Table Store

SDK Reference

SDK Reference

Java-SDK

Introduction

This document describes how to install and use the Java SDK (version 2.2.4 and later) for Table Store (formerly called OTS). This document assumes that you have activated Alibaba Cloud Table Store and created an AccessKeyID and AccessKeySecret.

> If you have not activated Table Store or want to know about the service, visit the Table Store product homepage.
>
> If you have not created an AccessKeyID and AccessKeySecret, log on to the Alibaba Cloud Access Key Console to create an access key.

Download the SDK package

> SDK package: aliyun_tablestore_java_sdk_2.2.5.zip
>
> GitHub：https://github.com/aliyun/aliyun-tablestore-java-sdk

For details about version iterations, click here.

Version

Latest version: 2.2.5

Compatibility

For the 2.x.x series SDK:

- Compatible

Change description

- A bug in OTSWriter that may cause program hang is fixed.

Environment preparation

- Applicable to JDK 6 and later.

Installation methods

Install Java SDK by using Maven

To use the Table Store Java SDK in Maven, you only need to add the corresponding dependency to the pom.xml file. Let’s take Java SDK 2.2.5 as an example. Type the following content in the “dependency” section:

```
<dependency>
<groupId>com.aliyun.openservices</groupId>
<artifactId>ots-public</artifactId>
<version>2.2.5</version>
</dependency>
```

Install Java SDK by importing the JAR package to Eclipse

Let’s take Java SDK 2.2.5 as an example. The process is as follows:

Download the Java SDK at aliyun_tablestore_java_sdk_2.2.5.zip.

Decompress the SDK.

Copy the file ots-public-<versionId>.jar in the decompressed folder and all files in the lib folder to your project.

In Eclipse, right-click your project and choose **Properties** > **Java Build Path** > **Add JARs** from the context menu.

Select all JAR files that you have copied in Step 3.

After completing the preceding steps, you can use the Table Store Java SDK in Eclipse.

Sample programs

The Table Store Java SDK provides diverse sample programs for your reference or use. You can get the sample programs as follows:

Download and decompress the Table Store Java SDK, and find the sample programs in the examples folder.

Access the GitHub project for the Table Store Java SDK at aliyun-tablestore-java-sdk.

OTSClient is the client for Table Store, providing callers with a series of methods for operating tables and reading/writing data from/to a single row or multiple rows. To use the Java SDK to initiate a request to Table Store, you need to initialize an OTSClient instance and modify the default configurations of ClientConfiguration based on your needs.

Determine an endpoint

An endpoint is the domain of Alibaba Cloud Table Store in a region. It supports the following format:

Example	Description
http://sun.cn-hangzhou.ots.aliyuncs.com	Accesses the sun instance in Hangzhou over the Internet using the HTTP protocol.
https://sun.cn-hangzhou.ots.aliyuncs.com	Accesses the sun instance in Hangzhou over the Internet using the HTTPS protocol.

TIPS:

Instances can also be accessed over the intranet.

You can log on to the Alibaba Cloud Table Store Console, go to the instance overview page, and locate the instance access address, which is the endpoint of the instance.

Configure an access key

To access the Alibaba Cloud Table Store service, you need a valid access key for signature authentication. The following types of access keys are supported:

AccessKeyID and AccessKeySecret of the primary account. The creation process is as follows:

Register an Alibaba Cloud account on the Alibaba Cloud website.

Log on to the AccessKey Console to apply for an access key.

AccessKeyID and AccessKeySecret of the sub-account authorized to access Table Store. The creation process is as follows:

Use the primary account to access RAM and create a sub-account or use an existing sub-account.

Use the primary account to authorize the sub-account to access Table Store.

After authorization, the AccessKeyID and AccessKeySecret of the sub-account can be used to access Table Store.

STS token for temporary access. The token acquisition process is as follows:

The application server accesses the RAM/STS server to obtain a temporary AccessKeyID, AccessKeySecret and token, and sends them to the user.

The user uses the temporary AccessKeyID, AccessKeySecret and token to access Table Store.

