Generated Date: 2026-10-19

Migration Guide

金蝶Apusic分布式缓存 V2.0.5

Copyright © Kingdee Apusic Cloud Computing Co., Ltd. Shenzhen 2026. All Rights Reserved.

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

Apusic 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 Preface
  - 1.1 Intended Audience
  - 1.2 Related Documents
  - 1.3 Technical Support
- 2 Overview
- 3 Migrating AMDC V2.0.2 and Earlier Versions to V2.0.5
  - 3.1 Preparation
    - 3.1.1 Environment Check
    - 3.1.2 Tool and File Preparation
      - 3.1.2.1 Required Tools
      - 3.1.2.2 Collecting Files to Migrate
  - 3.2 Standalone Mode Migration
    - 3.2.1 Migration Prerequisites
    - 3.2.2 Preparation
      - 3.2.2.1 Tools and Configuration Prerequisites
      - 3.2.2.2 Back Up Data and Stop the AMDC V2.0.2 or Earlier Standalone Node (Full Downtime)
    - 3.2.3 Migration Steps
      - 3.2.3.1 Configuration File Migration
      - 3.2.3.2 Cache Data Migration
      - 3.2.3.3 Starting the AMDC V2.0.5 Service
      - 3.2.3.4 Verifying the Startup
    - 3.2.4 Notes
  - 3.3 Master-Slave Mode Migration
    - 3.3.1 Migration Prerequisites
    - 3.3.2 Preparation
      - 3.3.2.1 Tools and Configuration Prerequisites
      - 3.3.2.2 Back Up Data and Stop the AMDC V2.0.2 or Earlier Master-Slave Cluster (Full Downtime)
    - 3.3.3 Migration Steps (Execute in the Order of "Master Node -> All Slave Nodes")
      - 3.3.3.1 Master Node Migration
      - 3.3.3.2 Slave Node Migration (Execute on All Slave Nodes Simultaneously)

- 3.3.4 Notes
- 3.4 Sentinel Mode Migration
  - 3.4.1 Migration Prerequisites
  - 3.4.2 Preparation
    - 3.4.2.1 Tools and Configuration Prerequisites
    - 3.4.2.2 Back Up Data and Fully Stop the AMDC V2.0.2 or Earlier Sentinel + Master-Slave Cluster
  - 3.4.3 Migration Steps (Execute in the Order of "AMDC V2.0.5 Master-Slave Cluster -> AMDC V2.0.5 Sentinel Cluster")
    - 3.4.3.1 Rebuild the AMDC V2.0.5 Master-Slave Cluster (Critical, Same as the Complete Steps in the Previous Chapter)
    - 3.4.3.2 Sentinel Node Migration (Execute on All Sentinel Nodes Simultaneously)
  - 3.4.4 Notes
- 3.5 Cluster Mode Migration
  - 3.5.1 Migration Prerequisites
  - 3.5.2 Preparation
    - 3.5.2.1 Tools and Configuration Prerequisites
    - 3.5.2.2 Unify the Environment and Record the Source Topology
    - 3.5.2.3 Back Up Data and Fully Stop the AMDC V2.0.2 or Earlier Cluster (No Writes Allowed)
    - 3.5.2.4 Data File Preprocessing (Execute on All Source Nodes)
  - 3.5.3 Migration Steps (Execute in the Order of "Configuration Conversion -> AMDC V2.0.5 Cluster Initialization -> Shard Data Import -> Cluster Status Verification")
    - 3.5.3.1 Initialize the AMDC V2.0.5 Empty Cluster

3.5.3.1.1 Verify that all nodes started successfully (the log shows "Ready to accept connections")

    - 3.5.3.2 Shard Data Import
    - 3.5.3.3 Cluster Status Repair and Data Verification
  - 3.5.4 Notes
- 4 Migrating Redis 8.6 and Earlier Versions to AMDC V2.0.5
  - 4.1 Preparation
    - 4.1.1 Environment Check
    - 4.1.2 Tool and File Preparation
  - 4.2 Standalone Mode Migration
    - 4.2.1 Migration Prerequisites

- 4.2.2 Preparation
  - 4.2.2.1 Data and Configuration Backup
  - 4.2.2.2 Back Up Data and Stop the Redis Standalone Node
- 4.2.3 Migration Steps
  - 4.2.3.1 Cache Data Migration
  - 4.2.3.2 Modifying the redis.conf Configuration
  - 4.2.3.3 Starting the AMDC Service
  - 4.2.3.4 Verifying the Startup
- 4.2.4 Notes
- 4.3 Master-Slave Mode Migration
  - 4.3.1 Migration Prerequisites
  - 4.3.2 Preparation
    - 4.3.2.1 Tool and File Preparation
    - 4.3.2.2 Back Up Data and Stop the Redis Master-Slave Cluster (Full Downtime)
  - 4.3.3 Migration Steps (Execute in the Order of "Master Node -> All Slave Nodes")
    - 4.3.3.1 Cache Data Migration
    - 4.3.3.2 Configuration File Migration (redis.conf)
    - 4.3.3.3 Starting the AMDC Master-Slave Node Services
    - 4.3.3.4 Master-Slave Relationship Verification
  - 4.3.4 Notes
- 4.4 Sentinel Mode Migration
  - 4.4.1 Migration Prerequisites
  - 4.4.2 Preparation
    - 4.4.2.1 Tool and File Preparation
    - 4.4.2.2 Back Up Data and Fully Stop the Redis Sentinel + Master-Slave Cluster
  - 4.4.3 Migration Steps
    - 4.4.3.1 Cache Data Migration
    - 4.4.3.2 Configuration File Migration (redis.conf)
    - 4.4.3.3 Starting the AMDC Master-Slave Node Services
    - 4.4.3.4 Starting the AMDC Sentinel Node Services
    - 4.4.3.5 Sentinel Functionality Verification
  - 4.4.4 Notes
- 4.5 Cluster Mode Migration
  - 4.5.1 Migration Prerequisites

- 4.5.2 Preparation
  - 4.5.2.1 Tool and File Preparation
  - 4.5.2.2 Back Up Data and Fully Stop the Redis Cluster (No Writes Allowed)
- 4.5.3 Migration Steps
  - 4.5.3.1 Data File Preprocessing (Execute on All Source Nodes)
  - 4.5.3.2 Configuration File Migration (redis.conf, node.conf)
  - 4.5.3.3 Starting the AMDC Cluster Nodes
  - 4.5.3.4 Cluster Status Verification
- 4.5.4 Notes
- 5 Troubleshooting Common Issues
  - 5.1 Tool Cannot Run
  - 5.2 File Verification or Repair Fails
  - 5.3 amdc-cli Cannot Connect to the AMDC Service
- 6 Summary
  - 6.1 Migration Process from AMDC V2.0.2 and Earlier Versions to AMDC V2.0.5
  - 6.2 Migration Process from Redis 8.6 and Earlier Versions to AMDC V2.0.5

1 Preface

This document is the migration guide for Apusic In-Memory Data Cache (AMDC) V2.0.5. It describes in detail how to migrate from Redis to AMDC, and how to migrate earlier AMDC versions to V2.0.5.

1.1 Intended Audience

This document is intended for AMDC product operations engineers, IT system operations engineers, development engineers, software architects, R&D managers, and other personnel.

1.2 Related Documents

For more information about AMDC V2.0.5, please refer to the following AMDC V2.0.5 product manual set:

No.	Manual	Description
1	Apusic In-Memory Data Cache V2.0.5 Quick Start Guide	Briefly introduces how to quickly get started with AMDC.
2	Apusic In-Memory Data Cache V2.0.5 Installation Guide	Describes in detail how to install AMDC on various operating systems, as well as AMDC service start/stop operations and the product registration process.
3	Apusic In-Memory Data Cache V2.0.5 Cache Core User Manual	Describes in detail the use, configuration, and management of AMDC features and the usage of supporting tools.
4	Apusic In-Memory Data Cache V2.0.5 Console User Manual	Describes in detail the use of AMDC console features and operation instructions.
5	Apusic In-Memory Data Cache V2.0.5 Development Guide	Describes in detail how to develop AMDC client applications based on various programming languages.
6	Apusic In-Memory Data Cache V2.0.5 Migration Guide	Describes how to migrate and upgrade earlier AMDC versions to V2.0.5, as well as how to migrate from Redis to AMDC.
7	Apusic In-Memory Data Cache V2.0.5 Operations Guide	Describes AMDC monitoring, operations, security hardening, and other operations instructions.
8	Apusic In-Memory Data Cache V2.0.5 Performance Tuning Guide	Describes AMDC performance tuning.

