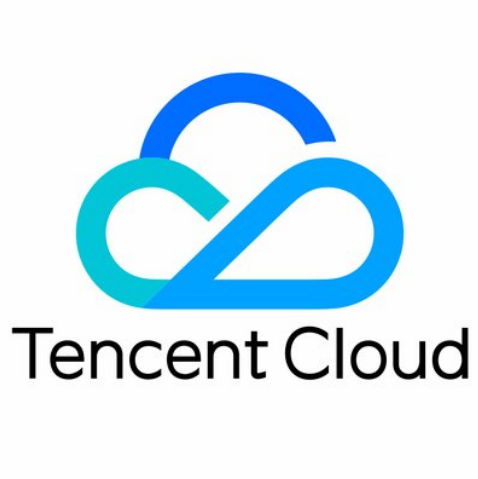
TencentDB for TcaplusDB

TcaplusDB SDK



Copyright Notice

Trademark Notice

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

TcaplusDB SDK Change History

Last updated：2024-10-15 17:42:29

C++ TDR Table SDK

TcaplusServiceApi3.55.0.208570.x86_64_release_20231226

Release time: 2023-12-26
Fix Issues:
bug = 114833923 api Resolve Domain Name returns IP list
other = Fix Inaccurate Statistics issue
bug = 117661097 restproxy core on Packet Capture (Most users do not need to focus on this, related to user API call register zone)
other = Update TSF4G compliant version TSF4G_BASE-2.7.54.66b1e7b79_X86_64_Release

TcaplusServiceApi3.46.0.200166.x86_64_release_20221222

Release time: 2022-12-22
Fix Issues: Fix listdeletbatch Return Index Number
story = 875138979 If the resolved domain's dir is unavailable, need to Try Parsing Domain Again.
story = 875202645 connect all Optimization
story = 875169197 API Link dir Pattern Reporting
bug = 100866863 Fix tdr-record GetData Failed to Obtain Some Fields issue
story = 871564819 service API support list Table fix
bug = 100488315 Fix reload issue
Bug = 100290921 Proxy upgrade cannot find routes

TcaplusServiceApi3.46.0.199774.x86_64_release_20220530

Release date: May 30, 2022
Fix Issues:
Bug = 99377291 Online API bug, hash ring is empty issue, cannot find routes, millisecond-level impact.
Bug = Fixed core issue with dir link exceeding 1s, introduced in minor version, not leaked online.
Optimization:
Story = 874388597 Support perf sampling, log performance statistics of individual API calls.
Story = 874388587 Error code optimization, network thread returns error code to main thread.
Story = 873358171 Memory optimization, reduced from 160M to 40M.
Story = 873358171 Removed SSL dependencies, upgraded BSON and JSON dependencies to TBase 2.7.40.

TcaplusDB SDK 3.34.0.191456

Release date: April 21, 2019
New:

- pb index tables can be created, and batch query can be performed on indices.
- Supported self-release optimization for callback objects.
- The async mode of packet sending/receipt is optimized.
- The log feature is optimized.

Fixes:

- Fixed multi-threading API object initialization failure issue.
- Get request now supports default Get version.
- Repeated connection errors between API and dir.
- Fixed callback error for large dir response packages.

TcaplusDB SDK 3.32.0.185732

Release date: November 22, 2018
Fixes: Multi-thread traversal issue fix.

TcaplusDB SDK 3.32.0.174230

Release date: June 6, 2018
Fixes: RegistAllTable error repair.

C++ PB Table SDK

TcaplusPbApi3.55.0.208570.x86_64_release_20231226

Release time: 2023-12-26
Fix Issues:
other = fix supports user-defined one-time packet quantity other = fixed memory leak code (exceptional scenarios, generally not reachable)
story = 885844745 optimized 3.55.0PbApi FieldSet performance bug = 114833923 API resolves domain name returning IP list other = fixed inaccurate statistics bug = 117661097 restproxy core on packet capture (most users do not need to focus on this, related to user API call register zone)
other = Update TSF4G compliant version TSF4G_BASE-2.7.54.66b1e7b79_X86_64_Release
TcaplusPbApi3.46.0.199774.x86_64_release_20220530
Release date: May 30, 2022
Fixed issues updated tbase2.7.40 updated corresponding version of C++API features

TcaplusPbApi3.55.0.207554.x86_64_release_20220225

Release date: February 25, 2022
gcc11.1 + pb3.14.0 version

Fix Issues:
FieldInc did not write klen field when creating record, fixed API to be compatible with this situation.

SDK for

TcaplusDBJavaApi 3.40.0 SP07

Release date: December 18, 2023
Branch: TcaplusDB 3.40.0
Version number: TcaplusJavaApi3.40.0SP07-20231218-release build at 20231218
CommitId:bde3903127f9ee73c64b569524d4a65a8e5347bb
Fix: fixed the issue where "CPU usage remained at 100% under low business load".

TcaplusDBJavaApi 3.40.0 SP06

Release date: December 4, 2023
Branch: TcaplusDB 3.40.0
Version number: TcaplusJavaApi3.40.0SP06-20231204-release build at 20231204
CommitId:839c2eb8caea1f6f2554386d2a9e0e74537b166f
Fix: fixed the issue where "Batch Get requests were concentrated on one Proxy"
Fix: Resolved the issue of "Occasional API_ERR_FAILED_TO_FIND_ROUTE error"
Fix: Resolved the issue where "Generic table traversal occasionally incomplete"
Fix: Resolved the issue of "Unable to retrieve the lastAccessTime attribute of records"

TcaplusDBJavaApi 3.40.0 SP05

Release time: 2023-08-24
Branch: TcaplusDB 3.40.0
Version number: TcaplusJavaApi3.40.0SP05-20230824-release build at 20230824
CommitId:cda9940259116b9d5403eb67d2c172c954c39b2e
Features: Support for Single Value Field of 10M
Features: New Table Structure Description Interface
Optimization: Optimize network connection, report SDK version information, etc.
Fix: Resolved the issue of "Incorrect error code returned when retrieving meta information for a non-existent table"
Fix: Resolved the issue where "Traversal requests from the same Java SDK instance are always sent to only one Proxy"

TcaplusDBJavaApi 3.40.0 SP01

Release time: 2021-12-17
Branch: TcaplusDB 3.40.0
Version number: TcaplusDBJavaApi3.40.0.193361.x86_64_release build at 20211217
CommitId:3e6bae452c6f80c520d2bcafd240af52ac73bc17

Fix: Fix the issue with 'invalid Version Field in records fetched by BatchGet and GetByPartKey interfaces'

SDK Download

Last updated: 2024-10-15 17:43:11

Introduction

To help you integrate TcaplusDB, this document provides a guide for downloading the TcaplusDB SDK.
Download link: SDK Download

Change History

Before downloading the SDK, please check the Change History.

C++ SDK API

Last updated: 2024-10-15 17:43:31

Introduction

The TcaplusDB Service-oriented API is a data access entry for applications accessing TcaplusDB, serving as a programming interface for storing and retrieving business data in TcaplusDB. Currently, TcaplusDB mainly uses Google Protocol Buffer (Protobuf) for communication and data meta Definition protocols.

Process

After you activate the service and create a table in the Console, the details page will provide access ID, access password, intranet IPv4 address, and the Protobuf Table Management page will provide information such as the table name and Table Group ID. Using the TcaplusDB C++ API, applications can operate multiple tables under this cluster.

Module

You can use the Protobuf protocol to Definition a table that conforms to TcaplusDB specifications. Upload the metafile of the table Definition to the TcaplusDB Console to create a new table or modify an existing table. Upon success, you can perform read and write operations on table data records via the TcaplusDB Protobuf API. The current supported operations of the TcaplusDB Protobuf API are as follows:

Operation	Feature Description
GET	Get a single value based on a single field
BATCHGET	Get multiple values based on multiple fields
ADD	Insert a record. If the field's corresponding record exists, an error will be reported
SET	Update data if the field's corresponding record exists, otherwise insert data request
DEL	Delete the corresponding record based on the field
FIELDINC	Increase the value of the specified field (integer) by a specified value
FIELDSET	Update the value of the specified field (single or multiple)
FIELDGET	Get the value of the specified field (single or multiple)
INDEXGET	Perform index query based on specified index and field

TRAVERSE	Perform a full table traversal based on the specified table name

TcaplusDB SDK Conventions

TcaplusDB requires Definition of the primary key fields in the table. The extension tcaplusservice.optionv1.proto exists in the TcaplusDB system. You only need to reference it when defining the table yourself. You do not need to upload it when creating a new table or modifying an existing table, as the system already has it built-in. The specific content is as follows:

```
extend google.protobuf.MessageOptions
{
    optional string tcaplus_primary_key             = 60000; // Primary key
of the Definition table
    repeated string tcaplus_index                   = 60001; // Index of
the Definition table
    optional string tcaplus_field_cipher_suite      = 60002; // Encryption
algorithm used by the fields of the Definition table
    optional string tcaplus_record_cipher_suite     = 60003; // Encryption
algorithm used by the fields of the Definition table, currently unused
    optional string tcaplus_cipher_md5              = 60004; // Used for
returning the digest of the cipher fields
    optional string tcaplus_sharding_key            = 60005; //Tcaplus
sharding key
}

extend google.protobuf.FieldOptions
{
    optional uint32 tcaplus_size                    = 60000; // Field size,
currently unused
    optional string tcaplus_desc                    = 60001; // Field
Description
    optional bool tcaplus_crypto                    = 60002; // Whether the
field is encrypted, the encryption algorithm is defined in
tcaplus_field_cipher_suite
}
```

The complete file can be found in the directory release\x86_64\include\tcaplus_pb_api\tcaplusservice.optionv1.proto.

1. The table name must start with a letter or underscore, cannot exceed 31 fields, and cannot contain special characters other than numbers, letters, and underscores.
2. The names of each field must be named with letters or underscores as restricted by protobuf.
3. The primary key can have a maximum of 4 fields and must be of the required type. The packaged length cannot exceed 1022 bytes.

