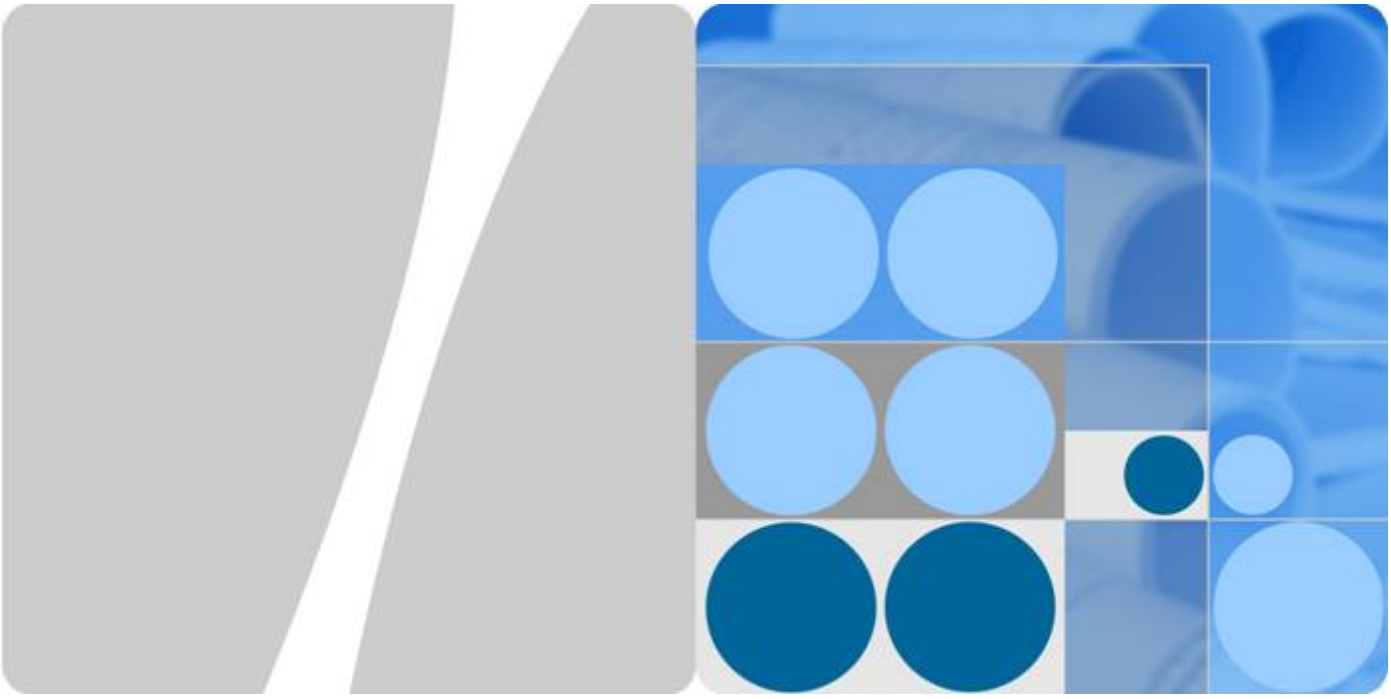


CloudTable Service

Developer Guide

Issue 06

Date 2019-04-04

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Application Development Process

This section describes how to call open source APIs of HBase, OpenTSDB, or GeoMesa in the CloudTable cluster mode to develop Java applications.

Figure 1-1 and **Table 1-1** describe the phases in the development process.

Figure 1-1 Application development process

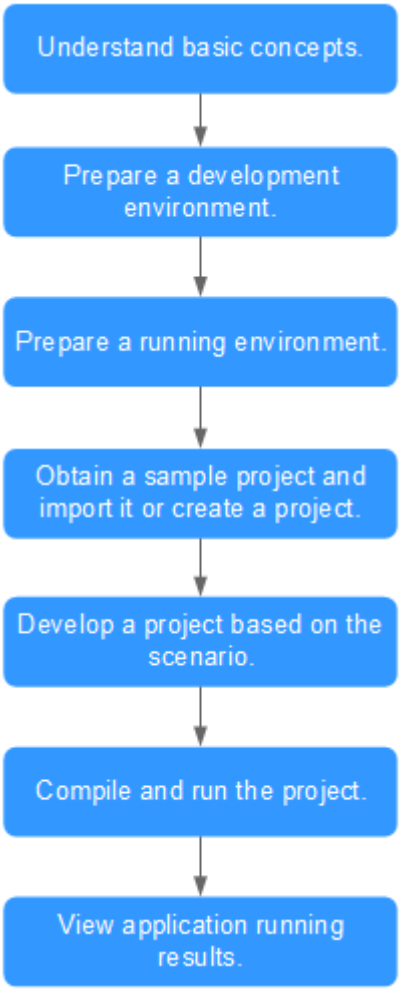


Table 1-1 Application development process details

Phase	Description	Reference
Understand basic concepts.	Before application development, learn basic concepts of HBase, OpenTSDB, or GeoMesa, understand the scenario requirements, and design tables.	**HBase**
Prepare a development environment.	The Java language is recommended for HBase, OpenTSDB, or GeoMesa application development. You can use the Eclipse tool.	**Development Environment Introduction**
Prepare a running environment.	The application running environment is a client. Install and configure the client according to the guide.	**Preparing a Windows Running Environment**
Prepare a project.	CloudTable provides example projects for different scenarios. You can import an example project to learn the application. You can also create a project according to the guide.	**Downloading a Sample Project** **Configuring and Importing a Project**
Develop a project based on the scenario.	An example project using Java is provided, including creating a table, writing data into the table, and deleting the table.	**Developing HBase Applications** **Developing OpenTSDB Applications** **Developing GeoMesa Applications** **Developing HBase Elasticsearch Full-Text Search Applications**
Compile and run the application.	You can compile the developed application and submit it for running.	**Compiling and Running Applications** **Compiling and Running an Application When a Client Is Installed** or **Compiling and Running an Application When No Client Is Installed**

Phase	Description	Reference
View application running results.	Application running results are exported to a path you specify. You can also view the application running status on the UI.	• In Windows: **Viewing Commissioning Results** • In Linux: **Viewing Commissioning Results**

2 Preparing a Development Environment

2.1 Development Environment Introduction

Table 2-1 describes the environment required for secondary development.

Table 2-1 Development environment

Item	Description
OS	Windows OS. Windows 7 or later is recommended.
JDK installation	Basic configurations of the development environment. JDK 1.7 or 1.8 is required. You are recommended to use JDK 1.8 for better compatibility of later versions. **NOTE** For security purpose, CloudTable supports only TLS 1.1 and TLS 1.2 encryption protocols. IBM JDK supports only 1.0 by default. If you use IBM JDK, set **com.ibm.jsse2.overrideDefaultTLS** to **true**. After the parameter setting, TLS1.0/1.1/1.2 can be supported at the same time. For details, see the related instructions on the IBM official website.
Eclipse installation and configuration	It is a tool used to develop CloudTable applications.
Network	Ensure that the development environment or client can communicate with the network of the CloudTable server.

2.2 Preparing a Running Environment

2.2.1 Preparing a Windows Running Environment

Scenario

The running environment for CloudTable application development can be deployed on Windows. You can perform the following operations to prepare the running environment.

Procedure

Step 1 Check whether a CloudTable cluster is installed and run properly.

Step 2 Prepare a Windows ECS.

For details about how to prepare an ECS, see **Preparing an ECS**.

Step 3 Install JDK 1.7 or later on the Windows ECS. However, you are recommended to use JDK 1.8 or later and install Eclipse that uses JDK 1.7 or later.

NOTE

- If you use IBM JDK, ensure that the JDK configured in Eclipse is IBM JDK.
- If you use Oracle JDK, ensure that the JDK configured in Eclipse is Oracle JDK.
- Do not use the same workspace and the sample project in the same path for different Eclipse programs.

----End

2.3 Downloading a Sample Project

Prerequisites

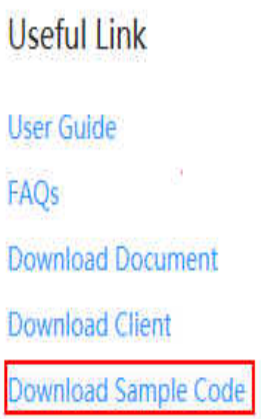
Ensure that CloudTable has been installed and is running properly.

Downloading the Sample Project (Cluster Mode)

Step 1 Download a sample project.

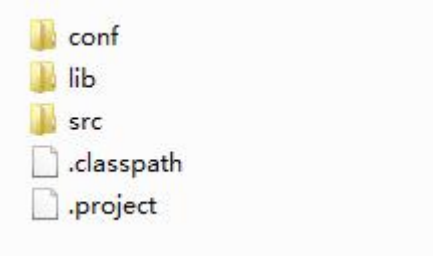
Log in to the CloudTable management console and click **Help**. In the **Helpful Links** area on the right, click **Download Sample Code** to download an installation package of a sample code project. **Figure 2-1** shows the location of **Download Sample Code**.

Figure 2-1 Downloading sample code



Step 2 After the download is complete, decompress the installation package of the sample code project to a local directory to obtain an Eclipse Java project. **Figure 2-1** shows the directory structure of the sample code project.

Figure 2-2 Directory structure of the sample code project



----End

2.4 Configuring and Importing a Project

Background Information

After importing the CloudTable sample code project to Eclipse, you can start learning CloudTable application development samples.

Prerequisites

You have correctly configured the running environment. For details about how to configure the running environment, see **Preparing a Windows Running Environment**.

Procedure

Step 1 Import the sample project to the Windows development environment. For details about how to obtain the sample project, see **Downloading a Sample Project**.

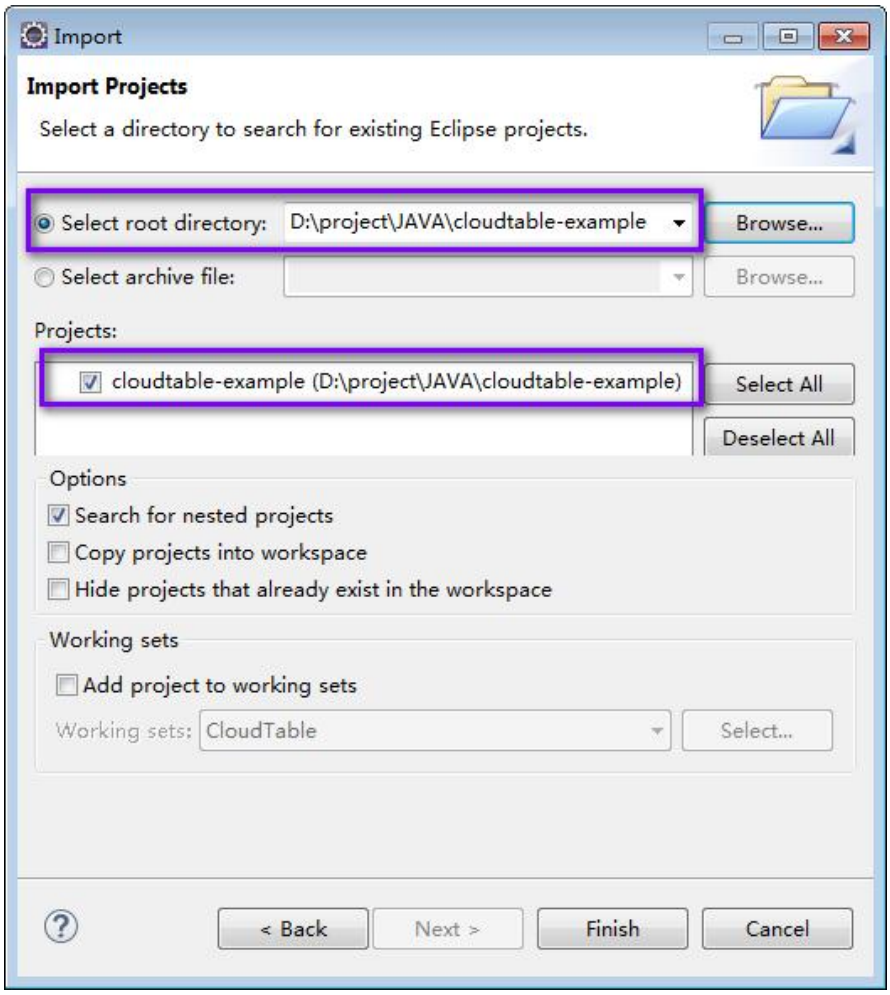
Step 2 In the application development environment, import the sample project to the Eclipse development environment.

1. Choose **File** > **Import** > **General** > **Existing Projects into Workspace** > **Next** > **Browse**.

   The **Browse Folder** dialog box is displayed, as shown in **Figure 2-3**.

2. Select the sample project folder, and click **Finish**.

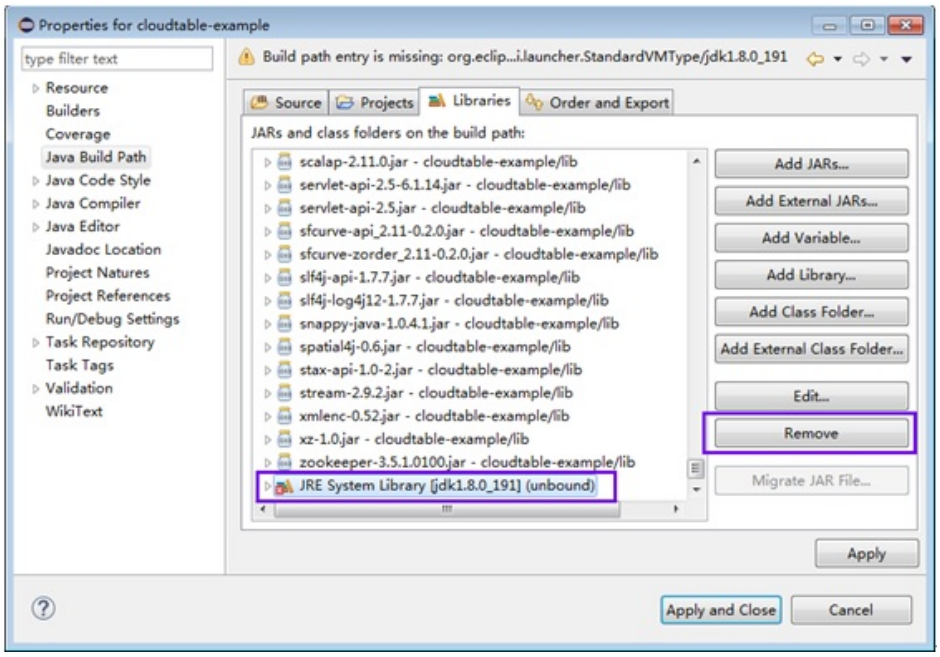
Figure 2-3 Importing a sample project



Step 3 Right-click the **cloudtable-example** project, and choose **Properties** from the shortcut menu. The **Properties for cloudtable-example** window is displayed.

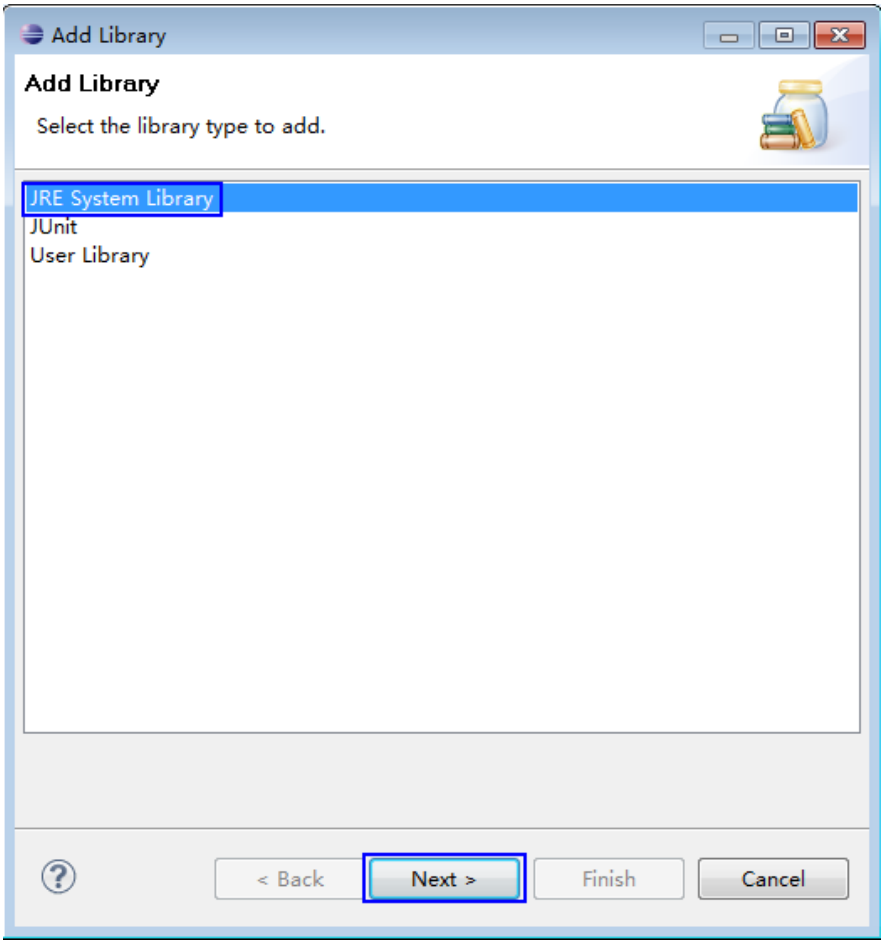
1. In the navigation tree, select **Java Build Path**. Click the **Libraries** tab, select all error JDKs, and click **Remove**, as shown in **Figure 2-4**.

