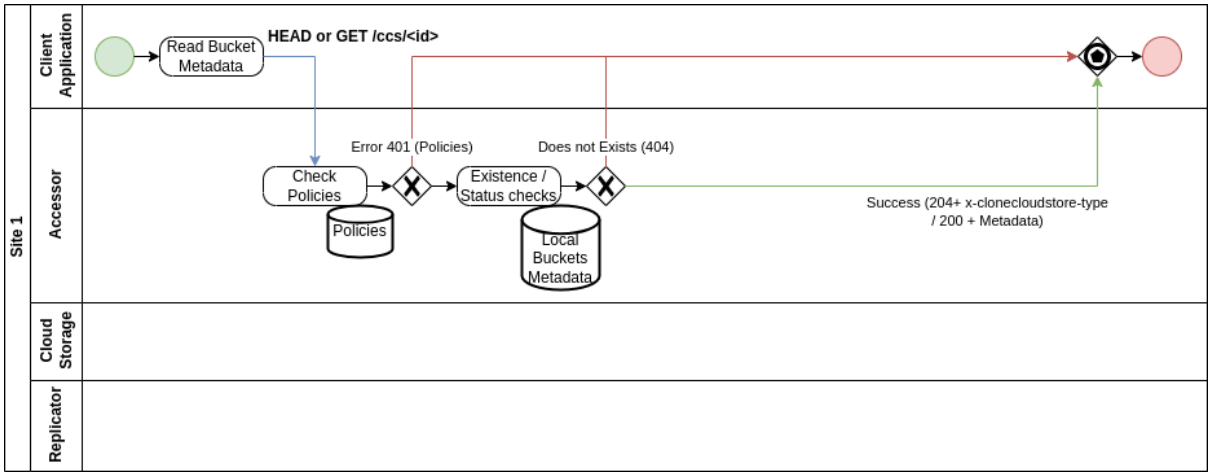

Fig. 2: Check Local Existence Bucket (GET for Metadata)

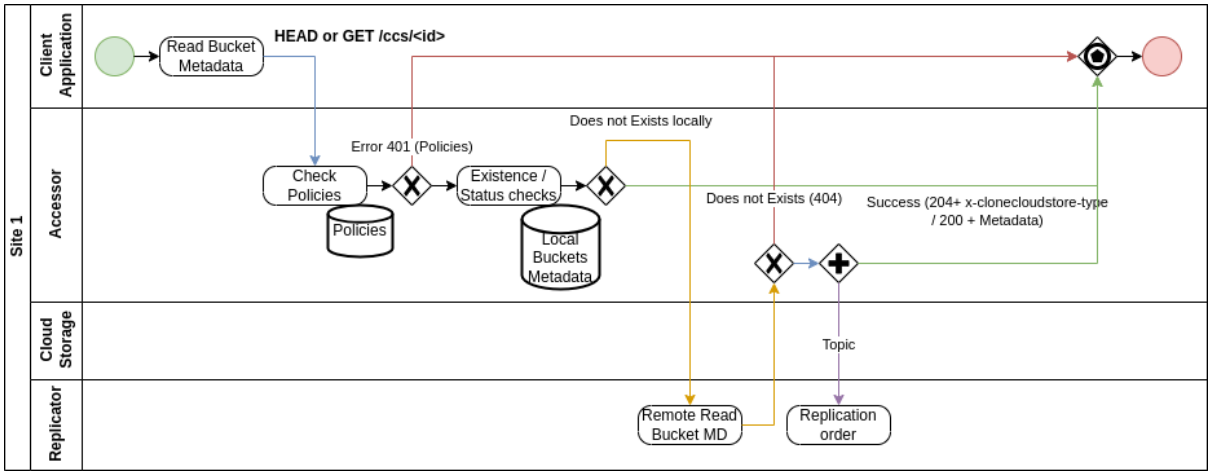

Fig. 3: Check Local/Remote Existence Bucket (GET for Metadata)

