ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

ApsaraDB for Redis

User Guide

Product Version: v3.16.2

Document Version: 20220913

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.What is KVStore for Redis?

KVStore for Redis is a database service that is compatible with open source Redis protocols. KVStore for Redis is based on a highly available hot standby architecture and can scale to meet the requirements of high-performance and low-latency read/write operations.

Features

- KVStore for Redis supports various data types, such as strings, lists, sets, sorted sets, hash tables, and streams. This service also supports advanced features, such as transactions, message subscription, and message publishing.
- KVStore for Redis Enhanced Edition (Tair), which is a key-value pair cloud caching service, is an advanced version of KVStore for Redis Community Edition.

Instance editions

Edition	Overview
Community Edition instances	KVStore for Redis Community Edition is compatible with the data cache service of open source Redis engines. It supports master-replica instances, cluster instances, and read/write splitting instances.
Performance-enhanced instances of KVStore for Redis Enhanced Edition	KVStore for Redis Enhanced Edition provides a multi-threading model and integrates some features of Alibaba Tair. KVStore for Redis Enhanced Edition (Tair) supports multiple data structures of Tair and is suitable for diverse scenarios.

2.Limits

Item	Description
LIST data type	The number of lists is unlimited. The size of each element in the list must be 512 MB or less. We recommend that you set the number of elements in a list to a value less than 8,192. The value length is 1 MB or less.
SET data type	The number of sets is unlimited. The size of each element is 512 MB or less. We recommend that you set the number of elements in a set to a value less than 8,192. The value length is 1 MB or less.
SORTED SET data type	The number of sorted sets is unlimited. The size of each element is 512 MB or less. We recommend that you set the number of elements in a sorted set to a value less than 8,192. The value length is 1 MB or less.
HASH data type	The number of fields is unlimited. The size of each element in a hash table is 512 MB or less. We recommend that you set the number of elements in a hash table to a value less than 8,192. The value length is 1 MB or less.
Number of databases (DBs)	A single instance supports a maximum of 256 databases.
Policy to delete expired data	• Two expiration policies are supported, which are active expiration and passive expiration. In active expiration, the system periodically detects and deletes expired keys in the background. This policy does not ensure timeliness. • In passive expiration, the system detects and deletes expired keys when you access these keys. **Note** In versions earlier than Redis 4.0, network jitter may occur due to high resource consumption caused by the deletion of large keys.
Mechanism to recycle idle connections	KVStore for Redis does not automatically recycle idle connections. You can manage the connections.
Policy for data persistence	KVStore for Redis sets appendfsync everysec. This configuration synchronizes append-only logs once every second.

3.Enhanced Edition and supported commands

3.1. Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair)

Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair) are suitable for scenarios that require high concurrency, high performance, and a large number of reads and writes on hot data. Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair) support multi-threading and integrate multiple Redis modules.

Benefits

Item	Description
Performance	• Provides read and write performance three times that of Redis-native databases or KVStore for Redis Community Edition with the same specifications. Performance-enhanced instances are suitable for scenarios that require high-frequency read and write requests for hot data. • Responds much faster when processing a large number of queries per second (QPS) compared with Redis-native databases. • Maintains stable performance in high-concurrency scenarios and eliminates connection issues that are caused by traffic spikes during peak hours. • Runs full and incremental synchronization tasks in input/output (I/O) threads to accelerate synchronization.
Enhanced module	Integrates multiple enhanced Redis modules that are developed by Alibaba Cloud. The modules are CAS and CAD commands, TairString commands, TairHash commands, TairGIS commands, TairBloom commands, and TairDoc commands. The enhanced modules provide various solutions, simplify business development in complex scenarios, and allow you to focus on your business development.
Compatibility	Compatible with open source Redis databases. You do not need to modify the code of your application when you use KVStore for Redis.
Scalability	Supports master-replica and cluster architectures. You can scale up or down the specifications of an instance, or upgrade an instance to a cluster instance as needed.

Scenarios

Suitable for scenarios such as live streaming, first-come, first-served events, and online education. Example:

Issue	Description
Community Edition is not suitable for scenarios that require high queries per second (QPS).	A business system can handle 200,000 QPS or higher for some cached hotkeys. The standard master-replica instances of KVStore for Redis Community Edition cannot maintain high performance during peak hours. Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair) that use the master-replica architecture can handle requests for popular commodities and provide an excellent user experience. This eliminates performance bottlenecks.
You want to use the current master-replica architecture and improve performance.	Cluster instances have specific limits. Therefore, the current master-replica architecture is retained. Performance-enhanced instances that use the master-replica architecture can improve performance and keep the current architecture unchanged. This eliminates the limits brought by cluster instances after you upgrade the instance to a cluster instance. Therefore, you do not need to adjust your business.
Self-managed Redis clusters contain a great number of shards, which increase the cost and degrade the performance.	Due to business growth, the number of shards increases. As a result, the management and maintenance costs increase and the performance decreases. Performance-enhanced cluster instances provide high performance and maintain only one third of the number of shards compared with self-managed Redis clusters. This reduces performance loss. KVStore for Redis provides various features to help you manage clusters.

Performance comparison

- Instances of KVStore for Redis Community Edition and open source Redis use a single-threading model. Each data node supports 80,000 to 100,000 QPS.
- KVStore for Redis performance-enhanced instances use a multi-threading model. In this model, I/O threads, worker threads, and auxiliary threads handle requests in parallel. The performance of a data shard of a performance-enhanced instance is three times the performance of a data shard of a Community Edition instance.

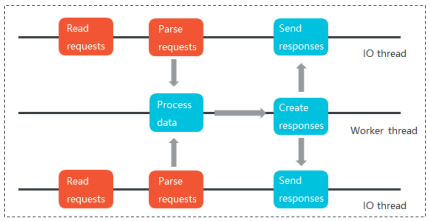
The following table describes different scenarios in which various types of instances and architectures are used.

Architecture	Instance type	Description
Standard master-replica instances	Community Edition instances	KVStore for Redis Community Edition instances cannot be used to process more than 100,000 QPS on a single data shard.
	Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair)	Performance-enhanced instances can be used to process more than 100,000 QPS on a single data shard.

Architecture	Instance type	Description
Cluster instances	Community Edition instances	A cluster instance of KVStore for Redis Community Edition contains multiple data shards. Each data shard provides performance similar to that of a master-replica instance. If one of the data shards stores hot data and receives a large number of concurrent requests for hot data, the read/write operations on the other data of this data shard may be delayed. As a result, performance bottlenecks may exist.
	Performance-enhanced instances of KVStore for Redis Enhanced Edition (Tair)	Performance-enhanced instances provide high performance in read/write operations on hot data and reduce maintenance costs.

Threading model comparison

Threading model	Description
Single-threading model 	During the process of request handling, native Redis databases and ApsaraDB for Redis Community Edition instances must undergo the following steps: read requests, parse requests, process data, and then send responses. In this case, network I/O operations and request parsing consume most of the resources that are available.

Threading model	Description
Multi-threading model  	To increase performance, each performance-enhanced instance of ApsaraDB for Redis runs multiple threads to process the tasks in these steps in parallel. • I/O threads are used to read requests, send responses, and parse commands. • Worker threads are used to process commands and timer events. • Auxiliary threads are used to monitor the statuses of nodes and heartbeats. Each performance-enhanced instance of ApsaraDB for Redis reads and parses requests in I/O threads, places the parsed requests as commands in a queue, and then sends the commands to the worker threads. Then, the worker threads run the commands to process the requests and send the responses to I/O threads by using a different queue. Each performance-enhanced instance of ApsaraDB for Redis supports a maximum of four parallel I/O threads. Unlocked queues and pipelines are used to transmit data between the I/O threads and the worker threads to improve multi-threading performance. **Note** • The running speeds of threads are accelerated for common data structures, such as string, list, set, hash, zset, hyperloglog, geo, and extension structures. • The replication of API operations such as pub, sub, and blocking is complete in the worker threads and can be accelerated to increase throughput. Performance can be increased by about 50%. • Transactions and Lua scripts require serial execution. No acceleration can be achieved.

> **Note** The multi-threading feature of native Redis 6.0 consumes a large number of CPU resources to deliver performance that is two times higher than the Real Multi-I/O feature of performance-enhanced instances of ApsaraDB for Redis. The Real Multi-I/O feature supports multiple connections, linearly increases in throughput, and provides fully accelerated I/O threads.

3.2. CAS and CAD commands

This topic describes the enhanced commands that you can run to process strings on performance-enhanced instances of KVStore for Redis Enhanced Edition. The commands include check-and-set (CAS) and compare-and-delete (CAD).

Prerequisites

The commands for TairHashes that are described in this topic can take effect only if the following conditions are met:

- Performance-enhanced instances of KVStore for Redis Enterprise Edition are used.
- The Redis strings to be managed are stored on the performance-enhanced instance.

> **Note** You can manage Redis strings and TairStrings on a performance-enhanced instance. However, CAS and CAD commands are applicable only to Redis strings.

Commands

Enhanced string commands

Statement	Syntax	Description
CAS	CAS <key> <oldvalue> <newvalue>	Changes the value of a specified key to newvalue if the current value of the key matches the oldvalue parameter. If the current value of the key does not match the oldvalue parameter, the value is not changed. **Note** The CAS command applies only to Redis strings. To change TairString values, run the EXCAS command.
CAD	CAD <key> <value>	Deletes a specified key if the current value of the key matches the oldvalue parameter. If the current value of the key does not match the oldvalue parameter, the key is not deleted. **Note** The CAD command applies only to Redis strings. To delete TairString keys, run the EXCAD command.

CAS

- Syntax

  CAS <key> <oldvalue> <newvalue>

- Time complexity

  O(1)

- Description

  This command can be used to change the value of a specified key to a new value if the current value of the key matches a specified value. If the current value of the key does not match the specified value, the value is not changed.

- Parameters and options

Parameter/option	Description
key	The key of the Redis string that you want to manage by using the command.
oldvalue	The value that you compare with the current value of the specified key.
newvalue	Changes the value of the specified key to the value of this parameter if the current value of the key matches the specified value.

- Returned values
  - If the operation is successful, a value of 1 is returned.
  - If the specified key does not exist, a value of -1 is returned.
  - If the operation fails, a value of 0 is returned.
  - Otherwise, an error message is returned.

- Example

```
127.0.0.1:6379> SET foo bar
OK
127.0.0.1:6379> CAS foo baa bzz
(integer) 0
127.0.0.1:6379> GET foo
"bar"
127.0.0.1:6379> CAS foo bar bzz
(integer) 1
127.0.0.1:6379> GET foo
"bzz"
```

CAD

- Syntax

  CAD <key> <value>

- Time complexity

  O(1)

- Description

This command can be used to delete a specified key if the current value of the key matches a specified value. If the current value of the key does not match the specified value, the key is not deleted.

- Parameters and options

Parameter/option	Description
key	The key of the Redis string that you want to manage by using the command.
value	The value that you compare with the current value of the specified key.

- Returned values
  - If the operation is successful, a value of 1 is returned.
  - If the specified key does not exist, a value of -1 is returned.
  - If the operation fails, a value of 0 is returned.
  - Otherwise, an error message is returned.
- Example

```
127.0.0.1:6379> SET foo bar
OK
127.0.0.1:6379> CAD foo bzz
(integer) 0
127.0.0.1:6379> CAD not-exists xxx
(integer) -1
127.0.0.1:6379> CAD foo bar
(integer) 1
127.0.0.1:6379> GET foo
(nil)
```

3.3. TairString commands

This topic describes the commands that are supported by TairStrings.

Overview

A TairString is a string that includes a version number. Redis-native strings use a key-value pair structure and contain only keys and values. TairStrings contain keys, values, and version numbers. TairStrings can be used in scenarios in which optimistic locking is applied. The **INCRBY** and **INCRBYFLOAT** commands are used to increase or decrease the values of Redis-native strings. You can use TairStrings to limit the range of the results that are returned by the commands. If a result is out of range, an error message is returned.

TairString has the following features:

- A TairString includes a version number.
- TairStrings can be used to limit the range of the results that are returned by the **INCRBY** and **INCRBYFLOAT** commands when you run these commands to increase the values of Redis-native string.

> **Warning** TairStrings are different from Redis-native strings. The commands that are supported by TairStrings and Redis-native strings are not interchangeable.

Prerequisites

The commands for TairHashes take effect only if the following conditions are met:

-
- The TairString to be managed is stored on a performance-enhanced instance.

> **Note** You can manage Redis-native strings and TairStrings on a performance-enhanced instance. However, Redis-native strings do not support the commands that are described in this topic.

Supported commands

TairString commands

Command	Syntax	Overview
EXSET	EXSET <key> <value> [EX time] [PX time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version]	Writes a value to a key.
EXGET	EXGET <key>	Retrieves the value and version number of a TairString.
EXSETVER	EXSETVER <key> <version>	Specifies the version number of a key.
EXINCRBY	EXINCRBY <key> <num> [EX time] [PX time] [EXAT time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version] [MIN minval] [MAX maxval]	Increases or decreases the value of a TairString. The value of the num parameter must be of the long type.
EXINCRBYFLOAT	EXINCRBYFLOAT <key> <num> [EX time] [PX time] [EXAT time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version] [MIN minval] [MAX maxval]	Increases or decreases the value of a TairString. The value of the num parameter must be of the double type.
EXCAS	EXCAS <key> <newvalue> <version>	Changes the value of a specified key when the current version number of the key matches the specified version number. If the update fails, the current value and version number of the key are returned.
EXCAD	EXCAD <key> <version>	Deletes a key when the current version number of the key matches the specified version number. If the operation fails, an error message is returned.

Command	Syntax	Overview
DEL	DEL <key> [key ...]	Deletes one or more TairStrings.

EXSET

- Syntax

  EXSET <key> <value> [EX time] [PX time] [EXAT time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version]

- Time complexity

  O(1)

- Description

  This command is used to write a value to a key.

- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
value	The value that you want to write to the specified key.
EX	The relative timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
EXAT	The absolute timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
PX	The relative timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
PXAT	The absolute timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
NX	Specifies that the value is written to the key only if the specified key does not exist.
XX	Specifies that the value is written to the key only if the specified key exists.

Parameter/option	Description
VER	The version number of the specified key. ○ If the specified key exists, the version number that is specified by this parameter is compared with the current version number. ■ If the version numbers match, the specified value is written to the key and the version number is increased by 1. ■ If this parameter does not match the current version number, an error message is returned. ○ If the specified key does not exist or the current version number of the key is 0, this parameter is ignored. The specified value is written to the key, and the version number is set to 1.
ABS	The absolute version number of the key. Writes the specified value to the key in disregard of the current version number of the key. Then, overwrites the version number with the ABS value.

- Returned values
  - If the operation is successful, OK is returned.
  - Otherwise, an error message is returned.
- Example

```
127.0.0.1:6379> EXSET foo bar XX
(nil)
127.0.0.1:6379> EXSET foo bar NX
OK
127.0.0.1:6379> EXSET foo bar NX
(nil)
127.0.0.1:6379> EXGET foo
1) "bar"
2) (integer) 1
127.0.0.1:6379> EXSET foo bar1 VER 10
(error) ERR update version is stale
127.0.0.1:6379> EXSET foo bar1 VER 1
OK
127.0.0.1:6379> EXGET foo
1) "bar1"
2) (integer) 2
127.0.0.1:6379> EXSET foo bar2 ABS 100
OK
127.0.0.1:6379> EXGET foo
1) "bar2"
2) (integer) 100
```

EXGET

- Syntax

EXGET <key>

- Time complexity

  O(1)

- Description

  This command is used to retrieve the value and version number of a TairString.

- Parameters and options

  key: the key of the TairString that you want to manage.

- Returned values
  - If the operation is successful, the value and version number of the TairString are returned.
  - Otherwise, an error message is returned.

- Example

```
127.0.0.1:6379> EXSET foo bar ABS 100
OK
127.0.0.1:6379> EXGET foo
1) "bar"
2) (integer) 100
127.0.0.1:6379> DEL foo
(integer) 1
127.0.0.1:6379> EXGET foo
(nil)
```

EXSETVER

- Syntax

  EXSETVER <key> <version>

- Time complexity

  O(1)

- Description

  This command is used to specify the version number of a key.

- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
version	The version number that you specify.

- Returned values
  - 1: the operation is successful.
  - 0: the specified key does not exist.
  - Otherwise, an error message is returned.

- Example

```
127.0.0.1:6379> EXSET foo bar
OK
127.0.0.1:6379> EXGET foo
1) "bar"
2) (integer) 1
127.0.0.1:6379> EXSETVER foo 2
(integer) 1
127.0.0.1:6379> EXGET foo
1) "bar"
2) (integer) 2
127.0.0.1:6379> EXSETVER not-exists 0
(integer) 0
```

EXINCRBY

- Syntax

  EXINCRBY |EXINCRBY <key> <num> [EX time] [PX time] [EXAT time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version] [MIN minval] [MAX maxval]

- Time complexity

  O(1)

- Description

  This command is used to increase or decrease the value of a TairString. The value of the num parameter must be of the long type.

- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
num	The value by which the specified TairString is increased. This value must be an integer.
EX	The relative timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
EXAT	The absolute timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
PX	The relative timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
PXAT	The absolute timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
NX	Specifies that the value is written to the key only if the specified key does not exist.
XX	Specifies that the value is written to the key only if the specified key exists.

Parameter/option	Description
VER	The version number of the specified key. ○ If the specified key exists, the version number that is specified by this parameter is compared with the current version number. ■ If the version numbers match, the value of the TairString is increased by num and the version number is increased by 1. ■ If this parameter does not match the current version number, an error message is returned. ○ If the specified key does not exist or the current version number of the key is 0, the specified version number does not take effect. In this case, the TairString value is increased by num and the version number is set to 1.
ABS	The absolute version number of the key. Increases the value of the TairString in disregard of the current version number of the key. Then, overwrites the version number with the ABS value.
MIN	The minimum value of the TairString.
MAX	The maximum value of the TairString.

- Returned values
  - If the operation is successful, the current value of the TairString is returned.
  - Otherwise, an error message is returned.
- Example

```
127.0.0.1:6379> EXINCRBY foo 100
(integer) 100
127.0.0.1:6379> EXINCRBY foo 100 MAX 150
(error) ERR increment or decrement would overflow
127.0.0.1:6379> FLUSHALL
OK
127.0.0.1:6379> EXINCRBY foo 100
(integer) 100
127.0.0.1:6379> EXINCRBY foo 100 MAX 150
(error) ERR increment or decrement would overflow
127.0.0.1:6379> EXINCRBY foo 100 MAX 300
(integer) 200
127.0.0.1:6379> EXINCRBY foo 100 MIN 500
(error) ERR increment or decrement would overflow
127.0.0.1:6379> EXINCRBY foo 100 MIN 500 MAX 100
(error) ERR min or max is specified, but not valid
127.0.0.1:6379> EXINCRBY foo 100 MIN 50
(integer) 300
```

EXINCRBYFLOAT

- Syntax

EXINCRBYFLOAT |EXINCRBYFLOAT <key> <num> [EX time] [PX time] [EXAT time] [EXAT time] [PXAT time] [NX | XX] [VER version | ABS version] [MIN minval] [MAX maxval]

- Time complexity

  O(1)

- Description

  This command is used to increase or decrease the value of a TairString. The value of the num parameter must be of the double type.

- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
num	The value by which the specified TairString is increased. The value must be a floating-point number.
EX	The relative timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
EXAT	The absolute timeout of the specified key. Unit: seconds. A value of 0 specifies that the key immediately expires.
PX	The relative timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
PXAT	The absolute timeout of the specified key. Unit: milliseconds. A value of 0 specifies that the key immediately expires.
NX	Specifies that the value is written to the key only if the specified key does not exist.
XX	Specifies that the value is written to the key only if the specified key exists.
VER	The version number of the specified key. ○ If the specified key exists, the version number that is specified by this parameter is compared with the current version number. ■ If the version numbers match, the value of the TairString is increased by num and the version number is increased by 1. ■ If this parameter does not match the current version number, an error message is returned. ○ If the specified key does not exist or the current version number of the key is 0, the specified version number does not take effect. In this case, the TairString value is increased by num and the version number is set to 1.
ABS	The absolute version number of the key. Increases the value of the TairString in disregard of the current version number of the key. Then, overwrites the version number with the ABS value.
MIN	The minimum value of the TairString.

Parameter/option	Description
MAX	The maximum value of the TairString.

- Returned values
  - If the operation is successful, the current value of the TairString is returned.
  - Otherwise, an error message is returned.
- Example

```
127.0.0.1:6379> EXSET foo 100
OK
127.0.0.1:6379> EXINCRBYFLOAT foo 10.123
"110.123"
127.0.0.1:6379> EXINCRBYFLOAT foo 20 MAX 100
(error) ERR increment or decrement would overflow
127.0.0.1:6379> EXINCRBYFLOAT foo 20 MIN 100
"130.123"
127.0.0.1:6379> EXGET foo
1) "130.123"
2) (integer) 3
```

EXCAS

- Syntax

  EXCAS <key> <newvalue> <version>
- Time complexity

  O(1)
- Description

  This command is used to change the version number of a specified key. The version number is changed only if the current version number of the key matches the specified version number.
- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
newvalue	When the current version number of the key matches the specified version number, the specified version number is overwritten by the value of the newvalue parameter.
version	The version number to be compared with the current version number of the specified key.

- Returned values
  - If the operation is successful, ["OK", "", version] is returned. The quotation marks ("") represent an empty string, and version represents the current version number of the key.

- If the operation fails, the following error message is returned: ["ERR update version is stale", value, version]. Value represents the current value of the key. Version represents the current version number of the key.
- Otherwise, an error message is returned.

- Example

```
127.0.0.1:6379> EXSET foo bar
OK
127.0.0.1:6379> EXCAS foo bzz 1
1) OK
2)
3) (integer) 2
127.0.0.1:6379> EXGET foo
1) "bzz"
2) (integer) 2
127.0.0.1:6379> EXCAS foo bee 1
1) ERR update version is stale
2) "bzz"
3) (integer) 2
```

EXCAD

- Syntax

  EXCAD <key> <version>

- Time complexity

  O(1)

- Description

  This command is used to delete a key when the current version number of the key matches the specified version number.

- Parameters and options

Parameter/option	Description
key	The key of the TairString that you want to manage by using the command.
newvalue	When the current version number of the key matches the specified version number, the specified version number is overwritten by the value of the newvalue parameter.
version	The version number to be compared with the current version number of the specified key.

- Returned values
  - 1: the operation is successful.
  - -1: the specified key does not exist.
  - 0: the operation fails.
  - Otherwise, an error message is returned.