Figure 2-4 Delete error JDKs



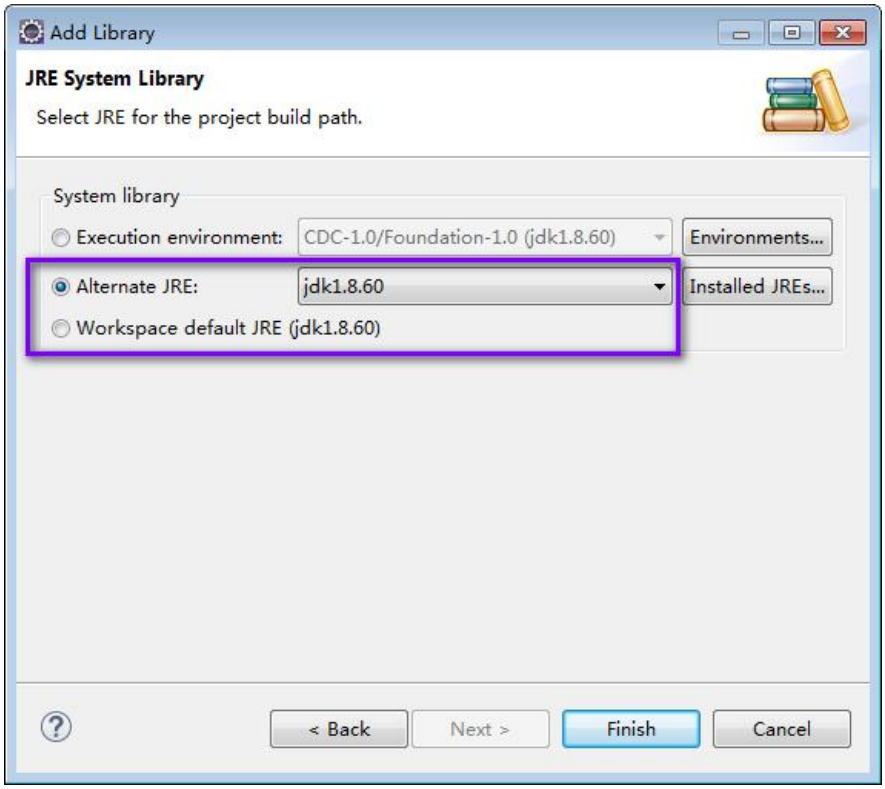
2. Click **Add Library...** shown in **Figure 2-5**. Select **JRE System Library** in the pop-up window.

Figure 2-5 Adding libraries



3. In the **Add Library** dialog box, select a JDK version from the drop-down list of **Alternate JRE** or **Workspace default JRE**. Select **Alternate JRE** and select the JDK version, as shown in **Figure 2-6**.

Figure 2-6 Selecting JRE



4. Click **Finish** to complete configuration and close the window.

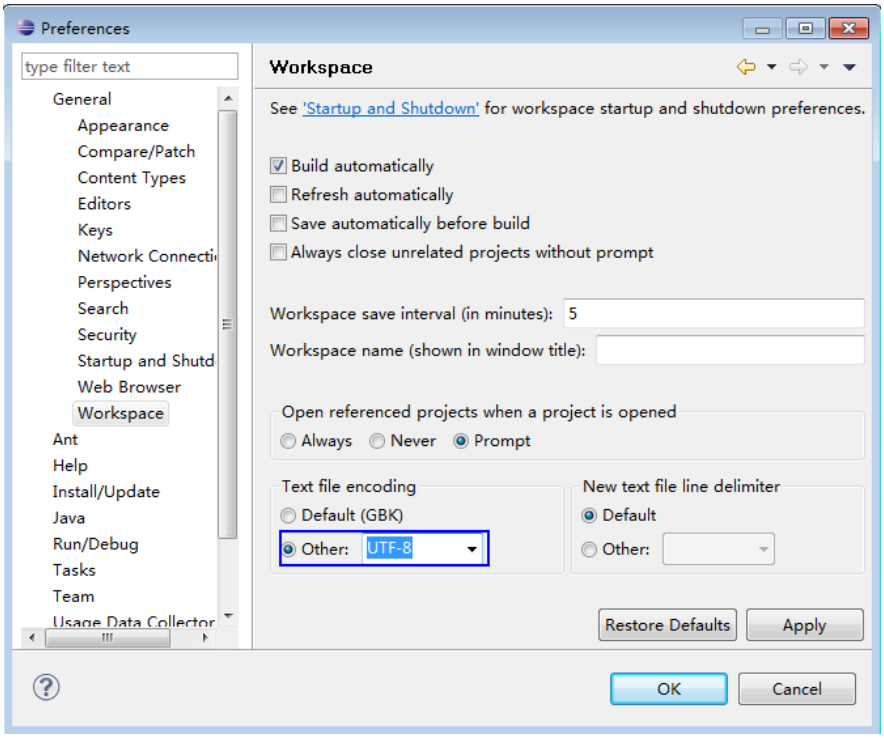
Step 4 Set the Eclipse text file coding format to prevent garbled characters.

1. On the Eclipse menu bar, choose **Window** > **Preferences**.

   The **Preferences** window is displayed.

2. In the navigation tree, choose **General** > **Workspace**. In the **Text file encoding** area, select **Other** and set the value to **UTF-8**. Click **Apply** and then **OK**. Figure 2-7 shows the settings.

Figure 2-7 Setting the Eclipse encoding format



Step 5 Open the **conf/hbase-site.xml** file in the sample project and change the value of **hbase.zookeeper.quorum** to the correct ZooKeeper address.

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>xxx-zk1.cloudtable.com,xxx-zk2.cloudtable.com,xxx-zk3.cloudtable.com</value>
</property>
```

value is the domain name of the ZooKeeper cluster. Log in to the CloudTable management console and choose **Cluster Mode**. In the cluster list, locate the required cluster and obtain its ZK link in the **ZK Link** column.

----End

3 Developing HBase Applications

3.1 Typical Application Scenario

You can quickly learn and master the HBase development process and know key interface functions in a typical application scenario.

Description

Develop an application to manage information about users who use service A in an enterprise. **Table 3-1** provides the user information. Procedures are as follows:

- Create a user information table.
- Add users' educational backgrounds and titles to the table.
- Query user names and addresses by user ID.
- Query information by user name.
- Query information about users whose age ranges from 20 to 29.
- Collect the number of users and their maximum, minimum, and average age.
- Deregister users and delete user data from the user information table.
- Delete the user information table after service A ends.

Table 3-1 User information

ID	Name	Gender	Age	Address
12005000201	A	Male	19	Shenzhen, Guangdong
12005000202	B	Female	23	Shijiazhuang, Hebei
12005000203	C	Male	26	Ningbo, Zhejiang
12005000204	D	Male	18	Xiangyang, Hubei

ID	Name	Gender	Age	Address
12005000205	E	Female	21	Shangrao, Jiangxi
12005000206	F	Male	32	Zhuzhou, Hunan
12005000207	G	Female	29	Nanyang, Henan
12005000208	H	Female	30	Kaixian, Chongqing
12005000209	I	Male	26	Weinan, Shaanxi
12005000210	J	Male	25	Dalian, Liaoning

Data Planning

Proper design of a table structure, RowKeys, and column names enable you to make full use of HBase advantages. In the sample project, a unique ID is used as a RowKey, and columns are stored in the **info** column family.

3.2 Development Guideline

Function Description

Determine functions to be developed based on the preceding scenario. **Table 3-2** describes functions to be developed.

Table 3-2 Functions to be developed in HBase

No.	Procedure	Code Implementation
1	Create a table based on the information in **Typical Application Scenario**.	For details, see **Creating a Table**.
2	Import user data.	For details, see **Inserting Data**.
3	Add an educational background column family, and add educational backgrounds and titles to the user information table.	For details, see **Modifying a Table**.
4	Query user names and addresses by user ID.	For details, see **Reading Data Using Get**.
5	Query information by user name.	For details, see **Using a Filter**.

No.	Procedure	Code Implementation
6	Deregister users and delete user data from the user information table.	For details, see **Deleting Data** .
7	Delete the user information table after service A ends.	For details, see **Deleting a Table**.

Key Design Principles

HBase is a distributed database system based on the lexicographic order of RowKeys. The RowKey design has great impact on performance, so the RowKeys must be designed based on specific services.

3.3 Sample Code Description

3.3.1 Parameter Configuration

Step 1 Before executing sample code, configure the correct ZooKeeper cluster address in the **hbase-site.xml** configuration file.

The configuration items are as follows:

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>xxx-zk1.cloudtable.com,xxx-zk2.cloudtable.com,xxx-zk3.cloudtable.com</value>
</property>
```

value is the domain name of the ZooKeeper cluster. Log in to the CloudTable management console and choose **Cluster Mode**. In the cluster list, locate the required cluster and obtain its ZK link in the **ZK Link** column.

Step 2 Modify the parameters related to IAM authentication in the sample code project.

- If IAM authentication is enabled for the CloudTable cluster:

  In the sample code project, **IAM_AUTH_MODE** in the com.huawei.cloudtable.hbase.examples.TestMain class must be set to **true** and the **user**, **ak,** and **sk** parameters need to be configured.

  The code is as follows:

  ```
  private static boolean IAM_AUTH_MODE = true;
  private static String user = "XXXXXX";
  private static String ak = "XXXXXX";
  private static String sk = "XXXXXX";
  ```

  - **user**: username If the cluster is created by a user's sub-user, **user** must be set to *Sub-user.End user* when the sub-user accesses the cluster, and set to the user name when the end user accesses the cluster.
  - **ak** and **sk**: Access Key ID (AK) and Secret Access Key (SK). Set them to the AK plaintext and SK plaintext, respectively. You can move your cursor over your account in the upper right corner of the management console and choose **My Credential**. Click **Access Keys** tab. On the **Access Keys** tab page, you can view the existing access keys or click **Add Access Key** to add an access key.

> NOTE
>
> The IAM authentication mode provides better security than the normal mode. Therefore, you are advised to enable IAM authentication for the CloudTable cluster and use IAM authentication in client or application code to connect to the cluster.

- If IAM authentication is disabled for the CloudTable cluster:

  IAM_AUTH_MODE in the com.huawei.cloudtable.hbase.examples.TestMain class must be set to **false**.

----End

3.3.2 Creating the Configuration Object

Function Description

HBase obtains configuration items by loading a configuration file.

> NOTE
>
> 1. Loading the configuration file is time-consuming. If unnecessary, use the same Configuration object.
> 2. Multi-thread synchronization is not considered in the sample code. If necessary, add it by yourself. Other sample codes are the same.

Sample Code

The following code snippets are in the **com.huawei.cloudtable.hbase.examples** packet.

```
private static void init() throws IOException {
  // Default load from conf directory
  conf = HBaseConfiguration.create(); // Note [1]
  String userdir = System.getProperty("user.dir") + File.separator + "conf" + File.separator;
  Path hbaseSite = new Path(userdir + "hbase-site.xml");
  if (new File(hbaseSite.toString()).exists()) {
    conf.addResource(hbaseSite);
  }
}
```

Precautions

- Note [1] If the **conf** directory of the configuration file is added to the **classpath** path, the code for loading the specified configuration file can be skipped.

3.3.3 Performing IAM Authentication for Clusters

Function Description

If IAM authentication is enabled for a CloudTable cluster, you need to use a tenant's AK and SK to perform authentication before performing subsequent operations.

Sample Code

```
private static boolean IAM_AUTH_MODE = true;
private static String user = "";
```



```
private static String ak = "";
private static String sk = "";
public static void login(Configuration conf) throws IOException {
  if (IAM_AUTH_MODE) {
    UserProviderExtend.loginWithAKSK(conf, user, ak, sk);
  }
}
```

NOTE

In the life cycle of a process, the **UserProviderExtend.loginWithAKSK** function needs to be invoked only once.

3.3.4 Creating the Connection Object

Function Description

HBase creates a Connection object using the **ConnectionFactory.createConnection(configuration)** method. The transferred parameter is the Configuration created in the last step.

Connection encapsulates the connections between underlying applications and servers and ZooKeeper. Connection is instantiated using the **ConnectionFactory** class. Creating Connection is a heavyweight operation. Connection is thread-safe. Therefore, multiple client threads can share one Connection.

In a typical scenario, a client program uses a Connection, and each thread obtains its own Admin or Table instance and invokes the operation interface provided by the Admin or Table object. You are not advised to cache or pool Table and Admin. The lifecycle of Connection is maintained by invokers who can release resources by invoking **close()**.

NOTE

When the service code is connected to the same CloudTable cluster, you are advised to create one Connection and reuse it for multiple threads. You do not need to create a Connection for every thread. Connection is a connector for connecting to a CloudTable cluster. Excessive Connections will increase loads on ZooKeeper and deteriorate service read/write performance.

Sample Code

The following code snippet is an example of creating a Connection object:

```
private TableName tableName = null;
private Connection conn = null;

public HBaseSample(Configuration conf) throws IOException {
  this.tableName = TableName.valueOf("hbase_sample_table");
  this.conn = ConnectionFactory.createConnection(conf);
}
```

3.3.5 Creating a Table

Function Description

In HBase, a table is created using the **createTable** method of the **org.apache.hadoop.hbase.client.Admin** object. You need to specify a table name

and a column family name. You can create a table by using either of the following methods, but the latter one is recommended:

- Quickly create a table. A newly created table contains only one region, which will be automatically split into multiple new regions as data increases.
- Create a table using pre-assigned regions. You can pre-assign multiple regions before creating a table. This mode accelerates data write at the beginning of massive data write.

NOTE

The table name and column family name of a table consist of letters, digits, and underscores (_) but cannot contain any special characters.

Sample Code

```
public void testCreateTable() {
  LOG.info("Entering testCreateTable.");

  // Specify the table descriptor.
  HTableDescriptor htd = new HTableDescriptor(tableName); // (1)

  // Set the column family name to info.
  HColumnDescriptor hcd = new HColumnDescriptor("info"); // (2)

  // Set data encoding methods. HBase provides DIFF,FAST_DIFF,PREFIX
  // and PREFIX_TREE
  hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);  // Note [1]

  // Set compression methods, HBase provides two default compression
  // methods:GZ and SNAPPY
  // GZ has the highest compression rate,but low compression and
  // decompression efficiency,fit for cold data
  // SNAPPY has low compression rate, but high compression and
  // decompression efficiency,fit for hot data.
  // it is advised to use SANPPY
  hcd.setCompressionType(Compression.Algorithm.SNAPPY);
  htd.addFamily(hcd); // (3)

  Admin admin = null;
  try {
    // Instantiate an Admin object.
    admin = conn.getAdmin(); // (4)
    if (!admin.tableExists(tableName)) {
      LOG.info("Creating table...");
      admin.createTable(htd); // Note [2] (5)
      LOG.info(admin.getClusterStatus());
      LOG.info(admin.listNamespaceDescriptors());
      LOG.info("Table created successfully.");
    } else {
      LOG.warn("table already exists");
    }
  } catch (IOException e) {
    LOG.error("Create table failed.", e);
  } finally {
    if (admin != null) {
      try {
        // Close the Admin object.
        admin.close();
      } catch (IOException e) {
        LOG.error("Failed to close admin ", e);
      }
    }
  }
  LOG.info("Exiting testCreateTable.");
}
```


Explanation

(1) Create a table descriptor.

(2) Create a column family descriptor.

(3) Add the column family descriptor to the table descriptor.

(4) Obtain the Admin object. You use the Admin object to create a table and a column family, check whether the table exists, modify the table structure and column family structure, and delete the table.

(5) Invoke the Admin object to create a table.

Precautions

- Note [1] Use the following code to set the compression mode for a column family:

```
// Set an encoding algorithm. HBase provides four encoding algorithms: DIFF, FAST_DIFF, PREFIX, and PREFIX_TREE.
 hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);

// Set a file compression mode. By default, HBase provides two compression algorithms: GZ and SNAPPY.
// GZ has a high compression rate but low compression and decompression performance. It is applicable to cold data.
 // SNAPPY has a low compression rate but high compression and decompression performance. It is applicable to hot data.
// It is recommended that SNAPPY compression be enabled by default.
 hcd.setCompressionType(Compression.Algorithm.SNAPPY);
```

- Note [2] Create a table by specifying the start and end RowKeys or pre-assigning regions using RowKey arrays. The code snippet is as follows:

```
// Create a table with pre-split regions.
 byte[][] splits = new byte[4][];
 splits[0] = Bytes.toBytes("A");
 splits[1] = Bytes.toBytes("H");
 splits[2] = Bytes.toBytes("O");
 splits[3] = Bytes.toBytes("U");
 admin.createTable(htd, splits);
```

3.3.6 Deleting a Table

Function Description

In HBase a table is deleted using the **deleteTable** method of **org.apache.hadoop.hbase.client.Admin**.

Sample Code

```
public void dropTable() {
  LOG.info("Entering dropTable.");
  Admin admin = null;
  try {
    admin = conn.getAdmin();
    if (admin.tableExists(tableName)) {
      // Disable the table before deleting it.
      admin.disableTable(tableName);
      // Delete table.
      admin.deleteTable(tableName);//Note [1]
    }
```



```
    LOG.info("Drop table successfully.");
  } catch (IOException e) {
    LOG.error("Drop table failed " ,e);
  } finally {
    if (admin != null) {
      try {
        // Close the Admin object.
        admin.close();
      } catch (IOException e) {
        LOG.error("Close admin failed " ,e);
      }
    }
  }
  LOG.info("Exiting dropTable.");
}
```

Precautions

Note [1] Only after the **disableTable** API is called, the table can be deleted by calling the **deleteTable** API. Therefore, **deleteTable** is often used together with **disableTable**, **enableTable**, **tableExists**, **isTableEnabled**, and **isTableDisabled**.

3.3.7 Modifying a Table

Function Description

In HBase, table information is modified using the **modifyTable** method of **org.apache.hadoop.hbase.client.Admin**.

Sample Code

```
public void testModifyTable() {
  LOG.info("Entering testModifyTable.");

  // Specify the column family name.
  byte[] familyName = Bytes.toBytes("education");
  Admin admin = null;
  try {
    // Instantiate an Admin object.
    admin = conn.getAdmin();
    // Obtain the table descriptor.
    HTableDescriptor htd = admin.getTableDescriptor(tableName);
    // Check whether the column family is specified before modification.
    if (!htd.hasFamily(familyName)) {
      // Create the column descriptor.
      HColumnDescriptor hcd = new HColumnDescriptor(familyName);
      htd.addFamily(hcd);
      // Disable the table to get the table offline before modifying
      // the table.
      admin.disableTable(tableName);
      // Submit a modifyTable request.
      admin.modifyTable(tableName, htd);  //Note [1]
      // Enable the table to get the table online after modifying the
      // table.
      admin.enableTable(tableName);
    }
    LOG.info("Modify table successfully.");
  } catch (IOException e) {
    LOG.error("Modify table failed " ,e);
  } finally {
    if (admin != null) {
      try {
        // Close the Admin object.
        admin.close();
      } catch (IOException e) {
```



```
      LOG.error("Close admin failed " ,e);
    }
   }
  }
  LOG.info("Exiting testModifyTable.");
}
```

Precautions

Note [1] Only after the **disableTable** API is called, the table can be modified by calling the **modifyTable** API. Then, call the **enableTable** API to enable the table again.

3.3.8 Inserting Data

Function Description

HBase is a column-based database. A row of data may have multiple column families, and a column family may contain multiple columns. When writing data, you must specify the columns (including the column family names and column names) to which data is written. In HBase, data (a row of data or data sets) is inserted using the **put** method of HTable.