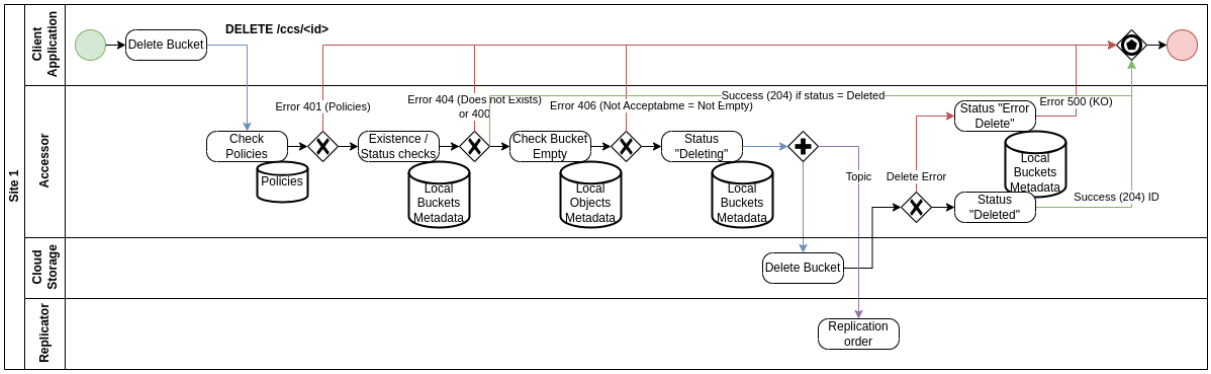

Fig. 4: Delete Bucket

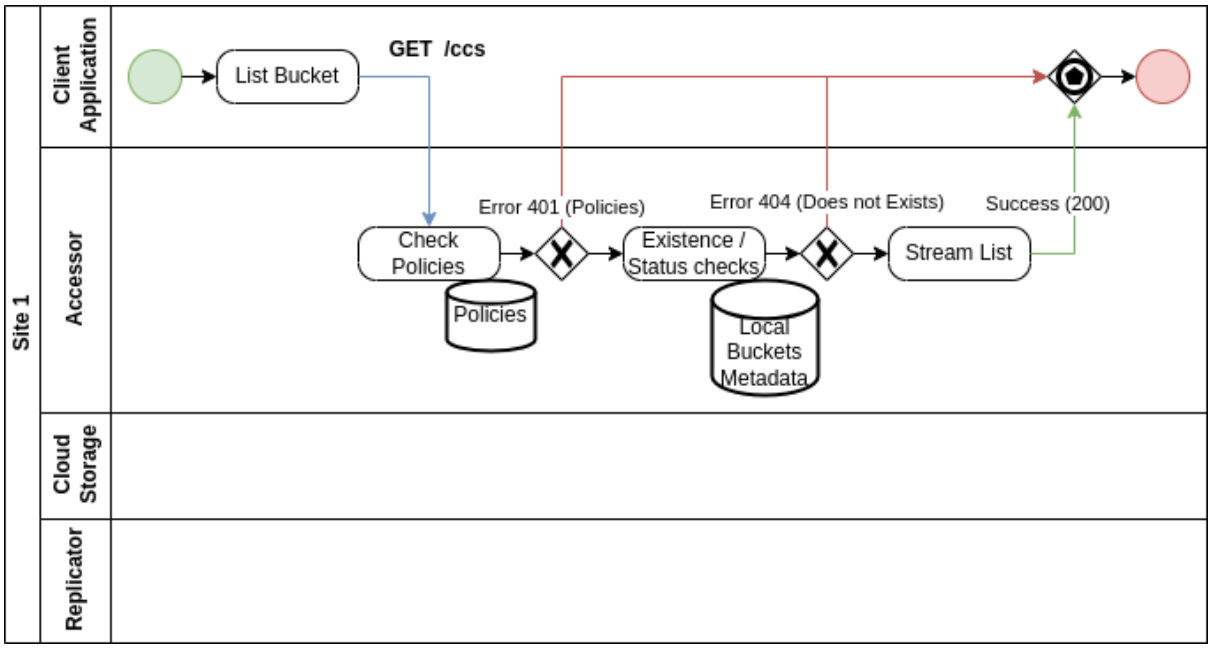

Fig. 5: List Buckets

3.1.2 Object

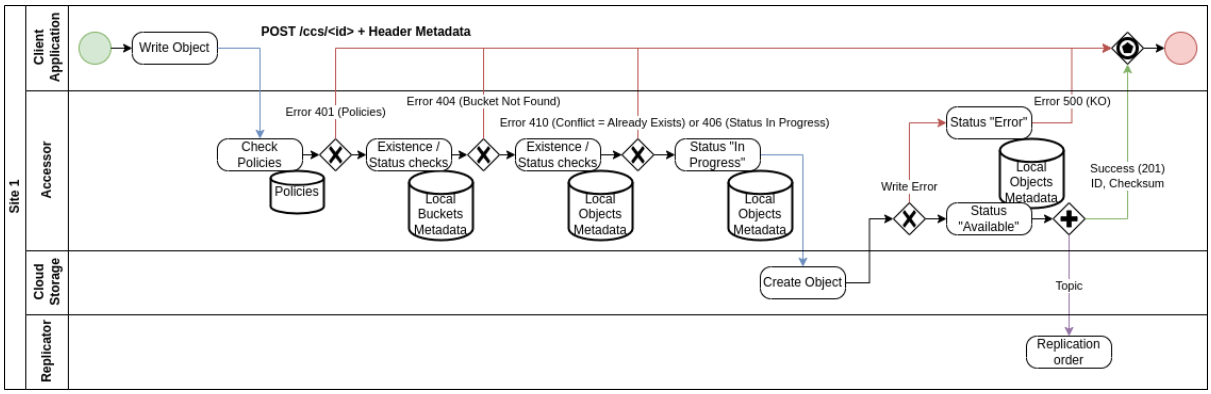

Fig. 6: Create Object

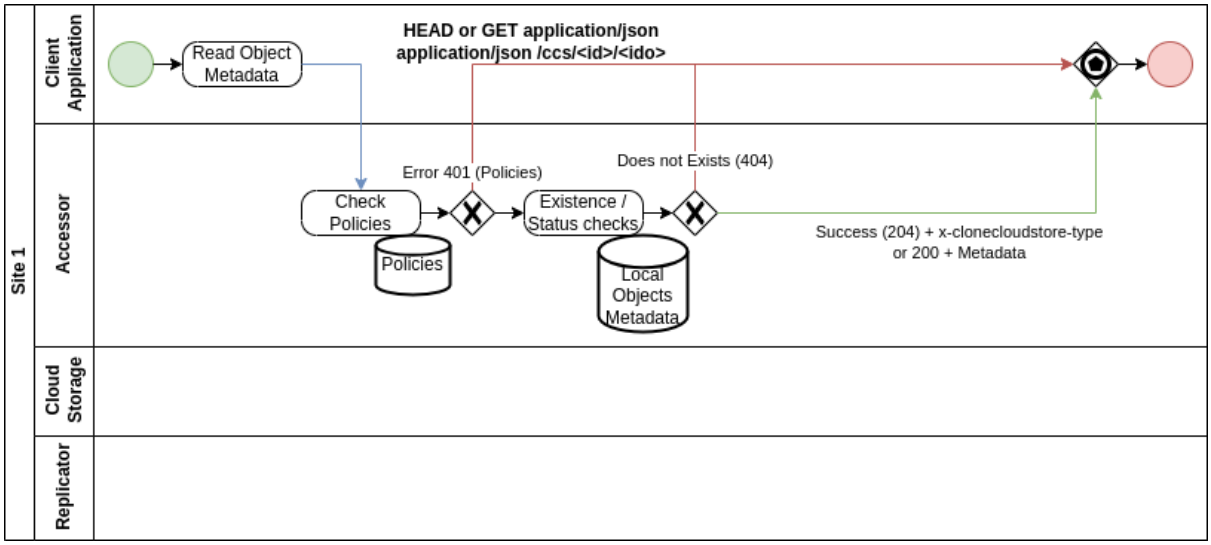

Fig. 7: Check Local Existence Object or GET Metadata

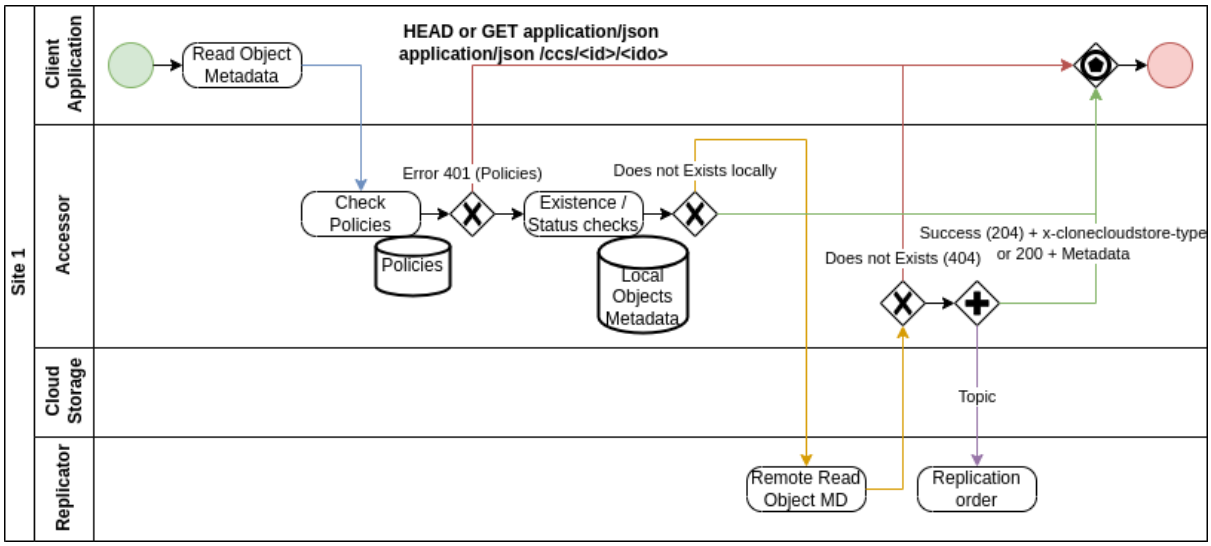

Fig. 8: Check Local/Remote Existence Object or GET Metadata

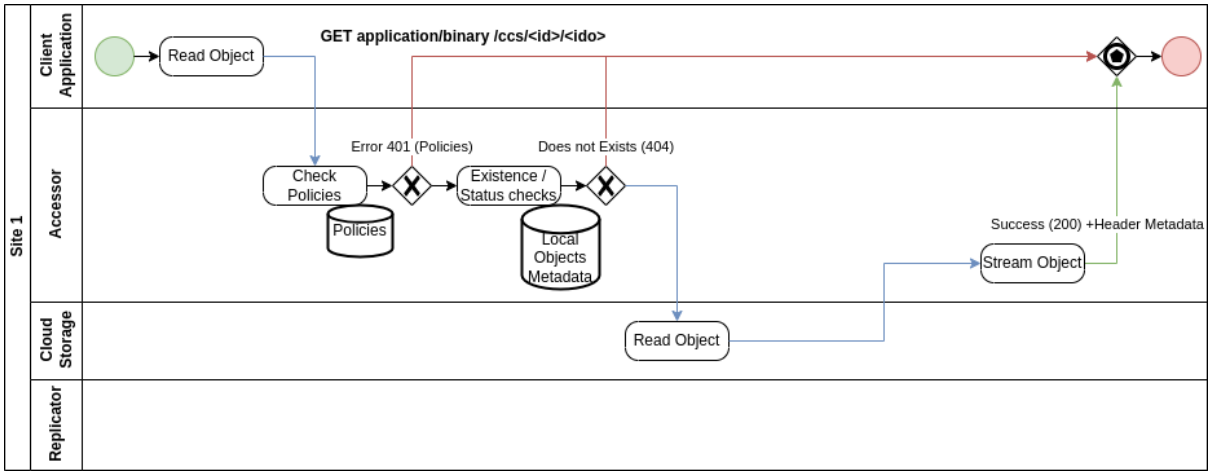

Fig. 9: Get Local Object's Content

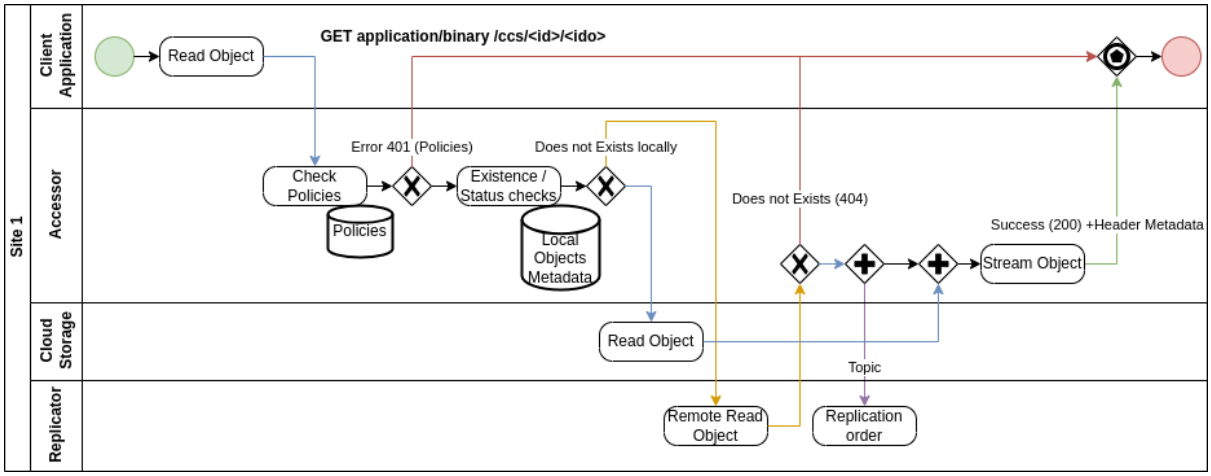

Fig. 10: Get Local/Remote Object's Content

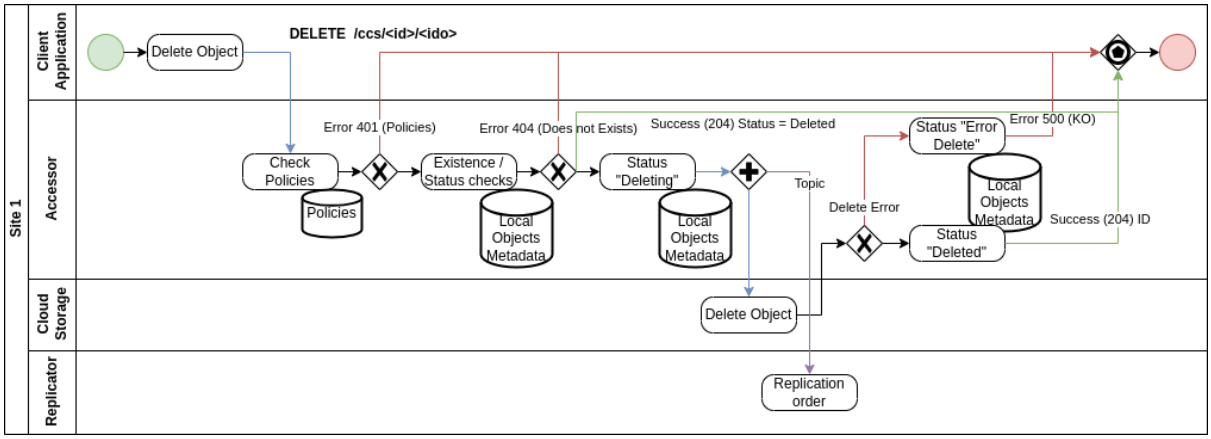

Fig. 11: Delete Object

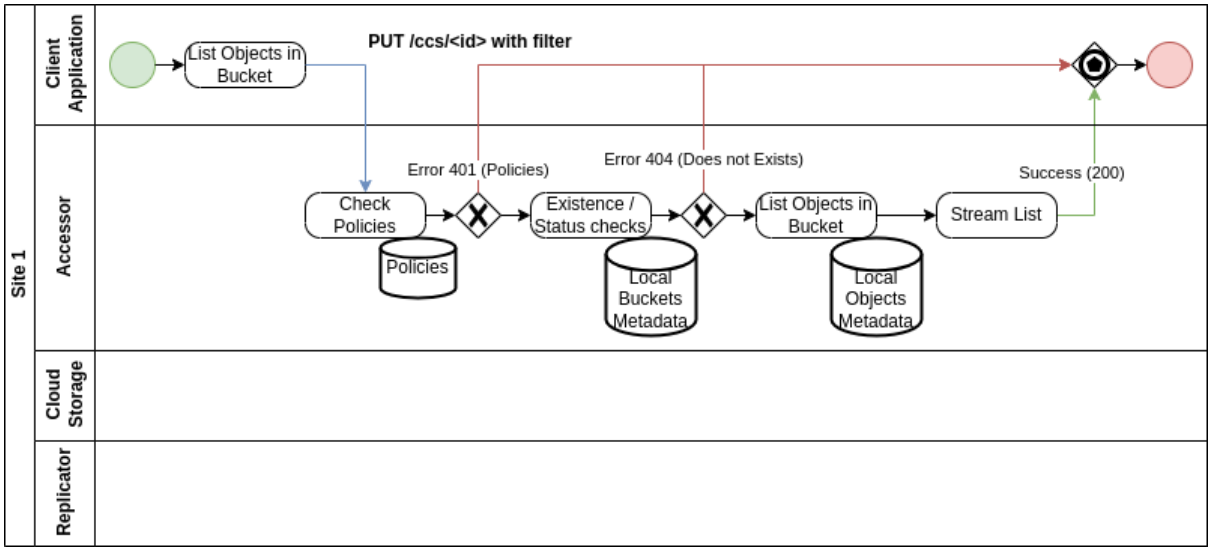

Fig. 12: List Objects in Bucket

3.1.3 Bucket Internal

Specific implementations for Internal Accessor:

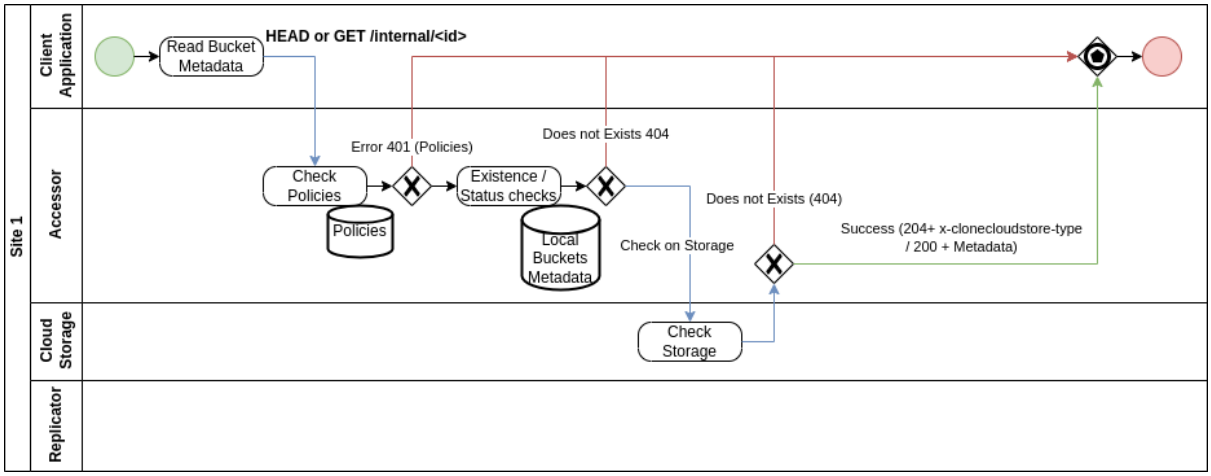

Fig. 13: Check Existence and Get Metadata for Local Bucket

3.1.4 Object Internal

Specific implementations for Internal Accessor:

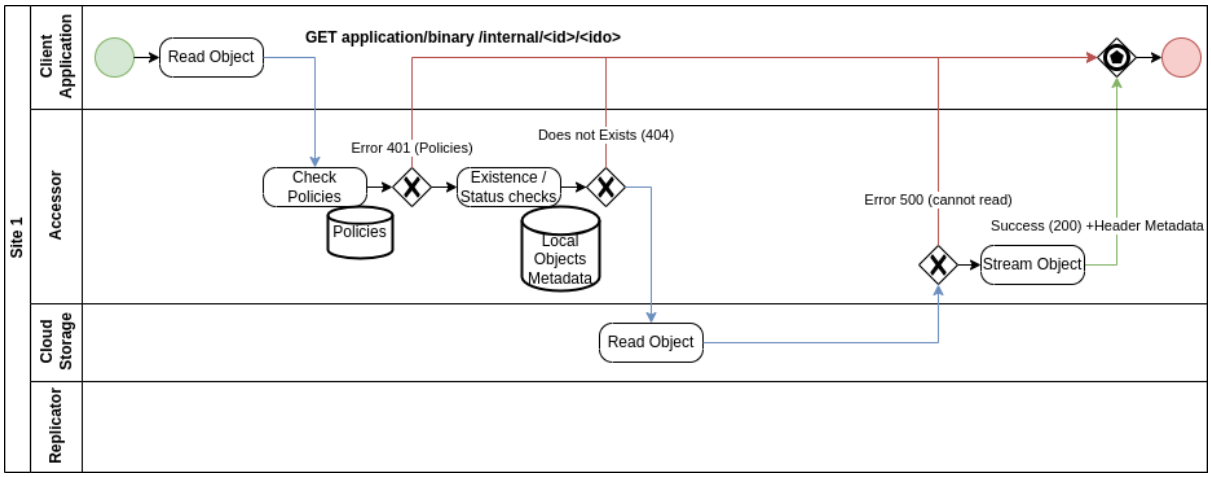

Fig. 14: Get Local Object's Content

3.2 Configuration

** TODO **

3.2.1 Various Accessor services

3.2.1.1 Accessor-Replicator

Used by remote request from Replicator. It listens to a Topic `replicator-action-in` from (local) Remote Replicator and is able, through the Local Replicator to access to remote Object if needed for local creation (clone) or to Accessor Internal Service.