1.3 Technical Support

AMDC provides comprehensive technical support services. You can obtain technical support in the following ways:

- Website: www.apusic.com
- Phone: 400-855-5800
- Email: support@apusic.com
- Kingdee Cloud Community: https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

When requesting technical support, please provide the following information:

1. Your name
2. Company information and contact details
3. Operating system and its version
4. Product version number
5. Detailed information such as logs and screenshots of any exceptions or errors

2 Overview

This manual describes two migration procedures to AMDC V2.0.5:

1. Migrating AMDC V2.0.2 and earlier versions to AMDC V2.0.5: use amdc-conf-conv to convert the configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format, reuse the original configuration and data files, and verify functionality after startup;
2. Migrating Redis 8.6 and earlier versions to AMDC V2.0.5: relies on AMDC's ability to directly load Redis configurations (redis.conf, etc.) and data files (RDB/AOF, etc.).

3 Migrating AMDC V2.0.2 and Earlier Versions to V2.0.5

3.1 Preparation

3.1.1 Environment Check

- AMDC V2.0.5 is installed (must match the tool version), and the installation directory permissions are normal (readable and executable);
- The AMDC V2.0.2 or earlier service to be migrated is accessible, or its RDB cache files and configuration files (YAML format) have been obtained;
- The migration target server (the server where AMDC V2.0.5 is deployed) has sufficient disk space to store the migrated cache data and configuration files;

3.1.2 Tool and File Preparation

3.1.2.1 Required Tools

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);
- amdc-conf-conv: used to convert configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format (deployed in the same directory);

3.1.2.2 Collecting Files to Migrate

- Configuration files of AMDC V2.0.2 and earlier versions: server configuration (such as conf.yaml) or Sentinel configuration (such as sentinel_go.yaml);
- RDB snapshot files of AMDC V2.0.2 and earlier versions: named dump.rdb by default, usually located in the installation directory (confirm the actual path);
- If AOF persistence is enabled in AMDC V2.0.2 and earlier versions, prefer the AOF files as the migration data source, with the RDB files as the backup migration data source;

3.2 Standalone Mode Migration

Standalone mode migration applies to scenarios where only a single AMDC service is deployed. The core process is "file preparation -> configuration conversion -> data verification and repair -> deployment and startup ->

verification". The operation is simple with no dependencies. The specific steps are as follows:

3.2.1 Migration Prerequisites

- AMDC V2.0.5 is installed on the target server, and the installation directory permissions are normal (readable and executable);
- The configuration file (such as conf.yaml) and RDB snapshot file (dump.rdb) of AMDC V2.0.2 and earlier versions have been collected; if AOF persistence is enabled, obtain the AOF files first;
- The target server has sufficient disk space, at least 2 times the size of the source data files (reserving space for backup and operation).

3.2.2 Preparation

3.2.2.1 Tools and Configuration Prerequisites

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);
- amdc-conf-conv: used to convert configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format (deployed in the same directory);

3.2.2.2 Back Up Data and Stop the AMDC V2.0.2 or Earlier Standalone Node (Full Downtime)

- After stopping write operations on the AMDC V2.0.2 or earlier standalone node, run the bgsave command again to generate the latest RDB file; back up the source AMDC configuration;
- The AMDC V2.0.2 or earlier standalone node must be stopped first, and any data writes are forbidden, to ensure the final consistency of the source data files.

```
# Execute on the node: stop the AMDC process of AMDC V2.0.2 and earlier versions
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs kill -9

# Verify the process has stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.2.3 Migration Steps

3.2.3.1 Configuration File Migration

Use the amdc-conf-conv tool to convert the server configuration of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format, as follows:

1. Back up the default AMDC V2.0.5 configuration file on the target server (to avoid loss by overwriting):

```
cp /amdc/amdc.yaml /amdc/amdc.yaml.bak
```

2. Perform the configuration conversion (specify the --server parameter to adapt to server configuration):

```
amdc-conf-conv --server --go-yaml /path/to/conf.yaml --c-yaml
/amdc/amdc.yaml
```

3. Verify the conversion result: open /amdc/amdc.yaml and confirm that the core configuration items (such as port, password, data directory, memory limit, etc.) have been inherited correctly with no syntax errors;

3.2.3.2 Cache Data Migration

1. Verify and repair the source RDB or AOF files (process AOF files first, as the data is more complete):

```
# If using AOF files
amdc-check-aof /path/to/appendonlydir/appendonly.aof.manifest

# If using RDB files
amdc-check-rdb /path/to/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof

amdc-check-rdb --fix /path/to/dump.rdb
```

2. Copy the repaired files to the AMDC V2.0.5 installation directory:

```
# Check the AMDC V2.0.5 data directory (confirm via the dir configuration item in amdc.yaml; default path as follows), and copy the RDB file
cp /path/to/dump.rdb/ amdc/

# If using AOF files, first enable the AOF configuration (set appendonly yes in amdc.yaml), then copy
cp /path/to/appendonlydir /amdc/
```

3.2.3.3 Starting the AMDC V2.0.5 Service

1. Start the AMDC V2.0.5 service, loading the converted configuration file:

```
amdc-server /amdc/amdc.yaml
```

2. Verify the service startup status:

```
# View the service process
ps -ef | grep amdc-server

# View the log (default log path: /amdc/server.log)
tail -100f /amdc/server.log
```

3.2.3.4 Verifying the Startup

1. Connect to the AMDC V2.0.5 service and verify data integrity:

```
# Connect using amdc-cli (default port 6359; add the -a option if a password is configured)
amdc-cli -p 6359 -a test123

# View keyspace statistics
info keyspace
```

2. Verify functional availability: perform write, modify, delete, and other operations, and confirm that the service responds normally with no configuration- or data-related errors:

3.2.4 Notes

- During migration, the AMDC V2.0.2 or earlier service must be stopped to avoid source file corruption caused by data writes;
- After migration, observe the service running for 10-20 minutes and confirm there are no log errors or other exceptions.

3.3 Master-Slave Mode Migration

Master-slave mode migration applies to scenarios where AMDC V2.0.2 and earlier versions cannot establish master-slave replication across versions with AMDC V2.0.5. The core principle is full downtime -> migrate the master node first -> then migrate the slave nodes -> data consistency verification. Migration is completed by importing full data via RDB or AOF, and a short downtime is required throughout (there is no cross-version synchronization phase). The specific steps are as follows:

3.3.1 Migration Prerequisites

- AMDC V2.0.5 is installed on both the master and slave target servers, the installation directory permissions are normal (readable, writable, and executable), and the master and slave nodes can communicate over the network (no firewall blocking port 6359);
- The configuration files (conf.yaml), AOF files (preferred, as the data is more complete), or RDB snapshot files of the AMDC V2.0.2 or earlier master and slave nodes have been collected, with a full backup made;
- Inform the business side of the downtime window in advance; it is recommended to operate during off-peak hours. The downtime is approximately the time for "data file copy + service startup" (adjust according to the data volume, generally 5-30 minutes);
- The target server has sufficient disk space, at least 2 times the size of the source data files (reserving space for backup and operation).

3.3.2 Preparation

3.3.2.1 Tools and Configuration Prerequisites

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);

- amdc-conf-conv: used to convert configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format (deployed in the same directory);

3.3.2.2 Back Up Data and Stop the AMDC V2.0.2 or Earlier Master-Slave Cluster (Full Downtime)

- After stopping write operations on the AMDC V2.0.2 or earlier nodes, run the bgsave command again to generate the latest RDB file; back up the source AMDC configuration;
- All AMDC V2.0.2 or earlier master and slave nodes must be stopped first, and any data writes are forbidden, to ensure the final consistency of the source data files.

```
# Execute uniformly on the master node + all slave nodes: stop the AMDC processes of AMDC V2.0.2 and earlier versions
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.3.3 Migration Steps (Execute in the Order of "Master Node -> All Slave Nodes")

3.3.3.1 Master Node Migration

The master node is the data core. Configuration conversion, data verification and import, and service startup must be completed first, as it serves as the synchronization source for the subsequent slave nodes.