- Example

```
127.0.0.1:6379> EXSET foo bar
OK
127.0.0.1:6379> EXGET foo
1) "bar"
2) (integer) 1
127.0.0.1:6379> EXCAD not-exists 1
(integer) -1
127.0.0.1:6379> EXCAD foo 0
(integer) 0
127.0.0.1:6379> EXCAD foo 1
(integer) 1
127.0.0.1:6379> EXGET foo
(nil)
```

3.4. TairHash commands

This topic describes the commands supported by a TairHash.

Overview

A TairHash is a hash that allows you to specify the expiration time and version number of a field. TairHashes and Redis-native hashes support multiple commands and provide high performance in data processing. However, Redis-native hashes allow you to specify the expiration time of only keys. TairHashes allow you to specify the expiration time of keys and fields. You can also use TairHashes to specify versions of fields. The improved features of TairHashes allow you to simplify the business development in most scenarios. TairHashes use the active expire algorithm to check the expiration time of fields and delete expired fields. This process does not increase the database response time.

TairHashes have the following features:

- The expiration time and version number for each field can be specified.
- Fields support the active expiration and passive expiration algorithms.
- TairHashes and Redis-native hashes use similar syntax.
- TairHashes support efficient active expiration policies. However, this can increase memory consumption to some extent.

> **Warning** TairHashes are different from Redis-native hashes. The commands that are supported by TairHashes and Redis-native hashes are not interchangeable.

Prerequisites

The following conditions must be met for TairHash commands to take effect:

- Performance-enhanced instances of KVStore for Redis Enterprise Edition are used.
- The TairHash to be managed is stored on the performance-enhanced instance.

> **Note** TairHashes and Redis-native hashes are managed on a performance-enhanced instance. The TairHash commands that are described in this topic cannot be applied to Redis-native hashes.

Commands

TairHash commands

Command	Syntax	Description
EXHSET	EXHSET <key> <field> <value> [EX time] [EXAT time] [PX time] [PXAT time] [NX/XX] [VER/ABS version] [NOACTIVE]	Adds a field to a specified TairHash. If the key does not exist, a key for the TairHash is created. If the field has an existing a value, this command overwrites the value of the field. When you run this command, the system uses the passive expiration algorithm to delete expired fields.
EXHMSET	EXHMSET <key> <field> <value> [field value...]	Sets specified fields to values in a TairHash that matches a specified key. If the key does not exist, a key for the TairHash is created. If the field has an existing value, this command overwrites the value of the field. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHPEXPIREAT	EXHPEXPIREAT <key> <field> <milliseconds-timestamp> [VER/ABS version] [NOACTIVE]	Specifies the absolute expiration time of a field in a specified TairHash. Unit: milliseconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHPEXPIRE	EXHPEXPIRE <key> <field> <milliseconds> [NOACTIVE]	Specifies the relative expiration time of a field in a TairHash that matches a specified key. Unit: milliseconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHEXPIREAT	EXHEXPIREAT <key> <field> <timestamp> [NOACTIVE]	Specifies the absolute expiration time of a field in a specified TairHash. Unit: seconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHEXPIRE	EXHEXPIRE <key> <field> <seconds> [NOACTIVE]	Specifies the relative expiration time of a field in a TairHash that matches a specified key. Unit: seconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHPTTL	EXHPTTL <key> <field>	Retrieves the remaining expiration time of a field in a specified TairHash. Unit: milliseconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.

Command	Syntax	Description
EXHTTL	EXHTTL <key> <field>	Retrieves the remaining expiration time of a field in a specified TairHash. Unit: seconds. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHVER	EXHVER <key> <field>	Retrieves the current version number of a field in a specified TairHash. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHSETVER	EXHSETVER <key> <field> <version>	Sets the version number of a field in a specified TairHash. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHINCRBY	EXHINCRBY <key> <field> <num> [EX time] [EXAT time] [PX time] [PXAT time] [VER/ABS version] [MIN minval] [MAX maxval]	Increases the field value in a specified TairHash by an integer. If the specified key does not exist, a TairHash is created. If the specified field does not exist, this command adds the field and sets the value of the field to 0 before a TairHash is created. You can also run the EX, EXAT, PX, or PXAT command to specify the expiration time for the field. When you run this command, fields are expired and deleted by using the passive expiration mechanism. **Note** To add a field that does not expire, you can run this command without the need to specify an expiration time.
EXHINCRBYFLOAT	EXHINCRBYFLOAT <key> <field> <value> [EX time] [EXAT time] [PX time] [PXAT time] [VER/ABS version] [MIN minval] [MAX maxval]	Increases the value of a field in a specified TairHash by a floating-point number. If the specified key does not exist, a TairHash is created. If the specified field does not exist, this command adds the field and sets the value of the field to 0 before a TairHash is created. You can also run the EX, EXAT, PX, or PXAT command to specify the expiration time for the field. When you run this command, fields are expired and deleted by using the passive expiration mechanism. **Note** To add a field that does not expire, you can run this command without the need to specify an expiration time.

Command	Syntax	Description
EXHGET	EXHGET <key> <field>	Retrieves the value of a specified field in a specified TairHash. If the specified key or field does not exist, a value of nil is returned. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHGETWITHVER	EXHGETWITHVER <key> <field>	Retrieves the value and version number of a field in a specified TairHash. If the specified key or field does not exist, a value of nil is returned. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHMGET	EXHMGET <key> <field> [field ...]	Retrieves multiple field values in a specified TairHash in each query if the key of a specified TairHash matches the specified key. If the specified key or field does not exist, a value of nil is returned. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHMGETWITHVER	EXHMGETWITHVER <key> <field> [field ...]	Retrieves the values and version numbers of multiple fields in a specified TairHash. If the specified key or field does not exist, a value of nil is returned. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHDEL	EXHDEL <key> <field> <field> <field> ...	Deletes a field from a specified TairHash. If the specified key or field does not exist, a value of 0 is returned. If the field is deleted, a value of 1 is returned. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHLEN	EXHLEN <key> [noexp]	Retrieves the number of fields in a specified TairHash. The returned value may include the number of expired fields that are not deleted. If you want to query only the number of fields that are not expired, you can set the *noexp* parameter.
EXHEXISTS	EXHEXISTS <key> <field>	Checks whether a field exists in a TairHash that matches a specified key. When you run this command, fields are expired and deleted by using the passive expiration mechanism.

Command	Syntax	Description
EXHSTRLEN	EXHSTRLEN <key> <field>	Retrieves the length of a field value in a specified TairHash. When you run this command, fields are expired and deleted by using the passive expiration mechanism.
EXHKEYS	EXHKEYS <key>	Retrieves all fields in a specified TairHash. Expired fields are filtered out in the returned results. To reduce response time, the system does not delete the expired fields while the system runs the command.
EXHVALS	EXHVALS <key>	Retrieves all field values in a specified TairHash. Expired fields are filtered out in the returned results. To reduce response time, the system does not delete the expired fields while the system runs the command.
EXHGETALL	EXHGETALL <key>	Retrieves all fields and associated values in a specified TairHash. Expired fields are filtered out in the returned results. To reduce response time, the system does not delete the expired fields while the system runs the command.
EXHSCAN	EXHSCAN <key> <op> <subkey> [MATCH pattern] [COUNT count]	Scans TairHashes that match a specified key. You can set the op parameter to values such as >, >=, <, <=, ==, ^, and $. This op parameter specifies a scan method. You can also set the MATCH parameter to specify a regular expression and filter out subkeys. The COUNT parameter limits the number of returned values. If you do not set the COUNT parameter, the default value is set to 10. Expired fields are filtered out in the returned results. To reduce response time, the system does not delete the expired fields while the system runs the command.
DEL	DEL <key> [key ...]	Deletes one or more TairHashes.

EXHSET

- Syntax

  EXHSET <key> <field> <value> [EX time] [EXAT time] [PX time] [PXAT time] [NX | XX] [VER/ABS version] [NOACTIVE]

- Time complexity

  O(1)

- Description

This command is used to add a field to the TairHash that matches a specified key. If the key does not exist, a key for the TairHash is created. If the field has an existing value, this command overwrites the value of the field.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
value	The value of the specified field. A field can have only one value.
EX	The relative expiration time of the specified field. Unit: seconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
EXAT	The absolute expiration time of the specified field. Unit: seconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PX	The relative expiration time of the specified field. Unit: milliseconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PXAT	The absolute expiration time of the specified field. Unit: milliseconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
NX	Specifies that the value is written only if the field does not exist.
XX	Specifies that the value is written only if the field exists.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the system continues running the command and increases the version number by 1. ■ If the version numbers do not match, an error message is returned. ○ If the specified field does not exist or the current version number of the field is 0, this parameter is ignored. The specified value is written to the field, and then the version number is set to 1.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly writes the specified value to the field regardless of whether the field has a value. Then, the version number is set to the specified ABS value when a field is added.
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

- Return values
  - 1: a new field is created and a value is set.
  - 0: the field has a value and the specified value overwrites the current value.
  - -1: the XX parameter is set but the specified field does not exist.
  - -1: the NX parameter is set and the specified field exists.
  - An error message that contains "ERR update version is stale" is returned. The message indicates that the value of the VER parameter does not match the current version number.
  - Otherwise, an exception is returned.

EXHGET

- Syntax

  EXHGET <key> <field>

- Time complexity

  O(1)

- Description

  This command is used to retrieve a value associated with the specified field in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - If the field exists and the operation is successful, the value of the field is returned.
  - nil: the key or field does not exist.
  - Otherwise, an exception is returned.

EXHMSET

- Syntax

  EXHMSET <key> <field> <value> [field value...]

- Time complexity

  O(1)

- Description

  This command is used to set specified fields to values in a TairHash that matches a specified key. If the key does not exist, a key for the TairHash is created. If the field has an existing value, this command overwrites the value of the field.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
value	The value of the specified field. A field can have only one value.

- Return values
  - If the operation is successful, OK is returned.
  - Otherwise, an exception is returned.

EXHPEXPIREAT

- Syntax

  EXHPEXPIREAT <key> <field> <milliseconds-timestamp> [VER/ABS version] [NOACTIVE]

- Time complexity

  O(1)

- Description

  This command is used to specify the absolute expiration time of a field in a TairHash that matches a specified key. Unit: milliseconds.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
milliseconds-timestamp	The UNIX timestamp. Unit: milliseconds.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the system continues running the command and increases the version number by 1. ■ If the version numbers do not match, an error message is returned. ○ If the specified field does not exist or the current version number of the field is 0, this parameter is ignored. The specified value is written to the field, and then the version number is set to 1.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly writes the specified value to the field regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value.

Parameter/option	Description
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

- Return values
  - 1: the field exists and a value is set.
  - 0: the field does not exist.
  - Otherwise, an exception is returned.

EXHPEXPIRE

- Syntax

  EXHPEXPIRE <key> <field> <milliseconds> [VER/ABS version] [NOACTIVE]

- Time complexity

  O(1)

- Description

  This command is used to specify the relative expiration time of a field in a TairHash that matches a specified key. Unit: milliseconds.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
milliseconds	The relative expiration time of the specified field. Unit: milliseconds.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the system continues running the command and increases the version number by 1. ■ If the version numbers do not match, an error message is returned. ○ If the specified field does not exist or the current version number of the field is 0, this parameter is ignored. The specified value is written to the field, and then the version number is set to 1.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly writes the specified value to the field regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value.

Parameter/option	Description
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

- Return values
  - 1: the field exists and a value is set.
  - 0: the field does not exist.
  - Otherwise, an exception is returned.

EXHEXPIREAT

- Syntax

  EXHEXPIREAT <key> <field> <timestamp> [VER/ABS version] [NOACTIVE]

- Time complexity

  O(1)

- Description

  This command is used to specify the absolute expiration time of a field in a TairHash that matches a specified key. Unit: seconds.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
timestamp	The UNIX timestamp. Unit: seconds.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ▪ If the version numbers match, the system continues running the command and increases the version number by 1. ▪ If the version numbers do not match, an error message is returned. ○ If the specified field does not exist or the current version number of the field is 0, this parameter is ignored. The specified value is written to the field, and then the version number is set to 1.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly writes the specified value to the field regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value.

Parameter/option	Description
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

- Return values
  - 1: the field exists and a value is set.
  - 0: the field does not exist.
  - Otherwise, an exception is returned.

EXHEXPIRE

- Syntax

  EXHEXPIRE <key> <field> <seconds>

- Time complexity

  O(1)

- Description

  This command is used to specify the relative expiration time of a field in a TairHash that matches a specified key. Unit: seconds.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
seconds	The relative expiration time of the specified field. Unit: seconds.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the system continues running the command and increases the version number by 1. ■ If the version numbers do not match, an error message is returned. ○ If the specified field does not exist or the current version number of the field is 0, this parameter is ignored. The specified value is written to the field, and then the version number is set to 1.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly writes the specified value to the field regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value.

Parameter/option	Description
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

- Return values
  - 1: the field exists and a value is set.
  - 0: the field does not exist.
  - Otherwise, an exception is returned.

EXHPTTL

- Syntax

  EXHPTTL <key> <field>
- Time complexity

  O(1)
- Description

  This command is used to retrieve the remaining expiration time of a field in a TairHash that matches a specified key. Unit: milliseconds.
- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - -2: the specified key or field does not exist.
  - -1: the specified field exists but the TTL value is not specified.
  - The expiration time of the field is returned if the field exists and the expiration time of the field is specified. Unit: milliseconds.
  - Otherwise, an exception is returned.

EXHTTL

- Syntax

  EXHTTL <key> <field>
- Time complexity

  O(1)
- Description

This command is used to retrieve the remaining expiration time of a field in a TairHash that matches a specified key. Unit: seconds.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - -2: the specified key or field does not exist.
  - -1: the specified field exists but the TTL value is not specified.
  - The expiration time of the field is returned if the field exists and the expiration time of the field is specified. Unit: seconds.
  - Otherwise, an exception is returned.

EXHVER

- Syntax

  EXHVER <key> <field>

- Time complexity

  O(1)

- Description

  This command is used to retrieve the current version number of a field in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - -1: the specified key does not exist.
  - -2: the specified field does not exist.
  - The version number of the specified field is returned if the operation is successful.
  - Otherwise, an exception is returned.

EXHSETVER

- Syntax

  EXHSETVER <key> <field> <version>

- Time complexity

O(1)

- Description

  This command is used to set the current version number of a field in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - 0: the specified TairHash or field does not exist.
  - 1: the version number is specified.
  - Otherwise, an exception is returned.

EXHINCRBY

- Syntax

  EXHINCRBY <key> <field> <num> [EX time] [EXAT time] [PX time] [PXAT time] [VER/ABS version] [MIN minval] [MAX maxval]

- Time complexity

  O(1)

- Description

  This command is used to increase the value of a field by num in a TairHash that matches a specified key. The value of the num parameter must be an integer. If the specified TairHash does not exist, a TairHash is created. If the specified field does not exist, this command adds the field and sets the value of the field to 0 before creating a TairHash. When you run this command, the system uses the passive expiration algorithm to delete expired fields.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
num	The integer by which you want to increase a specified field value.
EX	The relative expiration time of the specified field. Unit: seconds.A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.

Parameter/option	Description
EXAT	The absolute expiration time of the specified field. Unit: seconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PX	The relative expiration time of the specified field. Unit: milliseconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PXAT	The absolute expiration time of the specified field. Unit: milliseconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the TairHash is increased by num and the version number is increased by 1. ■ If the version numbers do not match, an error message is returned. ○ If the value of the *VER* parameter is 0, you do not need to check the version number.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly increases the TairHash by num regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value. The value of this parameter must not be 0.
MIN	The minimum value of the field. If the specified value is smaller than this lower limit, an exception is returned.
MAX	The maximum value of the field. If the specified value is larger than this upper limit, an exception is returned.
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

Note To add a field that does not expire, you can run this command without the need to specify an expiration time.

- Return values
  - The value increased by num is returned if the operation is successful.
  - Otherwise, an exception is returned.

EXHINCRBYFLOAT

- Syntax

  EXHINCRBYFLOAT <key> <field> <num> [EX time] [EXAT time] [PX time] [PXAT time] [VER/ABS version] [MIN minval] [MAX maxval]

- Time complexity

  O(1)

- Description

  This command is used to increase a specified field value by num in a TairHash that matches a specified key. The value of the num parameter must be a floating-point number. If the specified TairHash does not exist, a TairHash is created. If the specified field does not exist, this command adds the field and sets the value of the field to 0 before creating a TairHash. When you run this command, the system uses the passive expiration algorithm to delete expired fields.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.
num	The increment (a floating-point number) to be added to the specified field value.
EX	The relative expiration time of the specified field. Unit: seconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
EXAT	The absolute expiration time of the specified field. Unit: seconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PX	The relative expiration time of the specified field. Unit: milliseconds. A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
PXAT	The absolute expiration time of the specified field. Unit: milliseconds.A value of 0 specifies that the field immediately expires. If this option is not specified, the field does not expire.
VER	The version number of the specified field. ○ If the specified field exists, the version number specified by this parameter is matched with the current version number: ■ If the version numbers match, the TairHash is increased by num and the version number is increased by 1. ■ If the version numbers do not match, an error message is returned. ○ If the value of the *VER* parameter is 0, you do not need to check the version number.
ABS	The absolute version number of the field. If you set this parameter, the system forcibly increases the TairHash by num regardless of whether the field has a value. Then, the version number is overwritten with the specified ABS value. The value of this parameter must not be 0.
MIN	The minimum value of the field. If the specified value is smaller than this lower limit, an exception is returned.

Parameter/option	Description
MAX	The maximum value of the field. If the specified value is larger than this upper limit, an exception is returned.
NOACTIVE	When you set the *EX, EXAT, PX, or PXAT* parameter, you can set the *NOACTIVE* parameter to disable the active expiration policy for the field. This allows you to reduce the memory consumption.

> **Note** To add a field that does not expire, you can run this command without the need to specify an expiration time.

- Return values
  - The value increased by num is returned if the operation is successful.
  - Otherwise, an exception is returned.

EXHGETWITHVER

- Syntax

  EXHGETWITHVER <key> <field>

- Time complexity

  O(1)

- Description

  This command is used to retrieve the value and version number of a field in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - The value and version number of the field are returned if the field exists and the operation is successful.
  - nil: the key or field does not exist.
  - Otherwise, an exception is returned.

EXHMGET

- Syntax

  EXHMGET <key> <field> [field ...]

- Time complexity

O(1)

- Description

  This command is used to retrieve multiple field values in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - nil: the key does not exist.
  - An array is returned if the specified key and fields exist. Each element in the array corresponds to a field value.
  - An array is returned if the specified key exists but some fields do not exist. Each element in the array corresponds to a field value. The elements of the non-existing fields are displayed as nil.
  - Otherwise, an exception is returned.

EXHMGETWITHVER

- Syntax

  EXHMGETWITHVER <key> <field> [field ...]

- Time complexity

  O(1)

- Description

  This command is used to retrieve the values and version numbers of multiple fields in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - nil: the key does not exist.
  - An array is returned if the specified key and fields exist. Each element in the array corresponds to a field value and a version number.
  - An array is returned if the specified key exists but some fields do not exist. Each element in the array corresponds to a field value and a version number. The elements of the fields that do not exist are displayed as nil.
  - Otherwise, an exception is returned.

EXHDEL

- Syntax

  EXHDEL <key> <field> <field> <field> ...

- Time complexity

  O(1)

- Description

  This command is used to delete a field from a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - 0: the specified key or field does not exist.
  - 1: the operation is successful.
  - Otherwise, an exception is returned.

EXHLEN

- Syntax

  EXHLEN <key> [noexp]

- Time complexity

  O(1)

- Description

  This command is used to retrieve the number of fields in a TairHash that matches a specified key. The returned value may include the number of expired fields that are not deleted.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
noexp	By default, the **EXHLEN** command does not delete or filter out expired fields. Therefore, the results may include the number of expired fields that are not deleted. If you want to query only the number of fields that are not expired, you can set the *noexp* parameter. When you set the *noexp* parameter, ○ the response time of the **EXHLEN** command is based on the size of the Tairhash, because the system scans all TairHashes. ○ The result of the **EXHLEN** command does not include the number of expired fields that are not deleted.

- Return values
  - 0: the specified key or field does not exist.
  - The number of fields in the TairHash is returned if the operation is successful.
  - Otherwise, an exception is returned.

EXHEXISTS

- Syntax

  EXHEXISTS <key> <field>
- Time complexity

  O(1)
- Description

  This command is used to check whether a field exists in a TairHash that matches a specified key.
- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values
  - 0: the specified key or field does not exist.
  - 1: the specified field exists.
  - Otherwise, an exception is returned.

EXHSTRLEN

- Syntax

  EXHSTRLEN <key> <field>
- Time complexity

  O(1)
- Description

  This command is used to retrieve the length of a field value in a TairHash that matches a specified key.
- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.
field	An element of the TairHash. A TairHash key can be mapped to multiple fields.

- Return values

- 0: the specified key or field does not exist.
- The length of the specified field value is returned if the operation is successful.
- Otherwise, an exception is returned.

EXHKEYS

- Syntax

  EXHKEYS <key>

- Time complexity

  O(1)

- Description

  This command is used to retrieve all fields in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.

- Return values
  - If the specified key does not exist, an empty array is returned.
  - If the specified key exists, an array is returned. Each element in the array corresponds to a field.
  - Otherwise, an exception is returned.

EXHVALS

- Syntax

  EXHVALS <key>

- Time complexity

  O(1)

- Description

  This command is used to retrieve all field values in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.

- Return values
  - If the specified key does not exist, an empty array is returned.
  - If the specified key exists, an array is returned. Each element in the array corresponds to a field value.
  - Otherwise, an exception is returned.

EXHGETALL

- Syntax

  EXHGETALL <key>

- Time complexity

  O(1)

- Description

  This command is used to retrieve all fields and their values in a TairHash that matches a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.

- Return values
  - If the specified key does not exist, an empty array is returned.
  - If the specified key exists, an array is returned. Each element in the array corresponds to a field-value pair.
  - Otherwise, an exception is returned.

EXHSCAN

- Syntax

  EXHSCAN <key> <op> <subkey> [MATCH pattern] [COUNT count]

- Time complexity

  O(1) and O(N)

- Description

  This command is used to scan TairHashes that match a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairHash that you want to manage.