This service has no API and only uses a Topic as incoming requests.

3.2.1.2 Accessor-Simple-Gateway

Simple Cloud Storage Gateway without any database and replication nor reconciliation support.

It is means for an easy move from existing storage to Cloud Clone Store, to later on apply the real Accessor service.

3.2.1.3 Accessor-Server

Used by application (clients) to interact with Cloud Storage, enabling replication and remote access and reconciliation, using the public API.

It is also used internally by all services when they need to access or interact with buckets and objects, through the internal API, which must be not available to other services than Cloud Clone Store itself.

3.2.2 application.yaml configuration

Table 1: Accessor Cloud Clone Store Client Configuration

Property/Yaml property	Possible Values	Default Value
`quarkus.rest-client."io.clonecloudstore.accessor.client.api.AccessorBucketApi".url`	Http(s) url of the service	
`quarkus.rest-client."io.clonecloudstore.accessor.client.api.AccessorObjectApi".url`	Http(s) url of the service	

Table 2: Accessor Cloud Clone Store Internal Client Configuration

Property/Yaml property	Possible Values	Default Value
`quarkus.rest-client."io.clonecloudstore.accessor.client.internal.api.AccessorBucketInternalApi".url`	Http(s) url of the service	
`quarkus.rest-client."io.clonecloudstore.accessor.client.internal.api.AccessorObjectInternalApi".url`	Http(s) url of the service	

Table 3: Accessor Replicator Cloud Clone Store Service Configuration

Property/Yaml property or Environment variable	Possible Values	Default Value
`ccs.accessor.site`	Name of the site	`unconfigured`
`ccs.accessor.internal.compression`	`true` or `false`, True to allow compression between services	`false`
Redefining `mp.messaging.incoming.replicator-action-in` or env `CCS_REQUEST_ACTION`	Name of the incoming topic for Action Requests	`request-action`
`quarkus.mongodb.database`	Name of the associated database (if MongoDB used, with ccs.db.type = mongo)	
`quarkus.rest-client."io.clonecloudstore.replicator.client.api.LocalReplicatorApi".url`	Http(s) url of the service	

Table 4: Accessor Cloud Clone Store Service Configuration

Property/Yaml property or Environment variable	Possible Values	Default Value
ccs.accessor.site	Name of the site	unconfigured
ccs.accessor.remote.read	true or false, True to allow remote access when object not locally found	false
ccs.accessor.remote.fixOnAbsent	true or false, True to allow to fix using remote accessed object	false
ccs.accessor.internal.compression	true or false, True to allow compression between services	false
Redefining mp.messaging.outgoing.replicator-action-out or env CCS_REQUEST_ACTION	Name of the outgoing topic for Action Requests	request-action
Redefining mp.messaging.outgoing.replicator-request-out or env CCS_REQUEST_REPLICATION	Name of the outgoing topic for Replication Requests	request-replication
quarkus.mongodb.database	Name of the associated database (if MongoDB used, with ccs.db.type = mongo)	
quarkus.rest-client."io.clonecloudstore.replicator.client.api.LocalReplicatorApi".url	Http(s) url of the service	

For both *Accessor Replicator Cloud Clone Store Service* and *Accessor Cloud Clone Store Service*, an extra configuration is needed according to the Storage Driver used:

- Note that maxPartSizeForUnknownLength should be defined according to memory available and concurrent access, as each transfer (upload or download) could lead to one buffer of this size for each.

Table 5: Driver for S3 Service Configuration

Property/Yaml property	Possible Values
ccs.driver.s3.host	S3 Host (do not use quarkus.s3.endpoint-override)
ccs.driver.s3.keyId	S3 KeyId (do not use quarkus.s3.aws.credentials.static-provider.access-key-id nor aws.accessKeyId)
ccs.driver.s3.key	S3 Key (do not use quarkus.s3.aws.credentials.secret-access-key nor aws.secretAccessKey)
ccs.driver.s3.region	S3 Region (do not use quarkus.s3.aws.region)
ccs.driver.s3.maxPartSize	MultiPart size (minimum 5 MB, maximum 5 GB, default 256 MB)
ccs.driver.s3.maxPartSizeFor	512 MB as in ccs.driverMaxChunkSize, MultiPart size (minimum 5 MB, maximum ~2 GB): will be used to buffer InputStream if length is unknown, so take care of the Memory consumption associated (512 MB, default, will limit the total InputStream length to 5 TB since 10K parts)

Table 6: Driver for Azure Blob Storage Service Configuration

Property/Yaml property	Possible Values
`quarkus.azure.storage.blob.connection-stri`	Connection String to Azure Blob Storage (see https://docs.quarkiverse.io/quarkus-azure-services/dev/index.html)
`ccs.driver.azure.maxConcurrency`	2, Maximum concurrency in upload/download with Azure Blob Storage
`ccs.driver.azure.maxPartSize`	256 MB, MultiPart size (minimum 5 MB, maximum 4 GB, default 256 MB)
`ccs.driver.azure.maxPartSizeForU`	512 MB as in `ccs.driverMaxChunkSize`, MultiPart size (minimum 5 MB, maximum ~2 GB): will be used to buffer InputStream if length is unknown, so take care of the Memory consumption associated (512 MB, default, will limit the total InputStream length to 25 TB since 50K parts)

Table 7: Driver for Google Cloud Storage Service Configuration

Property/Yaml property	Possible Values
`quarkus.google.cloud.project-id`	Project Id in Google Cloud (and related Authentication see https://docs.quarkiverse.io/quarkus-google-cloud-services/main/index.html)
`ccs.driver.google.disableGzip`	`true`, Default is to use Gzip content, but may be disabled (default: true so disabled)
`ccs.driver.google.maxPartSize`	256 MB, MultiPart size (minimum 5 MB, maximum 4 GB, default 256 MB) (**Property ignored**)
`ccs.driver.google.maxBufSize`	128 MB; MultiPart size (minimum 5 MB, maximum ~2 GB): will be used to buffer InputStream if length is unknown, so take care of the Memory consumption associated (128 MB, default)

Table 8: Accessor Simple Gateway Cloud Clone Store Service Configuration

Property/Yaml property	Possible Values	Default Value
`ccs.accessor.site`	Name of the site	`unconfigured`

3.3 Open API

3.3.1 Accessor Service

3.3.1.1 Internal API / Bucket

`GET /ccs/internal`

List all buckets in repository

List all buckets in repository

Example request:

```
GET /ccs/internal HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[1] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[2] – Bad Request
- 401 Unauthorized[3] – Unauthorized
- 500 Internal Server Error[4] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

GET /ccs/internal/{bucketName}

Get bucket metadata

Get bucket metadata

Parameters

- **bucketName** (*string*) –

Example request:

```
GET /ccs/internal/{bucketName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[5] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[6] – Bad Request
- 401 Unauthorized[7] – Unauthorized

[1] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[2] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[3] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[4] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- 404 Not Found[8] – Bucket not found
- 410 Gone[9] – Bucket deleted
- 500 Internal Server Error[10] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /ccs/internal/{bucketName}`

Check if bucket exist

Check if bucket exist (fullcheck true implies check down to Storage) and return BUCKET/NONE in header

Parameters

- **bucketName** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[11] – OK
- 400 Bad Request[12] – Bad Request
- 401 Unauthorized[13] – Unauthorized
- 404 Not Found[14] – Bucket not found
- 500 Internal Server Error[15] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

3.3.1.2 Internal API / Directory or Object

`PUT /ccs/internal/{bucketName}`

List objects from filter

List objects from filter as a Stream of Json lines

Parameters

- **bucketName** (*string*) –

Status Codes

[5] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[6] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[7] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[8] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[9] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[10] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[11] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[12] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[13] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[14] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[15] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- 200 OK[16] – OK
- 400 Bad Request[17] – Bad Request
- 401 Unauthorized[18] – Unauthorized
- 404 Not Found[19] – Bucket not found
- 500 Internal Server Error[20] – Internal Error

Request Headers

- **Accept-Encoding** – May contain ZSTD for compression
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-namePrefix** – Operation ID
- **x-clonecloudstore-statuses** – Operation ID
- **x-clonecloudstore-creationBefore** – Operation ID
- **x-clonecloudstore-creationAfter** – Operation ID
- **x-clonecloudstore-expiresBefore** – Operation ID
- **x-clonecloudstore-expiresAfter** – Operation ID
- **x-clonecloudstore-sizeLT** – Operation ID
- **x-clonecloudstore-sizeGT** – Operation ID
- **x-clonecloudstore-metadataEq** – Operation ID

`GET /ccs/internal/{bucketName}/{objectName}`

Get object

Get object binary with type application/octet-stream and get object metadata with type application/json

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Example request:

```
GET /ccs/internal/{bucketName}/{objectName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[21] – OK

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "bucket": "string",
    "name": "string",
    "hash": "string",
    "status": "UNKNOWN",
    "creation": "2024-01-07T19:08:21.868377",
```

(continues on next page)

[16] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[17] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[18] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[19] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[20] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

(continued from previous page)


```
    "expires": "2024-01-07T19:08:21.868377",
    "size": 1,
    "metadata": {}
}
```


- 400 Bad Request[22] – Bad Request
- 401 Unauthorized[23] – Unauthorized
- 404 Not Found[24] – Object not found
- 500 Internal Server Error[25] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **Accept-Encoding** – May contain ZSTD for compression

Response Headers

- **x-clonecloudstore-id** – Id
- **x-clonecloudstore-site** – Site
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-creation** – Creation Date
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash SHA-256
- **x-clonecloudstore-metadata** – Object Metadata
- **x-clonecloudstore-status** – Object Status
- **x-clonecloudstore-expires** – Expiration Date

`HEAD /ccs/internal/{bucketName}/{pathDirectoryOrObject}`

Check if object or directory exist

Check if object or directory exist (fullcheck true implies check down to Storage)

Parameters

- **bucketName** (*string*) –
- **pathDirectoryOrObject** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[26] – OK
- 400 Bad Request[27] – Bad Request
- 401 Unauthorized[28] – Unauthorized
- 500 Internal Server Error[29] – Internal Error

[21] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[22] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[23] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[24] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[25] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

3.3.1.3 Public API / Bucket

`GET /cloudclonestore`

List all buckets in repository

List all buckets in repository

Example request:

```
GET /cloudclonestore HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[30] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[31] – Bad Request
- 401 Unauthorized[32] – Unauthorized
- 500 Internal Server Error[33] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`GET /cloudclonestore/{bucketName}`

Get bucket metadata

Get bucket metadata

Parameters

- **bucketName** (*string*) –

Example request:

[26] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[27] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[28] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[29] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[30] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[31] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[32] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[33] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

```
GET /cloudclonestore/{bucketName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[34] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[35] – Bad Request
- 401 Unauthorized[36] – Unauthorized
- 404 Not Found[37] – Bucket not found
- 410 Gone[38] – Bucket deleted
- 500 Internal Server Error[39] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`POST /cloudclonestore/{bucketName}`

Create bucket

Create bucket in storage

Parameters

- **bucketName** (*string*) –

Status Codes

- 201 Created[40] – Bucket created

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[41] – Bad request
- 401 Unauthorized[42] – Unauthorized

[34] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[35] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[36] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[37] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[38] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[39] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- 409 Conflict[43] – Bucket already exist
- 500 Internal Server Error[44] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`DELETE /cloudclonestore/{bucketName}`

