



APUSIC
固若长城
睿比世界

Product Whitepaper

Apusic Distributed Message Queue

Copyright Notice

All intellectual property rights including software copyrights involved in this document have been legally registered and duly owned by Kingdee Apusic Cloud Computing Co., Ltd. Such rights are protected by Copyright Law of the People's Republic of China, Regulations on Protection of Computer Software, Regulations on Intellectual Property Protection, relevant international copyright conventions, laws, regulations and other intellectual property legislations. Unauthorized use is strictly prohibited.

Disclaimer

All copyright information contained herein is legally owned by Kingdee Apusic Cloud Computing Co., Ltd. and protected by applicable laws. The Company shall bear no liability for any information unrelated to Kingdee Apusic contained in this document. You may view, copy and print this document only within the scope permitted by Copyright Law of the People's Republic of China. No entity or individual may use, modify or republish any part or content of this document without written authorization from Kingdee Apusic Cloud Computing Co., Ltd. Any violation shall constitute infringement, and the Company reserves the right to pursue legal liabilities accordingly.

Updates to this document shall not be notified separately. You may consult the Company regarding issues related to this document. All rights not expressly granted by the Company are reserved.

Trademark Notice

 is a registered trademark filed with the National Intellectual Property Administration of PRC by Kingdee Apusic Cloud Computing Co., Ltd. Shenzhen. Exclusive rights to the registered trademark are legally owned by Kingdee Apusic and protected by law. Without written consent of Kingdee Apusic, no entity or individual shall use, copy, modify, disseminate, transcribe, or bundle and sell any part of this trademark with other products in any manner or for any reason. Kingdee Apusic shall pursue legal liabilities against any infringement of its trademark rights. All other trademarks or registered trademarks mentioned in this document belong to their respective owners.

Contents

- 1 Version History
- 2 Introduction
- 3 Development History
- 4 Product Overview
 - 4.1 Product Names
 - 4.2 Target Audience
- 5 ADMQ Enterprise Message Queue
 - 5.1 Product Positioning
 - 5.2 Core Features
 - 5.3 Product Functional Architecture
 - 5.4 Main Features
 - 5.4.1 Unified Message Model
 - 5.4.2 Cross-region Replication
 - 5.4.3 Multi-tenancy
 - 5.4.4 Permission Control
 - 5.4.5 Transaction Support
 - 5.4.6 Message Chunking
 - 5.4.7 Transport Security
 - 5.4.8 Multi-protocol Access
 - 5.4.9 Multi-replica Storage
 - 5.4.10 Multi-cluster Channels
 - 5.4.11 Whitelist/Blacklist
 - 5.4.12 Visual Management Console
 - 5.4.13 Monitoring and Operations
 - 5.5 Technical Features
 - 5.5.1 Compute-Storage Separation
 - 5.5.2 High Performance and Low Latency
 - 5.5.3 Unlimited Topic Storage
 - 5.5.4 Seamless Failure Recovery
 - 5.5.5 Built-in Lightweight Computing Engine
 - 5.6 Product Advantages
 - 5.7 Application Scenarios

- 6 ADMQ Dedicated Message Queue
 - 6.1 Product Positioning
 - 6.2 Core Features
 - 6.3 Product Composition
 - 6.3.1 ADMQ for Kafka
 - 6.3.2 ADMQ for RocketMQ
 - 6.3.3 ADMQ for RabbitMQ
 - 6.3.4 ADMQ for ActiveMQ
 - 6.3.5 ADMQ for MQTT
 - 6.4 Product Functional Architecture
 - 6.5 Main Features
 - 6.5.1 Native Protocol Access
 - 6.5.2 Enterprise-grade Security
 - 6.5.3 High Availability Mechanisms
 - 6.5.4 Visual Management
 - 6.5.5 Monitoring and Operations
 - 6.6 Technical Features
 - 6.6.1 Zero-modification Migration
 - 6.6.2 Seamless Ecosystem Reuse
 - 6.6.3 Deep Adaptation to Domestic Environments
 - 6.6.4 Lightweight Deployment
 - 6.6.5 Stable and Reliable Operation
 - 6.7 Product Advantages
 - 6.8 Application Scenarios
- 7 Product Comparison
 - 7.1 Comparison of ADMQ's Two Product Forms
 - 7.2 Comparison of ADMQ Dedicated Message Queues
- 8 System Hardware and Software Requirements
 - 8.1 ADMQ Enterprise Message Queue
 - 8.2 ADMQ Dedicated Message Queue
 - 8.3 Software Environment
 - 8.4 Performance Specifications
- 9 Customer Cases
 - 9.1 System Decoupling

- 9.1.1 A Billing Platform
 - 9.1.1.1 Background
 - 9.1.1.2 Solution
- 9.2 Peak Shaving and Valley Filling
 - 9.2.1 An E-commerce Website
 - 9.2.1.1 Background
 - 9.2.1.2 Solution
- 9.3 Internet of Things (IoT)
 - 9.3.1 Smart Sensing for Power Grids
 - 9.3.1.1 Background
 - 9.3.1.2 Solution
- 9.4 Domestic Replacement

1 Version History

Date	Manual Version	Applicable Product	Update Description
Dec 2024	V2.0.2	ADMQ	Added copyright notices and other content
Dec 2025	V2.0.5	ADMQ	Product upgrade
Jun 2026	V2.0.6	ADMQ	Added the ADMQ dedicated engine series

2 Introduction

Message middleware primarily helps applications communicate asynchronously. In system architecture, it serves purposes such as peak shaving and valley filling, heterogeneous integration, application decoupling, and asynchronous isolation. Traditional message middleware evolved from the JMS specification — for example, the Kingdee Apusic message middleware that Apusic released around 2003. As internet applications grew, distributed message middleware such as RocketMQ, Kafka, and RabbitMQ began to emerge. Some of these products focus on throughput, some on high performance, and some on flexible routing; different products suit different application scenarios. Kafka, for instance, leans toward data stream processing and is mainly used for big data stream processing and log processing.

