ALIBABA CLOUD

Alibaba Cloud

Tablestore

API Reference

Document Version: 20220622

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.List of operations by function

1.1. Operation summary

This topic describes how to call Tablestore API operations and lists the API operations available for use in Tablestore.

Description

You can call Tablestore API operations only by using Tablestore SDKs for various programming languages. You cannot use HTTP methods to call Tablestore API operations.

List of operations by function

Category	API operation
Table operations	• CreateTable • ListTable • DeleteTable • UpdateTable • DescribeTable • ComputeSplitPointsBySize
Single-row operations	• GetRow • PutRow • UpdateRow • DeleteRow
Multi-row operations	• GetRange • BatchGetRow • BatchWriteRow
Stream operations	• ListStream • DescribeStream • GetShardIterator • GetStreamRecord
Index operations	• CreateIndex • DropIndex

Category	API operation
Operations that are related to the TimeSeries model	• CreateTimeseriesTable • ListTimeseriesTable • DescribeTimeseriesTable • UpdateTimeseriesTable • DeleteTimeseriesTable • PutTimeseriesData • GetTimeseriesData • QueryTimeseriesMeta • UpdateTimeseriesMeta • DeleteTimeseriesMeta

1.2. GetRow

Reads a single row of data based on the specified primary key.

Request syntax

```
message GetRowRequest {
    required string table_name = 1;
    required bytes primary_key = 2; // The data is encoded as binary data in the PlainBuffe
r format.
    repeated string columns_to_get = 3; // If you do not specify this parameter, all column
s are read.
    optional TimeRange time_range = 4;
    optional int32 max_versions = 5;
    optional bytes filter = 7;
    optional string start_column = 8;
    optional string end_column = 9;
    optional bytes token = 10;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table whose data you want to read. For more information, see CU consumption.
primary_key	bytes	Yes	All primary key columns of the row, including the names and values of the primary key columns. The primary key columns are encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.

Parameter	Type	Required	Description
columns_to_get	string	No	The names of the columns that you want to return. If you do not specify this parameter, all columns of the row are returned. The value of this parameter can contain up to 128 strings. If the specified column does not exist, data of the specified column is not returned. If you specify duplicate column names, the response includes this column only once.
time_range	TimeRange	No (You must specify at least one of time_range and max_versions.)	The timestamp range in which you want to read multiple versions of data. The unit is millisecond.The minimum value of the timestamp is 0, and the maximum value is INT64.MAX. If you want to query data within a specified time range, specify start_time and end_time. If you want to query data that contains a specified timestamp, specify specific_time. If the value of time_range is [100, 200), only the columns whose timestamp is within the range of [100, 200) are returned.
max_versions	int32	No (You must specify at least one of time_range and max_versions.)	The maximum number of versions of data that you want to return. If the value of max_versions is 2, up to two versions of data is returned for each column.
filter	bytes	No	The expression of the filter condition. The expression of the filter condition is serialized as binary data by using Protobuf. For more information, see Filter.
start_column	string	No	The column from which the read operation starts in the row. This parameter is used to read wide columns. The columns are sorted based on their names in alphabetical order. The response contains the specified start column. If a table contains columns a, b, and c, and the value of start_column is b, the read operation starts from column b, and columns b and c are returned.
end_column	string	No	The response does not contain the specified end column. The columns are sorted based on their names in alphabetical order. Example: If a table contains columns a, b, and c, and the value of end_column is b, the read operation ends at column b, and only column a is returned.

Parameter	Type	Required	Description
token	bytes	No	The position at which the next read operation starts in the wide column. This parameter is unavailable.

Response syntax

```
message GetRowResponse {
    required ConsumedCapacity consumed = 1;
    required bytes row = 2; // The data is encoded as binary data in the PlainBuffer format
.
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) consumed by this request.
row	bytes	The data that is read from the row. If the requested row does not exist, no data is returned. The returned data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.

Use Tablestore SDKs

You can use the following Tablestore SDKs to read a row of data:

- Tablestore SDK for Java: GetRow
- Tablestore SDK for Go: GetRow
- Tablestore SDK for Python: GetRow
- Tablestore SDK for Node.js: GetRow
- Tablestore SDK for .NET: GetRow
- Tablestore SDK for PHP: GetRow

CU consumption

- If the requested row does not exist, one read CU is consumed.
- If the requested row exists, the number of consumed read CUs is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = (Size of data in all primary key columns of the row + Size of data in the attribute columns that are read)/4 KB. For more information about how to calculate the data size, see Billing overview.
- If the request times out and the results are undefined, CUs may or may not be consumed.
- If an HTTP status code 5xx is returned, which indicates that an internal error occurred, the operation does not consume CUs. If other errors are returned, one read CU is consumed.

1.3. PutRow

Writes data to a row.

> **Note**
>
> - If the specified row does not exist, a new row is added. If the specified row exists, the row is overwritten.
> - If the response to the request does not contain errors, the request is successful.

Request syntax

```
message PutRowRequest {
    required string table_name = 1;
    required bytes row = 2; // The data is encoded as binary data in the PlainBuffer format
.
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table to which you want to write data.
row	bytes	Yes	The data that you want to write to the row, including primary key columns and attribute columns. The data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.
condition	Condition	Yes	Specifies whether to check the existence of the row before data is written. Default value: IGNORE. Valid values: • IGNORE: The row existence is not checked. • EXPECT_EXIST: The row is expected to exist. • EXPECT_NOT_EXIST: The row is expected not to exist.
return_content	ReturnContent	No	The type of data to return. Only the primary key can be returned. In most cases, this parameter is used by the auto-increment primary key column feature.

Response syntax

```
message PutRowResponse {
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) consumed by the operation. For more information about CU consumption, see CU consumption.
row	bytes	The data that is returned when return_content is specified. If return_content is not specified or no return value exists, NULL is returned. The returned data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.

Use Tablestore SDKs

You can use the following Tablestore SDKs to write a row of data:

- Tablestore SDK for Java: PutRow
- Tablestore SDK for Go: PutRow
- Tablestore SDK for Python: PutRow
- Tablestore SDK for Node.js: PutRow
- Tablestore SDK for .NET: PutRow
- Tablestore SDK for PHP: PutRow

CU consumption

- If the row to which you want to write data does not exist, the number of consumed CUs varies based on the value that you specified for the condition parameter.
  - If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in the attribute columns to which you want to write data)/4 KB.
  - If the value of the condition parameter is EXPECT_NOT_EXIST, both write and read CUs are consumed. The number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in the attribute columns to which you want to write data)/4 KB. The number of consumed read CUs is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = Size of data in all primary key columns of the row/4 KB.
  - If the value of the condition parameter is EXPECT_EXIST, data fails to be written to the row. One write CU and one read CU are consumed.
- If the row to which you want to write data exists, the number of consumed CUs varies based on the value that you specified for the condition parameter.
  - If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in the attribute columns to which you want to write data)/4 KB.

- If the value of the condition parameter is EXPECT_EXIST, both write and read CUs are consumed. The number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in the attribute columns to which you want to write data)/4 KB. The number of consumed read CUs is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = Size of data in all primary key columns of the row/4 KB.
- If the value of the condition parameter is EXPECT_NOT_EXIST, data fails to be written to the row. One write CU and one read CU are consumed.

For more information about how to calculate the data size, see Billing overview.

- When you use Conditional Update, the number of consumed CUs is calculated by using the preceding formulas if the operation succeeds. If the operation fails, one write CU and one read CU are consumed.
- If the request times out and the results are undefined, CUs may or may not be consumed.
- If an HTTP status code 5xx is returned, which indicates that an internal error occurred, the operation does not consume CUs. If other errors are returned, one write CU is consumed.

1.4. UpdateRow

Updates the data of a row.

Note If the specified row does not exist, a new row is added. If the specified row exists, the values of the specified columns are added, modified, or deleted based on the request.

Request syntax

```
message UpdateRowRequest {
    required string table_name = 1;
    required bytes row_change = 2;
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table whose data you want to update.

Parameter	Type	Required	Description
row_change	bytes	Yes	Specifies the data that you use to update the row, including primary key columns and attribute columns. The data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer. All attribute columns whose data you want to update for this row. Tablestore adds, modifies, or deletes the values in the specified attribute columns based on the content of UpdateType in row_change. The columns that exist in this row but are not specified by row_change are not affected. Valid values of UpdateType: • PUT: The value must be a valid attribute column value. If the specified column does not exist, a new column is added. If the specified column exists, the existing column is overwritten. • DELETE: The value must be empty, and a timestamp must be specified. This update type specifies that data of the specified version is deleted from the column. • DELETE_ALL: The value and the timestamp must be empty. This update type specifies that data of all versions in the column are deleted. Note A row exists even if all attribute columns in the row are deleted. To delete a row, call the DeleteRow operation.
condition	Condition	Yes	Specifies whether to check the row existence before data is updated. Default value: IGNORE. Valid values: • IGNORE: The row existence is not checked. If you set the condition parameter to IGNORE, the operation is not affected regardless of whether the row exists. • EXPECT_EXIST: The row is expected to exist. If the row is expected to exist but the row does not exist, the row cannot be updated and an error is returned.
return_content	ReturnContent	No	The type of data to return. Only the primary key can be returned. In most cases, this parameter is used by the auto-increment primary key column feature.

Response syntax

```
message UpdateRowResponse {
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) consumed by this request. For more information, see CU consumption.
row	bytes	The data that is returned when return_content is specified. If return_content is not specified or no value is returned, NULL is returned. The returned data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.

Use Tablestore SDKs

You can use the following Tablestore SDKs to update a row of data:

- Tablestore SDK for Java: UpdateRow
- Tablestore SDK for Go: UpdateRow
- Tablestore SDK for Python: UpdateRow
- Tablestore SDK for Node.js: UpdateRow
- Tablestore SDK for .NET: UpdateRow
- Tablestore SDK for PHP: UpdateRow

CU consumption

- If the row that you want to update does not exist, the number of consumed CUs varies based on the value that you specified for the condition parameter.
  - If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in attribute columns that you want to update)/4 KB. If UpdateRow contains an attribute column that you want to delete, the length of the column name is calculated as the size of data in the column.
  - If the value of the condition parameter is EXPECT_EXIST, data fails to be written to the row. One write CU and one read CU are consumed.
- If the row that you want to update does not exist, the number of consumed CUs varies based on the value that you specified for the condition parameter.
  - If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = (Size of data in all primary key columns of the row + Size of data in attribute columns that you want to update)/4 KB. If UpdateRow contains an attribute column that you want to delete, the length of the column name is calculated as the size of data in the column.

- If the value of the condition parameter is EXPECT_EXIST, both write and read CUs are consumed. The number of consumed write CUs is calculated by using the formula that is used when the value of the condition parameter is IGNORE. The number of consumed read CUs is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = Size of data in primary key columns of the row/4 KB.

  For more information about how to calculate the data size, see Billing overview.

- If the request times out and the results are undefined, CUs may or may not be consumed.
- If an HTTP status code 5xx is returned, which indicates that an internal error occurred, the operation does not consume CUs. If other errors are returned, one write CU and one read CU are consumed.

1.5. DeleteRow

Deletes a row of data.

Request syntax

```
message DeleteRowRequest {
    required string table_name = 1;
    required bytes primary_key = 2; // The data is encoded as binary data in the PlainBuffe
r format.
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the data table.
primary_key	bytes	Yes	The primary key of the row that you want to delete. The primary key is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.
condition	Condition	Yes	Specifies whether to check the existence of the row before data is deleted. Default value: IGNORE. Valid values: • IGNORE: The row existence is not checked. If you set the condition parameter to IGNORE, the operation is not affected regardless of whether the row exists. • EXPECT_EXIST: The row is expected to exist. When you set the condition parameter to EXPECT_EXIST, the delete operation is successful if the row exists. If the row does not exist, the delete operation fails and an error is reported.

Parameter	Type	Required	Description
return_content	ReturnContent	No	The type of data to return. Only the primary key can be returned. In most cases, this parameter is used by the auto-increment primary key column feature.

Response syntax

```
message DeleteRowResponse {
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) consumed by the operation. For more information, see CU consumption.
row	bytes	The data that is returned when return_content is specified. If return_content is not specified or no return value exists, NULL is returned. The returned data is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.

Use Tablestore SDKs

You can use the following Tablestore SDKs to delete a row of data:

- Tablestore SDK for Java: DeleteRow
- Tablestore SDK for Go: DeleteRow
- Tablestore SDK for Python: DeleteRow
- Tablestore SDK for Node.js: DeleteRow
- Tablestore SDK for .NET: DeleteRow
- Tablestore SDK for PHP: DeleteRow

CU consumption

- If the row that you want to delete does not exist, the number of consumed CUs varies based on the value that you specified for the condition parameter.
  - If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = Size of data in all primary key columns of the row/4 KB.
  - If the value of the condition parameter is EXPECT_EXIST, the row fails to be deleted. One write CU and one read CU are consumed.
- If the row that you want to delete exists, the number of consumed CUs varies based on the value that you specified for the condition parameter.

- If the value of the condition parameter is IGNORE, the number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = Size of data in all primary key columns of the row/4 KB.
- If the value of the condition parameter is EXPECT_EXIST, both write and read CUs are consumed. The number of consumed write CUs is rounded up from the value that is calculated by using the following formula: Number of consumed write CUs = Size of data in all primary key columns of the row/4 KB. The number of consumed read CUs is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = Size of data in all primary key columns of the row/4 KB.

For more information about how to calculate the data size, see Billing overview.

- If the request times out and the results are undefined, CUs may or may not be consumed.
- If an HTTP status code 5xx is returned, which indicates that an internal error occurred, the operation does not consume CUs. If other errors are returned, one read CU is consumed.

1.6. GetRange

Reads data whose primary keys are within a specified range.

