

Alibaba Cloud Table Store

Utilities

Issue: 20190606

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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand slave}</code>

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 TablestoreCli.....	1

1 TablestoreCli

Quick start

TablestoreCli is a command-line interface (CLI) tool used to manage data in Table Store. This tool provides simple and clear commands and supports Windows, Linux, and Mac operating systems.

Downloads

- Running environment
- TablestoreCli can run in Linux, Mac, and Windows 10 operating systems.
- Download the following binary versions:
 - [aliyun-tablestore-cli-linux-amd64-2018-11-11.zip](#)
 - [aliyun-tablestore-cli-macos-amd64-2018-11-11.zip](#)
 - [aliyun-tablestore-cli-windows-amd64-2018-11-11.zip](#), only applicable to Windows 10.

Get started

```
# 1 Start Tablestore Cli .
$./ ts

# 2 Configure the corresponding connection information
and modify the fields according to actual conditions .
ts > config -- endpoint https :// myinstance . cn - hangzhou .
ots . aliyuncs . com -- instance myinstance -- id test_acces
sid -- key test_acces skey

# 3 View all tables on the current instance .
ts > lt

# 4 Create a new table .
ts > create - t mysampletable -- pk '[" C ":" uid "," t ":"
string "], {" c ":" pid "," t ":" integer "}]' -- ttl 864000 --
version 1
'''
Create a table : mysampletable ,
and specify two primary keys : uid ( string ) and pid (
integer ).
Set the data declaratio n cycle to 864 , 000 seconds
or 10 days and the number of versions to 1 .
'''

# 5 View table informatio n .
ts > desc - t mysampletable

# 6 Use this table .
ts > use - t mysampletable

# 7 Insert data into the table .
```

```

ts > put -- pk '[" 86 ", 6771 ]' -- attr '[" c ":" name ", " v ":" redchen1 "], {" c ":" country ", " v ":" china "}]'
ts > put -- pk '[" 86 ", 6772 ]' -- attr '[" c ":" name ", " v ":" redchen2 "], {" c ":" country ", " v ":" china "}]'
ts > put -- pk '[" 86 ", 6773 ]' -- attr '[" c ":" name ", " v ":" redchen3 "], {" c ":" country ", " v ":" china "}]'
ts > put -- pk '[" 86 ", 6774 ]' -- attr '[" c ":" name ", " v ":" redchen4 "], {" c ":" country ", " v ":" china "}]'

# 8 Read data from the table .
ts > get -- pk '[" 86 ", 6771 ]'
ts > get -- pk '[" 86 ", 6772 ]'
ts > get -- pk '[" 86 ", 6773 ]'
ts > get -- pk '[" 86 ", 6774 ]'

# 9 Delete the table .
ts > drop - t mysampletable - y

```

View supported options

You can use the help option to view all options that TablestoreCli supports.

```

$./ ts help

Commands :
  alter          Alter table
  clear          Clear the screen
  config         Config the TableStore access
  informatio n
  create         Creat a new table
  del            Delete the specify row from
  TableStore
  desc           Show table meta
  drop           Drop the table
  exit           Exit the program
  export         Export the data of table to
disk from TableStore , not support multi version
  get            Get specify row from TableStore
  help           Display help
  import         Load the data to TableStore , not
support multi version
  list           List all tables
  points         Logically divide the data of
the full table into several shards close to the
specified size
  press_chec k   Check data for press
  press_inpu t   Input data for press
  put            Insert a row to TableStore
  quit           Quit the program
  update         Insert a row to TableStore
  use            Select table

```

Initialization

```

Sample :
  config -- endpoint https:// myinstance . cn - hangzhou . ots
. aliyuncs . com -- instance myinstance -- id test_acces sid
-- key test_acces skey
Flags :
  -- endpoint Endpoint      TableStore Endpoint
  -- id AccessKeyI d        User AccessKeyI d

```

```
-- instance Instance TableStore Instance
-- key AccessKeyS ecet User AccessKeyS ecet
```

Table-level operations

Create a table

```
Sample :
# Create a table that contains the first primary
key of string type and the second primary key of
integer type .
# Specify the second primary key as an auto -
increment column of integer type .
# Set Time To Live ( TTL ) and Max Versions .
create - t mytable -- pk '[" c ":" uid ", " t ":" string
", {" c ":" pid ", " t ":" integer "}]'
create - t mytable -- pk '[" c ":" uid ", " t ":" string
", {" c ":" pid ", " t ":" integer ", " opt ":" auto "}]'
create - t mytable -- pk '[" C ":" uid ", " t ":" string ",
{ " c ":" pid ", " t ":" integer "}]' -- ttl 864000 -- version 1
Flags :
- i , -- input string Create table json
- p , -- pk string Primary Key
-- read_cu string read capacity unit ( default
" 0 ")
- t , -- table string table name
-- ttl string Time to live , unit is
second , default ( - 1 ), means never timeout . ( default "- 1
")
-- version string Max version , default ( 1 ) (
default " 1 ")
-- write_cu string write capacity unit ( default
" 0 ")
```