In recent years, as cloud computing and microservice architectures have gained popularity, distributed message queues are increasingly used in scenarios such as the Internet of Things (IoT), distributed transactions, real-time computing, and large-scale cache synchronization. In banking, for example, a distributed message queue can split a transaction into multiple local sub-transactions and process them asynchronously through the queue, which improves transaction processing efficiency and concurrency while reducing the likelihood of synchronous blocking and dirty reads. Distributed message queues can also support rapid analysis and processing of large-scale data, improving the efficiency and accuracy of risk assessment and data analysis.

In 2020, Kingdee Apusic began upgrading its traditional message middleware for the financial and telecommunications industries, turning its attention to a new generation of financial-grade message queue technology: developing a new generation of message middleware products and providing the related support services. Through continuous product evolution, ADMQ has taken shape in two product forms — the Enterprise Message Queue for large-scale messaging platform scenarios, and the ADMQ Dedicated Message Queue for open-source message replacement scenarios — fully meeting enterprises' message middleware needs at different levels.

3 Development History

- Jun 2026 — ADMQ V2.0.6 released, adding dedicated message queues such as ADMQ for Kafka, ADMQ for RocketMQ, ADMQ for RabbitMQ, and ADMQ for MQTT.
- Dec 2024 — ADMQ V2.0.2 released, adding copyright notices and other content.
- Dec 2023 — ADMQ V2.0 Standard Edition adapted to multiple domestic software and hardware systems.
- Jul 2023 — ADMQ V2.0 Standard Edition released, providing enterprise-grade message middleware capabilities.
- Jun 2022 — ADMQ V2.1 released, supporting three-role permission management and multi-instance deployment management.
- Dec 2021 — ADMQ V2.0 released, providing plug-in extension capabilities and adapting to multiple domestic software and hardware systems.
- Aug 2021 — ADMQ V1.1 released, providing dynamic proxy and rapid installation features.
- Jul 2021 — A refined ADMQ V1.0 released, providing distributed messaging capabilities for enterprise applications.
- May 2021 — ADMQ V1.0 beta released, providing unified message access capabilities.
- Jan 2021 — Independent R&D of the distributed message queue software was initiated.

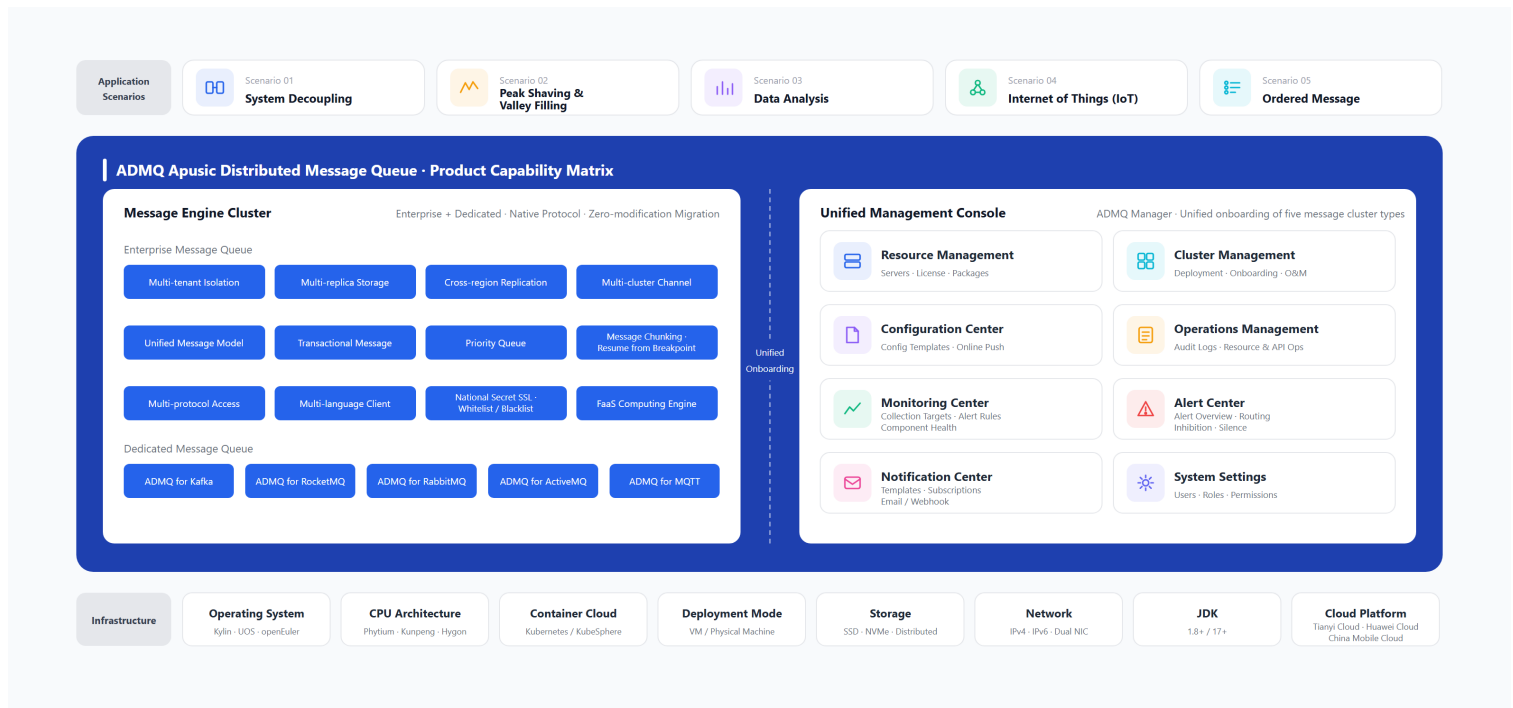
4 Product Overview

Apusic Distributed Message Queue (abbreviated as ADMQ) is a financial-grade, distributed message middleware product series that Kingdee Apusic has built for enterprise application scenarios. ADMQ offers two product forms to meet the needs of different scales and scenarios:

4.1 Product Names

Kingdee Apusic Distributed Message Queue Software V2.0 (Apusic Distributed Message Queue, abbreviated as ADMQ) offers two product forms, covering the full spectrum of scenarios from platform construction to domestic replacement:

- **ADMQ Enterprise Message Queue:** targets large-scale platform scenarios such as a unified enterprise message hub, multi-tenant shared infrastructure, cross-region data replication, and complex message processing; it provides enterprise-grade messaging foundation capabilities.
- **ADMQ Dedicated Message Queue:** targets domestic replacement and open-source messaging product replacement scenarios; it provides five dedicated message queues — ADMQ for Kafka, ADMQ for RocketMQ, ADMQ for RabbitMQ, ADMQ for ActiveMQ, and ADMQ for MQTT — enabling smooth migration and seamless replacement.