Initialization

After you obtain the AccessKeyID and AccessKeySecret, do the following to initialize an OTSClient instance:

Use the endpoint of the Table Store service to create a client. The API is as follows:

```
/**
* Use the specified Table Store endpoint and the default configurations to construct an instance.
*
* NOTE:
* 1. In most cases, you only need to create an OTSClient object globally (for thread security). You do not need to create an OTSClient object at each time of access.
* In the case of high concurrency (tens of thousands of QPSs), you can use multiple OTSClient objects for performance testing and tuning, but the number of OTSClient objects must be controlled within the proper range.
* 2. Each OTSClient occupies threads and connection resources. You can configure the number of
```

```
threads and connections in ClientConfiguration.
* 3. After use, call the shutdown method to release the threads and connection resources occupied by
the OTSClient.
*
* @param endpoint Endpoint of the TableStore service.
* @param accessKeyId AccessKeyID used to access the Table Store service.
* @param accessKeySecret AccessKeySecret used to access the Table Store service.
*/
public OTSClient(String endpoint, String accessKeyId,
String accessKeySecret, String instanceName);
```

Example of use:

```
    final String endPoint = "<your endpoint>";
final String accessId = "<your access id>";
final String accessKey = "<your access key>";
final String instanceName = "<your instance name>";

// Create an OTSClient object
OTSClient client = new OTSClient(endPoint, accessId, accessKey, instanceName);

// Use the client to read/write data from/to Table Store

// Close the client
client.shutdown();
```

TIPS:

In addition to the AccessKeyID and AccessKeySecret, the STS token can also be used to access Table Store.

- When you create an OTSClient instance, you can set parameters such as the timeout time, the maximum number of connections, and the maximum retry times in ClientConfiguration.

Configure the OTSClient.

To modify the default configurations of the OTSClient, you can import the ClientConfiguration instance when constructing the OTSClient. ClientConfiguration is a configuration-class OTSClient instance, allowing you to configure a proxy, the connection timeout time, the maximum number of connections, and other parameters. ClientConfiguration supports the following parameter settings:

Parameter	Description	Default value	Method
MaxConnections	Maximum number of HTTP connections that can be enabled.	300	ClientConfiguration.setMaxConnections
SocketTimeoutInMillisecond	Timeout time (in ms) for data transmission at the	15 × 1000	ClientConfiguration.setSocketTimeoutInMillisecond

	Socket layer. The value 0 indicates an infinite wait period.		
ConnectionTimeoutInMillisecond	Timeout time (in ms) for connection setup. The value 0 indicates an infinite wait period.	15 × 1000	ClientConfiguration.setConnectionTimeoutInMillisecond
RetryThreadCount	Number of threads in the thread pool used for retry when an error occurs.	1	ClientConfiguration.setRetryThreadCount
IoThreadCoun	Number of threads in IOReactor of the HttpAsyncClient.	CPU count	ClientConfiguration.setIoThreadCount
ProxyPort	Port on the proxy.	null	public void setProxyPort (int proxyPort)
ProxyHost	Host address of the proxy.	null	ClientConfiguration.setProxyHost
ProxyUsername	Username verified by the proxy.	null	ClientConfiguration.setProxyUsername
ProxyPassword	Password verified by the proxy.	null	ClientConfiguration.setProxyPassword
ProxyDomain	Windows domain name used to access the NTLM-verified proxy.	null	ClientConfiguration.setProxyDomain
ProxyWorkstation	Name of the Windows workstation for the NTLM proxy.	null	ClientConfiguration.setProxyWorkstation
UserAgent	User proxy, that is, the HTTP User-Agent header.	aliyun-tablestore-sdk-java	ClientConfiguration.setUserAgent

HTTPS

- Upgrade to Java 7 for HTTPS.

Multithreading

Multithreading is supported.

It is recommended that multiple threads use the same OTSClient object.

The Table Store SDK provides the following table-level operation interfaces: CreateTable, ListTable, DeleteTable, UpdateTable, and DescribeTable.

CreateTable

Creates a table based on the given table structure information.