Request syntax

```
message GetRangeRequest {
    required string table_name = 1;
    required Direction direction = 2;
    repeated string columns_to_get = 3; // If you do not specify this parameter, all column
s are read.
    optional TimeRange time_range = 4;
    optional int32 max_versions = 5;
    optional int32 limit = 6;
    required bytes inclusive_start_primary_key = 7; // The primary key is encoded as binary
data in the PlainBuffer format.
    required bytes exclusive_end_primary_key = 8; // The primary key is encoded as binary d
ata in the PlainBuffer format.
    optional bytes filter = 10;
    optional string start_column = 11;
    optional string end_column = 12;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table from which you want to read data.

Parameter	Type	Required	Description
direction	Direction	Yes	The order in which you want to sort the rows in the response. • If you set this parameter to FORWARD, the value of the inclusive_start_primary parameter must be smaller than the value of the exclusive_end_primary parameter, and the rows in the response are sorted in ascending order of primary key values. • If you set of this parameter to BACKWARD, the value of the inclusive_start_primary parameter must be greater than the value of the exclusive_end_primary parameter, and the rows in the response are sorted in descending order of primary key values.
columns_to_get	string	No	The names of the columns that you want to return. If you do not specify this parameter, all columns of the rows that meet the query conditions are returned. The value of this parameter can contain up to 128 strings. If duplicate column names exist, the response includes the column only once.
time_range	TimeRange	No (Either max_versions or time_range is required)	The timestamp range of data that you want to read. The unit is millisecond. Valid values: 0 to INT64.MAX. If you want to query the data within a specified timestamp range, specify start_time and end_time. If you want to query the data with a specified timestamp, specify specific_time. If the value of time_range is [100, 200), the timestamp of the data in the returned columns must be within the range of [100, 200).
max_versions	int32	No (Either max_versions or time_range is required)	The maximum number of versions of data that you want to return. If the value of max_versions is 2, up to two versions of data is returned for each column.

Parameter	Type	Required	Description
limit	int32	No	The maximum number of rows that you want to return. If the number of rows that meet the query conditions exceeds the value of this parameter, the response contains a breakpoint that records the position at which the read operation ends. The next read operation starts from this position. The value of this parameter must be greater than 0. Tablestore returns up to 5,000 rows of data regardless of whether you specify this parameter. The total size of returned rows cannot exceed 4 MB.
inclusive_start_primary_key	bytes	Yes	The primary key from which the read operation starts. The start primary key is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer. If the primary key of a row in the table from which you want to read data is the same as the start primary key, the row is included in the response.
exclusive_end_primary_key	bytes	Yes	The primary key at which the read operation ends. The end primary key is encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer. If the primary key of a row is the same as the end primary key, the row is not included in the response regardless of whether the row is in the table from which you want to read data. For the GetRange operation, the primary key columns in the values of inclusive_start_primary_key and exclusive_end_primary_key can be one of the following types dedicated to this operation: INF_MIN and INF_MAX. If the type of a primary key column in the start primary key or end primary key is INF_MIN, the value of the column in the start primary key or end primary key is smaller than the value of the column in other primary keys. If the type of a primary key column in the start primary key or end primary key is INF_MAX, the value of the column in the start primary key or end primary key is greater than the value of the column in other primary keys.
filter	bytes	No	The expression of the filter condition, which is serialized as binary data by using Protobuf. For more information, see Filter.

Parameter	Type	Required	Description
start_column	string	No	The column from which the read operation starts in a row. This parameter is used to read wide columns. The response contains the specified start column. The columns are sorted based on their names in alphabetical order. If a table contains columns a, b, and c, and the value of start_column is b, the read operation starts from column b, and columns b and c are returned.
end_column	string	No	The column at which the read operation ends in a row. This parameter is used to read wide columns. The response does not contain the specified end column. The columns are sorted based on their names in alphabetical order. If a table contains columns a, b, and c, and the value of end_column is b, the read operation ends at column b, and only column a is returned.

Response syntax

```
message GetRangeResponse {
    required ConsumedCapacity consumed = 1;
    required bytes rows = 2;
    optional bytes next_start_primary_key = 3;
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) that are consumed by the operation. For more information, see CU consumption.

Parameter	Type	Description
rows	bytes	The rows that are returned by the operation. The rows are encoded in the PlainBuffer format. For more information, see PlainBuffer. If direction in the request is set to FORWARD, the rows in the response are sorted in ascending order of primary key values. If direction in the request is set to BACKWARD, the rows in the response are sorted in descending order of primary key values. The primary key columns and attribute columns for each row in the response contain only the columns that you specified for columns_to_get in the request. The order of the columns for columns_to_get in the response may be different from the order of the columns that you specified for columns_to_get in the request. The order of the primary key columns in the response may be different from the order of the primary key columns when the table is created. If the value of the columns_to_get parameter in the request does not contain a primary key column, rows that do not contain one or more attribute columns specified in columns_to_get are not included in the response even if the primary keys of the rows are within the query range.

Parameter	Type	Description
next_start_primary_key	bytes	The breakpoint that records the position at which the read operation ends. The breakpoint is encoded in the PlainBuffer format. For more information, see PlainBuffer. If this parameter is left empty, the GetRange operation returns all rows that meet the query conditions. If this parameter is not left empty, the GetRange operation returns only the data within the range of [inclusive_start_primary_key, next_start_primary_key). If you want to obtain the remaining data, set inclusive_start_primary_key to the value of next_start_primary_key and retain the value of exclusive_end_primary_key in the original request to call the GetRange operation again. **Note** The GetRange operation can return up to 5,000 rows upon each request. The total size of the data in the rows cannot exceed 4 MB. The response to a GetRange request may contain a value of the next_start_primary_key parameter even if you do not specify a limit in the request. When you call the GetRange operation, you must check whether the response contains the next_start_primary_key parameter.

Use Tablestore SDKs

You can use the following Tablestore SDKs to read data whose primary keys are within a specified range:

- Tablestore SDK for Java: GetRange
- Tablestore SDK for Go: GetRange
- Tablestore SDK for Python: GetRange
- Tablestore SDK for Node.js: GetRange
- Tablestore SDK for .NET: GetRange
- Tablestore SDK for PHP: GetRange

CU consumption

- The number of read CUs that are consumed for the GetRange operation is rounded up from the calculation result of the following formula: Number of consumed read CUs = (Size of the data in all primary key columns of the rows that meet the query conditions + Size of the data in the attribute columns that are read)/4 KB. For more information about how to calculate the data size, see Billing overview.

- If the request times out and the results are undefined, CUs may or may not be consumed.
- If an HTTP status code 5xx is returned, which indicates that an internal error occurred, the operation does not consume CUs. If other errors are returned, one read CU is consumed.

1.7. BatchGetRow

Reads multiple rows of data from one or more tables at the same time.

The BatchGetRow operation is a set of GetRow operations. If you call the BatchGetRow operation, each GetRow operation is separately performed, the response to each GetRow operation is separately returned, and consumption units (CUs) are separately calculated for each GetRow operation.

If you call the BacthGetRow operation instead of calling the GetRow operation multiple times, the response time is reduced, and the data read performance is improved.

Request syntax

```
message BatchGetRowRequest {
    repeated TableInBatchGetRowRequest tables = 1;
}
```

Parameter	Type	Required	Description
tables	repeated TableInBatchGetRowRequest	Yes	The rows that you want to read from each table. If one of the following conditions is met, the operation fails and an error is returned: • A specified table does not exist. • The name of a specified table does not comply with the naming conventions. For more information, see Naming conventions and data types. • The primary key is not specified, the name of a primary key column does not comply with the naming conventions, or the type of a primary key column is invalid for a specified row. • The name of a column that is specified by using the columns_to_get parameter in a specified table does not comply with the naming conventions. For more information, see Naming conventions and data types. • Tables with the same names are specified. • A specified table contains rows with identical primary keys. • The number of RowInBatchGetRowRequest methods for all specified tables exceeds 100. • The RowInBatchGetRowRequest method is not specified for a specified table. • The number of columns that are specified by using the columns_to_get parameter exceeds 128 for a specified table.

Response syntax

> **Note** The BatchGetRow operation may partially fail at the row level. However, the operation still returns the HTTP 200 status code. The application must check errors in RowInBatchGetRowResponse to confirm the execution result for each row and then proceed accordingly.

```
message BatchGetRowResponse {
    repeated TableInBatchGetRowResponse tables = 1;
}
```

Parameter	Type	Description
tables	repeated TableInBatchGetRowResponse	The rows that are read from each table. The order of the TableInBatchGetRowResponse methods in the response is the same as the order of the TableInBatchGetRowRequest methods in BatchGetRowRequest. The order of the RowInBatchGetRowResponse methods in TableInBatchGetRowResponse is the same as the order of the RowInBatchGetRowRequest methods in TableInBatchGetRowRequest. If a row does not exist or if the columns that are specified by using the columns_to_get parameter in a row are empty, the RowInBatchGetRowResponse method still appears in the TableInBatchGetRowResponse method, but the primary_key_columns and attribute_columns parameters for the row are empty. If a row fails to be read, the value of is_ok in RowInBatchGetRowResponse is false for the row and the row is empty.

Use Tablestore SDKs

You can use the following Tablestore SDKs to read multiple rows of data from one or more tables at the same time:

- Tablestore SDK for Java: BatchGetRow
- Tablestore SDK for Go: BatchGetRow
- Tablestore SDK for Python: BatchGetRow
- Tablestore SDK for Node.js: BatchGetRow
- Tablestore SDK for .NET: BatchGetRow
- Tablestore SDK for PHP: BatchGetRow

CU consumption

- If the operation fails, no CUs are consumed.
- If the request times out and the results are undefined, CUs may or may not be consumed.
- In other scenarios, each RowInBatchGetRowRequest method is considered a GetRow operation for

which the write CUs are separately calculated. For more information, see GetRow.

1.8. BatchWriteRow

Inserts, modifies, or deletes multiple rows of data from one or more tables at the same time.

The BatchWriteRow operation is a set of PutRow, UpdateRow, or DeleteRow operations. If you call the BatchWriteRow operation, each PutRow, UpdateRow, or DeleteRow operation is separately performed, the response to each PutRow, UpdateRow, or DeleteRow operation is separately returned, and consumption units (CUs) are separately calculated for each PutRow, UpdateRow, or DeleteRow operation.

If you call the BacthWriteRow operation instead of calling the PutRow, UpdateRow, or DeleteRow operation multiple times, the response time is reduced, and the data write performance is improved.

Request syntax

```
message BatchWriteRowRequest {
    repeated TableInBatchWriteRowRequest tables = 1;
}
```

Parameter	Type	Required	Description

Parameter	Type	Required	Description
tables	TableInBatchWriteRowRequest	Yes	The rows on which you want to perform the batch write operation. If one of the following conditions is met, an error is returned: • A specified table does not exist. • Tables with the same names are specified. • The name of a specified table does not comply with the naming conventions. For more information, see Naming conventions and data types. • The primary key is not specified, the name of a primary key column does not comply with the naming conventions, or the type of a primary key column is invalid for a specified row. • The name of an attribute column in a specified row does not comply with the naming conventions. For more information, see Naming conventions and data types. • The name of an attribute column in a specified row is the same as the name of a primary key column in the row. • The value of a primary key column or an attribute column in a specified row exceeds the upper limit in size. For more information, see General limits. • The rows whose primary keys are the same exist in a specified table. • The total number of specified rows exceeds 200, or the total size of data in specified rows exceeds 4 MB. • If a specified table does not contain the rows on which you want to perform operations, the OTSParameterInvalidException error is returned. • The number of columns that are included in the PutRowInBatchWriteRowRequest method for a specified table exceeds 1,024. • The number of columns that are specified by using the ColumnUpdate parameter in the UpdateRowInBatchWriteRowRequest method for a specified table exceeds 1,024.

Response syntax

Note The BatchWriteRow operation may partially fail at the row level. However, the operation still returns the HTTP 200 status code. The application must check errors in RowInBatchWriteRowResponse to confirm the execution result for each row and then proceed accordingly.

```
message BatchWriteRowResponse {
    repeated TableInBatchWriteRowResponse tables = 1;
}
```

Parameter	Type	Description
tables	TableInBatchWriteRowResponse	The response to each operation for each table. The response contains the execution results, error codes, and the number of consumed CUs. The order of the TableInBatchWriteRowResponse methods in the response is the same as the order of the TableInBatchWriteRowRequest methods in BatchWriteRowRequest. The order of the RowInBatchWriteRowResponse methods that are contained in put_rows, update_rows, or delete_rows in each TableInBatchWriteRowRequest method is the same as the order of the PutRowInBatchWriteRowRequest, UpdateRowInBatchWriteRowRequest, or DeleteRowInBatchWriteRowRequest methods that are contained in put_rows, update_rows, or delete_rows in TableInBatchWriteRowRequest. If a row fails to be read, the value of is_ok in RowInBatchWriteRowResponse is false for the row.

Use Tablestore SDKs

You can use the following Tablestore SDKs to insert, modify, or delete multiple rows of data from one or more tables at the same time:

- Tablestore SDK for Java: BatchWriteRow
- Tablestore SDK for Go: BatchWriteRow
- Tablestore SDK for Python: BatchWriteRow
- Tablestore SDK for Node.js: BatchWriteRow
- Tablestore SDK for .NET: BatchWriteRow
- Tablestore SDK for PHP: BatchWriteRow

CU consumption

- If the operation fails, no CUs are consumed.
- If the request times out and the results are undefined, CUs may or may not be consumed.
- In other scenarios, each PutRowInBatchWriteRowRequest, UpdateRowInBatchWriteRowRequest, or DeleteRowInBatchWriteRowRequest method is considered a PutRow, UpdateRow, or DeleteRow operation for which CUs are separately calculated. For more information, see PutRow, UpdateRow, and DeleteRow.

1.9. CreateTable

Creates a data table based on the specified schema.

Usage notes

_

- After you create a data table, you cannot immediately perform read and write operations on the data table. In most cases, you can perform read/write operations on a table one minute after the table is created.
- You can create up to 64 tables in a single instance. If you want to create more tables in a single instance, submit a ticket.

Request syntax

```
message CreateTableRequest {
    required TableMeta table_meta = 1;
    required ReservedThroughput reserved_throughput = 2;
    optional TableOptions table_options = 3;
    optional StreamSpecification stream_spec = 5;
}
```

Parameter	Type	Required	Description
table_meta	TableMeta	Yes	The schema of the data table. The table name must be unique within the instance. A primary key can contain one to four primary key columns. The name of each primary key column must comply with the naming conventions and the data types of the primary key columns must be STRING, INTEGER, or BINARY. For more information about the naming conventions and data types, see Naming conventions and data types. After you create a table, you cannot modify the schema of the table.
reserved_throughput	ReservedThroughput	Yes	The reserved read or write throughput of the data table. The reserved read or write throughput of a table cannot exceed 100,000. The unit is CU. You can call the UpdateTable operation to change the reserved read or write throughput of a data table.
table_options	TableOptions	No	The settings of time to live (TTL) and max versions.
stream_spec	StreamSpecification	No	Specifies whether to enable Stream-related attributes.

Response syntax

```
message CreateTableResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to create a data table based on the specified schema:

- Tablestore SDK for Java: CreateTable
- Tablestore SDK for Go: CreateTable
- Tablestore SDK for Python: CreateTable
- Tablestore SDK for Node.js:CreateTable
- Tablestore SDK for .NET: CreateTable
- Tablestore SDK for PHP: CreateTable

1.10. ListTable

Queries the names of all tables that are created in an instance.

Usage notes

After you create a table, the name of the table appears in ListTableResponse. However, you cannot immediately perform read and write operations on the table.

Request syntax

```
message ListTableRequest {
}
```

Response syntax

```
message ListTableResponse {
    repeated string table_names = 1;
}
```

Parameter	Type	Description
table_names	repeated String	The names of all tables in the current instance.

Use Tablestore SDKs

You can use the following Tablestore SDKs to query the names of all tables that are created in an instance:

- Tablestore SDK for Java: ListTable
- Tablestore SDK for Go: ListTable
- Tablestore SDK for Python: ListTable
- Tablestore SDK for Node.js: ListTable
- Tablestore SDK for .NET: ListTable
- Tablestore SDK for PHP: ListTable

1.11. DeleteTable

Deletes a table from an instance.

Request syntax

```
message DeleteTableRequest {
    required string table_name = 1;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the data table that you want to delete.

Response syntax

```
message DeleteTableResponse {
}
```

If the response of the DeleteTable operation does not contain an error, the table is deleted.

Use Tablestore SDKs

You can use the following Tablestore SDKs to delete a data table:

- Tablestore SDK for Java: DeleteTable
- Tablestore SDK for Go: DeleteTable
- Tablestore SDK for Python: DeleteTable
- Tablestore SDK for Node.js: DeleteTable
- Tablestore SDK for .NET: DeleteTable
- Tablestore SDK for PHP: DeleteTable

1.12. UpdateTable

Modifies the reserved read/write throughput settings of a table. The new settings take effect within one minute.

Request syntax

```
message UpdateTableRequest {
    required string table_name = 1;
    optional ReservedThroughput reserved_throughput = 2;
    optional TableOptions table_options = 3;
    optional StreamSpecification stream_spec = 4;
}
```

Parameter	Type	Required	Description

Parameter	Type	Required	Description
table_name	string	Yes	The name of the data table for which you want to modify the reserved read/write throughput settings.
reserved_throughput	ReservedThroughput	No	The new reserved read/write throughput settings of the table. New settings take effect within one minute after a successful update. The read and write parameters of capacity_unit cannot be left empty at the same time. If you leave the read and write parameters empty at the same time, the request fails and an error message is returned.
table_options	TableOptions	No	The time to live and the maximum number of versions.
StreamSpecification	StreamSpecification	No	Specifies whether to enable Stream-related attributes.