Sample Code

```
public void testPut() {
  LOG.info("Entering testPut.");
  // Specify the column family name.
  byte[] familyName = Bytes.toBytes("info");
  // Specify the column name.
  byte[][] qualifiers = { Bytes.toBytes("name"), Bytes.toBytes("gender"),
      Bytes.toBytes("age"), Bytes.toBytes("address") };
  Table table = null;
  try {
    // Instantiate an HTable object.
    table = conn.getTable(tableName);
    List<Put> puts = new ArrayList<Put>();
    // Instantiate a Put object.
    Put put = new Put(Bytes.toBytes("012005000201"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("A"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("19"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Shenzhen, Guangdong"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000202"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("B"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Female"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("23"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Shijiazhuang, Hebei"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000203"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("C"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("26"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Ningbo, Zhejiang"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000204"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("D"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("18"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Xiangyang, Hubei"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000205"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("E"));
```



```
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Female"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("21"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Shangrao, Jiangxi"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000206"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("F"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("32"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Zhuzhou, Hunan"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000207"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("G"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Female"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("29"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Nanyang, Henan"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000208"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("H"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Female"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("30"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Kaixian, Chongqing"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000209"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("I"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("26"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Weinan, Shaanxi"));
    puts.add(put);
    put = new Put(Bytes.toBytes("012005000210"));
    put.addColumn(familyName, qualifiers[0], Bytes.toBytes("J"));
    put.addColumn(familyName, qualifiers[1], Bytes.toBytes("Male"));
    put.addColumn(familyName, qualifiers[2], Bytes.toBytes("25"));
    put.addColumn(familyName, qualifiers[3], Bytes.toBytes("Dalian, Liaoning"));
    puts.add(put);
    // Submit a put request.
    table.put(puts);

    LOG.info("Put successfully.");
  } catch (IOException e) {
    LOG.error("Put failed " ,e);
  } finally {
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed " ,e);
      }
    }
  }
  LOG.info("Exiting testPut.");
}
```

Precautions

Multiple threads are not allowed to use the same HTable instance at the same time. HTable is a non-thread-safe class. If an HTable instance is used by multiple threads at the same time, exceptions will occur.

3.3.9 Deleting Data

Function Description

In HBase, data (a row of data or data sets) is deleted using the **delete** method of a Table instance.

Sample Code

```
public void testDelete() {
  LOG.info("Entering testDelete.");

  byte[] rowKey = Bytes.toBytes("012005000201");

  Table table = null;
  try {
    // Instantiate an HTable object.
    table = conn.getTable(tableName);

    // Instantiate a Delete object.
    Delete delete = new Delete(rowKey);

    // Submit a delete request.
    table.delete(delete);

    LOG.info("Delete table successfully.");
  } catch (IOException e) {
    LOG.error("Delete table failed " ,e);
  } finally {
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed " ,e);
      }
    }
  }
  LOG.info("Exiting testDelete.");
}
```

3.3.10 Reading Data Using Get

Function Description

Before reading data from a table, create a table instance and a Get object. You can also set parameters for the Get object, such as the column family name and column name. Query results are stored in the Result object that stores multiple Cells.

Sample Code

```
public void testGet() {
  LOG.info("Entering testGet.");
  // Specify the column family name.
  byte[] familyName = Bytes.toBytes("info");
  // Specify the column name.
  byte[][] qualifier = { Bytes.toBytes("name"), Bytes.toBytes("address") };
  // Specify RowKey.
  byte[] rowKey = Bytes.toBytes("012005000201");
  Table table = null;
  try {
    // Create the Table instance.
    table = conn.getTable(tableName);
    // Instantiate a Get object.
    Get get = new Get(rowKey);
    // Set the column family name and column name.
    get.addColumn(familyName, qualifier[0]);
    get.addColumn(familyName, qualifier[1]);
    // Submit a get request.
    Result result = table.get(get);
    // Print query results.
    for (Cell cell : result.rawCells()) {
```



```
      LOG.info(Bytes.toString(CellUtil.cloneRow(cell)) + ":"
          + Bytes.toString(CellUtil.cloneFamily(cell)) + ","
          + Bytes.toString(CellUtil.cloneQualifier(cell)) + ","
          + Bytes.toString(CellUtil.cloneValue(cell)));
    }
    LOG.info("Get data successfully.");
  } catch (IOException e) {
    LOG.error("Get data failed " ,e);
  } finally {
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed " ,e);
      }
    }
  }
  LOG.info("Exiting testGet.");
}
```

3.3.11 Reading Data Using Scan

Function Description

Before reading data from a table, instantiate the Table instance of the table, and then create a Scan object and set parameters for the Scan object based on search criteria. To improve query efficiency, you are advised to specify StartRow and StopRow. Query results are stored in the ResultScanner object, where each row of data is stored as a Result object that stores multiple Cells.

Sample Code

```
public void testScanData() {
  LOG.info("Entering testScanData.");
  Table table = null;
  // Instantiate a ResultScanner object.
  ResultScanner rScanner = null;
  try {
    // Create the Configuration instance.
    table = conn.getTable(tableName);
    // Instantiate a Get object.
    Scan scan = new Scan();
    scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));
    // Set the cache size.
    scan.setCaching(1000);
    // Submit a scan request.
    rScanner = table.getScanner(scan);
    // Print query results.
    for (Result r = rScanner.next(); r != null; r = rScanner.next()) {
      for (Cell cell : r.rawCells()) {
        LOG.info(Bytes.toString(CellUtil.cloneRow(cell)) + ":"
            + Bytes.toString(CellUtil.cloneFamily(cell)) + ","
            + Bytes.toString(CellUtil.cloneQualifier(cell)) + ","
            + Bytes.toString(CellUtil.cloneValue(cell)));
      }
    }
    LOG.info("Scan data successfully.");
  } catch (IOException e) {
    LOG.error("Scan data failed " ,e);
  } finally {
    if (rScanner != null) {
      // Close the scanner object.
      rScanner.close();
    }
    if (table != null) {
```



```
    try {
      // Close the HTable object.
      table.close();
    } catch (IOException e) {
      LOG.error("Close table failed " ,e);
    }
  }
 }
 LOG.info("Exiting testScanData.");
}
```

Precautions

1. You are advised to specify StartRow and StopRow to ensure good performance with a specified Scan scope.
2. You can set **Batch** and **Caching**.
   - Batch

     Batch indicates the maximum number of records returned each time when the **next** API is invoked using Scan. This parameter is related to the number of columns read each time.
   - Caching

     Caching indicates the maximum number of next records returned for a remote procedure call (RPC) request. This parameter is related to the number of rows read by an RPC.

3.3.12 Using a Filter

Function Description

HBase Filter is used to filter data during Scan and Get. You can specify the filter criteria, such as filtering by RowKey, column name, or column value.

Sample Code

```
public void testSingleColumnValueFilter() {
 LOG.info("Entering testSingleColumnValueFilter.");
 Table table = null;
 ResultScanner rScanner = null;

 try {
  table = conn.getTable(tableName);
  Scan scan = new Scan();
  scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));
  // Set the filter criteria.
  SingleColumnValueFilter filter = new SingleColumnValueFilter(
      Bytes.toBytes("info"), Bytes.toBytes("name"), CompareOp.EQUAL,
      Bytes.toBytes("I"));
  scan.setFilter(filter);
  // Submit a scan request.
  rScanner = table.getScanner(scan);
  // Print query results.
  for (Result r = rScanner.next(); r != null; r = rScanner.next()) {
   for (Cell cell : r.rawCells()) {
    LOG.info(Bytes.toString(CellUtil.cloneRow(cell)) + ":"
        + Bytes.toString(CellUtil.cloneFamily(cell)) + ","
        + Bytes.toString(CellUtil.cloneQualifier(cell)) + ","
        + Bytes.toString(CellUtil.cloneValue(cell)));
   }
  }
  LOG.info("Single column value filter successfully.");
```



```
  } catch (IOException e) {
    LOG.error("Single column value filter failed " ,e);
  } finally {
      if (rScanner != null) {
          // Close the scanner object.
          rScanner.close();
        }
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed " ,e);
      }
    }
  }
  LOG.info("Exiting testSingleColumnValueFilter.");
}
```


4 Developing OpenTSDB Applications

4.1 Typical Application Scenario

You can quickly learn and master the OpenTSDB development process and know key API functions in a typical application scenario.

Description

Assume that a user develops an application to record and query weather information about a city. The following table lists the recorded data.

Table 4-1 Raw data

City	District	Date	Temperature	Humidity
Shenzhen	Longgang	2017/7/1 00:00:00	28	54
Shenzhen	Longgang	2017/7/1 01:00:00	27	53
Shenzhen	Longgang	2017/7/1 02:00:00	27	52
Shenzhen	Longgang	2017/7/1 03:00:00	27	51
Shenzhen	Longgang	2017/7/1 04:00:00	27	50
Shenzhen	Longgang	2017/7/1 05:00:00	27	49
Shenzhen	Longgang	2017/7/1 06:00:00	27	48
Shenzhen	Longgang	2017/7/1 07:00:00	27	46
Shenzhen	Longgang	2017/7/1 08:00:00	29	46
Shenzhen	Longgang	2017/7/1 09:00:00	30	48
Shenzhen	Longgang	2017/7/1 10:00:00	32	48
Shenzhen	Longgang	2017/7/1 11:00:00	32	49

City	District	Date	Temperature	Humidity
Shenzhen	Longgang	2017/7/1 12:00:00	33	49
Shenzhen	Longgang	2017/7/1 13:00:00	33	50
Shenzhen	Longgang	2017/7/1 14:00:00	32	50
Shenzhen	Longgang	2017/7/1 15:00:00	32	50
Shenzhen	Longgang	2017/7/1 16:00:00	31	51
Shenzhen	Longgang	2017/7/1 17:00:00	30	51
Shenzhen	Longgang	2017/7/1 18:00:00	30	51
Shenzhen	Longgang	2017/7/1 19:00:00	29	51
Shenzhen	Longgang	2017/7/1 20:00:00	29	52
Shenzhen	Longgang	2017/7/1 21:00:00	29	53
Shenzhen	Longgang	2017/7/1 22:00:00	28	54
Shenzhen	Longgang	2017/7/1 23:00:00	28	54

In this scenario, the temperature and humidity data of the Longgang district, Shenzhen, is recorded at 00:00 on July 1, 2017. OpenTSDB uses two groups of data points for modeling.

Table 4-2 Metric data point 1

Metric	City	District	Unix Timestamp	Value
city.temp	Shenzhen	Longgang	1498838400	28
city.temp	Shenzhen	Longgang	1498842000	27
city.temp	Shenzhen	Longgang	1498845600	27
city.temp	Shenzhen	Longgang	1498849200	27
city.temp	Shenzhen	Longgang	1498852800	27
city.temp	Shenzhen	Longgang	1498856400	27
city.temp	Shenzhen	Longgang	1498860000	27
city.temp	Shenzhen	Longgang	1498863600	27
city.temp	Shenzhen	Longgang	1498867200	29
city.temp	Shenzhen	Longgang	1498870800	30
city.temp	Shenzhen	Longgang	1498874400	32

Metric	City	District	Unix Timestamp	Value
city.temp	Shenzhen	Longgang	1498878000	32
city.temp	Shenzhen	Longgang	1498881600	33
city.temp	Shenzhen	Longgang	1498885200	33
city.temp	Shenzhen	Longgang	1498888800	32
city.temp	Shenzhen	Longgang	1498892400	32
city.temp	Shenzhen	Longgang	1498896000	31
city.temp	Shenzhen	Longgang	1498899600	30
city.temp	Shenzhen	Longgang	1498903200	30
city.temp	Shenzhen	Longgang	1498906800	29
city.temp	Shenzhen	Longgang	1498910400	29
city.temp	Shenzhen	Longgang	1498914000	29
city.temp	Shenzhen	Longgang	1498917600	28
city.temp	Shenzhen	Longgang	1498921200	28

Table 4-3 Metric data point 2

Metric	City	District	Unix Timestamp	Value
city.hum	Shenzhen	Longgang	1498838400	54
city.hum	Shenzhen	Longgang	1498842000	53
city.hum	Shenzhen	Longgang	1498845600	52
city.hum	Shenzhen	Longgang	1498849200	51
city.hum	Shenzhen	Longgang	1498852800	50
city.hum	Shenzhen	Longgang	1498856400	49
city.hum	Shenzhen	Longgang	1498860000	48
city.hum	Shenzhen	Longgang	1498863600	46
city.hum	Shenzhen	Longgang	1498867200	46
city.hum	Shenzhen	Longgang	1498870800	48
city.hum	Shenzhen	Longgang	1498874400	48
city.hum	Shenzhen	Longgang	1498878000	49

Metric	City	District	Unix Timestamp	Value
city.hum	Shenzhen	Longgang	1498881600	49
city.hum	Shenzhen	Longgang	1498885200	50
city.hum	Shenzhen	Longgang	1498888800	50
city.hum	Shenzhen	Longgang	1498892400	50
city.hum	Shenzhen	Longgang	1498896000	51
city.hum	Shenzhen	Longgang	1498899600	51
city.hum	Shenzhen	Longgang	1498903200	51
city.hum	Shenzhen	Longgang	1498906800	51
city.hum	Shenzhen	Longgang	1498910400	52
city.hum	Shenzhen	Longgang	1498914000	53
city.hum	Shenzhen	Longgang	1498917600	54
city.hum	Shenzhen	Longgang	1498921200	54

Each group of metric data points has two tags:

- Tags: **City** and **District**
- Tag values: ShenZhen and Longgang

You can perform the following operations on data:

- Obtain the daily monitored data and write data points of the two groups to the database through the **put** API of OpenTSDB.
- Use the query API of OpenTSDB to query and analyze the existing data.

4.2 Development Guideline

Function Description

Determine functions to be developed based on the preceding scenario. **Table 4-4** describes functions to be developed.

Table 4-4 Functions to be developed in OpenTSDB

No.	Procedure	Code Implementation
1	Build a data model based on the typical scenario.	For details, see **Writing Data**.

No.	Procedure	Code Implementation
2	Write metric data.	For details, see **Writing Data**.
3	Query data based on metrics.	For details, see **Querying Data**.
4	Delete data in a specified range.	For details, see **Deleting Data**.

4.3 Sample Code Description

4.3.1 Parameter Configuration

Step 1 Before executing the sample code, you must change the values of the following parameters in the **com.huawei.cloudtable.opentsdb.examples.OpenTsdbSample** class in the sample code project.

```
private final static String OPENTSDB_IP = "opentsdb-vdswm7gj45awlom.cloudtable.com";
private final static int OPENTSDB_PORT = 4242;
```

- **OPENTSDB_IP**: Change its value to the URL of OpenTSDB on the CloudTable cluster information page.
- **OPENTSDB_PORT**: Change the port number to **4242**.

Step 2 If IAM authentication is enabled for the CloudTable cluster, modify the following parameters in the **com.huawei.cloudtable.opentsdb.examples.OpenTsdbSample** class in the sample code project.

```
private final static boolean securityMode = true;
private final static String PROJECT_ID = "XXXXXX";
private final static String USER = "XXXXXX";
private final static String AK = "XXXXXX";
private final static String TOKEN = "XXXXXX";
```

- **securityMode**: Set it to **true**. If IAM authentication is disabled for the CloudTable cluster, this parameter must be set to **false**.

  NOTE

  The IAM authentication mode provides better security than the normal mode. Therefore, you are advised to enable IAM authentication for the CloudTable cluster and use IAM authentication in client or application code to connect to the cluster.

- **PROJECT_ID**: Set it to the project ID. Move the cursor over your account in the upper right corner of the management console and click **My Credential**. On the **Project List** tab page, you can view the project ID.
- **USER**: Set it to the IAM username. If the cluster is created by a user's sub-user, **user** must be set to *Sub-user.End user* when the sub-user accesses the cluster, and set to the user name when the end user accesses the cluster.
- AK: indicates the ID of the access key. You can move your cursor over your account in the upper right corner of the management console and choose **My Credential**. Click **Access Keys** tab. On the **Access Keys** tab page, you can view the existing access keys or click **Add Access Key** to add an access key.

- **TOKEN**: Generate a token using org.apache.hadoop.hbase.security.token.web.AKSKWebTokenCommonUtil#createPassword.

Step 3 (Optional) If the sample project is not used, the sample code needs to depend on the following third-party JAR files only.

1. gson
```
<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.2.4</version>
</dependency>
```
2. httpcore
```
<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpcore -->
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpcore</artifactId>
    <version>4.4.4</version>
</dependency>
```
3. httpclient
```
<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.2</version>
</dependency>
```

Step 4 A timeout interval must be set for each HTTP request using the following method:
```
public static void addTimeout(HttpRequestBase req) {
  RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(5000)
      .setConnectionRequestTimeout(10000).setSocketTimeout(60000).build();
  req.setConfig(requestConfig);
}
```

----End

4.3.2 Performing IAM Authentication for Clusters

Function Description

If IAM authentication is enabled for CloudTable, OpenTSDB must use HTTPS for connection, and the header of an HTTP request must carry parameters listed in the following table.

Table 4-5 Parameters carried in the HTTP header

HTTP Header	Value
X-TSD-IamAuth	true
X-Auth-ProjectId	ProjectID of the cluster
X-Auth-User	Tenant name
X-Auth-AK	Tenant's AccessKey
X-Auth-Token	Token information generated by using the tenant's AccessKey and SecretKey

NOTE

You can generate a token using the following method.

On the shell interface of the operating system of the client host, go to the HBase directory on the client host and run the token tool. The command format of the token tool is as follows:

./bin/hbase com.huawei.cloudtable.tool.RestTokenUtil <AccessKey> <SecretKey> <UserName>

AccessKey: User's AccessKey

SecretKey: User's SecretKey

UserName: Username

Example:

```
./bin/hbase com.huawei.cloudtable.tool.RestTokenUtil YourAccessKey YourSecretKey YourUserName
```

Sample Code

When the HTTPS connection is used, the application side does not need to verify the certificate. You can skip the certificate verification for the HTTP client that is created using the following method:

```
private static CloseableHttpClient createSSLClientDefault() {
  try {
    X509TrustManager x509mgr = new X509TrustManager() {
      public void checkClientTrusted(X509Certificate[] xcs, String string) {
      }

      public void checkServerTrusted(X509Certificate[] xcs, String string) {
      }

      public X509Certificate[] getAcceptedIssuers() {
        return null;
      }
    };
    SSLContext sslContext = SSLContext.getInstance("TLS");
    sslContext.init(null, new TrustManager[] { x509mgr }, null);
    @SuppressWarnings("deprecation")
    SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(sslContext,
        SSLConnectionSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);

    return HttpClients.custom().setSSLSocketFactory(sslsf).build();
  } catch (KeyManagementException e) {
    throw new RuntimeException(e);
  } catch (NoSuchAlgorithmException e) {
    throw new RuntimeException(e);
  } catch (Exception e) {
    throw new RuntimeException(e);
  }
}
```

When constructing an HTTP request, you need to add the required header to the HTTP request using the following method:

```
HttpPost httpPost = new HttpPost(PUT_URL);
httpPost .addHeader("X-TSD-IamAuth", "true");
httpPost .addHeader("X-Auth-ProjectId", PROJECT_ID);
httpPost .addHeader("X-Auth-User", USER);
httpPost .addHeader("X-Auth-AK", AK);
httpPost .addHeader("X-Auth-Token", TOKEN);
```


4.3.3 Writing Data

Function Description

You can use OpenTSDB APIs to write data.