When creating a table in Table Store, you must specify the table’ s Primary Keys. A Primary Key contains one to four Primary Key columns, and each Primary Key column has a name and a type.

API

```
    /**
* Create a table.
*
* @param createTableRequest Encapsulate the parameters required to perform the CreateTable operation.
* @return Content of the response to the CreateTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public CreateTableResult createTable(CreateTableRequest createTableRequest)
throws OTSException, ClientException;
```

TIPS:

- After a table is created in Table Store, it takes several seconds to load the table. Any read/write operations performed on the table during the loading process fail.

Example

Create a table with two Primary Key columns and a reserved read/write throughput of (0,0).

```
        //Create a schema for the Primary Key columns, including the quantity, names, and types of Primary Keys
//The first Primary Key column (sharding column) is named pk0 and belongs to the integer type.
//The second Primary Key column is named pk1 and belongs to the integer type.

TableMeta tableMeta = new TableMeta("SampleTable");
tableMeta.addPrimaryKeyColumn("pk0", PrimaryKeyType.INTEGER);
tableMeta.addPrimaryKeyColumn("pk1", PrimaryKeyType.INTEGER);

// Set the read and write CUs of the table to 0
CapacityUnit capacityUnit = new CapacityUnit(0, 0);

try
{
```

```
// Construct the CreateTableRequest object
CreateTableRequest request = new CreateTableRequest();

request.setTableMeta(tableMeta);
request.setReservedThroughput(capacityUnit);

// Call the CreateTable interface of the client. The table is created successfully if no exception is thrown. If an
exception is thrown, the table fails to be created.
client.createTable(request);

System.out.println("Create table succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Create table failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Create table failed.");
}
```

TIPS:

- Code details: createTable@GitHub.

ListTable

Obtain the names of all tables under the current instance.

API

```
    /**
* Return a list of table names.
*
* @return Content of the response to the ListTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public ListTableResult listTable() throws OTSException, ClientException;
```

Example

Obtain the names of all tables under an instance.

```
    try
{
// Call the ListTable interface
ListTableResult result = client.listTable();

// Print the table names
for (String tableName : result.getTableNames()) {
```

```
System.out.println(tableName);
}

System.out.println("List table succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("List table failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("List table failed.");
}
```

TIPS:

- Code details: listTable@GitHub.

UpdateTable

Update the reserved read/write throughput value of the specified table.

API

```
    /**
* Update the read/write CapacityUnit of a table.
*
* @param updateTableRequest Encapsulate the parameters required to perform the UpdateTable operation.
* @return UpdateTable Content of the response to the UpdateTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public UpdateTableResult updateTable(UpdateTableRequest updateTableRequest)
throws OTSException, ClientException;
```

Example

Update the read CU and write CU of a table to 1 and 2 respectively.

```
    // Update the reserved read throughput to 1 and the reserved write throughput to 2
ReservedThroughputChange cuChange = new ReservedThroughputChange();

// Do not set the read CU if you only need to adjust the write CU
cuChange.setReadCapacityUnit(1);

// Do not set the write CU if you only need to adjust the read CU
cuChange.setWriteCapacityUnit(2);

// Construct the UpdateTableRequest object
UpdateTableRequest request = new UpdateTableRequest();
request.setTableName("SampleTable");
```

```
equest.setReservedThroughputChange(reservedThroughput);

try
{
// Call the interface to update the reserved read/write throughput of the table
client.updateTable(request);

// Execution is successful if no exception is thrown.
System.out.println("Update table succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Update table failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Update table failed.");
}
```

TIPS:

- Code details: updateTable@GitHub.

DescribeTable

Query the structure information and the reserved read/write throughput value of the specified table.