4.2 Target Audience

- Distributed application architects and developers.

- Big data development engineers.
- System monitoring and operations personnel.
- Project leads and decision makers for enterprise domestic replacement projects.
- Users in industries such as finance, telecommunications, government, and energy.

5 ADMQ Enterprise Message Queue

5.1 Product Positioning

ADMQ Enterprise Message Queue is a financial-grade distributed message middleware with features such as multi-tenant isolation, cross-cluster replication, strong consistency, high reliability, high concurrency, cloud-native design, and a rich set of message types. It adopts a compute-storage separated architecture and can scale elastically. It supports native Java, C++, Python, and Go APIs, and accepts access from Kafka, RocketMQ, RabbitMQ, and ActiveMQ clients as well as protocols such as MQTT and JMS, which lowers the integration effort for different business systems.

ADMQ Enterprise Message Queue applies to scenarios such as peak shaving and rate limiting, system decoupling, asynchronous processing, and data analysis. In terms of performance, it meets the requirements of massive message backlog, high throughput, and low latency. It provides a unified visual console for resource management together with real-time monitoring.

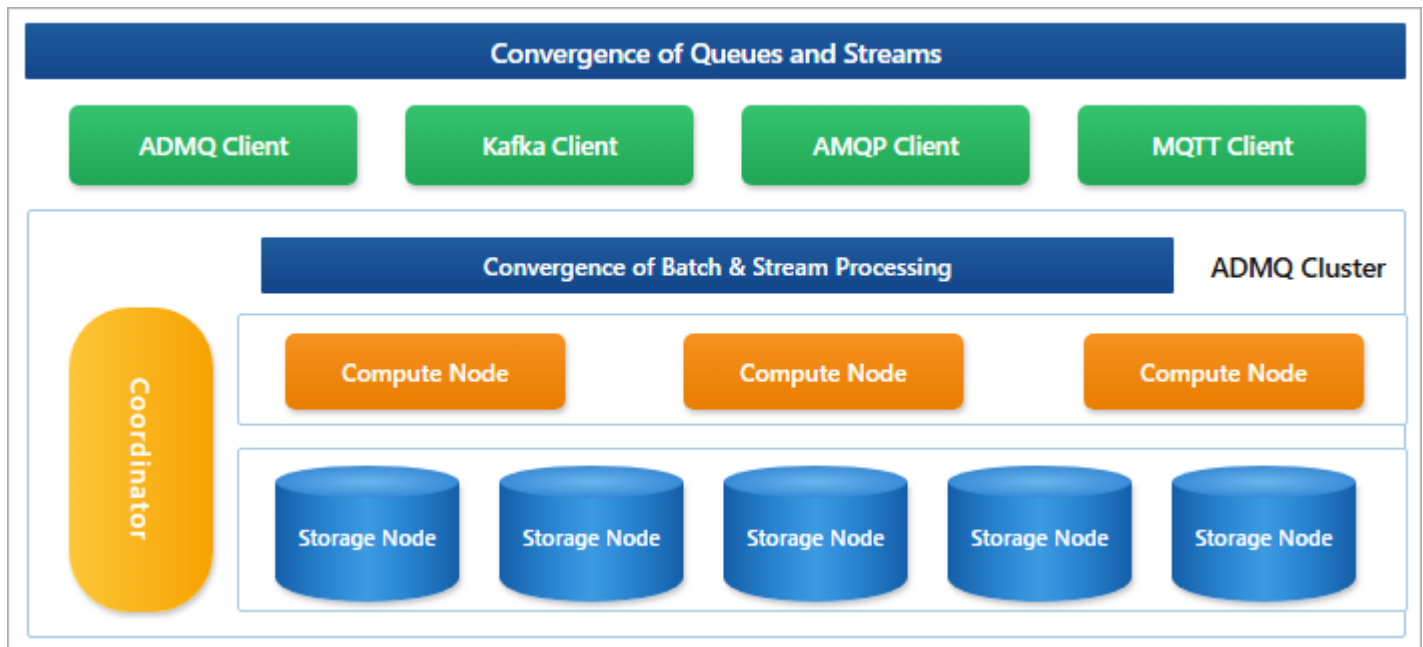
5.2 Core Features

Feature	Description
Message Publish/Subscribe	One-to-many consumption model: publishers can send messages to a topic, where they are consumed concurrently by one or more consumers. Point-to-point transfer mode is also supported.
Cross-region Replication	Cross-region, long-distance, millisecond-level real-time message synchronization, supporting cross-region publishing and subscribing of messages.
Multi-tenant Support	Multi-tenancy is built into the cluster, making full use of the infrastructure and simplifying operations and management.
Multi-language Clients	Client libraries for Java, Go, Python, C++, .NET, Node.js, and more.
Multi-protocol Access	Supports native access over the Kafka, RocketMQ, RabbitMQ, ActiveMQ, MQTT, and JMS protocols, so migration to ADMQ requires no code changes.
Message Compression	Supports sending compressed messages in multiple formats, including LZ4, ZLIB, ZSTD, SNAPPY, and GZIP.
Network Protocols	Supports IPv4 and IPv6; supports dual NICs on a host; supports specifying an IP address for message transfer.

