# Zbus Architecture

## zbus overall components

select

Route **Table**

MqClientPool **Table**

MqClient **detect connectivity**

MqClientPool

**Broker**

**Producer**

**Consumer**

Tracker

pub (TrackerInfo)

MqServer1

MqServer2

MqServer3

MqServer + Tracker = Zbus

ServerSelector

User Defined

1. Both Tracker and MqServer work inside of a same zbus instance.
2. **P**roducer, **B**roker, **C**onsumer, PBC model in high level API
3. PBC handles connection pooling, auto reconnect, high availability etc.

MqServer

MqClient

produce

consume

queryTopic/ConsumeGroup

removeTopic/ConsumeGroup

declareTopic/ConsumeGroup

| | | | | | | | | | | | |

**ConsumeGroup**1

**ConsumeGroup**2

**Topic** Writer

W

R

R

**Unicast** : readers share one same ConsumeGroup

**Broadcast**: readers use private own ConsumeGroups

**Multicast**: mix of the unicast and broadcast

Messaging Model = **Topic** + **ConsumeGroup**

Reader1

Reader

Reader

1. MqClient connects to MqServer, capable of **producing/consuming/managing** topic and consume group in MqServer.
2. Messaging Queue in MqServer is considered to be infinite of depth, messaging sequence follows FIFO, and **reading policy is controlled via ConsumeGroup** --each group can be shared or privately owned by readers. Simple craft on ConsumeGroup make unicast, broadcast and multicast messaging model work.

## zbus URL pattern

/produce/topic

/consume/topic/[group]

/declare/topic/[group]

/query/[topic]/[group]

/remove/topic/[group]

/pause/topic/[group]

/resume/topic/[group]

/empty/topic/[group]

/track\_sub

/track\_pub

/rpc/topic/method/param1/param2/…/[?module=xxx]

## Zbus Client Platforms

**HTTP Raw Client**

**JAVA**

**C#.NET**

**JavaScript (Browser WebSocket + Node.JS)**

**C**

**Python**