Step 1: Configuration File Conversion and Optimization

1. Back up the default configuration of the AMDC V2.0.5 master node (to avoid overwriting):

```
cp /amdc/amdc.yaml /amdc/amdc-master.yaml.bak
```

2. Convert the master node configuration of AMDC V2.0.2 and earlier versions to the dedicated AMDC V2.0.5 master node configuration (named amdc-master.yaml for easy distinction):

```
amdc-conf-conv --server --go-yaml /path/to/conf.yaml --c-yaml /amdc/amdc-master.yaml
```

3. Check the AMDC V2.0.5 master node configuration, enable the master identity (no replicaof configuration), and confirm the core configuration items:

```
vi /amdc/amdc-master.yaml
```

Key configuration confirmation/modification:

- port: 6359 (unified port; if it needs to be changed, synchronize all nodes);
- requirepass: your password (if the source master node has a password, keep it; slave nodes will need to configure masterauth later);
- dir: /amdc/ (data directory, consistent with the actual plan);
- appendonly: yes (it is recommended to enable AOF, consistent with the source; set to no if the source uses only RDB);
- logfile: amdc-master.log (independent log for the master node, convenient for troubleshooting).

Step 2: Data File Verification, Repair, and Import

1. Verify and repair the source AMDC V2.0.2 or earlier master node data files (executed on the master node; skip repair if the files are not corrupted):

```
# If using AOF files (recommended)
amdc-check-aof /path/to/appendonlydir/appendonly.aof.manifest

# If using RDB files
amdc-check-rdb /path/to/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof
```

```
amdc-check-rdb --fix /path/to/dump.rdb
```

2. Copy the repaired clean data files to the AMDC V2.0.5 master node data directory and modify permissions:

```
# Copy the AOF files (recommended; if using RDB, execute the next line instead)
cp /path/to/appendonlydir/ /amdc/

# Copy the RDB file
cp /path/to/dump.rdb /amdc/

# Grant permissions (to prevent the service from being unable to read)
chmod 644 /amdc/{appendonlydir/, dump.rdb}
```

Step 3: Start the AMDC V2.0.5 Master Node Service

```
# Start the master node with the dedicated configuration file
amdc-server /amdc/amdc-master.yaml

# Verify the master node startup status
ps -ef | grep amdc-server | grep master # View the process
tail -100f /amdc/amdc-master.log # View the log
```

The log showing "Ready to accept connections" indicates that the master node has started successfully;

Connect to the master node to verify data integrity (critical, to ensure the data import succeeded):

```
amdc-cli -p 6359 -a <password>  # Connect to the master node

info keyspace  # View keyspace statistics (confirm the data volume)
```

3.3.3.2 Slave Node Migration (Execute on All Slave Nodes Simultaneously)

Slave nodes do not need to import data separately. After startup, they pull the full data from the already-started AMDC V2.0.5 master node through AMDC V2.0.5 internal master-slave synchronization, ensuring the data is completely consistent with the master node.

Step 1: Configuration File Conversion and Master-Slave Association

1. Back up the default configuration of the AMDC V2.0.5 slave node:

```
cp /amdc/amdc.yaml /amdc/amdc-slave.yaml.bak
```

2. Convert the slave node configuration of AMDC V2.0.2 and earlier versions to the dedicated AMDC V2.0.5 slave node configuration (named amdc-slave.yaml):

```
amdc-conf-conv --server --go-yaml /path/to/conf.yaml --c-yaml
/amdc/amdc-slave.yaml
```

3. Check the AMDC V2.0.5 slave node configuration, specify the master node information, and enable the slave identity:

```
vi /amdc/amdc-slave.yaml
```

Key configuration confirmation/modification:

- port: 6360 (unified port; if it needs to be changed, synchronize all nodes);
- requirepass: your password (if the source master node has a password, keep it; slave nodes will need to configure masterauth later);
- dir: /amdc/ (data directory, consistent with the actual plan);
- appendonly: yes (it is recommended to enable AOF, consistent with the source; set to no if the source uses only RDB);
- replicaof: # points to the IP and port of the AMDC V2.0.5 master node;
- logfile: amdc-slave.log (independent log for the slave node, convenient for troubleshooting).

Step 2: Start the AMDC V2.0.5 Slave Node Services

No source data files need to be copied to the slave node data directory; the AMDC V2.0.5 slave nodes automatically synchronize from the master node:

```
# Start the slave node with the dedicated configuration file
amdc-server /amdc/amdc-slave.yaml

# Verify the slave node startup status
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/amdc-slave.log  # View the log
```

The log showing "MASTER <-> REPLICA sync: Finished with success" indicates that the slave node has successfully connected to the master node and started synchronizing.

Step 3: Verify the Slave Node Synchronization Status

Connect to a slave node and confirm that synchronization is complete and the data is consistent with the master node:

```
amdc-cli -p 6360 -a <password>  # Connect to the slave node

info replication  # View the synchronization status
```

Key status confirmations:

- role: slave: the slave node identity is normal;
- master_host: the master node IP, masterport: 6359: the master node pointing is correct;
- master_link_status: up: the master-slave connection is normal.

3.3.4 Notes

- Throughout the migration, all AMDC V2.0.2 or earlier master-slave nodes must be fully stopped first, and data writes are forbidden; otherwise the source data files will be inconsistent and data will be lost after import;
- After importing the master node data, data integrity must be verified before starting the slave nodes, to prevent slave nodes from synchronizing erroneous data;

- Slave nodes do not need any source data files copied; this is completed by AMDC V2.0.5 internal master-slave synchronization. Manual copying will cause synchronization conflicts;
- The core configurations of the master and slave nodes must be consistent (AOF switch, port, password, etc.), otherwise connection failures or synchronization exceptions will occur;
- After migration, observe the cluster running for 10-20 minutes and confirm that master-slave synchronization has no delay, there are no log errors, and business access is normal.

3.4 Sentinel Mode Migration

For scenarios where AMDC V2.0.2 and earlier versions cannot perform master-slave synchronization across versions with AMDC V2.0.5 and Sentinels cannot monitor cross-version nodes, the core principle is full downtime -> migrate the master-slave nodes first (same as Chapter 6) -> then migrate the Sentinel nodes -> re-associate monitoring. Migration is completed through "AMDC V2.0.5 master-slave cluster rebuild + Sentinel re-monitoring", and a short business downtime is required throughout. The specific steps are as follows:

3.4.1 Migration Prerequisites

- AMDC V2.0.5 is installed on all master-slave nodes and Sentinel nodes, and the network is interconnected (open service port 6359 and Sentinel port 26359);
- The master-slave node configurations, data files, and Sentinel configuration files (sentinel_go.yaml) of AMDC V2.0.2 and earlier versions have been collected, with a full backup made;
- Plan the downtime window in advance (off-peak hours). The downtime is the time for "AMDC V2.0.2 and earlier shutdown + AMDC V2.0.5 master-slave startup + Sentinel startup" (generally 10-40 minutes);
- All target nodes are time-synchronized and have sufficient disk space (reserve 2 times the source data space in the master node data directory).

3.4.2 Preparation

3.4.2.1 Tools and Configuration Prerequisites

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);
- amdc-conf-conv: used to convert configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format (deployed in the same directory);

3.4.2.2 Back Up Data and Fully Stop the AMDC V2.0.2 or Earlier Sentinel + Master-Slave Cluster

- After stopping write operations on the AMDC V2.0.2 or earlier nodes, run the bgsave command again to generate the latest RDB file; back up the source AMDC configuration;
- All AMDC V2.0.2 or earlier Sentinel nodes and master-slave nodes must be stopped first, and data writes and Sentinel monitoring are forbidden, to ensure source data consistency.

```
# Execute uniformly on all nodes: stop the AMDC master-slave +
Sentinel processes of AMDC V2.0.2 and earlier versions
ps -ef | grep -E 'amdc-server | amdc-sentinel' | grep -v grep | awk
'{print $2}' | xargs kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep -E 'amdc-server | amdc-sentinel' | grep -v grep
```

3.4.3 Migration Steps (Execute in the Order of "AMDC V2.0.5 Master-Slave Cluster -> AMDC V2.0.5 Sentinel Cluster")

3.4.3.1 Rebuild the AMDC V2.0.5 Master-Slave Cluster (Critical, Same as the Complete Steps in the Previous Chapter)

- Strictly follow all the steps of master-slave mode migration in the previous chapter to complete the setup of the AMDC V2.0.5 master-slave cluster:
- Migrate the AMDC V2.0.5 master node: configuration conversion -> data verification and import -> startup -> verify data integrity;
- Migrate all AMDC V2.0.5 slave nodes: configuration conversion (associate with the master node) -> startup -> verify master-slave synchronization;
- Complete the overall verification of the master-slave cluster: data consistency, master-slave functionality, and no log exceptions.