Delete bucket

Delete bucket in storage

Parameters

- **bucketName** (*string*) –

Status Codes

- 204 No Content[45] – Bucket deleted
- 400 Bad Request[46] – Bad Request
- 401 Unauthorized[47] – Unauthorized
- 404 Not Found[48] – Bucket not found
- 406 Not Acceptable[49] – Bucket found but not empty
- 410 Gone[50] – Bucket deleted
- 500 Internal Server Error[51] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /cloudclonestore/{bucketName}`

Check if bucket exist

Check if bucket exist and return BUCKET/NONE in header

Parameters

- **bucketName** (*string*) –

Status Codes

- 204 No Content[52] – OK
- 400 Bad Request[53] – Bad Request
- 401 Unauthorized[54] – Unauthorized
- 404 Not Found[55] – Bucket not found
- 500 Internal Server Error[56] – Internal Error

[40] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[41] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[42] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[43] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[44] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[45] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[46] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[47] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[48] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[49] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[50] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[51] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

3.3.1.4 Public API / Directory or Object

`PUT /cloudclonestore/{bucketName}`

List objects from filter

List objects from filter as a Stream of Json lines

Parameters

- **bucketName** (*string*) –

Status Codes

- 200 OK[57] – OK
- 400 Bad Request[58] – Bad Request
- 401 Unauthorized[59] – Unauthorized
- 404 Not Found[60] – Bucket not found
- 500 Internal Server Error[61] – Internal Error

Request Headers

- **x-clonecloudstore-namePrefix** – Filter based on name prefix
- **x-clonecloudstore-statuses** – Filter based on list of status
- **x-clonecloudstore-creationBefore** – Filter based on creation before
- **x-clonecloudstore-creationAfter** – Operation Filter based on creation after
- **x-clonecloudstore-expiresBefore** – Operation Filter based on expires before
- **x-clonecloudstore-expiresAfter** – Operation Filter based on expires after
- **x-clonecloudstore-sizeLT** – Operation Filter based on size less than
- **x-clonecloudstore-sizeGT** – Operation Filter based on size greater than
- **x-clonecloudstore-metadataEq** – Filter based on metadatata containing
- **Accept-Encoding** –
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

[52] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[53] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[54] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[55] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[56] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[57] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[58] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[59] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[60] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[61] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`GET /cloudclonestore/{bucketName}/{objectName}`

Get object

Get object binary with type application/octet-stream and get object metadata with type application/json

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Example request:

```
GET /cloudclonestore/{bucketName}/{objectName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[62] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "bucket": "string",
    "name": "string",
    "hash": "string",
    "status": "UNKNOWN",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "size": 1,
    "metadata": {}
}
```

- 400 Bad Request[63] – Bad Request
- 401 Unauthorized[64] – Unauthorized
- 404 Not Found[65] – Object not found
- 500 Internal Server Error[66] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **Accept-Encoding** – May contain ZSTD for compression

Response Headers

- **x-clonecloudstore-id** – Id
- **x-clonecloudstore-site** – Site
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-creation** – Creation Date
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash SHA-256
- **x-clonecloudstore-metadata** – Object Metadata
- **x-clonecloudstore-status** – Object Status
- **x-clonecloudstore-expires** – Expiration Date

`POST /cloudclonestore/{bucketName}/{objectName}`

Create object

Create object

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Status Codes

- 201 Created[67] – OK

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "bucket": "string",
    "name": "string",
    "hash": "string",
    "status": "UNKNOWN",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "size": 1,
    "metadata": {}
}
```

- 400 Bad Request[68] – Bad Request
- 401 Unauthorized[69] – Unauthorized
- 406 Not Acceptable[70] – Object already in creation
- 409 Conflict[71] – Conflict since Object already exist or invalid
- 500 Internal Server Error[72] – Internal Error

Request Headers

- **Content-Encoding** – May contain ZSTD for compression
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash
- **x-clonecloudstore-metadata** – Object Metadata as Json from Map<String,String>
- **x-clonecloudstore-expires** – Expiration Date
- **x-clonecloudstore-id** –
- **x-clonecloudstore-site** –

[62] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[63] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[64] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[65] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[66] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`DELETE /cloudclonestore/{bucketName}/{objectName}`

Delete object

Delete object

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Status Codes

- 204 No Content[73] – OK
- 400 Bad Request[74] – Bad Request
- 401 Unauthorized[75] – Unauthorized
- 404 Not Found[76] – Object not found
- 406 Not Acceptable[77] – Bucket is not empty
- 409 Conflict[78] – Conflict since Object status not compatible with Operation
- 410 Gone[79] – Object already deleted
- 500 Internal Server Error[80] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /cloudclonestore/{bucketName}/{pathDirectoryOrObject}`

Check if object or directory exist

Check if object or directory exist

Parameters

- **bucketName** (*string*) –
- **pathDirectoryOrObject** (*string*) –

Status Codes

- 204 No Content[81] – OK
- 400 Bad Request[82] – Bad Request
- 401 Unauthorized[83] – Unauthorized
- 500 Internal Server Error[84] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)

[67] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[68] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[69] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[70] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[71] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[72] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[73] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[74] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[75] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[76] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[77] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[78] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[79] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[80] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

3.3.2 Accessor Simple Gateway Service

3.3.2.1 Public API / Bucket

GET /cloudclonestore

List all buckets in repository

List all buckets in repository

Example request:

```
GET /cloudclonestore HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[85] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[86] – Bad Request
- 401 Unauthorized[87] – Unauthorized
- 500 Internal Server Error[88] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

GET /cloudclonestore/{bucketName}

Get bucket metadata

Get bucket metadata

Parameters

- **bucketName** (*string*) –

Example request:

[81] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[82] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[83] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[84] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[85] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[86] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[87] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[88] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

```
GET /cloudclonestore/{bucketName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[89] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[90] – Bad Request
- 401 Unauthorized[91] – Unauthorized
- 404 Not Found[92] – Bucket not found
- 410 Gone[93] – Bucket deleted
- 500 Internal Server Error[94] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`POST /cloudclonestore/{bucketName}`

Create bucket

Create bucket in storage

Parameters

- **bucketName** (*string*) –

Status Codes

- 201 Created[95] – Bucket created

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[96] – Bad request
- 401 Unauthorized[97] – Unauthorized

[89] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[90] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[91] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[92] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[93] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[94] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- 409 Conflict[98] – Bucket already exist
- 500 Internal Server Error[99] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`DELETE /cloudclonestore/{bucketName}`

Delete bucket

Delete bucket in storage

Parameters

- **bucketName** (*string*) –

Status Codes

- 204 No Content[100] – Bucket deleted
- 400 Bad Request[101] – Bad Request
- 401 Unauthorized[102] – Unauthorized
- 404 Not Found[103] – Bucket not found
- 406 Not Acceptable[104] – Bucket found but not empty
- 410 Gone[105] – Bucket deleted
- 500 Internal Server Error[106] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /cloudclonestore/{bucketName}`

Check if bucket exist

Check if bucket exist and return BUCKET/NONE in header

Parameters

- **bucketName** (*string*) –

Status Codes

- 204 No Content[107] – OK
- 400 Bad Request[108] – Bad Request
- 401 Unauthorized[109] – Unauthorized
- 404 Not Found[110] – Bucket not found
- 500 Internal Server Error[111] – Internal Error

[95] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[96] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[97] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[98] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[99] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[100] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[101] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[102] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[103] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[104] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[105] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[106] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

3.3.2.2 Public API / Directory or Object

`PUT /cloudclonestore/{bucketName}`

List objects from filter

List objects from filter as a Stream of Json lines

Parameters

- **bucketName** (*string*) –

Status Codes

- 200 OK[112] – OK
- 400 Bad Request[113] – Bad Request
- 401 Unauthorized[114] – Unauthorized
- 404 Not Found[115] – Bucket not found
- 500 Internal Server Error[116] – Internal Error

Request Headers

- **x-clonecloudstore-namePrefix** – Filter based on name prefix
- **x-clonecloudstore-statuses** – Filter based on list of status
- **x-clonecloudstore-creationBefore** – Filter based on creation before
- **x-clonecloudstore-creationAfter** – Operation Filter based on creation after
- **x-clonecloudstore-expiresBefore** – Operation Filter based on expires before
- **x-clonecloudstore-expiresAfter** – Operation Filter based on expires after
- **x-clonecloudstore-sizeLT** – Operation Filter based on size less than
- **x-clonecloudstore-sizeGT** – Operation Filter based on size greater than
- **x-clonecloudstore-metadataEq** – Filter based on metadatata containing
- **Accept-Encoding** –
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

[107] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[108] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[109] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[110] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[111] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[112] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[113] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[114] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[115] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[116] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`GET /cloudclonestore/{bucketName}/{objectName}`

Get object

Get object binary with type application/octet-stream and get object metadata with type application/json

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Example request:

```
GET /cloudclonestore/{bucketName}/{objectName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[117] – OK

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "id": "string",
      "site": "string",
      "bucket": "string",
      "name": "string",
      "hash": "string",
      "status": "UNKNOWN",
      "creation": "2024-01-07T19:08:21.868377",
      "expires": "2024-01-07T19:08:21.868377",
      "size": 1,
      "metadata": {}
  }
  ```

- 400 Bad Request[118] – Bad Request
- 401 Unauthorized[119] – Unauthorized
- 404 Not Found[120] – Object not found
- 500 Internal Server Error[121] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **Accept-Encoding** – May contain ZSTD for compression

Response Headers

- **x-clonecloudstore-id** – Id
- **x-clonecloudstore-site** – Site
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-creation** – Creation Date
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash SHA-256
- **x-clonecloudstore-metadata** – Object Metadata
- **x-clonecloudstore-status** – Object Status
- **x-clonecloudstore-expires** – Expiration Date

`POST /cloudclonestore/{bucketName}/{objectName}`

Create object

Create object

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Status Codes

- 201 Created[122] – OK

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "bucket": "string",
    "name": "string",
    "hash": "string",
    "status": "UNKNOWN",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "size": 1,
    "metadata": {}
}
```

- 400 Bad Request[123] – Bad Request
- 401 Unauthorized[124] – Unauthorized
- 406 Not Acceptable[125] – Object already in creation
- 409 Conflict[126] – Conflict since Object already exist or invalid
- 500 Internal Server Error[127] – Internal Error

Request Headers

- **Content-Encoding** – May contain ZSTD for compression
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash
- **x-clonecloudstore-metadata** – Object Metadata as Json from Map<String,String>
- **x-clonecloudstore-expires** – Expiration Date
- **x-clonecloudstore-id** –
- **x-clonecloudstore-site** –

[117] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[118] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[119] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[120] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[121] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`DELETE /cloudclonestore/{bucketName}/{objectName}`

Delete object

Delete object

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Status Codes

- 204 No Content[128] – OK
- 400 Bad Request[129] – Bad Request
- 401 Unauthorized[130] – Unauthorized
- 404 Not Found[131] – Object not found
- 406 Not Acceptable[132] – Bucket is not empty
- 409 Conflict[133] – Conflict since Object status not compatible with Operation
- 410 Gone[134] – Object already deleted
- 500 Internal Server Error[135] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /cloudclonestore/{bucketName}/{pathDirectoryOrObject}`

Check if object or directory exist

Check if object or directory exist

Parameters

- **bucketName** (*string*) –
- **pathDirectoryOrObject** (*string*) –

Status Codes

- 204 No Content[136] – OK
- 400 Bad Request[137] – Bad Request
- 401 Unauthorized[138] – Unauthorized
- 500 Internal Server Error[139] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)

[122] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[123] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[124] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[125] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[126] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[127] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[128] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[129] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[130] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[131] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[132] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.7
[133] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[134] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[135] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

[136] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[137] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[138] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[139] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