Response syntax

```
message UpdateTableResponse {
    required ReservedThroughputDetails reserved_throughput_details = 1;
    required TableOptions table_options = 2;
}
```

Parameter	Type	Description

Parameter	Type	Description
capacity_unit_details	ReservedThroughputDetails	After a successful update, the reserved read/write throughput settings of the table include the most recent reserved read/write throughput value, the time when the reserved read/write throughput value was last updated, and the number of times that the reserved read/write throughput value was decreased on the current day. **Note** • The reserved read/write throughput settings of a table are updated at a minimum interval of 2 minutes. If you call the UpdateTable operation again within 2 minutes after the previous request, the current request is rejected. • The number of times that you can increase or decrease the reserved read/write throughput settings of a table within a single day (from 00:00:00 to 00:00:00 on the next day in UTC) is not limited.
table_options	TableOptions	The value of the table_options parameter after the update.

Use Tablestore SDKs

You can use the following Tablestore SDKs to modify the reserved read/write throughput settings of a table:

- Tablestore SDK for Java: UpdateTable
- Tablestore SDK for Go: UpdateTable
- Tablestore SDK for Python: UpdateTable
- Tablestore SDK for Node.js: UpdateTable
- Tablestore SDK for .NET: UpdateTable

- Tablestore SDK for PHP: UpdateTable

1.13. DescribeTable

Queries the schema and the reserved read/write throughput settings of a table.

Request syntax

```
message DescribeTableRequest {
    required string table_name = 1;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table whose information you want to query.

Response syntax

```
message DescribeTableResponse {
    required TableMeta table_meta = 1;
    required ReservedThroughputDetails reserved_throughput_details = 2;
    required TableOptions table_options = 3;
    optional StreamDetails stream_details = 5;
    repeated bytes shard_splits = 6;
}
```

Parameter	Type	Description
table_meta	TableMeta	The schema of the table. The schema is the same as the schema that was defined when the table was created.
reserved_throughput_details	ReservedThroughputDetails	The reserved read/write throughput settings of the table. The reserved read/write throughput settings of the table include the most recent reserved read/write throughput value, the time when the reserved read/write throughput value was last updated, and the number of times that the reserved read/write throughput value was decreased on the current day.
table_options	TableOptions	The most recent value of the table_options parameter.
stream_details	StreamDetails	Indicates whether Stream-related attributes are enabled.

Parameter	Type	Description
shard_splits	bytes	The split points of all partitions in the table.

Use Tablestore SDKs

You can use the following Tablestore SDKs to query the schema and the reserved read/write throughput settings of a table:

- Tablestore SDK for Java: DescribeTable
- Tablestore SDK for Go: DescribeTable
- Tablestore SDK for Python: DescribeTable
- Tablestore SDK for Node.js: DescribeTable
- Tablestore SDK for .NET: DescribeTable
- Tablestore SDK for PHP: DescribeTable

1.14. ComputeSplitPointsBySize

Divides data in a table into several logical splits whose sizes are approximately the same as the specified value. The split points between the splits and the information about hosts on which the splits reside are returned. This operation is used by compute engines to determine execution plans such as concurrency plans.

Request syntax

```
message ComputeSplitPointsBySizeRequest {
    required string table_name = 1;
    required int64 split_size = 2; // in 100MB
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table whose data you want to divide.
split_size	int64	Yes	The approximate size of each split. Unit: megabytes.

Response syntax

```
message ComputeSplitPointsBySizeResponse {
    required ConsumedCapacity consumed = 1;
    repeated PrimaryKeySchema schema = 2;
    /**
     * Split points between splits, in the increasing order
     *
     * A split is a consecutive range of primary keys,
     * whose data size is about split_size specified in the request.
     * The size could be hard to be precise.
     *
     * A split point is an array of primary-key column w.r.t. table schema,
     * which is never longer than that of table schema.
     * Tailing -inf will be omitted to reduce transmission payloads.
     */
    repeated bytes split_points = 3;
    /**
     * Locations where splits lies in.
     *
     * By the managed nature of TableStore, these locations are no more than hints.
     * If a location is not suitable to be seen, an empty string will be placed.
     */
     message SplitLocation {
         required string location = 1;
         required sint64 repeat = 2;
     }
     repeated SplitLocation locations = 4;
}
```

Parameter	Type	Description
consumed	ConsumedCapacity	The number of capacity units (CUs) that are consumed by this request.
schema	PrimaryKeySchema	The schema of the table. The schema is the same as the schema that was defined when the table was created.
split_points	repeated bytes	The split points between splits. The split points must increase monotonically between these splits. Each split point is a row of data in the PlainBuffer format and contains only the primary key. The last -inf of each split point is not transmitted. This helps reduce the amount of transmitted data.
locations	repeated SplitLocation	The information about the hosts on which the split points reside. This parameter can be left empty.

For example, if a table contains three primary key columns and the data type of the first primary key column is string, the following splits are obtained after you call this operation: `(-inf,-inf,-inf)` to `("a",-inf,-inf)`, `("a",-inf,-inf)` to `("b",-inf,-inf)`, `("b",-inf,-inf)` to `("c",-inf,-inf)`, `("c",-inf,-inf)` to `("d",-inf,-inf)`, and `("d",-inf,-inf)` to `(+inf,+inf,+inf)`. The first three splits reside on machine-A and the other two splits reside on machine-B. In this case, the value of split_points is `[("a"),("b"),("c"),("d")]`, and the value of locations is `"machine-A"*3,"machine-B"*2`.

Use Tablestore SDKs

Tablestore SDK for Java: Split data by a specified size

CU consumption

The number of read CUs that are consumed is the same as the number of splits. No write CUs are consumed.

1.15. ListStream

Queries stream information about all tables in an instance.

Request syntax

```
message ListStreamRequest {
    optional string table_name = 1;
}
```

Parameter	Type	Required	Description
table_name	string	No	The name of the table to which the stream belongs.

Response syntax

```
message ListStreamResponse {
    repeated Stream streams = 1;
}
message Stream {
    required string stream_id = 1;
    required string table_name = 2;
    required int64 creation_time = 3;
}
```

Parameter	Type	Description
stream_id	string	The ID of the stream.

Parameter	Type	Description
table_name	string	The name of the table to which the stream belongs.
creation_time	int64	The time when the stream was created.

Use Tablestore SDKs

Tablestore SDK for Java: ListStream

1.16. DescribeStream

Queries the shard information about a stream.

Usage notes

Before data of a shard can be read, all data of the parent shard must be read.

Request syntax

```
message DescribeStreamRequest {
    required string stream_id = 1;
    optional string inclusive_start_shard_id = 2;
    optional int32 shard_limit = 3;
}
```

Parameter	Type	Required	Description
stream_id	string	Yes	The ID of the stream.
inclusive_start_shard_id	string	No	The ID of the start shard in the query.
shard_limit	int32	No	The maximum number of shards that you want to return for the query.

Response syntax

```
message DescribeStreamResponse {
    required string stream_id = 1;
    required int32 expiration_time = 2;
    required string table_name = 3;
    required int64 creation_time = 4;
    required StreamStatus stream_status = 5;
    repeated StreamShard shards = 6;
    optional string next_shard_id = 7;
}
message StreamShard {
    required string shard_id = 1;
    optional string parent_id = 2;
    optional string parent_sibling_id = 3;
}
```

Parameter	Type	Description
stream_id	string	The ID of the stream.
expiration_time	int32	The time when the stream expires.
table_name	string	The name of the table to which the stream belongs.
creation_time	int64	The time when the stream was created.
stream_status	StreamStatus	The status of the stream. Valid values: enabling and active.
shards	StreamShard	The shard information about the stream, including the shard ID, parent shard ID, and information about the shard that is adjacent to the parent shard. If parent shards are merged, this parameter is returned.
next_shard_id	string	The ID of the start shard in the next paging query.

Use Tablestore SDKs

Tablestore SDK for Java: DescribeStream

1.17. GetShardIterator

Queries the start iterator that is used to read data from a shard.

Request syntax

```
message GetShardIteratorRequest {
    required string stream_id = 1;
    required string shard_id = 2;
}
```

Parameter	Type	Required	Description
stream_id	string	Yes	The ID of the stream.
shard_id	string	Yes	The ID of the shard.

Response syntax

```
message GetShardIteratorResponse {
    required string shard_iterator = 1;
}
```

Parameter	Type	Description
shard_iterator	string	The start iterator that is used to read data from the shard.

Use Tablestore SDKs

Tablestore SDK for Java: GetShardIterator

1.18. GetStreamRecord

Queries the incremental data of a shard.

Request syntax

```
message GetStreamRecordRequest {
    required string shard_iterator = 1;
    optional int32 limit = 2;
}
```

Parameter	Type	Required	Description
shard_iterator	string	Yes	The iterator that is used to read the incremental data of the shard.
limit	int32	No	The maximum number of data records that you want to return.

Response syntax

```
message GetStreamRecordResponse {
    message StreamRecord {
        required ActionType action_type = 1;
        required bytes record = 2;
    }
    repeated StreamRecord stream_records = 1;
    optional raw_string next_shard_iterator = 2;
    optional ConsumedCapacity consumed = 3;
}
```

Parameter	Type	Description
StreamRecord	repeated StreamRecord	The record entry that is used to read data from the current shard.
shard_iterator	string	The iterator that is used to read data from the current shard in the next GetStreamRecord request.
consumed	ConsumedCapacity	The number of capacity units (CUs) that are consumed to read stream data is rounded up from the value that is calculated by using the following formula: Number of consumed read CUs = Total size of data in all rows that are read/4 KB. For more information about how to calculate the total size of data in all rows that are actually read, see Storage usage.

Use Tablestore SDKs

Tablestore SDK for Java: GetStreamRecord

1.19. Tablestore ProtocolBuffer message definitions

This topic describes the detailed definitions of table_store.proto and table_store_filter.proto.

table_store.proto

```
package com.alicloud.openservices.tablestore.core.protocol;
message Error {
    required string code = 1;
    optional string message = 2;
}
enum PrimaryKeyType {
    INTEGER = 1;
```

```
    STRING = 2;
    BINARY = 3;
}
enum PrimaryKeyOption {
    AUTO_INCREMENT = 1;
}
message PrimaryKeySchema {
    required string name = 1;
    required PrimaryKeyType type = 2;
    optional PrimaryKeyOption option = 3;
}
message TableOptions {
    optional int32 time_to_live = 1; // The value of this parameter can be dynamically chan
ged.
    optional int32 max_versions = 2; // The value of this parameter can be dynamically chan
ged.
 optional int64 deviation_cell_version_in_sec = 5; // The value of this parameter can be dy
namically changed.
}
message TableMeta {
    required string table_name = 1;
    repeated PrimaryKeySchema primary_key = 2;
}
enum RowExistenceExpectation {
    IGNORE = 0;
    EXPECT_EXIST = 1;
    EXPECT_NOT_EXIST = 2;
}
message Condition {
    required RowExistenceExpectation row_existence = 1;
    optional bytes column_condition      = 2;
}
message CapacityUnit {
    optional int32 read = 1;
    optional int32 write = 2;
}
message ReservedThroughputDetails {
    required CapacityUnit capacity_unit = 1; // This parameter specifies the current reserv
ed throughput of the table.
    required int64 last_increase_time = 2; // This parameter specifies the time when the re
served throughput was last increased.
    optional int64 last_decrease_time = 3; // This parameter specifies the time when the re
served throughput was last decreased.
}
message ReservedThroughput {
    required CapacityUnit capacity_unit = 1;
}
message ConsumedCapacity {
    required CapacityUnit capacity_unit = 1;
}
/* ############################################  CreateTable  ###########################
################ */
/**
 * table_meta is used to store schema attributes that cannot be modified in a table. The Re
```

```
servedThroughput and TableOptions attributes that can be modified are stored as parameters
for UpdateTable.
 * message CreateTableRequest {
 *         required TableMeta table_meta = 1;
 *         required ReservedThroughput reserved_throughput = 2;
 *         required TableOptions table_options = 3;
 * }
 */
message CreateTableRequest {
    required TableMeta table_meta = 1;
    required ReservedThroughput reserved_throughput = 2;
    optional TableOptions table_options = 3;
}
message CreateTableResponse {
}
/* ##########################################################################################
################ */
/* ##########################################  UpdateTable  ##############################
################ */
message UpdateTableRequest {
    required string table_name = 1;
    optional ReservedThroughput reserved_throughput = 2;
    optional TableOptions table_options = 3;
}
message UpdateTableResponse {
    required ReservedThroughputDetails reserved_throughput_details = 1;
    required TableOptions table_options = 2;
}
/* ##########################################################################################
################ */
/* ##########################################  DescribeTable  ############################
################## */
message DescribeTableRequest {
    required string table_name = 1;
}
message DescribeTableResponse {
    required TableMeta table_meta = 1;
    required ReservedThroughputDetails reserved_throughput_details = 2;
    required TableOptions table_options = 3;
    repeated bytes shard_splits = 6;
}
/* ##########################################################################################
################## */
/* ##########################################  ListTable  ################################
############### */
message ListTableRequest {
}
/**
 * Only a simple name list is returned.
 */
message ListTableResponse {
    repeated string table_names = 1;
}
/* ##########################################################################################
```

```
############### */
/* ##########################################  DeleteTable  ############################
################# */
message DeleteTableRequest {
    required string table_name = 1;
}
message DeleteTableResponse {
}
/* ##########################################  UnloadTable  ############################
################# */
message UnloadTableRequest {
    required string table_name = 1;
}
message UnloadTableResponse {
}
/* ###################################################################################
################# */
/**
 * The value of the timestamp ranges from 0 to INT64.MAX.
 * 1. If you want to query data within a specified timestamp range, specify start_time and
end_time.
 * 2. If you want to query data with a specified timestamp, specify specific_time.
 */
message TimeRange {
    optional int64 start_time = 1;
    optional int64 end_time = 2;
    optional int64 specific_time = 3;
}
/* ##########################################  GetRow  ################################
############ */
enum ReturnType {
    RT_NONE = 0;
    RT_PK = 1;
}
message ReturnContent {
    optional ReturnType return_type = 1;
}
/**
 * 1. You can specify a timestamp range or a timestamp to read the specified version range
or version of data in a column.
 * 2. You can read data in Tablestore only by row.
 */
message GetRowRequest {
    required string table_name = 1;
    required bytes primary_key = 2; // encoded as InplaceRowChangeSet, but only has primary
key
    repeated string columns_to_get = 3; // If you do not specify this parameter, all column
s are read.
    optional TimeRange time_range = 4;
    optional int32 max_versions = 5;
    optional bytes filter = 7;
    optional string start_column = 8;
    optional string end_column = 9;
    optional bytes token = 10;
```

```
}
message GetRowResponse {
    required ConsumedCapacity consumed = 1;
    required bytes row = 2; // encoded as InplaceRowChangeSet
    optional bytes next_token = 3;
}
/* ##########################################################################################
############ */
/* ##########################################  UpdateRow  ##############################
############### */
message UpdateRowRequest {
    required string table_name = 1;
    required bytes row_change = 2;
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
message UpdateRowResponse {
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
/* ##########################################################################################
############### */
/* ##########################################  PutRow  ##################################
############ */
/**
 * You can set a timestamp for each column instead of configuring a unified timestamp for t
he entire row.
 */
message PutRowRequest {
    required string table_name = 1;
    required bytes row = 2; // encoded as InplaceRowChangeSet
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
message PutRowResponse {
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
/* ##########################################################################################
############ */
/* ##########################################  DeleteRow  ##############################
############### */
/**
 * Tablestore only allows you to delete all versions of data in all columns of a specified
row.
 */
message DeleteRowRequest {
    required string table_name = 1;
    required bytes primary_key = 2; // encoded as InplaceRowChangeSet, but only has primary
key
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
message DeleteRowResponse {
    required ConsumedCapacity consumed = 1;
```

```
    required ConsumedCapacity consumed = 1;
    optional bytes row = 2;
}
/* ##################################################################################################
############### */
/* ##############################################  BatchGetRow  ##############################
################## */
message TableInBatchGetRowRequest {
    required string table_name = 1;
    repeated bytes primary_key = 2; // encoded as InplaceRowChangeSet, but only has primary
key
    repeated bytes token = 3;
    repeated string columns_to_get = 4;  // If you do not specify this parameter, all colum
ns are read.
    optional TimeRange time_range = 5;
    optional int32 max_versions = 6;
    optional bytes filter = 8;
    optional string start_column = 9;
    optional string end_column = 10;
}
message BatchGetRowRequest {
    repeated TableInBatchGetRowRequest tables = 1;
}
message RowInBatchGetRowResponse {
    required bool is_ok = 1;
    optional Error error = 2;
    optional ConsumedCapacity consumed = 3;
    optional bytes row = 4; // encoded as InplaceRowChangeSet
    optional bytes next_token = 5;
}
message TableInBatchGetRowResponse {
    required string table_name = 1;
    repeated RowInBatchGetRowResponse rows = 2;
}
message BatchGetRowResponse {
    repeated TableInBatchGetRowResponse tables = 1;
}
/* ##################################################################################################
################ */
/* ##############################################  BatchWriteRow  ############################
################### */
enum OperationType {
    PUT = 1;
    UPDATE = 2;
    DELETE = 3;
}
message RowInBatchWriteRowRequest {
    required OperationType type = 1;
    required bytes row_change = 2; // encoded as InplaceRowChangeSet
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
message TableInBatchWriteRowRequest {
    required string table_name = 1;
    repeated RowInBatchWriteRowRequest rows = 2;
```

```
}
message BatchWriteRowRequest {
    repeated TableInBatchWriteRowRequest tables = 1;
}
message RowInBatchWriteRowResponse {
    required bool is_ok = 1;
    optional Error error = 2;
    optional ConsumedCapacity consumed = 3;
    optional bytes row = 4;
}
message TableInBatchWriteRowResponse {
    required string table_name = 1;
    repeated RowInBatchWriteRowResponse rows = 2;
}
message BatchWriteRowResponse {
    repeated TableInBatchWriteRowResponse tables = 1;
}
/* ##########################################################################################
################## */
/* ##########################################  GetRange  ##############################
############## */
enum Direction {
    FORWARD = 0;
    BACKWARD = 1;
}
message GetRangeRequest {
    required string table_name = 1;
    required Direction direction = 2;
    repeated string columns_to_get = 3;  // If you do not specify this parameter, all colum
ns are read.
    optional TimeRange time_range = 4;
    optional int32 max_versions = 5;
    optional int32 limit = 6;
    required bytes inclusive_start_primary_key = 7; // encoded as InplaceRowChangeSet, but
only has primary key
    required bytes exclusive_end_primary_key = 8; // encoded as InplaceRowChangeSet, but on
ly has primary key
    optional bytes filter = 10;
    optional string start_column = 11;
    optional string end_column = 12;
    optional bytes token = 13;
}
message GetRangeResponse {
    required ConsumedCapacity consumed = 1;
    required bytes rows = 2; // encoded as InplaceRowChangeSet
    optional bytes next_start_primary_key = 3; // If you do not specify this parameter, all
data is read. next_start_primary_key is encoded as InplaceRowChangeSet, but only contains t
he primary key.
    optional bytes next_token = 4;
}
/* ###################  ComputeSplitPointsBySize  ################### */
message ComputeSplitPointsBySizeRequest {
    required string table_name = 1;
    required int64 split_size = 2; // in 100MB
```

```
}
message ComputeSplitPointsBySizeResponse {
    required ConsumedCapacity consumed = 1;
    repeated PrimaryKeySchema schema = 2;
    /**
     * Split points between splits, in the increasing order
     *
     * A split is a consecutive range of primary keys,
     * whose data size is about split_size specified in the request.
     * The size could be hard to be precise.
     *
     * A split point is an array of primary-key column w.r.t. table schema,
     * which is never longer than that of table schema.
     * Tailing -inf will be omitted to reduce transmission payloads.
     */
    repeated bytes split_points = 3;
    /**
     * Locations where splits lies in.
     *
     * By the managed nature of TableStore, these locations are no more than hints.
     * If a location is not suitable to be seen, an empty string will be placed.
     */
     message SplitLocation {
         required string location = 1;
         required sint64 repeat = 2;
     }
     repeated SplitLocation locations = 4;
}
```