This step is the foundation of Sentinel migration. The AMDC V2.0.5 master-slave cluster must be running completely normally before proceeding with the subsequent Sentinel migration.

3.4.3.2 Sentinel Node Migration (Execute on All Sentinel Nodes Simultaneously)

Sentinel nodes do not need to import data. You only need to convert the Sentinel configuration of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format, and change the monitoring target to the already-built

AMDC V2.0.5 master node.

Step 1: Configuration File Conversion and Monitoring Target Modification

1. Back up the default AMDC V2.0.5 Sentinel configuration:

```
cp /amdc/sentinel.yaml /amdc/sentinel-c.yaml.bak
```

2. Convert the Sentinel configuration of AMDC V2.0.2 and earlier versions to the dedicated AMDC V2.0.5 Sentinel configuration (named sentinel-c.yaml):

```
amdc-conf-conv --sentinel --go-yaml /path/to/sentinel_go.yaml --c-yaml /amdc/sentinel.yaml
```

3. Check the AMDC V2.0.5 Sentinel configuration and confirm the key Sentinel configuration:

```
vi /amdc/sentinel.yaml
```

Key configuration modification or confirmation (the core is the monitor configuration item, pointing to the AMDC V2.0.5 master node):

```
# Core Sentinel monitoring configuration (replace with the AMDC V2.0.5 master node information; keep the remaining parameters consistent with the source)

SENTINEL:

sentinel monitor: mymaster 10.0.1.100 (example IP; replace with the actual IP) 6359 2 # mymaster: monitoring name (consistent with the source), followed by the master node IP, port, and quorum threshold

sentinel down-after-milliseconds: mymaster 30000  # Subjective down timeout (consistent with the source)

sentinel failover-timeout: mymaster 180000  # Failover timeout (consistent with the source)
```

```
sentinel auth-pass: mymaster <password>   # Consistent with the
master node's requirepass; omit if there is no password
```

Other configuration confirmations:

- port: 26359 (Sentinel default port, consistent with the source);
- logfile: sentinel.log (independent Sentinel log);
- dir: /amdc/ (Sentinel data directory, storing monitoring state).

```
# Start all Sentinel nodes simultaneously with the dedicated
configuration file
amdc-sentinel /amdc/sentinel.yaml

# Verify the Sentinel startup status
ps -ef | grep amdc-sentinel   # View the process
tail -100f /amdc/sentinel.log   # View the log
```

Step 2: Start the AMDC V2.0.5 Sentinel Services

The log showing "Sentinel ID is xxxxxxxxxxxxxxxxxxxxxxx" indicates that the Sentinel has started successfully.

Key verification: the Sentinel has successfully discovered all nodes of the AMDC V2.0.5 master-slave cluster

```
# Connect to the Sentinel port (26359) and query the monitoring
status

amdc-cli -p 26359

info sentinel  # View Sentinel monitoring information

sentinel slaves mymaster  # View the list of monitored slave nodes
```

Verification result requirements:

- The IPs and ports shown in the slave node list are all AMDC V2.0.5 slave node information, with status online.

3.4.4 Notes

- Before migration, all AMDC V2.0.2 or earlier Sentinel + master-slave nodes must be fully stopped, to prevent Sentinels from continuously monitoring stopped nodes, or data writes from causing source file inconsistency;
- The AMDC V2.0.5 master-slave cluster must be set up and verified before starting the Sentinel nodes; otherwise the Sentinels will not be able to monitor valid nodes, causing the migration to fail;
- It is recommended that the monitoring name (such as mymaster), quorum threshold, and timeout in the Sentinel configuration remain consistent with the source, to reduce client configuration changes;
- The monitoring configuration of all Sentinel nodes must be completely consistent (monitoring target, password, thresholds, etc.), otherwise the Sentinel cluster will split and failover will be abnormal;
- After migration, observe for 10-20 minutes, focusing on: no errors in the Sentinel log, no abnormalities in master-slave switching, synchronization delay within an acceptable range, and no interruption of business access.

3.5 Cluster Mode Migration

Cluster mode migration applies to scenarios where AMDC V2.0.2 and earlier versions cannot establish a cluster across versions with AMDC V2.0.5 and cannot synchronize shard data across versions. The core principle is full downtime -> full export of shard data -> AMDC V2.0.5 cluster initialization -> batch import of shard data -> cluster verification. Because a cluster is a multi-master multi-slave sharding architecture, cross-version synchronization is not possible; the entire cluster must be migrated through "full data export-import", and business downtime is required throughout. The specific steps are as follows:

3.5.1 Migration Prerequisites

- AMDC V2.0.5 is installed on all nodes of the target cluster (same number of nodes as the source cluster: M masters and N slaves), and the network is interconnected (open service port 6359 and cluster gossip communication port 16359);
- The configuration files, AOF, and RDB data files of all nodes of the AMDC V2.0.2 or earlier cluster have been collected, and the source cluster topology information has been recorded (execute cluster nodes to obtain slot allocation and master-slave relationships);
- Plan a long downtime window in advance (adjust according to the data volume; off-peak hours recommended). The downtime is the time for "data export + cluster initialization + data import +

verification" (approximately 30-60 minutes for data volumes within 10G);

- All target nodes have sufficient disk space (reserve 2 times the source node data space on each node), are time-synchronized, and have ports 6359 and 16359 allowed through the firewall.

3.5.2 Preparation

3.5.2.1 Tools and Configuration Prerequisites

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);
- amdc-conf-conv: used to convert configuration files of AMDC V2.0.2 and earlier versions to an AMDC V2.0.5-compatible format (deployed in the same directory);

3.5.2.2 Unify the Environment and Record the Source Topology

1. Create a unified directory on all target nodes and grant permissions:
2. Observe the core topology of the AMDC V2.0.2 or earlier cluster:

Connect to any node of the AMDC V2.0.2 or earlier cluster, execute the following command, and save the output (it will be fully reused in the subsequent AMDC V2.0.5 cluster initialization):

```
amdc-cli cluster nodes  # Record: master-slave node IPs and ports,
node IDs, slot allocation (e.g., 0-5460)
```

Or view the node.conf file, which records the cluster node IPs, ports, node IDs, slot allocation, and other information.

```
29755a26cc60c056b3af39eb66dc5ccbfbdd20c1 127.0.0.1:6383@16383 slave
882b57434ef8c361fcea44c327379880181afla6 0 1769505692837 2 connected

882b57434ef8c361fcea44c327379880181afla6 127.0.0.1:6380@16380 master
- 0 1769505692432 2 connected 5461-10922

980c1df981902c2df617ca81d4df7219b1e86a8b 127.0.0.1:6384@16384
myself,slave 2793d6ad527eee582aa30890d7c8407e9c112b4f003 connected
```

```
43d25d48fac16f9b8f6b519f88a889179b8a6e87 127.0.0.1:6382@16382 slave
ac9f0c10e742b4570f6436db8af281c1d51ea570 0 1769505692331 1 connected

2793d6ad527eee582aa30890d7c8407e9c112b4f 127.0.0.1:6381@16381 master
- 0 1769505692332 3 connected 10923-16383

ac9f0c10e742b4570f6436db8af281c1d51ea570 127.0.0.1:6379@16379 master
- 0 1769505692228 1 connected 0-5460
```

Record the IP and port of each master node of AMDC V2.0.2 and earlier versions, and back up the AOF or RDB files on the corresponding machines for subsequent data migration to the AMDC V2.0.5 cluster:

3.5.2.3 Back Up Data and Fully Stop the AMDC V2.0.2 or Earlier Cluster (No Writes Allowed)

- After stopping write operations on the AMDC V2.0.2 or earlier nodes, run the bgsave command again to generate the latest RDB file; back up the source AMDC configuration;

```
# Execute uniformly on all cluster nodes of AMDC V2.0.2 and earlier
versions: stop the processes
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

3.5.2.4 Data File Preprocessing (Execute on All Source Nodes)

Verify and repair the AOF or RDB files of each AMDC V2.0.2 or earlier master node to ensure clean data files, avoiding data corruption after import into AMDC V2.0.5:

```
# Prefer AOF files (process each node separately; the cluster uses
sharded storage and each node's data is independent)
amdc-check-aof /path/to/nodeX/appendonlydir/appendonly.aof.manifest

# If there is no AOF, use the RDB file
amdc-check-rdb /path/to/nodeX/dump.rdb
```

```
# If a file is corrupted, perform repair
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.rdb /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.base.aof /
amdc-check-aof --fix
/path/to/appendonlydir/appendonly.aof.1.incr.aof

amdc-check-rdb --fix /path/to/nodeX/dump.rdb
```

3.5.3 Migration Steps (Execute in the Order of "Configuration Conversion -> AMDC V2.0.5 Cluster Initialization -> Shard Data Import -> Cluster Status Verification")

3.5.3.1 Initialize the AMDC V2.0.5 Empty Cluster

Step 1: Configuration Conversion and Cluster Enablement on All AMDC V2.0.5 Nodes

1. Configuration conversion:

```
amdc-conf-conv --server --go-yaml /path/to/nodeX/conf.yaml --c-yaml
/amdc/amdc.yaml
```

2. Check the configuration file:

```
vi /amdc/amdc.yaml
```

Key configuration:

- port: 6359 (cluster service port, consistent with the source);
- logfile: amdc-cluster-XXX.log (independent log for each node);
- dir: /amdc/ (unified data directory);
- requirepass: password (consistent with the source, the same on all nodes);
- cluster-enabled: yes (enable Cluster mode);
- cluster-config-file: nodes-XXX.conf (a separate cluster configuration file for each node);
- cluster-node-timeout: 15000 (cluster node timeout, consistent with the source);

- cluster-require-full-coverage: yes (enable full slot coverage check).

3. After all nodes are configured, start all AMDC V2.0.5 nodes (only start the services; the cluster is not yet formed):

```
amdc-server /amdc/amdc-cluster-XXX.yaml
```

3.5.3.1.1 Verify that all nodes started successfully (the log shows "Ready to accept connections")

Step 2: Perform Cluster Initialization (Slot + Master-Slave Allocation)

Select any AMDC V2.0.5 master node and execute the amdc-cli --cluster create command, strictly specifying the node IPs, ports, and slot allocation according to the source topology. The example uses 3 masters and 3 slaves (source topology slots 0-5460, 5461-10922, 10923-16383):

```
# Command format: amdc-cli --cluster create master1 master2 master3
slave1 slave2 slave3 --cluster-replicas 1
# --cluster-replicas 1: means 1 master and 1 slave, consistent with
the source

amdc-cli --cluster create 10.0.1.10 (example IP; replace with the
actual IP):6359 10.0.1.11 (example IP; replace with the actual
IP):6359 192.168.1.12:6359 192.168.1.13:6359 192.168.1.14:6359
192.168.1.15:6359 --cluster-replicas 1
```

After successful execution, verify the cluster initialization status:

```
amdc-cli -p 6359 -a <password> cluster nodes
```

Requirements: all slots are properly allocated to the cluster nodes, all nodes are in myself, master, or slave status, and no node is in fail status.

3.5.3.2 Shard Data Import

First start an AMDC V2.0.5 standalone node to load the RDB or AOF file, then use the AMDC V2.0.5 amdc-cli --cluster import command to automatically remap and import the data from the standalone node into the cluster. This command is the standardized cluster import method; there is no need to manually manipulate data files, avoiding issues such as file permissions and format incompatibility.

```
# Core command format (executed on one line; the line break \ is only for formatting)

amdc-cli --cluster import <cluster master node IP>:<cluster master node port> \

--cluster-from <standalone node IP>:<standalone node port> \

--cluster-copy \

--cluster-replace \

-a <cluster password>
```

Parameter details:

- --cluster-copy: a required parameter; preserves the original data of the source standalone node when copying data, preventing the source data from being deleted;
- --cluster-replace: an optional parameter; if keys with the same name as those on the source node exist in the cluster, they are overwritten directly (it is recommended to add this during migration to ensure the data is consistent with the source; choose according to business needs).

Because the cluster uses sharded storage, each source shard of AMDC V2.0.2 and earlier versions only stores the data of its corresponding slots. The import command must be executed shard by shard in the order of "source shard master node -> AMDC V2.0.5 corresponding shard master node". The whole process can be executed on any operable node of the AMDC V2.0.5 cluster; there is no need to operate locally on the target master node.

Example: using a 3-master 3-slave cluster (source master 1 -> target master 1, source master 2 -> target master 2, source master 3 -> target master 3)

```
# Import shard 1: source master 1 (192.168.1.1:6359) -> AMDC V2.0.5 master 1 (10.0.1.10 (example IP; replace with the actual IP):6359)
```

```
amdc-cli --cluster import 10.0.1.10 (example IP; replace with the
actual IP):6359 \

--cluster-from 192.168.1.1:6359 \

--cluster-copy \

--cluster-replace \

-a <cluster password>

# Import shard 2: source master 2 (192.168.1.2:6359) -> AMDC V2.0.5
master 2 (10.0.1.11 (example IP; replace with the actual IP):6359)

amdc-cli --cluster import 10.0.1.11 (example IP; replace with the
actual IP):6359 \

--cluster-from 192.168.1.2:6359 \

--cluster-copy \

--cluster-replace \

-a <cluster password>

# Import shard 3: source master 3 (192.168.1.3:6359) -> AMDC V2.0.5
master 3 (192.168.1.12:6359)

amdc-cli --cluster import 192.168.1.13:6359 \

--cluster-from 192.168.1.3:6359 \

--cluster-copy \

--cluster-replace \
```

```
-a <cluster password>
```

3.5.3.3 Cluster Status Repair and Data Verification

Because the data was imported manually, a cluster status repair must be performed to ensure that slots and data match, with no missing slots or data disorder:

```
# Verify slot coverage (must show that all 16384 slots are allocated)
amdc-cli -p 6359 -a <password> cluster info
cluster_slots_assigned:16384 in the cluster info output means all slots are allocated with no omissions.
```

3.5.4 Notes

- Cluster migration requires full downtime, and all nodes of the source cluster must be forbidden from writing, because the cluster uses sharded storage and a data write to a single node will cause shard data inconsistency;
- The cluster gossip communication port 16359 (6359+10000) must be allowed through the firewall; otherwise nodes cannot discover the cluster topology, causing the cluster to split;
- When executing the amdc-cli --cluster import command, --cluster-copy is a required parameter; omitting this parameter will directly delete the original data of the AMDC V2.0.2 or earlier source node, causing irreversible source data loss. It is recommended to add --cluster-replace during migration to ensure that same-named test keys in the AMDC V2.0.5 cluster are overwritten and the data is completely consistent with the source;
- If the source cluster has data skew (one shard has an excessively large data volume), evaluate the disk and memory resources of the corresponding AMDC V2.0.5 cluster nodes before import; after migration, promptly scale out the skewed shards to avoid cluster performance bottlenecks that affect business access speed;
- After migration is complete, it is recommended to observe the cluster running for 10-20 minutes, focusing on: no data loss in each shard, automatic triggering of master failover, normal execution of cross-shard operations, no persistent delay in master-slave synchronization, and business access latency meeting expectations.

4 Migrating Redis 8.6 and Earlier Versions to AMDC V2.0.5

4.1 Preparation

4.1.1 Environment Check

- AMDC V2.0.5 is installed on the target server, and the installation directory permissions are normal (readable and executable);
- The Redis RDB snapshot file (dump.rdb) has been collected; if AOF persistence is enabled, obtain the AOF files first;
- The target server has sufficient disk space, at least 2 times the size of the source data files (reserving space for backup and operation).

4.1.2 Tool and File Preparation

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);

4.2 Standalone Mode Migration

Standalone mode migration applies to scenarios where only a single AMDC service is deployed. The core process is "file preparation -> data verification and repair -> deployment and startup -> verification". The specific steps are as follows:

4.2.1 Migration Prerequisites

- AMDC V2.0.5 is installed on the target server, and the installation directory permissions are normal (readable and executable);
- The Redis RDB snapshot file (dump.rdb) has been collected; if AOF persistence is enabled, obtain the AOF files first;
- The target server has sufficient disk space, at least 2 times the size of the source data files (reserving space for backup and operation).

4.2.2 Preparation

4.2.2.1 Data and Configuration Backup

- Check the permissions of the AMDC V2.0.5 installation directory and data storage directory (ensure the AMDC user can read and write);
- Copy the RDB file and the Redis configuration file to the designated backup directory of the target AMDC (e.g., /amdc with 644), ensuring AMDC can read them.

4.2.2.2 Back Up Data and Stop the Redis Standalone Node

- Stop business writes to Redis, run the bgsave command again to generate the latest RDB file; back up the source Redis configuration (redis.conf);
- The Redis standalone node must be stopped first, and any data writes are forbidden, to ensure the final consistency of the source data files.