Function **genWeatherData ()** simulates the generated weather data. The function **put ()** sends weather data to the OpenTSDB server.

Sample Code

```
private static String PUT_URL = (securityMode ? "https://" : "http://") + OPENTSDB_IP + ":"
     + OPENTSDB_PORT + "/api/put/?sync&sync_timeout=60000";

static class DataPoint {
  public String metric;
  public Long timestamp;
  public Double value;
  public Map<String, String> tags;
  public DataPoint(String metric, Long timestamp, Double value, Map<String, String> tags) {
    this.metric = metric;
    this.timestamp = timestamp;
    this.value = value;
    this.tags = tags;
  }
}

private String genWeatherData() {
  List<DataPoint> dataPoints = new ArrayList<DataPoint>();
  Map<String, String> tags = ImmutableMap.of("city", "Shenzhen", "region", "Longgang");

  // Data of air temperature
  dataPoints.add(new DataPoint("city.temp", 1498838400L, 28.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498842000L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498845600L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498849200L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498852800L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498856400L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498860000L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498863600L, 27.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498867200L, 29.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498870800L, 30.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498874400L, 32.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498878000L, 32.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498881600L, 33.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498885200L, 33.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498888800L, 32.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498892400L, 32.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498896000L, 31.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498899600L, 30.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498903200L, 30.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498906800L, 29.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498910400L, 29.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498914000L, 29.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498917600L, 28.0, tags));
  dataPoints.add(new DataPoint("city.temp", 1498921200L, 28.0, tags));

  // Data of humidity
  dataPoints.add(new DataPoint("city.hum", 1498838400L, 54.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498842000L, 53.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498845600L, 52.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498849200L, 51.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498852800L, 50.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498856400L, 49.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498860000L, 48.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498863600L, 46.0, tags));
```



```
  dataPoints.add(new DataPoint("city.hum", 1498867200L, 46.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498870800L, 48.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498874400L, 48.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498878000L, 49.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498881600L, 49.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498885200L, 50.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498888800L, 50.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498892400L, 50.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498896000L, 51.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498899600L, 51.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498903200L, 51.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498906800L, 51.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498910400L, 52.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498914000L, 53.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498917600L, 54.0, tags));
  dataPoints.add(new DataPoint("city.hum", 1498921200L, 54.0, tags));

  Gson gson = new Gson();
  return gson.toJson(dataPoints);
}

public void put() throws ClientProtocolException, IOException {
  try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
    HttpPost httpPost = new HttpPost(PUT_URL);
    // A timeout interval needs to be set for the request.
    addTimeout(httpPost);
    String weatherData = genWeatherData();
    StringEntity entity = new StringEntity(weatherData, "ISO-8859-1");
    entity.setContentType("application/json");
    httpPost.setEntity(entity);
    HttpResponse response = httpClient.execute(httpPost);

    int statusCode = response.getStatusLine().getStatusCode();
    System.out.println("Status Code : " + statusCode);
    if (statusCode != HttpStatus.SC_NO_CONTENT) {
      System.out.println("Request failed! " + response.getStatusLine());
    }
  }
}
```

NOTE

The **sync** parameter is added to **PUT_URL**, indicating that data can be returned only after data is written to HBase. This parameter is strongly recommended. If **sync** is not used, data is asynchronously written to HBase, which may cause data loss. For details, see **OpenTSDB API Introduction**.

4.3.4 Querying Data

Function Description

You can use the OpenTSDB query API to read data.

Function **genQueryReq ()** generates a query request, and function **query ()** sends the query request to the OpenTSDB server.

Sample Code

```
private static String QUERY_URL = (securityMode ? "https://" : "http://") + OPENTSDB_IP + ":"
    + OPENTSDB_PORT + "/api/query";

static class Query {
  public Long start;
  public Long end;
  public boolean delete = false;
```

```
  public List<SubQuery> queries;
}

static class SubQuery {
  public String metric;
  public String aggregator;
  public SubQuery(String metric, String aggregator) {
    this.metric = metric;
    this.aggregator = aggregator;
  }
}

String genQueryReq() {
  Query query = new Query();
  query.start = 1498838400L;
  query.end = 1498921200L;
  query.queries = ImmutableList.of(new SubQuery("city.temp", "sum"), new SubQuery("city.hum", "sum"));
  Gson gson = new Gson();
  return gson.toJson(query);
}

public void query() throws ClientProtocolException, IOException {
  try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
    HttpPost httpPost = new HttpPost(QUERY_URL);
    // A timeout interval needs to be set for the request.
    addTimeout(httpPost);
    String queryRequest = genQueryReq();
    //System.out.println("Request=" + queryRequest);
    StringEntity entity = new StringEntity(queryRequest, "ISO-8859-1");
    entity.setContentType("application/json");
    httpPost.setEntity(entity);
    HttpResponse response = httpClient.execute(httpPost);

    int statusCode = response.getStatusLine().getStatusCode();
    System.out.println("Status Code : " + statusCode);
    if (statusCode != HttpStatus.SC_OK) {
      System.out.println("Request failed! " + response.getStatusLine());
    }

    String body = EntityUtils.toString(response.getEntity(), "ISO-8859-1");
    System.out.println("Response content : " + body);
  }
}
```

4.3.5 Deleting Data

Function Description

Add the **delete** parameter to the query API of OpenTSDB and set it to **true**. Function **genQueryReq ()** generates a deletion request, and function **delete()** sends the deletion request to the OpenTSDB server.

Sample Code

```
private static String QUERY_URL = (securityMode ? "https://" : "http://") + OPENTSDB_IP + ":"
      + OPENTSDB_PORT + "/api/query";

static class Query {
  public Long start;
  public Long end;
  public boolean delete = false;
  public List<SubQuery> queries;
}

static class SubQuery {
  public String metric;
```



```
  public String aggregator;
  public SubQuery(String metric, String aggregator) {
    this.metric = metric;
    this.aggregator = aggregator;
  }
}

String genQueryReq() {
  Query query = new Query();
  query.start = 1498838400L;
  query.end = 1498921200L;
  query.queries = ImmutableList.of(new SubQuery("city.temp", "sum"), new SubQuery("city.hum", "sum"));
  Gson gson = new Gson();
  return gson.toJson(query);
}

String genDeleteReq() {
  Query query = new Query();
  query.start = 1498838400L;
  query.end = 1498921200L;
  query.queries = ImmutableList.of(new SubQuery("city.temp", "sum"), new SubQuery("city.hum", "sum"));
  query.delete = true;

  Gson gson = new Gson();
  return gson.toJson(query);
}

public void delete() throws ClientProtocolException, IOException {
  try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
    HttpPost httpPost = new HttpPost(QUERY_URL);
    // A timeout interval needs to be set for the request.
    addTimeout(httpPost);
    String deleteRequest = genDeleteReq();
    StringEntity entity = new StringEntity(deleteRequest, "ISO-8859-1");
    entity.setContentType("application/json");
    httpPost.setEntity(entity);
    HttpResponse response = httpClient.execute(httpPost);

    int statusCode = response.getStatusLine().getStatusCode();
    System.out.println("Status Code : " + statusCode);
    if (statusCode != HttpStatus.SC_OK) {
      System.out.println("Request failed! " + response.getStatusLine());
    }
  }
}
```

NOTE

If **query.delete** is set to **true**, all queried data will be deleted. For details, see **OpenTSDB API Introduction**.

4.4 Tuning Performance

You can scale out the throughput of the OpenTSDB service by adding tsd nodes. Currently, the Round Robin DNS mechanism is used to load application requests to different tsd nodes. You can optimize the DNS domain name resolution cache on applications to evenly distribute HTTP requests from the applications at better fine-grained rate.

JVM Optimization

- Disable the DNS domain name cache of JVM in code, and use an HTTP connection pool to send requests.

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "0")
```

```
private static CloseableHttpClient httpClient;
static {
  PoolingHttpClientConnectionManager cm = new PoolingHttpClientConnectionManager();
  cm.setMaxTotal(200);
  cm.setDefaultMaxPerRoute(20);
  cm.setDefaultMaxPerRoute(50);
  httpClient = HttpClients.custom().setConnectionManager(cm).build();
}
```

- Set an interval of the DNS domain name cache of JVM in code to 1 second.

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
```

- If you want to make the new IP address be identified without restarting the application after capacity expansion, you can set a connection TTL, but performance will be affected.

```
HttpClients.custom().setConnectionTimeToLive(600, TimeUnit.SECONDS);
```

OS Optimization

- Disable the local DNS domain name resolution cache in the Linux OS. Check whether **nscd**, **named**, or **dnsmasq** exists in the **/etc/init.d/** directory. If it exists, run the following commands to disable the cache:

  ```
  /etc/init.d/nscd stop
  /etc/init.d/named stop
  /etc/init.d/dnsmasq stop
  ```

- Disable the local DNS domain name resolution cache in the Windows OS. Run the following command in cmd:

  ```
  net stop dnscache
  ```


5 Developing GeoMesa Applications

5.1 Typical Application Scenario

You can quickly learn and master the GeoMesa development process and know key API functions in a typical application scenario.

Description

Assume that a user develops an application to record and query information about dating friends, for example, when and where has the user done what with which friend? The following table provides the data record format.

Field	Field Type
Who	String
What	Long
When	Date
Where	Point
Why	String

5.2 Development Guideline

Determine functions to be developed based on the preceding scenario. The following table describes functions to be developed.

No.	Procedure	Code Implementation
1	Connect to a GeoMesa cluster.	For details, see **Creating a DataStore**.

No.	Procedure	Code Implementation
2	Create a schema.	For details, see **Creating a Schema**.
3	Insert data.	For details, see **Inserting Data**.
4	Query data.	For details, see **Querying Data**.

5.3 Sample Code Description

5.3.1 Parameter Configuration

Step 1 Before executing sample code, configure the correct ZooKeeper cluster address in the **hbase-site.xml** configuration file.

The configuration items are as follows:

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>xxx-zk1.cloudtable.com,xxx-zk2.cloudtable.com,xxx-zk3.cloudtable.com</value>
</property>
```

value is the domain name of the ZooKeeper cluster. Log in to the CloudTable management console and choose **Cluster Mode**. In the cluster list, locate the required cluster and obtain its ZK link in the **ZK Link** column.

----End

5.3.2 Creating a DataStore

Function Description

You can connect to the GeoMesa cluster to create a DataStore instance by using GeoMesa APIs. The DataStore instance provides APIs for operations on the GeoMesa specified directories.

Sample Code

```
// find out where -- in HBase -- the user wants to store data
CommandLineParser parser = new BasicParser();
Options options = getCommonRequiredOptions();
CommandLine cmd = parser.parse(options, new String[]{"--bigtable_table_name", "geomesa"});
// verify that we can see this HBase destination in a GeoTools manner
Map<String, Serializable> dsConf = getHBaseDataStoreConf(cmd);
DataStore dataStore = DataStoreFinder.getDataStore(dsConf);
```

5.3.3 Creating a Schema

Function Description

You can create a schema in the DataStore of the specified directory.

Sample Code

```
// establish specifics concerning the SimpleFeatureType to store
String simpleFeatureTypeName = "QuickStart";
// list the attributes that constitute the feature type
String attributes = "Who:String,What:java.lang.Long,When:Date,*Where:Point:srid=4326,Why:String";
// create the bare simple-feature type
SimpleFeatureType simpleFeatureType = SimpleFeatureTypes.createType(simpleFeatureTypeName,
attributes);
dataStore.createSchema(simpleFeatureType);
```

5.3.4 Inserting Data

Function Description

You can construct data and insert data into the specified table.

Sample Code

```
DefaultFeatureCollection featureCollection = new DefaultFeatureCollection();
String id;
Object[] NO_VALUES = {};
String[] PEOPLE_NAMES = {"Addams", "Bierce", "Clemens"};
Long SECONDS_PER_YEAR = 365L * 24L * 60L * 60L;
Random random = new Random(5771);
DateTime MIN_DATE = new DateTime(2014, 1, 1, 0, 0, 0, DateTimeZone.forID("UTC"));
Double MIN_X = -79.5;
Double MIN_Y = 37.0;
Double DX = 2.0;
Double DY = 2.0;
for (int i = 0; i < numNewFeatures; i++) {
  // create the new (unique) identifier and empty feature shell
  id = "Observation." + Integer.toString(i);
  SimpleFeature simpleFeature = SimpleFeatureBuilder.build(simpleFeatureType, NO_VALUES, id);
  // be sure to tell GeoTools explicitly that you want to use the ID you provided
  simpleFeature.getUserData().put(Hints.USE_PROVIDED_FID, java.lang.Boolean.TRUE);
  // populate the new feature's attributes
  // Who: string value
  simpleFeature.setAttribute("Who", PEOPLE_NAMES[i % PEOPLE_NAMES.length]);
  // What: long value
  simpleFeature.setAttribute("What", i);
  // Where: location: construct a random point within a 2-degree-per-side square
  double x = MIN_X + random.nextDouble() * DX;
  double y = MIN_Y + random.nextDouble() * DY;
  Geometry geometry = WKTUtils$.MODULE$.read("POINT(" + x + " " + y + ")");
  simpleFeature.setAttribute("Where", geometry);
  // When: date-time:construct a random instant within a year
  DateTime dateTime = MIN_DATE.plusSeconds((int) Math.round(random.nextDouble() *
SECONDS_PER_YEAR));
  simpleFeature.setAttribute("When", dateTime.toDate());
  // Why: another string value
  // left empty, showing that not all attributes need values
  // accumulate this new feature in the collection
  featureCollection.add(simpleFeature);
  FeatureStore featureStore = (FeatureStore) dataStore.getFeatureSource(simpleFeatureTypeName);
  featureStore.addFeatures(featureCollection);
}
```

5.3.5 Querying Data

Function Description

You can query data based on the specified time and space conditions.

Sample Code

```
// construct a (E)CQL filter from the search parameters,
// and use that as the basis for the query
Filter cqlFilter = createFilter(geomField, x0, y0, x1, y1, dateField, t0, t1, attributesQuery);
Query query = new Query(simpleFeatureTypeName, cqlFilter);
List<String> properties = new LinkedList<>();
// submit the query, and get back an iterator over matching features
FeatureSource featureSource = dataStore.getFeatureSource(simpleFeatureTypeName);
FeatureCollection featureCollection = featureSource.getFeatures(query);
FeatureIterator featureItr = featureCollection.features();
// loop through all results
int n = 0;
while (featureItr.hasNext()) {
  Feature feature = featureItr.next();
  feature.getIdentifier().getID();
  System.out.println((++n) + "." +
    feature.getProperty("Who").getValue() + "|" +
    feature.getProperty("What").getValue() + "|" +
    feature.getProperty("When").getValue() + "|" +
    feature.getProperty("Where").getValue() + "|" +
    feature.getProperty("Why").getValue());
}
featureItr.close();
```


6 Developing HBase Elasticsearch Full-Text Search Applications

6.1 Application Background

HBase-Elasticsearch stores user source data in HBase and uses the Elasticsearch search engine of Cloud Search Service (CSS for short) to supplement full-text search based on key-value query capabilities. You can define which fields in HBase need full-text search based on service requirements. When you create an HBase table, a CSS cluster you specify will be automatically connected and an index is created in Elasticsearch. Index data is stored in Elasticsearch. In addition, the native APIs (Put and Scan) of HBase support the write and query of index data.

6.2 Prerequisites

The created CloudTable cluster (HBase), ECS instance (functioning as an HBase client), and CSS cluster (Elasticsearch engine) must have the same VPC, subnet, and security group to ensure network connectivity.

For details about Elasticsearch, see **https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html**.

6.3 Typical Application Scenario

A professional document company uses HBase to collect information such as the document content, word count, score, and grade. A keyword search function is required to search for **contentCh** (Chinese content) and **contentEng** (English content) in the table.

The following table describes the document content fields.

Table 6-1 Field description

Field	Type of Field Mapped to Elasticsearch	Full-Text Index Required	Field Description
contentCh	text	Yes	Chinese content of the document
contentEng	text	Yes	Chinese content of the document
id	long	No	Document issue ID
charNum	int	No	Total number of words in the document
pageNum	short	No	Number of pages
level	byte	No	Document level
researchCost	double	No	Research expense
score	float	No	Document score
male	boolean	No	Author's gender
whatever	-	No	Other fields in the HBase table, but not mapped to Elasticsearch

6.4 HBase Elasticsearch Schema

HBase uses metadata of a table to store the definition of the Elasticsearch schema.

Table 6-2 Schema definition

Field	Description	Mandatory
hbase.index.es.enabled	Whether to create a full-text index for the HBase table in Elasticsearch. The value **true** indicates that the full-text index is created. The default value is **false**.	Yes
hbase.index.es.endpoint	Access address of the CSS cluster (Elasticsearch engine), for example, **ip1:port,ip2:port**	Yes
hbase.index.es.indexname	Index name of the HBase table in Elasticsearch. The index name must be in lower case.	Yes
hbase.index.es.shards	Number of index shards in Elasticsearch. The default value is **5**. The value is an integer greater than or equal to 1.	No

Field	Description	Mandatory
hbase.index.es.replicas	Number of index replicas in Elasticsearch. The default value is **1**. The value is an integer greater than or equal to 0.	No
hbase.index.es.schema	Field mapping between HBase and Elasticsearch. The value is characters in JSON array format. Each element contains the following fields: • **name**: Name of the field in Elasticsearch • **type**: Type of the field in Elasticsearch • **hbaseQualifier**: HBase qualifier of the data source • **analyzer**: You can configure **analyzer** to specify an analyzer for fields of the text type. Typically, the **ik_smart** analyzer is used for Chinese text. The default value is **Standard**, supporting English text. Example: '[{"name":"contentCh","type":"text","hbaseQualifier":"cf1:contentCh","analyzer":"ik_smart"}, {"name":"contentEng","type":"text","hbaseQualifier":"cf2:contentEng"},{"name":"id","type":"long","hbaseQualifier":"cf1:id"}]'	Yes

The data types supported by HBase-Elasticsearch full-text search are {"text", "long", "integer", "short", "byte", "double", "float","boolean"}, that is, the value type of **type** in the schema. **text** indicates the text type in Elasticsearch. Full-text search typically supports data of the text type and also supports accurate search of data of basic types.

6.5 Development Guideline

Table 6-3 Development guideline

No.	Procedure	Code Implementation
1	Enable an index in Elasticsearch when creating an HBase table.	For details, see **Enabling an Index in Elasticsearch When Creating a Table**.
2	Write data using HBase Put.	For details, see **Writing Data**.
3	Query data using HBase Scan.	For details, see **Querying Data**.

6.6 Sample Code Description

6.6.1 Parameter Configuration

Step 1 Before executing sample code, configure the correct ZooKeeper cluster address in the **hbase-site.xml** configuration file.

The configuration items are as follows:

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>xxx-zk1.cloudtable.com,xxx-zk2.cloudtable.com,xxx-zk3.cloudtable.com</value>
</property>
```

value is the domain name of the ZooKeeper cluster. Log in to the CloudTable management console and choose **Cluster Mode**. In the cluster list, locate the required cluster and obtain its ZK link in the **ZK Link** column.