table_store_filter.proto

```
package com.alicloud.openservices.tablestore.core.protocol;
enum FilterType {
    FT_SINGLE_COLUMN_VALUE = 1;
    FT_COMPOSITE_COLUMN_VALUE = 2;
    FT_COLUMN_PAGINATION = 3;
}
enum ComparatorType {
    CT_EQUAL = 1;
    CT_NOT_EQUAL = 2;
    CT_GREATER_THAN = 3;
    CT_GREATER_EQUAL = 4;
    CT_LESS_THAN = 5;
    CT_LESS_EQUAL = 6;
}
message SingleColumnValueFilter {
    required ComparatorType comparator = 1;
    required string column_name = 2;
    required bytes column_value = 3;
    required bool filter_if_missing = 4;
    required bool latest_version_only = 5;
}
enum LogicalOperator {
    LO_NOT = 1;
    LO_AND = 2;
    LO_OR = 3;
}
message CompositeColumnValueFilter {
    required LogicalOperator combinator = 1;
    repeated Filter sub_filters = 2;
}
message ColumnPaginationFilter {
    required int32 offset = 1;
    required int32 limit = 2;
}
message Filter {
    required FilterType type = 1;
    required bytes filter = 2;  // Serialized string of filter of the type
}
```

1.20. CreateIndex

Creates an index table for a data table.

Request syntax

```
message CreateIndexRequest {
    required string main_table_name = 1;
    required IndexMeta index_meta = 2;
    optional bool include_base_data = 3;
}
```

Parameter	Type	Required	Description
main_table_name	string	Yes	The name of the data table for which you want to create an index table.
index_meta	IndexMeta	Yes	The schema of the index table that you want to create.
include_base_data	bool	No	Specifies whether to include the existing data of the data table in the index table that you want to create.

Response syntax

```
message CreateIndexResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to create an index table for a data table:

- Tablestore SDK for Java: CreateIndex
- Tablestore SDK for Go: CreateIndex
- Tablestore SDK for Python: CreateIndex
- Tablestore SDK for Node.js: CreateIndex
- Tablestore SDK for .NET: CreateIndex
- Tablestore SDK for PHP: CreateIndex

1.21. DropIndex

Deletes an index table that is created for a data table.

Request syntax

```
message DropIndexRequest {
    required string main_table_name = 1;
    required string index_name = 2;
}
```

Parameter	Type	Required	Description
main_table_name	string	Yes	The data table for which you want to delete an index table.
index_name	string	Yes	The name of the index table that you want to delete.

Response syntax

```
message DropIndexResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to delete an index table:

- Tablestore SDK for Java: DeleteIndex
- Tablestore SDK for Go: DeleteIndex
- Tablestore SDK for Python: DeleteIndex
- Tablestore SDK for Node.js: DeleteIndex
- Tablestore SDK for .NET: DeleteIndex
- Tablestore SDK for PHP: DeleteIndex

1.22. CreateTimeseriesTable

Creates a time series table.

Request syntax

```
message CreateTimeseriesTableRequest {
  required TimeseriesTableMeta table_meta = 1;
}
```

Parameter	Type	Required	Description
table_meta	TimeseriesTableMeta	Yes	The configurations of the time series table.

Response syntax

```
message CreateTimeseriesTableResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to create a time series table:

- Tablestore SDK for Java: CreateTimeseriesTable
- Tablestore SDK for Go: CreateTimeseriesTable

1.23. ListTimeseriesTable

Queries the names of time series tables in an instance.

Request syntax

```
message ListTimeseriesTableRequest {
}
```

Response syntax

```
message ListTimeseriesTableResponse {
  repeated TimeseriesTableMeta table_metas = 1;
}
```

Parameter	Type	Description
table_metas	TimeseriesTableMeta	The configurations of all time series tables.

Use Tablestore SDKs

You can use the following Tablestore SDKs to query the names of time series tables in an instance:

- Tablestore SDK for Java: ListTimeseriesTable
- Tablestore SDK for Go: ListTimeseriesTable

1.24. DescribeTimeseriesTable

Queries information about a time series table.

Request syntax

```
message DescribeTimeseriesTableRequest {
  required string table_name = 1;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.

Response syntax

```
message DescribeTimeseriesTableResponse {
  required TimeseriesTableMeta table_meta = 1;
}
```

Parameter	Type	Description
table_meta	TimeseriesTableMeta	The configurations of the time series table.

Use Tablestore SDKs

You can use the following Tablestore SDKs to query information about a time series table:

- Tablestore SDK for Java: DescribeTimeseriesTable
- Tablestore SDK for Go: DescribeTimeseriesTable

1.25. UpdateTimeseriesTable

Updates the configurations of a time series table.

Request syntax

```
message UpdateTimeseriesTableRequest {
  required string table_name = 1;
  optional TimeseriesTableOptions table_options = 2;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
table_options	TimeseriesTableOptions	Yes	The configurations of the time series table.

Response syntax

```
message UpdateTimeseriesTableResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to update the configurations of a time series table:

- Tablestore SDK for Java: UpdateTimeseriesTable
- Tablestore SDK for Go: UpdateTimeseriesTable

1.26. DeleteTimeseriesTable

Deletes a time series table.

Request syntax

```
message DeleteTimeseriesTableRequest {
  required string table_name = 1;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.

Response syntax

```
message DeleteTimeseriesTableResponse {
}
```

Use Tablestore SDKs

You can use the following Tablestore SDKs to delete a time series table:

- Tablestore SDK for Java: DeleteTimeseriesTable
- Tablestore SDK for Go: DeleteTimeseriesTable

1.27. PutTimeseriesData

Writes multiple rows of time series data to a time series table.

Request syntax

```
message PutTimeseriesDataRequest {
  required string table_name = 1;
  required TimeseriesRows rows_data = 2;
  optional MetaUpdateMode meta_update_mode = 3;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
rows_data	TimeseriesRows	Yes	The rows of time series data that you want to write to the time series table.
meta_update_mode	MetaUpdateMode	No	The metadata update mode.

Response syntax

```
message PutTimeseriesDataResponse {
  repeated FailedRowInfo failed_rows = 1;
  optional MetaUpdateStatus meta_update_status = 2;
}
```

Parameter	Type	Description
failed_rows	FailedRowInfo	The information about the rows that fail to be written.
meta_update_status	MetaUpdateStatus	The metadata update status.

Use Tablestore SDKs

You can use the following Tablestore SDKs to write multiple rows of time series data to a time series table:

- Tablestore SDK for Java: PutTimeseriesData
- Tablestore SDK for Go: PutTimeseriesData

1.28. GetTimeseriesData

Reads the data in a time series.

Request syntax

```
message GetTimeseriesDataRequest {
  required string table_name = 1;
  required TimeseriesKey time_series_key = 2;
  optional int64 begin_time = 3;
  optional int64 end_time = 4;
  optional int64 specific_time = 5; // not used
  optional bytes token = 6; // bytes
  optional int32 limit = 7;
  optional bool backward = 8;
  repeated TimeseriesFieldsToGet fields_to_get = 9;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
time_series_key	TimeseriesKey	Yes	The identifier of the time series.
begin_time	int64	No	The start of the time range to read.
end_time	int64	No	The end of the time range to read.
specific_time	int64	No	The specific point in time of the time series data that you want to read.
token	bytes	No	The token that is used to read the remaining data.
limit	int32	No	The maximum number of rows that you want to return.
backward	bool	No	Specifies whether to read data in reverse chronological order. By default, data is read in chronological order.
fields_to_get	TimeseriesFieldsToGet	No	The data columns that you want to read.

Response syntax

```
message GetTimeseriesDataResponse {
  required bytes rows_data = 1;
  optional bytes next_token = 2;
}
```

Parameter	Type	Description
rows_data	bytes	The rows of time series data that are returned.
next_token	bytes	The token that is used to read the remaining data.

Use Tablestore SDKs

You can use the following Tablestore SDKs to read data in a time series:

- Tablestore SDK for Java: GetTimeseriesData
- Tablestore SDK for Go: GetTimeseriesData

1.29. QueryTimeseriesMeta

Retrieves the time series metadata from a time series table.

Request syntax

```
message QueryTimeseriesMetaRequest {
  required string table_name = 1;
  optional MetaQueryCondition condition = 2;
  optional bool get_total_hit = 3;
  optional bytes token = 4;
  optional int32 limit = 5;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
condition	MetaQueryCondition	No	The condition that is used to retrieve the time series metadata.
get_total_hit	bool	No	Specifies whether to return the total number of rows that meet the specified retrieval condition.
token	bytes	No	The token that is used to read the remaining data.
limit	int32	No	The maximum number of rows that you want to return.

Response syntax

```
message QueryTimeseriesMetaResponse {
  repeated TimeseriesMeta timeseries_metas = 1;
  optional int64 total_hit = 2;
  optional bytes next_token = 3;
}
```

Parameter	Type	Description
timeseries_metas	TimeseriesMeta	The list of time series metadata.
total_hit	int64	The total number of rows that meet the specified retrieval condition.
next_token	bytes	The token that is used to read the remaining data.

Use Tablestore SDKs

You can use the following Tablestore SDKs to retrieve the time series metadata from a time series table:

- Tablestore SDK for Java: QueryTimeseriesMeta
- Tablestore SDK for Go: QueryTimeseriesMeta

1.30. UpdateTimeseriesMeta

Updates the time series metadata in a time series table.

Request syntax

```
message UpdateTimeseriesMetaRequest {
  required string table_name = 1;
  repeated TimeseriesMeta timeseries_meta = 2;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
timeseries_meta	TimeseriesMeta	Yes	The list of time series metadata that you want to update.

Response syntax

```
message UpdateTimeseriesMetaResponse {
  repeated FailedRowInfo failed_rows = 1;
}
```

Parameter	Type	Description
failed_rows	FailedRowInfo	The information about the rows that fail to be updated.

Use Tablestore SDKs

You can use the following Tablestore SDKs to update the time series metadata in a time series table:

- Tablestore SDK for Java: UpdateTimeseriesMeta
- Tablestore SDK for Go: UpdateTimeseriesMeta

1.31. DeleteTimeseriesMeta

Deletes the metadata of a time series.

Request syntax

```
message DeleteTimeseriesMetaRequest {
  required string table_name = 1;
  repeated TimeseriesKey timeseries_key = 2;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
timeseries_key	TimeseriesKey	Yes	The identifier of the time series whose metadata you want to delete.

Response syntax

```
message DeleteTimeseriesMetaResponse {
  repeated FailedRowInfo failed_rows = 1;
}
```

Parameter	Type	Description
failed_rows	FailedRowInfo	The information about the rows that fail to be deleted.

Use Tablestore SDKs

Tablestore SDK for Java: DeleteTimeseriesMeta

2.Data Types

2.1. List of operations by data type

This topic describes the API operations by data type available for use in Tablestore.