```
# Execute on the node: stop the Redis process
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify the process has stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

4.2.3 Migration Steps

4.2.3.1 Cache Data Migration

1. Verify and repair the source RDB or AOF files (process AOF files first, as the data is more complete):

```
# If using AOF files
amdc-check-aof /path/to/appendonly.aof

# If using RDB files
amdc-check-rdb /path/to/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix /path/to/appendonly.aof

amdc-check-rdb --fix /path/to/dump.rdb
```

2. Copy the repaired files to the AMDC V2.0.5 installation directory:

```
# Check the AMDC V2.0.5 data directory (confirm via the dir configuration item in amdc.yaml; default path as follows), and copy the RDB file
cp /path/to/dump.rdb/ amdc/

# If using AOF files, first enable the AOF configuration (set appendonly yes in amdc.yaml), then copy
cp /path/to/appendonly.aof /amdc/
```

4.2.3.2 Modifying the redis.conf Configuration

1. Modify the Redis service configuration file (redis.conf) and append the following configuration item to the bottom of the configuration file:

```
worker-threads 1
```

4.2.3.3 Starting the AMDC Service

1. Start the AMDC V2.0.5 service, loading the Redis configuration file directly:

```
amdc-server /amdc/redis.conf
```

2. Verify the service startup status:

```
# View the service process
ps -ef | grep amdc-server

# View the log (default log path: /amdc/server.log)
tail -100f /amdc/server.log
```

4.2.3.4 Verifying the Startup

1. Connect to the AMDC V2.0.5 service and verify data integrity:

```
# Connect using amdc-cli (Redis default port 6379; add the -a option if a password is configured)
amdc-cli -p 6379 -a test123

# View keyspace statistics
info keyspace
```

2. Verify functional availability: perform write, modify, delete, and other operations, and confirm that the service responds normally with no configuration- or data-related errors:

4.2.4 Notes

- During migration, the Redis service must be stopped to avoid source file corruption caused by data writes;
- After migration, observe the service running for 10-20 minutes and confirm there are no log errors or other exceptions.

4.3 Master-Slave Mode Migration

Master-slave mode migration applies to scenarios where a one-master multi-slave AMDC service is deployed. The core process is "file preparation -> data verification and repair -> deployment and startup -> verification". The specific steps are as follows:

4.3.1 Migration Prerequisites

- AMDC V2.0.5 is installed on both the master and slave target servers, the installation directory permissions are normal (readable, writable, and executable), and the master and slave nodes can communicate over the network (no firewall blocking port 6359);
- The configuration files (redis.conf), AOF files (preferred, as the data is more complete), or RDB snapshot files of the Redis master and slave nodes have been collected, with a full backup made;
- Inform the business side of the downtime window in advance; it is recommended to operate during off-peak hours. The downtime is approximately the time for "data file copy + service startup" (adjust according to the data volume, generally 5-30 minutes);
- The target server has sufficient disk space, at least 2 times the size of the source data files (reserving space for backup and operation).

4.3.2 Preparation

4.3.2.1 Tool and File Preparation

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);

4.3.2.2 Back Up Data and Stop the Redis Master-Slave Cluster (Full Downtime)

- Stop business writes to Redis, run the bgsave command again to generate the latest RDB file; back up the source Redis configuration (redis.conf);
- All Redis master and slave nodes must be stopped first, and any data writes are forbidden, to ensure the final consistency of the source data files.

```
# Execute uniformly on the master node + all slave nodes: stop the Redis processes
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

4.3.3 Migration Steps (Execute in the Order of "Master Node -> All Slave Nodes")

4.3.3.1 Cache Data Migration

1. Verify and repair the source Redis master node data files (executed on the master node; skip repair if the files are not corrupted):

```
# If using AOF files (recommended)
amdc-check-aof /path/to/appendonly.aof

# If using RDB files
amdc-check-rdb /path/to/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix /path/to/appendonly.aof
```

```
amdc-check-rdb --fix /path/to/dump.rdb
```

2. Copy the repaired clean data files to the AMDC V2.0.5 master node data directory and modify permissions:

```
# Copy the AOF files (recommended; if using RDB, execute the next line instead)
cp /path/to/appendonly.aof /amdc/

# Copy the RDB file
cp /path/to/dump.rdb /amdc/

# Grant permissions (to prevent the service from being unable to read)
chmod 644 /amdc/{appendonly.aof, dump.rdb}
```

4.3.3.2 Configuration File Migration (redis.conf)

1. Copy the redis.conf of the Redis master/slave nodes to the root directory of the AMDC master/slave nodes.

2. Modify the Redis service configuration file (redis.conf) and append the following configuration item to the bottom of the configuration file:

```
worker-threads 1
```

4.3.3.3 Starting the AMDC Master-Slave Node Services

1. Start the AMDC master node first: switch to the master node's AMDC installation directory, execute the startup command, and confirm that the master node service started successfully;

```
# Start the node with the dedicated configuration file
amdc-server /amdc/redis.conf

# Verify the node startup status
```

```
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/server.log  # View the log
```

2. Then start all AMDC slave nodes: start the slave node AMDC services one by one, check the startup log of each slave node, and confirm there are no errors.

```
# Start the slave node with the dedicated configuration file
amdc-server /amdc/redis.conf

# Verify the slave node startup status
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/server.log  # View the log
```

4.3.3.4 Master-Slave Relationship Verification

- Execute the info replication command on the AMDC master node, view the slave node list, and confirm that all slave nodes have connected to the master node normally (master_link_status is up);
- Execute the info replication command on each AMDC slave node and confirm that synchronization with the master node is normal (offsets are consistent);
- Perform data read/write tests: set a key on the master node and get the key on a slave node to confirm that data synchronization is normal.

4.3.4 Notes

- Throughout the migration, all Redis master-slave nodes must be fully stopped first, and data writes are forbidden; otherwise the source data files will be inconsistent and data will be lost after import;
- After importing the master node data, data integrity must be verified before starting the slave nodes, to prevent slave nodes from synchronizing erroneous data;
- Slave nodes do not need any source data files copied; this is completed by AMDC V2.0.5 internal master-slave synchronization. Manual copying will cause synchronization conflicts;
- The core configurations of the master and slave nodes must be consistent (AOF switch, port, password, etc.), otherwise connection failures or synchronization exceptions will occur;
- After migration, observe the cluster running for 10-20 minutes and confirm that master-slave synchronization has no delay, there are no log errors, and business access is normal.

4.4 Sentinel Mode Migration

AMDC V2.0.5 supports loading the Redis configuration files redis.conf and sentinel.conf. The core process is full downtime -> migrate the master-slave nodes first -> then migrate the Sentinel nodes -> re-associate monitoring. Migration is completed through "master-slave cluster rebuild + Sentinel re-monitoring", and a short business downtime is required throughout. The specific steps are as follows:

4.4.1 Migration Prerequisites

- AMDC V2.0.5 is installed on all master-slave nodes and Sentinel nodes, and the network is interconnected (open service port 6379 and Sentinel port 26379);
- The Redis master-slave node configurations (redis.conf), data files, and Sentinel configuration files (sentinel.conf) have been collected, with a full backup made;
- Plan the downtime window in advance (off-peak hours). The downtime is the time for "Redis shutdown + AMDC V2.0.5 master-slave startup + Sentinel startup" (generally 10-40 minutes);
- All target nodes are time-synchronized and have sufficient disk space (reserve 2 times the source data space in the master node data directory).

4.4.2 Preparation

4.4.2.1 Tool and File Preparation

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);

4.4.2.2 Back Up Data and Fully Stop the Redis Sentinel + Master-Slave Cluster

- Stop business writes to Redis, run the bgsave command again to generate the latest RDB file; back up the source Redis configuration (redis.conf);
- All Redis Sentinel nodes and master-slave nodes must be stopped first, and data writes and Sentinel monitoring are forbidden, to ensure source data consistency.