API

```
    /**
* Return the table structure information.
*
* @param describeTableRequest Encapsulate the parameters required to perform the DescribeTable operation.
* @return DescribeTable Content of the response to the DescribeTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public DescribeTableResult describeTable(DescribeTableRequest describeTableRequest) throws OTSException,
ClientException;
```

Example

Obtain the descriptive information of a table.

```
    try
{
// Construct a request object
DescribeTableRequest request = new DescribeTableRequest();
request.setTableName("SampleTable");

DescribeTableResult result = client.describeTable(request);
```

```
// Print the descriptive information of SampleTable
TableMeta tableMeta = result.getTableMeta();
System.out.println("Table name : " + tableMeta.getTableName());
System.out.println("Table Pk : ");
for (String keyName : tableMeta.getPrimaryKey().keySet()) {
System.out.println(keyName + " : " + tableMeta.getPrimaryKey().get(keyName));
}
ReservedThroughputDetails reservedThroughputDetails = result.getReservedThroughputDetails();
System.out.println("Table read CU : "
+ reservedThroughputDetails.getCapacityUnit().getReadCapacityUnit());
System.out.println("Table write CU : "
+ reservedThroughputDetails.getCapacityUnit().getWriteCapacityUnit());
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Descibe table failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Describe row failed.");
}
```

TIPS:

- Code details: describeTable@GitHub.

DeleteTable

Delete the specified table under an instance.

API

```
    /**
* Delete a table.
*
* @param deleteTableRequest Encapsulate the parameters required to perform the DeleteTable operation.
* @return DeleteTable Content of the response to the DeleteTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public DeleteTableResult deleteTable(DeleteTableRequest deleteTableRequest)
throws OTSException, ClientException;
```

Example

Delete a table.

```
    DeleteTableRequest request = new DeleteTableRequest();
request.setTableName("SampleTable");
```

```
try
{
// Delete the table
client.deleteTable(request);

// The table is deleted successfully if no exception is thrown.
System.out.println("Delete table succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Delete table failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Delete table failed.");
}
```

TIPS:

- Code details: deleteTable@GitHub.

The Table Store SDK provides the following single-row operation interfaces: PutRow, GetRow, UpdateRow, and DeleteRow.

PutRow

Insert data into the specified row.

API

```
    /**
* Insert a data row to a table, or overwrite a data row in the table.
*
* @param putRowRequest Encapsulate the parameters required to perform the PutRow operation.
* @return PutRow Content of the response to the PutRow operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public PutRowResult putRow(PutRowRequest putRowRequest)
throws OTSException, ClientException;
```

Example 1

Insert a data row.

```
    // Define the primary keys of the row, which must be consistent with the primary keys defined in TableMeta
during table creation
RowPrimaryKey primaryKey = new RowPrimaryKey();
primaryKey.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(1));
primaryKey.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(101));
```

```
RowPutChange rowChange = new RowPutChange("SampleTable");
rowChange.setPrimaryKey(primaryKey);

// Define the attribute columns of the 100 rows
rowChange.addAttributeColumn("col0", ColumnValue.fromLong(10));
rowChange.addAttributeColumn("col1", ColumnValue.fromLong(111111));
rowChange.addAttributeColumn("col2", ColumnValue.fromString("Beijing, Shanghai, Hangzhou, Shenzhen"));

// RowExistenceExpectation.EXPECT_NOT_EXIST indicates that data is inserted only when the specified row does not exist.
rowChange.setCondition(new Condition(RowExistenceExpectation.EXPECT_NOT_EXIST));

try
{
// Construct a PutRow request object
PutRowRequest request = new PutRowRequest();
request.setRowChange(rowChange);

// Call PutRow to insert data
client.PutRow(request);

// Execution is successful if no exception is thrown.
System.out.println("Put row succeeded.");
} catch (ClientException ex) {
System.out.println("Put row failed.");
} catch (OTSException ex) {
System.out.println("Put row failed.");
}
```

TIPS:

RowExistenceExpectation.IGNORE indicates that new data is still inserted even when the specified row does not exist. If the inserted data is the same as the existing data, the existing data is overwritten.

RowExistenceExpectation.EXPECT_EXIST indicates that new data is inserted only when the specified row exists. The existing data is overwritten.

RowExistenceExpectation.EXPECT_NOT_EXIST indicates that data is inserted only when the specified row does not exist.

Code details: PutRow@GitHub.