- ActionType
- CapacityUnit
- ColumnPaginationFilter
- ComparatorType
- CompositeColumnValueFilter
- PrimaryKeyType
- Condition
- ConsumedCapacity
- Direction
- Error
- Filter
- FilterType
- LogicalOperator
- OperationType
- PartitionRange
- PlainBuffer
- PrimaryKeyOption
- RowInBatchGetRowResponse
- PrimaryKeySchema
- ReservedThroughput
- ReservedThroughputDetails
- ReturnContent
- ReturnType
- RowExistenceExpectation
- RowInBatchWriteRowRequest
- RowInBatchWriteRowResponse
- SingleColumnValueFilter
- ValueTransferRule
- VariantType
- StreamDetails
- StreamRecord
- StreamSpecification
- TableInBatchGetRowRequest
- TableInBatchGetRowResponse
- TableInBatchWriteRowRequest

- TableInBatchWriteRowResponse
- TableMeta
- TableOptions
- TimeRange

2.2. ActionType

ActionType specifies the operation type in the response message of the GetStreamRecord operation. Operation types include:

- PUT_ROW, which indicates that the operation type is PutRow.
- UPDATE_ROW, which indicates that the operation type is UpdateRow.
- DELETE_ROW, which indicates that the operation type is DeleteRow.

Enumeration value list

```
enum ActionType {
    PUT_ROW = 1;
    UPDATE_ROW = 2;
    DELETE_ROW = 3;
}
```

Related operations

GetStreamRecord

2.3. CapacityUnit

CapacityUnit indicates the value of the capacity units consumed in an operation, or a table's reserved read/write throughput.

Data structure

```
message CapacityUnit {
    optional int32 read = 1;
    optional int32 write = 2;
}
```

name	type	description
read	int32	The read capacity units consumed by this operation or the reserved read throughput for this table. Unit is CU.
write	int32	The write capacity units consumed by this operation or the reserved write throughput for this table. Unit is CU.

Related operations

UpdateRow

BatchWriteRow

2.4. ColumnPaginationFilter

ColumnPaginationFilter specifies the filtering conditions for reading a wide row. It is applicable to the Filter function.

Data structure

```
message ColumnPaginationFilter {
    required int32 offset = 1;
    required int32 limit = 2;
}
```

offset:

- Type: int32
- The position of the starting column, that is, the first column to be read

limit:

- Type: int32
- The number of columns to be read

Related operations

Filter

GetRow

GetRange

BatchGetRow

2.5. ComparatorType

ComparatorType is an relational operator. It includes the following enumeration types:

- CT_EQUAL, which indicates equal.
- CT_NOT_EQUAL, which indicates not equal.
- CT_GREATER_THAN, which indicates greater than.
- CT_GREATER_EQUAL, which indicates not less than.
- CT_LESS_THAN, which indicates less than.
- CT_LESS_EQUAL, which indicates not greater than.

Enumeration value list

```
enum ComparatorType {
    CT_EQUAL                        = 1;
    CT_NOT_EQUAL                 = 2;
    CT_GREATER_THAN           = 3;
    CT_GREATER_EQUAL         = 4;
    CT_LESS_THAN                 = 5;
    CT_LESS_EQUAL                = 6;
}
```

2.6. CompositeColumnValueFilter

CompositeColumnValueFilter specifies a group of conditions, for example, column_a > 5 AND column_b = 10. It is applicable to the ConditionUpdate and Filter functions.

Data structure

```
message CompositeColumnValueFilter {
    required LogicalOperator combinator = 1;
    repeated Filter sub_filters = 2;
}
```

name	type	description
combinator	LogicalOperator	The logical operator.
sub_filters	Filter	The sub-condition expression.

Related operations

- ConditionUpdate
  - PutRow
  - UpdateRow
  - DeleteRow
  - BatchWriteRow
- Filter
  - GetRow
  - GetRange
  - BatchGetRow

2.7. PrimaryKeyType

PrimaryKeyType specifies the type of the primary key.

Enumeration data type

- INTEGER: Integer
- STRING: String

- BINARY: Binary

```
enum PrimaryKeyType {
    INTEGER = 1;
    STRING = 2;
    BINARY = 3;
}
```

2.8. Condition

Condition specifies the row judgment conditions in PutRow, UpdateRow, and DeleteRow. It contains the row_existence and column_condition parameters.

Data structure

```
message Condition {
    required RowExistenceExpectation row_existence = 1;
    optional bytes column_condition      = 2;
}
```

row_existence:

- Type: see RowExistenceExpectation
- Description: the row existence check settings for this row.

column_condition:

- Type: bytes
- Description: the column-based condition settings. This parameter indicates the binary data (in bytes) after the filter is serialized in the Protobuf format. For more information, see Filter.

Operations

PutRow

UpdateRow

DeleteRow

BatchWriteRow

2.9. ConsumedCapacity

ConsumedCapacity indicates capacity units consumed by an operation.

Data structure

```
message ConsumedCapacity {
    required CapacityUnit capacity_unit = 1;
}
```

capacity_unit:

- Type: CapacityUnit
- The value of capacity units consumed by this operation.

2.10. Direction

Direction indicates the data query order in the GetRange operation. The two Direction operations are:

- FORWARD, which indicates that the query proceeds from the smallest to the largest primary key.
- BACKWARD, which indicates that the query proceeds from the largest to the smallest primary key.

Enumeration value list

```
enum Direction {
    FORWARD = 0;
    BACKWARD = 1;
}
```

Related operations

GetRange

2.11. Error

Error indicates an error message returned when an operation fails. This parameter is also used to indicate errors for single-row requests in the BatchGetRow and BatchWriteRow operations.

Data structure

```
Error {
    required string code = 1;
    optional string message = 2;
}
```

code:

- Type: string
- Description: the error code for the current single-row operation.

message:

- Type: string
- Description: the error message for the current single-row operation. For more information about error messages, see the topic about error code.

Operations

BatchGetRow

BatchWriteRow

2.12. Filter

Filter specifies the column-based conditions. It is applicable to the Conditional Update and Filter functions.

Data structure

```
message Filter {
    required FilterType type = 1;
    required bytes filter = 2;
}
```

type:

- Type: FilterType
- Column condition type.

filter:

- Type: bytes
- The binary data of the condition statement in CompositeColumnValueFilter, ColumnPaginationFilter, or SingleColumnValueFilter type after being serialized in Protobuf format.

Related operations

- ConditionUpdate

  PutRow

  UpdateRow

  DeleteRow

  BatchWriteRow
- Filter

  GetRow

  GetRange

  BatchGetRow

2.13. FilterType

FilterType specifies the type for condition updating or filtering. The types include:

- FT_SINGLE_COLUMN_VALUE, which is a single-column condition.
- FT_COMPOSITE_COLUMN_VALUE, which is a multi-column composite condition.
- FT_COLUMN_PAGINATION, which is a condition for wide-row read.

Enumeration value list

```
enum FilterType {
    FT_SINGLE_COLUMN_VALUE = 1;
    FT_COMPOSITE_COLUMN_VALUE = 2;
    FT_COLUMN_PAGINATION = 3;
}
```

2.14. LogicalOperator

LogicalOperator includes the following enumeration types:

- LO_NOT, which indicates NOT.
- LO_AND, which indicates AND.
- LO_OR, which indicates OR.

Enumeration value list

```
enum LogicalOperator {
    LO_NOT                      = 1;
    LO_AND                      = 2;
    LO_OR                        = 3;
}
```

2.15. OperationType

In UpdateRow, OperationType indicates the modification method for one column. The types of operations include:

- PUT, which indicates that a column is inserted or this column's data is overwritten.
- UPDATE, which indicates that a column is updated.
- DELETE, which indicates that a column is deleted.

Enumeration value list

```
enum OperationType {
    PUT = 1;
    UPDATE = 2;
    DELETE = 3;
}
```

2.16. PartitionRange

PartitionRange specifies the shard range.

```
message PartitionRange {
    required bytes begin = 1; // encoded as SQLVariant
    required bytes end = 2; // encoded as SQLVariant
}
```

begin:

- Type: Bytes
- The starting key of the shard. The shard is a left-closed right-open interval, which includes the starting key. It is in Plainbuffer format. For more information about encoding, see Plainbuffer encoding.

end:

- Type: Bytes
- The ending key of the shard. The shard is a left-closed right-open interval, which does not include the ending key. It is in Plainbuffer format. For more information about encoding, see Plainbuffer encoding.

2.17. PlainBuffer

The PlainBuffer format is defined in Table Store to indicate row data, because it delivers better performance for serialization, and small object resolution, compared to Protocol Buffer.

Format definition

```
plainbuffer = tag_header row1  [row2]  [row3]
row = ( pk [attr] | [pk] attr | pk attr ) [tag_delete_marker] row_checksum;
pk = tag_pk cell_1 [cell_2] [cell_3]
attr  = tag_attr cell1 [cell_2] [cell_3]
cell = tag_cell cell_name [cell_value] [cell_op] [cell_ts] cell_checksum
cell_name = tag_cell_name  formated_value
cell_value = tag_cell_value formated_value
cell_op = tag_cell_op  cell_op_value
cell_ts = tag_cell_ts cell_ts_value
row_checksum = tag_row_checksum row_crc8
cell_checksum = tag_cell_checksum row_crc8
formated_value = value_type value_len value_data
value_type = int8
value_len = int32
cell_op_value = delete_all_version | delete_one_version
cell_ts_value = int64
delete_all_version = 0x01 (1byte)
delete_one_version = 0x03 (1byte)
```

Tag value

```
tag_header = 0x75 (4byte)
tag_pk = 0x01 (1byte)
tag_attr = 0x02 (1byte)
tag_cell = 0x03 (1byte)
tag_cell_name = 0x04 (1byte)
tag_cell_value = 0x05 (1byte)
tag_cell_op = 0x06 (1byte)
tag_cell_ts = 0x07 (1byte)
tag_delete_marker = 0x08 (1byte)
tag_row_checksum = 0x09 (1byte)
tag_cell_checksum = 0x0A (1byte)
```

ValueType value

The values of value_type in formated_value are as follows:

```
VT_INTEGER = 0x0
VT_DOUBLE = 0x1
VT_BOOLEAN = 0x2
VT_STRING = 0x3
VT_NULL = 0x6
VT_BLOB = 0x7
VT_INF_MIN = 0x9
VT_INF_MAX = 0xa
VT_AUTO_INCREMENT = 0xb
```

Calculate the checksum

The basic logic for calculating the checksum is as follows:

- Calculate the name/value/type/time stamp of each cell.
- Calculate the delete in the row. If delete mark exists in the row, supplement a single-byte 1; otherwise, supplement a single-byte 0.
- Because the checksum is calculated for each cell, the cell checksum is used to calculate the row checksum, that is, the CRC operation is only performed on the checksums of cells in the row, not data in the row.

Java implementation:

> **Note** The following code is from `$tablestore-4.2.1-sources/com/alicloud/openservices/tablestore/core/protocol/PlainBufferCrc8.java`. For more information, see Java SDK.

```
public static byte getChecksum(byte crc, PlainBufferCell cell) throws IOException {
    if (cell.hasCellName()) {
        crc = crc8(crc, cell.getNameRawData());
    }
    if (cell.hasCellValue()) {
        if (cell.isPk()) {
            crc = cell.getPkCellValue().getChecksum(crc);
        } else {
            crc = cell.getCellValue().getChecksum(crc);
        }
    }
    if (cell.hasCellTimestamp()) {
        crc = crc8(crc, cell.getCellTimestamp());
    }
    if (cell.hasCellType()) {
        crc = crc8(crc, cell.getCellType());
    }
    return crc;
}
public static byte getChecksum(byte crc, PlainBufferRow row) throws IOException {
    for (PlainBufferCell cell : row.getPrimaryKey()) {
        crc = crc8(crc, cell.getChecksum());
    }
    for (PlainBufferCell cell : row.getCells()) {
        crc = crc8(crc, cell.getChecksum());
    }
    byte del = 0;
    if (row.hasDeleteMarker()) {
        del = (byte)0x1;
    }
    crc = crc8(crc, del);
    return crc;
}
```

Example

A data row contains two primary key columns and four data columns. The primary key types are string and integer, and the data types are string, int, and double. The versions are 1001, 1002, and 1003. A column is also contained to delete all versions.

- Primary key column:
  - [pk1:string:iampk]
  - [pk2:integer:100]
- Attribute column:
  - [column1:string:bad:1001]
  - [column2:integer:128:1002]
  - [column3:double:34.2:1003]
  - [column4:del_all_versions]

Encoding:

```
  <HHeader starting>[0x75]
  <rimary key column starting>[0x1]
  <Cell1>[0x3][0x4][0x3][3][pk1][0x5][0x3][5][iampk][_cell_checksum]
    <Cell2>[0x3][0x4][0x3][3][pk2][0x5][0x0][8][100][_cell_checksum]
  <Attribute column starting>[0x2]
    <Cell1>[0x3][0x4][0x3][7][column1][0x5][0x3][3][bad][0x7][1001][_cell_checksum]
    <Cell2>[0x3][0x4][0x3][7][column2][0x5][0x0][8][128][0x7][1002][_cell_checksum]
    <Cell3>[0x3][0x4][0x3][7][column3][0x5][0x1][8][34.2][0x7][1003][_cell_checksum]
    <Cell4>[0x3][0x4][0x3][7][column4][0x6][1][_cell_checksum]
  [_row_check_sum]
```

2.18. PrimaryKeyOption

PrimaryKeyOption specifies the attribute value of the primary key. Currently, it only supports AUTO_INCREMENT.

Enumeration value list

```
enum PrimaryKeyOption {
    AUTO_INCREMENT = 1;
}
```

2.19. RowInBatchGetRowResponse

RowInBatchGetRowResponse indicates a data row in the response message of the BatchGetRow operation.

Data structure

```
message RowInBatchGetRowResponse {
    required bool is_ok = 1 [default = true];
    optional Error error = 2;
    optional ConsumedCapacity consumed = 3;
    optional bytes row = 4;
 optional bytes next_token = 5;
}
```

is_ok:

- Type: Bool
- Determines whether the operation for this row is successful. If the value is true, the row is read successfully and error is invalid. If the value is false, this row fails to be read and row is invalid.

error:

- Type: Error
- The error message for this row operation.

consumed:

- Type: ConsumedCapacity

- The capacity units consumed by this row operation.

row:

- Type: Bytes
- The read data is encoded in Plainbuffer format. For more information, see Plainbuffer encoding.
- If this row does not exist, null is returned.

next_token:

- Type: Bytes
- It indicates the starting position of a wide row to be read during the next operation of which is unavailable currently.

Related operations

BatchGetRow

2.20. PrimaryKeySchema

PrimaryKeySchema specifies the attribute value of the primary key.

Data structure

```
message PrimaryKeySchema {
    required string name = 1;
    required PrimaryKeyType type = 2;
    optional PrimaryKeyOption option = 3;
}
```

name:

- Type: String
- The name of the column.

type:

- Type: PrimaryKeyType
- The type of the column.

option:

- Type: PrimaryKeyOption
- The additional attribute value of the column.

2.21. ReservedThroughput

ReservedThroughput indicates the provisioned throughput capacity reserved for read/write value set for a table.

Data structure

```
message ReservedThroughput {
    required CapacityUnit capacity_unit = 1;
}
```

capacity_unit:

- Type: CapacityUnit
- The current reserved read/write throughput value.

Related operations

CreateTable

UpdateRow

DescribeTable

2.22. ReservedThroughputDetails

ReservedThroughputDetails indicates the provisioned throughput capacity reserved for read/write information for a table.