File Transfer	Supports reliable file transfer, resume from breakpoint, and multi-threaded concurrent sending.
Message Transfer	Supports reliable message transfer and resume from breakpoint.
Unified Authentication and Authorization	Provides a unified visual interface to create and destroy tenants, namespaces, and topics, and to assign the corresponding usage permissions to users.
Priority Queue	Supports message priority handling.
Message Query	Supports message query and full-chain message trace tracking.
National Secret SSL Transmission	Supports National Secret SSL encrypted transmission during message transfer.

5.3 Product Functional Architecture

ADMQ Enterprise Message Queue consists of three parts: a stateless service layer, which processes messages sent by producers and dispatches them to consumers; a storage layer, which handles persistent message storage; and ZooKeeper, which performs metadata storage, cluster configuration, and coordination.



5.4 Main Features

5.4.1 Unified Message Model

A unified consumption model — producer, topic, subscription, consumer — is abstracted through the concept of a subscription. ADMQ's message model supports both the queue model and the stream model. A message is stored

only once on the topic, but users can consume those messages through different subscription types:

- Consumers are grouped together to consume messages, and each consumer group is a subscription.
- Each topic can have different consumer groups.
- Each consumer group is a subscription to the topic.
- Each consumer group can use its own consumption model:
 - Exclusive: at any time, exactly one consumer in a consumer group consumes the messages on the topic.
 - Failover: multiple consumers can attach to the same subscription, but only one of them is selected as the primary consumer for that subscription; the others are designated as failover consumers.
 - Shared: any number of consumers can attach to the same subscription according to application needs. All messages in the subscription are distributed round-robin to multiple consumers, and each message is delivered to only one consumer.
 - Key_Shared: within the same subscription, messages with the same key are delivered to the same consumer according to a given mapping rule.

5.4.2 Cross-region Replication

Cross-region replication backs up the messages persisted in ADMQ across multiple clusters. ADMQ builds this capability on top of its core services: users can choose a synchronous or asynchronous configuration at deployment time, and can configure the replication mechanism per topic. Producers can write to a shared topic from any region, and ADMQ ensures that this information is visible to consumers everywhere.

With only simple configuration, ADMQ can replicate a single topic in multiple directions to any number of external data centers. Data synchronization happens in the background and is handled by the storage layer, so it does not affect other workloads being processed by the service layer.

5.4.3 Multi-tenancy

Each tenant can have its own authentication and authorization mechanism. A tenant is also the unit of management for storage quotas, message TTL, and isolation policies. Multi-tenant infrastructure can be shared across multiple users and organizations while keeping them isolated from one another. Activity in one tenant does not affect the security or SLA of other tenants.

5.4.4 Permission Control

User permissions can be managed and assigned at the tenant and namespace levels. When message producers and consumers connect to the cluster, permissions must be assigned for the resources they use in order to keep data secure.

A parent-child account permission model is supported, in which a child account holds a subset of the parent account's permissions.

5.4.5 Transaction Support

Multiple message sends and receives can be included in a single transaction, and ADMQ guarantees that all the message send and message receive unit operations in a transaction either succeed or fail as a whole. This transaction support lets ADMQ group multiple message sends into one transaction, making it easy to implement business flows such as related-party transactions.

5.4.6 Message Chunking

ADMQ imposes no strict limit on the length of a single message. The message transfer protocol uses 4 bytes to store the payload length, so in theory it supports message content of up to 2 GB.

To ensure transfer performance, the length of each message is best kept within 10 MB. For larger messages, ADMQ supports chunked transfer: the client splits the message into multiple message chunks, and once all chunks have been transferred successfully, ADMQ reassembles them in order into one complete message and delivers it to the subscribing client.

5.4.7 Transport Security

The TLS protocol is supported, protecting the transport channels between clients and servers as well as between servers; National Secret SSL is also supported.

Messages in transit can be encrypted. Producers and consumers can agree on a private key in advance: the producer encrypts the message when sending it, and the consumer decrypts it with the private key, which keeps messages secure during transfer.

5.4.8 Multi-protocol Access

ADMQ provides powerful multi-protocol access. The compute component abstracts an API layer for everything related to message handling, including the read and write of messages, topic publishing and management, partition handling, message content formatting, consumption progress, and message query. When a client connects using another protocol, the corresponding operations can be conveniently translated into methods of this abstract API.

This approach makes it easy to adapt clients of other protocols. The Kafka, RocketMQ, RabbitMQ, AMQP, JMS, and MQTT protocols are currently supported.

5.4.9 Multi-replica Storage

After receiving a message, ADMQ can store its replicas on multiple nodes. Replicas can also be placed according to region or data center policies, ensuring that they are spread evenly across different regions and data centers, which improves data availability.

5.4.10 Multi-cluster Channels

ADMQ has cross-cluster message transfer built in: when one cluster receives a message, it automatically forwards the message to other clusters, enabling message transfer across multiple clusters and networks.

The message forwarding destination can be configured at the tenant and namespace level.

5.4.11 Whitelist/Blacklist

ADMQ supports IP and network segment whitelist/blacklist filtering, strictly controlling the IP addresses or network segments from which clients may access.

5.4.12 Visual Management Console

- Cluster Management: deploy and manage ADMQ clusters; certificate management, dependency management, plug-in management, and node management; configure cluster information and node parameters; automatic scaling; node resource monitoring, and more.
- User Management: add, edit, and delete user information; change user passwords; view tokens; view permissions.
- Channel Management: a visual interface for cross-cluster message replication, including adding, editing, and deleting channel information.
- Resource Management: manage and monitor tenants, namespaces, and topics.
- Message Query: monitor and query the trace of messages.
- System Operations: monitoring and operations, system alerts, function verification, and node logs.
- System Configuration: server management, license management, software package management, default configuration management, operation logs, and event management.
- Plug-in Management: Kafka management, RocketMQ management, RabbitMQ management, and ActiveMQ management.

5.4.13 Monitoring and Operations

The monitoring and operations module built into the console can monitor and manage cluster information, presenting it visually in charts.

AUC (login platform) and AMP (monitoring platform) are integrated, so cluster nodes can jump automatically from the console to the AMP monitoring charts to monitor node resources.