CHAPTER

FOUR

REPLICATOR

4.1 BPMN for Replicator

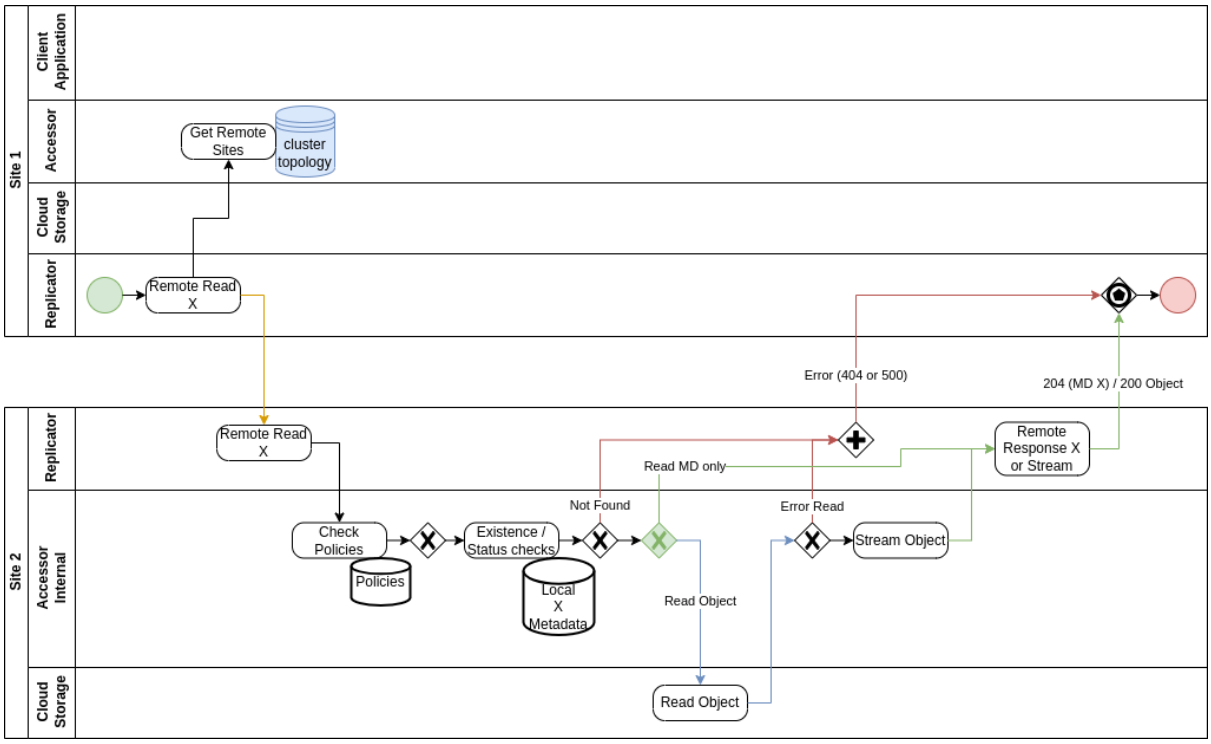

Fig. 1: Remote Read

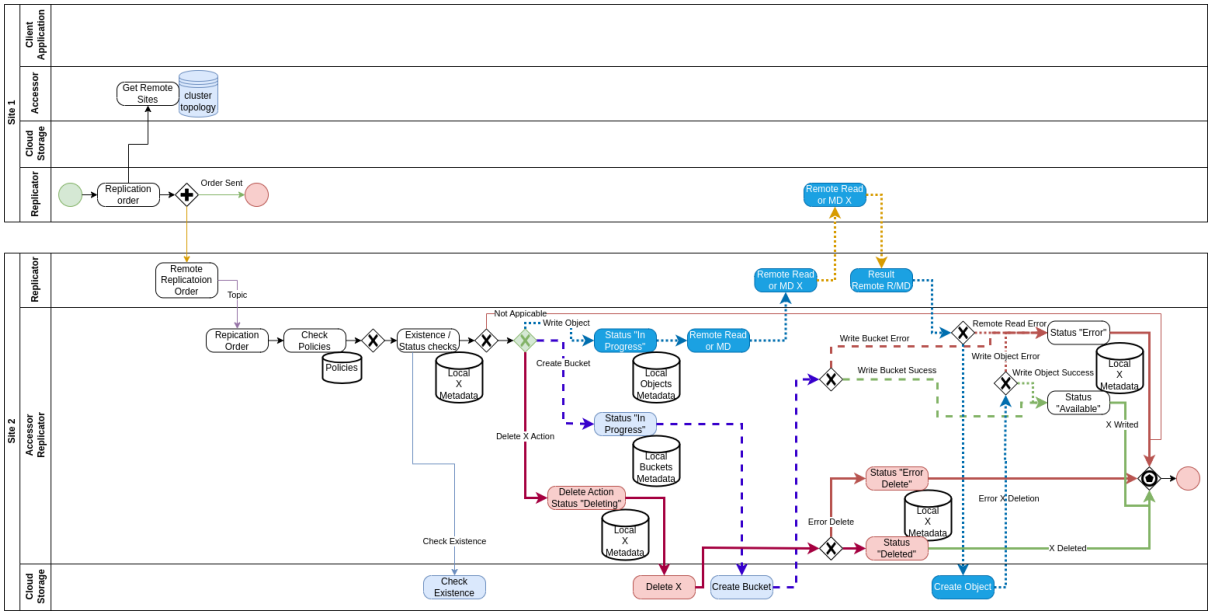

Fig. 2: Replication order

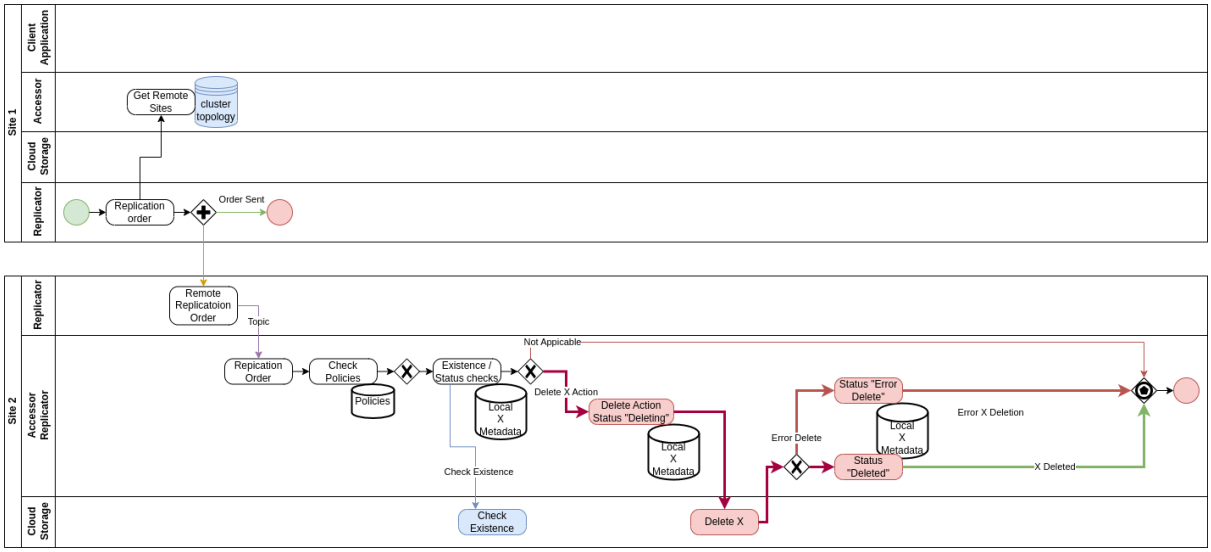

Fig. 3: Replication order for Delete

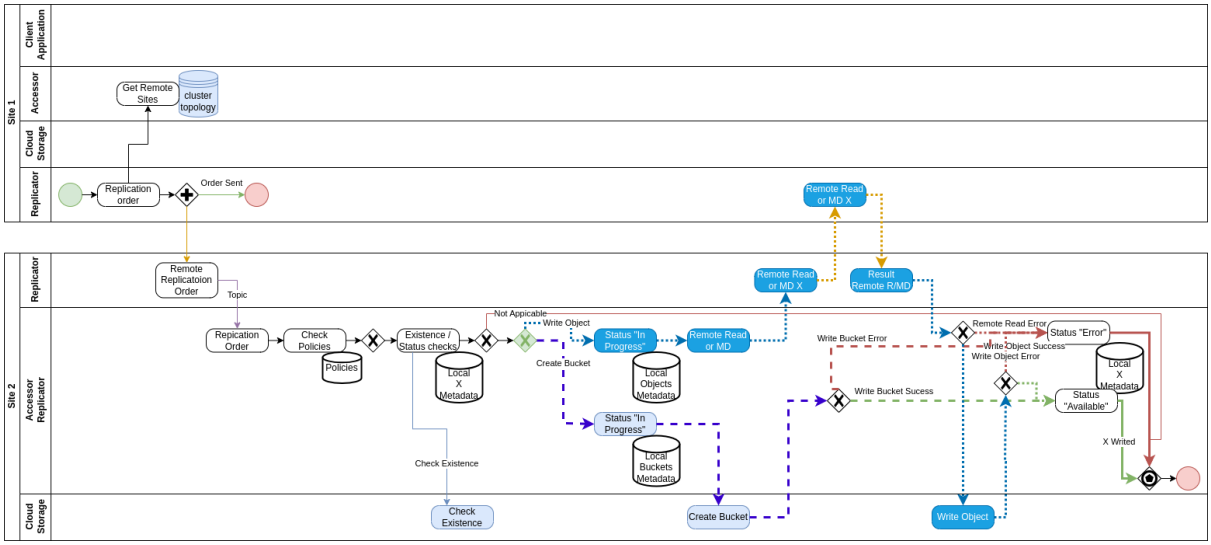

Fig. 4: Replication order for Create

4.2 Configuration

** TODO **

4.2.1 Various Replicator services

Both services run in the same server.

4.2.1.1 Local Replicator

It's role is to be contacted by local services to interact with remote services. - Through API for remote access, it proxies the request from Accessor Services (Accessor Public Service or Accessor Replicator Service) to the remote Replicator (remote site). - Through topic for Replication Requests, which are sent to remote replicator service through API.

So its role is to handle outgoing requests.

4.2.1.2 Remote Replicator

It's role is to handle remote requests: - Remote access through API goes to Accessor Internal Local service - Remote Replication Request are pushed into topic for Replication Actions. Those are handle then by the Accessor Replicator Local service.

So its role is to handle incoming requests.

4.2.2 application.yaml configuration

Table 1: Replicator Cloud Clone Store Client Configuration

Property/Yaml property or Environment variable	Possible Values	Default Value
`quarkus.rest-client."io.clonecloudstore.replicator.client.api.LocalReplicatorApi".url`	Http(s) url of the service	
Redefining `mp.messaging.outgoing.replicator-request-out` or env `CCS_REQUEST_REPLICATION`	Name of the outgoing topic for Replication Requests	`request-replication`

Table 2: Replicator Cloud Clone Store Service Configuration

Property/Yaml property or Environment variable	Possible Values	Default Value
`ccs.accessor.site`	Name of the site	`unconfigured`
`ccs.accessor.internal.compression`	`true` or `false`, True to allow compression between services	`false`
Redefining `mp.messaging.outgoing.replicator-action-out` or env `CCS_REQUEST_ACTION`	Name of the outgoing topic for Action Requests	`request-action`
Redefining `mp.messaging.outgoing.replicator-request-out` or env `CCS_REQUEST_REPLICATION`	Name of the incoming and outgoing topic for Replication Requests	`request-replication`
`quarkus.rest-client."io.clonecloudstore.accessor.client.internal.api.AccessorBucketInternalApi".url`	Http(s) url of the service	
`quarkus.rest-client."io.clonecloudstore.accessor.client.internal.api.AccessorObjectInternalApi".url`	Http(s) url of the service	
`quarkus.rest-client."io.clonecloudstore.replicator.server.remote.client.api.RemoteReplicatorApi".url`	Http(s) url of the remote service	
`quarkus.rest-client."io.clonecloudstore.topology.client.api.TopologyApi".url`	Http(s) url of the service	

4.3 Open API

4.3.1 Replicator/local

`GET /replicator/local/buckets/{bucketName}`

Get bucket metadata

Get bucket metadata

Parameters

- **bucketName** (*string*) –

Example request:

```
GET /replicator/local/buckets/{bucketName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[140] – OK

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "string",
    "site": "string",
    "name": "string",
    "creation": "2024-01-07T19:08:21.868377",
    "expires": "2024-01-07T19:08:21.868377",
    "status": "UNKNOWN"
}
```

- 400 Bad Request[141] – Bad Request
- 401 Unauthorized[142] – Unauthorized
- 404 Not Found[143] – Bucket not found
- 410 Gone[144] – Bucket deleted
- 500 Internal Server Error[145] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-target-id** – Target ID

HEAD /replicator/local/buckets/{bucketName}

Check if bucket exists on a remote replicator

Loops through the topology and search for a remote replicator owning the bucket

Parameters

- **bucketName** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[146] – OK
- 401 Unauthorized[147] – Unauthorized
- 404 Not Found[148] – Bucket not found
- 500 Internal Server Error[149] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-target-id** – Target ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

[140] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[141] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[142] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[143] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[144] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[145] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- **x-clonecloudstore-target-id** – Id of Remote Topology