Data structure

```
message ReservedThroughputDetails {
    required CapacityUnit capacity_unit = 1;
    required int64 last_increase_time = 2;
    optional int64 last_decrease_time = 3;
    required int32 number_of_decreases_today = 4;
}
```

capacity_unit:

- Type: CapacityUnit
- The provisioned throughput capacity reserved for read/write value for the specified table.

last_increase_time:

- Type: int64
- The last time the provisioned throughput capacity reserved for read/write settings were raised for this table, expressed in UTC time at second level.

last_decrease_time:

- Type: int64
- The last time the provisioned throughput capacity reserved for read/write settings were lowered for this table, expressed in UTC time at second level.

number_of_decreases_today:

- Type: int32
- The number of times the table's provisioned throughput capacity reserved for read/write settings were lowered on the current calendar day.

Related operations

UpdateTable

DescribeTable

2.23. ReturnContent

ReturnContent specifies the content of the returned data.

Data structure

```
message ReturnContent {
    optional ReturnType return_type = 1;
}
```

return_type:

- Type: ReturnType
- The type of the returned data.

2.24. ReturnType

ReturnType specifies the type of the returned data.

Enumeration data type

```
enum ReturnType {
    RT_NONE = 0;
    RT_PK = 1;
}
```

- RT_NONE: The default value. It indicates that no value is returned.
- RT_PK: It indicates that the primary key is returned.

2.25. RowExistenceExpectation

RowExistenceExpectations includes conditions for row existence in enumeration type. The conditions include:

- IGNORE, which indicates that the row existence check is not performed.
- EXPECT_EXIST, which indicates that the row is expected to exist.
- EXPECT_NOT_EXIST, which indicates that this row is not expected to exist.

Enumeration value list

```
enum RowExistenceExpectation {
    IGNORE = 0;
    EXPECT_EXIST = 1;
    EXPECT_NOT_EXIST = 2;
}
```

Related operations

PutRow

UpdateRow

DeleteRow

BatchWriteRow

2.26. RowInBatchWriteRowRequest

RowInBatchWriteRowRequest indicates the information of the row to be written, updated, or deleted in the BatchWriteRow operation.

Data structure

```
message RowInBatchWriteRowRequest {
    required OperationType type = 1;
    required bytes row_change = 2; // encoded as InplaceRowChangeSet
    required Condition condition = 3;
    optional ReturnContent return_content = 4;
}
```

type:

- Type: OperationType
- The operation type.

row_change:

- Type: Bytes
- The row data, which includes the primary key columns and attribute columns. The columns are encoded in Plainbuffer format. For more information, see Plainbuffer encoding.

condition:

- Type: Condition
- The value of conditional update, including the row existence condition and column-based condition.

return_content:

- Type: ReturnContent
- Required parameter: Yes
- The data type after the row is successfully written. Currently, only the primary key can be returned, which is used for the auto-increment function of the primary key column.

Related operations

BatchWriteRow

2.27. RowInBatchWriteRowResponse

RowInBatchWriteRowResponse indicates the write operation result for a row in the response message of the BatchWriteRow operation.

Data structure

```
message RowInBatchWriteRowResponse {
    required bool is_ok = 1 [default = true];
    optional Error error = 2;
    optional ConsumedCapacity consumed = 3;
}
```

is_ok:

- Type: Bool
- Determines whether the operation for this row is successful. If the value is true, the row is written successfully and error is invalid. If the value is false, the row fails to be written.

error:

- Type: Error
- The error message for this row operation.

consumed:

- Type: ConsumedCapacity
- The capacity units consumed by this row operation.

Related operations

BatchWriteRow

2.28. SingleColumnValueFilter

SingleColumnValueFilter specifies a single filter condition, for example, column_a>5. It is applicable to ConditionUpdate (conditional update) and Filter features.

Data structure

```
message SingleColumnValueFilter {
    required ComparatorType comparator = 1;
    required string column_name = 2;
    required bytes column_value = 3;
    required bool filter_if_missing = 4;
    required bool latest_version_only = 5;
    optional ValueTransferRule value_transfer_rule =6;
}
```

Parameter	Type	Required	Description
comparator	ComparatorType	Yes	The relational operator.
column_name	string	Yes	The name of the column.
column_value	bytes	Yes	The column value after PlainBuffer encoding.
filter_if_missing	bool	Yes	Specifies whether the filter condition is applicable when the specified column in a row does not exist. Valid values: • true: The row is kept. This is the default value. • false: The row is filtered out. For example, if the filter condition is column_a>0 and filter_if_missing is set to true, when column_a does not exist in a row, the row is kept.
latest_version_only	bool	Yes	Specifies whether the filter condition applies only to the latest version. Valid values: • true: returns a row only when the latest version of the column value matches the condition. This is the default value. • false: returns a row when any version of the column value matches the condition.
value_transfer_rule	ValueTransferRule	No	Converts the data type of a value that is obtained by matching a regular expression to String, Integer, or Double. If you store column data in a custom format such as JSON string and want to filter and query the column data by using a subfield value, you must set this parameter.

Related operations

- Conditional update

  PutRow

  UpdateRow

  DeleteRow

  BatchWriteRow

- Configure a filter

  GetRow

GetRange

BatchGetRow

2.29. ValueTransferRule

ValueTransferRule converts a string to the String, Integer, or Double type after using a regular expression to match the string.

Data structure

```
message ValueTransferRule {
    required string regex = 1;
    optional VariantType cast_type = 2;
}
```

Parameter	Type	Required	Description

Parameter	Type	Required	Description
regex	string	Yes	A regular expression that is used to match strings. The regular expression must meet the following conditions: • Contains a maximum of 256 bytes in length. • Supports the syntax of regular expressions in Perl. • Supports the single-byte regular expressions. • Supports only the expression matching in languages excluding Chinese. • Supports full matching and partial matching of regular expressions. In partial matching mode, regular expressions are separated by a pair of parentheses (). If the full matching mode is used, the first matching result is returned. If particle matching mode is used the first submatch is returned. For example, if the column value is 1aaa51bbb5 and the regular expression is 1[a-z]+5, the return value is 1aaa5. If the regular expression is 1([a-z]+)5, the return value is aaa.
cast_type	VariantType	Yes	Converts a string to the String, Integer, or Double type for subsequent relational operations.

2.30. VariantType

VariantType indicates the data types after string conversion.

Enumeration data structure

```
enum VariantType {
    VT_INTEGER = 0;
    VT_DOUBLE = 1;
    VT_STRING = 3;
}
```

- VT_INTEGER: integer type
- VT_DOUBLE: double-precision floating-point type
- VT_STRING: string type

2.31. StreamDetails

StreamDetails indicates the stream of a table.

Data structure

```
message StreamDetails {
    required bool enable_stream = 1;
    optional string stream_id = 2;
    optional int32 expiration_time = 3;
    optional int64 last_enable_time = 4;
}
```

名称	类型	描述
enable_stream	bool	Determines whether the stream is enabled for the table.
stream_id	string	The ID of the stream of the table.
expiration_time	int32	The expiration time of the stream of the table.
last_enable_time	int64	The time for enabling the stream of the table.

Related operations

DescribeTable

2.32. StreamRecord

StreamRecord indicates a data row in the response message of the GetStreamRecord operation.

Data structure

```
 message StreamRecord {
        required ActionType action_type = 1;
        required bytes record = 2;
}
```

action_type:

- Type: Required ActionType
- The operation type of the row.

record:

- Type: Required bytes
- The data content of the row.

Related operations

GetStreamRecord

2.33. StreamSpecification

StreamSpecification indicates the stream of a table.

Data structure

```
message StreamSpecification {
    required bool enable_stream = 1;
    optional int32 expiration_time = 2;
}
```

enable_stream:

- Type: Bool
- Determines whether the stream is enabled for the table.

expiration_time:

- Type: int32
- The expiration time of the stream of the table.

Related operations

CreateTable

DescribeTable

UpdateTable

2.34. TableInBatchGetRowRequest

In the BatchGetRow operation, TableInBatchGetRowRequest indicates a request to read the data in a table.

Data structure

```
message TableInBatchGetRowRequest {
    required string table_name = 1;
    repeated bytes primary_key = 2; // The primary key columns are encoded in the PlainBuff
er format.
    repeated bytes token = 3;
    repeated string columns_to_get = 4;  // If this parameter is not specified, all columns
are read.
    optional TimeRange time_range = 5;
    optional int32 max_versions = 6;
    optional bool cache_blocks = 7 [default = true]; // This parameter specifies whether th
e read data is written to BlockCache.
    optional bytes filter = 8;
    optional string start_column = 9;
    optional string end_column = 10;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the table.
primary_key	repeated bytes	Yes	All the primary key columns of the row, including the primary key column names and values. The primary key columns are encoded in the PlainBuffer format. For more information about PlainBuffer, see PlainBuffer.
token	repeated bytes	No	The start position of a wide row in the next read request. This parameter is unavailable.
columns_to_get	repeated string	No	The names of the columns to return.
time_range	TimeRange	At least one of time_range and max_versions is required.	The timestamp range of data to read. • The minimum value is 0, and the maximum value is INT64.MAX. Unit: milliseconds. • To query the data within a time range, specify start_time and end_time. • To query the data with a specific timestamp, specify specific_time. Example: If the value of time_range is [100, 200), the timestamp of the data in the returned column must be within the range of [100, 200).

Parameter	Type	Required	Description
max_versions	int32	At least one of max_versions and time_range is required.	The maximum number of versions of data to return. Example: If the value of max_versions is 2, a maximum of two versions of data is returned for each column.
cache_blocks	bool	No	Specifies whether the read data is written to BlockCache. The default value is true. You cannot set the value to false in the current version.
filter	bytes	No	The expression of the filter condition. The expression of the filter condition is serialized as binary data by using Protobuf. For more information, see Filter.
start_column	string	No	The column from which the read starts in a row. This parameter is used to read wide rows. • The response contains the specified start column. • The columns are sorted based on their names in alphabetical order. Example: If a table contains columns a, b, and c, and the value of start_column is b, the current read starts from column b, and columns b and c are returned.
end_column	string	No	The column at which the read ends in the row. This parameter is used to read wide rows. • The response does not contain the specified end column. • The columns are sorted based on their names in alphabetical order. Example: If a table contains columns a, b, and c, and the value of end_column is b, the current read ends at column b, and only column a is returned.

Related operation

BatchGetRow

2.35. TableInBatchGetRowResponse

TableInBatchGetRowResponse indicates data in a table in the messages returned by the BatchGetRow operation.

Data structure

```
message TableInBatchGetRowResponse {
    required string table_name = 1;
    repeated RowInBatchGetRowResponse rows = 2;
}
```

table_name:

- Type: string
- The name of the table.

rows:

- Type: repeated RowInBatchGetRowResponse
- All row data read in the table.

Related operations

BatchGetRow

2.36. TableInBatchWriteRowRequest

TableInBatchWriteRowRequest indicates the request information of the table to be written in the BatchWriteRow operation.

Data structure

```
message TableInBatchWriteRowRequest {
    required string table_name = 1;
    repeated RowInBatchWriteRowRequest rows = 2;
}
```

table_name:

- Type: string
- The name of the table.

rows:

- Type: repeated RowInBatchWriteRowRequest
- The information of the row to be inserted, updated, or deleted in the table.

Related operations

BatchWriteRow

2.37. TableInBatchWriteRowResponse

TableInBatchWriteRowResponse indicates the write results for a table in the BatchWriteRow operation.

Data structure

```
message TableInBatchWriteRowResponse {
    required string table_name = 1;
    repeated RowInBatchWriteRowResponse put_rows = 2;
    repeated RowInBatchWriteRowResponse update_rows = 3;
    repeated RowInBatchWriteRowResponse delete_rows = 4;
}
```

table_name:

- Type: string
- The name of the table.

put_rows:

- Type: RowInBatchWriteRowResponse
- The results of the PutRow operation for the table.

update_rows:

- Type: RowInBatchWriteRowResponse
- The results of the UpdateRow operation for the table.

delete_rows:

- Type: RowInBatchWriteRowResponse
- The results of the DeleteRow operation for the table.

Related operations

BatchWriteRow

2.38. TableMeta

TableMeta indicates the structure information of a table.

Data structure

```
message TableMeta {
    required string table_name = 1;
    repeated PrimaryKeySchema primary_key = 2;
}
```

table_name:

- Type: string
- The name of the table.

primary_key:

- Type: repeated PrimaryKeySchema
- All primary key columns for the table.

Related operations

CreateTable

DescribeTable

2.39. TableOptions

TableOptions indicates the table parameters, including TimeToLive and MaxVersions.

Data structure

```
message TableOptions {
    optional int32 time_to_live = 1; // It can be dynamically modified
    optional int32 max_versions = 2; // It can be dynamically modified
    optional int64 deviation_cell_version_in_sec = 5; // It can be dynamically modified
}
```

time_to_live:

- Type: int32
- The retention time of data stored in this table (unit: second).

max_versions:

- Type: int32
- The maximum number of versions stored in this table.

deviation_cell_version_in_sec:

- Type: int64
- The maximum version deviation. This parameter is used to forbid writing data that deviates from the expected output. For example, if the value of deviation_cell_version_in_sec is 1000 and the current time stamp is 10000, the allowed time stamp range to be written is [10000 - 1000, 10000 + 1000].

2.40. TimeRange

TimeRange specifies the timestamp range or timestamp value to query.

Data structure

```
message TimeRange {
    optional int64 start_time = 1;
    optional int64 end_time = 2;
    optional int64 specific_time = 3;
}
```

start_time:

- Type: int64
- Description: the starting timestamp. Unit: milliseconds. Valid values: [0, INT64.MAX).

end_time:

- Type: int64
- Description: the ending timestamp. Unit: milliseconds. Valid values: [0, INT64.MAX).

specific_time:

- Type: int64
- Description: the specific timestamp. You can set one of specific_time and [start_time, end_time). Unit: milliseconds. Valid values: [0, INT64.MAX).

> **Note** [start_time, end_time) indicates a time range that starts from the start timestamp and ends at the end timestamp. The range includes the start timestamp and excludes the end timestamp.

2.41. TimeseriesTableMeta

TimeseriesTableMeta specifies the schema and configurations of a time series table.

Data structure

```
message TimeseriesTableMeta {
  required string table_name = 1;
  optional TimeseriesTableOptions table_options = 2;
  optional string status = 3;
}
```

Parameter	Type	Required	Description
table_name	string	Yes	The name of the time series table.
table_options	TimeseriesTableOptions	No	The configurations of the time series table.
status	string	No	The status of the time series table.

2.42. TimeseriesTableOptions

TimeseriesTableOptions specifies the configurations of a time series table.

Data structure

```
message TimeseriesTableOptions {
  optional int32 time_to_live = 1;
}
```

Parameter	Type	Required	Description
time_to_live	int32	Yes	The time to live (TTL) of data in the time series table. Unit: seconds.

2.43. TimeseriesRows

TimeseriesRows specifies multiple rows of time series data.

Data structure

```
message TimeseriesRows {
  required RowsSerializeType type = 1;
  required bytes rows_data = 2;
  optional int32 flatbuffer_crc32c = 3;
}
```

Parameter	Type	Required	Description
type	RowsSerializeType	Yes	The serialization type of the time series data.
rows_data	bytes	Yes	The serialized data.
flatbuffer_crc32c	int32	No	The CRC32C checksum value for serialization by using FlatBuffer.

2.44. RowsSerializeType

RowsSerializeType specifies the serialization type for rows of time series data.

Enumerated data structures

```
enum RowsSerializeType {
  RST_FLAT_BUFFER = 0;
}
```

RST_FLAT_BUFFER: serialization by using FlatBuffer.

2.45. MetaUpdateMode

MetaUpdateMode specifies the update mode for time series metadata.

Enumerated data types