List table names

```
$/ ts list
List all table name from instance
```

Update the table

```
Sample :
alter - t mytable -- ttl 86400 -- version 1 -- read_cu
0 -- write_cu 0
Flags :
-- read_cu string read capacity unit
- t , -- table string table name
-- ttl string Time to live , unit is
second
-- version string Max version
-- write_cu string write capacity unit
```

Query the description of the table

```
Sample :
desc - t tablename
Flags :
- o , -- output string Output file path
```

```
- t , -- table string Table name
```

Delete the table

```
Sample :
    drop - t tablename
Flags :
    - t , -- table string table name
    - y , -- yes confirm yes
```

Select a table for data operations

```
Sample :
    use - t tablename
Flags :
    - t , -- table string table name
```

Single-row operations

Insert a row into the table

```
Sample :
    # Insert a row into the table . This row contains
    two primary keys and two attribute columns . The first
    primary key is " 86 ", and the second primary key
    is 6771 . The first attribute column is named as "
    name " and the attribute value is " redchen ".
    # Select a check logic for inserting the row : 1
    . Ignore the check . 2 . Allow inserting the row that
    contains the same primary key . 3 . Allow inserting
    the row that does not contain the same primary key
    .
    # Allow inserting the specified timestamp .
    # Allow inserting the row that contains an auto -
    increment column as the primary key .
    # Read and insert data from a file .
    put -- pk '[' 86 ", 6771 ]' -- attr '['{" c ":" name ", " v
    ":" redchen "}, {" c ":" country ", " v ":" china "}]'
    put -- pk '[' 86 ", 6771 ]' -- attr '['{" c ":" name ", " v
    ":" redchen "}, {" c ":" country ", " v ":" china "}]' -- condition
    ignore / exist / not_exist
    put -- pk '[' 86 ", 6771 ]' -- attr '['{" c ":" name ",
    " v ":" redchen "}, {" c ":" country ", " v ":" china ", " ts ":"
    1532779853 4 }]'
    put -- pk '[' 86 ", null ]' -- attr '['{" c ":" name ", " v
    ":" redchen "}, {" c ":" country ", " v ":" china "}]'
    put - i / tmp / input
Flags :
    -- attr string Attribute column value
    -- condition string input condition , ignore / exist /
    not_exist ( default " ignore ")
    - i , -- input string input file
    -- pk string PrimaryKey value
```

Read a row

```
Sample :
    get -- pk '[' 86 ", 6771 ]'
Flags :
    - o , -- output string Output file path
```

-- pk	string	Input	primary	key	column	value
-------	--------	-------	---------	-----	--------	-------

Update a row

```
Sample :
update -- pk '[' 86 ', 6771 ]' -- attr '[" c ":" name ", " v
":" redchen "], {" c ":" country ", " v ":" china "}]'
update -- pk '[' 86 ', 6771 ]' -- attr '[" c ":" name ", " v
":" redchen "], {" c ":" country ", " v ":" china "}]' -- condition
ignore / exist / not_exist
update - i / tmp / input
Flags :
-- attr string Attribute column value
-- condition string input condition , ignore / exist /
not_exist ( default " ignore ")
- i , -- input string input file
-- pk string PrimaryKey value
```

Delete a row

```
Sample :
del -- pk '[' 86 ', 6771 ]'
Flags :
-- pk string PrimaryKey value
```

Simple stress testing

Enable stress testing

```
Sample :
# Create a table . In the table , the first
primary key is " Partition ", and the second primary
key is " Row number ".
# Use this table to insert 10 rows into
Partition " redchen ".
create - t mytable -- pk '[" c ":" uid ", " t ":"
string "], {" c ":" pid ", " t ":" integer "}]'
use - t mytable
press_inpu t -- part redchen -- count 10
Flags :
-- begin string Begin index , default 0
-- count string Input count , length of single
row is 1KB
-- part string Partition key , default ' redchen ' (
default " redchen ")
```

Check the stress testing status

```
Sample :
create - t mytable -- pk '[" c ":" uid ", " t ":"
string "], {" c ":" pid ", " t ":" integer "}]'
use - t mytable
press_chec k -- part redchen -- begin 0 -- count
1000
Flags :
-- begin string Begin index , default 0
-- count string Check count
-- part string Partition key
```

```
- y , -- yes                show    cost    time
```

Data backup

Export data from a table to a disk file

```
Sample :
# Export all data from the current table to a
disk file .
# Export data from the specified column .
export - o / tmp / mydata
export - o / tmp / mydata - c attr1 , attr2 , attr3
Flags :
- c , -- columns string      Column names
- o , -- output  string      Output file path
```

Import data from a disk file to a table

```
Sample :
# Read data from a disk file and export the
data to a table .
# Ignore the timestamp check and use the current
time .
import - i / tmp / mydata
import - i / tmp / mydata -- ignore_ts
,,,
file format :
{" PK ":{ " Values ":[ " redchen ", 0 ]}," Attr ":{ " Values ":[{"
C ":" country "," V ":" china0 "},{ " C ":" name "," V ":" redchen0
"}]}}
{" PK ":{ " Values ":[ " redchen ", 1 ]}," Attr ":{ " Values ":[{"
C ":" country "," V ":" china1 "},{ " C ":" name "," V ":" redchen1
"}]}}
,,,
Flags :
-- ignore_ts          ignore timestamp
- i , -- input string Input file name
```