Step 2 Modify the parameters related to IAM authentication in the sample code project.

- If IAM authentication is enabled for the CloudTable cluster:

  In the sample code project, **IAM_AUTH_MODE** in the **com.huawei.cloudtable.hbase.examples.ES.HBaseESTestMain** class must be set to **true** and the **user**, **ak,** and **sk** parameters need to be configured.

  The code is as follows:

  ```
  private static boolean IAM_AUTH_MODE = true;
  private static String user = "XXXXXX";
  private static String ak = "XXXXXX";
  private static String sk = "XXXXXX";
  ```

  - **user**: username If the cluster is created by a user's sub-user, **user** must be set to *Sub-user.End user* when the sub-user accesses the cluster, and set to the user name when the end user accesses the cluster.
  - **ak** and **sk**: Access Key ID (AK) and Secret Access Key (SK). Set them to the AK plaintext and SK plaintext, respectively. You can move your cursor over your account in the upper right corner of the management console and choose **My Credential**. Click **Access Keys** tab. On the **Access Keys** tab page, you can view the existing access keys or click **Add Access Key** to add an access key.

  NOTE

  The IAM authentication mode provides better security than the normal mode. Therefore, you are advised to enable IAM authentication for the CloudTable cluster and use IAM authentication in client or application code to connect to the cluster.

- If IAM authentication is disabled for the CloudTable cluster:

  IAM_AUTH_MODE in the **com.huawei.cloudtable.hbase.examples.ES.HBaseESTestMain** class must be set to **false**.

----End

6.6.2 Creating the Configuration Object

Function Description

HBase obtains configuration items by loading a configuration file.

NOTE

1. Loading the configuration file is time-consuming. If unnecessary, use the same Configuration object.
2. Multi-thread synchronization is not considered in the sample code. If necessary, add it by yourself. Other sample codes are the same.

Sample Code

The following code snippets are in the **com.huawei.cloudtable.hbase.examples.ES.HBaseESTestMain** packet.

```
private static void init() throws IOException {
  // Default load from conf directory
  conf = HBaseConfiguration.create(); // Note [1]
  String userdir = System.getProperty("user.dir") + File.separator + "conf" + File.separator;
  Path hbaseSite = new Path(userdir + "hbase-site.xml");
  if (new File(hbaseSite.toString()).exists()) {
    conf.addResource(hbaseSite);
  }
}
```

Precautions

- Note [1] If the **conf** directory of the configuration file is added to the **classpath** path, the code for loading the specified configuration file can be skipped.

6.6.3 Enabling an Index in Elasticsearch When Creating a Table

Function Description

Follow instructions in **Creating a Table** to create a table. To configure table properties, follow instructions in **HBase Elasticsearch Schema** to configure the field to enable an index in Elasticsearch.

Sample Code

```
public void createTable() {
  LOG.info("Entering testCreateTable.");

  // Specify the table descriptor.
  HTableDescriptor htd = new HTableDescriptor(tableName);

  // Set the column family name to info.
  HColumnDescriptor hcd = new HColumnDescriptor(CF1);

  // Set data encoding methods. HBase provides DIFF,FAST_DIFF,PREFIX
  // and PREFIX_TREE
  hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);

  // Set compression methods, HBase provides two default compression
  // methods:GZ and SNAPPY
  // GZ has the highest compression rate,but low compression and
  // decompression efficiency,fit for cold data
  // SNAPPY has low compression rate, but high compression and
  // decompression efficiency,fit for hot data.
  // it is advised to use SANPPY
  hcd.setCompressionType(Compression.Algorithm.SNAPPY);
```



```
    HColumnDescriptor hcd2 = new HColumnDescriptor(CF2);
    hcd2.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);
    hcd2.setCompressionType(Compression.Algorithm.SNAPPY);

    htd.addFamily(hcd);
    htd.addFamily(hcd2);

    //add HBase ES schema
    String ESJsonSchema = "[" +
      "{\"name\":\"contentCh\",\"type\":\"text\",\"hbaseQualifier\":\"cf1:contentCh\",\"analyzer\":\"ik_smart
\"}," +
      "{\"name\":\"contentEng\",\"type\":\"text\",\"hbaseQualifier\":\"cf2:contentEng\"}," +
      "{\"name\":\"id\",\"type\":\"long\",\"hbaseQualifier\":\"cf1:id\"}," +
      "{\"name\":\"charNum\",\"type\":\"integer\",\"hbaseQualifier\":\"cf1:charNum\"}," +
      "{\"name\":\"pageNum\",\"type\":\"short\",\"hbaseQualifier\":\"cf1:pageNum\"}," +
      "{\"name\":\"level\",\"type\":\"byte\",\"hbaseQualifier\":\"cf1:level\"}," +
      "{\"name\":\"researchCost\",\"type\":\"double\",\"hbaseQualifier\":\"cf1:researchCost\"}," +
      "{\"name\":\"score\",\"type\":\"float\",\"hbaseQualifier\":\"cf1:score\"}," +
      "{\"name\":\"male\",\"type\":\"boolean\",\"hbaseQualifier\":\"cf1:male\"}" +
      "]";
    htd.setValue(HBaseESConst.HBASE_INDEX_ES_ENABLED, "true");
    htd.setValue(HBaseESConst.HBASE_INDEX_ES_ENDPOINT, ESClusterHosts);//(1)
    htd.setValue(HBaseESConst.HBASE_INDEX_ES_INDEXNAME, ES_INDEX_NAME);//(2)
    htd.setValue(HBaseESConst.HBASE_INDEX_ES_SCHEMA, ESJsonSchema);//(3)

    Admin admin = null;
    try {
      // Instantiate an Admin object.
      admin = conn.getAdmin();
      if (!admin.tableExists(tableName)) {
        LOG.info("Creating table...");
        admin.createTable(htd);
        LOG.info(admin.getClusterStatus());
        LOG.info(admin.listNamespaceDescriptors());
        LOG.info("Table created successfully.");
      } else {
        LOG.warn("table already exists");
      }
    } catch (IOException e) {
      LOG.error("Create table failed.", e);
    } finally {
      if (admin != null) {
        try {
          // Close the Admin object.
          admin.close();
        } catch (IOException e) {
          LOG.error("Failed to close admin ", e);
        }
      }
    }
    LOG.info("Exiting testCreateTable.");
  }
```

Note:

(1) Indicates the CSS (Elasticsearch engine) cluster access address obtained from the CSS management console.

(2) Indicates the user-defined index name.

(3) Indicates the field mapping between HBase and Elasticsearch.

6.6.4 Writing Data

Function Description

Follow instructions in **Inserting Data** to write data. Note that the data to be written must contain all fields of the index in Elasticsearch. If not all fields are contained, an Elasticsearch index data exception may occur. The user client needs to process **ESDocIndexException** based on business scenario requirements. The user can retry or ignore or perform other operations.

Sample Code

The following is sample code. Replace ****replace_with_Chinese_characters**** in code with Chinese characters.

```
public void putData() {
  LOG.info("Entering testPut.");

  Table table = null;
  try {
    // Instantiate an HTable object.
    table = conn.getTable(tableName);
    List<Put> puts = new ArrayList<Put>();
    // Instantiate a Put object.
    Put put = new Put(Bytes.toBytes("rowkey001"));
    put.addColumn(CF1, QUA_ARTICLE_CONTENT_CHINESE,
Bytes.toBytes("***replace_with_Chinese_characters***"));
    put.addColumn(CF2, QUA_ARTICLE_CONTENT_English, Bytes.toBytes("how many apples in the
market"));
    put.addColumn(CF1, QUA_ARTICLE_ID, Bytes.toBytes(1111l));
    put.addColumn(CF1, QUA_CHARACTER_NUM, Bytes.toBytes(1));
    short shortNum = 1;
    put.addColumn(CF1, QUA_PAGE_NUM, Bytes.toBytes(shortNum));
    char c = 'A';
    put.addColumn(CF1, QUA_ARTICLE_LEVEL, new byte[]{(byte)c});
    put.addColumn(CF1, QUA_RESEARCH_COST, Bytes.toBytes(111.11d));
    put.addColumn(CF1, QUA_ARTICLE_SCORE, Bytes.toBytes(80.5f));
    put.addColumn(CF1, QUA_AUTHOR_MALE, Bytes.toBytes(true));
    put.addColumn(CF1, QUA_WHATEVER, Bytes.toBytes("happy life and sweet love"));
    puts.add(put);

    put = new Put(Bytes.toBytes("rowkey002"));
    put.addColumn(CF1, QUA_ARTICLE_CONTENT_CHINESE,
Bytes.toBytes("***replace_with_Chinese_characters***"));
    put.addColumn(CF2, QUA_ARTICLE_CONTENT_English, Bytes.toBytes("how many people in the
swimming pool"));
    put.addColumn(CF1, QUA_ARTICLE_ID, Bytes.toBytes(2222l));
    put.addColumn(CF1, QUA_CHARACTER_NUM, Bytes.toBytes(2));
    shortNum = 2;
    put.addColumn(CF1, QUA_PAGE_NUM, Bytes.toBytes(shortNum));
    c = 'B';
    put.addColumn(CF1, QUA_ARTICLE_LEVEL, new byte[]{(byte)c});
    put.addColumn(CF1, QUA_RESEARCH_COST, Bytes.toBytes(222.22d));
    put.addColumn(CF1, QUA_ARTICLE_SCORE, Bytes.toBytes(170.5f));
    put.addColumn(CF1, QUA_AUTHOR_MALE, Bytes.toBytes(true));
    put.addColumn(CF1, QUA_WHATEVER, Bytes.toBytes("forever young"));
    puts.add(put);

    put = new Put(Bytes.toBytes("rowkey003"));
    put.addColumn(CF1, QUA_ARTICLE_CONTENT_CHINESE,
Bytes.toBytes("***replace_with_Chinese_characters***"));
    put.addColumn(CF2, QUA_ARTICLE_CONTENT_English, Bytes.toBytes("we play video game in night"));
    put.addColumn(CF1, QUA_ARTICLE_ID, Bytes.toBytes(3333l));
    put.addColumn(CF1, QUA_CHARACTER_NUM, Bytes.toBytes(3));
    shortNum = 3;
```



```
    put.addColumn(CF1, QUA_PAGE_NUM, Bytes.toBytes(shortNum));
    c = 'C';
    put.addColumn(CF1, QUA_ARTICLE_LEVEL, new byte[]{(byte)c});
    put.addColumn(CF1, QUA_RESEARCH_COST, Bytes.toBytes(333.33d));
    put.addColumn(CF1, QUA_ARTICLE_SCORE, Bytes.toBytes(180.5f));
    put.addColumn(CF1, QUA_AUTHOR_MALE, Bytes.toBytes(false));
    put.addColumn(CF1, QUA_WHATEVER, Bytes.toBytes("wishes always with you"));
    puts.add(put);

    // Submit a put request.
    try {
      table.put(puts);
    }
    // if your put operation does not contain the all field in your schema, you will get ESDocIndexException.
    // ESDocIndexException means data/document indexing to ES failed, but remember data put in HBase success.
    // so here you handle this exception depends on your business( you can retry or ignore).
    catch (ESDocIndexException e) {
      //TODO
    }

    LOG.info("Put successfully.");
  } catch (IOException e) {
    LOG.error("Put failed ", e);
  } finally {
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed ", e);
      }
    }
  }
  LOG.info("Exiting testPut.");
 }
```

6.6.5 Querying Data

Function Description

HBase uses the native Scan API to support the full-text index, which is similar to that in **Reading Data Using Scan**. In addition, you need to set **HBASE_CLIENT_CONNECTION_IMPL** in **configuration** to **org.apache.hadoop.hbase.client.LemonConnectionImplementation** and transfer full-text search criteria to the filter **ESColumnValueFilter** customized by CloudTable.

Sample Code

1. In **configuration**, **HBASECLIENTCONNECTION_IMPL** is defined as the **LemonConnectionImplementation** implementation class.

   The sample code is as follows:

   ```
   conf = HBaseConfiguration.create();
   conf.set(HConnection.HBASE_CLIENT_CONNECTION_IMPL,
   "org.apache.hadoop.hbase.client.LemonConnectionImplementation");
   Connection conn = ConnectionFactory.createConnection(conf);
   ```

2. The full-text search criteria are transferred through **httpParameters** or **reqBodyJson** of **ESColumnValueFilter**. The parameter format is the same as the official document of the open source Elasticsearch. For details, see **URI Search** and **Request Body Search**.

The following code uses **httpParameters** for query. (Replace ****replace_with_Chinese_keywords_01**** in code with Chinese keywords.)

```
public void testScanDataWithES1() {
  LOG.info("Entering testScanDataWithES1.");

  Table table = null;
  // Instantiate a ResultScanner object.
  ResultScanner rScanner = null;
  try {
    // Create the Configuration instance.
    table = conn.getTable(tableName);
    assert table instanceof LemonWrapperHTable;

    // set specified qualifier.
    Scan scan = new Scan();
    scan.addColumn(CF1, QUA_ARTICLE_CONTENT_CHINESE);
    scan.addColumn(CF2, QUA_ARTICLE_CONTENT_English);
    scan.addColumn(CF1, QUA_ARTICLE_ID);

    // Set the cache size.
    scan.setCaching(1000);

    // with special filter
    Map<String, String> httpParameters = new HashMap<>();
    httpParameters.put("q", "contentCh:***replace_with_Chinese_keywords_01***");
    List<String> indexNames = new ArrayList();
    indexNames.add(ES_INDEX_NAME);
    ESColumnValueFilter filter = new ESColumnValueFilter(indexNames, httpParameters, "");
    scan.setFilter(filter);

    // Submit a scan request.
    rScanner = table.getScanner(scan);

    // Print query results.
    for (Result r = rScanner.next(); r != null; r = rScanner.next()) {
      for (Cell cell : r.rawCells()) {
        LOG.info("testScanDataWithES1 by text type condition, scan result:" +
Bytes.toString(CellUtil.cloneRow(cell)) + ":"
          + Bytes.toString(CellUtil.cloneFamily(cell)) + ","
          + Bytes.toString(CellUtil.cloneQualifier(cell)) + ","
          + convertCellValue(Bytes.toString(CellUtil.cloneQualifier(cell)), CellUtil.cloneValue(cell)));
      }
    }

    LOG.info("Scan data successfully.");
  } catch (IOException e) {
    LOG.error("Scan data failed ", e);
  } finally {
    if (rScanner != null) {
      // Close the scanner object.
      rScanner.close();
    }
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed ", e);
      }
    }
  }
  LOG.info("Exiting testScanDataWithES1.");
}
```

The following code uses **reqBodyJson** for query. (Replace ****replace_with_Chinese_keywords_02**** in code with Chinese keywords.)

```
public void testScanDataWithES2() {
  LOG.info("Entering testScanDataWithES2.");
```



```
  Table table = null;
  // Instantiate a ResultScanner object.
  ResultScanner rScanner = null;
  try {
    // Create the Configuration instance.
    table = conn.getTable(tableName);
    assert table instanceof LemonWrapperHTable;

    // set specified qualifier.
    Scan scan = new Scan();
    scan.addColumn(CF1, QUA_ARTICLE_CONTENT_CHINESE);
    scan.addColumn(CF2, QUA_ARTICLE_CONTENT_English);
    scan.addColumn(CF1, QUA_ARTICLE_ID);

    // Set the cache size.
    scan.setCaching(1000);

    // with special filter
    Map<String, String> httpParameters = Collections.emptyMap();
    List<String> indexNames = new ArrayList();
    indexNames.add(ES_INDEX_NAME);
    String reqBodyJson =
      "{" +
        "    \"query\" : {" +
        "        \"term\" : { \"contentCh\" : \"***replace_with_Chinese_keywords_02***\" }" +
        "    }" +
        "}";
    ESColumnValueFilter filter = new ESColumnValueFilter(indexNames, httpParameters,
reqBodyJson);
    scan.setFilter(filter);

    // Submit a scan request.
    rScanner = table.getScanner(scan);

    // Print query results.
    for (Result r = rScanner.next(); r != null; r = rScanner.next()) {
      for (Cell cell : r.rawCells()) {
        LOG.info("testScanDataWithES2 by text type condition, scan result:" +
Bytes.toString(CellUtil.cloneRow(cell)) + ":"
          + Bytes.toString(CellUtil.cloneFamily(cell)) + ","
          + Bytes.toString(CellUtil.cloneQualifier(cell)) + ","
          + convertCellValue(Bytes.toString(CellUtil.cloneQualifier(cell)), CellUtil.cloneValue(cell)));
      }
    }

    LOG.info("Scan data successfully.");
  } catch (IOException e) {
    LOG.error("Scan data failed ", e);
  } finally {
    if (rScanner != null) {
      // Close the scanner object.
      rScanner.close();
    }
    if (table != null) {
      try {
        // Close the HTable object.
        table.close();
      } catch (IOException e) {
        LOG.error("Close table failed ", e);
      }
    }
  }

  LOG.info("Exiting testScanDataWithES2.");
 }
```


7 Commissioning Applications

7.1 Commissioning Applications on Windows

7.1.1 Compiling and Running Applications

Scenario

You can run applications in the Windows development environment after application code development is complete.

Procedure

In the development environment (for example, Eclipse), right-click **TestMain.java** and choose **Run as** > **Java Application** from the shortcut menu to run the corresponding application project.

7.1.2 Viewing Commissioning Results

If no exception or failure information is displayed, running the application is successful.

NOTE

The following exception may occur when the sample code is running in the Windows OS, but it will not affect services.

java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.

Log description:

The log level is INFO by default and you can view more detailed information by changing the log level, such as DEBUG, INFO, WARN, ERROR, and FATAL. You can modify the **log4j.properties** file to change log levels, for example:

```
hbase.root.logger=INFO,console
log4j.logger.org.apache.zookeeper=INFO
#log4j.logger.org.apache.hadoop.fs.FSNamesystem=DEBUG
log4j.logger.org.apache.hadoop.hbase=INFO
```



```
# Make these two classes DEBUG-level. Make them DEBUG to see more zk debug.
log4j.logger.org.apache.hadoop.hbase.zookeeper.ZKUtil=INFO
log4j.logger.org.apache.hadoop.hbase.zookeeper.ZooKeeperWatcher=INFO
```

7.2 Commissioning Applications on Linux

7.2.1 Compiling and Running an Application When a Client Is Installed

Scenario

HBase applications can run in a Linux environment where an HBase client is installed. After application code development is complete, you can upload a JAR file to the Linux environment to run applications.