```
# Execute uniformly on all nodes: stop the Redis master-slave +
Sentinel processes
ps -ef | grep -E 'redis-server | redis-sentinel' | grep -v grep |
```

```
awk '{print $2}' | xargs kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep -E 'redis-server | redis-sentinel' | grep -v grep
```

4.4.3 Migration Steps

4.4.3.1 Cache Data Migration

1. Verify and repair the source Redis master node data files (executed on the master node; skip repair if the files are not corrupted):

```
# If using AOF files (recommended)
amdc-check-aof /path/to/appendonly.aof

# If using RDB files
amdc-check-rdb /path/to/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix /path/to/appendonly.aof

amdc-check-rdb --fix /path/to/dump.rdb
```

2. Copy the repaired clean data files to the AMDC V2.0.5 master node data directory and modify permissions:

```
# Copy the AOF files (recommended; if using RDB, execute the next
line instead)
cp /path/to/appendonly.aof /amdc/

# Copy the RDB file
cp /path/to/dump.rdb /amdc/

# Grant permissions (to prevent the service from being unable to
read)
chmod 644 /amdc/{appendonly.aof, dump.rdb}
```

4.4.3.2 Configuration File Migration (redis.conf)

1. Copy the redis.conf of the Redis master/slave nodes to the root directory of the AMDC master/slave nodes.

2. Modify the Redis service configuration file (redis.conf) and append the following configuration item to the bottom of the configuration file:

```
worker-threads 1
```

3. The Sentinel configuration file (sentinel.conf) does not need to be modified; it can be copied to the root directory of the AMDC Sentinel nodes.

4.4.3.3 Starting the AMDC Master-Slave Node Services

1. Start the AMDC master node first: switch to the master node's AMDC installation directory, execute the startup command, and confirm that the master node service started successfully;

```
# Start the node with the dedicated configuration file
amdc-server /amdc/redis.conf

# Verify the node startup status
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/server.log  # View the log
```

2. Then start all AMDC slave nodes: start the slave node AMDC services one by one, check the startup log of each slave node, and confirm there are no errors.

```
# Start the slave node with the dedicated configuration file
amdc-server /amdc/redis.conf

# Verify the slave node startup status
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/server.log  # View the log
```

4.4.3.4 Starting the AMDC Sentinel Node Services

1. Then start all AMDC Sentinel nodes, check the Sentinel service processes and log status, and ensure

successful startup.

```
# Start the node with the dedicated configuration file
amdc-sentinel /amdc/sentinel.conf

# Verify the node startup status
ps -ef | grep amdc-sentinel   # View the process
tail -100f /amdc/sentinel.log  # View the log
```

4.4.3.5 Sentinel Functionality Verification

- Execute the info sentinel command on any AMDC Sentinel node, view the number and status of master-slave nodes monitored by the Sentinel, and confirm that the Sentinel can correctly identify the AMDC master-slave nodes;
- Test whether the business can automatically obtain the new master node address through the Sentinel and whether read/write operations are normal; after the test is complete, restore the original AMDC master node service.

4.4.4 Notes

- Before migration, all Redis Sentinel + master-slave nodes must be fully stopped, to prevent Sentinels from continuously monitoring stopped nodes, or data writes from causing source file inconsistency;
- After copying sentinel.conf, if the AMDC master node IP of the source differs from that of the target, the master node IP in the sentinel monitor parameter must be modified; otherwise the Sentinel cannot monitor the target master node;
- The master-slave nodes must be started before the Sentinel nodes, to ensure that the Sentinels can discover the master-slave nodes normally at startup and avoid monitoring failure.
- After migration, observe for 10-20 minutes, focusing on: no errors in the Sentinel log, no abnormalities in master-slave switching, synchronization delay within an acceptable range, and no interruption of business access.

4.5 Cluster Mode Migration

AMDC V2.0.5 supports loading the Redis configuration files redis.conf and node.conf. The core process is "file preparation -> data verification and repair -> deployment and startup -> verification". The specific steps are as follows:

4.5.1 Migration Prerequisites

- AMDC V2.0.5 is installed on all nodes of the target cluster (same number of nodes as the source cluster: M masters and N slaves), and the network is interconnected;
- The configuration files, AOF, and RDB data files of all Redis cluster nodes have been collected, and the source cluster topology information has been recorded (execute cluster nodes to obtain slot allocation and master-slave relationships);
- Plan a long downtime window in advance (adjust according to the data volume; off-peak hours recommended). The downtime is the time for "data export + cluster initialization + data import + verification" (approximately 30-60 minutes for data volumes within 10G);
- All target nodes have sufficient disk space (reserve 2 times the source node data space on each node).

4.5.2 Preparation

4.5.2.1 Tool and File Preparation

- amdc-check-rdb: used to verify and repair RDB cache files of AMDC V2.0.2 and earlier versions (deployed with the AMDC V2.0.5 installation package, located in the /amdc/ directory);
- amdc-check-aof: used to verify and repair AOF files of AMDC V2.0.2 and earlier versions (deployed in the same directory);

4.5.2.2 Back Up Data and Fully Stop the Redis Cluster (No Writes Allowed)

- Stop business writes to Redis, run the bgsave command again to generate the latest RDB file; back up the source Redis configuration (redis.conf);

```
# Execute uniformly on all Redis cluster nodes: stop the processes
ps -ef | grep amdc-server | grep -v grep | awk '{print $2}' | xargs
kill -9

# Verify the processes have stopped (no output means success)
ps -ef | grep amdc-server | grep -v grep
```

4.5.3 Migration Steps

4.5.3.1 Data File Preprocessing (Execute on All Source Nodes)

Verify and repair the AOF or RDB files of each Redis master node to ensure clean data files, avoiding data corruption after import into AMDC:

```
# Prefer AOF files (process each node separately; the cluster uses sharded storage and each node's data is independent)
amdc-check-aof /path/to/nodeX/appendonly.aof

# If there is no AOF, use the RDB file
amdc-check-rdb /path/to/nodeX/dump.rdb

# If a file is corrupted, perform repair
amdc-check-aof --fix /path/to/appendonly.aof

amdc-check-rdb --fix /path/to/nodeX/dump.rdb
```

4.5.3.2 Configuration File Migration (redis.conf, node.conf)

1. Copy the redis.conf of each Redis cluster node to the root directory of the corresponding AMDC cluster node.
2. Modify the Redis service configuration file (redis.conf) and append the following configuration item to the bottom of the configuration file:

```
worker-threads 1
```

3. Copy the node.conf of each Redis cluster node to the root directory of the corresponding AMDC cluster node.

4.5.3.3 Starting the AMDC Cluster Nodes

- Start all AMDC cluster nodes one by one, execute the startup command, check the startup log of each node, and confirm there are no errors (focus on cluster connection and slot loading).