Parameter/option	Description
op	The position from which a scan starts. Valid values: ○ >: specifies that the scan starts from the first field with the value of the key greater than the subkey. ○ >=: specifies that the scan starts from the first field with the value of the key greater than or equal to the subkey. ○ <: specifies that the scan starts from the first field with the value of the key smaller than the subkey. ○ <=: specifies that the scan starts from the first field with the value of the key smaller than or equal to the subkey. ○ ==: specifies that the scan starts from the first field with the value of the key equal to the subkey. ○ ^: specifies that the scan starts from the first field. ○ $: specifies that the scan starts from the last field.
subkey	Specifies the position from which a scan starts. This parameter is set together with the op parameter. If op is set to ^ or $, this parameter does not take effect.
MATCH	The criteria used to filter the scanning result.

- Return values
  - If the specified key does not exist, an empty array is returned.
  - If the specified key exists, an array is returned. Each element in the array corresponds to a field-value pair.
  - Otherwise, an exception is returned.

3.5. TairGIS commands

TairGIS is a data structure that uses R-tree indexes and supports the API operations for Geographic Information System (GIS). Compared with Redis GEO commands, which allow you to use GeoHash and Redis Sorted Set to query points, TairGIS allows you to query points, lines, and planes, and provides more features.

Prerequisites

-
- The KVStore for Redis instance is upgraded to the latest minor version. For more information about how to upgrade the minor version of a KVStore for Redis instance, see *Upgrade the minor version* in *User Guide*.

Features

- Supports R-tree indexing.
- Allows you to query points, lines, and planes. This includes queries for the intersection of sets.
- Compatible with Redis-native GEO commands.

Commands

TairGIS commands

Statement	Syntax	Description
GIS.ADD	`GIS.ADD <area> <polygonName> <polygonWkt> [<polygonName> <polygonWkt> ...]`	Adds one or more specified polygons to a specified area. The polygons are described in well-known text (WKT). **Note** WKT is a text markup language that you can use to represent vector geometry objects on a map and the spatial reference systems of spatial objects. WKT also allows you to perform transformations between spatial reference systems.
GIS.GET	`GIS.GET <area> <polygonName>`	Retrieves the WKT information about a specified polygon in an area.
GIS.GETALL	`GIS.GETALL <area> [WITHOUTWKT]`	Queries all polygons in a specified area. The names and WKT information of the polygons are returned.
GIS.DEL	`GIS.DEL <area> <polygonName>`	Deletes a specified polygon in an area.
DEL	`DEL <key> [key ...]`	Deletes one or more TairGIS data structures. This is a Redis-native command.
GIS.CONTAINS	`GIS.CONTAINS <area> <polygonWkt> [WITHOUTWKT]`	Checks whether a polygon in a specified area consists of a specified point, line, or plane.
GIS.INTERSECTS	`GIS.INTERSECTS <area> <polygonWkt>`	Queries the intersection relationship between a polygon in a specified area and a specified point, line, or plane.
GIS.SEARCH	`GIS.SEARCH [RADIUS longitude latitude distance m|km|ft|mi] [MEMBER field distance m|km|ft|mi] [GEOM geom] [COUNT count] [ASC|DESC] [WITHDIST] [WITHOUTWKT]`	Retrieves points within a sphere whose radius, latitude, and longitude are specified.
GIS.WITHIN	`GIS.WITHIN <area> <polygonWkt> [WITHOUTWKT]`	Retrieves points, lines, or planes within a specified polygon in an area.

Parameters

Parameter	Description
area	The geometric area in which you want to manage the data.

Parameter	Description
PolygonName	The name of the polygon that you want to manage.
polygonWkt	The description of a polygon, which is described by using WKT. The following types are supported: • POINT: the WKT information that describes a point, such as `'POINT (30 11)'`. • LINESTRING: the WKT information that describes a line, such as `'LINESTRING (30 10, 40 40)'`. • POLYGON: the WKT information that describes a polygon, such as `'POLYGON ((31 20, 29 20, 29 21, 31 31))'`. **Note** MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRY, and COLLECTION are not supported.
WITHOUTWKT	Specifies whether to return the WKT information of polygons. If you run the GIS.GETALL, GIS.CONTAINS, GIS.SEARCH, or GIS.WITHIN command and specify the WITHOUTWKT parameter, the WKT information of the polygon is not returned.

GIS.ADD

- Syntax

  GIS.ADD <area> <polygonName> <polygonWkt> [<polygonName> <polygonWkt>...]

- Time complexity, which is used to indicate the trend of statement execution time as the data size increases.

  O(log n)

- Description

  This command is used to add one or more polygons to a specified area. The polygons are described in WKT.

- Returned values
  - If the operation is successful, the number of successful inserts and updates are returned.
  - Otherwise, an exception is returned.

- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
```

GIS.GET

- Syntax

```
GIS.GET <area> <polygonName>
```

- Time complexity

  O(1)

- Description

This command is used to retrieve the WKT information about a specified polygon in an area.

- Returned values
  - If the operation is successful, the WKT information about the polygon is returned.
  - If the specified area or polygon name does not exist, a value of nil is returned.
  - Otherwise, an exception is returned.
- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
127.0.0.1:6379> GIS.GET hangzhou campus
"POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.GET hangzhou not-exists
(nil)
127.0.0.1:6379> GIS.GET not-exists campus
(nil)
```

GIS.GETALL

- Syntax

```
GIS.GETALL <area> [WITHOUTWKT]
```

- Time complexity

  O(n)
- Description

  This command is used to query all polygons in a specified area. The names and WKT information of the polygons are returned. If you specify the *WITHOUTWKT* parameter, only the name of the polygon is returned.
- Returned values
  - If the operation is successful, the name and WKT information of the polygon are returned. If you specify the *WITHOUTWKT* parameter, only the name of the polygon is returned.
  - If no data is found, a value of nil is returned.
  - Otherwise, an exception is returned.
- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
127.0.0.1:6379> GIS.GETALL hangzhou
1) "campus"
2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.GETALL hangzhou WITHOUTWKT
1) "campus"
```

GIS.DEL

- Syntax

  GIS.DEL <area> <polygonName>

- Time complexity

  O(log n)

- Description

  This command is used to delete a specified polygon in an area.

- Returned values
  - If the operation is successful, OK is returned.
  - If the specified area or polygon name does not exist, a value of nil is returned.
  - Otherwise, an exception is returned.

- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
127.0.0.1:6379> GIS.GET hangzhou campus
"POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.DEL hangzhou not-exists
(nil)
127.0.0.1:6379> GIS.DEL not-exists campus
(nil)
127.0.0.1:6379> GIS.DEL hangzhou campus
OK
127.0.0.1:6379> GIS.GET hangzhou campus
(nil)
```

GIS.CONTAINS

- Syntax

  GIS.CONTAINS <area> <polygonWkt>

- Time complexity
  - Optimal time complexity: $O(\log_M n)$
  - Least desirable time complexity: log(n).

- Description

  This command is used to check whether a polygon in a specified area contains a specified point, line, or plane.

- Returned values
  - If the operation is successful, the name and WKT information of the specified polygon that contains the specified point, line, or plane are returned. If you specify the *WITHOUTWKT* parameter, only the name of the polygon is returned.
  - If no data is found, a value of nil is returned.
  - Otherwise, an exception is returned.

- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
127.0.0.1:6379> GIS.CONTAINS hangzhou 'POINT (30 11)'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.CONTAINS hangzhou 'LINESTRING (30 10, 40 40)'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.CONTAINS hangzhou 'POLYGON ((31 20, 29 20, 29 21, 31 31))'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
```

GIS.INTERSECTS

- Syntax

  GIS.INTERSECTS <area> <polygonWkt>

- Time complexity
  - Optimal time complexity: $O(\log_M n)$
  - Least desirable time complexity: log(n).
- Description

  This command is used to query the intersection relationship between a polygon in a specified area and a specified point, line, or plane.

- Returned values
  - If the operation is successful, the name and WKT information of the specified polygon that intersects with the specified point, line, or plane are returned.
  - If no data is found, a value of nil is returned.
  - Otherwise, an exception is returned.
- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
(integer) 1
127.0.0.1:6379> GIS.INTERSECTS hangzhou 'POINT (30 11)'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.INTERSECTS hangzhou 'LINESTRING (30 10, 40 40)'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379> GIS.INTERSECTS hangzhou 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
1) "0"
2) 1) "campus"
   2) "POLYGON((30 10,40 40,20 40,10 20,30 10))"
127.0.0.1:6379>
```

GIS.SEARCH

- Syntax

```
GIS.SEARCH [RADIUS longitude latitude distance m|km|ft|mi]
[MEMBER field distance m|km|ft|mi]
[GEOM geom]
[COUNT count]
[ASC|DESC]
[WITHDIST]
[WITHOUTWKT]
```

- Time complexity
  - Optimal time complexity: $O(\log_M n)$
  - Least desirable time complexity: log(n).
- Description

  This command is used to retrieve points within a sphere whose radius, latitude, and longitude are specified. The following parameters are supported:
  - *RADIUS*: searches points by the longitude, latitude, and radius. Specify the parameter values in the following order: longitude, latitude, radius, and radius unit, such as `RADIUS 15 37 200 km`.
  - *MEMBER*: searches points by the latitude and longitude from an existing polygon, and a specified radius. Specify the parameter values in the following order: polygon name, radius, and radius unit, such as, `MEMBER Agrigento 100 km`.
  - *GEOM*: specifies the search range in the WKT format for a random polygon, such as `GIS.SEARCH Sicily "POINT (13.361389 38.115556)"`.
  - *COUNT*: limits the number of returned entries, such as `COUNT 3`.
  - *ASC|DESC*: sorts returned entries by distance. For example, ASC indicates that the entries are sorted from nearest to farthest from the center.
  - *WITHDIST*: specifies whether to return the distance.
  - *WITHOUTWKT*: specifies whether to return the WKT information of polygons.
- Returned values
  - If the operation is successful, the name and WKT information of the specified polygon are returned.
  - If no data is found, a value of nil is returned.
  - Otherwise, an exception is returned.
- Example

```
127.0.0.1:6379> GIS.ADD Sicily "Palermo" "POINT (13.361389 38.115556)" "Catania" "POINT(1
5.087269 37.502669)"
(integer) 2
127.0.0.1:6379> GIS.SEARCH Sicily RADIUS 15 37 200 km WITHDIST
1) "2"
2) 1) "Palermo"
   2) "POINT(13.361389 38.115556)"
   3) "190.4424"
   4) "Catania"
   5) "POINT(15.087269 37.502669)"
   6) "56.4413"
127.0.0.1:6379> GIS.SEARCH Sicily RADIUS 15 37 200 km WITHDIST WITHOUTWKT
1) "2"
2) 1) "Palermo"
   2) "190.4424"
   3) "Catania"
   4) "56.4413"
127.0.0.1:6379> GIS.SEARCH Sicily RADIUS 15 37 200 km WITHDIST WITHOUTWKT ASC
1) "2"
2) 1) "Catania"
   2) "56.4413"
   3) "Palermo"
   4) "190.4424"
127.0.0.1:6379> GIS.SEARCH Sicily RADIUS 15 37 200 km WITHDIST WITHOUTWKT ASC COUNT 1
1) "2"
2) 1) "Catania"
   2) "56.4413"
127.0.0.1:6379> GIS.ADD Sicily "Agrigento" "POINT (13.583333 37.316667)"
(integer) 1
127.0.0.1:6379> GIS.SEARCH Sicily MEMBER Agrigento 100 km
1) "2"
2) 1) "Palermo"
   2) "POINT(13.361389 38.115556)"
   3) "Agrigento"
   4) "POINT(13.583333 37.316667)"
```

GIS.WITHIN

- Syntax

```
GIS.WITHIN <area> <polygonWkt> [WITHOUTWKT]
```

- Time complexity
  - Optimal time complexity: $O(\log_M n)$
  - Least desirable time complexity: log(n).
- Description

  This command is used to retrieve points, lines, or planes within a specified polygon in an area. If you specify the *WITHOUTWKT* parameter, only the name of the polygon is returned.
- Returned values
  - If the operation is successful, the name and WKT information of the polygon are returned.

- If no data is found, a value of nil is returned.
- Otherwise, an exception is returned.

- Example

```
127.0.0.1:6379> GIS.ADD hangzhou campus 'POINT (30 10)'
(integer) 1
127.0.0.1:6379> GIS.ADD hangzhou campus1 'LINESTRING (30 10, 40 40)'
(integer) 1
127.0.0.1:6379> GIS.WITHIN hangzhou 'POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))'
1) "2"
2) 1) "campus"
   2) "POINT(30 10)"
   3) "campus1"
   4) "LINESTRING(30 10,40 40)"
```

3.6. TairBloom commands

This topic describes the commands that are supported by a TairBloom.

Overview

TairBloom is a Bloom filter that supports dynamic scaling. TairBloom is a space-efficient probabilistic data structure that consumes minimal memory to check whether an element exists. TairBloom supports dynamic scaling and maintains a stable false positive rate during scaling.

You can use bitmaps on Redis data structures, such as hashes, sets, and strings, to implement similar features of TairBloom. However, these data structures may consume a large amount of memory or fail to maintain a stable false positive rate during dynamic scaling. You can use TairBloom to check whether large volumes of data exist. In this case, a specific false positive rate is allowed. You can use the built-in Bloom filter of TairBloom without further development or the need to create an extra Bloom filter.

Key features:

- Consumes minimal memory.
- Enables dynamic scaling.
- Maintains a stable custom false positive rate during scaling.

Prerequisites

The commands for TairBloom take effect only if the following conditions are met:

- Performance-enhanced instances of KVStore for Redis Enterprise Edition are used.
- The TairBloom that you want to manage is stored on a performance-enhanced instance.

Commands

TairBloom commands

Command	Syntax	Description
BF.RESERVE	BF.RESERVE <key> <error_rate> <capacity>	Creates an empty TairBloom filter with a specific capacity. The error_rate parameter specifies the false positive rate of the TairBloom filter.

Command	Syntax	Description
BF.ADD	BF.ADD <key> <item>	Adds an item to a specified TairBloom filter.
BF.MADD	BF.MADD <key> <item> [item...]	Adds multiple items to a specified TairBloom filter.
BF.EXISTS	BF.EXISTS <key> <item>	Checks whether an item exists in a specified TairBloom filter.
BF.MEXISTS	BF.MEXISTS <key> <item> [item...]	Checks whether multiple items exist in a specified TairBloom filter.
BF.INSERT	BF.INSERT <key> [CAPACITY cap] [ERROR error] [NOCREATE] ITEMS <item...>	Adds multiple items to a specified TairBloom filter. If the TairBloom filter does not exist, you can specify whether to create a TairBloom filter. You can also specify the capacity and false positive rate of the new TairBloom filter.
BF.DEBUG	BF.DEBUG <key>	Retrieves the information about a specified TairBloom filter. The information includes the number of layers, the number of items at each layer, and the false positive rate.
DEL	DEL <key> [key ...]	Deletes one or more TairBlooms. **Note** You cannot delete the items that are added to a TairBloom. You can run the DEL command to delete the TairBloom.

BF.RESERVE

- Syntax

  BF.RESERVE <key> <error_rate> <capacity>

- Time complexity

  O(1)

- Description

  This command is used to create an empty TairBloom filter with a specific capacity. The error_rate parameter specifies the false positive rate of the TairBloom filter.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
error_rate	The expected false positive rate. The value of this parameter must be between 0 and 1. A lower value indicates higher memory usage and CPU utilization of the TairBloom filter.

Parameter/option	Description
capacity	The initial capacity of the TairBloom filter. This parameter specifies the maximum number of items that can be added to the TairBloom filter. If the number of items that are added to the TairBloom filter exceeds the specified capacity, TairBloom expands the capacity by increasing the layers of the Bloom filter. During the scaling process, the number of items in the Tairbloom filter exponentially increases and the performance linearly decreases. To query a specific item after a layer is added to the filter, TairBloom may iterate through multiple layers of the filter. The capacity of each new layer is twice that of the previous layer. If your workloads require high performance, we recommend that you add items to TairBloom based on your business requirements to avoid automatic scaling.

- Returned values
  - If the operation is successful, OK is returned.
  - If the operation is not successful, an error message is returned.

BF.ADD

- Syntax

  BF.ADD <key> <item>
- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.
- Description

  This command is used to add an item to a specified TairBloom filter.
- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
item	The item that you want to add to the TairBloom filter.

- Returned values
  - 1: The item does not exist in the filter.
  - 0: The item may exist in the filter.
  - If the operation is not successful, an error message is returned.

BF.MADD

- Syntax

  BF.MADD <key> <item> [item...]
- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.

- Description

  This command is used to add multiple items to a specified TairBloom filter.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
item	The items that you want to add to the TairBloom filter. You can specify multiple items.

- Returned values
  - If the operation is successful, an array is returned. The values in the returned array can be 1 or 0. If a specified item does not exist, the value is 1. If a specified item may exist, the value is 0.
  - If the operation is not successful, an error message is returned.

BF.EXISTS

- Syntax

  BF.EXISTS <key> <item>

- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.

- Description

  This command is used to check whether an item exists in a specified TairBloom filter.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
item	The item that you want to query in the TairBloom filter.

- Returned values
  - 0: The specified item does not exist in the filter.
  - 1: The specified item may exist in the filter.
  - If the operation is not successful, an error message is returned.

BF.MEXISTS

- Syntax

  BF.MEXISTS <key> <item> [item...]

- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.

- Description

  This command is used to check whether multiple items exist in a specified TairBloom filter.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
item	The items that you want to query in the TairBloom filter. You can specify multiple items.

- Returned values
  - If the operation is successful, an array is returned. The values in the returned array can be 1 or 0. If a specified item does not exist, the value is 0. If a specified item may exist, the value is 1.
  - If the operation is not successful, an error message is returned.

BF.INSERT

- Syntax

  BF.INSERT <key> [CAPACITY cap] [ERROR error] [NOCREATE] ITEMS <item...>

- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.

- Description

  This command is used to add multiple items to a specified TairBloom filter. If the TairBloom filter does not exist, you can specify whether to create a TairBloom filter. You can also specify the capacity and false positive rate of the new TairBloom filter.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.
CAPACITY	The initial capacity of the TairBloom filter. This parameter specifies the maximum number of items that can be added to the TairBloom filter. If the filter exists, you do not need to specify this parameter. If the number of items that are added to the TairBloom filter exceeds the specified capacity, TairBloom expands the capacity by increasing the layers of the Bloom filter. During the scaling process, the number of items in the Tairbloom filter exponentially increases and the performance linearly decreases. To query a specific item after a layer is added to the filter, TairBloom may iterate through multiple layers of the filter. The capacity of each new layer is twice that of the previous layer. If your workloads require high performance, we recommend that you add items to TairBloom based on your business requirements to avoid automatic scaling.
ERROR	The expected false positive rate. If the filter exists, you do not need to specify this parameter. The value of this parameter must be between 0 and 1. A lower value indicates higher memory usage and CPU utilization of the TairBloom filter.

Parameter/option	Description
NOCREATE	Specifies that the specified TairBloom filter is not automatically created if the filter does not exist. This parameter cannot be specified together with CAPACITY or ERROR.
ITEMS	All items that you want to add to the TairBloom filter.

- Returned values
  - If the operation is successful, an array is returned. The values in the returned array can be 1 or 0. If a specified item does not exist, the value is 1. If a specified item may exist, the value is 0.
  - If the operation is not successful, an error message is returned.

BF.DEBUG

- Syntax

  BF.DEBUG <key>

- Time complexity

  O(log N). N specifies the number of layers of the TairBloom filter.

- Description

  This command is used to retrieve the information about a specific TairBloom filter. The information includes the number of layers, the number of items at each layer, and the false positive rate.

- Parameters and options

Parameter/option	Description
key	The key of the TairBloom filter that you want to manage.

- Returned values
  - If the operation is successful, an array is returned. The values in the returned array can be 1 or 0. If a specified item does not exist, the value is 1. If a specified item may exist, the value is 0.
  - If the operation is not successful, an error message is returned.

Memory usage test result

Capacity (number of elements)	false positive:0.01	false positive:0.001	false positive:0.0001
100000	0.12 MB	0.25 MB	0.25 MB
1000000	2 MB	2 MB	4 MB
10000000	16 MB	32 MB	32 MB
100000000	128 MB	256 MB	256 MB
1000000000	2 GB	2 GB	4 GB

3.7. TairDoc commands

This topic describes the commands supported by TairDocs.

Overview

A TairDoc is a document data structure. You can use TairDocs to add, modify, query, or delete JavaScript Object Notation (JSON) data.

TairDoc has the following features:

- Supports JSON standards.
- Fully compatible with RedisJSON.
- Supports the syntax of JSONPath and JSON Pointer.
- Stores data in a binary tree and simplifies the retrieval of child elements.
- Supports conversion from the JSON format to the Extensible Markup Language (XML) or YAML Ain't Markup Language (YAML) format.

Prerequisites

The commands described in this topic take effect only if the following conditions are met:

- A **performance-enhanced** instance of ApsaraDB for Redis Enhanced Edition is used.
- The TairDoc to be managed is stored on the performance-enhanced instance.

Commands

TairDoc commands

Command	Syntax	Description
JSON.SET	JSON.SET <key> <path> <json> [NX or XX]	Writes a JSON value to the path of a specified key. If the specified key does not exist, the path must be the root directory. If the specified key and path exist, the specified JSON value overwrites the current JSON value in the path.
JSON.GET	JSON.GET <key> [PATH] [FORMAT <XML/YAML>] [ROOTNAME <root>] [ARRNAME <arr>]	Retrieves JSON data from a TairDoc path of a specified key.
JSON.DEL	JSON.DEL <key> [path]	Deletes JSON data from a TairDoc path of a specified key. If the path is not specified, the key is deleted. This command does not take effect if the key or path does not exist.
JSON.TYPE	JSON.TYPE <key> [path]	Retrieves the type of JSON data from a TairDoc path of a specified key.
JSON.NUMINCRBY	JSON.NUMINCRBY <key> [path] <value>	Increases JSON data in a TairDoc path by a specified value. The path must exist, and both the JSON data and increased value must be of the int or double type.

Command	Syntax	Description
JSON.STRAPPEND	JSON.STRAPPEND <key> [path] <json-string>	Appends a string specified in json-string to the end of the string in a TairDoc path. If you do not specify the path, the root directory is used.
JSON.STRLEN	JSON.STRLEN <key> [path]	Retrieves the JSON value length in a TairDoc path. If you do not specify the path, the root directory is used.
JSON.ARRAPPEND	JSON.ARRAPPEND <key> <path> <json> [<json> ...]	Appends one or more JSON values to the end of an array in a TairDoc path.
JSON.ARRPOP	JSON.ARRPOP <key> <path> [index]	Removes an element specified by the index parameter from an array in a specified TairDoc path and returns the removed element.
JSON.ARRINSERT	JSON.ARRINSERT <key> <path> <index> <json> [<json> ...]	Adds one or more JSON elements to an array in a TairDoc path. The index parameter specifies the position to which the JSON elements are added.
JSON.ARRLEN	JSON.ARRLEN <key> [path]	Retrieves the length of the array in a TairDoc path.
JSON.ARRTRIM	JSON.ARRTRIM <key> <path> <start> <stop>	Trims a JSON array in a specified TairDoc path. The start value and the stop value specify the range in which the JSON data is retained.
DEL	DEL <key> [key ...]	Deletes one or more TairDocs.