`GET /replicator/local/buckets/{bucketName}/{objectName}`

Read Object from a remote replicator

Loops through topology and search for a remote replicator able to service the request. Open up a stream with remote replicator which reads from its local accessor

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Example request:

```
GET /replicator/local/buckets/{bucketName}/{objectName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[150] – OK
- 401 Unauthorized[151] – Unauthorized
- 404 Not Found[152] – Object not found
- 500 Internal Server Error[153] – Internal Error

Request Headers

- **Accept-Encoding** – May contain ZSTD for compression
- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-target-id** – Target ID
- **x-clonecloudstore-op-id** – Operation ID

`HEAD /replicator/local/buckets/{bucketName}/{pathDirectoryOrObject}`

Check if object exists on a remote replicator

Loops through the topology and search for a remote replicator owning the object

Parameters

- **bucketName** (*string*) –
- **pathDirectoryOrObject** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[154] – OK
- 401 Unauthorized[155] – Unauthorized
- 404 Not Found[156] – Object not found
- 500 Internal Server Error[157] – Internal Error

[146] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[147] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[148] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[149] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[150] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[151] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[152] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[153] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID
- **x-clonecloudstore-target-id** – Target ID

Response Headers

- **x-clonecloudstore-type** – Type as StorageType
- **x-clonecloudstore-target-id** – Id of Remote Topology

4.3.2 Replicator/remote

GET /replicator/remote/buckets/{bucketName}

Get bucket metadata

Get bucket metadata

Parameters

- **bucketName** (*string*) –

Example request:

```
GET /replicator/remote/buckets/{bucketName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[158] – OK

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "id": "string",
      "site": "string",
      "name": "string",
      "creation": "2024-01-07T19:08:21.868377",
      "expires": "2024-01-07T19:08:21.868377",
      "status": "UNKNOWN"
  }
  ```

- 400 Bad Request[159] – Bad Request
- 401 Unauthorized[160] – Unauthorized
- 404 Not Found[161] – Bucket not found
- 410 Gone[162] – Bucket deleted
- 500 Internal Server Error[163] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

[154] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[155] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[156] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[157] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`HEAD /replicator/remote/buckets/{bucketName}`

Check if bucket exists on a remote replicator

Loops through the topology and search for a remote replicator owning the bucket

Parameters

- **bucketName** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[164] – OK
- 401 Unauthorized[165] – Unauthorized
- 404 Not Found[166] – Bucket not found
- 500 Internal Server Error[167] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** –

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

`GET /replicator/remote/buckets/{bucketName}/{objectName}`

Read Object from a remote replicator

Loops through topology and search for a remote replicator able to service the request. Open up a stream with remote replicator which reads from its local accessor

Parameters

- **bucketName** (*string*) –
- **objectName** (*string*) –

Example request:

```
GET /replicator/remote/buckets/{bucketName}/{objectName} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[168] – OK
- 401 Unauthorized[169] – Unauthorized
- 404 Not Found[170] – Object not found
- 500 Internal Server Error[171] – Internal Error

Request Headers

- **Accept-Encoding** – May contain ZSTD for compression

[158] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[159] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[160] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[161] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[162] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.11
[163] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[164] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[165] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[166] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[167] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** – Operation ID

Response Headers

- **x-clonecloudstore-id** – Id
- **x-clonecloudstore-site** – Site
- **x-clonecloudstore-bucket** – Bucket Name
- **x-clonecloudstore-name** – Object Name
- **x-clonecloudstore-creation** – Creation Date
- **x-clonecloudstore-size** – Object Size
- **x-clonecloudstore-hash** – Object Hash SHA-256
- **x-clonecloudstore-metadata** – Object Metadata
- **x-clonecloudstore-status** – Object Status
- **x-clonecloudstore-expires** – Expiration Date

`HEAD /replicator/remote/buckets/{bucketName}/{pathDirectoryOrObject}`

Check if object exists on a remote replicator

Loops through the topology and search for a remote replicator owning the object

Parameters

- **bucketName** (*string*) –
- **pathDirectoryOrObject** (*string*) –

Query Parameters

- **fullCheck** (*boolean*) – If True implies Storage checking

Status Codes

- 204 No Content[172] – OK
- 401 Unauthorized[173] – Unauthorized
- 404 Not Found[174] – Object not found
- 500 Internal Server Error[175] – Internal Error

Request Headers

- **x-clonecloudstore-client-id** – Client ID (Required)
- **x-clonecloudstore-op-id** –

Response Headers

- **x-clonecloudstore-type** – Type as StorageType

[168] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[169] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[170] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[171] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[172] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[173] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[174] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[175] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

POST /replicator/remote/orders

Create order

Create replication order remotely

Example request:

```
POST /replicator/remote/orders HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "opId": "string",
    "fromSite": "string",
    "toSite": "string",
    "clientId": "string",
    "bucketName": "string",
    "objectName": "string",
    "size": 1,
    "hash": "string",
    "action": "CREATE"
}
```

Status Codes

- 201 Created[176] – Order created
- 400 Bad Request[177] – Bad request
- 401 Unauthorized[178] – Unauthorized
- 409 Conflict[179] – Bucket already exist
- 500 Internal Server Error[180] – Internal Error

POST /replicator/remote/orders/multiple

Create orders

Create replication orders remotely

Example request:

```
POST /replicator/remote/orders/multiple HTTP/1.1
Host: example.com
Content-Type: application/json

[
    {
        "opId": "string",
        "fromSite": "string",
        "toSite": "string",
        "clientId": "string",
        "bucketName": "string",
        "objectName": "string",
        "size": 1,
        "hash": "string",
        "action": "CREATE"
    }
]
```

Status Codes

- 201 Created[181] – Order created
- 400 Bad Request[182] – Bad request
- 401 Unauthorized[183] – Unauthorized
- 409 Conflict[184] – Bucket already exist

[176] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[177] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[178] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[179] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[180] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.5.1

- 500 Internal Server Error[185] – Internal Error

[181] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[182] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[183] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[184] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10
[185] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

CHAPTER

FIVE

RECONCILIATOR

5.1 BPMN for Reconciliator

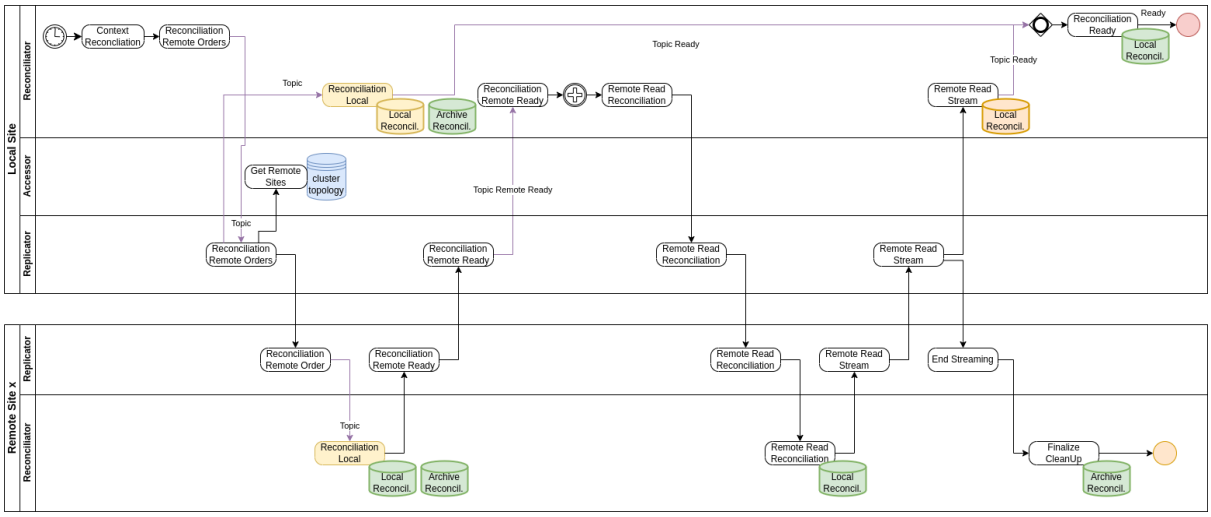

Fig. 1: Create context and Fusion local Reconciliation

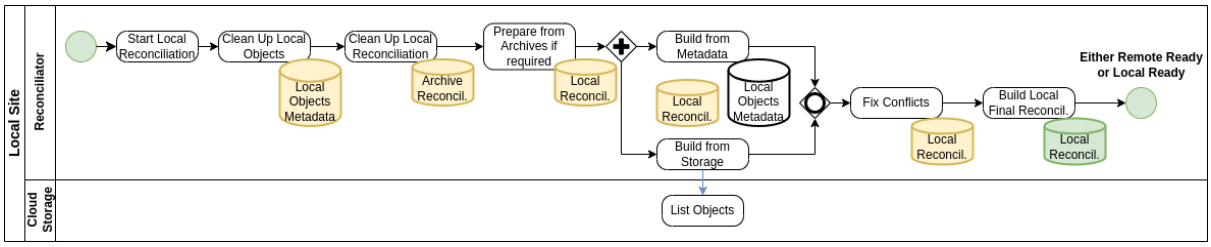

Fig. 2: Local Reconciliation

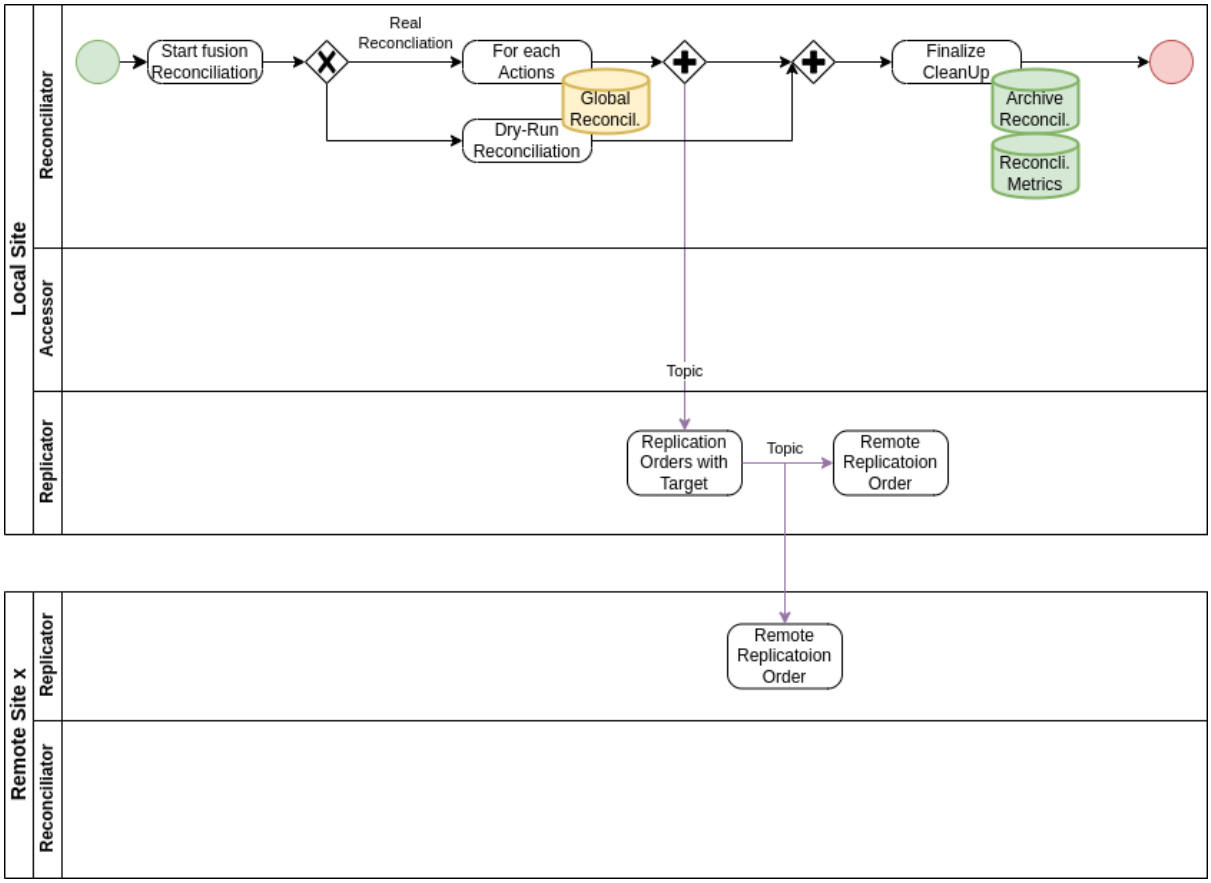

Fig. 3: Reconciliation Actions

5.2 Reconciliator's Algorithm

Still in progress

Table 1: Recurrent Purge on Expired date

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Purge	once	expired						
Object DB	•	•	•	•	•	x	•	•
Object Fix	•	•	•	•	•	Delete (check)	•	•
Purge	once	expired	if	archive	enabled			
Object DB	•	•	x	•	•	•	•	•
Object Fix	•	•	Archive (Move to another Bucket)	•	•	•	•	•