4. Field values cannot exceed 256KB when packaged, and the entire record length cannot exceed 256KB when packaged.
5. There must be at least one common field besides the primary key fields.
6. The table is by default of Generic Type but does not display the Generic Type externally.
7. Currently, primary key fields can only be of protobuf-defined Scalar Value Types and cannot include other composite types or custom Definition types.

Common API Description

- `int Get(::google::protobuf::Message &msg);`

```
 @brief According to the index name, msg value, offset, and limit in req
input by the user, obtain multiple record values through the index and
fill them into the vec structure in res, and return the total record
count and remaining records
 @param [INOUT] req User input req
 @param [INOUT] res User input res
 @retval <0 Failed, return the corresponding error code
 @retval 0 Success
```

- `int BatchGet(std::vector< ::google::protobuf::Message * > &msgs);`

```
@brief Batch acquire field values of msg messages based on the field
values in msgs input by the user and fill them into msgs
@param [INOUT] msgs User input field list, return the specified fields
filled into msgs
@retval <0 Failed, return the corresponding error code
@retval 0 Success, at least one field query is successful and will return
0
```

- `int Add(::google::protobuf::Message \*msg);`

```
 @brief Insert msg data records based on the fields in msg input by the
user. If the field exists, an error is reported and exits
 @param [INOUT] msg User input field values and data record msg to be
inserted
 @retval <0 Failed, return the corresponding error code
 @retval 0 Success
```

- `int Set(const ::google::protobuf::Message &msg);`

```
 @brief Based on the field in the user input msg, if the record exists,
update the specified record's value, otherwise insert the specified
record
 @param [INOUT] msg The user input field value and the data record to be
set msg
 @retval <0 Failed, return the corresponding error code
 @retval 0 Success
```

- `int Del(const ::google::protobuf::Message &msg);`

```
@brief Based on the field value in the user input msg, delete msg
@param [IN] msg The user input field value, return the specified field
filled into msg
@retval <0 Failed, return the corresponding error code
@retval 0 Success, at least one field query is successful and will return
0
```

- `int FieldInc(::google::protobuf::Message &msg, const std::set &dottedpaths);`

```
 @brief Based on the field value in the user input msg and the
incremental values of the values, and the field names specified by
dottedpaths, increase the value of the specified field in msg. The field
is a numerical variable
 @param [INOUT] msg The data record msg, containing the user input field
value, return the result value of the incremental field updated into msg
 @param [IN] dottedpaths A dot-separated nested string set of field names
 @retval <0 Failed, return the corresponding error code, indicating no
field was updated
 @retval 0 Success, all fields updated successfully
```

- `int FieldGet(const std::set<std::string> &dottedpaths, ::google::protobuf::Message *msg, std::set<std::string> *failedpaths);`

```
 @brief Based on the field value in the user input msg and the field
names specified by dottedpaths, get the value of the specified field and
fill it into msg
 @param [INOUT] msg The data record msg, containing the user input field
value, return the specified field filled into msg
 @param [IN] dottedpaths A dot-separated nested string set of field names
 @param [OUT] failedpaths Return a dot-separated nested string set of
field names that failed to be found
 @retval <0 Failed, return the corresponding error code
```

```
 @retval 0 Success, at least one field query is successful and will
return 0
```

- `int FieldSet(::google::protobuf::Message &msg, const std::set &dottedpaths);`

```
 @brief Based on the field value in the user input msg and the field
names specified by dottedpaths, update the value of the specified field.
Values that do not exist on the server will be added
 @param [IN] msg User input field value. The specified field will be
returned and filled into msg
 @param [IN] dottedpaths A dot-separated nested string set of field names
 @retval <0 Failed, return the corresponding error code, indicating no
field was updated
 @retval 0 Success, all fields updated successfully
```

- `int Get(NS_TCAPLUS_PROTOBUF_API::IndexGetRequest& req, NS_TCAPLUS_PROTOBUF_API::IndexGetResponse *res);`

```
 @brief According to the index name, msg value, offset, and limit in req
input by the user, obtain multiple record values through the index and
fill them into the vec structure in res, and return the total record
count and remaining records
 @param [INOUT] req User input req
 @param [INOUT] res User input res
 @retval <0 Failed, return the corresponding error code
 @retval 0 Success
```

- `int Traverse(::google::protobuf::Message *msg, TcaplusTraverseCallback \*cb);`

```
 @brief Traverse the table, messages will be filled into msg
 @param [INOUT] msg Return the specified field and fill it into msg
 @param [INOUT] cb Callback function
 @retval <0 Failed, return the corresponding error code
 @retval 0 Traversal completed successfully
```

Rules and Constraints

Get Operation Constraints

- Cannot query records of specific version numbers.
- If the field to be queried does not exist in the original records, it will return the default field value.

BatchGet Operation Constraints

- Batch query operations must be routed through tcaproxy; the tcapsvr process does not support batch queries.
- A batch query request returns a batch query result, with a timeout of 10 seconds.
- The number of records in the batch query result is the same as in the request. Even non-existent or failed records will have an empty entry in the result set, so FetchRecord is required to retrieve records.
- The total size of the batch query result set must not exceed 256K. If it exceeds, the content of the oversized records cannot be returned and will be empty.
- The order of the batch query result set is not guaranteed to match the order of the request records.

FieldInc Operational Constraints

- The fields specified in the message must already exist.
- The fields specified in dottedpaths must be of numeric data type.
- The total number of elements in the set containing dottedpaths cannot exceed 128 elements.
- The length of each element in the set containing dottedpaths cannot exceed 1023 bytes.
- The field nesting represented by each element in the set containing dottedpaths cannot exceed 32 levels.

FieldGet Operational Constraints

- The total number of elements in the set containing dottedpaths cannot exceed 128 elements.
- The length of each element in the set containing dottedpaths cannot exceed 1023 bytes.
- The field nesting represented by each element in the set containing dottedpaths cannot exceed 32 levels.

FieldSet Operational Constraints

- The total number of elements in the set containing dottedpaths cannot exceed 128 elements.
- The length of each element in the set containing dottedpaths cannot exceed 1023 bytes.
- The field nesting represented by each element in the set containing dottedpaths cannot exceed 32 levels.

SDK Source Files Directory Structure

```
    `-- release
        `-- x86_64
            |-- docs                                    Document Directory
            |   `-- tcaplus
```

```
|       `-- readme.txt                      Usage Description
Guide File for this C++ SDK
|-- examples                                Sample Directory for
this C++ SDK, divided into Synchronization and Asynchronous sections, can
be directly modified for use
|-- include                                 Header File
Directory for this C++ SDK
|   `-- tcaplus_pb_api                      TcaplusDB PB API
Header Folder
|       |-- cipher_suite_base.h             Data Encryption
Algorithm Suite Base Class
|       |-- default_aes_cipher_suite.h      Data Encryption
Algorithm Suite Default Implementation Class
|       |-- tcaplus_async_pb_api.h          Header File for
Asynchronous Mode Class, using the TcaplusAsyncPbApi Class in Asynchronous
mode
|       |-- tcaplus_coroutine_pb_api.h      Header File for
Coroutine Mode Class, using the TcaplusCoroutinePbApi Class in Coroutine
mode
|       |-- tcaplus_error_code.h            Error Code Header
File, all related error codes can be found here with their Definitions and
descriptions
|       |-- tcaplus_protobuf_api.h          API Summary Header
File, including other header files, only this one file needs to be
referenced for convenient development
|       |-- tcaplus_protobuf_define.h       Basic Structure and
Macro Definition Header File, including ClientOptions Structure and
MESSAGE_OPTION_* Macro Definition
|       |-- tcaplusservice.optionv1.pb.h    Tcaplus Table Common
Definition Protobuf Header File
|       `-- tcaplusservice.optionv1.proto   Tcaplus Table Common
Definition Proto Source File, must include this file when defining tables
|-- lib                                     Library File
Directory for this C++ SDK
|   `-- libtcaplusprotobufapi.a             Library File for
this C++ SDK, must include in the program's final link library
`-- version                                 Version Record File
for this C++ SDK
```

TcaplusDB Error Codes

Last updated: 2024-10-15 17:43:52

Serial number	Error Codes	Description
1	-525	BatchGet operation request timeout
2	-39685	Single data shard exceeds the maximum limit of 256GB, write failure
3	-34565	Index does not exist
4	-34309	Value exceeds the maximum limit of 10MB
5	-34053	Key exceeds the maximum limit of 1KB
6	-33541	Internal error, please contact administrator
7	-18957	SQL filtering failure in distributed index
8	-18701	InsertByPart key not supported
9	-18451	API and svr meta comparison failure
10	-18445	Document type request not supported
11	-18195	API to proxy link failure
12	-17939	Pulling proxy list from dir timeout
13	-17683	Dir authentication failure
14	-17427	Dir link failure
15	-17171	API awaiting network thread startup failure
16	-17165	PB table tag not match
17	-17157	Decompression failed
18	-16915	API resolution dir domain name resolution failed

19	-16909	PB table FieldIncRecord operation failed, please contact administrator
20	-16901	Compression failed
21	-16659	SQL driver in use, wait response timed out
22	-16653	PB table FieldSetRecord operation failed, please contact administrator
23	-16403	Connection abnormal
24	-16397	PB table non-primary key field value exceeds the restriction size (256KB)
25	-16389	Storage layer value number of blocks abnormal, please contact administrator
26	-16147	Initialization log module failed
27	-16141	PB table GetRecord operation failed, please contact administrator
28	-15885	Invalid field value
29	-15635	Traversal device already terminated or stopped when receiving the package, generally not returned to the user, can be ignored
30	-15379	The operated zoneid does not exist
31	-15377	Access layer could not find user authentication information
32	-15373	Request denied, server is in relocation state, routing error.
33	-15121	Access layer could not find user authentication information
34	-15117	Invalid array size
35	-14865	Access layer could not find user authentication information
36	-14609	Access layer could not find user authentication information
37	-14353	Access layer authentication failed, business authentication information not found
38	-14099	The operated single record exceeds 10M after packaging
39	-14097	Unsupported SQL
40	-14093	For union field type, server-side failed to obtain select entry
41	-13843	Backend network exception, request cannot be sent successfully. Please contact administrator if the issue persists