For the protocol plug-ins in the Enterprise Message Queue, the console monitors resource production and consumption from dimensions such as tenant, topic, and subscription group.

An HTTP monitoring interface is provided.

5.5 Technical Features

5.5.1 Compute-Storage Separation

ADMQ Enterprise Message Queue uses a cloud-native architecture with compute-storage separation, decoupling message storage from services so that the storage layer and the service layer can scale independently. The scaling process requires no data rebalancing and does not copy old data from existing storage nodes to new storage nodes, which reduces network bandwidth and I/O consumption.

5.5.2 High Performance and Low Latency

It efficiently supports millions of message productions and consumptions, with a massive message backlog and no upper limit on backlog capacity. A single cluster exceeds 100,000 QPS, and protective mechanisms on latency guarantee low latency to meet business performance requirements.

5.5.3 Unlimited Topic Storage

Topics are partitioned into shards and stored in a distributed manner, so a topic's capacity is not limited by the capacity of a single node.

5.5.4 Seamless Failure Recovery

Because the service layer consists of stateless nodes, the failure of any single node has no effect on overall production and consumption; the cluster redistributes topics to available nodes according to load.

Before a message is acknowledged, it is persisted to N storage nodes. The data is accessible from multiple nodes and can survive the failure of N-1 nodes. Failed nodes are replaced by adding new storage nodes, without affecting the availability of the entire cluster.

5.5.5 Built-in Lightweight Computing Engine

A built-in lightweight computing engine lets developers implement functions (processing logic) in Java or Python, giving users a FaaS (Function as a Service) platform that is simple to deploy and convenient to operate. It brings users the benefits of serverless computing, similar to AWS Lambda.

5.6 Product Advantages

Unified Message Hub Capabilities

ADMQ Enterprise Message Queue can serve as an enterprise's unified message hub, providing multi-protocol access, multi-tenant isolation, unified management, cross-region replication, and other capabilities, so that messaging infrastructure can be built once and shared.

Global Dynamic Configuration

A unified visual interface is provided to create and destroy tenants, namespaces, and topics and to assign the corresponding usage permissions to users, keeping user data secure.

Unified Queue and Stream Models

ADMQ provides a unified message model and API, achieving consistency between the queue model and the stream model. At the topic level, only one copy of the data needs to be stored, and it can be consumed multiple times, giving users more flexibility.

Efficient Operations

A unified console combines management and control and supports visual service deployment, which simplifies the deployment process. Multi-dimensional monitoring and alerting are available for cluster resources, bringing implementation and operations personnel a simple, fast way to work.

Multi-tenant Isolation

Isolation between tenants ensures that tenants do not interfere with one another and that user privacy is protected. The resources of each tenant are managed and monitored, with production and consumption visible in real time.

Full Adaptation to Domestic Environments

It runs stably and efficiently on domestic operating systems and hardware architectures. Operating systems include UOS, Huawei openEuler, Kylin V10, and Zhongke Fangde; hardware platforms include Phytium, Kunpeng, Zhaoxin, and Hygon.

5.7 Application Scenarios

Unified Enterprise Message Hub

Large enterprises and organizations often have many business systems, differing message protocols, and fragmented management. ADMQ Enterprise Message Queue can serve as the enterprise's unified message hub, so that messaging infrastructure is built once and shared.

Cross-region Data Synchronization

Through cross-region replication, ADMQ can synchronize and share messages across multiple data centers, meeting enterprise needs such as multi-active deployments across sites and disaster recovery.

Large-scale IoT Platforms

ADMQ Enterprise Message Queue suits large-scale IoT data collection and processing, supporting massive device access, highly concurrent message processing, and real-time data distribution.

Complex Event Processing

With the built-in lightweight computing engine, ADMQ handles complex event processing requirements such as real-time data preprocessing, event routing, and data transformation.

6 ADMQ Dedicated Message Queue

6.1 Product Positioning

The ADMQ Dedicated Message Queue is a series of dedicated message queues that Kingdee Apusic offers for enterprise domestic replacement and open-source message replacement needs. The series comprises five message queues — ADMQ for Kafka, ADMQ for RocketMQ, ADMQ for RabbitMQ, ADMQ for ActiveMQ, and ADMQ for MQTT — each serving the users of a different open-source messaging product.

While retaining native protocol compatibility, these products add enterprise-grade enhancements and domestic adaptation. They provide commercial capabilities such as secure authentication, permission control, visual management, operations monitoring, and performance optimization, helping enterprises complete the replacement of message middleware in domestic environments at the lowest modification cost and with the greatest efficiency.

6.2 Core Features

Feature	Description
Native Protocol Compatibility	Fully compatible with the native protocols, clients, and APIs of the corresponding messaging product, so business systems can migrate without code changes.
Enterprise-grade Security	Supports identity authentication, ACL access control, TLS/National Secret SSL transport encryption, audit logs, and other enterprise-grade security capabilities.
Enhanced High Availability	Supports high-availability mechanisms such as master-slave replication, multi-replica, and automatic failover.
Visual Management	Provides a unified web console supporting cluster management, resource management, message query, monitoring and alerting, and more.
Performance Optimization	Performance tuning for domestic environments and enterprise application scenarios, with best practice guidance.
Domestic Adaptation	Adapts to mainstream domestic operating systems, chips, databases, and middleware ecosystems.
Multi-language Clients	Supports clients in Java, Go, Python, C++, .NET, Node.js, and other languages.

6.3 Product Composition

6.3.1 ADMQ for Kafka

A high-throughput streaming messaging solution for Kafka users. It supports:

- Full compatibility with Kafka's produce/consume protocol, partition assignment protocol, and offset management protocol.
- Kafka native clients (2.x–3.x) and ecosystem tools.
- Core features such as topic partitions, the replica mechanism, consumer groups, message compression, and offset commit.
- High-throughput scenarios such as log collection, data synchronization, and stream processing.

