Writing Blobs

To input data directly associated to a time-annotated resource, use the TimeAnnotatedBlobWriter operator, as shown below:

```
Resource stream = Resource.uriRef("urn:stream");
TimeAnnotatedBlobWriter bw = new TimeAnnotatedBlobWriter();
bw.setStream(stream);
bw.addInput(TemporalAnnotation.getInstant(0),new FileInputStream("data000.jpg"));
context.perform(bw);

//write some metadata about these resources

TripleWriter tw = new TripleWriter();
tw.add(new Triple(stream,Dc.FORMAT,"image/jpeg"); context.perform(tw);
```

the addInput method receives a time annotation and an input stream. The input stream provides access to the data that will be written while the time annotation indicates the exact frame in the stream, or time-annotated resource, that will contain the data. You can write several blobs in one operation as long as they are aassociated with time annotations of the same resource:

```
bw = new TimeAnnotatedBlobWriter();
bw.setStream(stream);
bw.addInput(TemporalAnnotation.getInstant(1), new FileInputStream("data001.jpg"));
bw.addInput(TemporalAnnotation.getInstant(2), new FileInputStream("data002.jpg"));
bw.addInput(TemporalAnnotation.getInstant(3), new FileInputStream("data003.jpg"));
bw.addInput(TemporalAnnotation.getInstant(4), new FileInputStream("data004.jpg"));
context.perform(bw);
```

Blobs that are written in one operation will be stored as a single logical group in tupelo to optimize their indexing and retrieval by ranges, although they still can be accessed separately. This allows the API clients to control the granularity of the storage used for blobs. The performance gains are reduced if blobs tend be accessed by points in time instead of ranges, or if write operations are performed over discontiguous groups of blobs.