Example 2

Insert a data row only when the following conditions are met: The specified row exists, the value of col0 is smaller than 5, and that of col2 is not "beijing".

```
    // Define the primary keys of the row, which must be consistent with the primary keys defined in TableMeta
during table creation
RowPrimaryKey primaryKey = new RowPrimaryKey();
primaryKey.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(1));
primaryKey.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(101));

RowPutChange rowChange = new RowPutChange("SampleTable");
rowChange.setPrimaryKey(primaryKey);

// Define the attribute columns of the 100 rows
rowChange.addAttributeColumn("col0", ColumnValue.fromLong(10));
rowChange.addAttributeColumn("col1", ColumnValue.fromLong(111111));
rowChange.addAttributeColumn("col2", ColumnValue.fromString("Beijing, Shanghai, Hangzhou, Shenzhen"));

try
{
//Condition 1: the value of col0 is smaller than 5
ColumnCondition filter1 = new RelationalCondition(
"col0",
RelationalCondition.CompareOperator.LESS_THAN,
ColumnValue.fromLong(5));

//Condition 2: the value of col2 is not "beijing"
ColumnCondition filter2 = new RelationalCondition(
"col2",
RelationalCondition.CompareOperator.NOT_EQUAL,
ColumnValue.fromString("beijing"));

// Construct a combination of Condition 1 and Condition 2 with the AND relationship
ColumnCondition cond = new CompositeCondition(CompositeCondition.LogicOperator.AND)
.addCondition(filter1).addCondition(filter2);

// RowExistenceExpectation.EXPECT_NOT_EXIST indicates that the subsequent logic is executed only when the
specified row does not exist; otherwise, results are returned directly.
Condition condition = new Condition(RowExistenceExpectation.EXPECT_NOT_EXIST);

// Configure the following column-specific conditions for data insertion: the value of col0 is smaller than 5 and that
of col2 is not "beijing"
condition.setColumnCondition(cond);

// Configure conditions
rowChange.setCondition(cond);

// Construct a PutRow request object
PutRowRequest request = new PutRowRequest();
request.setRowChange(rowChange);

// Call PutRow to insert data
client.PutRow(request);

// Execution is successful if no exception is thrown.
System.out.println("Put row succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Put row failed.");
```

```
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Put row failed.");
}
```

GetRow

Read a single data row based on a given primary key.

API

```
    /**
* Return a data row in a table.
*
* @param getRowRequest Encapsulate the parameters required to perform the GetRow operation.
* @return Content of the response to the GetRow operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public GetRowResult getRow(GetRowRequest getRowRequest)
throws OTSException, ClientException;
```

Example 1

Read a data row.

```
    // Define the primary keys of the row, which must be consistent with the primary keys defined in TableMeta
during table creation
RowPrimaryKey primaryKeys = new RowPrimaryKey();
primaryKeys.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(1));
primaryKeys.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(101));

SingleRowQueryCriteria criteria = new SingleRowQueryCriteria("SampleTable");
criteria.setPrimaryKey(primaryKeys);

try
{
// Construct a query request object. The entire row is read if no column is specified.
GetRowRequest request = new GetRowRequest();
request.setRowQueryCriteria(criteria);

// Call the GetRow interface to query data
GetRowResult result = client.getRow(request);

// Output the data row
Row row = result.getRow();
int consumedReadCU = result.getConsumedCapacity().getCapacityUnit().getReadCapacityUnit();
System.out.println("Consumed capacity unti : " + consumedReadCU);
System.out.println("col0 : " + row.getColumns().get("col0"));
System.out.println("col1 : " + row.getColumns().get("col1"));
```

```
System.out.println("col2 : " + row.getColumns().get("col2"));

// Data is read successfully if no exception is thrown.
System.out.println("Get row succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Get row failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Get row failed.");
}
```

TIPS:

If you query a data row, the system returns the data in all columns of the row. You can use the criteria.addColumnsToGet parameter to read the data in specified columns. For example, the system only returns the data in col0 and col1 if col0 and col1 are inserted into columnsToGet.

Conditional filter is supported. For example, you can configure the system to return results only when the value of col0 is greater than 24.