42	-13841	Index field does not exist
43	-13837	Internal error, please contact administrator
44	-13587	Protocol compression on commands that do not support protocol compression, called SetCompressSwitch
45	-13585	SQL statement failure in parsing global index
46	-13329	Timeout querying global index
47	-13075	Rest API limits request response body to 10M; currently HTTP request limits the size of the entire package to 10M
48	-13073	Querying global index failed
49	-13069	Data shard does not exist
50	-12819	The operated table does not exist
51	-12817	Allocation failed, please contact the administrator to increase the transaction concurrency configuration
52	-12563	Magic number mismatch, communication issue, if the issue persists, please contact the administrator
53	-12561	Allocation failed, please contact the administrator to increase the transaction concurrency configuration
54	-12557	Internal parameter error
55	-12307	API send buffer full
56	-12305	Invalid SQL request
57	-12301	Exceeds user Definition scope
58	-12051	REST proxy request did not set table name
59	-12049	SQL Query Manager creation failed
60	-12045	Storage layer retrieval cursor traversal failed, there may be too many concurrent traversal requests, which is uncommon
61	-11793	Access layer overload
62	-11789	Table Definition not found in storage layer, internal error, please contact the administrator
63	-11539	Invalid request

64	-11537	Send buffer is full, API processing response is too slow
65	-11533	Data sharding not found in storage layer, internal error, please contact administrator
66	-11283	Uninitialized, usually occurs in traversal requests
67	-11277	Method of operation table does not exist
68	-11027	Batch operation record count exceeded limit, supports up to 1024 records in a batch
69	-11021	API version is too low, please upgrade the API
70	-10771	Invalid result flag
71	-10769	Access layer failed to process cache request
72	-10765	Storage layer unpack failed
73	-10513	Insufficient buffer during route change
74	-10509	Internal error, failed to package field, please contact administrator
75	-10259	Internal error, missing binary field version
76	-10257	No cache request exists during route change
77	-10253	Merge value field failed, internal error, please contact administrator
78	-10003	api_get_meta_size_error
79	-10001	Cache request failed during route change
80	-9997	Merge value field failed, internal error, please contact administrator
81	-9747	Memory Allocation failed
82	-9745	Cache request timed out during route change
83	-9491	Missing part key field
84	-9489	Traversal request returned this error code because the table is in relocation status
85	-9485	Unsupported command literal
86	-9235	Local index does not exist
87	-9233	Access layer access denied

88	-9229	Internal error, unsupported access layer protocol magic
89	-8979	Unpacking Failure in GetData struct
90	-8973	Storage Layer binlog flow disk insufficient disk space
91	-8723	Unpacking Failure in GetData union
92	-8721	Initialization log module failed
93	-8717	Storage Layer data disk insufficient disk space
94	-8467	Unpacking Failure in GetData array
95	-8465	Direct return packet at access layer, equivalent to bypassing business logic, usually used for testing API
96	-8461	Storage Layer overload, please contact DBA or Tcaplus_Helper for investigation
97	-8211	Packaging Failure in setdata struct
98	-8209	Non-timeout error in access layer transaction, please contact administrator
99	-8205	Storage Layer internal error, please contact administrator
100	-7955	Packaging Failure in setdata union
101	-7953	Timed out in access layer
102	-7949	Please check the optimistic lock, the requested record version number is inconsistent with the actual record version number
103	-7697	Request speed exceeds quota
104	-7693	Request directed to the Storage Layer backup node, possibly during primary-standby switch. If this error code persists for a long time, please contact administrator
105	-7669	Failed to package array in setdata
106	-7443	Incompatible table structure, please check whether the local table structure and server table structure are compatible.
107	-7441	Unsupported compression type
108	-7437	Storage Layer read-only, usually due to disk full or during primary-standby switch

109	-7185	Routing node not found, please contact administrator
110	-7181	Access layer stopping, no business concern, API or automatic switching to a new access layer
111	-6929	Missing key field in the request
112	-6925	Incorrect result_flag setting, please refer to the result_flag description in the SDK
113	-6673	No key field
114	-6669	Version too high
115	-6417	Lossy relocation, some commands are not supported, mainly lossless relocation currently, so you won't encounter this issue
116	-6412	Version too low
117	-6161	Invalid message
118	-6157	List elements exceed Definition range, please set element eviction
119	-5907	Memory too small
120	-5905	Failed to parse message
121	-5651	PB unsupported field type, such as repeat type
122	-5649	Insufficient memory
123	-5395	Missing primary key in the request
124	-5393	Failed to send message
125	-5139	Invalid union select
126	-5137	Failed to package
127	-4883	Invalid array size
128	-4627	Incorrect data size, usually due to inconsistency between user's local table definition and the server-side
129	-4621	Request is missing primary key field or indexed field
130	-4377	Syntax error in condition or operational statements
131	-4371	1. API was invoked without initialization for the database request 2. The table to be operated on was not registered

132	-4365	Exceeding the maximum number of fields
133	-4115	Value field quantity limit exceeded, generic table limit is 128. List table limit: 127. Version 3.46.0 increased generic table limit to 256, List table limit to 255
134	-4109	List data type element index out of range
135	-3859	Exceeding the maximum limit of keys, current maximum is 8 keys, List table allows 7 keys
136	-3603	No more records in response packet during FetchRecord
137	-3347	Mismatched command in response package
138	-3341	Field value size exceeds the limit of its definition type
139	-3328	SortList table storage engine internal error, please contact Tcaplus_Helper for investigation
140	-3091	API insert route failed, generally should not occur, if it does, please contact Tcaplus for investigation
141	-3089	Routing error
142	-3085	SetFieldName operation specified the wrong field
143	-3079	Data error, please contact administrator
144	-3072	Insufficient memory
145	-2829	Delete engine record failed, please contact administrator
146	-2823	Engine error, please contact administrator
147	-2816	Unsupported operation, please contact administrator
148	-2583	Dir address error
149	-2579	Response message unpacking failed during GetData
150	-2573	Write engine error, please contact administrator
151	-2560	Parameter error, please check API log or contact administrator
152	-2323	Failed to package during setdata
153	-2317	tcapsvr_fail_sync_write
154	-2304	Table is in Access Denied status, common in Tencent Cloud Environment

155	-2067	Table type and command word do not match. For example, a generic table used a list table's command word; or the flag is unsupported, SetLimit, SetFlag and other settings flag do not match the command
156	-2061	Incorrect table field type
157	-2048	Capacity exceeded, readable and deletable, but cannot write
158	-1811	Parameter error return, generally in API invocation interface, unreasonable parameter setting return
159	-1805	Record exceeds maximum length, serialized data must not exceed 10MB
160	-1792	Table is in read-only mode, please check whether RCU, WCU, or capacity exceeds the user-defined threshold
161	-1555	Field value data type does not match its definition type. For example, if a key field type is int8, attempting to use GetKeyFloat to retrieve that field's value will fail. Please check the table definition to confirm the field data type.
162	-1549	Field does not exist, please ensure the table structure has been updated to the backend
163	-1310	Table does not exist
164	-1299	Field does not exist, please ensure the field name is spelled correctly
165	-1297	PROXY exception
166	-1293	Record to be inserted already exists
167	-1280	Error code table initialization failed, duplicate error code defined
168	-1054	zoneid does not exist (go API error code)
169	-1050	Distributed index query failed
170	-1043	Field value length exceeds the limit specified in the table definition, such as a string defined as 100B but the actual input is 101B; or single key field content exceeds 1024
171	-1037	Storage layer overloaded
172	-787	Field name size exceeds limit (32B)
173	-785	Some commands are not supported in lossy migration. Currently, most are lossless migration, so this error code will not occur
174	-781	Buf too short, internal error

175	-773	Engine file repeated opening
176	-531	Value field quantity limit exceeded, generic table limit is 128. List table limit: 127. Version 3.46.0 increased generic table limit to 256, List table limit to 255
177	-529	Some commands are not supported in lossy migration. Currently, most are lossless migration, so this error code will not occur
178	-517	Internal parameter error
179	-279	Dir authentication failure
180	-275	Exceeding the maximum limit of keys, current maximum is 8 keys, list table 7 keys. If the API version is relatively old, it is recommended to upgrade to the latest version 3.46.0
181	-273	DB server-side internal parameter error
182	-261	DB server-side internal parameter error
183	261	This record does not exist
184	269	New insertion in the sortlist table exceeds the maximum number of list elements, insertion failure
185	2309	The record has no expiration time set

SDK Installation
Installation Guide for Linux

Last updated：2024-10-15 17:44:22

Overview

This document guides you on installing and using TcaplusDB PB tables in a Linux system environment.

Running Environment

Linux 2.6, SUSE 12 64-bit.

Prerequisites

Before installing and using TcaplusDB PB, install Protobuf. TcaplusDB currently supports Protocol Buffers versions 2.6.1 and 3.5.0.
Protobuf is a hybrid language data standard launched by Google and a lightweight structured data storage format. The TcaplusDB system supports using Protobuf format Definition files (.proto) to define data tables. Before using the TcaplusDB PB API, Protobuf must be installed on the development server. It is recommended to use the source code to install Protobuf. The installation method is as follows:

1. Prepare the CVM environment.
   First, prepare a server with the CentOS6-x86_64 or CentOS7-x86_64 operating system installed. To compile and build Protobuf, the following software needs to be installed. Please refer to relevant resources for the installation methods:
   - autoconf
   - automake
   - libtool
   - curl (used to download gmock)
   - make
   - g++
   - unzip
2. Download the Protobuf source code installation package. Please see SDK Download.
3. Install the specified version of ProtoBuf according to your actual needs. The following uses protobuf version 2.6.1 as an example.
4. Run the following commands to decompress the source code installation package and enter the source code root directory.

```
tar -xzvf protobuf-2.6.1.tar.gz
cd ./protobuf-2.6.1
```

5. Configure and specify the installation path prefix. It is recommended to install it to the path `/usr/local/protobuf`.

```
./configure --prefix=/usr/local/protobuf
```

6. Compile and install.

```
make
make check
make install
```

7. Test using the protoc command to verify if the installation is successful. It should display as follows to indicate a successful installation.

```
# protoc --version
libprotoc 2.6.1
```

Directions

Step 1: Install the TcaplusDB SDK

First, download the TcaplusDB SDK to the development server, then execute the commands to install the files to the specified installation directory.

```
tar -xzf <Installation Package Path> -C <Installation Directory>
```

Step 2: Verify

After installation, the root directory structure is as follows:

Directory and Files	Description
include/tcaplus_service/	TCAPLUS Service-oriented API Header Files
lib/libtcaplusserviceapi.a	TCAPLUS Service-oriented API Library Files
include/tcaplus_service/protobuf/	Protobuf API Header Files
lib/libtcaplusprotobufapi.a	TCAPLUS Protobuf API Library Files

examples/tcaplus/ProtoBuf	TCAPLUS Protobuf API Application Example

Step 3: Run example to access TcaplusDB

For developing data access logic in the GameSvr game server, refer to the API examples in the example directory.