Table 2: Pre Reconciliation Purge

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Purge								
Object DB	x	•	•	•	•	•	•	•
Object Fix	Delete	•	•	•	•	•	•	•
Purge	status	older	than	reference				
Object DB	•	x	•	x	x	•	x	•
Object Fix	•	(check) >	X	< (check)	(check) >	X	< (check)	•

Table 3: Pre Result Reconciliation Purge

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Previous Result	•	x	x	x	x	x	x	x
Result Fix	Delete	x	x	Delete	x	Delete	Delete	x

Table 4: Fix LocalSite Reconciliation: Driver not DB

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Sites	•	•	x	•	•	•	•	•
Object Fix	•	•	X (to be updated)	•	•	•	•	•
Sites	•	•	•	•	•	•	•	X

Table 5: Fix LocalSite Reconciliation: DB not Driver with Available

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Sites	•	x	x	x	•	•	•	x
Object Fix	•	X	•	•	•	•	•	•
Sites	•	X	•	•	•	•	•	•

Table 6: Fix LocalSite Reconciliation: DB not Driver with Delete

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Sites	•	•	•	•	x	x	x	•
Object Fix	•	•	•	•	•	X	•	•
Sites	•	•	•	•	•	X	•	•

Table 7: Local to Remote Site Reconciliation: DB > Driver (date) with Available

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Object DB	•	x	x	x	•	•	•	x(M)
Object Driver	•	•	x	•	•	•	•	•
Object Fix	•	•	X	•	•	•	•	X(M if 7)
Sites	•	•	X	•	•	•	•	•

Table 8: Local to Remote Site Reconciliation: DB > Driver (date) with Delete

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Object DB	•	•	•	•	x	x	x	•
Object Driver	•	•	x	•	•	•	•	•
Object Fix	•	•	•	•	X	•	•	•
Sites	•	•	•	•	•	X	•	•

Table 9: Local to Remote Site Reconciliation: DB < Driver (date)

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Object DB	•	x	x	x	x	x	x	x(M)
Object Driver	•	•	x	•	•	•	•	•
Object Fix	•	•	X	•	•	•	•	X(M if 7)
Sites	•	•	X	•	•	•	•	•

Table 10: Remote to Action Reconciliation

From	Unknown	Upload	Ready	Err Upl	Deleting	Deleted	Err Del	To Update
Actions type 1	•	•	X	•	•	•	•	•
Object DB	x	x	•	x	x	x	x	x
Object Fix	•	•	X (S if Driver not exists + M else M if 1/7)	•	•	•	•	•
Actions type 2	•	•	X	•	•	•	•	•
Object DB	•	•	x	•	•	•	•	•
No Object Fix	•	•	•	•	•	•	•	•
Actions type 3	•	•	•	•	•	X	•	•
Object DB	x	x	x	x	x	x	x	x
Object Fix	•	•	•	•	•	X (no error on Driver)	•	•

5.3 Configuration

** TODO **

5.3.1 Various Reconciliation services

5.3.1.1 Remote Listing

5.3.1.2 Local Reconciliation

5.3.2 application.yaml configuration

5.4 Open API

** TODO **

5.4.1 default

`HEAD /reconciliator`

Status Codes

- 204 No Content[186] – OK

[186] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5

CHAPTER

SIX

ADMINISTRATION

6.1 BPMN for Administration

** TODO **

Among Administration services, there are:

- Topology serves the multi-sites topology for Replicator service.
- Client application identification
  - Currently not implemented, but could be based on MTLS or OIDC
- Managing Reconcilitation jobs

6.2 Configuration

** TODO **

6.2.1 Various Administration services

6.2.1.1 Topology

6.2.2 application.yaml configuration

Table 1: Topology Cloud Clone Store Client Configuration

Property/Yaml property	Possible Values	Default Value
`quarkus.rest-client."io.clonecloudstore.topology.client.api.TopologyApi".url`	Http(s) url of the service	

Table 2: Topology Cloud Clone Store Service Configuration

Property/Yaml property	Possible Values	Default Value
`ccs.accessor.site`	Name of the site	`unconfigured`
`quarkus.mongodb.database`	Name of the associated database (if MongoDB used, with ccs.db.type = mongo)	

6.3 Open API

6.3.1 Administration API / Topologies

`GET /replicator/topologies`

Get list of replicators

Get list of replicators in the topology

Query Parameters

- **status** (*string*) –

Example request:

```
GET /replicator/topologies HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[187] – Successfully retrieved list of topology

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  [
      {
          "id": "string",
          "name": "string",
          "uri": "string",
          "status": "UP"
      }
  ]
  ```

- 500 Internal Server Error[188] – Internal server error

`PUT /replicator/topologies`

Update a replicator in the topology

Update a replicator in the topology

Example request:

```
PUT /replicator/topologies HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "id": "string",
    "name": "string",
    "uri": "string",
    "status": "UP"
}
```

Status Codes

- 202 Accepted[189] – Successfully added topology
- 400 Bad Request[190] – Topology not valid
- 404 Not Found[191] – Topology not found
- 500 Internal Server Error[192] – Internal server error

[187] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[188] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`POST /replicator/topologies`

Add a replicator to topology

Add a replicator to topology

Example request:

```
POST /replicator/topologies HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "id": "string",
    "name": "string",
    "uri": "string",
    "status": "UP"
}
```

Status Codes

- 201 Created[193] – Successfully added topology
- 400 Bad Request[194] – Topology not valid
- 500 Internal Server Error[195] – Internal server error

`GET /replicator/topologies/{site}`

Get a replicator from topology

Get a replicator full details from topology based on its site

Parameters

- **site** (*string*) –

Example request:

```
GET /replicator/topologies/{site} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK[196] – Successfully retrieved topology

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "id": "string",
      "name": "string",
      "uri": "string",
      "status": "UP"
  }
  ```

- 400 Bad Request[197] – Topology id not valid
- 404 Not Found[198] – Topology not found
- 500 Internal Server Error[199] – Internal server error

[189] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.3
[190] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[191] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[192] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[193] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[194] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[195] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1
[196] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[197] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[198] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[199] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

`DELETE /replicator/topologies/{site}`

Delete a replicator from topology

Delete a replicator from topology

Parameters

- **site** (*string*) –

Status Codes

- 204 No Content[200] – Successfully deleted topology
- 400 Bad Request[201] – Topology id not valid
- 404 Not Found[202] – Topology not found
- 500 Internal Server Error[203] – Internal server error

[200] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[201] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[202] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[203] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1

CHAPTER

SEVEN

OBJECT STORAGE DRIVER

7.1 Driver API

The Driver API is the core of integration of multiple Object Storage solutions. Each integration must implement this interface in order to be able to add this as a plugin within Cloud Cloud Store.

7.1.1 Global logic of API

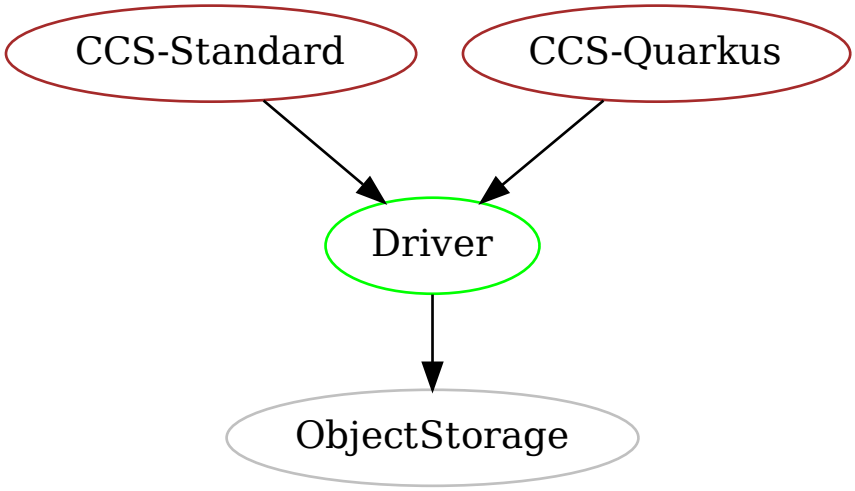

Fig. 1: Relation between Cloud Cloud Store, Driver and Object Storage

7.1.2 3 implementations

There are 3 implementations:

- S3 like support (whatever Amazon, Minio or any S3 compatible implementations)
- Azure Blob Storage support
- Google Cloud Storage support

All of these implementations respect the API of the Driver.

7.1.3 Driver API details

The DriverApiFactory might be created by explicitly create the right implementation (for instance `Arc.container().select(DriverApiFactory.class).get();`).

The DriverApi is then created through this Factory with `factory.getInstance()`.

However, each DriverApiFactory must register once configured within DriverApiRegister, such that the factory is get using `DriverApiRegister.getDriverApiFactory()`.

Note that all methods accept both String for Bucket / Object and StorageBucket / StorageObject, except `bucketCreate` and `objectPrepareCreateInBucket` since those methods need a full StorageBucket or StorageObject.

For instance: `bucketDelete(String bucket)` can be written as `bucketDelete(StorageBucket bucket)`, or `objectGetInputStreamInBucket(String bucket, String object)` as `objectGetInputStreamInBucket(StorageObject object)`

7.1.3.1 Bucket operations

Listing 1: Java API for **Buckets**

```
/**
 * Count Buckets
 */
long bucketsCount() throws DriverException;

/**
 * List Buckets
 */
Stream<StorageBucket> bucketsList() throws DriverException;
```

Listing 2: Java API for **Bucket**

```
/**
 * Create one Bucket and returns it
 *
 * @param bucket contains various information that could be implemented within Object Storage, but, except the name
 *               of the bucket, nothing is mandatory
 * @return the StorageBucket as instantiated within the Object Storage (real values)
 */
StorageBucket bucketCreate(StorageBucket bucket)
    throws DriverNotAcceptableException, DriverAlreadyExistException, DriverException;

/**
 * Delete one Bucket if it exists and is empty
 */
void bucketDelete(String bucket) throws DriverNotAcceptableException, DriverNotFoundException, DriverException;

/**
 * Check existence of Bucket
 */
boolean bucketExists(String bucket) throws DriverException;
```

7.1.3.2 Object operations

Listing 3: Java API for **Objects**

```
/**
 * Count Objects in specified Bucket
 */
long objectsCountInBucket(final String bucket) throws DriverNotFoundException, DriverException;

/**
 * Count Objects in specified Bucket with filters (all optionals)
 */
long objectsCountInBucket(String bucket, String prefix, Instant from, Instant to)
    throws DriverNotFoundException, DriverException;

/**
 * List Objects in specified Bucket.
```

(continues on next page)

(continued from previous page)

```
 */
Stream<StorageObject> objectsListInBucket(String bucket)
    throws DriverNotFoundException, DriverException;

/**
 * List Objects in specified Bucket with filters (all optionals)
 */
Stream<StorageObject> objectsListInBucket(String bucket, String prefix, Instant from, Instant to)
    throws DriverNotFoundException, DriverException;
```

Listing 4: Java API for **Object**

```
/**
 * Check if Directory or Object exists in specified Bucket (based on prefix)
 */
StorageType directoryOrObjectExistsInBucket(final String bucket, final String directoryOrObject)
    throws DriverException;

/**
 * First step in creation of an object within a Bucket. The InputStream is ready to be read in
 * a concurrent independent thread to be provided by the driver. Sha256 might be null or empty. Len might be 0,
 * meaning unknown.
 *
 * @param object contains various information that could be implemented within Object Storage, but, except the name
 *               of the bucket and the key of the object, nothing is mandatory
 */
void objectPrepareCreateInBucket(StorageObject object, InputStream inputStream)
    throws DriverNotFoundException, DriverAlreadyExistException, DriverException;

/**
 * Second step in creation of an object within a Bucket. Sha256 might be null or empty. Reallen must not be 0.
 * This method waits for the prepare method to end and returns the final result.
 *
 * @return the StorageObject as instantiated within the Object Storage (real values)
 */
StorageObject objectFinalizeCreateInBucket(String bucket, String object, long realLen, String sha256)
    throws DriverNotFoundException, DriverAlreadyExistException, DriverException;

/**
 * Get the content of the specified Object within specified Bucket
 */
InputStream objectGetInputStreamInBucket(String bucket, String object) throws DriverNotFoundException,
    DriverException;

/**
 * Get the Object metadata from this Bucket (those available from Object Storage)
 */
StorageObject objectGetMetadataInBucket(String bucket, String object)
    throws DriverNotFoundException, DriverException;

/**
 * Delete the Object from this Bucket
 */
void objectDeleteInBucket(String bucket, String object)
    throws DriverNotAcceptableException, DriverNotFoundException, DriverException;
```

CHAPTER

EIGHT

DEV DETAIL

8.1 POM Version management

In order to maintain as much as possible the simplicity and compatibility, here are some rules:

- Version is defined statically in the highest Pom (also named pom parent but in the same project)
- version is defined statically in all sub pom (child poms) for reference to Parent
- In highest POM (parent pom of the project):
  - Use `dependencyManagement` to define all modules version from this project, but use as version ${project.version}
    - Place at first in this management the quarkus.platform pom with scope import
  - If needed, you can add extra dependencies there, to specify version in top Pom
- In sub POM, you shall define real `dependencies` this time, but with no version, since they shall be managed by the parent POM

To update the version of all project, use the **versions-maven-plugin**:

Listing 1: Example for **versions-maven-plugin**

```
mvn versions:set -DnewVersion=x.y.z-SNAPSHOT
mvn versions:set -DnewVersion=x.y.z
mvn versions:revert
mvn versions:commit
```

After using *versions:set* command, you can check if the result is correct (for instance using `git diff`).