When the columnsToGet and Condition parameters are both used, the system returns results first based on columnsToGet and then filters the returned columns based on Condition.

When a specified column does not exist, PassIfMissing controls the action to be taken.

Code details: GetRow@GitHub.

Example 2

Read a data row using the filter feature.

The following code queries data and configures the system to return only the data in col0 and col2 and filter the data based on the condition that the value of col0 is 24 or the value of col2 is not “chengdu”.

```
    // Define the primary keys of the row, which must be consistent with the primary keys defined in TableMeta
during table creation
RowPrimaryKey primaryKeys = new RowPrimaryKey();
primaryKeys.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(1));
primaryKeys.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(101));

SingleRowQueryCriteria criteria = new SingleRowQueryCriteria("SampleTable");
criteria.setPrimaryKey(primaryKeys);
```

```
try
{
//Condition 1: the value of col0 is 24
ColumnCondition filter1 = new RelationalCondition(
"col0",
RelationalCondition.CompareOperator.EQUAL,
ColumnValue.fromLong(24));

//Condition 2: the value of col2 is not "chengdu"
ColumnCondition filter2 = new RelationalCondition(
"col1",
RelationalCondition.CompareOperator.NOT_EQUAL,
ColumnValue.fromString("chengdu"));

// Construct a combination of Condition 1 and Condition 2 with the OR relationship
criteria.setFilter(new CompositeCondition(CompositeCondition.LogicOperator.OR)
.addCondition(filter1).addCondition(filter2));

// Construct a query request object. The entire row is read if no column is specified.
GetRowRequest request = new GetRowRequest();
request.setRowQueryCriteria(criteria);

// Call the GetRow interface to query data
GetRowResult result = client.getRow(request);

// Output the data row
Row row = result.getRow();
int consumedReadCU = result.getConsumedCapacity().getCapacityUnit().getReadCapacityUnit();
System.out.println("Consumed capacity unti : " + consumedReadCU);
System.out.println("col0 : " + row.getColumns().get("col0"));
System.out.println("col2 : " + row.getColumns().get("col2"));

// Data is read successfully if no exception is thrown.
System.out.println("Get row succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Get row failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Get row failed.");
}
```

UpdateRow

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

API

```
    /**
* Update the read/write CapacityUnit of a table.
*
```

```
* @param updateTableRequest Encapsulate the parameters required to perform the UpdateTable operation.
* @return Content of the response to the UpdateTable operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public UpdateTableResult updateTable(UpdateTableRequest updateTableRequest)
throws OTSException, ClientException;
```

Example

Update the data of the specified row.

```
    // Define the primary keys of the row, which must be consistent with the primary keys defined in TableMeta
during table creation
RowUpdateChange rowChange = new RowUpdateChange("SampleTable");
RowPrimaryKey primaryKeys = new RowPrimaryKey();
primaryKeys.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(1));
primaryKeys.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(101));
rowChange.setPrimaryKey(primaryKeys);

// Define the attribute columns of the 100 rows
rowChange.addAttributeColumn("col0", ColumnValue.fromLong(99));
rowChange.addAttributeColumn("col2", ColumnValue.fromString("Beijing, Shanghai, Hangzhou, Shenzhen"));

// Delete col1
rowChange.deleteAttributeColumn("col1");

rowChange.setCondition(new Condition(RowExistenceExpectation.EXPECT_EXIST));

try
{
// Construct an UpdateRow request object
UpdateRowRequest request = new UpdateRowRequest();
request.setRowChange(rowChange);

// Call the UpdateRow interface
client.updateRow(request);

// Execution is successful if no exception is thrown.
System.out.println("Update row succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Update row failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Update row failed.");
}
```

TIPS:

Row updating supports the use of conditional statements.

Code details: UpdateRow@GitHub.