1. Unzip the TcaplusDB PB API release package.

```
tar -xzvf TcaplusPbApi3.36.0.152096.x86_64_release_20170712.tar.gz
```

2. Configure TcaplusDB System Connection Information.

2.1 Enter the following code in the command line to navigate to the directory:

```
cd
TcaplusPbApi3.36.0.152096.x86_64_release_20170712/release/x86_64/examp
les/tcaplus/C++_common_for_pb2
```

2.2 Enter the command vi common.h in the command line to edit the common.h header file. Modify the content according to the business scenario as shown in the image.

DIR_URL_ARRAY: Cluster access IP address and port number.
DIR_URL_COUNT: Fixed value, keep it as 1.
TABLE_NAME: Target table to access.
APP_ID: Access ID.
ZONE_ID: Table group ID to access.
SIGNATURE: Cluster access password.

```
/**********************测试例子前需要用户手动修改的地方BEGIN**************************************
// 目标业务的tcapdir地址
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
        "tcp://10.191.***.99:9999"
        "tcp://10.191.***.88:9999"
};

// 目标业务的tcapdir 地址个数
static const int32_t DIR_URL_COUNT = 1;
// 目标业务的表名
static const char * TABLE_NAME = "tb_online";
// 目标业务的App ID
static const int32_t APP_ID = 3;
// 目标业务的Zone ID
static const int32_t ZONE_ID = 1;
// 目标业务的业务密码
static const char * SIGNATURE = "**********";
/**********************测试例子前需要用户手动修改的地方END**************************************/
```

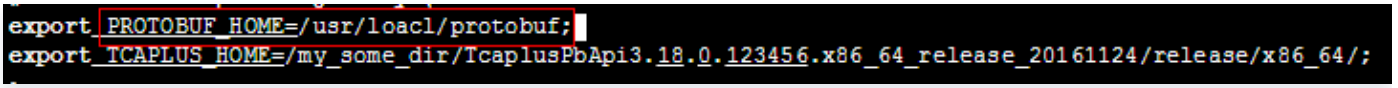
3. Modify the environment configuration file.

In the

`TcaplusPbApi3.36.0.152096.x86_64_release_20170712/release/x86_64/examples/tcaplus` directory, there are examples of invoking API in asynchronous mode and coroutine mode. Here, we'll use coroutine mode to call the Set interface to configure data:
Enter the following code in the command line to navigate to the directory:

```
cd
TcaplusPbApi3.18.0.152096.x86_64_release_20170712/release/x86_64/examples
/tcaplus/C++_pb2_coroutine_simpletable/SingleOperation/set
```

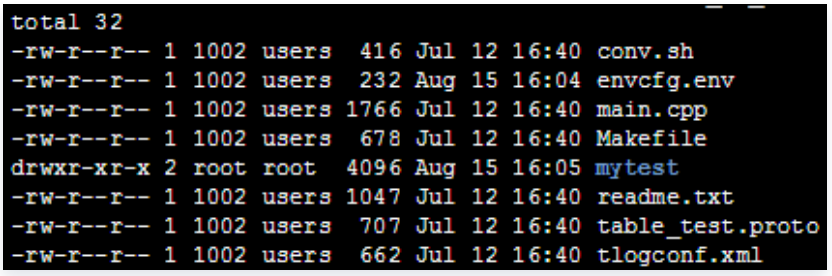
All the codes for the coroutine mode Set example are in this directory. Edit the envcfg.env file, set the PROTOBUF_HOME environment variable to the installation path of protobuf on your local machine (--prefix specified), and set the TCAPLUS_HOME environment variable to the absolute path of the release/x86_64 directory under the Tcaplus PB API package, as shown in the image:



4. Set environment variables and execute the following command in the code directory:

```
source envcfg.env
bash conv.sh
```

5. Compile binary program.
Execute the `make` command to compile the example binary. A successful compile generates the mytest executable file.



6. Run binary program.
Enter `./mytest` on the command line to run the binary program. The execution result will be displayed in the command line standard output. If an error occurs, please check the tcaplus_pb.log log file in the code directory.

Installation Guide for Windows

Last updated: 2024-10-15 17:44:43

Overview

This document introduces how to operate the Tcaplus Protobuf API on a Windows (x64) system.

Running Environment

- Operating system: Microsoft Windows x86_64
- Compilation environment: Microsoft Visual Studio 2015 (VC14.0)

Directions

Download the software package

1. Download the dependency packages and Tcaplus Protobuf API package. See SDK Download.
2. Decompress the package. Below is the structure of the package.

```
Tcaplus_PbAPI_3.32.0.171987_Win64Vc14MT_Release_20180413
|-- cfg                                          # Configuration
Directory
|-- docs                                         # Documentation
Directory
|   `-- tcaplus
|-- include                                      # Dependency
Header File Directory
|   `-- tcaplus_pb_api
|-- lib                                          # Library
Directory
|    |-- Debug
|    `-- Release
|-- examples                                     # Example
Directory
`-- tcaplus
    |-- C++_common_for_pb2                      # Common header file
directory for examples
    |-- C++_pb2_asyncmode_simpletable           # Asynchronous mode
simple pb example
    |-- C++_pb2_coroutine_simpletable           # Coroutine mode
simple pb example
```

Preparation

1. Please ensure that you have enabled the game business in TcaplusDB and have obtained the corresponding App information (e.g., AppId, ZoneId, AppKey).
2. Decompress the dependency package and install it. This document uses the dependency package `TSF4G_BASE-2.7.28.164975_Win64Vc14Mt_Release.zip` as an example. The actual dependency package should be based on Windows Platform Dependency Package Download. Assume the installation root path is `D:\Tencent\tsf4gMT`. Related files will be installed under the path `D:\Tencent\tsf4gMT\win64vc14MT`.
3. Compile and install `Porotbuf-3.5.1`.
   - Source Code Repository
   - Compilation and Installation Guide
   - Assume the installation path is `D:\protobuf-3.5.1`
4. Compile and install `OpenSSL-1.1.0f`.
   - Source Code Address
   - Compilation and Installation Guide
   - Assume the installation path is `D:\openssl-1.1.0f`
5. Set environment variables.
   - `TSF4G_HOME="D:\Tencent\tsf4gMT"`
   - `PROTOBUF_HOME="D:\protobuf-3.5.1"`
   - `OPENSSL_HOME="D:\openssl-1.1.0f"`

Build

1. Decompress the Tcaplus Pb API Installation Package.
2. Set App Information in the `examples/tcaplus/C++_common_for_pb2/common.h` file.
   - Tcapdir Access Point Address List – `DIR_URL_ARRAY`
   - Number of Tcapdir Access Point Addresses – `DIR_URL_COUNT`
   - User Table Name, before using it, users need to create the table with the table_test.xml file from the sample directory – `TABLE_NAME`
   - User Business ID – `APP_ID`
   - User Business Region Server ID – `ZONE_ID`
   - User Business Password – `SIGNATURE`

```
//examples/tcaplus/C++_common_for_pb2/common.h
 /***************User Self Definitions*************************/
 // Tcapdir Access Point Address List
 static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
 {
 "tcp://10.xx.xx.21:9999"
 };
```

```
    // Number of Tcapdir Access Point Addresses
    static const int32_t DIR_URL_COUNT = 1;
    // User Table Name
    static const char * TABLE_NAME = "tb_online";
    // User Business ID
    static const int32_t APP_ID = 4;
    // User Business Region Server ID
    static const int32_t ZONE_ID = 1;
    // User Business Password
    static const char * SIGNATURE = "8e24269ba91fxxxa7e89b1cbb77368e";
    /****************User Self Definitions****************************/
```

3. Taking `examples\tcaplus\C++_pb2_coroutine_simpletable\SingleOperation\set` as an example.

```
set/
|-- main.cpp                                # Sample main function code
|-- readme.txt
|-- table_test.proto                          # T Table Definition proto
file, table needs to be created beforehand
|-- pb_co_set.sln                             # Project VisualStudio
solution file
|-- pb_co_set.vcxproj                         # Project VisualStudio
project file
|-- proto_generate.cmd                        # Compile proto file script
`-- tlogconf.xml
```