Prerequisites

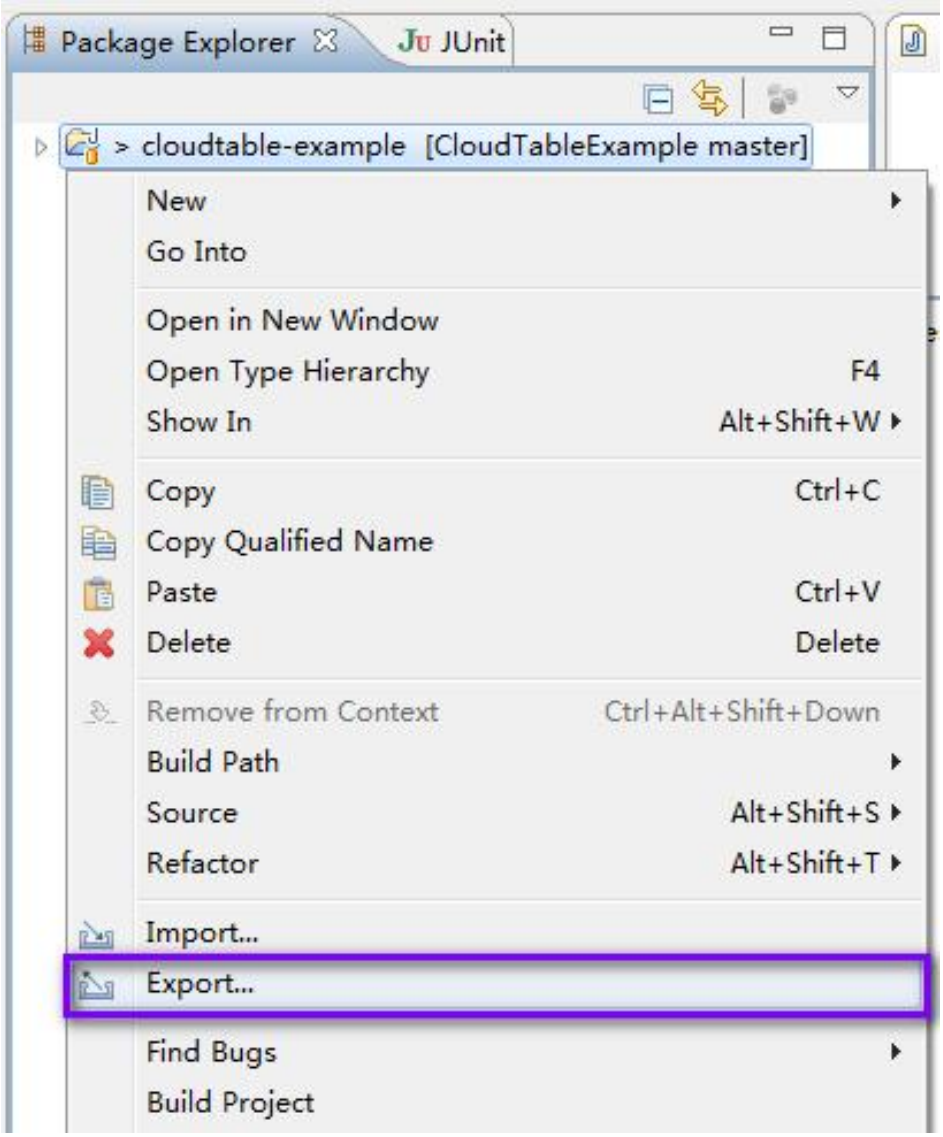
- You have installed an HBase client.
- You have installed a JDK in the Linux environment. The version of the JDK must be consistent with that of the JDK used by Eclipse to export the JAR file.

Procedure

Step 1 Export a JAR file.

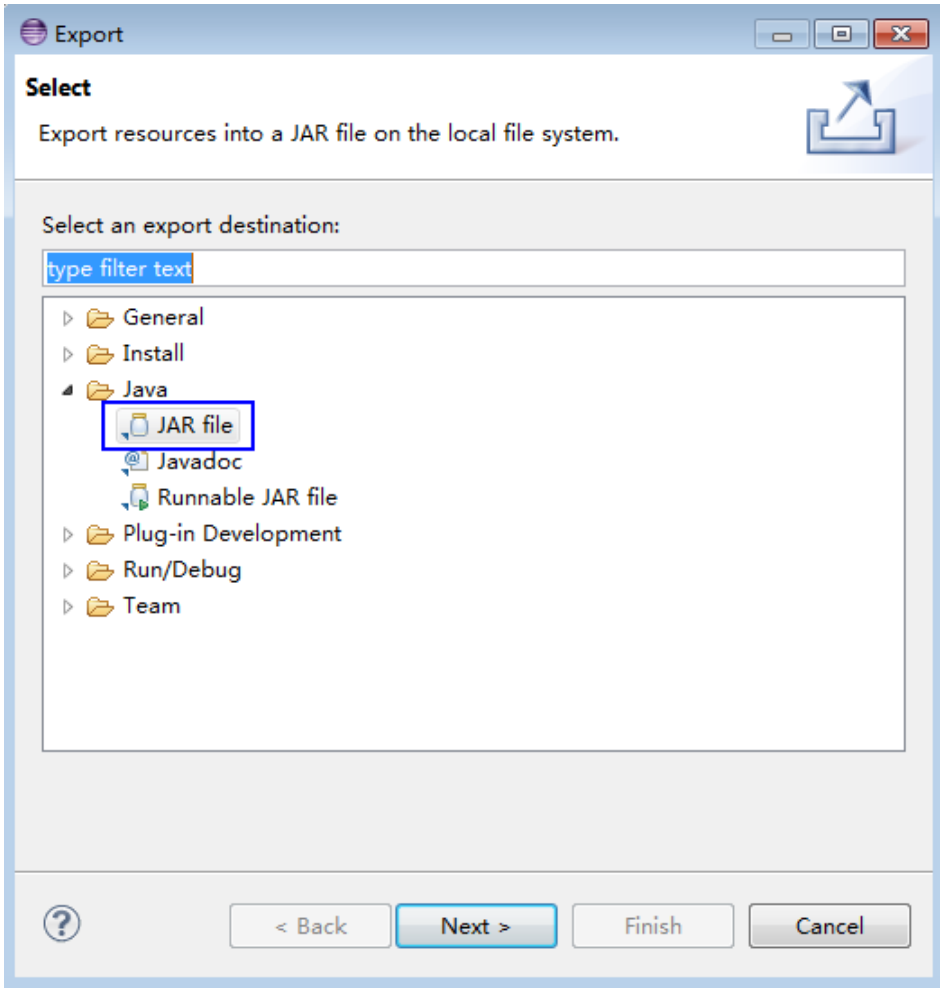
1. Right-click the sample project and choose **Export** from the shortcut menu.

Figure 7-1 Exporting a JAR file



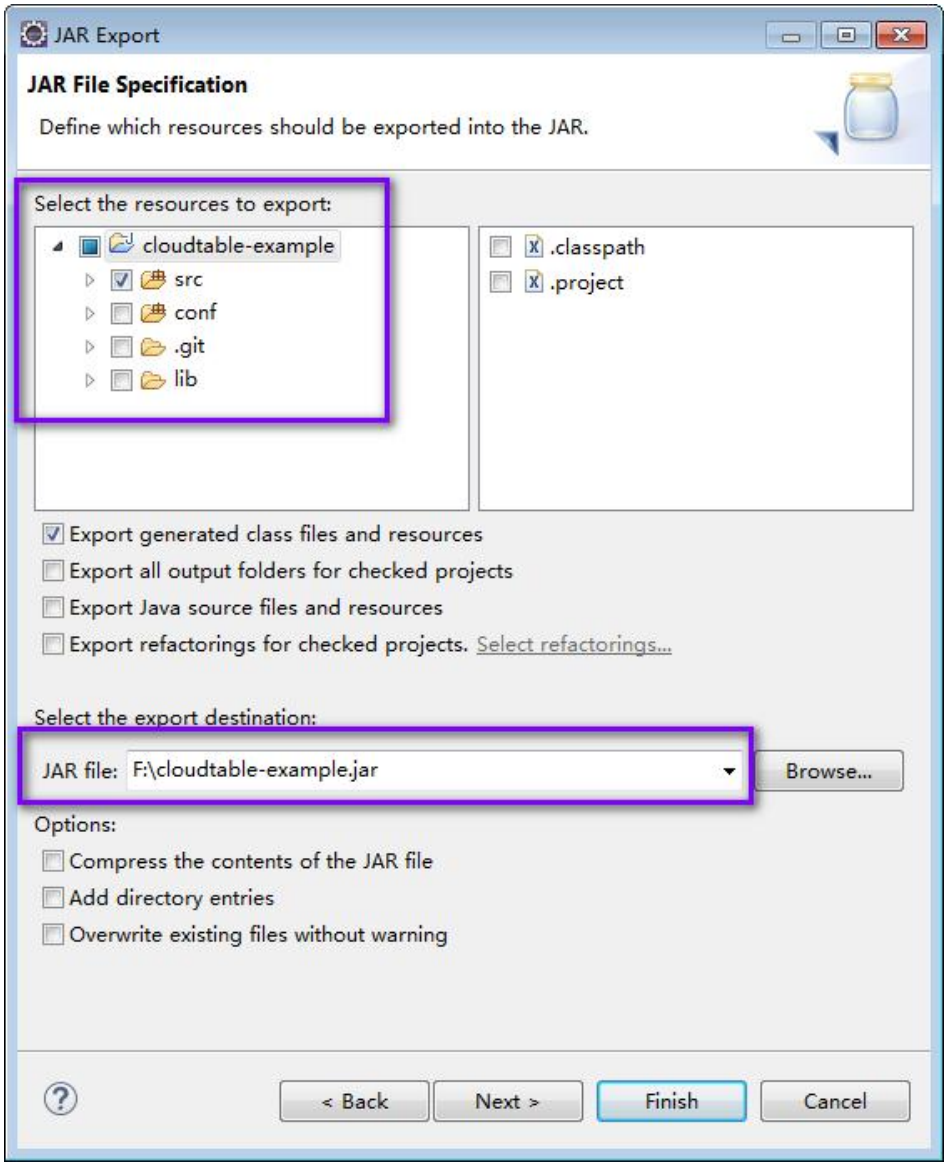
2. Select **JAR file** and click **Next**.

Figure 7-2 Selecting JAR file



3. Select the **src** and **conf** directories, and export the JAR file to the specified location. Click **Next** twice.

Figure 7-3 Selecting a path for exporting the JAR file



4. Click **Finish**. Exporting the JAR file is complete.

Step 2 Run the JAR file.

1. Before running the JAR file on the Linux client, copy the JAR file generated in the application development environment to the **lib** directory in the client installation directory, and ensure that the permission of the JAR file is the same as that of other files.
2. Switch to the **bin** directory in the client directory as the installation user, and then run the following command to Run the JAR file:

```
[Ruby@cloudtable-08261700-hmaster-1-1 bin]# ./hbase
com.huawei.cloudtable.hbase.examples.TestMain
```

com.huawei.cloudtable.hbase.examples.TestMain is used as an example. Use the actual code instead.

----End

7.2.2 Compiling and Running an Application When No Client Is Installed

Scenario

HBase applications can run in a Linux environment where an HBase client is not installed. After application code development is complete, you can upload a JAR file to the Linux environment to run applications.

Prerequisites

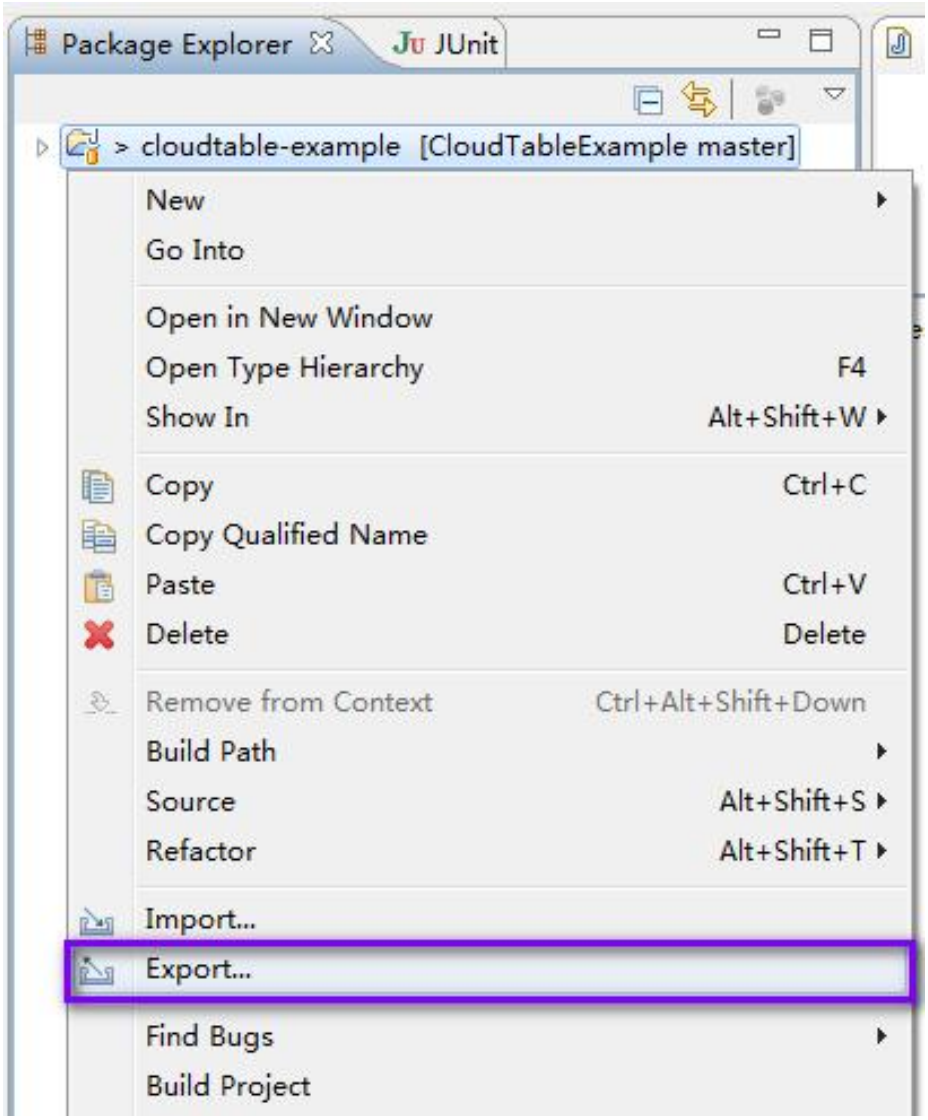
- You have installed a JDK in the Linux environment. The version of the JDK must be consistent with that of the JDK used by Eclipse to export the JAR file.

Procedure

Step 1 Export a JAR file.

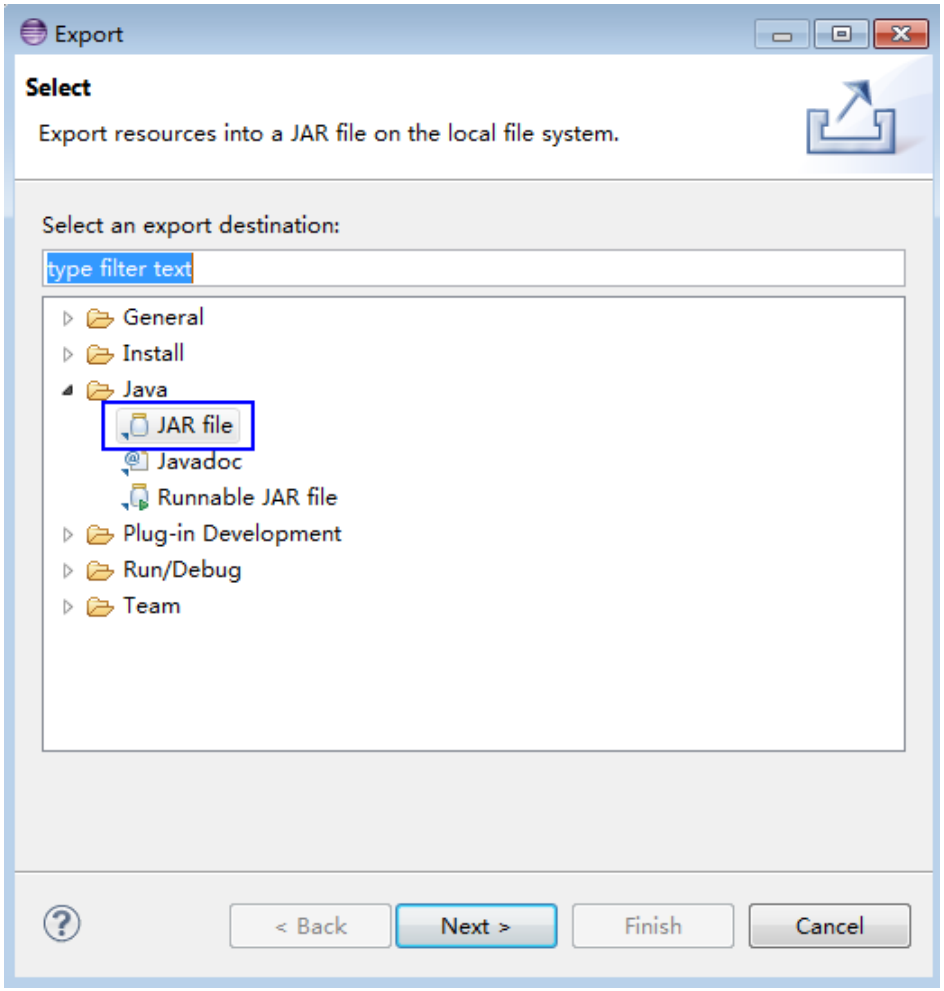
1. Right-click the sample project and choose **Export** from the shortcut menu.

