# Queues and topics

Messages are sent over the message bus. For optimal performance, messages may be sent over dedicated queues and topics. This page describes a suggested use of queues and topics.



## Types of destinations

The destinations are split into different categories, inspired by Java JMS. There are four types of possible JMS destinations:

### Topics

Properties:

- Messages are not persistent on a topic, messages will only be received if someone listens at the time of the message.
- Topics can have multiple listeners that will receive a message.
- Topic are globally created and managed

### Queues

Properties:

- Messages are persistent on the queue, meaning that if a listener looses connection or is restarted, it can pick up hte messages when it comes back.
- Only one listener will receive a message if there are multiple listeners.
- Queue are globally created and managed

### Durable topics

Properties:

- Multiple listeners can all receive a message, like a topic
- Messages are persistent, that is they are remembered until all known listeners have acknowledged them, even if the listener is not currently available. (Danger of flooding the queue if a listener never returns to acknowledge messages)
- Durable topics are globally created and managed

### Temporary queues

Properties:

- Only one listener receives the message, like a queue
- Messages are not persistent (since the queue disappears with the client).
- Queue is generated by a client.

## Concrete destinations in a repository

The following sections documents the  destinations used in the reference architecture together with alternatives. Note all destinations in the reference architecture are local to a given repository.

### The repository topic

**LISTENERS**: Every component participating in a Bit Repository.
**MESSAGES**: IdentifyPillarFor<Op>Request.
**NAMING**: "type:<repository-id>"
**DISCUSSION**: This topic is the abstraction layer, that makes it possible to use the protocol with no knowledge of other participants. The destination is used to send the initial 'Identify...' requests used to start a conversation. See Message flow documentation for details.
**IMPLEMENTATION**: The topic is defined in the RepositorySettings.
**ALTERNATIVES**:

- The repository topic might be used for other message types. It might in fact be used for all the messages in a repository, if the separation recommended here isn't necessary.
- It is possible to define dedicated topics for broadcasting pillar, audittrail and status specific requests in the RepositorySettings. This can be used to optimize the Identify request broadcast, so the requests are only distributed to the relevant components.

### The per-pillar topic

**LISTENERS**: One specific pillar
**MESSAGES**: <Op>Request
**NAMING**: "type:<repository-id>.<pillar-id>"
**DISCUSSION**: This topic is the interface for requesting operations on a pillar.
**IMPLEMENTATION**: Pillar topics are identified through the IdentifyPillarFor<Op>Responses returned as a result of a IdentifyPillarFor<Op>Request.
**GENERATION**: These topics are generated for each pillar by the message bus administrator when a new repository is setup.
**ALTERNATIVES**: It is protocol-safe to make a different choice for making pillar destinations on a per-pillar basis. This can be replaced with a queue or a temporary queue, and the naming is quite optional. A topic seems appropriate, since it will not have persistent messages lying around if the pillar is not available at the time of request. However, there could also be advantages of a queue, since you could replicate you pillar instances and only one would receive the message. It may be considered desirable to use temporary queues, so the queue name will never be reused in a later operation.

## The per-client temporary queue

**LISTENERS**: Clients
**MESSAGE**: IdentifyPillarFor<Op>Response, <Op>Response, <Op>Complete
**NAMING**: "type:<repository-id>.client-name.unique-client-identifier"
**DISCUSSION**: This queue is intended to receive replies to requests sent to the clients. The reference maintains a conversation state while running, which isen't shared between client, nor persisted. This means replies sent to other clients or previous instances of a client are irrelevant. Thus a client is only interested in replies to it's own responses.
**IMPLEMENTATION**: A client initiates a temporary queue on startup with a unique identifier.
**ALTERNATIVES**: It is protocol-safe to make a different choice for making client destinations on a per-client basis. This can be replaced with a queue or a (durable shared) topic, and the naming is quite optional. In case of stateless client, clients with persisted state or shared state this might be relevant.

## Alarms

Alarms are probably best modelled in a separate destination.

### The repository alarm durable topic

**LISTENERS**: One specific client
**MESSAGE**: Alarms
**NAMING**: "type:<repository-id>.alarms"
**DISCUSSION**: This queue is intended to receive alarms that are reported during a SLA-specific conversation.
**IMPLEMENTATION**: All participants in the protocol are given the alarm destination id in it's configuration, and send alarms to this topic.
**ALTERNATIVES**: The repository topic could be used instead.