3.1 First, confirm that the table has been successfully created in the target App using `table_test.proto`.

3.2 Run the `proto_generate.cmd` script to generate dependency files in the current path.

- `table_test.pb.cc`
- `table_test.pb.h`

3.3 Open the project file `pb_co_set.sln` in Microsoft Visual Studio 2015.

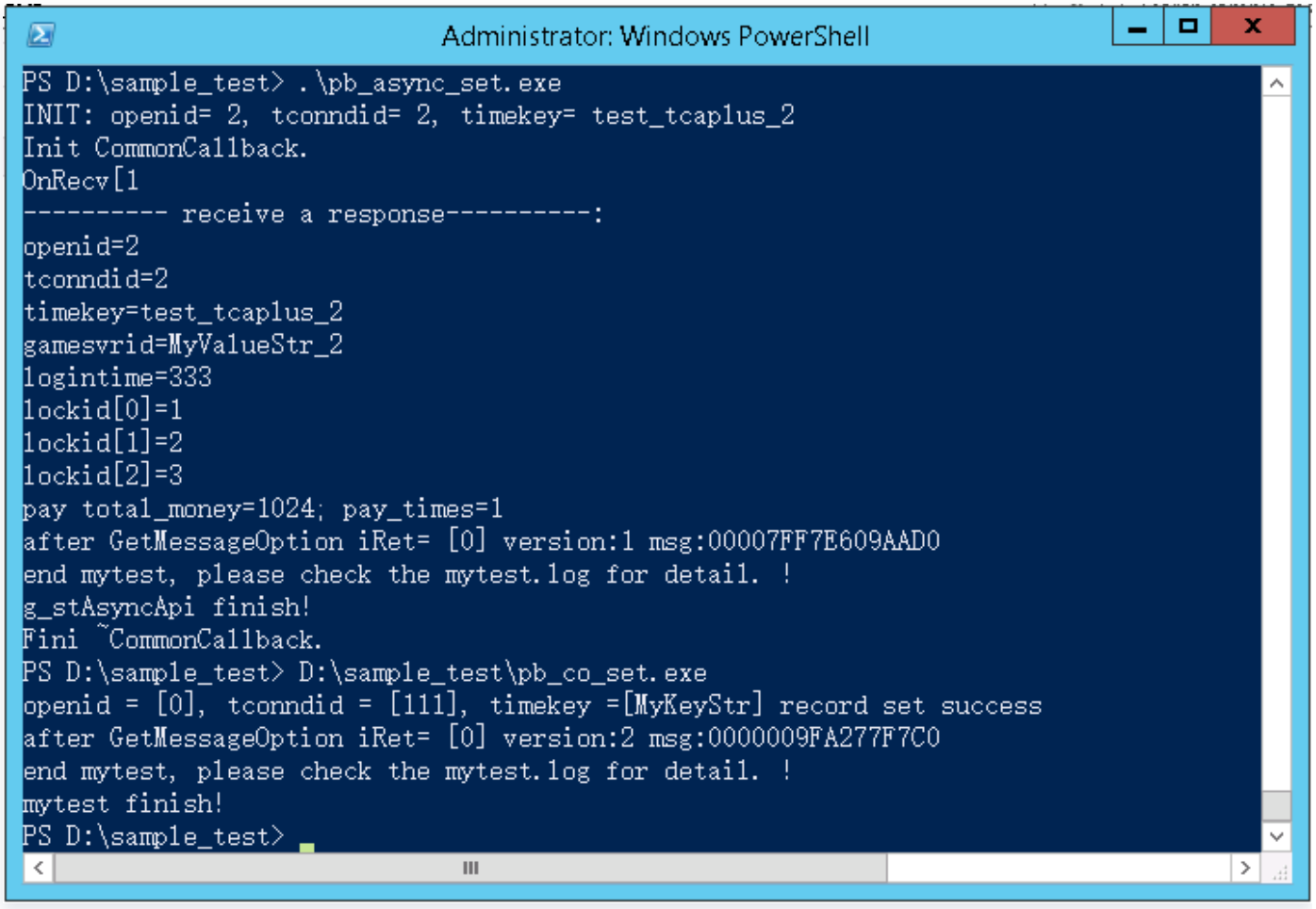
3.4 Build the solution.

3.5 If no errors occur, an executable file `pb_co_set.exe` will be generated under the path `examples\tcaplus\C++_pb2_coroutine_simpletable\SingleOperation\set/x64`.

Testing

1. Copy the files `pb_co_set.exe` and `tlogconf.xml` to the same directory.
2. Switch to the `administrator` account and run the executable file `pb_co_set.exe` using cmd.exe or powershell.exe.

3. Check the output.
4. For detailed runtime information, refer to the log file `tcaplus_pb.log`.
5. If you need to run the `*_crypto` example, make sure the `libcrypto-1_1-x64.dll` file is in the system Path. This file is located in the openssl compilation directory.



Tcaplus Pb API command list

Tcaplus Pb API supports various types of operations, including asynchronous and coroutine modes. Users can find relevant usage in the examples. The following is the Tcaplus Pb API command list:

Command	Description
SET	This API is used to set a record by specifying all primary keys of the record. If the record exists, an overwrite operation is performed; otherwise, an insert operation is executed.
GET	This API is used to query a record from a Tcaplus Pb table by specifying all primary keys of the record. If the data record does not exist, an error will be returned.

ADD	This API is used to insert a record by specifying all primary keys of the record. If the record exists, an error will be returned.
DELETE	This API is used to delete a record by specifying all primary keys of the record. If the data does not exist, an error will be returned.
BATCHGET	This API is used to query multiple records from a Tcaplus Pb table by specifying multiple sets of primary keys.
TRAVERSE	This API is used to traverse a Tcaplus Pb table and returns multiple records.
FIELDGET	This API is used to query a record from a Tcaplus Pb table by specifying all primary keys of the record. Only the user-specified field values are queried and transmitted, reducing network traffic. If the data record does not exist, an error will be returned.
FIELDSET	This API is used to modify specified fields of a record by specifying all primary keys of the record. Only the specified field values are transmitted, reducing network traffic. If the data record exists, an update operation is performed; otherwise, an error will be returned.
FIELDINC	This API is used to perform an auto-increment operation on specified fields of a record by specifying all primary keys of the record. This command only supports int32, int64, uint32, and uint64 field types. The feature is similar to FIELDSET.
GETBYPARTKEY	This API is used to query multiple records that meet the conditions by specifying partial primary keys. The primary key set must have an index created when the table is created; otherwise, an error will be returned.

Directions for Protobuf Table SDK for C++ Data Writing

Last updated: 2024-10-15 17:45:23

Prerequisites

Successfully created: cluster, table group, tb_online table
tb_online table description file table_test.proto as follows: (tcaplusservice.optionv1.proto is the dependent table)

```
syntax = "proto2";

package myTcaplusTable;

import "tcaplusservice.optionv1.proto";

message tb_online {
    option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey";

    required int32 openid = 1; //QQ Uin
    required int32 tconndid = 2;
    required string timekey = 3;
    required string gamesvrid = 4;
    optional int32 logintime = 5 [default = 1];
    repeated int64 lockid = 6 [packed = true]; // repeated type fields use
the packed keyword
    optional pay_info pay = 7;

    message pay_info {
        optional uint64 total_money = 1;
        optional uint64 pay_times = 2;
    }
}
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
```

```
    "tcp://10.191.***.99:9999",
    "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Table name of the target business tb_online
static const char * TABLE_NAME = "tb_online";
```

Step 2: Initialize the TcaplusDB API client for Protobuf

```
// TcaplusDB API client for Protobuf
TcaplusAsyncPbApi g_stAsyncApi;
int32_t InitAsyncPbApi()
{
    // PB API configuration
    ClientOptions cfg;
    cfg.app_id = APP_ID;
    cfg.zones.push_back(ZONE_ID);
    strcpy(cfg.signature, SIGNATURE);
    for (int32_t i = 0; i < DIR_URL_COUNT; i++)
    {
        cfg.dirs.push_back(DIR_URL_ARRAY[i]);
    }
    // The Protobuf table to be accessed
    cfg.tables.push_back(TABLE_NAME);
    // Log configuration
    strncpy(cfg.log_cfg, "tlogconf.xml", sizeof(cfg.log_cfg));
    // Timeout period for connection initialization: 5 seconds
    cfg.timeout = 5000;

    // Initialize connection
    int32_t iRet = g_stAsyncApi.Init(cfg);
    if (0 != iRet)
    {
        cout << "ERROR: g_stAsyncApi.Init failed, log cfg: " << cfg.log_cfg
<< ", iRet: " << iRet << "." << endl;
        return iRet;
    }
```

```
    return iRet;
}
```

Step 3. Define asynchronous callbacks

```
// Count the number of received response messages
uint32_t g_dwTotalRevNum = 0;
class CommonCallback : public TcaplusPbCallback
{
public:
    CommonCallback()
    {
        cout << "Init CommonCallback." << endl;
    }

    ~CommonCallback()
    {
        cout << "Fini ~CommonCallback." << endl;
    }
    // Callback for received response messages, msgs is an array of records
    int OnRecv(const std::vector< ::google::protobuf::Message *> &msgs)
    {
        cout << "OnRecv[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            cout << "---------- receive a response----------:" << endl;
            cout << "openid=" << t->openid() << endl;
            cout << "tconndid=" << t->tconndid() << endl;
            cout << "timekey=" << t->timekey() << endl;
            cout << "gamesvrid=" << t->gamesvrid() << endl;
            cout << "logintime=" << t->logintime() << endl;
            for (int32_t i = 0; i < t->lockid_size();i++)
            {
                cout << "lockid[" << i << "]=" << t->lockid(i) << endl;
            }
```

```
            tb_online_pay_info pay = t->pay();
            cout << "pay total_money=" << pay.total_money() << ";
pay_times=" << pay.pay_times() << endl;

            // Get the version of the record
            std::string version;
            int iRet = g_stAsyncApi.GetMessageOption(*t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, &version);
            cout << "after GetMessageOption iRet= [" << iRet << "]
version:" << version.c_str() << " msg:" << t << endl;
        }

        return 0;
    }

    // A callback for a response error
    int OnError(const std::vector< ::google::protobuf::Message *> &msgs,
int errorcode)
    {
        cout << "OnError[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            if (TcapErrCode::TXHDB_ERR_RECORD_NOT_EXIST == errorcode)
            {
                cout << "ERROR: openid= " << t->openid() << ", tconndid= "
<< t->tconndid() << ", timekey= " << t->timekey() << ", record not exists"
<< endl;
            }
            cout << "ERROR: openid = [" << t->openid() << "], tconndid = ["
<< t->tconndid() << "], timekey =[" << t->timekey() << "] failed:%d"  <<
errorcode << endl;
        }

        return 0;
    }
```

```
    // A callback for a timed-out message
    int OnTimeout(const std::vector< ::google::protobuf::Message *> &msgs)
    {
        cout << "OnTimeout[" << msgs.size() << endl;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "TIMEOUT: msgs[" << idx << "] not tb_online type!"
<< endl;
                return -1;
            }

            cout << "TIMEOUT: openid = [" << t->openid() << "], tconndid =
[" << t->tconndid() << "], timekey =[" << t->timekey() << "] timeout" <<
endl;
        }

        return 0;
    }

    int OnFinish(const NS_TCAPLUS_PROTOBUF_API::MsgParam &param)
    {
        cout << "OnFinish: " << param.m_nOperation << " req: " <<
param.m_vecMsgs.size()<< endl;
        return 0;
    }
};
```