```
# Start the node with the dedicated configuration file
amdc-server /amdc/redis.conf

# Verify the node startup status
ps -ef | grep amdc-server   # View the process
tail -100f /amdc/server.log  # View the log
```

4.5.3.4 Cluster Status Verification

- Execute the cluster info command on any AMDC node, confirm that the cluster status is ok, all slots are fully allocated, and there are no unallocated slots;
- Execute the cluster slots command, verify that the slot allocation is consistent with the source and that each node is responsible for the correct slots;
- Perform data read/write tests and cross-shard query tests to confirm that the data is complete and business is normal;

4.5.4 Notes

- Cluster migration requires full downtime, and all nodes of the source cluster must be forbidden from writing, because the cluster uses sharded storage and a data write to a single node will cause shard data inconsistency;
- If the source cluster has data skew (one shard has an excessively large data volume), evaluate the disk and memory resources of the corresponding AMDC V2.0.5 cluster nodes before import; after migration, promptly scale out the skewed shards to avoid cluster performance bottlenecks that affect business access speed;
- After migration is complete, it is recommended to observe the cluster running for 10-20 minutes, focusing on: no data loss in each shard, automatic triggering of master failover, normal execution of cross-shard operations, no persistent delay in master-slave synchronization, and business access latency meeting expectations.

5 Troubleshooting Common Issues

5.1 Tool Cannot Run

Issue 1: entering amdc-check-rdb in the terminal prompts "command not found" (Linux);

- Cause: the tool path has not been added to the system environment variables;
- Solution: run with the full path (e.g., /usr/local/amdc/amdc-check-rdb), or add the environment variable (echo "export PATH=/usr/local/amdc/:$PATH" >> /etc/profile, then source /etc/profile).

5.2 File Verification or Repair Fails

Issue 1: amdc-check-rdb or amdc-check-aof prompts "Permission denied";

- Cause: no file read or write permission;
- Solution: use chmod to modify the file permissions (chmod 644 dump.rdb), or run the tool with sudo (sudo amdc-check-rdb --fix dump.rdb).

Issue 2: AMDC still cannot start after repair;

- Cause: the file is too severely corrupted, or the tool version does not match the AMDC version;
- Solution: restore from a backup file, and upgrade or downgrade the tool version to match the AMDC service version.

5.3 amdc-cli Cannot Connect to the AMDC Service

Issue 1: connection prompts "Could not connect to AMDC at 127.0.0.1:6359: Connection refused";

- Cause: the AMDC service is not started;
- Solution: start the AMDC service (amdc-server /path/to/amdc.yaml, Linux).

Issue 2: remote connection prompts "Could not connect to AMDC at 10.0.1.100 (example IP; replace with the actual IP):6359: Connection timed out";

- Cause: port 6359 is not open in the firewall, or the AMDC configuration does not allow remote connections;

- Solution: open the firewall port (firewall-cmd --add-port=6359/tcp --permanent, Linux), modify the AMDC configuration (bind 0.0.0.0, protected-mode no), and restart the AMDC service.

Issue 3: executing a command after connecting prompts "NOAUTH Authentication required";

- Cause: the AMDC service has a password set, but it was not entered when connecting;
- Solution: enter the password with the -a option when connecting, or enter the AUTH password in interactive mode.

6 Summary

This document describes in detail the adaptation and migration processes from AMDC V2.0.2 and earlier versions and Redis 8.6 and earlier versions to AMDC V2.0.5. The core summary is as follows:

6.1 Migration Process from AMDC V2.0.2 and Earlier Versions to AMDC V2.0.5

- Before migration, complete the environment check and tool and file preparation, ensuring sufficient permissions and complete files;
- Configuration file migration is implemented with the amdc-conf-conv tool; you need to specify the configuration type and the source or target file path, and be sure to back up the target configuration file before conversion;
- Cache data migration is based on RDB files; first verify file integrity with amdc-check-rdb, repair corrupted files, and then deploy them to the AMDC V2.0.5 installation directory;
- After migration, verify the service startup status, data integrity, and functional availability to ensure a successful migration;
- If you encounter issues such as tool execution, file processing, or service startup during migration, refer to the Troubleshooting Common Issues chapter for quick resolution.

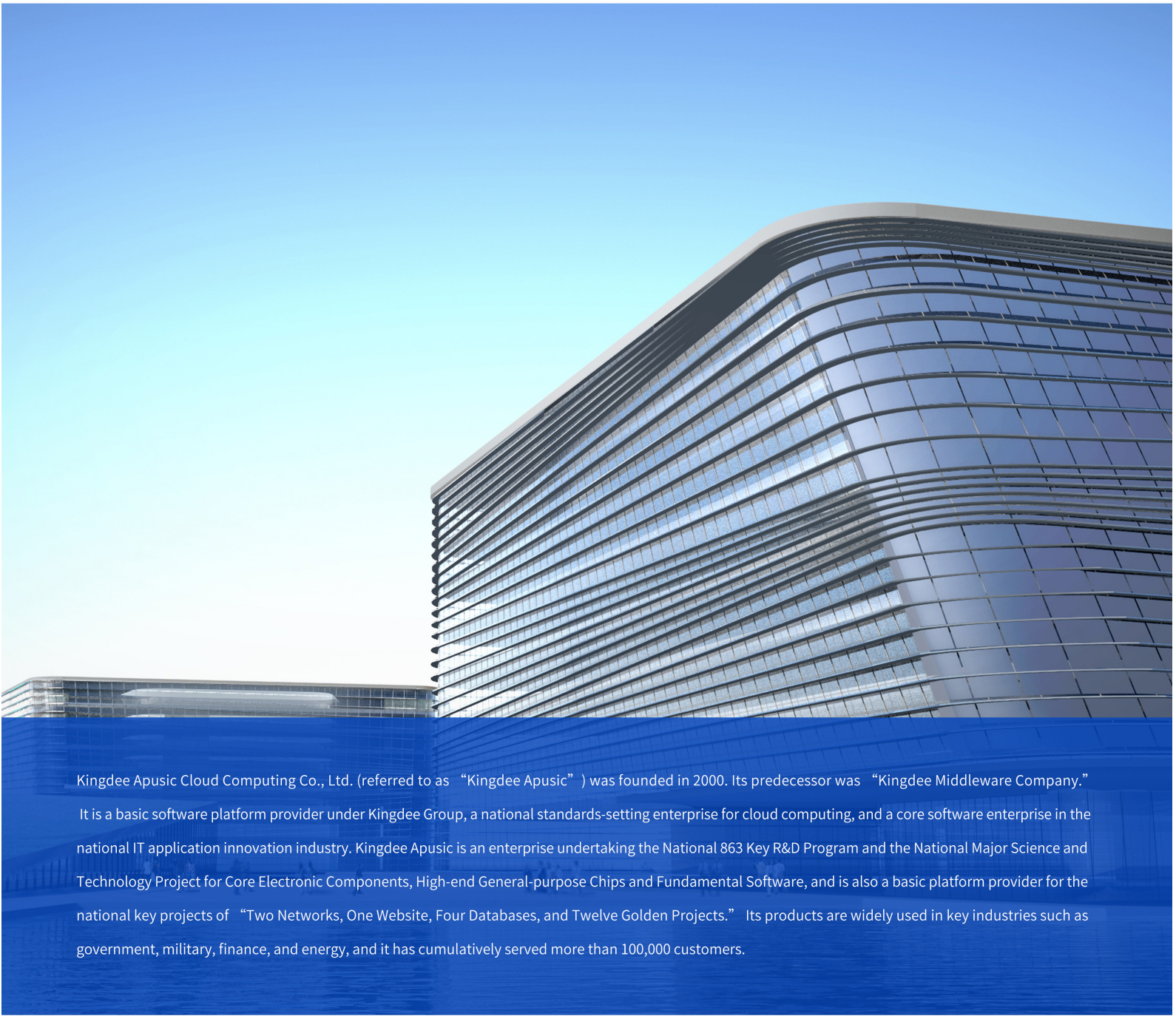
6.2 Migration Process from Redis 8.6 and Earlier Versions to AMDC V2.0.5

- Install AMDC: extract the installation package to the target directory and complete the basic installation;
- Stop the source Redis service, collect the core files of the corresponding mode (configuration files + data files), and back them up;
- Start the AMDC nodes using the Redis configuration files in the order of the corresponding mode, ensuring there are no errors;
- Verify that data and functionality are normal, switch business traffic, and keep the source for rollback.

By following the procedures in this document, you can achieve a smooth migration from AMDC V2.0.2 and earlier versions and Redis 8.6 and earlier versions to AMDC V2.0.5, ensuring the continuity of the cache service and data security.

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.

Apusic 金蝶天燕

National standard setter for cloud computing

Basic software enterprise under Kingdee Group

Core enterprise for IT application innovation

Website:www.apusic.com