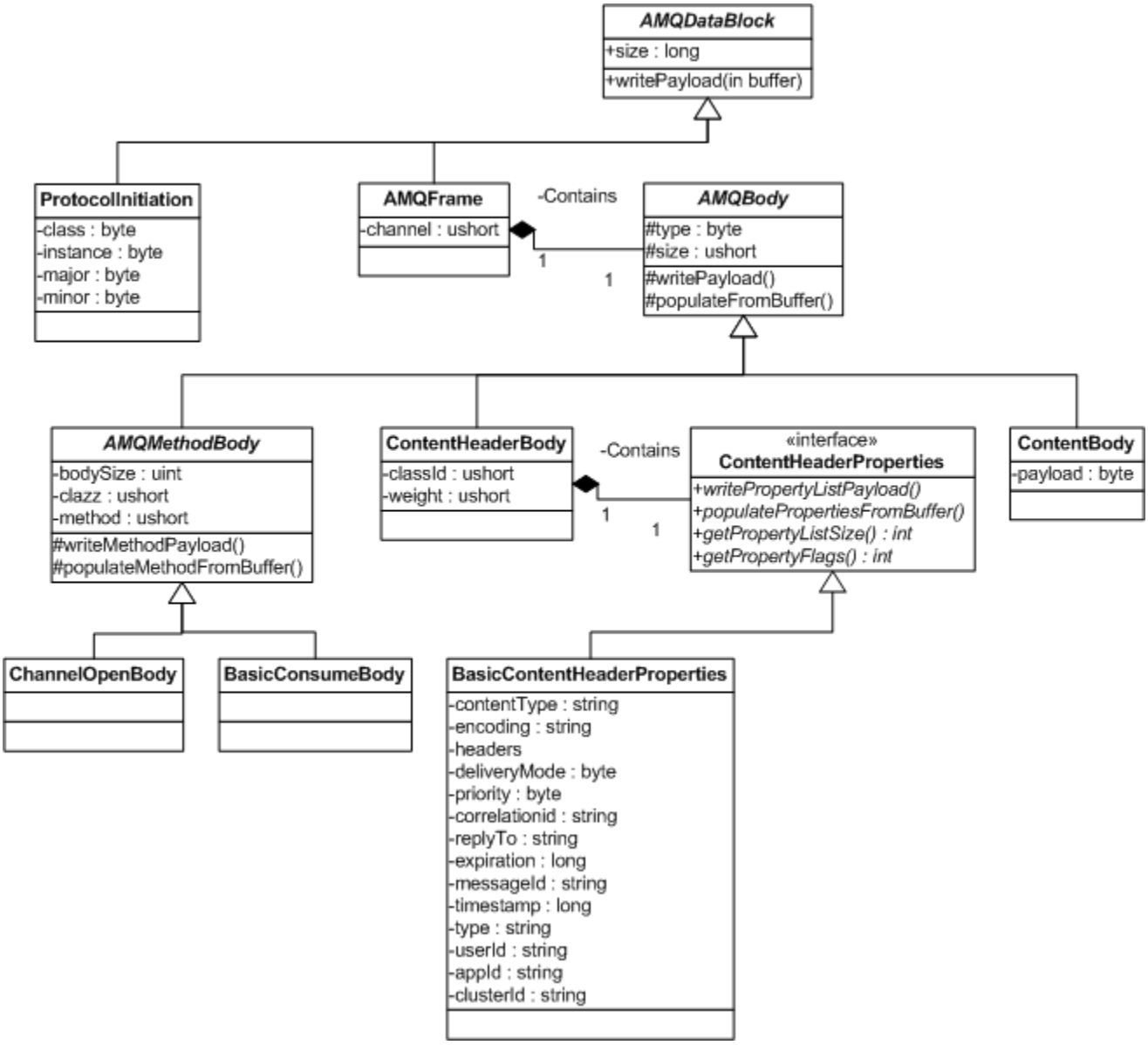
# Qpid Design - Framing

This page describes the classes backing the AMQP 0-8..0-91 protocols. It will also be out of date in some places.

## Frame Classes

The framing definition in the protocol specification maps quite nicely to an object-oriented representation. The class diagram is shown below:




The `AMQDataBlock` at the root of the hierarchy defines a `writePayload` method that subclasses implement in order to be able to transform themselves into bytes. This is called by the encoder, documented below. The decoding (from bytes into objects) is slightly more complex since it involves factories for the instantiation of the correct objects (again documented below).