Step 4. Send the add request

```
int SendAddRequest()
{
    static tb_online t;
    t.set_openid(2);
    t.set_tconndid(2);
    t.set_timekey("test_tcaplus_2");
    t.set_gamesvrid("MyValueStr_2");
    t.set_logintime(333);
    for (int32_t i = 0; i < TOTAL_V3_ARRAY_NUM;i++)
    {
        t.add_lockid(i+1);
    }
```

```
    tb_online_pay_info* pay = new tb_online_pay_info();
    pay->set_total_money(1024);
    pay->set_pay_times(1);
    t.set_allocated_pay(pay);

    cout << "INIT: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << endl;

    std::string version = "-1";
    g_stAsyncApi.SetMessageOption(t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, version);

    static CommonCallback cb;
    int32_t iRet = g_stAsyncApi.Add(&t, &cb);
    if (iRet != TcapErrCode::GEN_ERR_SUC)
    {
        cout << "ERROR: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << ", Set Error iRet = " <<
iRet << endl;
        return -1;
    }
    return 0;
}
```

Example

main.cpp File

```
int main(void) {

    // Initialize the API client
    int ret = InitAsyncPbApi();
    if ( ret != 0)
    {
        printf("InitAsyncPbApi failed\n");
        return -1;
    }

    // Send the request
    ret = SendAddRequest();
    if (0 != ret)
    {
        printf("SendAddRequest failed\n");
```

```
        return -1;
    }

    // Receive the response
    do
    {
        // Update and receive the response packet
        g_stAsyncApi.UpdateNetwork();
        usleep(1000 * 10);
    } while (g_dwTotalRevNum != 1);
    return 0;
}
```

Reading data

Last updated: 2024-10-15 17:45:45

Prerequisites

Successfully created: cluster, table group, tb_online table
tb_online table description file table_test.proto as follows: (tcaplusservice.optionv1.proto is the dependent table)

```
syntax = "proto2";

package myTcaplusTable;

import "tcaplusservice.optionv1.proto";

message tb_online {
    option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey";

    required int32 openid = 1; //QQ Uin
    required int32 tconndid = 2;
    required string timekey = 3;
    required string gamesvrid = 4;
    optional int32 logintime = 5 [default = 1];
    repeated int64 lockid = 6 [packed = true]; // repeated type fields use
the packed keyword
    optional pay_info pay = 7;

    message pay_info {
        optional uint64 total_money = 1;
        optional uint64 pay_times = 2;
    }
}
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;

// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Table name of the target business tb_online
static const char * TABLE_NAME = "tb_online";
```

Step 2: Initialize the TcaplusDB API client for Protobuf

```
// TcaplusDB API client for Protobuf
TcaplusAsyncPbApi g_stAsyncApi;
int32_t InitAsyncPbApi()
{
    // PB API configuration
    ClientOptions cfg;
    cfg.app_id = APP_ID;
    cfg.zones.push_back(ZONE_ID);
    strcpy(cfg.signature, SIGNATURE);
    for (int32_t i = 0; i < DIR_URL_COUNT; i++)
    {
        cfg.dirs.push_back(DIR_URL_ARRAY[i]);
    }
    // The Protobuf table to be accessed
    cfg.tables.push_back(TABLE_NAME);
    // Log configuration
    strncpy(cfg.log_cfg, "tlogconf.xml", sizeof(cfg.log_cfg));
    // Timeout period for connection initialization: 5 seconds
    cfg.timeout = 5000;

    // Initialize connection
    int32_t iRet = g_stAsyncApi.Init(cfg);
    if (0 != iRet)
    {
        cout << "ERROR: g_stAsyncApi.Init failed, log cfg: " << cfg.log_cfg
<< ", iRet: " << iRet << "." << endl;
        return iRet;
    }
    return iRet;
}
```

Step 3. Define asynchronous callbacks

```
// Count the number of received response messages
uint32_t g_dwTotalRevNum = 0;
class CommonCallback : public TcaplusPbCallback
{
public:
    CommonCallback()
    {
        cout << "Init CommonCallback." << endl;
    }

    ~CommonCallback()
    {
        cout << "Fini ~CommonCallback." << endl;
    }
    // Callback for received response messages, msgs is an array of records
    int OnRecv(const std::vector< ::google::protobuf::Message *> &msgs)
    {
        cout << "OnRecv[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            cout << "---------- receive a response----------:" << endl;
            cout << "openid=" << t->openid() << endl;
            cout << "tconndid=" << t->tconndid() << endl;
            cout << "timekey=" << t->timekey() << endl;
            cout << "gamesvrid=" << t->gamesvrid() << endl;
            cout << "logintime=" << t->logintime() << endl;
            for (int32_t i = 0; i < t->lockid_size();i++)
            {
                cout << "lockid[" << i << "]=" << t->lockid(i) << endl;
            }
            tb_online_pay_info pay = t->pay();
            cout << "pay total_money=" << pay.total_money() << ";
pay_times=" << pay.pay_times() << endl;
```

```
            // Get the version of the record
            std::string version;
            int iRet = g_stAsyncApi.GetMessageOption(*t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, &version);
            cout << "after GetMessageOption iRet= [" << iRet << "]
version:" << version.c_str() << " msg:" << t << endl;
        }

        return 0;
    }

    // A callback for a response error
    int OnError(const std::vector< ::google::protobuf::Message *> &msgs,
int errorcode)
    {
        cout << "OnError[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            if (TcapErrCode::TXHDB_ERR_RECORD_NOT_EXIST == errorcode)
            {
                cout << "ERROR: openid= " << t->openid() << ", tconndid= "
<< t->tconndid() << ", timekey= " << t->timekey() << ", record not exists"
<< endl;
            }
            cout << "ERROR: openid = [" << t->openid() << "], tconndid = ["
<< t->tconndid() << "], timekey =[" << t->timekey() << "] failed:%d"  <<
errorcode << endl;
        }

        return 0;
    }

    // A callback for a timed-out message
    int OnTimeout(const std::vector< ::google::protobuf::Message *> &msgs)
    {
```

```
        cout << "OnTimeout[" << msgs.size() << endl;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "TIMEOUT: msgs[" << idx << "] not tb_online type!"
<< endl;
                return -1;
            }

            cout << "TIMEOUT: openid = [" << t->openid() << "], tconndid =
[" << t->tconndid() << "], timekey =[" << t->timekey() << "] timeout" <<
endl;
        }

        return 0;
    }

    int OnFinish(const NS_TCAPLUS_PROTOBUF_API::MsgParam &param)
    {
        cout << "OnFinish: " << param.m_nOperation << " req: " <<
param.m_vecMsgs.size()<< endl;
        return 0;
    }
};
```

Step 4. Send the get request

```
int SendGetRequest()
{
    static tb_online t;
    t.set_openid(1);
    t.set_tconndid(1);
    t.set_timekey("test_tcaplus");

    cout << "INIT: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey=" << t.timekey() << endl;

    static CommonCallback cb;
    int32_t iRet = g_stAsyncApi.Get(&t, &cb);
    if (iRet != TcapErrCode::GEN_ERR_SUC)
    {
```

```
        cout << "ERROR: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << ", Get Error iRet = " <<
iRet << endl;
        return -1;
    }
    return 0;
}
```

Example

```
int main(void) {

    // Initialize the API client
    int ret = InitAsyncPbApi();
    if ( ret != 0)
    {
        printf("InitAsyncPbApi failed\n");
        return -1;
    }

    // Send the request
    ret = SendGetRequest();
    if (0 != ret)
    {
        printf("SendSetRequest failed\n");
        return -1;
    }

    // Receive the response
    do
    {
        // Update and receive the response packet
        g_stAsyncApi.UpdateNetwork();
        usleep(1000 * 10);
    } while (g_dwTotalRevNum != 1);
    return 0;
}
```

Updating Data

Last updated: 2024-10-15 17:46:06

Prerequisites

Successfully created: cluster, table group, tb_online table
tb_online table description file table_test.proto as follows: (tcaplusservice.optionv1.proto is the dependent table)

```
syntax = "proto2";

package myTcaplusTable;

import "tcaplusservice.optionv1.proto";

message tb_online {
    option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey";

    required int32 openid = 1; //QQ Uin
    required int32 tconndid = 2;
    required string timekey = 3;
    required string gamesvrid = 4;
    optional int32 logintime = 5 [default = 1];
    repeated int64 lockid = 6 [packed = true]; // repeated type fields use
the packed keyword
    optional pay_info pay = 7;

    message pay_info {
        optional uint64 total_money = 1;
        optional uint64 pay_times = 2;
    }
}
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Table name of the target business tb_online
static const char * TABLE_NAME = "tb_online";
```

Step 2: Initialize the TcaplusDB API client for Protobuf

```
// TcaplusDB API client for Protobuf
TcaplusAsyncPbApi g_stAsyncApi;
int32_t InitAsyncPbApi()
{
    // PB API configuration
    ClientOptions cfg;
    cfg.app_id = APP_ID;
    cfg.zones.push_back(ZONE_ID);
    strcpy(cfg.signature, SIGNATURE);
    for (int32_t i = 0; i < DIR_URL_COUNT; i++)
    {
        cfg.dirs.push_back(DIR_URL_ARRAY[i]);
    }
    // The Protobuf table to be accessed
    cfg.tables.push_back(TABLE_NAME);
    // Log configuration
    strncpy(cfg.log_cfg, "tlogconf.xml", sizeof(cfg.log_cfg));
    // Timeout period for connection initialization: 5 seconds
    cfg.timeout = 5000;

    // Initialize connection
    int32_t iRet = g_stAsyncApi.Init(cfg);
    if (0 != iRet)
    {
        cout << "ERROR: g_stAsyncApi.Init failed, log cfg: " << cfg.log_cfg
<< ", iRet: " << iRet << "." << endl;
        return iRet;
    }
    return iRet;
}
```