```
enum MetaUpdateMode {
  MUM_NORMAL = 0;
  MUM_IGNORE = 1;
}
```

- MUM_NORMAL: normal mode.
- MUM_IGNORE: does not update time series metadata.

2.46. FailedRowInfo

FailedRowInfo specifies information about the row that failed to be written.

Data structure

```
message FailedRowInfo {
  required int32 row_index = 1;
  optional string error_code = 2;
  optional string error_message = 3;
}
```

Parameter	Type	Required	Description
row_index	int32	Yes	The sequence number of the row in the request.
error_code	string	No	The error code.
error_message	string	No	The error message.

2.47. MetaUpdateStatus

MetaUpdateStatus specifies the update status of the metadata.

Data structure

```
message MetaUpdateStatus {
  repeated uint32 row_ids = 1;
  repeated uint32 meta_update_times = 2;
}
```

Parameter	Type	Required	Description
row_ids	uint32	Yes	The sequence number of the row in the request.
meta_update_times	uint32	Yes	The point in time at which the metadata of the specified row is updated.

2.48. TimeseriesKey

TimeseriesKey specifies the identifier of the time series.

Data structure

```
message TimeseriesKey {
  required string measurement = 1;
  required string source = 2;
  required string tags = 3;
}
```

Parameter	Type	Required	Description
measurement	string	Yes	The name of the metric.
source	string	Yes	The identifier of the data source.
tags	string	Yes	The tags.

2.49. TimeseriesMeta

TimeseriesMeta specifies the time series metadata.

Data structure

```
message TimeseriesMeta {
  required TimeseriesKey time_series_key = 1;
  optional string attributes = 2;
  optional int64 update_time = 3;
}
```

Parameter	Type	Required	Description
time_series_key	TimeseriesKey	Yes	The identifier of the time series.
attributes	string	No	The properties of the time series metadata.
update_time	int64	No	The point in time at which the time series metadata is updated.

2.50. MetaQueryCondition

MetaQueryCondition specifies the condition that is used to retrieve the time series metadata.

Data structure

```
message MetaQueryCondition {
  required MetaQueryConditionType type = 1;
  required bytes proto_data = 2;
}
```

Parameter	Type	Required	Description
type	MetaQueryConditionType	Yes	The type of the retrieval condition.
proto_data	bytes	Yes	The serialization result of the retrieval condition.

2.51. MetaQueryConditionType

MetaQueryConditionType specifies the type of the condition that is used to retrieve the time series metadata.

Enumerated data structures

```
enum MetaQueryConditionType {
  COMPOSITE_CONDITION = 1;
  MEASUREMENT_CONDITION = 2;
  SOURCE_CONDITION = 3;
  TAG_CONDITION = 4;
  UPDATE_TIME_CONDITION = 5;
  ATTRIBUTE_CONDITION = 6;
}
```

- COMPOSITE_CONDITION: composite condition.
- MEASUREMENT_CONDITION: metric name condition.
- SOURCE_CONDITION: data source condition.
- TAG_CONDITION: tag condition.
- UPDATE_TIME_CONDITION: update time condition.
- ATTRIBUTE_CONDITION: property condition.

2.52. TimeseriesFieldsToGet

TimeseriesFieldsToGet specifies the name and data type of the time series data column that you want to query.

Data structure