An `AMQFrame` is the basic unit transmitted over the network, and contains a body which is the real payload. There are numerous method frames, which are subclasses of `AMQMethodBody`. The method body subclasses are all code generated from the protocol specification. The ContentHeaderBody can support different types of content properties or metadata (examples being file or stream in addition to *basic* which is standard JMS-style messaging).

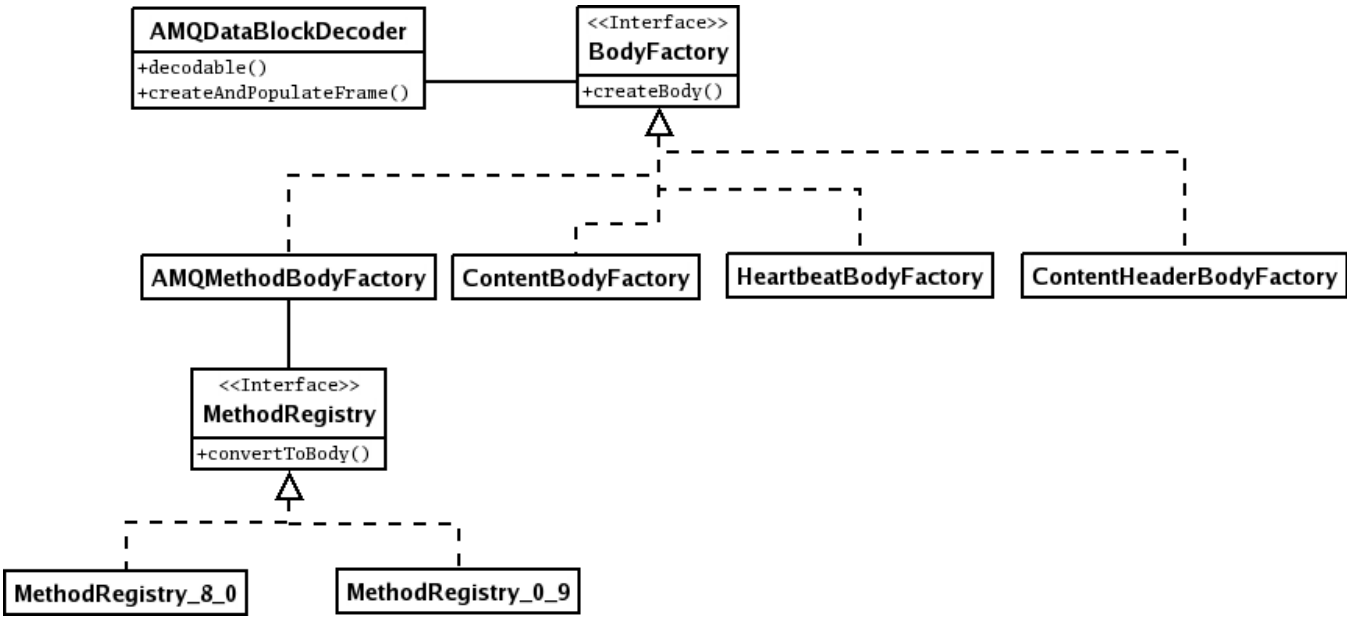
`ContentBody` is a lightweight wrapper for message data.

## Encoding

Encoding is a straightforward process. The `AMQDataBlock` class has only two method: `getSize()` and `writePayloadToBuffer(ByteBuffer)`. The encoder simply needs to ask the data block its size, allocate a buffer of that size, then ask the data block to write itself into the buffer.

## Decoding

The classes involved in decoding are illustrated in this UML class diagram:




The `AMQDataBlockDecoder` has only two methods: `decodable()` in which it attempts to read enough information from the supplied buffer to determine whether it has all the data and whether it appears to represent a known data block. If it needs more data, it return false. If the frame appears to be invalid it throws an exception.

The decoder stores the factories for HeartbeatBody, ContentHeaderBody and ContentBody frame types in an array, indexed on type. The AMQMethodBody factory is version specific and retrieved from the current session. The decoder constructs an `AMQFrame`, passing in the factory the appropriate factory. The result of that call is either a fully populated frame or an exception being thrown if data is invalid or inconsistent.

The `MethodBodyDecoderRegistry` is generated from the protocol XML. Each method is registered by protocol class and protocol method and when looked up by the `AMQMethodBodyFactory` an instance of the appropriate method body is returned. The generated code for the methods handles the reading and writing of the bytes to and from ByteBuffers as well as calculation of the size of the populated method bodies.