Step 3. Define asynchronous callbacks

```
// Count the number of received response messages
uint32_t g_dwTotalRevNum = 0;
class CommonCallback : public TcaplusPbCallback
{
public:
    CommonCallback()
    {
        cout << "Init CommonCallback." << endl;
    }

    ~CommonCallback()
    {
        cout << "Fini ~CommonCallback." << endl;
    }
    // Callback for received response messages, msgs is an array of records
    int OnRecv(const std::vector< ::google::protobuf::Message *> &msgs)
    {
        cout << "OnRecv[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            cout << "---------- receive a response----------:" << endl;
            cout << "openid=" << t->openid() << endl;
            cout << "tconndid=" << t->tconndid() << endl;
            cout << "timekey=" << t->timekey() << endl;
            cout << "gamesvrid=" << t->gamesvrid() << endl;
            cout << "logintime=" << t->logintime() << endl;
            for (int32_t i = 0; i < t->lockid_size();i++)
            {
                cout << "lockid[" << i << "]=" << t->lockid(i) << endl;
            }
            tb_online_pay_info pay = t->pay();
            cout << "pay total_money=" << pay.total_money() << ";
pay_times=" << pay.pay_times() << endl;
```

```
            // Get the version of the record
            std::string version;
            int iRet = g_stAsyncApi.GetMessageOption(*t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, &version);
            cout << "after GetMessageOption iRet= [" << iRet << "]
version:" << version.c_str() << " msg:" << t << endl;
        }

        return 0;
    }

    // A callback for a response error
    int OnError(const std::vector< ::google::protobuf::Message *> &msgs,
int errorcode)
    {
        cout << "OnError[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            if (TcapErrCode::TXHDB_ERR_RECORD_NOT_EXIST == errorcode)
            {
                cout << "ERROR: openid= " << t->openid() << ", tconndid= "
<< t->tconndid() << ", timekey= " << t->timekey() << ", record not exists"
<< endl;
            }
            cout << "ERROR: openid = [" << t->openid() << "], tconndid = ["
<< t->tconndid() << "], timekey =[" << t->timekey() << "] failed:%d"  <<
errorcode << endl;
        }

        return 0;
    }

    // A callback for a timed-out message
    int OnTimeout(const std::vector< ::google::protobuf::Message *> &msgs)
    {
```

```
        cout << "OnTimeout[" << msgs.size() << endl;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "TIMEOUT: msgs[" << idx << "] not tb_online type!"
<< endl;
                return -1;
            }

            cout << "TIMEOUT: openid = [" << t->openid() << "], tconndid =
[" << t->tconndid() << "], timekey =[" << t->timekey() << "] timeout" <<
endl;
        }

        return 0;
    }

    int OnFinish(const NS_TCAPLUS_PROTOBUF_API::MsgParam &param)
    {
        cout << "OnFinish: " << param.m_nOperation << " req: " <<
param.m_vecMsgs.size()<< endl;
        return 0;
    }
};
```

Step 4. Send the set request

```
int SendSetRequest()
{
    // Definition PB Record structure and assignment, converted via proto
    static tb_online t;
    t.set_openid(2);
    t.set_tconndid(2);
    t.set_timekey("test_tcaplus_2");
    t.set_gamesvrid("MyValueStr_2");
    t.set_logintime(333);
    for (int32_t i = 0; i < 3;i++)
    {
        t.add_lockid(i+1);
    }

    tb_online_pay_info* pay = new tb_online_pay_info();
```

```
    pay->set_total_money(1024);
    pay->set_pay_times(1);
    t.set_allocated_pay(pay);

    cout << "INIT: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << endl;

    // Specify the version number which is used by optimistic locking
    std::string version = "-1"; // std::string version="1"; -1 means no
comparison, 1 means server data version is 1
    g_stAsyncApi.SetMessageOption(t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, version);

    //Callback for Definition, automatically called upon receiving the
Response Message
    static CommonCallback cb;
    int32_t iRet = g_stAsyncApi.Set(&t, &cb);
    if (iRet != TcapErrCode::GEN_ERR_SUC)
    {
        cout << "ERROR: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << ", Set Error iRet = " <<
iRet << endl;
        return -1;
    }
    return 0;
}
```

Example

```
int main(void) {

    // Initialize the API client
    int ret = InitAsyncPbApi();
    if ( ret != 0)
    {
        printf("InitAsyncPbApi failed\n");
        return -1;
    }

    // Send the request
    ret = SendSetRequest();
    if (0 != ret)
    {
        printf("SendSetRequest failed\n");
```

```
        return -1;
    }

    // Receive the response
    do
    {
        // Update and receive the response packet
        g_stAsyncApi.UpdateNetwork();
        usleep(1000 * 10);
    } while (g_dwTotalRevNum != 1);
    return 0;
}
```

Deleting Data

Last updated: 2024-10-15 17:46:28

Prerequisites

Successfully created: Cluster, Table Group, tb_online Table
tb_online table description file table_test.proto as follows: (tcaplusservice.optionv1.proto is the dependent table)

```
syntax = "proto2";

package myTcaplusTable;

import "tcaplusservice.optionv1.proto";

message tb_online {
    option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey";

    required int32 openid = 1; //QQ Uin
    required int32 tconndid = 2;
    required string timekey = 3;
    required string gamesvrid = 4;
    optional int32 logintime = 5 [default = 1];
    repeated int64 lockid = 6 [packed = true]; // repeated type fields use
the packed keyword
    optional pay_info pay = 7;

    message pay_info {
        optional uint64 total_money = 1;
        optional uint64 pay_times = 2;
    }
}
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Table name of the target business tb_online
static const char * TABLE_NAME = "tb_online";
```

Step 2: Initialize the TcaplusDB API client for Protobuf

```
// TcaplusDB API client for Protobuf
TcaplusAsyncPbApi g_stAsyncApi;
int32_t InitAsyncPbApi()
{
    // PB API configuration
    ClientOptions cfg;
    cfg.app_id = APP_ID;
    cfg.zones.push_back(ZONE_ID);
    strcpy(cfg.signature, SIGNATURE);
    for (int32_t i = 0; i < DIR_URL_COUNT; i++)
    {
        cfg.dirs.push_back(DIR_URL_ARRAY[i]);
    }
    // The Protobuf table to be accessed
    cfg.tables.push_back(TABLE_NAME);
    // Log configuration
    strncpy(cfg.log_cfg, "tlogconf.xml", sizeof(cfg.log_cfg));
    // Timeout period for connection initialization: 5 seconds
    cfg.timeout = 5000;

    // Initialize connection
    int32_t iRet = g_stAsyncApi.Init(cfg);
    if (0 != iRet)
    {
        cout << "ERROR: g_stAsyncApi.Init failed, log cfg: " << cfg.log_cfg
<< ", iRet: " << iRet << "." << endl;
        return iRet;
    }
    return iRet;
}
```

Step 3. Define asynchronous callbacks

```
// Count the number of received response messages
uint32_t g_dwTotalRevNum = 0;
class CommonCallback : public TcaplusPbCallback
{
public:
    CommonCallback()
    {
        cout << "Init CommonCallback." << endl;
    }

    ~CommonCallback()
    {
        cout << "Fini ~CommonCallback." << endl;
    }
    // Callback for received response messages, msgs is an array of records
    int OnRecv(const std::vector< ::google::protobuf::Message *> &msgs)
    {
        cout << "OnRecv[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            cout << "---------- receive a response----------:" << endl;
            cout << "openid=" << t->openid() << endl;
            cout << "tconndid=" << t->tconndid() << endl;
            cout << "timekey=" << t->timekey() << endl;
            cout << "gamesvrid=" << t->gamesvrid() << endl;
            cout << "logintime=" << t->logintime() << endl;
            for (int32_t i = 0; i < t->lockid_size();i++)
            {
                cout << "lockid[" << i << "]=" << t->lockid(i) << endl;
            }
            tb_online_pay_info pay = t->pay();
            cout << "pay total_money=" << pay.total_money() << ";
pay_times=" << pay.pay_times() << endl;
```

```
            // Get the version of the record
            std::string version;
            int iRet = g_stAsyncApi.GetMessageOption(*t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, &version);
            cout << "after GetMessageOption iRet= [" << iRet << "]
version:" << version.c_str() << " msg:" << t << endl;
        }

        return 0;
    }

    // A callback for a response error
    int OnError(const std::vector< ::google::protobuf::Message *> &msgs,
int errorcode)
    {
        cout << "OnError[" << msgs.size() << endl;
        g_dwTotalRevNum++;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "ERROR: msgs[" << idx << "] not tb_online type." <<
endl;
                return -1;
            }

            if (TcapErrCode::TXHDB_ERR_RECORD_NOT_EXIST == errorcode)
            {
                cout << "ERROR: openid= " << t->openid() << ", tconndid= "
<< t->tconndid() << ", timekey= " << t->timekey() << ", record not exists"
<< endl;
            }
            cout << "ERROR: openid = [" << t->openid() << "], tconndid = ["
<< t->tconndid() << "], timekey =[" << t->timekey() << "] failed:%d"  <<
errorcode << endl;
        }

        return 0;
    }

    // A callback for a timed-out message
    int OnTimeout(const std::vector< ::google::protobuf::Message *> &msgs)
    {
```

```
        cout << "OnTimeout[" << msgs.size() << endl;
        for (size_t idx = 0; idx < msgs.size(); idx++)
        {
            tb_online* t = dynamic_cast<tb_online *>(msgs[idx]);
            if (NULL == t)
            {
                cout << "TIMEOUT: msgs[" << idx << "] not tb_online type!"
<< endl;
                return -1;
            }

            cout << "TIMEOUT: openid = [" << t->openid() << "], tconndid =
[" << t->tconndid() << "], timekey =[" << t->timekey() << "] timeout" <<
endl;
        }

        return 0;
    }

    int OnFinish(const NS_TCAPLUS_PROTOBUF_API::MsgParam &param)
    {
        cout << "OnFinish: " << param.m_nOperation << " req: " <<
param.m_vecMsgs.size()<< endl;
        return 0;
    }
};
```