If OK, then commit it (it will simply remove the backup file).

If KO, then revert and redo (it will replace the current pom with the backup one).

Note that it will update all modules recursively in the project.

8.2 Full Build on local

Use the `-P benchmark` to allow to run benchmark tests, place in IT tests.

Note that current implementation changes most of other real IT tests to ITTest tests, such that they are launched event without this profile `benchmark`. The main reasons are: - Most of those tests are really “simple” IT tests, meaning they are part of Junit tests. - Aggregation of coverage is easier for those non IT tests

In the same idea, if the CI/CD does support the Sphinx process to build the HTML and PDF documentations, the profile `-P doc` can be included.

In order to launch them locally, you have to do the following:

Listing 2: Example for **maven with Doc generation**

```
mvn verify
mvn package -P doc
```

You can launch only with container (implying all tests plus the one that are using IT name but QuarkusTest, not QuarkusIntegrationTest), using only **verify** phase.

You can launch only documentation generation, using **package** phase (documentation is build on pre-package phase).

8.2.1 How to integrate Containers in Quarkus tests

By default, if any dependencies use containers for testing (Quarkus Dev Support), it will be launch on each and every tests. Most of the time, that is not an issue, but in some cases we do want to control when a container is launched or not.

Moreover, if multiple "containers" are defined in the dependencies, they will all be launched for each and every tests, even if not needed.

So the following is an option to remove those constraints and still being able to launch tests with or without explicit container(s).

Of course, if using default Dev services from Quarkus is not an issue, you can still rely on it and therefore ignore the following.

Full examples are available within ccs-test-support test sources.

8.2.1.1 Properties

Add the following to your application.properties for test (in `src/test/resources`):

Listing 3: Example for **properties** for Dev Containers

```
# Global stop (needed to prevent Ryuk to be launched)
quarkus.devservices.enabled=false
# Below according to what is used in the tests
# Particular stop for S3
quarkus.s3.devservices.enabled=false
# Particular stop for database (when not using PostgreSQL but MongoDB, set to true if reversed)
quarkus.hibernate-orm.enabled=false
#DO NOT SET THIS: quarkus.hibernate-orm.database.generation = drop-and-create
```

8.2.1.2 Handling startup of containers

The idea is to launch the container as needed and only when needed.

The following example is for S3.

8.2.1.2.1 Use QuarkusTestResourceLifecycleManager and QuarkusTestProfile

`QuarkusTestResourceLifecycleManager` is intended to provide manual control on resources needed before the Quarkus test startups. (see Quarkus TESTING YOUR APPLICATION / Starting services before the Quarkus application starts https://quarkus.io/guides/getting-started-testing#quarkus-test-resource)

2 kinds of `QuarkusTestResourceLifecycleManager` can be done.

8.2.1.2.1.1 For no container at all

Listing 4: Example for **NoResource** for no container

```
public class NoResource implements QuarkusTestResourceLifecycleManager {
  @Override
  public Map<String, String> start() {
    return SingletonUtils.singletonMap();
  }

  @Override
  public void stop() {
    // Nothing
  }
}
```

8.2.1.2.1.2 For a real container

2 classes are needed, one for the Resource, one for the Container using TestContainers.

The first one defines the Resource to be used and launched before Quarkus starts (mandatory).

Listing 5: Example for **MinioResource** for Minio S3 container

```
public class MinIoResource implements QuarkusTestResourceLifecycleManager {
  private static final String ACCESS_KEY = "accessKey";
  private static final String SECRET_KEY = "secretKey";
  public static MinioContainer minioContainer =
      new MinioContainer(new MinioContainer.CredentialsProvider(ACCESS_KEY, SECRET_KEY));

  public static String getAccessKey() {
    return minioContainer.getAccessKey();
 }

  public static String getSecretKey() {
    return minioContainer.getSecretKey();
  }

  public static String getUrlString() {
    return minioContainer.getUrlString();
  }

  public static String getRegion() {
    return Regions.EU_WEST_1.name();
  }

  @Override
  public Map<String, String> start() {
    if (!minioContainer.isRunning()) {
      minioContainer.start();
    }
    return minioContainer.getEnvMap();
  }

  @Override
  public void stop() {
    minioContainer.stop();
  }
}
```

The second one defines the container to start (using here TestContainers).

Listing 6: Example for **MinioContainer** for Minio S3 container

```
public class MinioContainer extends GenericContainer<MinioContainer> {
  private static final int DEFAULT_PORT = 9000;
  private static final String DEFAULT_IMAGE = "minio/minio";

  private static final String MINIO_ACCESS_KEY = "MINIO_ACCESS_KEY";
  private static final String MINIO_SECRET_KEY = "MINIO_SECRET_KEY";

  private static final String DEFAULT_STORAGE_DIRECTORY = "/data";
  private static final String HEALTH_ENDPOINT = "/minio/health/ready";

  public MinioContainer(final CredentialsProvider credentials) {
    this(DEFAULT_IMAGE, credentials);
```

(continues on next page)

(continued from previous page)

```
  }

  public MinioContainer(final String image, final CredentialsProvider credentials) {
    super(image == null ? DEFAULT_IMAGE : image);
    withNetworkAliases("minio-" + Base58.randomString(6));
    withExposedPorts(DEFAULT_PORT);
    if (credentials != null) {
      withEnv(MINIO_ACCESS_KEY, credentials.getAccessKey());
      withEnv(MINIO_SECRET_KEY, credentials.getSecretKey());
    }
    withCommand("server", DEFAULT_STORAGE_DIRECTORY);
    setWaitStrategy(new HttpWaitStrategy().forPort(DEFAULT_PORT).forPath(HEALTH_ENDPOINT)
        .withStartupTimeout(Duration.ofMinutes(2)));
  }

  public URL getURL() throws MalformedURLException {
    return new URL(getUrlString());
  }

  public String getUrlString() {
    return "http://" + getHost() + ":" + getMappedPort(DEFAULT_PORT);
  }

  public String getAccessKey() {
    return getEnvMap().get(MINIO_ACCESS_KEY);
  }

  public String getSecretKey() {
    return getEnvMap().get(MINIO_SECRET_KEY);
  }

  public static class CredentialsProvider {
    private final String accessKey;
    private final String secretKey;

    public CredentialsProvider(final String accessKey, final String secretKey) {
      this.accessKey = accessKey;
      this.secretKey = secretKey;
    }

    public String getAccessKey() {
      return accessKey;
    }

    public String getSecretKey() {
      return secretKey;
    }
  }
}
```

8.2.1.2.1.3 QuarkusTestProfile

Once build, the recommended way is to use a QuarkusTestProfile.

Listing 7: Example for **NoResourceProfile** for no Dev services

```
public class NoResourceProfile  implements QuarkusTestProfile {
  @Override
  public Map<String, String> getConfigOverrides() {
    return Map.of(ResourcesConstants.QUARKUS_DEVSERVICES_ENABLED, "false");
  }

  @Override
  public boolean disableGlobalTestResources() {
    return true;
  }

  @Override
  public String getConfigProfile() {
    return "test-noresource";
  }
}
```

Listing 8: Example for **MinioProfile** for Minio S3 container

```
public class MinioProfile  implements QuarkusTestProfile {
  @Override
```

(continues on next page)

(continued from previous page)

```
  public Map<String, String> getConfigOverrides() {
    return Map.of(ResourcesConstants.QUARKUS_DEVSERVICES_ENABLED, "false");
  }

  @Override
  public boolean disableGlobalTestResources() {
    return true;
  }

  @Override
  public String getConfigProfile() {
    return "test-minio";
  }

  @Override
  public List<TestResourceEntry> testResources() {
    return Collections.singletonList(new TestResourceEntry(MinIoResource.class));
  }
}
```

Special attention on Database support: In order to be able to choose between PostgreSql implementation or MongoDB implementation at runtime, the following properties are needed additionally:

Listing 9: Additional properties for PostgreSql/MongoDb support at runtime

```
@Override
public Map<String, String> getConfigOverrides() {
  return Map.of(ResourcesConstants.QUARKUS_DEVSERVICES_ENABLED, "false",
      // Specify false for Mnngo, True for Postgre
      ResourcesConstants.QUARKUS_HIBERNATE_ORM_ENABLED, "false",
      // Specify MONGO for Mnngo, POSTGRE for Postgre
      ResourcesConstants.CCS_DB_TYPE, ResourcesConstants.MONGO);
}
```

In Production configuration, the same 2 properties are to be setup:

Listing 10: Additional properties for PostgreSql/MongoDb support at runtime

```
quarkus.hibernate-orm.enabled= false / true
ccs.db.type=mongo / postgre
```

8.2.1.2.1.4 In the test classes

The 2 first examples are about testing in Test mode (not IT) without any container launched.

First example is about using no Container in the test: Class Test name can be without IT but as `XxxTest`.

Listing 11: Example of usage of **NoResource** for No container in a test

```
// Do not use @QuarkusIntegrationTest
@QuarkusTest
@TestProfile(NoResourceProfile.class)
public class DriverS3NoS3ConfiguredTest {

}
```

Second example is the same, without any container, but without using the **NoResourceProfile.class**. This will probably generate some warn log about non available services, but they should not be harmful.

Listing 12: Example without usage of **NoResource** in a test

```
// Do not use @QuarkusIntegrationTest
@QuarkusTest
public class DriverS3NoS3ConfiguredTest {

}
```

Third example is about using a Container in the test: Class Test name must end with IT as `XxxIT`.

Listing 13: Example of usage of **MinioProfile** for Minio S3 container in a test

```
// Do not use @QuarkusIntegrationTest
@QuarkusTest
// Define Minio Profile
@TestProfile(MinioProfile.class)
public class DriverS3MinioIT extends DriverS3Base {

  @BeforeAll
  static void setup() {
    // Example: usage of MinioResource to setup the parameters that should be loaded from properties in normal code
    StgDriverS3Properties.setDynamicS3Parameters(MinIoResource.getUrlString(), MinIoResource.getAccessKey(),
        MinIoResource.getSecretKey(), MinIoResource.getRegion());
  }
}
```

8.3 Using fake Streams in tests

Often we need to have a Fake InputStream or a Fake OutputStream, without having to generate a Stream fully in memory.

Listing 14: Example of usage of **FakeInputStream VoidOutputStream** in a test

```
  @Test
  void createConsumeInputOutputStream() {
     long length = x;
     try (InputStream inputStream = new FakeInputStream(length);  // Return a Fake InputStream with random content
          OutputStream outputStream = new VoidOutputStream()) {   // DevNull OutputStream
        assertEquals(length, inputStream.transferTo(outputStream));
     }
     try (InputStream inputStream = new FakeInputStream(length, (byte) 'A');  // Content will be fill with 'A'
          OutputStream outputStream = new VoidOutputStream()) {   // DevNull OutputStream
        assertEquals(length, inputStream.transferTo(outputStream));
     }
  }
}
```

CHAPTER

NINE

ANNEXES

CONTENTS:

LIST OF FIGURES

LIST OF TABLES

LIST OF CODE BLOCKS

HTTP ROUTING TABLE

/ccs

/cloudclonestore

/reconciliator

/replicator