DeleteRow

API

```
    /**
* Delete a data row in a table.
*
* @param deleteRowRequest Encapsulate the parameters required to perform the DeleteRow operation.
* @return Content of the response to the DeleteRow operation.
* @throws OTSException Error message returned by Table Store
* @throws ClientException The returned results are invalid because of a network error or timeout.
*/
public DeleteRowResult deleteRow(DeleteRowRequest deleteRowRequest)
throws OTSException, ClientException;
```

Example

Delete a data row.

```
    // The values of the primary key columns of the row to be deleted are 0 and 100.
RowDeleteChange rowChange = new RowDeleteChange("SampleTable");
RowPrimaryKey primaryKeys = new RowPrimaryKey();
primaryKeys.addPrimaryKeyColumn("pk0", PrimaryKeyValue.fromLong(0));
primaryKeys.addPrimaryKeyColumn("pk1", PrimaryKeyValue.fromLong(100));
rowChange.setPrimaryKey(primaryKeys);

try
{
// Construct a request
DeleteRowRequest request = new DeleteRowRequest();
request.setRowChange(rowChange);

// Call the DeleteRow interface to delete the row
client.deleteRow(request);

// The row is deleted successfully if no exception is thrown.
System.out.println("Delete table succeeded.");
} catch (ClientException ex) {
// A parameter error or any other error occurs if a client-side exception is thrown.
System.out.println("Delete row failed.");
} catch (OTSException ex) {
// The request fails if a server-side exception is thrown.
System.out.println("Delete row failed.");
}
```

TIPS:

Row deletion supports the use of conditional statements.

Code details: DeleteRow@GitHub.

Python-SDK

Download SDK

Java SDK

SDK whose version above 4.0.0 provides time to live and multiple versions of data, but not incompatible with the SDK in Version 2.x.x.

SDK version: 4.1.0

Release date: 2016-10-11

Download: aliyun_tablestore_java_sdk_4.1.0.zip

Updates:

- The split points of partitions can be get from reponse of DescribeTable.

SDK version: 4.0.0

Release date: 2016-08-01

Download:aliyun_tablestore_java_sdk_4.0.0.zip

Updates:

Provides Time to live of data。

Provides multiple versions of data。

SDK version: 2.2.4

Release date: 2016-05-12

Download: aliyun_tablestore_java_sdk_2.2.4.zip

Updates:

Add API condition update.

Add filter.

SDK version 2.1.0

Release date: 2015-11-12

Download: aliyun-ots-java-sdk-2.1.0.zip

Updates:

- Asynchronous network transmission and performance tuning: When the CPU usage is the same, the QPS is increased by several times.
- Flexible and easy-to-use asynchronous interfaces: Callback is introduced and Future is returned simultaneously.
- Unbundled from OSS SDK: The new version only includes code of TableStore SDK. The directory is slightly adjusted.
- Optimized retry logic: The default retry logic is optimized. An erroneous single row can be retried independently during batch operations. The retry logic custom method is clearer.
- Optimized log: Logs are recorded for each step from request sending to request receiving. Logs of slow requests are recorded. Logs of the whole chain from SDK to back-end services are recorded using TraceId.
- OTSWriter interface supporting batch data importing: This interface aims at providing easy-to-use and efficient data importing service for users.
- Other optimized functions: Toolbox functions for various data classes are enriched and interfaces for data size computing are provided.

NOTE:

The new SDK may be slightly incompatible with the SDK in Version 2.0.4.

- When an old SDK is replaced with the new one, you need to change the import paths of a few classes, because the packages of these data classes are adjusted. For example, the package of “ClientConfiguration” is changed from “com.aliyun.openservices” to “com.aliyun.openservices.ots”. The major reason why the package is adjusted is that Table

Store SDK is unbundled from OSS SDK. It is thus more appropriate to put the classes of data commonly used into the package of OTS.
- When you no longer use an OTSClient instance (for example, before the program ends), call the shutdown method of OTSClient to release the thread and connection resources occupied by the OTSClient object.
- Names of some configuration items in “ClientConfiguration” are adjusted. For example, a time unit is added as a configuration item.
- The dependency between packages in the new SDK is changed. For example, “HttpAsyncClient” and “Jodatime” are used. If any problem occurs during SDK running, check whether a conflicting dependency is introduced.