Figure 7-4 Exporting a JAR file



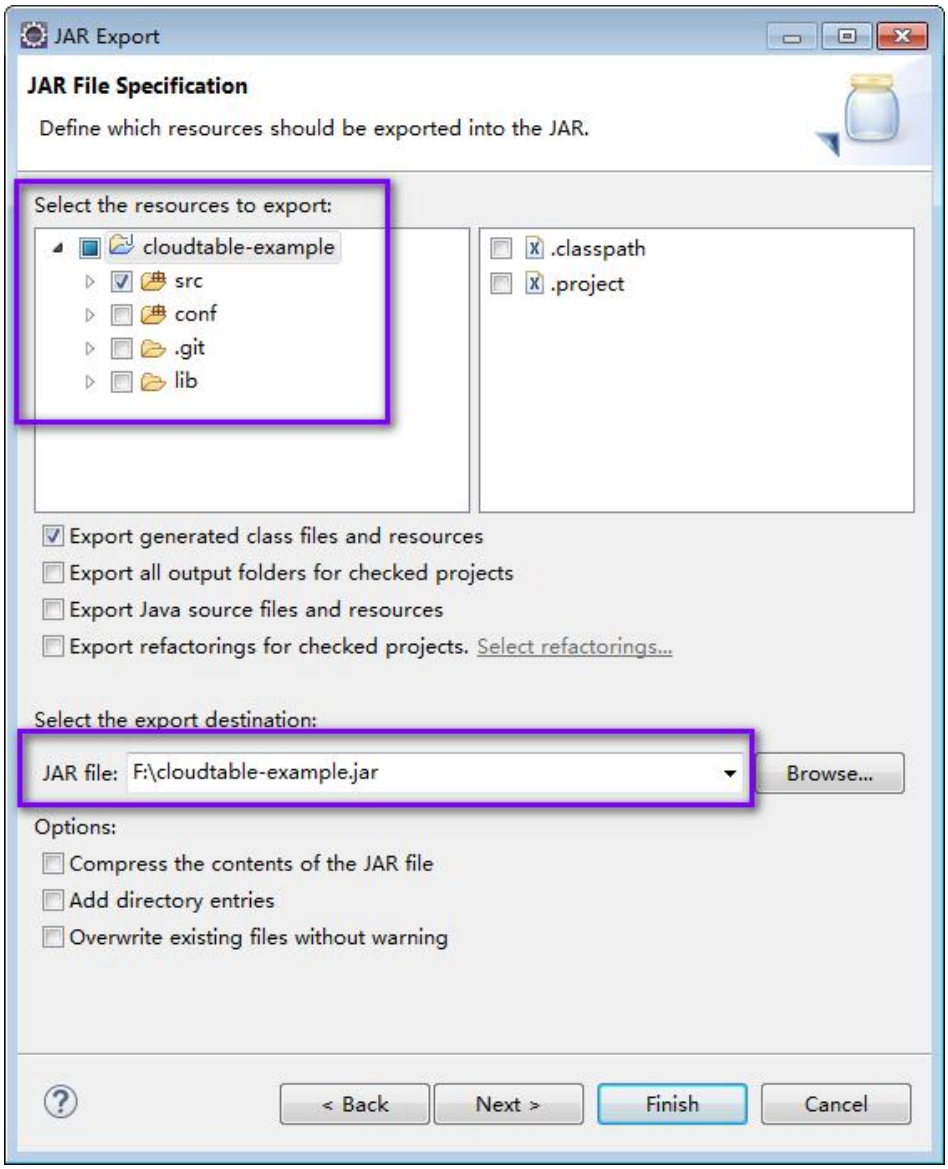
2. Select **JAR file** and click **Next**.

Figure 7-5 Selecting JAR file



3. Select the **src** directory, and export the JAR file to the specified location. Click **Next** twice.

Figure 7-6 Selecting a path for exporting the JAR file



4. Click **Finish**. Exporting the JAR file is complete.

Step 2 Prepare the required JAR file and configuration file.

1. In the Linux environment, create a directory, for example, **/opt/test**, and create subdirectories **lib** and **conf**. Upload the JAR file in **lib** in the sample project and the JAR file exported in **Step 1** to the **lib** directory on Linux. Upload the configuration file in **conf** in the sample project to the **conf** directory on Linux.
2. In the **/opt/test** root directory, create the **run.sh** script, modify the following content, and save the file:

```
#!/bin/sh
BASEDIR=`pwd`
cd ${BASEDIR}
for file in ${BASEDIR}/lib/*.jar
do
i_cp=$i_cp:$file
echo "$file"
done
for file in ${BASEDIR}/conf/*
```



```
do
i_cp=$i_cp:$file
done
java -cp .${i_cp} com.huawei.cloudtable.hbase.examples.TestMain
```

Step 3 Go to **/opt/test** and run the following command to run the JAR file:

sh run.sh

----End

7.2.3 Viewing Commissioning Results

If no exception or failure information is displayed, running the application is successful.

The log level is INFO by default and you can view more detailed information by changing the log level, such as DEBUG, INFO, WARN, ERROR, and FATAL. You can modify the **log4j.properties** file to change log levels, for example:

```
hbase.root.logger=INFO,console
log4j.logger.org.apache.zookeeper=INFO
#log4j.logger.org.apache.hadoop.fs.FSNamesystem=DEBUG
log4j.logger.org.apache.hadoop.hbase=INFO
# Make these two classes DEBUG-level. Make them DEBUG to see more zk debug.
log4j.logger.org.apache.hadoop.hbase.zookeeper.ZKUtil=INFO
log4j.logger.org.apache.hadoop.hbase.zookeeper.ZooKeeperWatcher=INFO
```


8 External APIs

8.1 HBase Java API

HBase adopts the same APIs as those of Apache HBase. For details, visit **https://hbase.apache.org/1.2/apidocs/index.html**.

8.2 OpenTSDB API

8.2.1 OpenTSDB API Introduction

OpenTSDB provides HTTP-based or HTTPS-based APIs. A request method is to send a standard HTTP request containing the GET and POST methods to a path of a resource. The API is the same as that of the open source OpenTSDB. For details, visit **https://opentsdb.net/docs/build/html/api_http/index.html**.

The request and response entity type is application/JSON.

The code of the request and response entity is ISO-8859-1.

NOTE

HTTP has security risks and HTTPS is a secure protocol. You are advised to use HTTPS for connection.

Common OpenTSDB APIs are as follows:

- **Writing Data**
- **Querying Data**
- **Querying the First Piece of Data**
- **Querying the Last Piece of Data**
- **Managing the Data Lifecycle**

Obtaining an OpenTSDB Link

Before calling an OpenTSDB API, perform the following operations to obtain the OpenTSDB link:

To obtain the OpenTSDB link, log in to the CloudTable management console and choose **Cluster Mode** in the left navigation pane. Click the cluster name to access the basic cluster information page. Locate **OpenTSDB Link** in the **Cluster Information** area, as shown in the following figure.

Figure 8-1 OpenTSDB link

8.2.2 Writing Data

Function Description

You can write multiple pieces of data to OpenTSDB with one request. Multiple pieces of data can be independent of each other. Each piece of data can be independently processed, and one error pieces of data will not affect other data. However, if a request contains a large number of data points, the API may take a long time to respond.

URI

- URI format
  a. Write data.
     POST {OpenTSDB URL}/api/put
  b. Write data and return summary information.
     POST {OpenTSDB URL}/api/put?summary
  c. Write data and return details.
     POST {OpenTSDB URL}/api/put?details

     NOTE

     If both the **summary** and **details** flags exist in the query character string, the API returns the detailed information.
  d. Write data and wait for the data to be written to disks.
     POST {OpenTSDB URL}/api/put?sync

     NOTE

     It is strongly recommended that the **sync** parameter be used. Otherwise, the API returns a response before data is successfully written, which may cause data loss.
  e. Write data, wait for the data to be written to disks, and set a timeout interval. When a timeout occurs, the number of successful and failed data points will be returned if the **details** flag is used.
     POST {OpenTSDB URL}/api/put?sync&sync_timeout=60000

Request

- Sample request: Writing a single data point

```
{
   "metric": "sys.cpu.nice",
   "timestamp": 1346846400,
   "value": 18,
   "tags": {
      "host": "web01",
      "dc": "lga"
   }
}
```

- Sample request: Writing multiple data points

```
[
   {
      "metric": "sys.cpu.nice",
      "timestamp": 1346846400,
      "value": 18,
      "tags": {
         "host": "web01",
         "dc": "lga"
      }
   },
   {
      "metric": "sys.cpu.nice",
      "timestamp": 1346846400,
      "value": 9,
      "tags": {
         "host": "web02",
         "dc": "lga"
      }
   }
]
```

- Parameter description

Table 8-1 Request parameters

Attribute	Type	Mandatory	Description	Restrictions
metric	String	Yes	Metric name	• Contains only uppercase and lowercase letters, digits, hyphens (-), underscores (_), periods (.), and slashes (/). • Cannot contain spaces or other characters. • The value is case-sensitive.

Attribute	Type	Mandatory	Description	Restrictions
timestamp	Integer	Yes	Timestamp. The unit is second.	• Unix/POSIX Epoch timestamp (unit: second). The value is the seconds starting from 00:00:00 UTC on January 1, 1970. **NOTE** You are advised to set the timestamp to a value between 4334400 seconds and 4291718400 seconds, that is, from 1970/02/20 12:00:00 to 2106/01/01 00:00:00. • Must be an integer. • Contains a maximum of 13 digits.
value	Integer,Long,Double,Boolean	Yes	Data value	Integer, single-precision (Float) floating point number, double-precision (Double) floating point number, or Boolean

Attribute	Type	Mandatory	Description	Restrictions
tags	Map	Yes	Key-value pair of Tagk and Tagv	• Contains only uppercase and lowercase letters, digits, hyphens (-), underscores (_), periods (.), and slashes (/). • Cannot contain spaces or other characters. • The value is case-sensitive. • At least one key pair and at most eight Tagk and Tagv key-value pairs

Response

- Sample response: summary

```
{
   "failed": 1,
   "success": 0
}
```

Sample response: details

```
{
   "errors": [
      {
         "datapoint": {
            "metric": "sys.cpu.nice",
            "timestamp": 1365465600,
            "value": "NaN",
            "tags": {
               "host": "web01"
            }
         },
         "error": "Unable to parse value to a number"
      }
   ],
   "failed": 1,
   "success": 0
}
```

- Parameter description

Table 8-2 Attributes of the returned information

Parameter	Type	Description
success	Integer	Number of data points that are successfully written
failed	Integer	Number of data points that fail to be written
errors	Array	The value and causes of the data point that fails to be written. This parameter is valid only in details mode.

Status Code

For details about status codes, see **Response Code**.

8.2.3 Querying Data

Function Description

Query data from the OpenTSDB database.

URI

- URI format

  POST {OpenTSDB URL}/api/query

Request

- Sample request

```
{
   "start": 1504527820,
   "end": 1504557820,
   "queries": [
      {
         "aggregator": "sum",
         "metric": "cpu.system",
         "rate": "true",
         "filters": [
            {
               "type":"regexp",
               "tagk":"host",
               "filter":"web[0-9]+.lax.mysite.com",
               "groupBy":true
            },
            {
               "type":"literal_or",
               "tagk":"dc",
               "filter":"lax|dal",
               "groupBy":false
            },
         ]
      }
   ]
}
```


- Parameter description

Table 8-3 Request parameters

Parameter	Type	Mandatory	Description
start	Integer	Yes	Start Time (unit: second) The query result contains the parameter value. **NOTE** You are advised to set this parameter to a value between 4334400 seconds and 4291718400 seconds, that is, from 1970/02/20 12:00:00 to 2106/01/01 00:00:00. You can also set it to 0. Otherwise, the query result may be incorrect.
end	Integer	No	End time (unit: second). The default value is the current system time of OpenTSDB. The query result contains the parameter value. **NOTE** You are advised to set this parameter to a value between 4334400 seconds and 4291718400 seconds, that is, from 1970/02/20 12:00:00 to 2106/01/01 00:00:00. Otherwise, the query result may be incorrect.
queries	Array	Yes	Multiple queries are supported. For details, see **Table 8-4**.
delete	Boolean	No	If this parameter is set to **true**, all query results that meet the search criteria will be deleted. **NOTE** Deleting data is performed in the unit of one hour. Therefore, data of all hours associated with the query result will be deleted.
noAnnotations	Boolean	No	Whether to return comment information • true: Not returned • false: Returned The default value is **false**.
globalAnnotations	Boolean	No	Whether to return global comments within the time span • true: Returned • false: Not returned The default value is **false**.

Parameter	Type	Mandatory	Description
showTSUIDs	Boolean	No	Whether to return the TSUID associated with the time series in the result • true: If multiple time series are aggregated into a set, multiple TSUIDs will be returned in a sort manner. • false: Not returned The default value is **false**.
showSummary	Boolean	No	Whether to return the time summary in the result. For details, see **https://opentsdb.net/docs/build/html/user_guide/query/stats.html**. • true: Returned • false: Not returned The default value is **false**.
showStats	Boolean	No	Whether to return the detailed time in the result. For details, see **https://opentsdb.net/docs/build/html/user_guide/query/stats.html**. • true: Returned • false: Not returned The default value is **false**.
showQuery	Boolean	No	Whether to return the original subquery with the query result. • true: Returned • false: Not returned The default value is **false**.
msResolution	Boolean	No	By default, the timestamp in the query result is expressed in seconds. If this parameter is set to **true**, the timestamp in the query result is expressed in milliseconds.
returnCount	Boolean	No	Whether to return the number of query results. • true: Returned • false: Not returned The default value is **false**.

Parameter	Type	Mandatory	Description
reverse	Boolean	No	Whether to return query results in descending order of time. • true: Yes • false: No The default value is **false**.
onlyDelete	Boolean	No	• true: The result data that meets the query conditions is deleted, but the query result is not returned. • false: This parameter is invalid. The default value is **false**.
returnBoolean	Boolean	No	Whether to convert the returned **value** into a Boolean value. • true: yes. If **value** is greater than 0, the value is converted into **true**. If **value** is equal to or less than 0, the value is converted into **false**. • false: no The default value is **false**.

Table 8-4 Subquery parameters

Parameter	Type	Mandatory	Description
aggregator	String	Yes	Aggregation function. For details, see **aggregator description**.
metric	String	Yes	Metric
rate	Boolean	No	Whether or not the data should be converted into deltas before returning. • true: Returned • false: Not returned The default value is **false**.
downsample	String	No	An optional downsampling function to reduce the amount of data returned. For details, see **•downsample description**.

Parameter	Type	Mandatory	Description
filters	List	No	Filter. You can set multiple filters. For details about the format of each filter, see **•filter parameter description**.
explicitTags	Boolean	No	Whether to return the series that include only the tag keys provided in the filters. • true: Returned • false: Not returned The default value is **false**.
useMultiGets	Boolean	No	Whether to query data using MultiGet. The default value is **false**. This parameter is valid only when **type** of **filters** is set to **literal_or** and **explicitTags** is set to **true**.
valueFilter	String	No	Specifies the expression for filtering values. For details, see **•valueFilter description**.

- **downsample** description

  If a user queries for data over an hour time span, for example, the temperature data is written to OpenTSDB every second, the user will receive 3600 data points. If the user query data of a week, 604800 data points are returned. It becomes messy to display so much data and usually it is unnecessary to provide data to such precision degree. Using a downsampler, multiple data points within a time range are aggregated into one data point. For example, data points over an hour time span are aggregated into one data point. In this way, only 168 data points will be displayed.

  Syntax:

  ```
  <Interval><units>-<aggregator>[c][-<fill policy>]
  ```

  - **Interval**: A time range across which to aggregate the values. The **units** parameter indicates a time unit, such as s for second, m for minute, h for hour, and d for day. Example: 1h, 30m, 24h
  - **aggregator**: aggregation policy, that is, a policy to aggregate time points in a period of time into one data point. For details, see **•aggregator description**.
  - **fill policy**: fill policy. When the aggregator calculates the aggregated value within a period, this policy is used to supplement data when a data point is missing. For details, see **Table 8-5**.

Table 8-5 Fill policy parameters

Parameter	Description
none	Default behavior that does not emit missing values
nan	Emits a NaN.
null	Emits a null.
zero	Zero if missing
pre	Emits the previous value (after aggregation).

NOTE

downsample aggregates data of time lines into a new time line, and then filters the start and end time in the query condition.

- **aggregator** description

  aggregator is used when downsampling is performed and multiple time lines are aggregated. An operator is used to aggregate multiple data points into one data point. For details about aggregation operators, see **Table 8-6**.

Table 8-6 Operators

Operator	Description	Value Filling Method
avg	Averages the data points.	Linear interpolation
count	Number of raw data points in the set	Zero if missing
dev	Calculates the standard deviation.	Linear interpolation
ep50r3	Calculates the estimated 50th percentile with the R-3 method.	Linear interpolation
ep50r7	Calculates the estimated 50th percentile with the R-7 method.	Linear interpolation
ep75r3	Calculates the estimated 75th percentile with the R-3 method.	Linear interpolation

Operator	Description	Value Filling Method
ep75r7	Calculates the estimated 75th percentile with the R-7 method.	Linear interpolation
ep90r3	Calculates the estimated 90th percentile with the R-3 method.	Linear interpolation
ep90r7	Calculates the estimated 90th percentile with the R-7 method.	Linear interpolation
ep95r3	Calculates the estimated 95th percentile with the R-3 method.	Linear interpolation
ep95r7	Calculates the estimated 95th percentile with the R-7 method.	Linear interpolation
ep99r3	Calculates the estimated 99th percentile with the R-3 method.	Linear interpolation
ep99r7	Calculates the estimated 99th percentile with the R-7 method.	Linear interpolation
ep999r3	Calculates the estimated 99.9th percentile with the R-3 method.	Linear interpolation
ep999r7	Calculates the estimated 99.9th percentile with the R-7 method.	Linear interpolation
first	Returns the first data point in the set.	-
last	Returns the last data point in the set.	-
mimmin	Selects the smallest data point.	Maximum if missing

Operator	Description	Value Filling Method
mimmax	Selects the largest data point.	Minimum if missing
min	Selects the smallest data point.	Linear interpolation
max	Selects the largest data point.	Linear interpolation
none	Skips group by aggregation of all time series.	Zero if missing
p50	Calculates the 50th percentile.	Linear interpolation
p75	Calculates the 75th percentile.	Linear interpolation
p90	Calculates the 90th percentile.	Linear interpolation
p95	Calculates the 95th percentile.	Linear interpolation
p99	Calculates the 99th percentile.	Linear interpolation
p999	Calculates the 99.9th percentile.	Linear interpolation
sum	Adds the data points together.	Linear interpolation
zimsum	Adds the data points together.	Zero if missing

NOTE

For details about the R-3 and R-7 methods, see **https://en.wikipedia.org/wiki/Quantile**.