6.3.2 ADMQ for RocketMQ

A highly reliable, financial-grade messaging solution for RocketMQ users. It supports:

- Full compatibility with RocketMQ's produce/consume protocol, NameServer routing protocol, and message storage protocol.
- Core features such as ordered messages, transactional messages, scheduled/delayed messages, broadcast consumption, and message tracing.
- High-availability mechanisms such as master-slave replication and DLedger multi-replica.
- High-reliability scenarios such as financial transactions, order processing, and distributed transactions.

6.3.3 ADMQ for RabbitMQ

A flexible-routing messaging solution for RabbitMQ users. It supports:

- Full compatibility with the AMQP 0-9-1 protocol and official clients for Java, Python, Go, Node.js, and others.
- Multiple exchange types, including Direct, Fanout, Topic, and Headers.
- Multiple queue types, including Classic, Quorum, and Stream.
- Scenarios such as flexible routing, asynchronous decoupling, and peak shaving and valley filling.

6.3.4 ADMQ for ActiveMQ

A traditional JMS messaging solution for ActiveMQ users. It supports:

- Full compatibility with the JMS specification, supporting both queues and topics as message models.
- The JMS 1.1/2.0 specifications, and compatibility with traditional Java EE applications.
- Scenarios such as traditional enterprise applications and Java EE system migration.

6.3.5 ADMQ for MQTT

A lightweight messaging solution for IoT and mobile internet scenarios, built on enterprise-grade enhancements to EMQX. It supports:

- Full compatibility with the MQTT 3.1, 3.1.1, and 5.0 protocols, supporting massive MQTT client access.

- Core features such as the publish/subscribe model, QoS 0/1/2 message quality levels, will messages, retained messages, and session persistence.
- Multiple access methods, including MQTT over WebSocket, MQTT over SSL/TLS, and MQTT over QUIC.
- High-availability capabilities such as rule engine, data bridge, message persistence, and cluster deployment.
- Stateless node clusters and automatic fault recovery, providing high availability and horizontal scaling.
- Scenarios such as IoT platforms, connected vehicles, mobile message push, and smart device management.

6.4 Product Functional Architecture

Each ADMQ Dedicated Message Queue adopts an enterprise-enhanced architecture for its corresponding messaging product. On top of a native protocol compatibility layer, they uniformly provide enterprise-grade capabilities such as secure authentication, permission control, visual management, monitoring and alerting, and operations management.

Every product includes the following core components:

- **Message Queue:** routes, stores, forwards, and consumes messages, and fully implements the core protocol of the corresponding messaging product.
- **NameServer/Broker/Node:** provides message service nodes and supports cluster deployment and high availability.
- **Enterprise Security Layer:** provides unified authentication, ACL authorization, TLS/National Secret SSL encryption, and other security capabilities.
- **Visual Management Console:** provides cluster management, resource management, message query, monitoring and alerting, and more.
- **Operations Monitoring System:** provides Prometheus metrics export, log management, alert notifications, and other operations capabilities.

6.5 Main Features

6.5.1 Native Protocol Access

ADMQ Dedicated Message Queues fully implement the core protocols of the corresponding messaging products and support direct access from native clients:

- Kafka protocol: supports PLAINTEXT, SSL, SASL_PLAINTEXT, and SASL_SSL.
- RocketMQ protocol: supports the Remoting protocol, gRPC protocol, MQTT protocol, and HTTP protocol.
- AMQP protocol: supports the AMQP 0-9-1 protocol, SSL/TLS encryption, and SASL authentication.
- MQTT protocol: supports the MQTT 3.1/3.1.1/5.0 protocols as well as access methods such as MQTT over WebSocket, MQTT over SSL/TLS, and MQTT over QUIC.

- JMS protocol: supports the JMS 1.1/2.0 specifications and is compatible with traditional Java EE applications.

6.5.2 Enterprise-grade Security

- **Identity Authentication:** supports username/password, AK/SK, SASL/PLAIN, SCRAM, and other authentication methods.
- **Access Control:** supports ACL access control, RBAC role management, and Vhost/namespace isolation.
- **Transport Encryption:** supports TLS/SSL and National Secret SSL encrypted transmission.
- **Audit Logs:** records all management operations and key message operations to meet compliance requirements.

6.5.3 High Availability Mechanisms

Product	High Availability Mechanism
ADMQ for Kafka	Multi-replica mechanism, ISR synchronization, Leader election, Controller management
ADMQ for RocketMQ	Master-slave replication, DLedger multi-replica, NameServer cluster
ADMQ for RabbitMQ	Mirrored queues, Quorum Queue, Stream Queue multi-replica
ADMQ for ActiveMQ	Master/Slave, Network of Brokers
ADMQ for MQTT	Stateless node cluster, automatic fault recovery, horizontal scaling

6.5.4 Visual Management

- **Cluster Management:** create, monitor, and manage message cluster nodes.
- **Resource Management:** manage resources such as topics, queues, exchanges, vhosts, and consumer groups.
- **Message Query:** query messages by time, key, ID, and other conditions, with message trace tracking.
- **Monitoring and Alerting:** view cluster status, performance metrics, message backlog, and consumption latency in real time.
- **System Management:** license management, user management, operation logs, and configuration management.

6.5.5 Monitoring and Operations

- Prometheus metrics export.
- Grafana dashboard integration.
- Log management and auditing.
- Alert rules and notifications.
- Health checks and automatic recovery.

6.6 Technical Features

6.6.1 Zero-modification Migration

ADMQ Dedicated Message Queues are fully compatible with the protocols and clients of the corresponding messaging products, so business systems can migrate by changing only the connection address, with no code changes. The migration cost approaches zero.

6.6.2 Seamless Ecosystem Reuse

They are compatible with the native tools, monitoring systems, and data synchronization tools of the corresponding messaging products, so enterprises can reuse their existing technical investment and operations practices without additional adaptation development.

6.6.3 Deep Adaptation to Domestic Environments

They adapt to mainstream domestic operating systems (Kylin, UOS, and others) and domestic chips (ARM, RISC-V, x86, and others), and are compatible with domestic database and middleware ecosystems, meeting the deployment requirements of government and enterprise domestic replacement projects.