SDK version 2.0.4

Release date: 2015-09-25

Download: aliyun-ots-java-sdk-2.0.4.zip

PHP SDK

PHP SDK version 2.1.1

Release date: 2017-1-14

Download: aliyun-tablestore-php-sdk-2.1.1.zip

Updates:

- Support 32 bit operation system.

PHP SDK version 2.1.0

Release date: 2016-11-16

Download: aliyun-ots-php-sdk-2.1.0.zip

Updates:

- Provides Conditional Update and Filter.
- Compatible with PHP version 5.5 and 5.6.

PHP SDK version 2.0.3

Release date: 2016-05-18

Download: aliyun-ots-php-sdk-2.0.3.zip

Updates：

- Remove delete table from example code.

PHP SDK version 2.0.2

Release date: 2016-04-11

Download: aliyun-ots-php-sdk-2.0.2.zip

Updates:

- Fix a bug of pb decode.

PHP SDK version 2.0.0

Download: aliyun-ots-php-sdk-2.0.0.zip

Updates:

- Support all of OTS APIs.
- Compatible with PHP version 5.3, 5.4, 5.5 and 5.6.
- Include standard retry strategy.
- Use Guzzle Http Client as network library.
- Use composer as dependency management and engineering organization tools.
- Use phpDocumentor 2 to generate programming document in HTML format.

Python SDK

Python SDK Development Kit Version 2.1.0

Release date: 2016-10-15

Download: aliyun-ots-python-sdk-2.1.0.zip

Updates:

- Provides Conditional Update and Filter.

Python SDK Development Kit Version 2.0.8

Release date: 2016-03-30

Download: aliyun-ots-python-sdk-2.0.8.zip

Updates:

- Provides HTTPS access and certificate verification.

Python SDK Development Kit Version 2.0.7

Release date: 2015-12-30

Download: aliyun-ots-python-sdk-2.0.7.zip

Updates:

- Refine the example code.

Python SDK Development Kit Version 2.0.6

Release date: 2015-10-23

Download: aliyun-ots-python-sdk-2.0.6.zip

Updates:

- Adjust the backoff retry strategy for specified exceptions.

Python SDK Development Kit Version 2.0.5

Python SDK Development Kit Download:ots_python_sdk-2.0.5.zip

Python SDK Development Kit Version 2.0.6

Download: aliyun-ots-python-sdk-2.0.6.zip

Updates:

- Adjust retry backoff strategy for part of error exceptions

C# .NET SDK

C# .NET SDK Development Kit Version3.0.0

Release date: 2016-05-05

Download: aliyun_table_store_dotnet_sdk_3.0.0.zip

Updates:

Provides Conditional Update and Filter.

Remove complie warning.

Remove unused dependencies.

Remove unused code.

Add the illegal parameter check.

Ttrim configuration parameters.

Add UserAgent header.

Remove Condition.IGNORE、Condition.EXPECT_EXIST 和 Condition.EXPECT_NOT_EXIST。

Rename DLL file Aliyun.dll to Aliyun.TableStore.dll.

Open to GitHub.

C# .NET SDK Development Kit Version2.2.1

Release date: 2016-04-14

Download: aliyun-ots-dotnet-sdk-2.2.1.zip

Updates:

- Http headers are case-insensitive when SDK verify http requests.

C# .NET SDK Development Kit Version2.2.0

Release date: 2016-04-05

Download: aliyun-ots-dotnet-sdk-2.2.0.zip

更新日志：

连接池的默认连接个数从 50 调整到 300。

新增 conditional update 功能。

C# .NET SDK Development Kit Version2.1.0

Release date: 2015-12-30

Download: aliyun-ots-dotnet-sdk-2.1.0.zip

更新日志：

根据按量计费方式，重新调整了示例代码中的预留 CU 设置。

丰富了 BatchGetRow 和 BatchWriteRow 测试用例。

C# .NET SDK Development Kit Version 2.0.3

Download: OTS SDK For .NET 2.0.3.zip

In case the chm document can not be displayed, please use msi package to install:aliyun-ots-c#-.net-sdk-2.0.3.zip