- **filter** parameter description

Table 8-7 filter parameters

Parameter	Type	Mandatory	Description	Example Value
type	String	Yes	Filter type. For details, see **Table 8-8**.	regexp

Parameter	Type	Mandatory	Description	Example Value
tagk	String	Yes	Tag key to invoke the filter on	host
filter	String	Yes	filter expression	web[0-9]+.lax.mysite.com
groupBy	Boolean	No	Whether or not to group the results by each value matched by the filter. The default value is **false**.	false

Table 8-8 file type parameters

Parameter	Description	Example Value
literal_or	Values that tagv is equal to. The values are case sensitive.	{"type":"literal_or","tagk":"host","filter":"web01|web02|web03","groupBy":false}
iliteral_or	Values that tagv is equal to. The values are case insensitive.	{"type":"iliteral_or","tagk":"host","filter":"web01|web02|web03","groupBy":false}
not_literal_or	Values that tagv is not equal to. The values are case sensitive.	{"type":"not_literal_or","tagk":"host","filter":"web01|web02|web03","groupBy":false}
not_iliteral_or	Values that tagv is not equal to. The values are case insensitive.	{"type":"not_iliteral_or","tagk":"host","filter":"web01|web02|web03","groupBy":false}
wildcard	Asterisk wildcard that tagv must comply with. The values are case sensitive.	In the URI: host=wildcard(*mysite.com) {"type":"wildcard","tagk":"host","filter":"web*.tsdb.net","groupBy":false}
iwildcard	Asterisk wildcard that tagv must comply with. The values are case insensitive.	{"type":"iwildcard","tagk":"host","filter":"web*.tsdb.net","groupBy":false}
regexp	Regular expression that tagv must comply with	{"type":"regexp","tagk":"host","filter":".*","groupBy":false}

Parameter	Description	Example Value
not_key	Values that tagk is not equal to	{"type":"not_key","tagk":"host","filter":"","groupBy":false}

- **valueFilter** description

  Filter the returned data points based on the configured numeric condition expression. Operators such as ">", "<", "=", "<=", ">=", "!=", "&&", and "||" are supported.

  Operators such as ">", "<", "=", "<=", ">=", and "!=" are unary operators and used for number operations. Operators "&&" and "||" are binary operators used for expression operations.

  An example expression is as follows:

  a. >=10: indicates that only the result whose value is greater than or equal to 10 is returned.

  b. <5||>10: indicates that only the result whose value is less than 5 or greater than 10 is returned.

  c. >=6&&!=11: indicates that only the result whose value is greater than or equal to 6 and not equal to 11 is returned.

  NOTE

  When both operators "&&" and "||" exist in an expression, operator "&&" takes priority over operator"||". That is, expression >=6||>11&&!=15 is equivalent to expression **>=6||(>11&&!=15)**.

Response

- Sample response

```
[
  {
    "metric": "tsd.hbase.puts",
    "tags": {
      "host": "tsdb-1.mysite.com"
    },
    "aggregatedTags": [],
    "dps": {
      "1365966001": 3758788892,
      "1365966061": 3758804070,
...
      "1365974281": 3778141673
    }
  },
  {
    "metric": "tsd.hbase.puts",
    "tags": {
      "host": "tsdb-2.mysite.com"
    },
    "aggregatedTags": [],
    "dps": {
      "1365966001": 3902179270,
      "1365966062": 3902197769,
...
      "1365974281": 3922266478
    }
  }
]
```


- Parameter description

Table 8-9 Response parameters

Parameter	Description
metric	Metric
tags	Tagv that is not aggregated. If aggregation exists, the value is null.
aggregateTags	If aggregation exists, this parameter indicates the aggregated tagv.
dps	Data point pair. A data point consists of a timestamp and a value and is serialized. If **returnCount** is set to **true** in the query request, **dps** will not be returned. **NOTE** If the value is a floating-point number, it is displayed in double-precision (Double) mode and can be converted to a single-precision (Float) or double-precision (Double) floating point number as required.
annotations	Comment information
globalAnnotations	Global comments within the time span
count	If **returnCount** is set to **true** in the query request, **count** indicating the number of query results will be returned.

Status Code

For details about status codes, see **Response Code**.

8.2.4 Querying the First Piece of Data

Function Description

You can query a data point with the earliest timestamp by specifying a metric.

If the metric has multiple data points with different tags sharing the earliest timestamp, only one data point is returned. The unit of the timestamp is millisecond.

URI

- URI format

  POST {OpenTSDB URL}/api/query/first

Request

- Sample request


```
{
   "resolveNames":true,
   "backScan":20,
   "queries":[
      {
         "metric":"sys.cpu.nice"
      },
      {
         "metric":"cpu.system"
      }
   ]
}
```

- Parameter description

Table 8-10 Request parameters

Parameter	Type	Mandatory	Description
resolveNames	Boolean	No	Whether to convert tsuid in the return result to the corresponding metric, Tagk, and Tagv names. ● true: Convert ● false: Do not convert
backScan	Integer	No	Past hours in which you want to scan data. The unit is hour. Data is stored by hour. Assume that the current system time is 2018/07/20 19:30:00 and **backScan** is set to **2**. The query time range is from 2018/07/20 17:00:00 to 2106/01/01 00:00:00 instead of starting from 2018/07/20 17:30:00. The data scan starts from 2018/07/20 17:00:00 until the first piece of data is found. If you leave **backScan** blank or set it to **0**, data in all time will be scanned.
queries	Array	Yes	List of metrics to be queried. The tag and tsuid cannot be specified. Multiple subqueries can be performed.

Response

- Sample response

```
[
   {
      "metric": "sys.cpu.nice",
      "timestamp": 1346846400000,
      "value": "18",
      "tags": {
         "host": "web01",
         "dc": "lga"
      },
      "tsuid": "00000E0000090007E500000A0007E6"
   },
   {
```



```
    "metric": "cpu.system",
    "timestamp": 1346846400000,
    "value": "9",
    "tags": {
        "host": "web02",
        "dc": "lga"
    },
    "tsuid": "00000F0000090007E700000A0007E6"
  }
]
```

- Parameter description

Table 8-11 Response parameters

Parameter	Type	Description
metric	String	Metric
timestamp	long	Timestamp. The unit is millisecond.
value	String	Data value
tags	Map	Key-value pair of Tagk and Tagv
tsuid	String	**tsuid** corresponding to the metric, Tagk, and Tagv

Status Code

For details about status codes, see **Response Code**.

8.2.5 Querying the Last Piece of Data

Function Description

You can query a data point with the latest timestamp by specifying a metric or tsuid.

If the metric or tsuid has multiple data points with different tags sharing the latest timestamp, only one data point is returned. The unit of the timestamp is millisecond.

URI

- URI format

  POST {OpenTSDB URL}/api/query/last

Request

- Sample request (specified metric)

```
{
  "resolveNames":true,
  "backScan":20,
  "queries":[
    {
      "metric":"sys.cpu.nice"
    },
    {
```

```
        "metric":"cpu.system"
      }
   ]
}
```

- Sample request (specified metric and tag)

```
{
   "resolveNames":true,
   "backScan":20,
   "queries":[
        {
      "metric":"sys.cpu.nice",
      "tags": {
            "host": "web01",
            "dc": "lga"
        }
       },
       {
      "metric":"cpu.system",
      "tags": {
            "host": "web01",
            "dc": "lga"
        }
       }
   ]
}
```

- Sample request (specified tsuid)

```
{
   "resolveNames":true,
   "backScan":20,
   "queries":[
          {
              "tsuids": [
                        "00000E0000090007E500000A0007E6",
                        "00000F0000090007E500000A0007E6"
                    ]
          }
   ]
}
```

- Parameter description

Table 8-12 Request parameters

Parameter	Type	Mandatory	Description
resolveNames	Boolean	No	Whether to convert tsuid in the return result to the corresponding metric, Tagk, and Tagv names. • true: Convert • false: Do not convert

Parameter	Type	Mandatory	Description
backScan	Integer	No	Past hours in which you want to scan data. The unit is hour. Data is stored by hour. Assume that the current system time is 2018/07/20 19:30:00 and **backScan** is set to **2**. • If only **metric** is specified in the subquery of **queries**, and **tags** or **tsuids** is not specified, the time range of the subquery is from 2018/07/20 17:00:00 to 2106/01/01 00:00:00. The data scan starts from 2106/01/01 00:00:00 until the latest piece of data is found. • If both **metric** and **tags** are specified or **tsuids** is specified in the subquery of **queries**, the time range of the sub-query is from 2018/07/20 17:00:00 to 2018/07/20 19:30:00. The data scan starts from 2018/07/20 19:30:00 until the latest piece of data is found. • If you leave **backScan** blank or set it to **0**, data in all time will be scanned.
queries	Array	Yes	List of metrics, tags, and tsuids to be queried. Multiple subqueries can be performed. For details, see **Table 8-13**.

Table 8-13 queries parameters

Parameter	Type	Mandatory	Description
metric	String	No	Metric
tags	Map	No	Key-value pair of Tagk and Tagv
tsuids	List	No	**tsuid** corresponding to the metric, Tagk, and Tagv

NOTE

In the subquery of **queries**, either **metric** or **tsuids** must be set.

Response

- Sample response

```
[
  {
```

```
    "metric": "sys.cpu.nice",
    "timestamp": 1346846400000,
    "value": "18",
    "tags": {
        "host": "web01",
        "dc": "lga"
    },
    "tsuid": "00000E0000090007E500000A0007E6"
  },
  {
    "metric": "cpu.system",
    "timestamp": 1346846400000,
    "value": "9",
    "tags": {
        "host": "web02",
        "dc": "lga"
    },
    "tsuid": "00000F0000090007E700000A0007E6"
  }
]
```


- Parameter description

Table 8-14 Response parameters

Parameter	Type	Description
metric	String	Metric
timestamp	long	Timestamp. The unit is millisecond.
value	String	Data value
tags	Map	Key-value pair of Tagk and Tagv
tsuid	String	**tsuid** corresponding to the metric, Tagk, and Tagv

Status Code

For details about status codes, see **Response Code**.

8.2.6 Managing the Data Lifecycle

Function Description

If Time To Live (TTL) of the OpenTSDB database is set for a metric, the system periodically deletes the data whose data point timestamp is earlier than the time calculated by the formula "Current system time – TTL".

The unit of TTL is hour, and data is cleared by hour. For example, if the current system time is 19:30 and the TTL of a metric is set to 2, the data before 17:00 will be cleared rather than the data before 17:30.

- If the TTL of a metric is not set, the global TTL is used. If the global TTL is not set, the data will not be cleared. If the TTL of a metric is set, the metric's own TTL is used.
- If the TTL of a metric is 0, TTL is not set for the metric and data will not be cleared.

URI

- URI format
  - Set TTL.

    PUT {OpenTSDB URL}/api/ttl
  - Query TTL.

    POST {OpenTSDB URL}/api/ttl
  - Delete TTL.

    DELETE {OpenTSDB URL}/api/ttl

Request

- **Sample request: Setting TTL**

```
{
   "global":36,
   "ttl":{
      "sys.cpu.nice":24,
      "cpu.system":12
   }
}
```

The following table describes the parameters. At least one of the **global** and **ttl** parameters must be specified.

Table 8-15 Parameters

Parameter	Type	Mandatory	Description
global	Integer	No	Global TTL. The unit is hour. It indicates that the validity period of the metric data whose TTL is not configured will be set to the value of **global**. If **global** is set to **0**, the global TTL is invalid.
ttl	Map	No	List of metrics for which the TTL is set. A key-value format is used. A key is a metric name, and a value is the TTL (unit: hour). For example: **"sys.cpu.nice": 24** indicates that the TTL of the **sys.cpu.nice** metric is 24 hours.

- **Sample request: Querying TTL**

  Querying the TTL of the specified metric:

```
{
      "metrics":["sys.cpu.nice", "cpu.system"]
}
```

  Querying all metrics for which the TTL is set and the global TTL

```
{
       "all":true
}
```


The following table describes the parameters. At least one of the following parameters must be specified. If the **metrics** parameter is specified, at least one metric must be specified.

Table 8-16 Parameters

Parameter	Type	Mandatory	Description
global	Boolean	No	Whether to query the global TTL. The value **true** indicates that the value of the **global** parameter is queried.
all	Boolean	No	Whether to query the TTL of all metrics. If **all** is set to **true**, the TTL of all metrics and the global TTL are returned. The **metrics** parameter is invalid.
metrics	String/ Array	No	Indicates the name of the metric to be queried. If one metric is set, the type is String. If multiple metrics are set, the type is Array.

- **Sample request: Deleting TTL**

```
{
   "global":true,
      "metrics":["sys.cpu.nice", "cpu.system"]
}
```

The following table describes the parameters. At least one metric must be specified in **metrics** or **global** is set to **true**.

Table 8-17 Parameters

Parameter	Type	Mandatory	Description
global	Integer, Boolean	No	Whether to delete the global TTL. The value **true** indicates that the value of the **global** parameter is deleted.
all	Boolean	No	Whether to delete all TTL values. The value **true** indicates that all TTL values are deleted. In this case, the **metrics** parameter is invalid.
metrics	String/ Array	No	Names of the metrics to be deleted. If one metric is set, the type is String. If multiple metrics are set, the type is Array.

Response

- **Sample response: Querying TTL**

```
{
   "global": 36,
   "ttl": {
      "sys.cpu.nice": 24,
      "cpu.system": 12
   }
}
```

Table 8-18 Response parameters

Parameter	Type	Description
global	Integer	Global TTL. The unit is hour. It indicates that the validity period of the metric data whose TTL is not configured will be set to the value of **global**. If **global** is set to **0**, the global TTL is invalid.
ttl	Map	List of metrics for which the TTL is set. A key-value format is used. A key is a metric name, and a value is the TTL (unit: hour). For example: **"sys.cpu.nice": 24** indicates that the TTL of the **sys.cpu.nice** metric is 24 hours.

- **Sample Response: Setting or deleting TTL**

  No content is returned in the body.

Status Code

For details about status codes, see **Response Code**.

8.2.7 Response Code

A standard HTTP response code will be returned for each request. Many responses contain content, especially error codes.

Table 8-19 Successful response code

Response Code	Description
200	The request completed successfully.
204	The server has completed the request successfully but is not returning content in the body. This is primarily used for storing data points as it is not necessary to return data to caller.
301	This may be used in the event that an API call has migrated or should be forwarded to another server.

Table 8-20 Error response code

Response Code	Description
400	Information provided by the API user, via a query string or content data, was in error or missing. This will usually include information in the error body about what parameter caused the issue. Correct the data and try again.
404	The requested endpoint or file was not found. This is usually related to the static file endpoint.
405	The requested verb or method was not allowed. Please see the documentation for the endpoint you are attempting to access.
406	The request could not generate a response in the format specified. For example, if you ask for a PNG file of the logs endpoint, you will get a 406 response since log entries cannot be converted to a PNG image (easily).
408	The request has timed out. This may be due to a timeout fetching data from the underlying storage system or other issues.
413	The results returned from a query may be too large for the server's buffers to handle. This can happen if you request a lot of raw data from OpenTSDB. In such cases break your query up into smaller queries and run each individually.
500	An internal error occurred within OpenTSDB. Make sure all of the systems OpenTSDB depends on are accessible and check the bug list for issues.
501	The requested feature has not been implemented yet. This may appear with formatters or when calling methods that depend on plugins
503	A temporary overload has occurred. Check with other users/applications that are interacting with OpenTSDB and determine if you need to reduce requests or scale your system.

Table 8-21 Return error response

Attribute	Type	Mandatory	Description	Example Value
code	Integer	Yes	HTTP response code	400
message	String	Yes	Brief description about an error	Missing required parameter

Attribute	Type	Mandatory	Description	Example Value
details	String	No	Details about an error	Missing value: type

The HTTP status code and response content are returned for all error responses.

Example:

```
{
  "error": {
  "code": 400,
  "message": "Missing parameter <code>type</code>"
  }
}
```

8.2.8 Use Restrictions

Use Restrictions

- In the system, a maximum of 16777215 metrics, tagks, or tagvs can be created.
- A query delay is affected by query criteria and time ranges. If there is a large amount of data to be aggregated, the query delay is longer.
- Exercise caution when you query a large amount of data. Querying a large amount of data is time-consuming and will affect service usage. You cannot manually stop the query operation until the query result is returned.
- The maximum request body size to support for incoming HTTP requests is 32 MB.
- You are advised to set the timestamp to a value between 4334400 seconds and 4291718400 seconds, that is, from 1970/02/20 12:00:00 to 2106/01/01 00:00:00. Otherwise, the query result may be incorrect.
- When you query data, if the value of the data point is a floating-point number, it is displayed in double-precision (Double) mode and can be converted to a single-precision or double-precision floating point number as required.

8.3 GeoMesa Java API

If the current **classpath** contains the GeoMesa code, you can obtain the HBase datastore instance using a GeoTools API. To obtain the HBase datastore, **classpath** must contain **hbase-site.xml**, including HBase datastore connection parameters **hbase.zookeeper.quorum** and **hbase.zookeeper.property.clientPort**.

```
Map<String, Serializable> parameters = new HashMap<>();
parameters.put("bigtable.table.name", "geomesa");
org.geotools.data.DataStore dataStore =
    org.geotools.data.DataStoreFinder.getDataStore(parameters);
```

DataStore contains only one parameter.

- **bigtable.table.name**: Name of an HBase table for storing feature data.

For more information about Java APIs, see the GeoTools API and visit http://docs.geotools.org/stable/userguide/.

NOTE

- LockingManager and Listener related APIs are not implemented due to GeoMesa lock management mechanism and other causes.
- In the current version, you are not advised to use the following APIs to modify the feature attribute.

```
FeatureStore.modifyFeatures(Name[], Object[], Filter)
FeatureStore.modifyFeatures(AttributeDescriptor[], Object[], Filter)
FeatureStore.modifyFeatures(Name, Object, Filter)
FeatureStore.modifyFeatures(AttributeDescriptor, Object, Filter)
SimpleFeatureStore.modifyFeatures(String, Object, Filter)
SimpleFeatureStore.modifyFeatures(String[], Object[], Filter)
```

  You can modify the feature attributes by searching for and insert a feature to overwrite the old one.

```
FeatureSource.getFeatures(Filter)
FeatureStore.addFeatures(FeatureCollection<T, F>)
```


A Change History

Release Date	Description
2019-03-06	This issue the fifth official release. • Modified the following section: – **Application Development Process** • Added the following sections: – **Developing HBase Elasticsearch Full-Text Search Applications**
2018-09-25	This issue the fourth official release. • Added the following sections: – **Parameter Configuration** – **Parameter Configuration** – **Parameter Configuration** – **OpenTSDB API Introduction** • Modified the following section: – **Querying Data**
2018-9-10	This issue the third official release. • Modified the following section: – **Querying Data**

Release Date	Description
2018-8-3	This issue the second official release. • Added the following sections: – **Performing IAM Authentication for Clusters** – **Performing IAM Authentication for Clusters** – **Querying the First Piece of Data** – **Querying the Last Piece of Data** – **Managing the Data Lifecycle** • Modified the following sections: – **Sample Code Description** – **Sample Code Description** – **Querying Data**
2018-06-30	This issue the first official release.