6.6.4 Lightweight Deployment

ADMQ Dedicated Message Queues use a lightweight architecture with low resource usage and simple deployment, and can be deployed and run quickly on physical machines, virtual machines, and container environments.

6.6.5 Stable and Reliable Operation

Mechanisms such as multi-replica, master-slave replication, and automatic failover safeguard the high availability of message services and the reliability of data.

6.7 Product Advantages

Native Protocol Compatibility

They fully implement the core protocols of the corresponding messaging products with behavior identical to the native products, ensuring smooth migration for business systems.

Enterprise-grade Enhancements

On top of the native capabilities, they add enterprise-grade capabilities such as secure authentication, permission control, audit logs, transport encryption, and enhanced high availability.

Domestic Compliance

They fully adapt to domestic software and hardware ecosystems and support national cryptographic algorithms, meeting government and enterprise domestic replacement and security compliance requirements.

Unified Management

A unified web console provides centralized management and monitoring for different messaging products.

Professional Service Support

Kingdee Apusic provides professional technical support, training, and consulting services to help enterprises use and operate the products smoothly.

6.8 Application Scenarios

Open-source Messaging Product Replacement

In domestic replacement programs across government, finance, telecommunications, and other industries, enterprises need to replace their existing open-source message middleware with secure and controllable commercial products. ADMQ Dedicated Message Queues provide natively compatible alternatives that require no changes to business code.

Domestic Environment Construction

ADMQ Dedicated Message Queues fully adapt to domestic operating systems, chips, and middleware ecosystems, helping enterprises build independent and controllable messaging infrastructure.

Big Data Stream Processing

ADMQ for Kafka connects seamlessly with the big data ecosystem and supports stream computing engines such as Flink, Spark Streaming, and Storm. It is widely used in log collection, data collection, and real-time computing scenarios.

Financial-grade Reliable Messaging

ADMQ for RocketMQ supports transactional messages, ordered messages, message tracing, and dead letter queues, meeting the financial industry's strict requirements for message reliability, consistency, and traceability.

Flexible Message Routing

ADMQ for RabbitMQ supports a variety of exchange and queue types, suiting messaging scenarios that require flexible routing and complex business rules.

Traditional Enterprise Application Migration

ADMQ for ActiveMQ is fully compatible with the JMS specification and supports smooth migration of traditional enterprise Java EE applications to domestic environments.

IoT and Mobile Messaging

ADMQ for MQTT targets IoT and mobile internet scenarios, supporting massive device access, low-bandwidth network transmission, and mobile message push. It is widely used in connected vehicles, the industrial internet, smart homes, smart energy, and other fields.

7 Product Comparison

7.1 Comparison of ADMQ's Two Product Forms

Comparison Dimension	ADMQ Enterprise Message Queue	ADMQ Dedicated Message Queue
Product Form	Unified messaging platform	Specialized message middleware
Deployment Architecture	Compute-storage separation	Dedicated native architecture
Resource Requirements	Higher; suited to large-scale deployment	Lower; suited to lightweight deployment
Scaling Method	Independent scaling of the storage and compute layers	Node-level horizontal scaling
Applicable Scale	Large enterprises / platform scenarios	Department-level / system-level replacement scenarios
Protocol Support	Unified multi-protocol access	Deep native protocol compatibility
Multi-tenancy	Strong multi-tenant isolation	Namespace/Vhost isolation
Cross-region Replication	Built-in cross-region replication	Achieved through tools/mirroring
Primary Scenarios	Unified message hub, complex platforms	Open-source replacement, domestic replacement

7.2 Comparison of ADMQ Dedicated Message Queues

Product	Core Protocol	Typical Features	Applicable Scenarios
ADMQ for Kafka	Kafka protocol	High throughput, partitions, consumer groups	Log collection, stream processing, data synchronization
ADMQ for RocketMQ	RocketMQ protocol	Transactional messages, ordered messages, scheduled messages	Financial transactions, order processing, distributed transactions
ADMQ for RabbitMQ	AMQP protocol	Flexible routing, multiple exchange/queue types	Asynchronous decoupling, peak shaving and valley filling, complex

			routing
ADMQ for ActiveMQ	JMS protocol	Queues/topics, traditional Java EE compatibility	Traditional enterprise applications, Java EE migration
ADMQ for MQTT	MQTT protocol	Massive connections, QoS, rule engine, data bridge, stateless cluster	IoT, connected vehicles, mobile push

8 System Hardware and Software Requirements

8.1 ADMQ Enterprise Message Queue

Component	Minimum Requirements	Recommended Configuration
CPU	8 cores	16 cores or more
Memory	16 GB	32 GB or more
Disk	500 GB SSD	1 TB SSD or more
Network	10 Gbps	25 Gbps
Operating System	CentOS 7+, Ubuntu 18.04+, Kylin, UOS	-

8.2 ADMQ Dedicated Message Queue

Product	CPU	Memory	Disk	Operating System
ADMQ for Kafka	4 cores+	8 GB+	100 GB SSD+	CentOS 7+, Ubuntu 18.04+, Kylin, UOS
ADMQ for RocketMQ	4 cores+	8 GB+	100 GB SSD+	CentOS 7+, Ubuntu 18.04+, Kylin, UOS
ADMQ for RabbitMQ	4 cores+	8 GB+	100 GB SSD+	CentOS 7+, Ubuntu 18.04+, Kylin, UOS
ADMQ for ActiveMQ	4 cores+	8 GB+	100 GB SSD+	CentOS 7+, Ubuntu 18.04+, Kylin, UOS
ADMQ for MQTT	4 cores+	8 GB+	100 GB SSD+	CentOS 7+, Ubuntu 18.04+, Kylin, UOS

Note: The above are minimum hardware specifications.