JSON.SET

- Syntax

  JSON.SET <key> <path> <json> [NX | XX]

- Time complexity

  O(N)

- Description

  This command is used to write a JSON value to the path of a specified key. If the specified key does not exist, the path must be the root directory. If the specified key and path exist, the specified JSON value overwrites the current JSON value in the path.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data. ○ If the specified key does not exist, the path must be the root directory. ○ If the specified key and path exist, the specified JSON value overwrites the current JSON value in the path.

Parameter/option	Description
json	If the specified key and path exist, the specified JSON value overwrites the current JSON value in the TairDoc.
NX	Specifies that a JSON value is written only if the required path does not exist.
XX	Specifies that a JSON value is written only if the required path exists.

- Returned values
  - OK: the operation is successful.
  - null: The operation fails. This occurs when you specify the NX or XX parameter.
  - Otherwise, an exception is returned.
- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.SET doc .foo '"flower"'
OK
127.0.0.1:6379> JSON.GET doc .foo
"flower"
127.0.0.1:6379> JSON.SET doc .not-exists 123 XX
127.0.0.1:6379> JSON.SET doc .not-exists 123 NX
OK
127.0.0.1:6379> JSON.GET doc .not-exists
123
```

JSON.GET

- Syntax

  JSON.GET <key> <path> [FORMAT <XML | YAML>] [ROOTNAME <root>] [ARRNAME <arr>]
- Time complexity

  O(N)
- Description

  This command is used to retrieve JSON data from a TairDoc path of a specified key.
- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
FORMAT	The format of the JSON data to be returned. Valid values: XML and YAML.
ROOTNAME	The tag that specifies a root element in an XML document.

Parameter/option	Description
ARRNAME	The tag that specifies an array element in an XML document.

- Returned values
  - The JSON data stored in the path is returned if the operation is successful.
  - Otherwise, an exception is returned.
- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.GET doc
{"foo":"bar","baz":42}
127.0.0.1:6379> JSON.GET doc .foo
"bar"
127.0.0.1:6379> JSON.GET doc .not-exists
ERR pointer illegal or array index error or object type is not array or map
127.0.0.1:6379> JSON.GET doc . format xml
<? xml version="1.0" encoding="UTF-8"? ><root><foo>bar</foo><baz>42</baz></root>
127.0.0.1:6379> JSON.GET doc . format xml rootname ROOT arrname ARRAY
<? xml version="1.0" encoding="UTF-8"? ><ROOT><foo>bar</foo><baz>42</baz></ROOT>
127.0.0.1:6379> JSON.GET doc . format yaml
foo: bar
baz: 42
```

JSON.DEL

- Syntax

  JSON.DEL <key> [path]
- Time complexity

  O(N)
- Description

  This command is used to delete JSON data from a TairDoc path of a specified key. If the path is not specified, the key is deleted. This command does not take effect if the key or path does not exist.
- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.

- Returned values
  - 1: the operation is successful.
  - 0: the operation fails.

- Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.DEL doc .foo
1
127.0.0.1:6379> JSON.DEL doc .not-exists
ERR old item is null for remove or replace
127.0.0.1:6379> JSON.DEL not-exists
0
127.0.0.1:6379> JSON.GET doc
{"baz":42}
127.0.0.1:6379> JSON.DEL doc
1
127.0.0.1:6379> JSON.GET doc
127.0.0.1:6379>
```

JSON.TYPE

- Syntax

  JSON.TYPE <key> [path]

- Time complexity

  O(N)

- Description

  This command is used to retrieve the type of JSON data from a TairDoc path of a specified key.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.

- Returned values
  - The type of JSON data is returned if the operation is successful. The type includes boolean, null, number, string, array, object, raw, reference, and const.
  - null: the specified key or path does not exist.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.TYPE doc
object
127.0.0.1:6379> JSON.TYPE doc .foo
string
127.0.0.1:6379> JSON.TYPE doc .baz
number
127.0.0.1:6379> JSON.TYPE doc .not-exists
127.0.0.1:6379>
```

JSON.NUMINCRBY

- Syntax

  JSON.NUMINCRBY <key> [path] <value>

- Time complexity

  O(N)

- Description

  This command is used to increase JSON data in a TairDoc path by a specified value. The path must exist, and the JSON data and increased value must be both of the type of int or double.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
value	The increment to be added to the JSON data in the specified path.

- Returned values
  - The increased value in the specified path is returned if the operation is successful.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.NUMINCRBY doc .baz 1
43
127.0.0.1:6379> JSON.NUMINCRBY doc .baz 1.5
44.5
127.0.0.1:6379> JSON.NUMINCRBY doc .foo 1
ERR node not exists or not number type
127.0.0.1:6379> JSON.NUMINCRBY doc .not-exists 1
ERR node not exists or not number type
127.0.0.1:6379>
```

JSON.STRAPPEND

- Syntax

  JSON.STRAPPEND <key> [path] <json-string>

- Time complexity

  O(N)

- Description

  This command is used to append a string specified in json-string to the end of the string in a TairDoc path. If you do not specify the path, the root directory is used.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
json-string	The string to be appended to the specified path.

- Returned values
  - The length of the increased value in the path is returned if the operation is successful.
  - -1: the specified key does not exist.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.STRAPPEND doc .foo rrrrr
8
127.0.0.1:6379> JSON.GET doc .foo
"barrrrrr"
127.0.0.1:6379> JSON.STRAPPEND doc .not-exists
ERR node not exists or not string type
127.0.0.1:6379> JSON.STRAPPEND not-exists abc
-1
```

JSON.STRLEN

- Syntax

  JSON.STRLEN <key> [path]

- Time complexity

  O(N)

- Description

  This command is used to retrieve the JSON value length in a TairDoc path. If you do not specify the path, the root directory is used.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.

- Returned values
  - The length of the value in the path is returned if the operation is successful.
  - -1: the specified key does not exist.
  - Otherwise, an exception is returned.
- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : 42}'
OK
127.0.0.1:6379> JSON.STRLEN doc .foo
3
127.0.0.1:6379> JSON.STRLEN doc .baz
ERR node not exists or not string type
127.0.0.1:6379> JSON.STRLEN not-exists
-1
```

JSON.ARRAPPEND

- Syntax

  JSON.ARRAPPEND <key> <path> <json> [<json> ...]
- Time complexity

  O(M×N). M specifies the number of JSON elements to be appended and N specifies the number of elements in the array.
- Description

  This command is used to append one or more JSON values to the end of an array in a TairDoc path.
- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
json	The JSON value to be inserted to a specified array.

- Returned values
  - The number of elements in the array is returned if the operation is successful. The added elements are included.
  - -1: the specified key does not exist.

- Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"id": [1,2,3]}'
OK
127.0.0.1:6379> JSON.GET doc .id
[1,2,3]
127.0.0.1:6379> JSON.ARRAPPEND doc .id null false true
6
127.0.0.1:6379> JSON.GET doc .id
[1,2,3,null,false,true]
127.0.0.1:6379> JSON.GET doc .id.2
3
127.0.0.1:6379> JSON.ARRAPPEND not-exists .a 1
-1
```

JSON.ARRPOP

- Syntax

  JSON.ARRPOP <key> <path> [index]

- Time complexity

  O(M×N). M specifies the child elements that the specified key contains and N specifies the number of elements in the array.

- Description

  This command is used to remove an element specified by an index from an array in a specified TairDoc path and return the removed element.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
index	The index of the array, which specifies the value to be removed. If you do not specify this parameter, the last value in the array is removed. A negative value specifies reverse numbering from the end of the array.

- Returned values
  - The removed element is returned if the operation is successful.
  - An error message is returned if the array is empty: 'ERR array index outflow'.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"id": [1,2,3]}'
OK
127.0.0.1:6379> JSON.ARRPOP doc .id 1
2
127.0.0.1:6379> JSON.GET doc .id
[1,3]
127.0.0.1:6379> JSON.ARRPOP doc .id -1
3
127.0.0.1:6379> JSON.GET doc .id
[1]
127.0.0.1:6379> JSON.ARRPOP doc .id 10
ERR array index outflow
127.0.0.1:6379> JSON.ARRPOP doc .id
1
127.0.0.1:6379> JSON.ARRPOP doc .id
ERR array index outflow
127.0.0.1:6379>
```

JSON.ARRINSERT

- Syntax

  JSON.ARRINSERT <key> <path> <index> <json> [<json> ...]

- Time complexity

  O(M×N). M specifies the number of JSON elements to be appended and N specifies the number of elements in the array.

- Description

  This command is used to add one or more JSON elements to an array in a TairDoc path. The index parameter specifies the position to which the JSON elements are added.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
index	The index of the array, which specifies the value to be removed. If you do not specify this parameter, the last value in the array is removed. A negative value specifies reverse numbering from the end of the array.
json	The JSON value to be inserted to a specified array.

- Returned values
  - The number of elements in the array is returned if the operation is successful. The added elements are included.
  - An error message is returned if the array is empty: 'ERR array index outflow'.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"id": [2,3,5]}'
OK
127.0.0.1:6379> JSON.ARRINSERT doc .id 0 0 1
5
127.0.0.1:6379> JSON.GET doc .id
[0,1,2,3,5]
127.0.0.1:6379> JSON.ARRINSERT doc .id 4 4
6
127.0.0.1:6379> JSON.GET doc .id
[0,1,2,3,4,5]
127.0.0.1:6379>
```

JSON.ARRLEN

- Syntax

  JSON.ARRLEN <key> [path]

- Time complexity

  O(N)

- Description

  This command is used to retrieve the length of the array in a TairDoc path.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.

- Returned values
  - The length of the queried array is returned if the operation is successful.
  - -1: the specified key does not exist.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"id": [2,3,5]}'
OK
127.0.0.1:6379> JSON.ARRLEN doc .id
3
127.0.0.1:6379> JSON.ARRLEN not-exists
-1
```

JSON.ARRTRIM

- Syntax

  JSON.ARRTRIM <key> <path> <start> <stop>

- Time complexity

  O(N)

- Description

  This command is used to trim a JSON array in a TairDoc path. The start value and the stop value specify the range in which the JSON data is retained.

- Parameters and options

Parameter/option	Description
key	The key of the TairDoc that you want to manage.
path	The TairDoc path where you want to manage JSON data.
start	The start of the range in which elements are retained after a trim. The value is an index that starts from 0. The element at the start position is retained.
stop	The end of the range in which elements are retained after a trim. The value is an index that starts from 0. The element at the end position is retained.

- Returned values
  - The length of the trimmed array is returned if the operation is successful.
  - -1: the specified key does not exist.
  - Otherwise, an exception is returned.

- Examples

```
127.0.0.1:6379> JSON.SET doc . '{"id": [1,2,3,4,5,6]}'
OK
127.0.0.1:6379> JSON.ARRTRIM doc .id 3 4
2
127.0.0.1:6379> JSON.GET doc .id
[4,5]
127.0.0.1:6379> JSON.ARRTRIM doc .id 3 4
ERR array index outflow
127.0.0.1:6379> JSON.ARRTRIM doc .id -2 -5
ERR array index outflow
127.0.0.1:6379>
```

JSON Pointer and JSONPath

TairDoc supports the JSONPointer syntax and also supports some of the JSONPath syntax. The following table shows the syntax examples.

JSONPointer	JSONPath

`127.0.0.1:6379> JSON.SET doc . '{"foo": "bar", "baz" : [1,2,3]}'`
`OK`
`127.0.0.1:6379> JSON.GET doc .foo`
`"bar"`
`127.0.0.1:6379> JSON.GET doc .baz[0]`
`1`
	`127.0.0.1:6379> JSON.SET doc "" '{"foo": "bar", "baz" : [1,2,3]}'`
	`OK`
	`127.0.0.1:6379> JSON.GET doc /foo`
	`"bar"`
	`127.0.0.1:6379> JSON.GET doc /baz/0`
	`1`

The following table shows how TairDoc supports JSONPath and JSON Pointer.

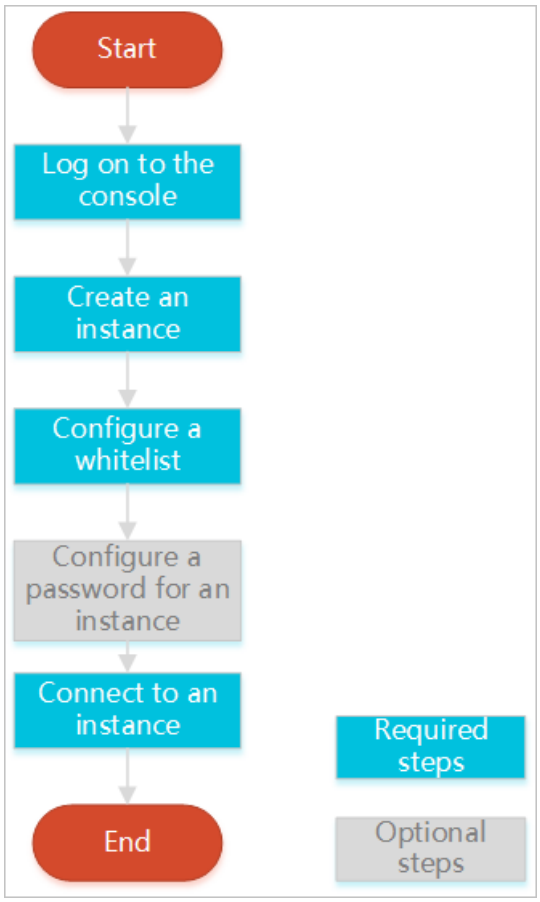
Item	JSONPath	JSONPointer
Root element	.	“”
An individual element in a path	.a.b.c	/a/b/c
Array	.a[2]	/a/2
Multiple elements in a path	.a["b.c"]	/a/b.c
Multiple elements in a path	.a['b.c']	/a/b.c

4.Quick Start

4.1. Get started with KVStore for Redis

This topic describes all operations that you can perform on an instance from instance creation to database logon. This topic helps you understand how to create and manage an instance.

The following figure shows how to manage a KVStore for Redis instance.



- Log on to the KVStore for Redis console

  This topic describes how to log on to the KVStore for Redis console.

- Create an instance

  KVStore for Redis supports classic networks and virtual private clouds (VPCs). You can create KVStore for Redis instances in these networks.

- Configure a whitelist

  Before you use a KVStore for Redis instance, add IP addresses or CIDR blocks that are used to access the database to the whitelist of the instance to improve the security and stability of the database.

- If you do not specify a password when you create the instance, specify a password on the **Instance Information** page.

- Connect to a KVStore for Redis instance

To connect to the KVStore for Redis instance, you can use a client that supports Redis protocols or use the Redis command-line interface (redis-cli) tool.

4.2. Log on to the Apsara Uni-manager Management Console

This topic describes how to log on to the Apsara Uni-manager Management Console.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

   > **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, which include ! @ # $ %

3. Click **Log On**.
4. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - It is the first time that you log on to the console after MFA is forcibly enabled by the administrator.
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the account and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device.

     Enter a six-digit MFA authentication code and click **Authenticate**.

     > **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Operations Console User Guide*.

4.3. Create an instance

This topic describes how to create a KVStore for Redis instance in the KVStore for Redis console.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. Click **Create Instance** in the upper-right corner of the page.
3. Configure the parameters that are described in the following table.

Configure the following parameters

Section	Parameter	Description
Basic Settings	**Organization**	The organization in which you want to create the instance.
	Resource Set	The resource set in which you want to create the instance. **Notice** After you select a resource set, the instance is accessible only to the members in the resource set.
Region	**Region**	The region in which you want to create the instance.
	Zone	The zone in which you want to create the instance. If you specify multiple zones, zone-disaster recovery can be implemented. **Note** Multiple zones can be specified only for **standard** and **cluster** instances.
	Edition	○ **community**: This edition is compatible with the open source Redis protocol and provides high performance. ○ **enterprise**: This edition is developed based on the community edition. The read /write performance of this edition can be three times that of the community edition. It is integrated with multiple self-developed Redis modules to improve the applicability. For more information, see *Performance-enhanced instances of the KVStore for Redis Enhanced Edition (Tair)* in *Product Introduction*.
	Chip Architecture	The chip architecture of the machine to which the instance belongs. **Note** If you do not have permissions to select an option, contact the operations administrator to grant such permissions to your account.

Section	Parameter	Description
Specifications	**Engine Version**	The engine version of the instance. Redis 4.0 and 5.0 are supported. You can select Redis 4.0 only if you want to create a Community Edition instance.
	Architecture Type	○ **Standard**: runs in a master-replica architecture, provides high-performance caching services, and ensures high data availability. ○ **Cluster**: eliminates the performance bottleneck that is caused by the single-threading model. You can use the high-performance cluster instance to process large-capacity workloads. ○ **read-write split**: ensures high availability and high performance, and supports multiple specifications. The read/write splitting architecture allows a large number of concurrent requests to read hot data from read replicas. This reduces the loads on the master node and minimizes O&M costs. **Note** The **read/write splitting** architecture is supported only by Community Edition.
	Node Type	**Master-replica** is automatically selected and cannot be changed. The node type has one master node and one replica node to ensure availability.
	Instance Type	The instance type of the instance. The maximum number of connections and maximum internal bandwidth vary based on the instance type. For more information, see *Instance types* in *Product Introduction*.
Network	**Network Type**	○ Classic network: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked by using security groups or whitelists. ○ VPC: A virtual private cloud (VPC) helps you build an isolated network environment on Apsara Stack. You can specify the route table, CIDR block, and gateway of a VPC. You can also migrate applications to the cloud without service interruption. You can use an Express Connect circuit or a VPN to connect self-managed data centers to cloud resources in a VPC. **Note** Before you can select VPC, you must create a VPC. For more information, see *Create a VPC* and *Create a vSwitch* in *VPC User Guide*.
	Instance Name	The name of the instance, which is used to identify and manage the instance. ○ The name must be 2 to 128 characters in length. ○ The name can contain letters, digits, underscores (_), and hyphens (-), and must start with a letter.

Section	Parameter	Description
Password	**Password Setting**	You can select **Set Now** or **Set after Purchase**.
	Logon Password	The password used to access the instance. The password must meet the following requirements: ○ The password must be 8 to 30 characters in length. ○ The password must contain uppercase letters, lowercase letters, and digits. The password cannot contain special characters.
	Confirm Password	Enter the specified password again.

4. Click **Submit**.

4.4. Configure a whitelist

Before you use a KVStore for Redis instance, add IP addresses or CIDR blocks that are used to access the database to the whitelist of the instance to improve the security and stability of the database.

Context

> **Note** A properly configured whitelist can ensure a higher level of security protection for your KVStore for Redis instance. We recommend that you maintain the whitelist on a regular basis.

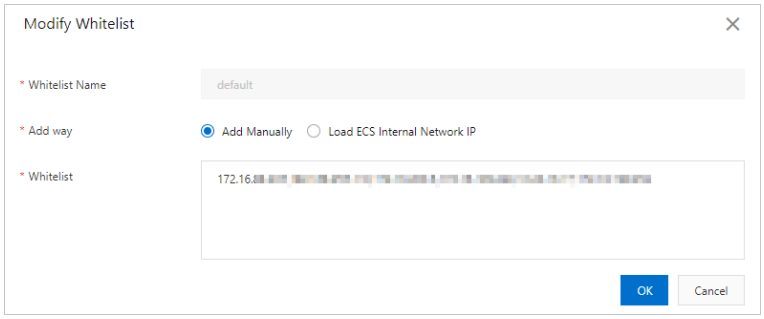
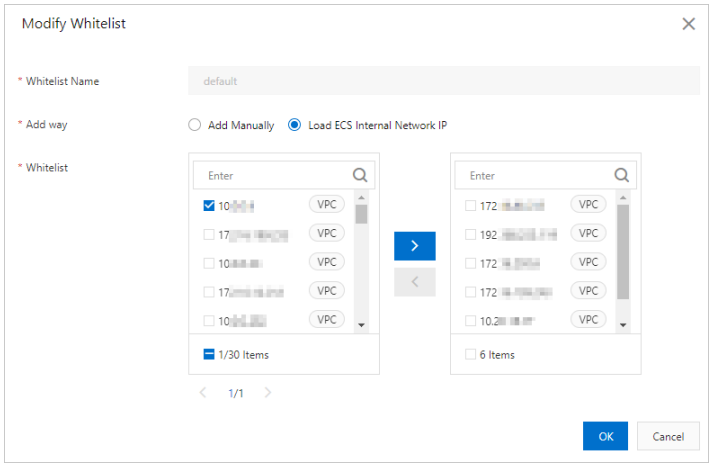
Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Whitelist Settings**.
4. Find the IP address whitelist that you want to manage and click **Modify**.

   > **Note** You can also click **Add Whitelist** to create an IP address whitelist. The name of the IP address whitelist must be 2 to 32 characters in length and can contain lowercase letters, digits, and underscores (_). The name of the whitelist must start with a lowercase letter and end with a lowercase letter or digit.

5. In the dialog box that appears, perform one of the following operations:

Action	Procedure

Action	Procedure
Manually modify the IP address whitelist	Enter IP addresses or CIDR blocks. Manually modify the IP address whitelist  **Note** ○ Separate multiple IP addresses with commas (,). You can add up to 1,000 unique IP addresses. Supported formats are specific IP addresses such as 10.23.12.24 and CIDR blocks such as 10.23.12.24/24. /24 indicates the length of the IP address prefix. An IP address prefix can be 1 to 32 bits in length. ○ If you set the prefix length to 0, for example, 0.0.0.0/0 or 127.0.0.1/0, all IP addresses are allowed to access the instance. In this case, the security risk of your instance is high. Proceed with caution.
Add private IP addresses of ECS instances to an IP address whitelist	i. Click **Load ECS Internal Network IP**. ii. Select IP addresses based on your business requirements. Add private IP addresses of ECS instances  **Note** To find the ECS instance to which a specific IP address is assigned, you can move the pointer over the IP address. Then, the system displays the ID and name of the ECS instance to which the IP address is assigned.

Action	Procedure
Remove IP addresses from the IP address whitelist	To remove all IP addresses from the IP address whitelist and retain the IP address whitelist, click **Delete**.

6. Then, click **OK**.

4.5. Connect to an instance

4.5.1. Use a Redis client

KVStore for Redis is compatible with open source Redis. You can connect to KVStore for Redis and open source Redis in a similar manner. Therefore, you can use a client that is compatible with the Redis protocols to connect to KVStore for Redis. You can connect to a KVStore for Redis instance by using clients of different programming languages.

Prerequisites

The private IP address of an Elastic Compute Service (ECS) instance or the public IP address of an on-premises machine is added to a whitelist of the instance. For more information, see Configure a whitelist.

Obtain connection information

When you use a client to connect to a KVStore for Redis instance, you must obtain the following information and specify the information in the code:

Information	Description
Endpoint	You can find the endpoint in the **Connection Information** section on the **Instance Information** page. **Note** KVStore for Redis instances support multiple types of endpoints. We recommend that you use endpoints in a VPC for higher security and lower network latency.
Port number	The default port number is 6379.
The account of the instance. This parameter is not required by some clients.	By default, a KVStore for Redis instance provides a database account that is named after the instance ID, such as, r-bp10noxlhcoim2****.
The password of the account.	If you forget your password, you can reset the password. For more information, see Change the password.

Commonly used clients

For the list of clients supported by Redis, see Redis clients.

- Jedis client
- PhpRedis client
- Redis-py client
- C or C++ client
- .NET client
- Node-redis client
- C# client StackExchange.Redis

Jedis client

1. Download and install the Jedis client. For more information, see Jedis.
2. Select a connection method to meet your business requirements.
   - JedisPool-based connection: This connection method is recommended.
     a. Launch the Eclipse client, create a project, and then configure the following pom file:

```
<dependency>
<groupId>redis.clients</groupId>
<artifactId>jedis</artifactId>
<version>2.7.2</version>
<type>jar</type>
<scope>compile</scope>
</dependency>
```

     b. Enter the following code in the project to add the relevant applications:

```
import org.apache.commons.pool2.PooledObject;
import org.apache.commons.pool2.PooledObjectFactory;
import org.apache.commons.pool2.impl.DefaultPooledObject;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;
```

     c. Enter the following code in the project based on the Jedis client version and modify the code based on the comments.

> **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

- Jedis 2.7.2

```
JedisPoolConfig config = new JedisPoolConfig();
//Maximum number of idle connections. You can configure this parameter. Make su
re that the specified value does not exceed the maximum number of connections t
hat the KVStore for Redis instance supports.
config.setMaxIdle(200);
//Maximum number of connections. You can configure this parameter. Make sure th
at the specified value does not exceed the maximum number of connections that t
he KVStore for Redis instance supports.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
JedisPool pool = new JedisPool(config, host, 6379, 3000, password);
Jedis jedis = null;
try {
jedis = pool.getResource();
/// ... do stuff here ... for example
jedis.set("foo", "bar");
String foobar = jedis.get("foo");
jedis.zadd("sose", 0, "car");
jedis.zadd("sose", 0, "bike");
Set<String> sose = jedis.zrange("sose", 0, -1);
} finally {
if (jedis != null) {
jedis.close();
}
}
/// ... when closing your application:
pool.destroy();
```

- Jedis 2.6 or Jedis 2.5

```
JedisPoolConfig config = new JedisPoolConfig();
//Maximum number of idle connections. You can configure this parameter. Make su
re that the specified value does not exceed the maximum number of connections t
hat the KVStore for Redis instance supports.
config.setMaxIdle(200);
//Maximum number of connections. You can configure this parameter. Make sure th
at the specified value does not exceed the maximum number of connections that t
he KVStore for Redis instance supports.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
JedisPool pool = new JedisPool(config, host, 6379, 3000, password);
Jedis jedis = null;
boolean broken = false;
try {
     jedis = pool.getResource();
     /// ... do stuff here ... for example
     jedis.set("foo", "bar");
     String foobar = jedis.get("foo");
     jedis.zadd("sose", 0, "car");
     jedis.zadd("sose", 0, "bike");
     Set<String> sose = jedis.zrange("sose", 0, -1);
}
catch(Exception e)
{
     broken = true;
} finally {
if (broken) {
     pool.returnBrokenResource(jedis);
} else if (jedis != null) {
     pool.returnResource(jedis);
 }
}
```

- Single Jedis connection: This connection method does not allow a client to automatically reconnect to the KVStore for Redis instance after a connection times out. Therefore, this connection is not recommended.

  Launch the Eclipse client, create a project, enter the following code, and then modify the code based on the comments.

  > **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

```
import redis.clients.jedis.Jedis;
public class jedistest {
public static void main(String[] args) {
try {
     String host = "xx.kvstore.aliyuncs.com";//You can find the connection address in
the console.
     int port = 6379;
     Jedis jedis = new Jedis(host, port);
     //Authentication information.
     jedis.auth("password");//password
     String key = "redis";
     String value = "aliyun-redis";
     //Select a database. Default value: 0.
     jedis.select(1);
     //Specify a key.
     jedis.set(key, value);
     System.out.println("Set Key " + key + " Value: " + value);
     //Obtain the configured key and value.
     String getvalue = jedis.get(key);
     System.out.println("Get Key " + key + " ReturnValue: " + getvalue);
     jedis.quit();
     jedis.close();
}
catch (Exception e) {
 e.printStackTrace();
 }
}
}
```

3. Run the project. If Eclipse returns the following result, it indicates that the client is connected to the KVStore for Redis instance.

```
Set Key redis Value aliyun-redis
Get Key redis ReturnValue aliyun-redis
```

PhpRedis client

1. Download and install the PhpRedis client. For more information, see PhpRedis.
2. Enter the following code in a PHP editor and modify the code based on the comments.

> **Note** For more information about how to obtain the connection address, account, and password of the KVStore for Redis instance, see Obtain connection information.

```
<?php
 /* Replace the parameter values with the endpoint and port number of the instance. */
 $host = "r-bp10noxlhcoim2****.redis.rds.aliyuncs.com";
 $port = 6379;
 /* Replace the following parameter values with the ID and password of the instance. */
 $user = "test_username";
 $pwd = "test_password";
 $redis = new Redis();
 if ($redis->connect($host, $port) == false) {
         die($redis->getLastError());
   }
 if ($redis->auth($pwd) == false) {
         die($redis->getLastError());
  }
  /* You can manage the database after you pass the authentication. For more informatio
n, visit https://github.com/phpRedis/phpredis. */
 if ($redis->set("foo", "bar") == false) {
         die($redis->getLastError());
 }
 $value = $redis->get("foo");
 echo $value;
 ?>
```

3. Run the preceding code to connect to the instance.

   For more information, see PhpRedis.

Redis-py client

1. Download and install the redis-py client. For more information, see redis-py.
2. Enter the following code in a Python editor and modify the code based on the comments.

> **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
import redis
#Replace the value of the host parameter with the endpoint of the instance and replace
the value of the port parameter with the port number.
host = 'localhost'
port = 6379
#Replace the following parameter value with the password of the instance.
pwd = 'test_password'
r = redis.StrictRedis(host=host, port=port, password=pwd)
#You can perform database operations after you establish a connection. For more informa
tion, visit https://github.com/andymccurdy/redis-py.
r.set('foo', 'bar');
print r.get('foo')
```

3. Run the preceding code to connect to the instance.

C or C++ client

1. Run the following commands to download, compile, and install the C client:

```
git clone https://github.com/redis/hiredis.git
cd hiredis
make
sudo make install
```

2. Enter the following code in a C or C++ editor and modify the code based on the comments.

> **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
unsigned int j;
redisContext *c;
redisReply *reply;
if (argc < 4) {
        printf("Usage: example xxx.kvstore.aliyuncs.com 6379 instance_id password\n");
        exit(0);
}
const char *hostname = argv[1];
const int port = atoi(argv[2]);
const char *instance_id = argv[3];
const char *password = argv[4];
struct timeval timeout = { 1, 500000 }; // 1.5 seconds
c = redisConnectWithTimeout(hostname, port, timeout);
if (c == NULL || c->err) {
if (c) {
        printf("Connection error: %s\n", c->errstr);
        redisFree(c);
} else {
        printf("Connection error: can't allocate redis context\n");
}
exit(1);
}
/* AUTH */
reply = redisCommand(c, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);
/* PING server */
reply = redisCommand(c,"PING");
printf("PING: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key */
reply = redisCommand(c,"SET %s %s", "foo", "hello world");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key using binary safe API */
reply = redisCommand(c,"SET %b %b", "bar", (size_t) 3, "hello", (size_t) 5);
printf("SET (binary API): %s\n", reply->str);
```

```
freeReplyObject(reply);
/* Try a GET and two INCR */
reply = redisCommand(c,"GET foo");
printf("GET foo: %s\n", reply->str);
freeReplyObject(reply);
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
freeReplyObject(reply);
/* again ... */
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
freeReplyObject(reply);
/* Create a list of numbers, from 0 to 9 */
reply = redisCommand(c,"DEL mylist");
freeReplyObject(reply);
for (j = 0; j < 10; j++) {
        char buf[64];
        snprintf(buf,64,"%d",j);
        reply = redisCommand(c,"LPUSH mylist element-%s", buf);
        freeReplyObject(reply);
    }
/* Let's check what we have inside the list */
reply = redisCommand(c,"LRANGE mylist 0 -1");
if (reply->type == REDIS_REPLY_ARRAY) {
        for (j = 0; j < reply->elements; j++) {
        printf("%u) %s\n", j, reply->element[j]->str);
}
}
freeReplyObject(reply);
/* Disconnects and frees the context */
redisFree(c);
return 0;
}
```

3. Compile the code.

```
gcc -o example -g example.c -I /usr/local/include/hiredis -lhiredis
```

4. Perform a test run and connect to the instance.

```
example xxx.kvstore.aliyuncs.com 6379 instance_id password
```

.NET client

Warning If you need to switch or select a database from multiple databases in a cluster instance, you must set the cluster_compat_enable parameter to *0* and restart the client application. This disables the support for the cluster syntax of open source Redis. Otherwise, the system sends the following error message: `Multiple databases are not supported on this server; cannot switch to database`. For more information, see Parameter configuration.

1. Run the following command to download the .NET client.

```
git clone https://github.com/ServiceStack/ServiceStack.Redis
```

2. Create a .NET project on the .NET client.
3. Add a reference. The reference file is stored in the library file directory ServiceStack.Redis/lib/tests.
4. Enter the following code in the .NET project and modify the code based on the comments. For more information, see ServiceStack.Redis.

> **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

```
using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Text;
 using System.Threading.Tasks;
 using ServiceStack.Redis;
 namespace ServiceStack.Redis.Tests
 {
         class Program
 {
 public static void RedisClientTest()
 {
         string host = "127.0.0.1";/*The endpoint of the host.*/
         string password = "password";/*The password*/
         RedisClient redisClient = new RedisClient(host, 6379, password);
         string key = "test-aliyun";
         string value = "test-aliyun-value";
         redisClient.Set(key, value);
         string listKey = "test-aliyun-list";
         System.Console.WriteLine("set key " + key + " value " + value);
         string getValue = System.Text.Encoding.Default.GetString(redisClient.Get(key))
;
         System.Console.WriteLine("get key " + getValue);
         System.Console.Read();
 }
 public static void RedisPoolClientTest()
 {
         string[] testReadWriteHosts = new[] {
         "redis://password@127.0.0.1:6379"/* redis://password@endpoint:port */
 };
 RedisConfig.VerifyMasterConnections = false;//Required.
 PooledRedisClientManager redisPoolManager = new PooledRedisClientManager(10/*Number of
connection pools*/, 10/*Connection pool timeout value*/, testReadWriteHosts);
 for (int i = 0; i < 100; i++){
         IRedisClient redisClient = redisPoolManager.GetClient();//Obtain the connectio
n.
         RedisNativeClient redisNativeClient = (RedisNativeClient)redisClient;
         redisNativeClient.Client = null;//KVStore for Redis does not support the CLIEN
T SETNAME command. Set Client to null.
 try
 {
         string key = "test-aliyun1111";
         string value = "test-aliyun-value1111";
         redisClient.Set(key, value);
```

```
        string listKey = "test-aliyun-list";
        redisClient.AddItemToList(listKey, value);
        System.Console.WriteLine("set key " + key + " value " + value);
        string getValue = redisClient.GetValue(key);
        System.Console.WriteLine("get key " + getValue);
        redisClient.Dispose();//
}catch (Exception e)
{
        System.Console.WriteLine(e.Message);
}
}
        System.Console.Read();
}
static void Main(string[] args)
{
        //Single-connection mode.
        RedisClientTest();
        //Connection-pool mode.
        RedisPoolClientTest();
}
}
}
```

Node-redis client

1. Download and install the node-redis client.

```
npm install hiredis redis
```

2. Enter the following code in the node-redis client and modify the code based on the comments.

> **Note** For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see Obtain connection information.

```
var redis = require("redis"),
client = redis.createClient(<port>, <"host">, {detect_buffers: true});
client.auth("password", redis.print)
```

Parameters:

- <port>: the service port number of the KVStore for Redis database. The default port number is 6379.
- <"host">: the endpoint of the KVStore for Redis instance.

Configuration examples:

```
var redis = require("redis"),
client = redis.createClient(6379, "r-abcdefg.redis.rds.aliyuncs.com", {detect_buffers:
true});
client.auth("password", redis.print)
```

3. Run the preceding code to connect to the KVStore for Redis instance.
4. Use KVStore for Redis.

```
//Write data to the instance.
 client.set("key", "OK");
 //Query data on the instance. The returned data is of the STRING type.
 client.get("key", function (err, reply) {
 console.log(reply.toString()); // print `OK`
 });
 //If the input parameter is a buffer, the returned value is also a buffer.
 client.get(new Buffer("key"), function (err, reply) {
 console.log(reply.toString()); // print `<Buffer 4f 4b>`
 });
 client.quit();
```

C# client StackExchange.Redis

Warning If you need to switch or select a database from multiple databases in a cluster instance, you must set the cluster_compat_enable parameter to *0* and restart the client application. This disables the support for the cluster syntax of open source Redis. Otherwise, the system sends the following error message: `RedisCommandException: Multiple databases are not supported on this server; cannot switch to database: 1` . For more information, see Parameter configuration.

1. Download and install the StackExchange.Redis client.
2. Add a reference.

```
using StackExchange.Redis;
```

3. Initialize ConnectionMultiplexer.

   ConnectionMultiplexer is the core of StackExchange.Redis and is shared and reused in the entire application. You must use ConnectionMultiplexer as a singleton. ConnectionMultiplexer is initialized in the following method:

   Note

   - For more information about how to obtain the connection endpoint and password of the KVStore for Redis instance, see .
   - ConfigurationOptions contains multiple options, such as keepAlive, connectRetry, and name. For more information, see ConfigurationOptions.

```
// redis config
private static ConfigurationOptions configurationOptions = ConfigurationOptions.Parse(
"127.0.0.1:6379,password=xxx,connectTimeout=2000");
 //the lock for singleton
private static readonly object Locker = new object();
 //singleton
private static ConnectionMultiplexer redisConn;
//singleton
public static ConnectionMultiplexer getRedisConn()
{
    if (redisConn == null)
    {
        lock (Locker)
        {
            if (redisConn == null || !redisConn.IsConnected)
            {
                redisConn = ConnectionMultiplexer.Connect(configurationOptions);
            }
        }
    }
    return redisConn;
}
```

4. GetDatabase() returns a lightweight object. You can obtain this object from the object of ConnectionMultiplexer.

```
redisConn = getRedisConn();
var db = redisConn.GetDatabase();
```

5. You can use the client to perform database operations.

> **Note** The following examples describe the commands for common data types. These commands are slightly different from the Redis-native commands.

- String

```
//set get
string strKey = "hello";
string strValue = "world";
bool setResult = db.StringSet(strKey, strValue);
Console.WriteLine("set " + strKey + " " + strValue + ", result is " + setResult);
//incr
string counterKey = "counter";
long counterValue = db.StringIncrement(counterKey);
Console.WriteLine("incr " + counterKey + ", result is " + counterValue);
//expire
db.KeyExpire(strKey, new TimeSpan(0, 0, 5));
Thread.Sleep(5 * 1000);
Console.WriteLine("expire " + strKey + ", after 5 seconds, value is " + db.StringGet(
strKey));
//mset mget
KeyValuePair<RedisKey, RedisValue> kv1 = new KeyValuePair<RedisKey, RedisValue>("key1
", "value1");
KeyValuePair<RedisKey, RedisValue> kv2 = new KeyValuePair<RedisKey, RedisValue>("key2
", "value2");
db.StringSet(new KeyValuePair<RedisKey, RedisValue>[] {kv1,kv2});
RedisValue[] values = db.StringGet(new RedisKey[] {kv1.Key, kv2.Key});
Console.WriteLine("mget " + kv1.Key.ToString() + " " + kv2.Key.ToString() + ", result
is " + values[0] + "&&" + values[1]);
```

- Hash

```
string hashKey = "myhash";
//hset
db.HashSet(hashKey,"f1","v1");
db.HashSet(hashKey,"f2", "v2");
HashEntry[] values = db.HashGetAll(hashKey);
//hgetall
Console.Write("hgetall " + hashKey + ", result is");
for (int i = 0; i < values.Length;i++)
{
  HashEntry hashEntry = values[i];
  Console.Write(" " + hashEntry.Name.ToString() + " " + hashEntry.Value.ToString());
}
Console.WriteLine();
```

- List

```
//list key
string listKey = "myList";
//rpush
db.ListRightPush(listKey, "a");
db.ListRightPush(listKey, "b");
db.ListRightPush(listKey, "c");
//lrange
RedisValue[] values = db.ListRange(listKey, 0, -1);
Console.Write("lrange " + listKey + " 0 -1, result is ");
for (int i = 0; i < values.Length; i++)
{
 Console.Write(values[i] + " ");
}
Console.WriteLine();
```

- Set

```
//set key
string setKey = "mySet";
//sadd
db.SetAdd(setKey, "a");
db.SetAdd(setKey, "b");
db.SetAdd(setKey, "c");
//sismember
bool isContains = db.SetContains(setKey, "a");
Console.WriteLine("set " + setKey + " contains a is " + isContains );
```

- Sorted Set

```
string sortedSetKey = "myZset";
//sadd
db.SortedSetAdd(sortedSetKey, "xiaoming", 85);
db.SortedSetAdd(sortedSetKey, "xiaohong", 100);
db.SortedSetAdd(sortedSetKey, "xiaofei", 62);
db.SortedSetAdd(sortedSetKey, "xiaotang", 73);
//zrevrangebyscore
RedisValue[] names = db.SortedSetRangeByRank(sortedSetKey, 0, 2, Order.Ascending);
Console.Write("zrevrangebyscore " + sortedSetKey + " 0 2, result is ");
for (int i = 0; i < names.Length; i++)
{
  Console.Write(names[i] + " ");
}
Console.WriteLine();
```

4.5.2. Use redis-cli

The redis-cli tool is a command-line interface (CLI) of Redis. You can use redis-cli to connect to a KVStore for Redis instance from an Elastic Compute Service (ECS) instance or on-premises machine and manage data.

Prerequisites

- The ECS instance and KVStore for Redis instance are deployed in the same classic network or virtual private cloud (VPC).

- The private IP address of the ECS instance is added to the whitelist of the KVStore for Redis instance. For more information, see Configure a whitelist.
- The ECS instance runs a Linux operating system and open source Redis is installed. For more information, see Redis official website.

Procedure

1. Log on to the CLI of the ECS instance and run the following command to connect to the Redis instance:

```
src/redis-cli -h <hostname> -p <port>
```

Parameters

Parameter	Description
<hostname>	The internal endpoint of the KVStore for Redis instance. You can find the endpoint in the **Connection Information** section on the **Instance Information** page.
<port>	The service port number of the KVStore for Redis instance. Default value: 6379.

Example

```
src/redis-cli -h r-bp1zxszhcgatnx****.redis.rds.aliyuncs.com -p 6379
```

2. Run the following command to verify the password:

```
AUTH <password>
```

<password>: the password that you specify when you create the instance. If you forget your password, you can reset the password. For more information, see Change the password.

Example:

```
AUTH testaccount:Rp829dlwa
```

If the password verification is successful, the following result is returned:

```
OK
```

5.Instance management

5.1. Change a password

If you forget your password, need to change your password, or have not set a password for an instance, you can set a new password for the instance.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the upper-right corner of the **Basic Information** page, click **Modify Password**.
4. In the dialog box that appears, enter the current password and a new password.

   > **Note**
   > - If you forget your password, you can click **Forgot password** in the Change Password dialog box and enter a new password.
   > - The password must be 8 to 32 characters in length.
   > - The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include `! @ # $ % ^ & * ( ) _ + - =` .

5. Click **OK**.

5.2. Configure a whitelist

Before you use a KVStore for Redis instance, add IP addresses or CIDR blocks that are used to access the database to the whitelist of the instance to improve the security and stability of the database.

Context

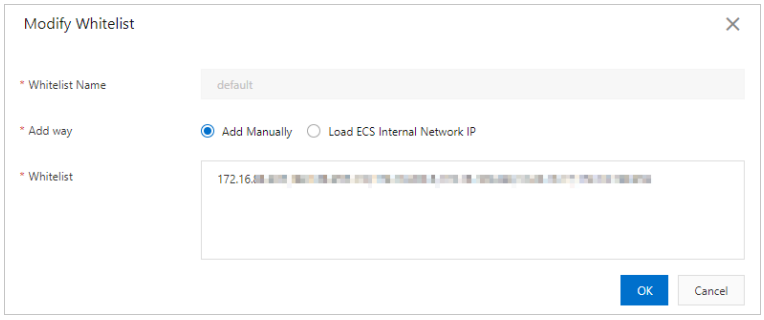
> **Note** A properly configured whitelist can ensure a higher level of security protection for your KVStore for Redis instance. We recommend that you maintain the whitelist on a regular basis.

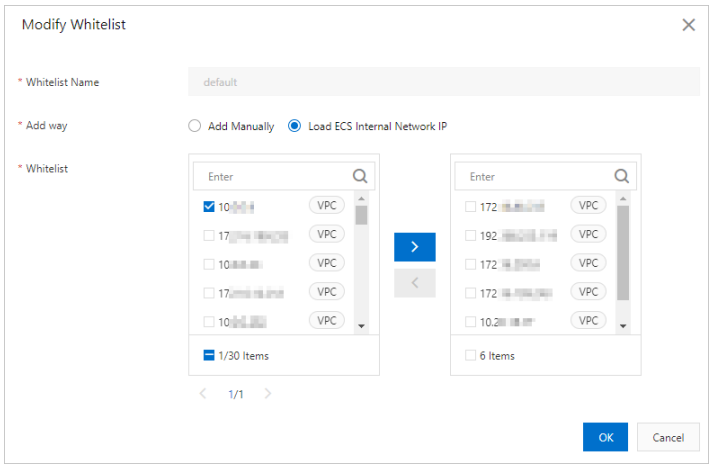
Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Whitelist Settings**.
4. Find the IP address whitelist that you want to manage and click **Modify**.

> **Note** You can also click **Add Whitelist** to create an IP address whitelist. The name of the IP address whitelist must be 2 to 32 characters in length and can contain lowercase letters, digits, and underscores (_). The name of the whitelist must start with a lowercase letter and end with a lowercase letter or digit.

5. In the dialog box that appears, perform one of the following operations:

Action	Procedure
Manually modify the IP address whitelist	Enter IP addresses or CIDR blocks. Manually modify the IP address whitelist  **Note** ○ Separate multiple IP addresses with commas (,). You can add up to 1,000 unique IP addresses. Supported formats are specific IP addresses such as 10.23.12.24 and CIDR blocks such as 10.23.12.24/24. /24 indicates the length of the IP address prefix. An IP address prefix can be 1 to 32 bits in length. ○ If you set the prefix length to 0, for example, 0.0.0.0/0 or 127.0.0.1/0, all IP addresses are allowed to access the instance. In this case, the security risk of your instance is high. Proceed with caution.

Action	Procedure
Add private IP addresses of ECS instances to an IP address whitelist	i. Click **Load ECS Internal Network IP**. ii. Select IP addresses based on your business requirements. Add private IP addresses of ECS instances  **Note** To find the ECS instance to which a specific IP address is assigned, you can move the pointer over the IP address. Then, the system displays the ID and name of the ECS instance to which the IP address is assigned.
Remove IP addresses from the IP address whitelist	To remove all IP addresses from the IP address whitelist and retain the IP address whitelist, click **Delete**.

6. Then, click **OK**.

5.3. Change configurations

This topic describes how to change the configuration of a KVStore for Redis instance.

Precautions

After configuration changes, the system migrates data and switches the instance. The instance is disconnected for a few seconds during this process. We recommend that you upgrade or downgrade the instance during off-peak hours.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. On the page that appears, configure the required parameters.

Parameter	Description
Architecture Type	The architecture type of the instance. ○ **Standard**: runs in a master-replica architecture, provides high-performance caching services, and ensures high data reliability. ○ **Cluster**: eliminates the performance bottleneck that is caused by a single-threading model. You can use the high-performance cluster instance to process large-capacity workloads. ○ **Read/Write Splitting**: ensures high availability and high performance, and supports multiple specifications. The read/write splitting architecture allows a large number of concurrent requests to read hot data from read replicas. This reduces the loads on the master node and minimizes the O&M cost.
Instance Type	The specifications of the instance. The maximum connections and maximum internal network bandwidth vary based on the instance type.

4. Click **Submit**.

5.4. Specify a maintenance window

You can modify the default maintenance window to perform maintenance on KVStore for Redis during off-peak hours.

Context

To ensure the stability of KVStore for Redis instances on the Alibaba Cloud platform, the backend system performs maintenance on instances and servers occasionally.

To guarantee the stability of the maintenance process, instances will enter the Maintaining Instance status before the preset maintenance window on the day of maintenance. While an instance is in this state, data in the database can still be accessed and query operations such as performance monitoring are still available. However, change operations such as configuration change are temporarily unavailable for this instance in the console.

> **Note** During the maintenance process, instances may be disconnected in the process of maintenance. We recommend that you set the maintenance window to a period during off-peak hours.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the **Basic Information** section, click **Settings** on the right of **Maintenance Window**.
4. Select a time period for maintenance and click **Save**.

   > **Note** The time periods are in UTC+8.

5.5. Upgrade the minor version

Alibaba Cloud has continuously optimized the kernel of KVStore for Redis to fix security vulnerabilities and provide more stable services. You can upgrade the kernel version (minor version) of a KVStore for Redis instance with one click in the console.

Context

> **Note**
> - We recommend that you upgrade instance versions during off-peak hours and ensure that your application supports automatic reconnection.
> - The system automatically checks the kernel version of an instance. If the current version is the latest, the **Minor Version Upgrade** button in the upper-right corner of the **Basic Information** section for this instance will appear dimmed.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. On the **Instance Information** page, click **Minor Version Upgrade** in the upper-right corner of the **Basic Information** section.
4. In the **Minor Version Upgrade** dialog box that appears, click **Upgrade Now**.

   On the **Instance Information** page, the instance status will become Upgrading a minor version. When the instance status returns to Available, the upgrade has been completed.

5.6. Configure SSL encryption

This topic describes how to enable SSL encryption for a KVStore for MongoDB instance to enhance link security. After you enable SSL encryption, you must install SSL certificates that are issued by certificate authorities (CAs) on your application. SSL encryption can encrypt connections at the transport layer to increase data security and ensure data integrity.

Prerequisites

- The major version of your KVStore for Redis instance is Redis 2.8. The instance can be a standard instance or a cluster instance.
- The major version of the instance is Redis 4.0 or 5.0. The instance is a cluster instance.

Context

> **Note** SSL encryption may increase the network response time of instances. We recommend that you enable this feature only when necessary.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis

console.

2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **SSL Settings**.
4. In the upper-right corner of the **SSL Settings** page, click **Configure SSL**.
5. In the dialog box that appears, turn on **Enable SSL Certificate**.

> **Note** If this option is not supported in the current version of the instance, update the minor version. For more information, see Upgrade the minor version.

6. Click **OK**.

> **Note**
> - The result of the operation is displayed after a short period of time.
> - In the upper-right corner of the SSL Settings page, you can also click **Update Validity** and **Download CA Certificate** to perform relevant operations.

5.7. Enable TDE

KVStore for Redis provides Transparent Data Encryption (TDE) that can be used to encrypt and decrypt Redis Database (RDB) files. You can enable TDE in the KVStore for Redis console to allow the system to encrypt and decrypt RDB files. This improves data security and compliance.

Prerequisites

- The KVStore for Redis instance is a KVStore for Redis Enhanced Edition (Tair) instance.
- The instance uses the latest minor version. For more information about how to update the minor version, see Upgrade the minor version.

Context

TDE encrypts RDB files before they are written to disks and decrypts RDB files when they are read to the memory from disks. TDE does not increase the sizes of RDB files. When you use TDE, you do not need to modify your client.

Impacts

You cannot disable TDE after it is enabled. You must evaluate the impacts on your business before you enable TDE. Take note of the following impacts:

- After TDE is enabled for an instance, the cache analysis feature is not supported for the instance. For more information, see Cache analysis.
- After TDE is enabled for an instance, instance data cannot be migrated or synchronized by using Data Transmission Service (DTS) or redis-shake.

Precautions

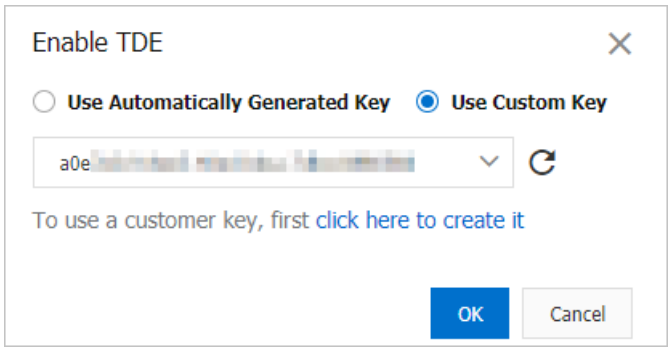
- TDE can be enabled for an instance but not for a key or a database.
- TDE encrypts RDB files that are written to disks, such as *dump.rdb*.
- Key Management Service (KMS) generates and manages the keys used by TDE. KVStore for Redis does

not provide keys or certificates required for encryption.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **TDE**.
4. Turn on the switch next to **TDE Status** to enable TDE.
5. In the dialog box that appears, select **Use Automatically Generated Key** or **Use Custom Key** and then click **OK**.

   Select key type for enabling TDE

   Note

   - The first time you enable TDE for an instance within your Alibaba Cloud account, follow the instructions on the page to assign the **AliyunRdsInstanceEncryptionDefaultRole** role to KVStore for Redis. KVStore for Redis can access KMS resources only after it assumes the role.
   - For more information about how to create a custom key, see **Create a CMK** in **User Guide** of *KMS*.

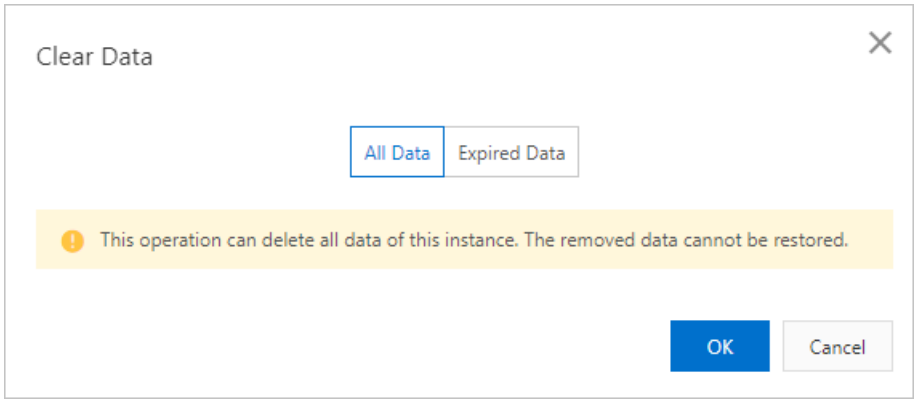
   When the instance state changes from **Modifying TDE** to **Running**, the configurations are complete.

5.8. Delete data

You can delete all data or expired data of an ApsaraDB for Redis instance in the ApsaraDB for Redis console.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the upper-right corner of the **Instance Information** page, click **Clear Data**.
4. In the dialog box that appears, select the data that you want to delete.



- **All Data**: runs the **FLUSHALL** command to delete all data of the instance. Deleted data cannot be recovered.
- **Expired Data**: runs the **SCAN** command to delete all expired data of the instance. Deleted data cannot be recovered. You can select **Update Now** or Update During Maintenance. For more information, see Set a maintenance window.

> **Warning** Data deletion immediately takes effect and deleted data cannot be recovered. This may affect your business. Proceed with caution. We recommend that you back up the data of an ApsaraDB for Redis instance before you delete data. For more information, see Back up data manually.

5. In the message that appears, click **OK**.

> **Note** If you select **All Data**, you can select whether to back up the data after you click **OK**.

5.9. Release an instance

You can release a KVStore for Redis instance at any time based on your business needs. This topic describes how to release a KVStore for Redis instance.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instances** page, click the instance ID or choose ⋮ > **Release** in the **Actions** column.

> **Warning** After an instance is released, the instance cannot be restored. Proceed with caution. We recommend that you back up your data before you release an instance.

3. In the **Release Instance** message that appears, click **OK**.

5.10. Manage database accounts

KVStore for Redis allows you to create up to 20 database accounts for an instance. You can grant permissions to these accounts and manage your instance to prevent user errors.

Prerequisites

The engine version of the instance is Redis 4.0 or later.

> **Note** If the engine version of the instance is not Redis 4.0, only the default account that is created when you create the instance is available. For more information about how to change the password of the default account, see Change the password.

Context

You can create accounts, delete accounts, reset the password, and change the permissions. After an account is created, you can use this account to log on to the database and use the command-line tool to perform operations on the database with the account and granted permissions.

Create an account

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. On the **Instances** page, find the instance that you want to manage and click the instance ID. In the left-side navigation pane, click **Account Management**.

   > **Note** If Account Management is unavailable for an instance of Redis 4.0 or later, you must upgrade the minor version of the instance. For more information, see Upgrade the minor version.

4. In the upper-right corner of the **Account Management** page, click **Create**.
5. In the dialog box that appears, configure the required parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Account	The account name must be 1 to 16 characters in length. The name can contain lowercase letters, digits, and underscores (_) and must start with a letter.
Privilege	Specify the permissions that are granted to the account. Valid values: Read-only, Read/Write, and Replicate. If you select Replicate, you can run the SYNC and PSYNC commands after you connect to an instance by using your account. **Note** You can create accounts that have the replicate permissions only for standard instances of Redis 4.0 or later.
Password Settings	Specify a password for the account. The password must be 8 to 32 characters in length. The password must contain at least three of the following types of characters: uppercase letters, lowercase letters, digits, and special characters. The following special characters are supported: `!@ # $ % ^ & * ( ) + - = _` .
Confirm Password	Enter the password again.
Description	The description of the account.

5.11. Restart an instance

You can restart an instance from the Instance List page of the console.

Procedure

1. Log on to the KVStore for Redis console.
2. In the **Actions** column, choose ⋮ > **Restart**.

   Warning During the restart, the instance may be disconnected for a few seconds. We recommend that you restart instances during off-peak hours. You must also make sure that your application supports automatic reconnection.

3. In the dialog box that appears, specify the upgrade time and click **OK**.
   - Restart Immediately: restarts the instance immediately.
   - Restart Within Maintenance Window: restarts the instance within the preset maintenance window.

5.12. Export the list of instances

You can export the list of KVStore for Redis instances from the KVStore for Redis console for offline management.

Procedure

1. Log on to the KVStore for Redis console.
2. In the upper-right corner of the **Instance List** page, click the **Export Instances** icon.
3. In the upper-right corner of the **Instance List** page, click the **Export Instances** icon.
4. In the **Export Instance List** dialog box that appears, select the columns to export and click **OK**.

   Note After you click **OK**, the browser begins to download the CSV file. You can use Excel or a text editor to view this file.

5.13. Use a Lua script

KVStore for Redis instances of all editions support Lua commands.

Support for Lua commands

Lua scripts improve the performance of KVStore for Redis. With support for the Lua environment, KVStore for Redis is able to perform check-and-set (CAS) operations, allowing you to combine and run multiple commands in an efficient manner.

> **Note** If the EVAL command cannot be executed, for example, the "`ERR command eval not support for normal user`" message appears, you can Upgrade the minor version. During the upgrade, the instance may be disconnected and become read-only for a few seconds. We recommend that you upgrade instance versions during off-peak hours.

Limits on Lua scripts

A Lua script, which is supported by the cluster instance of KVStore for Redis, has the following limits to ensure that all operations in the script are performed within the same hash slot:

- The Lua script uses the **redis.call/redis.pcall** function to run Redis commands. . For Redis commands, the keys must be passed by using the KEYS array, which cannot be replaced by Lua variables. If the KEYS array is not used, the following error message is returned:

```
-ERR bad lua script for redis cluster, all the keys that the script uses should be passed using the KEYS array\r\n
```

- All keys that are used by the script must be allocated to the same hash slot. If the keys are allocated to different hash slots, the following error message is returned:

```
-ERR eval/evalsha command keys must be in same slot\r\n
```

- Keys must be included in all commands that you want to run. If the keys are not included in all commands, the following error message is returned:

```
-ERR for redis cluster, eval/evalsha number of keys can't be negative or zero\r\n
```

- The following Pub/Sub commands are not supported: **PSUBSCRIBE**, **PUBSUB**, **PUBLISH**, **PUNSUBSCRIBE**, **SUBSCRIBE**, and **UNSUBSCRIBE**.
- The UNPACK function is not supported.

> **Note** If you do not want to impose the preceding limits but can make sure that all operations are performed in the same hash slot in the code, you can set the script_check_enable parameter to 0 in the console to disable the backend script check.

6.Connection management

6.1. View endpoints

You can view the internal and public endpoints of instances in the KVStore for Redis console.

Context

> **Note**
> - The virtual IP address of a KVStore for Redis instance may change when you maintain or modify the instance. To ensure that the connection is available, we recommend that you use an endpoint to access the KVStore for Redis instance.
> - For more information about how to apply for a public endpoint, see Applies for a public connection string.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the **Connection Information** section, you can view the private and public endpoints of the instance.

> **Note** By default, only a private endpoint is provided by a KVStore for Redis instance. If you want to connect to a KVStore for Redis instance over the Internet, you must apply for a public endpoint. For more information, see Apply for a public endpoint.

6.2. Apply for a public endpoint

This topic describes how to apply for a public endpoint for a KVStore for Redis instance.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the **Connection Information** section, click **Apply for Public Endpoint**.
4. In the dialog box that appears, enter an endpoint and a port number and click **OK**.

> **Note**
> - The custom endpoint must be 8 to 64 characters in length and can contain lowercase letters and digits. The endpoint must start with a lowercase letter.
> - The custom port number ranges from 1024 to 65535. The default value is 6379.
> - After you apply for a public endpoint, you must add the public endpoint to the IP address whitelist of the instance. This way, you can connect to the instance over the Internet. For more information, see Configure a whitelist.

5. On the **Instance Information** page, view the **Public Endpoint** in the **Connection Information** section.

> **Note** If you no longer use the public endpoint, click **Release Public Endpoint** next to **Public Endpoint** to release the endpoint.

6.3. Modify the endpoint of an KVStore for Redis instance

KVStore for Redis allows you to modify internal and public endpoints for instances. When changing the KVStore for Redis instance, you can change the endpoint of the new instance to the endpoint of the original instance without modifying the application.

Prerequisites

The instance is in the Running state.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. On the **Instance Information** page, click **Modify Public Endpoint** in the **Connection Information** section.
4. In the **Modify Public Endpoint** dialog box, set the following parameters:

Parameter	Description
Connection type	Select **Internal Endpoint** or **Public Endpoint**.
Endpoint	Set the prefix of the endpoint. ○ The endpoint can contain lowercase letters and digits. ○ It must start with a lowercase letter. ○ The endpoint must be 8 to 64 characters in length.
Port	Specify a port number. Valid value: 1024 to 65535. **Note** It takes about 10 minutes for the modified port number of the public endpoint to take effect. You can refresh the page to view the latest port number information.

5. In the **Modify Public Endpoint** dialog box, modify **Connection Type**, **Endpoint**, and **Port**, and then click **OK**.

Note

- The custom endpoint prefix must be 8 to 64 characters in length and can contain lowercase letters and digits. It must start with a lowercase letter.
- The custom port number ranges from 1024 to 65535. The default value is 6379.

7.Performance monitoring

7.1. Query monitoring data

You can query the monitoring data of a KVStore for Redis instance for a specified period within the last month.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Performance Monitor**.
4. Select the start and end time and click **OK**.

> **Note** For more information about the metrics, see Understand metrics.

7.2. Select metrics

You can select the metrics to be displayed on the Historical Monitoring Data page of the KVStore for Redis console as needed.

Context

KVStore for Redis supports more than 10 monitoring groups. By default, the Performance Monitor page displays the metrics of the basic monitoring group. You can click **Customize Metrics** to switch to the metrics of other monitoring groups. The following table describes the monitoring groups.

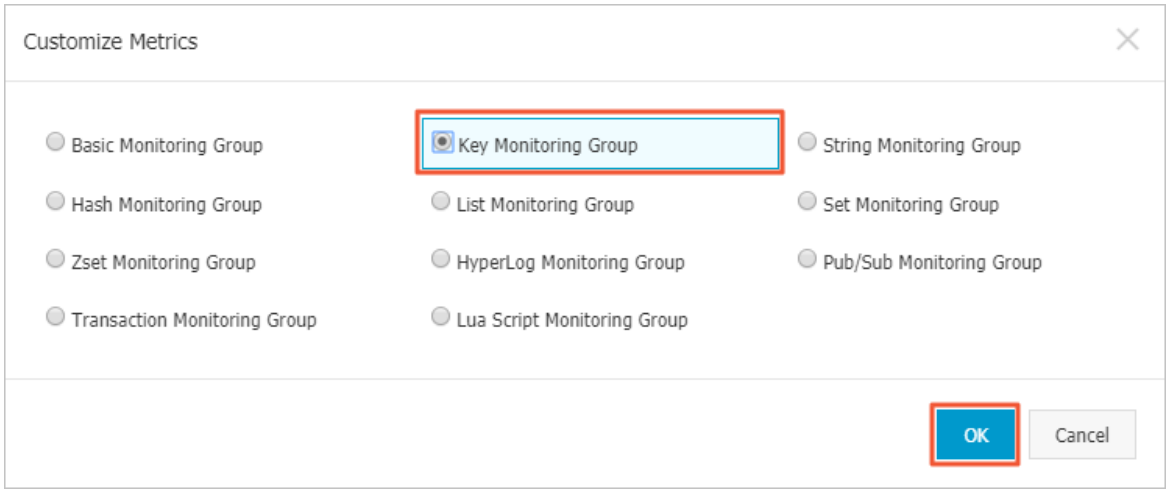
Monitoring group	Description
Basic monitoring group	The basic monitoring information about an instance, such as the queries per second (QPS), bandwidth, and memory usage of the instance.
Key monitoring group	The monitoring information about the use of key-value related commands, such as the number of times that the DEL and EXITS commands are executed.
String monitoring group	The monitoring information about the use of string commands, such as the number of times that the APPEND and MGET commands are executed.
Hash monitoring group	The monitoring information about the use of hash commands, such as the number of times that the HGET and HDEL commands are executed.
List monitoring group	The monitoring information about the use of list commands, such as the number of times that the BLPOP and BRPOP commands are executed.
Set monitoring group	The monitoring information about the use of set commands, such as the number of times that the SADD and SCARD commands are executed.
Zset monitoring group	The monitoring information about the use of zset commands, such as the number of times that the ZADD and ZCARD commands are executed.

Monitoring group	Description
HyperLog monitoring group	The monitoring information about the use of HyperLogLog commands, such as the number of times that the PFADD and PFCOUNT commands are executed.
Pub/Sub monitoring group	The monitoring information about the use of publication and subscription commands, such as the number of times that the PUBLISH and SUBSCRIBE commands are executed.
Transaction monitoring group	The monitoring information about the use of transaction commands, such as the number of times that the WATCH, MULTI, and EXEC commands are executed.
Lua script monitoring group	The monitoring information about the use of Lua script commands, such as the number of times that the EVAL and SCRIPT commands are executed.

For more information about the definitions of the metrics in each monitoring group, see Understand metrics.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Performance Monitor**.
4. On the **Historical Monitoring Data** page, click **Customize Metrics** in the **Data Index** section.
5. In the dialog box that appears, specify a monitoring group and click **OK**.



On the Historical Monitoring Data page, the metrics in the selected monitoring group appear.

7.3. Modify the data collection interval

KVStore for Redis console allows you to set the frequency at which monitoring data is collected.

Context

You can set the monitoring frequency to either 5 or 60 seconds to specify how often monitoring data to be collected by KVStore for Redis. The default monitoring time of 60 seconds is sufficient to meet common monitoring requirements. If you need to observe certain metrics at a higher frequency and lower latency, you can change the monitoring frequency to 5 seconds as described in the following section. Monitoring data does not occupy instance storage space, and collection of monitoring data does not affect normal running of the instance.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Performance Monitor**.
4. In the upper-right corner of the **Historical Monitoring Data** page, click **Monitoring Frequency**.
5. In the **Monitoring Frequency** dialog box that appears, select the new monitoring frequency and click **OK**.

7.4. Understand metrics

KVStore for Redis updates more than 10 monitoring groups of metrics in real time. This allows you to monitor the status of KVStore for Redis instances. This topic describes the metrics of the monitoring groups.

Metrics of basic monitoring groups

Metric	Unit	Description	Statistical method
CpuUsage	%	The CPU utilization	Check the CPU utilization when the monitoring data is collected.
UsedMemory	Bytes	The amount of the used memory.	Check the memory usage when the monitoring data is collected.
TotalQps	Counts/s	The number of requests that are received by the instance per second.	Divide the number of requests in a monitoring cycle by the number of seconds in the monitoring cycle.
ConnCount	Counts	The number of connections.	Check the number of connections when collecting monitoring data.
InFlow	KBps	The amount of data received by the instance per second.	Divide the amount of data that is received in a monitoring cycle by the number of seconds in the monitoring cycle.
OutFlow	KBps	The amount of data sent by the instance per second.	Divide the amount of data that is sent in a monitoring cycle by the number of seconds in the monitoring cycle.

Metric	Unit	Description	Statistical method
FailedCount	Counts/s	The average number of abnormal requests per second.	Divide the total number of abnormal requests in a monitoring cycle by the number of seconds in the monitoring cycle.
AvgRt	us	The average response time of all requests. Note For more information, see Response time (RT) metrics.	Divide the processing time of all requests in a monitoring cycle by the number of requests in the monitoring cycle.
MaxRt	us	The maximum response time of requests. Note For more information, see Response time (RT) metrics.	The maximum time that is required to process a request in a monitoring cycle.
Keys	Counts	The total number of keys.	The number of keys when the monitoring data is collected.
Expires	Counts	The total number of keys for which an expiration time is configured.	The total number of keys for which an expiration time is set when the monitoring data is collected.
ExpiredKeys	Counts	The total number of expired keys.	The cumulative sum when the monitoring data is collected. After the instance is restarted, the cumulative sum is calculated again.
EvictedKeys	Counts	The total number of keys that are evicted because the memory is exhausted.	The cumulative sum when the monitoring data is collected. After the instance is restarted, the cumulative sum is calculated again.
request	Bytes	The total amount of request data received by KVStore for Redis nodes in a monitoring cycle.	See the description of this metric.
response	Bytes	The total amount of response data sent by KVStore for Redis nodes in a monitoring cycle.	See the description of this metric.
request_max	Bytes	The maximum amount of data in a request in a monitoring cycle.	See the description of this metric.

Metric	Unit	Description	Statistical method
response_max	Bytes	The maximum amount of data in a response in a monitoring cycle.	See the description of this metric.
traffic_control_input	Counts	The number of times that downstream throttling is triggered.	Monitor the cumulative sum in a monitoring cycle.
traffic_control_output	Counts	The number of times that uplink throttling is triggered.	Monitor the cumulative sum in a monitoring cycle.
traffic_control_input_status	Counts	Indicates whether downstream throttling was triggered in a monitoring cycle. A value of 0 indicates that throttling was not triggered. A value of 1 indicates that throttling was triggered.	See the description of this metric.
traffic_control_output_status	Counts	Indicates whether upstream throttling was triggered in a monitoring cycle. A value of 0 indicates that throttling was not triggered. A value of 1 indicates that throttling was triggered.	See the description of this metric.
hit_rate	%	The request hit rate, which is the probability that data exists in a KVStore for Redis instance for a data access request.	Calculate the percentage of the hit requests to the total number of requests in a monitoring cycle.
hit	Counts	The number of hit requests.	Check the number of hit requests in a monitoring cycle.
miss	Counts	The number of missed requests.	Check the number of missed requests in a monitoring cycle.
evicted_keys_per_sec	Counts/s	The number of keys that are evicted per second.	Divide the total number of keys that are evicted in a monitoring cycle by the number of seconds in the monitoring cycle.

Metrics in other monitoring groups

The system also uses other metrics to monitor specific types of data or specific features. The metrics are classified into:

- Metrics that indicate the number of times that commands are used. For example, the del, dump, and exists metrics that are used to monitor keys indicate the number of times that the DEL, DUMP, and EXISTS commands are executed.
- Response time (RT) metrics of commands. For example, the metrics that end with avg_rt, such as del_avg_rt dump_avg_rt, and exists_avg_rt, in the key monitoring group are used to monitor the average response time of the DEL, DUMP, and EXISTS commands in a monitoring cycle.

Response time (RT) metrics

All monitoring groups have RT metrics. RT metrics end with Rt or rt. For example, the AvgRt and MaxRt metrics are used in the basic monitoring group and the del_avg_rt and exists_avg_rt metrics are used to monitor keys.

The AvgRt and MaxRt metrics in the basic monitoring group are the most frequently used RT metrics. These metrics have different meanings for proxy nodes and data nodes.

- For a cluster instance or a read/write splitting instance, the AvgRt metric of a proxy node indicates the average time consumed by the proxy node to process all commands. The following process shows how a proxy node processes a command:
  i. The proxy node receives a command and forwards the command to a data node.
  ii. The data node processes the command and responds to the proxy node.
  iii. The proxy node returns the command processing result.

  The AvgRt metric of the proxy node includes the amount of time consumed by the data node to process a command and the time that is required to wait for the command to be processed. This metric also includes the amount of time consumed for network communication between the proxy node and the data node.
- For data nodes of a cluster instance or a read/write splitting instance or for a standard instance, the AvgRt metric indicates the average time consumed by a data node to process all commands. This metric records the period of time from the time when the data node receives the command to the time when the data node returns the result. This metric does not include the time consumed by the proxy node to process a command and the time that is required for network communication.
- The MaxRt metric indicates the maximum response time of requests. The statistical method of this metric is similar to the statistical method of the AvgRt metric for all KVStore for Redis instances.

8.Parameter settings

KVStore for Redis allows you to customize certain instance parameters. This topic describes parameters and the common methods to modify them in the KVStore for Redis console.

Limits

- To ensure the stability of KVStore for Redis instances, only specific parameters can be set. The parameters that are not described in this topic cannot be set.
- After you submit the modifications for specific parameters, your instance is automatically restarted. The instance experiences transient connections that last for a few seconds during the restart. When you set an instance parameter in the KVStore for Redis console, pay attention to the **Restart and Take Effect** column corresponding to the parameter.

Parameters

Parameters

Parameter	Description
#no_loose_check-whitelist-always	Specifies whether to check that the client IP address is in a whitelist of the KVStore for Redis instance if password-free access over a virtual private cloud (VPC) is enabled. If you set this parameter to yes, the whitelist still takes effect for password-free access over a VPC. Default value: no. Valid values: • yes: The system checks whether a client IP address is in a whitelist. • no: The system does not check whether a client IP address is in a whitelist.
#no_loose_disabled-commands	Specifies the commands that you want to disable. The commands that can be disabled include **FLUSHALL**, **FLUSHDB**, **KEYS**, **HGETALL**, **EVAL**, **EVALSHA**, and **SCRIPT**. Separate multiple commands with commas (,).
#no_loose_ssl-enabled	Specifies whether to enable SSL encryption. Default value: no. Valid values: • yes: SSL encryption is enabled. • no: SSL encryption is disabled.
#no_loose_sentinel-enabled	Specifies whether to enable the Sentinel-compatible mode. Default value: no. Valid values: • yes: The Sentinel-compatible mode is enabled. • no: The Sentinel-compatible mode is disabled.

Parameter	Description
client-output-buffer-limit pubsub	Specifies output buffer limits of publisher and subscriber clients. The clients are disconnected when the specified limits are reached. Specify this parameter in the following format: `<hard limit> <soft limit> <soft seconds>` . • hard limit: disconnects a client if the output buffer of the client is greater than or equal to the hard limit value. The hard limit value is measured in bytes. • soft limit and soft seconds: disconnect a client if two conditions are met. One condition is that the output buffer of the client is greater than or equal to the soft limit value. The other condition is that the situation lasts for a period longer than or equal to the soft seconds value. The soft limit value is measured in bytes, and the soft seconds value is measured in seconds.
dynamic-hz	Specifies whether to enable a dynamic hz value. Default value: yes. Valid values: • yes: enables a dynamic hz value. • no: disables a dynamic hz value.
hash-max-ziplist-entries	Specifies the maximum number of key-value pairs stored in a hash. Ziplist encoding is used only if both of the following conditions are met: 1. The number of bytes of the key or value of each key-value pair stored in the hash is less than the value of the hash-max-ziplist-value parameter. 2. The number of key-value pairs stored in the hash is less than the value of the hash-max-ziplist-entries parameter.
hash-max-ziplist-value	Specifies the maximum size of the keys and values of key-value pairs stored in a hash. Ziplist encoding is used only if both of the following conditions are met: 1. The number of bytes of the key or value of each key-value pair stored in the hash is less than the value of the hash-max-ziplist-value parameter. 2. The number of key-value pairs stored in the hash is less than the value of the hash-max-ziplist-entries parameter.
hz	Specifies how frequently tasks are performed in the background. For example, you can specify how frequently tasks are performed to evict expired keys. Valid values: 1 to 500. The default value is 10, which indicates that each task is performed 10 times per second. A greater value results in higher CPU consumption but allows the system to delete expired keys more frequently and close timeout connections more precisely. We recommend that you specify a value less than or equal to 100.

Parameter	Description
lazyfree-lazy-eviction	Specifies whether to enable the eviction feature based on the lazyfree mechanism. Default value: no. Valid values: • yes: enables the eviction feature based on the lazyfree mechanism. • no: disables the eviction feature based on the lazyfree mechanism.
lazyfree-lazy-expire	Specifies whether to delete expired keys based on the lazyfree mechanism. Default value: yes. Valid values: • yes: deletes expired keys based on the lazyfree mechanism. • no: does not delete expired keys based on the lazyfree mechanism.
lazyfree-lazy-server-del	Specifies whether to asynchronously delete data based on the lazyfree mechanism for an implicit **DEL** command. Default value: yes. Valid values: • yes: asynchronously deletes data based on the lazyfree mechanism. • no: does not asynchronously delete data based on the lazyfree mechanism.
list-compress-depth	Specifies the number of nodes that are not compressed at both ends of a list. Default value: 0. Valid values: 0 to 65535. • 0: specifies that nodes in the list are not compressed. • 1: specifies that the first node from each end of the list is not compressed, but all nodes at both ends of the list between these two nodes are compressed. • 2: specifies that the first two nodes from each end of the list are not compressed, but all nodes at both ends of the list between these four nodes are compressed. • 3: specifies that the first three nodes from each end of the list are not compressed, but all nodes at both ends of the list between these six nodes are compressed. • You can specify other values based on the same rule.

list-max-ziplist-size	• Specifies the maximum size of each ziplist in a quicklist. A positive number indicates the maximum number of elements in each ziplist of a quicklist. For example, if you set this parameter to 5, each ziplist of a quicklist can contain a maximum of five elements.
	• A negative number indicates the maximum number of bytes in each ziplist of a quicklist. Default value: -2. Valid values: [-5, -1]. The following list describes the values:
	  ○ -5: specifies that each ziplist of a quicklist cannot exceed 64 KB (1 KB = 1,024 bytes).
	  ○ -4: specifies that each ziplist of a quicklist cannot exceed 32 KB.
	  ○ -3: specifies that each ziplist of a quicklist cannot exceed 16 KB.
	  ○ -2: specifies that each ziplist of a quicklist cannot exceed 8 KB.
	  ○ -1: specifies that each ziplist of a quicklist cannot exceed 4 KB.

Parameter	Description
maxmemory-policy	Specifies the policy used to evict keys if memory is exhausted. LRU means least recently used. LFU means least frequently used. LRU, LFU, and time-to-live (TTL) policies are implemented by using approximation and randomized algorithms. Valid values: • volatile-lru: evicts the approximated LRU keys among keys that have TTL configured. • allkeys-lru: evicts the approximated LRU keys. • volatile-lfu: evicts the approximated LFU keys among keys that have TTL configured. • allkeys-lfu: evicts the approximated LFU keys. • volatile-random: evicts random keys among keys that have TTL configured. • allkeys-random: evicts random keys. • volatile-ttl: evicts keys with the minimum TTL among keys that have TTL configured. • noeviction: does not evict keys, but returns an error for write operations.
notify-keyspace-events	The types of events of which the server can notify clients. The value of this parameter is a combination of the following characters, each of which specifies a type of events: • K: keyspace events, published with the __keyspace@<db>__ prefix. • E: key events, published with the __keyevent@<db>__ prefix. • g: generic events that are not related to specific commands, such as DEL, EXPIRE, and RENAME. • l: events of list commands. • s: events of set commands. • h: events of hash commands. • z: events of sorted set commands. • x: events of expired keys. An expiration event is triggered when an expired key is deleted. • e: events of evicted keys. An eviction event is triggered when a key is deleted due to the policy specified by the maxmemory-policy parameter. • A: the alias for g$lshzxe.

Parameter	Description
set-max-intset-entries	Specifies the maximum number of data entries in a set to support intset encoding. A set uses intset encoding when the following conditions are met: 1. All data entries in the set are strings. 2. The set contains only radix-10 integers in the range of 64-bit signed integers.
slowlog-log-slower-than	Specifies whether to log slow queries. • Negative value: does not log slow queries. • 0: logs all queries. • Positive value: logs queries of which the duration exceeds the specified value. Valid values: 0 to 10000000. Default value: 10000. Unit: microseconds.
slowlog-max-len	The maximum number of slow query logs that can be stored. Valid values: 100 to 10000. Default value: 1024.
stream-node-max-bytes	The maximum amount of memory in bytes that each macro node in a stream can consume. Valid values: 0 to 999999999999999. Unit: bytes. A value of 0 indicates that no limit exists.
stream-node-max-entries	The maximum number of entries that can be stored on each macro node in a stream. Valid values: 0 to 999999999999999. A value of 0 indicates that no limit exists.
timeout	Specifies a timeout period. The system closes a connection to a client if the client has been idle for the specified amount of time. Valid values: 0 to 100000. Unit: seconds. A value of 0 indicates that no timeout period is specified for connections.
zset-max-ziplist-entries	Specify the maximum number of key-value pairs stored in a sorted set. Ziplist encoding is used only if the following conditions are met: 1. The number of bytes of the key or value of each key-value pair stored in the sorted set is less than the value of the zset-max-ziplist-value parameter. 2. The number of key-value pairs stored in the sorted set is less than the value of the zset-max-ziplist-entries parameter.

Parameter	Description
zset-max-ziplist-value	Specify the maximum size of the keys and values of key-value pairs stored in a sorted set. Ziplist encoding is used only if the following conditions are met: 1. The number of bytes of the key or value of each key-value pair stored in the sorted set is less than the value of the zset-max-ziplist-value parameter. 2. The number of key-value pairs stored in the sorted set is less than the value of the zset-max-ziplist-entries parameter.
list-max-ziplist-entries	Specifies the maximum number of elements stored in a list. Ziplist encoding is used only if both of the following conditions are met: 1. The elements stored in the list are all smaller than the value of the list-max-ziplist-value parameter. The elements are measured in bytes. 2. The number of elements stored in the list is smaller than the value of the list-max-ziplist-entries parameter.
list-max-ziplist-value	Specifies the maximum size of the elements stored in a list. Ziplist encoding is used only if both of the following conditions are met: 1. The elements stored in the list are all smaller than the value of the list-max-ziplist-value parameter. The elements are measured in bytes. 2. The number of elements stored in the list is smaller than the value of the list-max-ziplist-entries parameter.
cluster_compat_enable	Specifies whether to enable support for the syntax of native Redis clusters. Default value: 1. Valid values: • 0: disables the support. • 1: enables the support.
script_check_enable	Specifies whether to check that the keys used in Lua scripts are mapped to the same slot. Default value: 1. Valid values: • 0: does not check whether the keys are mapped to the same slot. • 1: checks whether the keys are mapped to the same slot.

Note The maxclients parameter specifies the maximum number of concurrent connections to data nodes in KVStore for Redis. The default value is 10000 and cannot be modified.

Modify parameters in the KVStore for Redis console

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.

3. In the left-side navigation pane of the **Instance Information** page, click **System Parameters**.
4. Find the target parameter and click **Modify** in the **Action** column.
5. In the dialog box that appears, modify the parameter value and click **OK**.

9.Backup and recovery

9.1. Automatically back up data

An increasing number of applications use Redis for persistent storage. In this case, an automatic backup mechanism is required to back up data on a regular basis so that you can restore data if user errors occur. KVStore for Redis uses Redis database backup (RDB) snapshots to back up data on replica nodes. The backup process does not have negative impacts on the performance of your instance. You can configure a custom backup policy in the console.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Backup and Recovery**.
4. Click the **Backup Settings** tab.
5. Click **Edit** and specify Backup Cycle and Backup Time.
   - **Retention Days**: The number of days for which backups are retained. This parameter is set to seven days and cannot be changed.
   - **Backup Cycle**: You can select one or more days in a week. By default, one backup is created per day.
   - **Backup Time**: You can specify a period of time in hours within a day. We recommend that you back up data during off-peak hours.
6. Click **OK**.

9.2. Back up an instance

You can initiate a manual backup task in the console at any time.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Backup and Recovery**.
4. In the upper-right corner of the page, click **Create Backup**.
5. Click **OK**.

> **Note** On the **Data Backup** tab, you can select a time range to query existing backups. Backups are retained for seven days.

9.3. Download backup files

To archive backup files for a long period, you can copy the URLs in the console and download the database backup files to an on-premises machine.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Backup and Recovery**.
4. Find the backup file that you want to download and click **Download** in the **Actions** column.

> **Note** For a cluster instance, you must download the backup file for each data shard at the same point in time to ensure data consistency.

9.4. Restore data

KVStore for Redis allows you to restore data from a specified backup set to the current KVStore for Redis instance.

Prerequisites

The instance must be a master-replica or cluster instance.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Backup and Recovery**.
4. Perform one of the following operations based on the architecture of your KVStore for Redis instance:
   - Master-replica instance: Find the backup set that you want to restore and click **Restore Data** in the **Actions** column.
   - Cluster instance: Select the backup sets of all data shards that were generated at the same point in time and click **Restore Data** in the upper-right corner.

   > **Warning** Risks may occur when you restore data. Proceed with caution. Verify the data that you want to restore before you restore the data.

5. In the message that appears, read the content and click **Continue**.

   You can also restore backup data by cloning an instance. For more information, see Clone an instance.

9.5. Clone an instance

KVStore for Redis allows you to create an instance from a specified backup set. The data in the new instance is the same as the data in the backup set. This feature can be applied in scenarios such as data recovery, quick workload deployment, and data verification.

Prerequisites

The instance must be a master-replica or cluster instance.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, click **Backup and Recovery**.
4. Find the backup set that you want to restore and click **Clone Instance** in the **Actions** column.

   > **Note** For a cluster instance, you must select the backup file for each data shard at the same point in time.

5. In the message that appears, click **OK**.
6. On the Restore Instance page, configure the parameters and click **Submit**.

   > **Note** For more information about the configurations of the new instance, see Create an instance.

10.CloudDBA

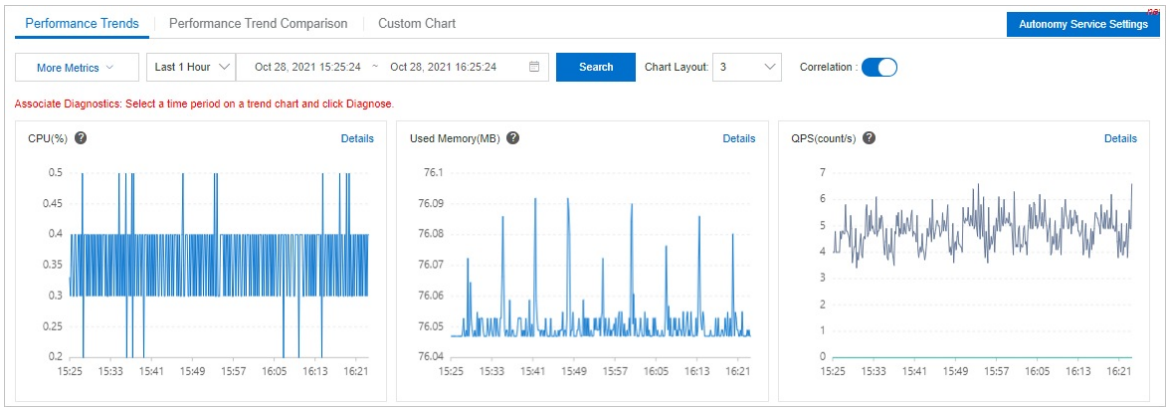
10.1. Performance trends

CloudDBA provides the performance trends feature that allows you to monitor the basic performance of a KVStore for Redis instance and the operational trends within a specified period of time. The performance trends include the CPU utilization, memory usage, queries per second (QPS), total connections, response time, data transfer, and key hit ratio.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Performance Trends**.
4. You can use the following methods to view performance trends:

> **Note** If the KVStore for Redis instance uses a cluster architecture, the Performance Trends page displays the information about the nodes. The performance data during the last 1 hour is displayed. If you click the node ID, you can view the details of a specified node.



- **Performance trends**

  On the **Performance Trends** tab, specify a time range and click **Search**.

  > **Note**
  >
  > - By default, **Correlation** is enabled. If you move the pointer over the CPU chart to view the CPU metric of the KVStore for Redis instance at 09:00, other charts also display other metrics of the instance at 09:00.
  > - To view the definition of the performance metric and the performance trend, click **Definition?** and **Details** in the upper-right corner of the chart.

- **Performance trend comparison**

  To compare the performance trends within two periods of time, click the **Performance Trend Comparison** tab, specify two periods of time, select more metrics, and then click **Search**.

- **Custom chart**

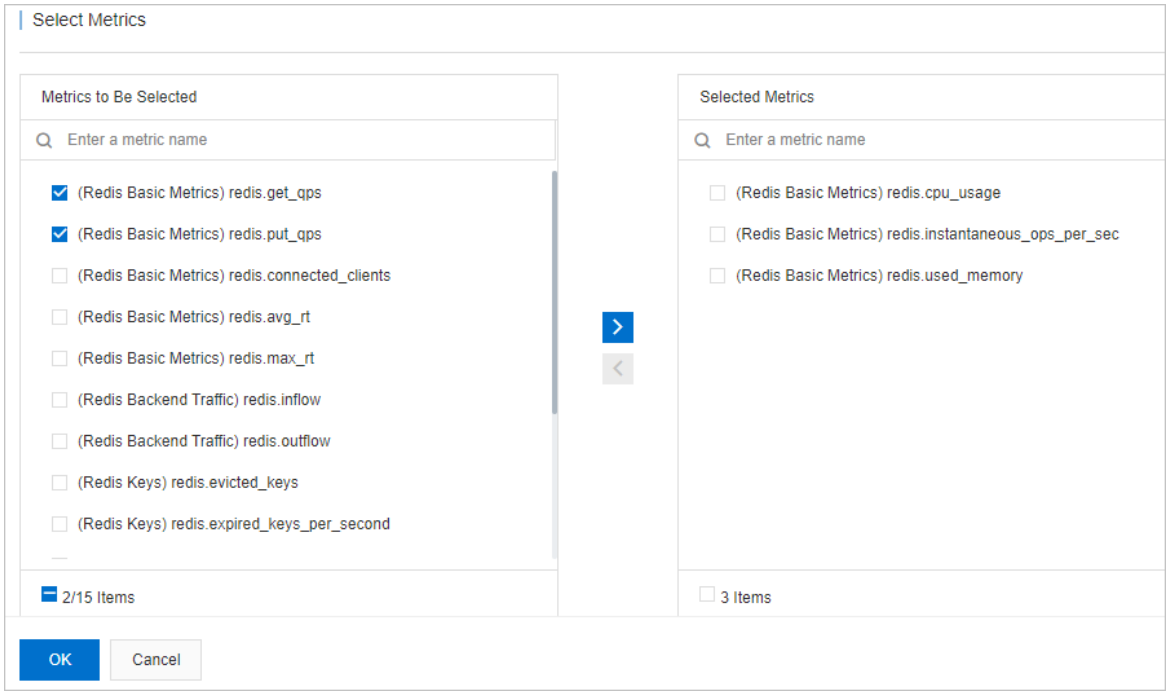
The preceding two methods display the basic metrics of a KVStore for Redis instance. If you want to display only basic metrics, you can configure custom performance trend charts. For more information, see Add a performance trend chart.

10.2. Add a performance trend chart

The default performance trends tab displays the basic performance metrics of a KVStore for Redis instance. You can add a chart that contains only specified performance metrics to analyze the performance trends of your instances. This topic describes how to add a performance trend chart to a dashboard for KVStore for Redis instances.

Procedure

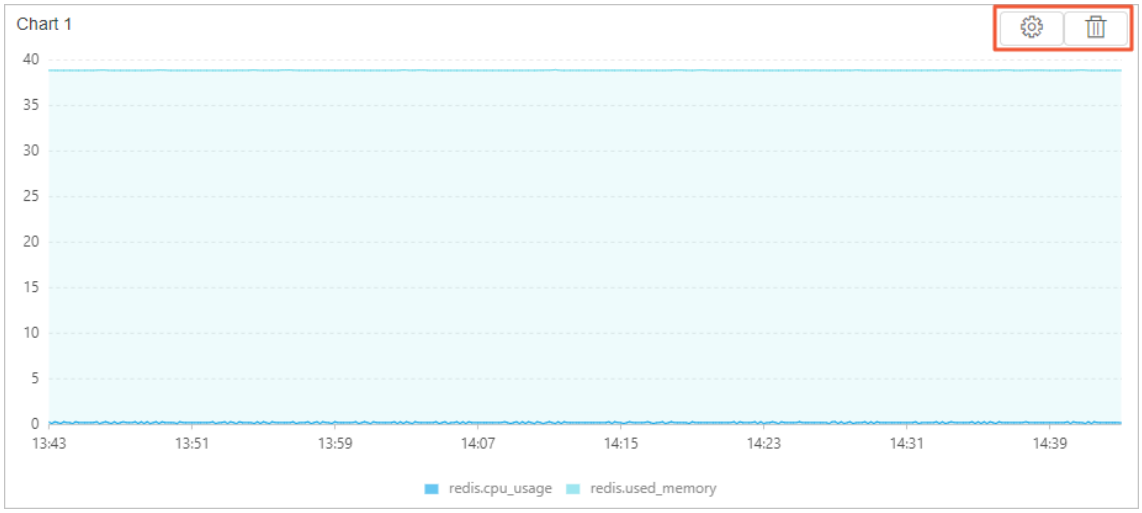
1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Performance Trends**.
4. Click the **Custom Chart** tab.
5. Click **Add Monitoring Dashboard**. In the Create Monitoring Dashboard dialog box, enter a dashboard name and click **OK**.
6. Click **+ Add Chart** or **Add Monitoring Chart**. Select the metrics that you want to add and click **OK**.



7. (Optional)You can view, modify, and delete monitoring dashboards.
   - View a monitoring dashboard

     Select the monitoring dashboard, specify a time range, and then click **Search**.
   - Modify a monitoring dashboard

     Click the following icons to modify or delete a chart in the monitoring dashboard.



- Delete a monitoring dashboard

  Choose **Operate Dashboard > Delete Monitoring Dashboard**.

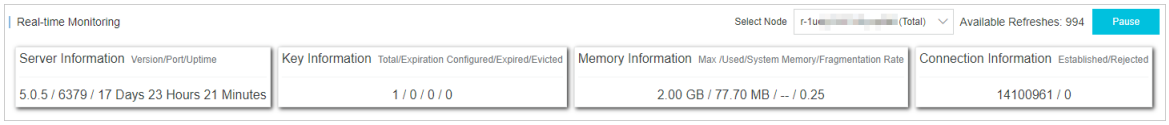
10.3. View performance metrics in real time

CloudDBA allows you to view the performance metrics of KVStore for Redis instances in real time. The performance metrics include information about CPU utilization, memory usage, queries per second (QPS), network traffic, servers, keys, clients, and connections.

Procedure

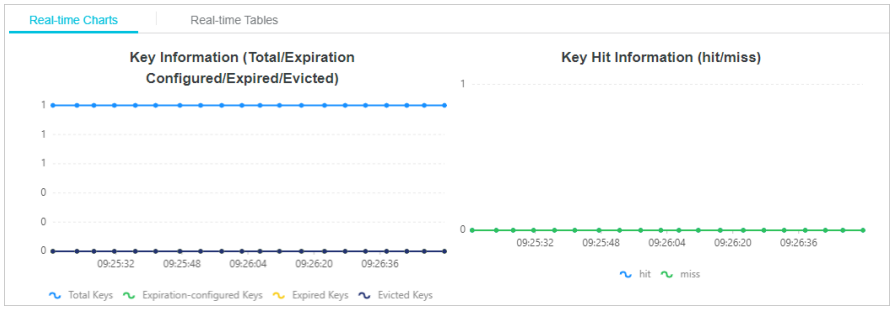
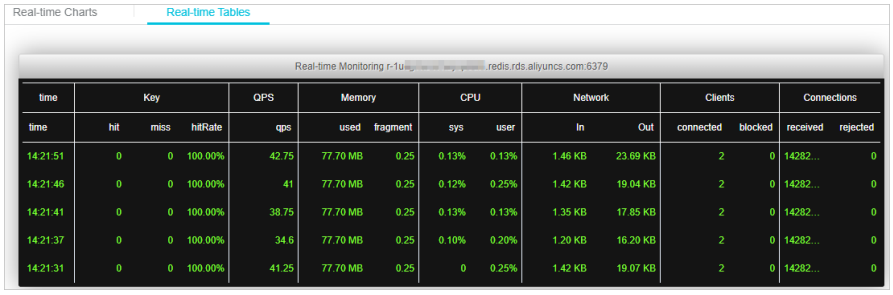
1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Real-time Performance**.
4. Select a view based on your business requirements.

   In the upper part of the page, performance metrics are displayed in real time. The metrics include information about the server, keys, memory, clients, and connections. The details about the performance metrics are displayed in **Real-time Charts** and **Real-time Tables**.



> **Note** The metrics are automatically updated every 5 minutes. Therefore, you can view real-time changes in performance. The remaining updating times are displayed in the upper-right corner. To stop updating the performance metrics, you can click **Pause**.

View	Description

View	Description
Real-time Charts	Displays the real-time performance metrics of an instance in curve charts, such as key information, key hit information, key hit ratio, CPU utilization, memory information, QPS, and network traffic. 
Real-time Tables	Displays the information about keys, QPS, memory usage, CPU utilization, network traffic, clients, and connections. The table displays up to 999 records. A new record is added to the table every 5 seconds. 

time	Key			QPS	Memory		CPU		Network		Clients		Connections	
time	hit	miss	hitRate	qps	used	fragment	sys	user	In	Out	connected	blocked	received	rejected
14:21:51	0	0	100.00%	42.75	77.70 MB	0.25	0.13%	0.13%	1.46 KB	23.69 KB	2	0	14282...	0
14:21:46	0	0	100.00%	41	77.70 MB	0.25	0.12%	0.25%	1.42 KB	19.04 KB	2	0	14282...	0
14:21:41	0	0	100.00%	38.75	77.70 MB	0.25	0.13%	0.13%	1.35 KB	17.85 KB	2	0	14282...	0
14:21:37	0	0	100.00%	34.6	77.70 MB	0.25	0.10%	0.20%	1.20 KB	16.20 KB	2	0	14282...	0
14:21:31	0	0	100.00%	41.25	77.70 MB	0.25	0	0.25%	1.42 KB	19.07 KB	2	0	14282...	0

10.4. Instance sessions

Instance sessions allow you to view the statistics for sessions between a KVStore for Redis instance and clients in real time. These statistics include clients, commands that were run, and connection durations. You can also close abnormal sessions based on your business requirements.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Instance Sessions**.
4. Perform the following operations to manage your instance sessions:
   - View sessions: By default, the details of all sessions are displayed. You can move the pointer over a specific parameter name to view its information.

> **Note**
>
> - You can enter keywords in the search box to filter sessions.
> - To refresh instance sessions, click **Refresh** in the upper-left corner or enable **Auto Refresh**. If you enable auto-refresh, the system automatically refreshes the page every 30 seconds.

- Close sessions: You can close the specified one or more sessions at a time or close all sessions. To close one or more sessions at a time, select the specified session or press the Shift key and select multiple sessions, and then click **Kill Selected** in the upper-right corner. To close all sessions, click **Kill All**.

> **Warning** To prevent unexpected consequences, we recommend that you do not close system-level sessions.

- View session statistics: Session statistics record the total number of clients, active clients, and source IP addresses involved in instance sessions.

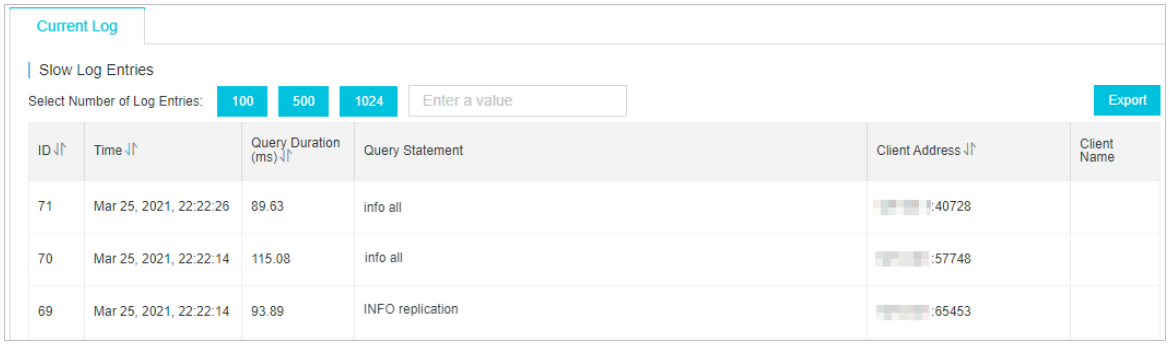
> **Note** In the **Statistics by Source** table, click the icon next to the source IP address to modify the source alias. In the Total Sessions column, click a value to view the details of source IP addresses.

10.5. Slow queries

Slow queries reduce the stability of KVStore for Redis instances. To monitor and analyze slow queries, you can view the details about slow query logs in CloudDBA.

Procedure

1. Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Slow Queries**.
4. Query the details about the slow query logs.



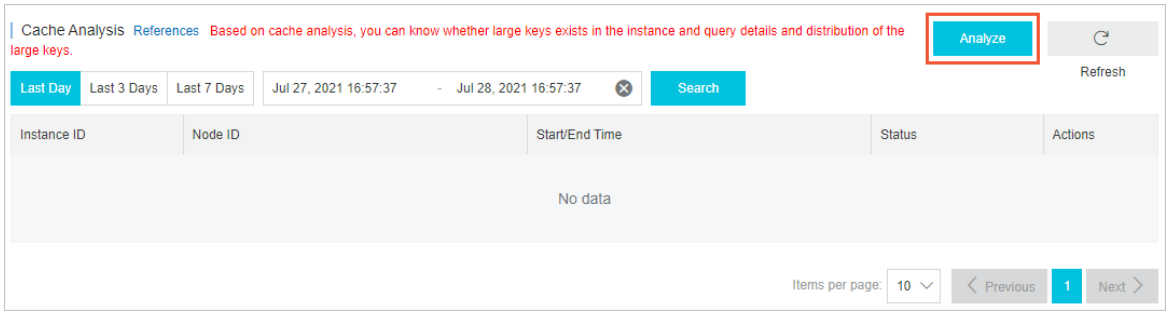
> **Note** You can select the number of log entries to be displayed or enter keywords in the search box to filter log entries.

10.6. Cache analysis

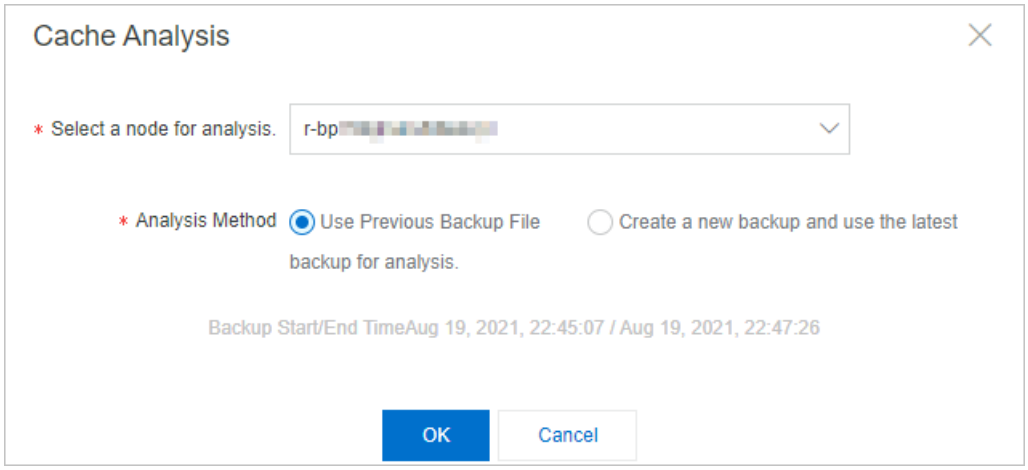
The cache analysis feature allows you to analyze backup files of KVStore for Redis instances to identify large keys. This feature allows you to view the information about an instance, such as the memory usage, distribution, and expiration time of keys in the instance. You can optimize the instance based on the analysis results. This feature helps you resolve issues such as insufficient memory and performance degradation that are caused by skewed distribution of keys.

Procedure

1. Log on to the KVStore for Redis console. For more information, see Log on to the KVStore for Redis console.
2. On the **Instance List** page, click the ID of the instance.
3. In the left-side navigation pane, choose **CloudDBA > Offline Key Analysis**.
   By default, the analysis results of the last day are displayed on the **Cache Analysis** tab. You can specify another time range based on your requirements.
4. On the **Cache Analysis** page, click **Analyze**.



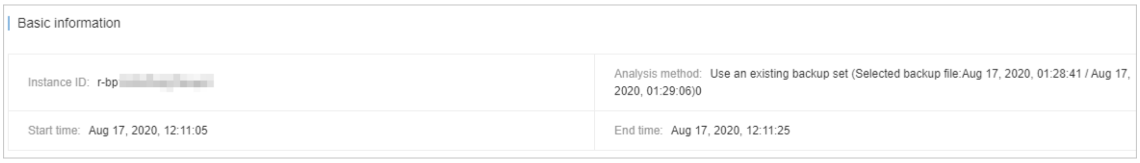
5. In the dialog box that appears, specify the node and analysis method.



Parameter	Description
Select a node for analysis	The ID of the node on which you want to perform cache analysis. **Note** You can select an instance or a node for analysis. If you select an instance that contains more than eight nodes, the system analyzes only the top eight nodes that have the highest memory usage.

Parameter	Description
Analysis Method	The method that you want to use to analyze the cache. Valid values: **Use Previous Backup File** and **Create a new backup and use the latest backup for analysis**.

6. Click **OK**.
   The system performs cache analysis and displays the analysis state. You can click **Refresh** to update the analysis state.
7. Find the completed analysis task and click **Details** in the **Actions** column to view the detailed results.
   - **Basic information**: displays basic information of an instance such as the basic instance attributes and the cache analysis method.

     Basic information

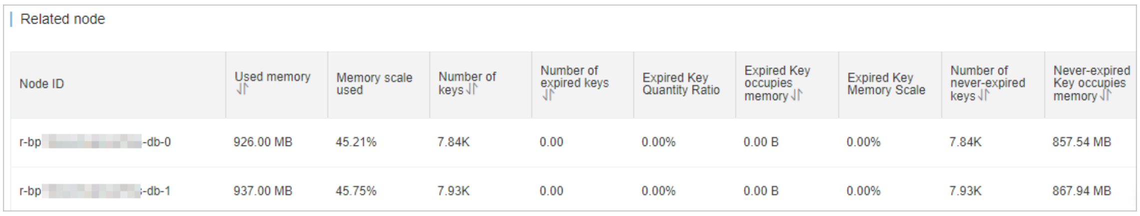
     Instance ID: r-bp[redacted]

     Analysis method: Use an existing backup set (Selected backup file:Aug 17, 2020, 01:28:41 / Aug 17, 2020, 01:29:06)0

     Start time: Aug 17, 2020, 12:11:05

     End time: Aug 17, 2020, 12:11:25

   - **Relevant Nodes**: displays the memory usage and key statistics on each node of the instance.

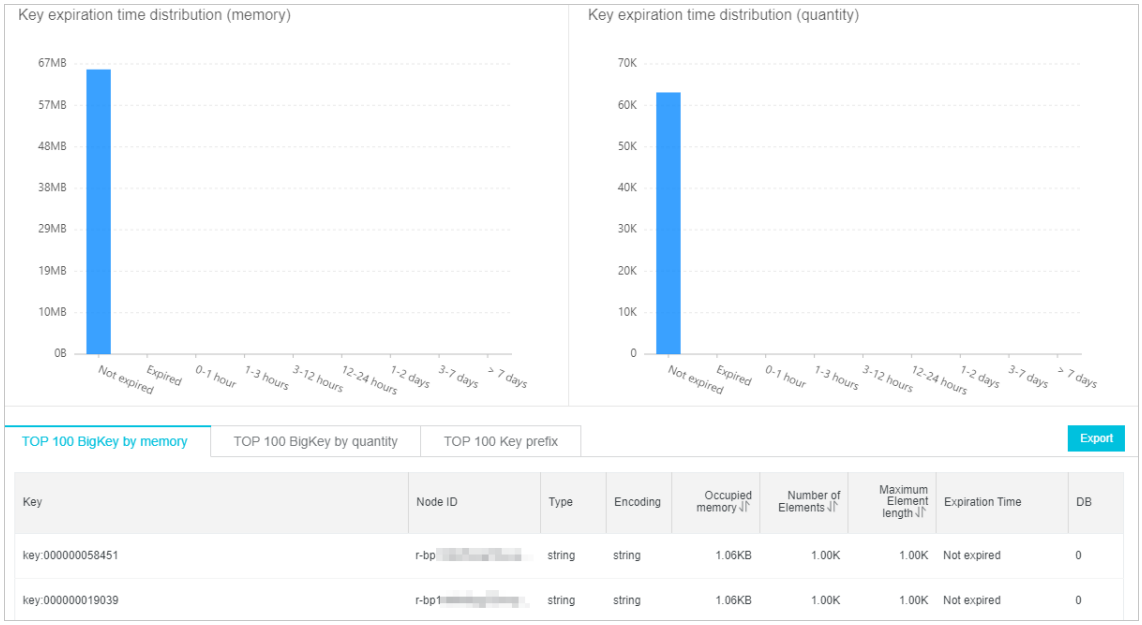
     

     Related node

Node ID	Used memory	Memory scale used	Number of keys	Number of expired keys	Expired Key Quantity Ratio	Expired Key occupies memory	Expired Key Memory Scale	Number of never-expired keys	Never-expired Key occupies memory
r-bp[redacted]-db-0	926.00 MB	45.21%	7.84K	0.00	0.00%	0.00 B	0.00%	7.84K	857.54 MB
r-bp[redacted]-db-1	937.00 MB	45.75%	7.93K	0.00	0.00%	0.00 B	0.00%	7.93K	867.94 MB

     > **Note** If you select a cluster instance from the Select a node for analysis drop-down list, you can view node information in the **Relevant Nodes** section and select a node in the Details section of the **Details** page.

   - **Details**: displays the details about an instance or a node, such as the memory usage and distribution of keys, memory usage and distribution of elements in keys, distribution of key expiration time, and ranking of large keys.


Key expiration time distribution (memory)
67MB
57MB
48MB
38MB
29MB
19MB
10MB
0B
Not expired
Expired
0-1 hour
1-3 hours
3-12 hours
12-24 hours
1-2 days
3-7 days
> 7 days
Key expiration time distribution (quantity)
70K
60K
50K
40K
30K
20K
10K
0
Not expired
Expired
0-1 hour
1-3 hours
3-12 hours
12-24 hours
1-2 days
3-7 days
> 7 days
TOP 100 BigKey by memory
TOP 100 BigKey by quantity
TOP 100 Key prefix
Export
Key
Node ID
Type
Encoding
Occupied memory
Number of Elements
Maximum Element length
Expiration Time
DB
key:000000058451
r-bp
string
string
1.06KB
1.00K
1.00K
Not expired
0
key:000000019039
r-bp1
string
string
1.06KB
1.00K
1.00K
Not expired
0