```
message TimeseriesFieldsToGet {
  optional string name = 1;
  optional int32 type = 2;
}
```

Parameter	Type	Required	Description
name	string	Yes	The name of the column.

Parameter	Type	Required	Description
type	int32	Yes	The ID of the data type. A value of 1 indicates the LONG type. A value of 2 indicates the BOOLEAN type. A value of 3 indicates the DOUBLE type. A value of 4 indicates the STRING type. A value of 5 indicates the BINARY type.

3.Error codes

This topic describes common errors that are reported by Tablestore, including permission verification errors, HTTP message errors, and API operation errors. You can fix the errors based on the error messages.

> Note

If the value in the Retry column is Yes, you can retry the operation to fix the error. Otherwise, you cannot retry the operation to fix the error.

Permission verification errors

HTTP status code	Error code	Error message	Description	Retry
403	OTSAuthFailed	The AccessKeyID does not exist.	The error message returned because the AccessKey ID does not exist.	No
403	OTSAuthFailed	The AccessKeyID is disabled.	The error message returned because the AccessKey ID is disabled.	No
403	OTSAuthFailed	The user does not exist.	The error message returned because the specified user does not exist.	No
403	OTSAuthFailed	The instance is not found.	The error message returned because the specified instance does not exist.	No
403	OTSAuthFailed	The user has no privilege to access the instance.	The error message returned because you do not have permissions to access the instance.	No

HTTP status code	Error code	Error message	Description	Retry
403	OTSAuthFailed	The instance is not running.	The error message returned because the instance is not in the Running state.	No
403	OTSAuthFailed	Signature mismatch.	The error message returned because the request signature that is calculated by Alibaba Cloud does not match the signature that you provided.	No
403	OTSAuthFailed	Mismatch between system time and x-ots-date: `{Date}`.	The error message returned because the difference between the server time and the time specified in the x-ots-date parameter in the request header is out of the specified range.	No

HTTP message errors

HTTP status code	Error code	Error message	Description	Retry
413	OTSRequestBodyTooLarge	The size of POST data is too large.	The error message returned because the size of data that is sent by using the POST request exceeds the specified threshold.	No

HTTP status code	Error code	Error message	Description	Retry
408	OTSRequestTimeout	Request timeout.	The error message returned because the request has timed out on the client side.	No
405	OTSMethodNotAllowed	OTSMethodNotAllowedOnly POST method for requests is supported.	The error message returned because the request method is not supported. Only POST requests are supported.	No
403	OTSAuthFailed	Mismatch between MD5 value of request body and x-ots-contentmd5 in header.	The error message returned because the MD5 value that is calculated by using the request body is different from the MD5 value that is specified in the x-ots-contentmd5 parameter in the request header.	No
400	OTSParameterInvalid	Missing header: `{HeaderName}` .	The error message returned because a required header is missing.	No
400	OTSParameterInvalid	Invalid date format: `{Date}` .	The error message returned because the format of the specified date is invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Unsupported operation: `{Operation}` .	The error message returned because the name of the operation in the request URL is invalid.	No
400	OTSParameterInvalid	Can not reserve read capacity unit on capacity.	The error message returned because reserved read/write throughput cannot be specified for tables in capacity instances.	No
400	OTSForbiddenUpdateCapacityUnit	Your instance is forbidden to update capacity unit.	The error message returned because reserved read/write throughput cannot be updated for tables in capacity instances.	No

API operation errors

HTTP status code	Error code	Error message	Description	Retry

HTTP status code	Error code	Error message	Description	Retry
500	OTSInternalServerError	Internal server error.	The error message returned because an internal error occurred. If the error persists after you retry the operation multiple times, submit a ticket.	Yes
403	OTSQuotaExhausted	Too frequent table operations.	The error message returned because the frequency at which operations are performed on the table exceeds the specified threshold.	Yes
403	OTSQuotaExhausted	Number of tables exceeded the quota.	The error message returned because the maximum number of tables has been reached.	No
400	OTSParameterInvalid	Invalid instance name: `{InstanceName}` .	The error message returned because the specified instance name is invalid.	No
400	OTSParameterInvalid	Invalid table name: `{TableName}` .	The error message returned because the specified table name is invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid column name: `{ColumnName}`.	The error message returned because the specified column name is invalid.	No
400	OTSParameterInvalid	`{ColumnType}` is an invalid type for the primary key.	The error message returned because the type of the primary key column is invalid.	No
400	OTSParameterInvalid	`{ColumnType}` is an invalid type for the attribute column.	The error message returned because the type of the attribute column is invalid.	No
400	OTSParameterInvalid	The number of primary key columns must be in range: `[1, {Limit}]`.	The error message returned because the number of primary key columns is 0 or exceeds the specified threshold.	No
400	OTSParameterInvalid	Value of column `{ColumnName}` must be UTF8 encoding.	The error message returned because the values of this column are not encoded in UTF-8.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	The length of attribute column: `{ColumnName}` exceeded the MaxLength: `{MaxSize} with Current Length: {CellSize}` .	The error message returned because the maximum length of the attribute column name has been reached.	No
400	OTSParameterInvalid	No row specified in the request of BatchGetRow.	The error message returned because no row is specified for the BatchGetRow operation.	No
400	OTSParameterInvalid	Duplicated table name: `{TableName}` .	The error message returned because tables with the same name are specified for the BatchGetRow or BatchWriteRow operation.	No
400	OTSParameterInvalid	Duplicated primary key name: `'{PKName}'` .	The error message returned because duplicate primary keys exist.	No
400	OTSParameterInvalid	The limit must be greater than 0.	The error message returned because the value specified for the limit parameter is less than 0. The value of the limit parameter must be greater than 0.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Duplicated attribute column name with primary key column: `{ColumnName}` .	The error message returned because the name of an attribute column is the same as the name of a primary key column in the row.	No
400	OTSParameterInvalid	Rows count exceeds the upper limit: `{limit}` .	The error message returned because the maximum number of rows that can be written into a data table has been reached.	No
400	OTSParameterInvalid	No version condition is specified while querying row.	The error message returned because the number of versions is not specified as a query condition.	No
400	OTSParameterInvalid	No row is specified in BatchWriteRow.	The error message returned because no row is specified for the BatchWriteRow operation. At least one row must be specified for the BatchWriteRow operation.	No
400	OTSParameterInvalid	No operation is specified for table: `{TableName}` .	The error message returned because no operation is specified for the table in the BatchWriteRow operation.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	The type of row in batch write operation is invalid.	The error message returned because the row type that is specified in the BatchWriteRow operation is invalid.	No
400	OTSParameterInvalid	The modify type is invalid.	The error message returned because the operation type that is specified in the BatchWriteRow operation is invalid. Only the PUT, UPDATE, and DELETE operation types are supported.	No
400	OTSParameterInvalid	The total data size of BatchWriteRow request exceeds the limit, limit size: `{LimitSize}`, data size: `{UserSize}`.	The error message returned because the maximum size of data allowed for the BatchWriteRow operation has been reached.	No
400	OTSParameterInvalid	Time-to-live is missing while creating table.	The error message returned because time to live (TTL) is not specified when the table is created.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	MaxVersions is missing while creating table.	The error message returned because no value is specified for the max versions parameter when the table is created.	No
400	OTSParameterInvalid	Start and end time must be given at the same time.	The error message returned because the StartTime and EndTime parameters must be configured in pairs for the GetRange operation.	No
400	OTSParameterInvalid	Invalid column type, only STRING,INTEGER,BINARY is allowed.	The error message returned because the specified column type is invalid. The primary key columns support only the STRING, INTEGER, and BINARY types.	No
400	OTSParameterInvalid	Invalid column type: `{ColumnType}` .	The error message returned because the column type is invalid.	No
400	OTSParameterInvalid	Invalid return type: `{ReturnType}` .	The error message returned because the return type is invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid condition: `{Condition}` .	The error message returned because the specified condition is invalid.	No
400	OTSParameterInvalid	The value of read capacity unit can not be less than `{Limit}` .	The error message returned because the read capacity unit (CU) value is less than the lower limit.	No
400	OTSParameterInvalid	The value of write capacity unit can not be less then `{Limit}` .	The error message returned because the write CU value is less than the lower limit.	No
400	OTSParameterInvalid	AUTO_INCREMENT primary key must be integer.	The error message returned because only integer columns can be set to auto-increment primary key columns.	No
400	OTSParameterInvalid	AUTO_INCREMENT primary key count must <= 1.	The error message returned because multiple auto-increment primary key columns are specified. You can specify only one auto-increment primary key column.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Column value cannot be given when type is DELETE_ONE_VERSION,DELETE_ALL_VERSION.	The error message returned because attribute column values cannot be specified for delete operations, such as DELETE_ONE_VERSION or DELETE_ALL_VERSION.	No
400	OTSParameterInvalid	Timestamp must be given when type is DELETE_ONE_VERSION.	The error message returned because no version number is specified. To delete a version, you must specify a version number.	No
400	OTSParameterInvalid	Timestamp cannot be given when type is DELETE_ALL_VERSION.	The error message returned because a version number is specified. To delete all versions, you cannot specify a version number.	No
400	OTSParameterInvalid	No name is given for attribute column.	The error message returned because no attribute column name is specified.	No
400	OTSParameterInvalid	No value is given for column name.	The error message returned because no attribute column value is specified.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	OpType cannot be given for column name: `{ColumnName}` in PutRow.	The error message returned because an operation type is specified for PutRow. You can specify an operation type only for UpdateRow.	No
400	OTSParameterInvalid	Attribute column is missing.	The error message returned because no attribute column is specified.	No
400	OTSParameterInvalid	The number of attribute columns exceeds the limit, limit count: `{Limit}` , column count: `{Count}` .	The error message returned because the number of attribute columns exceeds the value specified for the limit parameter.	No
400	OTSParameterInvalid	The length of primary key column: `{ColumnName}` exceeds the MaxLength: `{Limit}` with CurrentLength: `{Current}` .	The error message returned because the maximum length of the primary key has been reached.	No
400	OTSParameterInvalid	No name is given for primary key.	The error message returned because no name is specified for the primary key column.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	No value is given for primary key name: `{PkName}`.	The error message returned because no value is specified for the primary key column.	No
400	OTSParameterInvalid	OpType cannot be given for primary key name : `{PkName}`.	The error message returned because an operation type is specified for the primary key. You cannot specify an operation type for the primary key.	No
400	OTSParameterInvalid	Timestamp cannot be given for primary key name: `{PkName}`.	The error message returned because the timestamp is specified for the primary key. You cannot specify the timestamp for the primary key.	No
400	OTSParameterInvalid	Duplicated primary key name: `{PkName}`.	The error message returned because duplicate primary keys exist.	No
400	OTSParameterInvalid	Attribute column cannot be given while reading data.	The error message returned because the value of the attribute column does not need to be specified for a read operation.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Delete marker cannot be given.	The error message returned because the operation does not support the DeleteMarker parameter. Only delete operations support the DeleteMarker parameter.	No
400	OTSParameterInvalid	The number of columns from the request exceeds the limit, limit count: `{Limit}`, column count: `{current}`.	The error message returned because the maximum number of columns allowed for the request has been reached.	No
400	OTSParameterInvalid	Timestamp must be in range [0, INT64_MAX/1000).	The error message returned because the timestamp must be greater than or equal to 0 and less than the value specified by INT64_MAX/1000.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	TimeToLive cannot be 0 or less than -1.	The error message returned because the value of TTL cannot be set to 0 or a value less than -1. However, you can set the value of TTL to -1.	No
400	OTSParameterInvalid	The maximum versions cannot be less than or equal to 0.	The error message returned because the value specified for the max versions parameter cannot be less than or equal to 0.	No
400	OTSParameterInvalid	Specific timestamp cannot be less than 0.	The error message returned because the specified timestamp is less than 0.	No
400	OTSParameterInvalid	The maximum deviation must be in range [0, INT64_MAX/1000000].	The error message returned because the value of the max version offset parameter is out of the specified range.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Deserialize filter failed.	The error message returned because filter deserialization failed. The filter format encapsulated by the SDK is invalid.	No
400	OTSParameterInvalid	Offset in ColumnPagination Filter must be greater than or equal to 0.	The error message returned because the offset value of ColumnPagination Filter is less than 0. The offset value of ColumnPagination Filter must be greater than or equal to 0.	No
400	OTSParameterInvalid	Limit in ColumnPagination Filter must be greater than 0.	The error message returned because the Limit value of ColumnPagination Filter is less than 0. The Limit value of ColumnPagination Filter must be greater than or equal to 0.	No
400	OTSParameterInvalid	Deserialize relation filter failed.	The error message returned because deserialization for the single column filter failed. The filter format encapsulated by the SDK may be invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Deserialize composite filter failed.	The error message returned because deserialization for the combined filter failed. The filter format encapsulated by the SDK may be invalid.	No
400	OTSParameterInvalid	Deserialize column pagination filter failed.	The error message returned because deserialization for the wide row filter failed. The filter format encapsulated by the SDK may be invalid.	No
400	OTSParameterInvalid	The count of filter exceeds the max: `{Limit}`.	The error message returned because the maximum number of filters has been reached.	No
400	OTSParameterInvalid	Specific tiemstamp and max versions cannot be given at the same time.	The error message returned because a timestamp and the max versions parameter cannot be specified at the same time.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Specific tiemstamp and time range cannot be given at the same time.	The error message returned a timestamp and a timestamp range cannot be specified at the same time.	No
400	OTSParameterInvalid	Specific tiemstamp, time range and max versions cannot be given at the same time.	The error message returned because a timestamp, a timestamp range, and the max versions parameter cannot be specified at the same time. Only one element can be specified.	No
400	OTSParameterInvalid	Duplicated attribute column name with primary key column: `{PkName}` .	The error message returned because the name of the attribute column is the same as the name of the primary key column.	No
400	OTSParameterInvalid	The total data size of PutRow request exceeds the limit, limit size: `{Limit}` , data size: `{Current}` .	The error message returned because the maximum size of data allowed for the PutRow operation has been reached.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	The total data size of UpdateRow request exceeds the limit, limit size: `{Limit}`, data size: `{Current}`.	The error message returned because the maximum size of data allowed for the UpdateRow operation has been reached.	No
400	OTSParameterInvalid	No parameter is specified in table options.	The error message returned because no parameter is specified in TableOption.	No
400	OTSParameterInvalid	Input encoded PK broken, invalidate cell key type:xx.	The error message returned because the primary key that is encoded in the PlainBuffer format is invalid. The cell type is invalid.	No
400	OTSParameterInvalid	Input encoded PK broken, length is shorter than expect.	The error message returned because the primary key that is encoded in the PlainBuffer format is invalid. The length of the primary key is less than expected.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	PKAutoIncr can't be used for read operations.	The error message returned because the auto-increment primary key column attribute is specified for the read operation. You cannot specify the auto-increment primary key column attribute for read operations.	No
400	OTSParameterInvalid	Begin key must less than end key in FORWARD.	The error message returned because the value of the start key is not less than the value of the end key for forward query.	No
400	OTSParameterInvalid	Begin key must more than end key in BACKWARD.	The error message returned because the value of the start key is not greater than the value of the end key for backward query.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Repeated rows do not support row exist expectation check in BatchModify.	The error message returned because row existence check is not allowed for rows on which duplicate operations are performed in BatchWriteRow.	No
400	OTSParameterInvalid	Repeated rows do not support row condition check in BatchModify.	The error message returned because conditional update is not allowed for rows on which duplicate operations are performed in BatchWriteRow.	No
400	OTSParameterInvalid	PK column size not the same for all rows in table: `{TableName}` .	The error message returned because different rows have different numbers of primary key columns.	No
400	OTSParameterInvalid	Duplicate rows detected in MultiPut.	The error message returned because duplicate rows exist in MultiPut.	No
400	OTSParameterInvalid	Duplicate rows detected in MultiGet.	The error message returned because duplicate rows exist in MultiGet.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Column does not exist: `{ColumnName}`.	The error message returned because the specified column does not exist.	No
400	OTSParameterInvalid	Invalid max scan size: `{Limit}`.	The error message returned because the value that is specified for the limit parameter for range query exceeds the threshold.	No
400	OTSParameterInvalid	data format is invalid.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid.	No
400	OTSParameterInvalid	double can not be used as primary key.	The error message returned because the double type cannot be used for primary keys.	No
400	OTSParameterInvalid	data format is invalid, unknown variant type occured.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The value type is invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Parse PBMessage from RawString failed.	The error message returned because deserialization by using Protobuf failed. The SDK serialization code may be invalid.	No
400	OTSParameterInvalid	Cell data broken, mismatch header, actual: `{Header}` , expect: `{Header}` .	The error message returned because the row data that is encoded in the PlainBuffer format is invalid due to header mismatch.	No
400	OTSParameterInvalid	Cell data broken, PK value cannot be NULL, name: `{PkName}` .	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The value of the primary key that corresponds to PkName is empty.	No
400	OTSParameterInvalid	Cell data broken, PK is empty.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The primary key is not specified.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Cell data broken, no PK follow PKTag.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The primary key does not follow the primary key tag.	No
400	OTSParameterInvalid	Cell data broken, attr has no name.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The attribute column name is not specified.	No
400	OTSParameterInvalid	Cell data broken, attr has no value, name: `{Name}` .	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The attribute column value is not specified.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Cell data broken, no Cells follow CellTag.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The cell content does not follow the cell tag.	No
400	OTSParameterInvalid	Cell data broken, no valid element in Cell, name: `{Name}`.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid due to different cell formats.	No
400	OTSParameterInvalid	Cell data broken, cell is not end with checksum[`{0x0A}`]"] tag.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The cell does not end with CheckSumFlag after cell encoding is complete.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Cell data broken, row is not end with checksum tag.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The row data does not end with CheckSumTag after data encoding is complete.	No
400	OTSParameterInvalid	Cell data broken, mismatch tag, actual tag(`{TAG}`), expect(`{TAG}`).	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The specified tag does not exist.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Cell data broken, wrong string format, size: `{Size}`.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The encoding format of the string is invalid.	No
400	OTSParameterInvalid	Invalid Capacity Unit for UpdateTable: Either read(`{CU}`) or write(`{CU}`) should be non-negative.	The error message returned because the number of read CUs or write CUs is less than 0. The number of read CUs and the number of write CUs must both be greater than or equal to 0.	No
400	OTSParameterInvalid	Auto increment pk must be integer, name: `{PkName}`.	The error message returned because the auto-incremental primary key column is not of the integer type.	No
400	OTSParameterInvalid	Cannot deserialize the request data.	The error message returned because the deserialization of data in the request failed due to the empty value or invalid format.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid row: missing checksum.	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The checksum value of the row is not specified.	No
400	OTSParameterInvalid	Invalid Column(`{ColumnName}`): missing checksum.	The error message returned because the column data that is encoded in the PlainBuffer format is invalid. The checksum value of the cell is not specified.	No
400	OTSParameterInvalid	Cell data broken, checksum is mismatch, `{UserChecksum}` : `{SeviceChecksum}` .	The error message returned because the row data that is encoded in the PlainBuffer format is invalid. The value of UserChecksum is different from the value of SeviceChecksum that is calculated by the server.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Cell data broken, has more data in cell, but they can't be parsed.	The error message returned because the column data that is encoded in the PlainBuffer format is invalid due to excessive data.	No
400	OTSParameterInvalid	NaN can't be set to double value.	The error message returned because NaN is set to a value of the double type.	No
400	OTSParameterInvalid	Infinity can't be set to double value.	The error message returned because Infinity is set to a value of the double type.	No
400	OTSParameterInvalid	Invalid max versions: `{Version}`. Reason: Max versions must be positive.	The error message returned because the value specified for the max versions parameter is invalid. The value must be greater than 0 or equal to -1.	No
400	OTSParameterInvalid	invalid range, the begin's type is different from the end's, `{BeginType}`: `{EndType}`.	The error message returned because the type of the start key is different from that of the end key for range query.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid offset of Filter: `{Filter}` . Reason: Offset must be nonnegative.	The error message returned because the value of offset in the filter is less than 0. The value of offset in the filter must be greater than or equal to 0.	No
400	OTSParameterInvalid	Invalid Capacity Unit: `{CU}` . Capacity Unit must be nonnegative.	The error message returned because the value of CU is less than 0. The value of CU must be greater than or equal to 0.	No
400	OTSParameterInvalid	Unknown or unsupported CellType in primary key: `{PkName}` .	The error message returned because the cell type of the primary key cannot be identified.	No
400	OTSParameterInvalid	Invalid boolean real value, value length: `{Length}` .	The error message returned because the column data that is encoded in the PlainBuffer format is invalid. The length of the value of the Boolean type is invalid.	No
400	OTSParameterInvalid	invalid name of column: empty name.	The error message returned because the name of the attribute column is left empty.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid ModifyType: `{ModifyType}` .	The error message returned because the update type is invalid.	No
400	OTSParameterInvalid	Invalid relation operator: `{Operation}` .	The error message returned because the relational operator is invalid.	No
400	OTSParameterInvalid	Invalid type of filter: `{Type}` .	The error message returned because the filter type is invalid.	No
400	OTSParameterInvalid	Invalid NOT operator: the number of sub-filters must be 1.	The error message returned because more than one subfilter is specified for the NOT operator. The NOT operator supports only one subfilter.	No
400	OTSParameterInvalid	Invalid AND/OR operator: the number of sub-filters must be 2.	The error message returned because the number of specified subfilters is not two. You must specify two subfilters for the AND or OR operator.	No
400	OTSParameterInvalid	Invalid type of sub-filter: `{Filter}` .	The error message returned because the subfilter type is invalid.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	`{Count}` pk auto increment exist in row.	The error message returned because more than one auto-increment primary key columns exist.	No
400	OTSParameterInvalid	Invalid action: `{Action}` .	The error message returned because the operation type is invalid.	No
400	OTSParameterInvalid	Invalid row-existence expectation: `{Expectation}` .	The error message returned because the existence value of the specified row is invalid.	No
400	OTSParameterInvalid	Invalid row to delete which with pk auto increment.	The error message returned because the auto-increment primary key column attribute is specified for the delete operation. You cannot specify the auto-increment primary key column attribute for delete operations.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid expect: `{RowExpect}` when modify row with pk auto increment.	The error message returned because the existence value is invalid for the specified row with an auto-increment primary key column.	No
400	OTSParameterInvalid	Can't set condition when modify row with pk auto increment.	The error message returned because conditional update is configured for the row with an auto-increment primary key column. You cannot configure conditional update for rows with auto-increment primary key columns.	No
400	OTSParameterInvalid	Invalid expect: `{RowExpect}` when modify a specific row for table with pk auto increment.	The error message returned because the existence condition of the specified row that needs to be modified in the table with an auto-increment primary key column is invalid.	No
400	OTSParameterInvalid	Invalid cell: `{Cell}`. missing timestamp.	The error message returned because the timestamp is not specified.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid cell timestamp: `{Timestamp}`. Expect: [0, Version::kMax.mVersion / kUsecPerMsec].	The error message returned because the specified timestamp is invalid.	No
400	OTSParameterInvalid	Invalid timestamp. Cell: `{CellName}`.	The error message returned because data can only be written when the specified timestamp in milliseconds divided by 1000 is within the valid version range. The valid version range is calculated by using the following formula: Valid version range = [Data written time - Max version offset, Data written time + Max version offset).	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid timestamp for cell: `{Timestamp}`. timestamp is inapplicable to DELETE_ALL_VERSION.	The error message returned because a version is specified when DELETE_ALL_VERSION is used to delete all versions. You cannot specify a version when DELETE_ALL_VERSION is used to delete all versions.	No
400	OTSParameterInvalid	Invalid op type of cell: `{OpType}`.	The error message returned because the operation type is invalid.	No
400	OTSParameterInvalid	Invalid cell: `{Cell}`. Reason: missing value.	The error message returned because the value of the cell is not specified.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid request of delete row: missing RowDeleteMarker in request.	The error message returned because RowDeleteMarker is not contained in the delete request.	No
400	OTSParameterInvalid	Invalid request of put/update row: unexpected RowDeleteMarker in request.	The error message returned because RowDeleteMarker is contained in PutRow or UpdateRow. RowDeleteMarker cannot be contained in PutRow or UpdateRow.	No
400	OTSParameterInvalid	Invalid update row request: missing cells in request.	The error message returned because values of the attribute columns are not specified in UpdateRow.	No
400	OTSParameterInvalid	Invalid delete row request: unexpected cells in request.	The error message returned because the value of an attribute column is specified in DeleteRow. The values of the attribute columns cannot be specified in DeleteRow.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	Invalid request of put row: find cells without values.	The error message returned because values of the attribute columns are not specified in PutRow.	No
400	OTSParameterInvalid	Can not reserve read capacity unit on hybrid storage cluster: `{TableName}` .	The error message returned because reserved read CUs are specified for capacity instances. Reserved read CUs cannot be specified for capacity instances.	No
400	OTSParameterInvalid	Can not reserve write capacity unit on hybrid storage cluster: `{TableName}` .	The error message returned because reserved write CUs are specified for capacity instances. Reserved write CUs cannot be specified for capacity instances.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	First primary key can't be auto increment, table: `{TableName}` .	The error message returned because the first primary key column is set to an auto-increment primary key column. The first primary key column cannot be set to an auto-increment primary key column.	No
400	OTSParameterInvalid	PkAutoIncr cannot be set as first pk column.	The error message returned because the auto-increment primary key column is set as the first primary key column. The auto-increment primary key column cannot be set as the first primary key column.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSParameterInvalid	the count of auto-incremental primary keys can not be more than 1.	The error message returned because multiple auto-increment primary key columns are specified. You can specify only one auto-increment primary key column.	No
400	OTSParameterInvalid	Can't write index table directly.	The error message returned because write operations cannot be performed on index tables.	No
400	OTSParameterInvalid	Try to call method using explicit transaction on explicit-transaction-disabled table.	The error message returned because local transaction is disabled for the table.	No
400	OTSParameterInvalid	Table[tablename] who has at least one search index cannot be deleted.	The error message returned because tables with indexes cannot be deleted.	No
400	OTSParameterInvalid	nested field aggregation/groupby must not be recursive.	The error message returned because recursive GroupBy or Aggregation cannot be used in nested fields.	No

Tablestore errors

HTTP status code	Error code	Error message	Description	Retry
503	OTSServerBusy	Server is busy.	The error message returned because an internal server of Tablestore is busy. Try again later.	Yes
503	OTSPartitionUnavailable	The partition is not available.	The error message returned because the partition is being loaded. Try again later.	Yes
503	OTSTimeout	Operation timeout.	The error message returned because an internal operation of Tablestore has timed out.	Yes
503	OTSServerUnavailable	Server is not available.	The error message returned because an internal server of Tablestore is inaccessible.	Yes
409	OTSRowOperationConflict	Data is being modified by the other request.	The error message returned because concurrent requests to write data in the same row cause a conflict.	Yes
409	OTSObjectAlreadyExist	Requested table already exists.	The error message returned because the data table or index table that you want to create already exists.	No

HTTP status code	Error code	Error message	Description	Retry
404	OTSObjectNotExist	Requested table does not exist.	The error message returned because the requested table does not exist.	No
404	OTSTableNotReady	The table is not ready.	The error message returned because the table is being initialized. Wait for a few seconds and try again.	Yes
403	OTSTooFrequentReservedThroughputAdjustment	Capacity unit adjustment is too frequent.	The error message returned because the read/write CUs are frequently adjusted.	Yes
403	OTSCapacityUnitExhausted	Capacity unit is not enough.	The error message returned because read/write CUs for the partition are exhausted.	No
403	OTSConditionCheckFail	Condition check failed.	The error message returned because condition check failed.	No

HTTP status code	Error code	Error message	Description	Retry
400	OTSOutOfRowSizeLimit	The total data size of columns in one row exceeded the limit.	The error message returned because the maximum size of data in all columns of the row has been reached.	No
400	OTSOutOfColumnCountLimit	The number of columns in one row exceeded the limit.	The error message returned because the maximum number of columns in the row has been reached.	No
400	OTSInvalidPK	Primary key schema mismatch.	The error message returned because the specified primary key does not match the primary key of the data table.	No