8.3 Software Environment

- Operating systems: UOS, Kylin, openEuler, Deepin, Zhongke Fangde, CentOS, Ubuntu
- JDK: 1.8+ (required at runtime for RocketMQ/Kafka/ActiveMQ)

8.4 Performance Specifications

Product	Deployment Scale	Throughput	Latency
ADMQ Enterprise Message Engine	Single cluster	100,000+ QPS	Low latency

ADMQ for Kafka	3-node cluster	500,000+ messages/s	P99 < 5 ms
ADMQ for RocketMQ	3-node cluster	300,000+ messages/s	P99 < 3 ms
ADMQ for RabbitMQ	3-node cluster	200,000+ messages/s	P99 < 1 ms
ADMQ for ActiveMQ	3-node cluster	100,000+ messages/s	P99 < 5 ms
ADMQ for MQTT	3-node cluster	1,000,000+ connections	Millisecond-level message latency

9 Customer Cases

9.1 System Decoupling

9.1.1 A Billing Platform

9.1.1.1 Background

This billing platform aggregates major domestic and international payment channels and provides multi-dimensional services such as account management, precision marketing, security risk control, audit and settlement splitting, and billing analytics. The platform handles hundreds of millions in daily revenue, serves 180+ countries (regions), tens of thousands of business code modules, and over 1 million settlement merchants, and manages more than 30 billion custodial accounts — a comprehensive one-stop billing platform.

The transaction engine is the most critical system on the billing platform: every transaction order must be observed by dozens of downstream business systems, including item pricing, item delivery, loyalty points, and stream computing analytics. Because these systems process messages according to different logic, no single system can adapt to every related business. ADMQ makes efficient asynchronous communication and application decoupling possible here, ensuring business continuity.

9.1.1.2 Solution

This billing platform covers 80+ channels with distinct characteristics and 300+ different business logics. A single payment flow often spans many different internal and external systems, so the call chain is relatively long and the probability of exceptions is correspondingly higher — network timeouts in particular (for example, in overseas payment services).

Using message queues and timed message resending, the entire transaction resumes from the breakpoint, ensuring consistency for hundreds of millions of daily transaction requests.

ADMQ's real-time management capabilities, together with a stream computing framework, provide real-time reconciliation and monitoring of billing records, jointly ensuring the timeliness and consistency of the entire transaction.

9.2 Peak Shaving and Valley Filling

9.2.1 An E-commerce Website

9.2.1.1 Background

An e-commerce website was about to launch a new phone, and users with a reservation code could buy it first. To reserve, a user simply registers an account and receives a reservation code; more than 10 million reserving users were expected.

In IO-latency-sensitive scenarios such as Double 11 flash sales and phone reservation rush buys, if external requests exceed the system's processing capacity and the system has no protection in place, the accumulated backlog of timed-out requests can cause every request the system processes to time out and fail. The service capacity presented to the outside then drops to zero, and in that state the service cannot recover on its own.

9.2.1.2 Solution

Message middleware was introduced to make non-immediate business logic asynchronous — for example, the three distinct business steps of receiving a request, processing it, and returning a response.

When the reservation campaign started and a flood of concurrent requests arrived:

- Every customer's reservation request returned success immediately on the page, so customers could close the page and go do something else. The reservation code was pushed to the customer's email or phone later.
- The more than ten million registrations and reservation requests were buffered in the message queue cluster first.
- Backend services then processed the registrations and reservation requests according to the database's actual select, insert, and update capacity.
- Results were returned to users once processing succeeded. After the campaign closed, users received their reservation codes within about 5–30 minutes.

9.3 Internet of Things (IoT)

9.3.1 Smart Sensing for Power Grids

9.3.1.1 Background

This smart-sensing scenario is based mainly on a joint research project with a power grid company on integrating intelligent multi-parameter sensors into transmission lines. The sensors in the project came from different vendors and were distributed at various points along the transmission lines, so the sensor types differed too — more than ten kinds, including towers, on-tower positions, and transmission-line side positions. The system currently covers roughly 600 kilometers of transmission lines, with sensors on more than 600 towers. This scenario mainly provides online monitoring and alerting for data from the various sensors; separately, we also performed transient voltage analysis on the voltage sensors.

This scenario has two difficulties. First, sensors from different vendors use no common communication protocol: some use the power-industry IEC 104 standard, others use protobuf or vendor-specific protocols. Second, the project's data volume is large — some sensors may produce messages of 20 MB or more in a single reading, while others upload data once per second.

9.3.1.2 Solution

With ADMQ, we chose to do no data processing at the producer end and forward the data directly into ADMQ, then use ADMQ Functions for further data preprocessing and other business operations. Taking the voltage sensor as an example, it produces three types of data: heartbeat data, steady-state waveform data, and transient waveform data. The heartbeat and steady-state waveform data are transmitted over the protobuf protocol, while the transient data is transmitted as zipped files. After receiving the protobuf data, we use ADMQ Functions to run a series of processing steps: a decryption function handles data decryption and protobuf deserialization, the data is then routed, the corresponding ETL function performs data processing and parsing, and finally Schema Mapping writes the data into the database.

9.4 Domestic Replacement

In key industries such as government, finance, telecommunications, and energy, ADMQ Dedicated Message Queues help enterprises replace their existing open-source message middleware with secure, controllable, and independently developed domestic commercial products. They support domestic operating systems, chips, and middleware ecosystems, meeting enterprises' compliance requirements.

UNIFIED SERVICE HOTLINE

4008-555-800

Kingdee Apusic Cloud Computing Co., Ltd. (referred to as “Kingdee Apusic”) was founded in 2000. Its predecessor was “Kingdee Middleware Company.” It is a basic software platform provider under Kingdee Group, a national standards-setting enterprise for cloud computing, and a core software enterprise in the national IT application innovation industry. Kingdee Apusic is an enterprise undertaking the National 863 Key R&D Program and the National Major Science and Technology Project for Core Electronic Components, High-end General-purpose Chips and Fundamental Software, and is also a basic platform provider for the national key projects of “Two Networks, One Website, Four Databases, and Twelve Golden Projects.” Its products are widely used in key industries such as government, military, finance, and energy, and it has cumulatively served more than 100,000 customers.



National standard setter for cloud computing
Basic software enterprise under Kingdee Group
Core enterprise for IT application innovation
Website: www.apusic.com