Step 4. Send the delete request

```
int SendDelRequest()
{
    static tb_online t;
    t.set_openid(1);
    t.set_tconndid(1);
    t.set_timekey("test_tcaplus");

    static CommonCallback cb;
    std::string version = "-1";
    g_stApi.SetMessageOption(t,
NS_TCAPLUS_PROTOBUF_API::MESSAGE_OPTION_DATA_VERSION, version);
    int32_t iRet = g_stAsyncApi.Del(&t, &cb);
    if (iRet != TcapErrCode::GEN_ERR_SUC)
    {
```

```
        cout << "ERROR: openid= " << t.openid() << ", tconndid= " <<
t.tconndid() << ", timekey= " << t.timekey() << ", Delete Error iRet = " <<
iRet << endl;
        return -1;
    }
    return 0;
}
```

Example

```
int main(void) {

    // Initialize the API client
    int ret = InitAsyncPbApi();
    if ( ret != 0)
    {
        printf("InitAsyncPbApi failed\n");
        return -1;
    }

    // Send the request
    ret = SendDelRequest();
    if (0 != ret)
    {
        printf("SendDelRequest failed\n");
        return -1;
    }

    // Receive the response
    do
    {
        // Update and receive the response packet
        g_stAsyncApi.UpdateNetwork();
        usleep(1000 * 10);
    } while (g_dwTotalRevNum != 1);
    return 0;
}
```

Directions for TDR Table SDK for C++ Data Writing

Last updated: 2024-10-15 17:47:03

Prerequisites

Successfully created: Cluster, Table Group, PLAYERONLINECNT Table
PLAYERONLINECNT Table description file table_test.xml as follows:

```
<?xml version="1.0" encoding="GBK" standalone="yes" ?>
<metalib name="tcaplus_tb" tagsetversion="1" version="1">
<struct name="PLAYERONLINECNT" version="1" primarykey="TimeStamp,GameSvrID"
splittablekey="TimeStamp">
    <entry name="TimeStamp"         type="uint32"     desc="Unit in
minutes" />
    <entry name="GameSvrID"         type="string"     size="64" />
    <entry name="GameAppID"         type="string"     size="64"
desc="gameapp id" />
    <entry name="OnlineCntIOS"      type="uint32"     defaultvalue="0"
desc="iOS Online User Count" />
    <entry name="OnlineCntAndroid"  type="uint32"     defaultvalue="0"
desc="Android Online User Count" />
    <entry name="BinaryLen"         type="smalluint" defaultvalue="1"
desc="Data source length; if length is 0, source check is ignored"/>
    <entry name="binary"            type="tinyint"     desc="Binary" count=
"1000" refer="BinaryLen" />
    <entry name="binary2"           type="tinyint"     desc="Binary2"
count= "1000" refer="BinaryLen" />
    <entry name="strstr"            type="string"     size="64"
desc="String"/>
    <index name="index_id"  column="TimeStamp"/>
</struct>
</metalib>
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
```

```
};
// The number of target cluster addresses
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Target business table name PLAYERONLINECNT
static const char * TABLE_NAME = "PLAYERONLINECNT";
```

Step 2: Initialize TcaplusAPI log handles

Log configuration file: tlogconf.xml

```
// TCaplus service log class
TcaplusService::TLogger* g_pstTlogger;
LPTLOGCATEGORYINST g_pstLogHandler;
LPTLOGCTX g_pstLogCtx;
int32_t InitLog()
{
    // Absolute path of the log configuration file
    const char* sLogConfFile = "tlogconf.xml";
    // Log class name
    const char* sCategoryName = "mytest";
    // Initialize the log handle using the configuration file
    g_pstLogCtx = tlog_init_from_file(sLogConfFile);
    if (NULL == g_pstLogCtx)
    {
        fprintf(stderr, "tlog_init_from_file failed.\n");
        return -1;
    }
    // Get the log class
    g_pstLogHandler = tlog_get_category(g_pstLogCtx, sCategoryName);
    if (NULL == g_pstLogHandler)
    {
        fprintf(stderr, "tlog_get_category(mytest) failed.\n");
        return -2;
    }
    // Initialize the log handle
    g_pstTlogger = new TcaplusService::TLogger(g_pstLogHandler);
    if (NULL == g_pstTlogger)
    {
```

```
        fprintf(stderr, "TcaplusService::TLogger failed.\n");
        return -3;
    }

    return 0;
}
```

Step 3: Initialize TcaplusAPI client

```
// The client main class of the TCaplus service API
TcaplusService::TcaplusServer g_stTcapSvr;
// Meta information for the table
extern unsigned char g_szMetalib_tcaplus_tb[];
LPTDRMETA g_szTableMeta = NULL;
int32_t InitServiceAPI()
{
    // Initialize
    int32_t iRet = g_stTcapSvr.Init(g_pstTlogger, /*module_id*/0, /*app
id*/APP_ID, /*zone id*/ZONE_ID, /*signature*/SIGNATURE);
    if (0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.Init failed, iRet:
%d.", iRet);
        return iRet;
    }

    // Add the directory server
    for (int32_t i = 0; i< DIR_URL_COUNT; i++)
    {
        iRet = g_stTcapSvr.AddDirServerAddress(DIR_URL_ARRAY[i]);
        if (0 != iRet)
        {
            tlog_error(g_pstLogHandler, 0, 0,
"g_stTcapSvr.AddDirServerAddress(%s) failed, iRet: %d.", DIR_URL_ARRAY[i],
iRet);
            return iRet;
        }
    }

    // Get the meta description of the table
    g_szTableMeta =
tdr_get_meta_by_name((LPTDRMETALIB)g_szMetalib_tcaplus_tb, TABLE_NAME);
    if(NULL == g_szTableMeta)
    {
```

```
        tlog_error(g_pstLogHandler, 0, 0,"tdr_get_meta_by_name(%s)
failed.", TABLE_NAME);
        return -1;
    }

    // Register the data table (connect to the directory server, verify the
password, and get the table routing) with 10-second timeout period
    iRet = g_stTcapSvr.RegistTable(TABLE_NAME, g_szTableMeta,
/*timeout_ms*/10000);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.RegistTable(%s)
failed, iRet: %d.", TABLE_NAME, iRet);
        return iRet;
    }

    // Connect to all TcaplusDB proxy servers corresponding to the table
    iRet = g_stTcapSvr.ConnectAll(/*timeout_ms*/10000, 0);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.ConnectAll failed,
iRet: %d.", iRet);
        return iRet;
    }

    return 0;
}
```

Step 4. Send a replace request via the API

You can also send TCAPLUS_API_INSERT_REQ to insert data. If data exists, it will return failed. TCAPLUS_API_REPLACE_REQ to insert data. If data exists, it will replace.

```
int32_t SendReplaceRequest()
{
    // Request object class
    TcaplusService::TcaplusServiceRequest* pstRequest =
g_stTcapSvr.GetRequest(TABLE_NAME);
    if (NULL == pstRequest)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.GetRequest(%s)
failed.", TABLE_NAME);
        return -1;
    }
```

```
    // Initialize the request object
    int iRet = pstRequest->Init(TCAPLUS_API_REPLACE_REQ, NULL, 0, 0, 0, 0);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest-
>Init(TCAPLUS_API_REPLACE_REQ) failed, iRet: %d.", iRet);
        return iRet;
    }

    // Add a record to the table according to the request
    TcaplusService::TcaplusServiceRecord* pstRecord = pstRequest-
>AddRecord();
    if (NULL == pstRecord)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest->AddRecord()
failed.");
        return -1;
    }

    PLAYERONLINECNT stPLAYERONLINECNT;
    memset(&stPLAYERONLINECNT, 0, sizeof(stPLAYERONLINECNT));

    // Specify the information of the key to be updated
    stPLAYERONLINECNT.dwTimeStamp = 1;
    snprintf(stPLAYERONLINECNT.szGameSvrID,
sizeof(stPLAYERONLINECNT.szGameSvrID), "%s", "mysvrid");

    // Specify the information of the value to be updated
    snprintf(stPLAYERONLINECNT.szGameAppID,
sizeof(stPLAYERONLINECNT.szGameAppID), "%s", "myappid");
    stPLAYERONLINECNT.dwOnlineCntIOS = 1;
    stPLAYERONLINECNT.dwOnlineCntAndroid = 1;

    // Set the record based on the TDR description
    iRet = pstRecord->SetData(&stPLAYERONLINECNT,
sizeof(stPLAYERONLINECNT));
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRecord->SetData() failed,
iRet: %d.", iRet);
        return iRet;
    }

    // Send the request packet
```

```
    iRet= g_stTcapSvr.SendRequest(pstRequest);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.SendRequest failed,
iRet: %d.", iRet);
        return iRet;
    }
    return 0;
}
```

Step 5. Receive the response via the API

```
int RecvResp(TcaplusServiceResponse*& response)
{
    // Block the reception of response packets 5,000 times with each block
lasting for 1 ms
    unsigned int sleep_us = 1000;
    unsigned int sleep_count = 5000;
    do
    {
        usleep(sleep_us);
        response = NULL;
        int ret = g_stTcapSvr.RecvResponse(response);
        if (ret < 0)
        {
            tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse
failed. ret:%d", ret);
            return ret;
        }

        // Receive a response packet
        if (1 == ret)
        {
             break;
        }
    } while ((--sleep_count) > 0);

    // Timeout period: 5 seconds
    if (0 == sleep_count)
    {
        tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse wait
timeout.");
        return -1;
    }
```

```
    return 0;
}
```

Example

Please refer to main.cpp file

```
int main(void) {
    // Initialize the log
    int ret = InitLog();
    if ( ret != 0)
    {
        printf("init log failed\n");
        return -1;
    }

    // Initialize the API client
    ret = InitServiceAPI();
    if ( ret != 0)
    {
        printf("init InitServiceAPI failed\n");
        return -1;
    }

    // Send the request
    ret = SendReplaceRequest();
    if (0 != ret)
    {
        printf("SendReplaceRequest failed\n");
        return -1;
    }

    // Receive the response
    TcaplusServiceResponse* response = NULL;
    ret = RecvResp(response);
    if (0 != ret)
    {
        printf("RecvResp failed\n");
        return -1;
    }

    //Get the operation result, 0 indicates success
    int32_t result = response->GetResult();
    if (0 != result)
```

```
    {
        printf("the result is %d\n", result);
        return -1;
    }
    return 0;
}
```

Reading data

Last updated: 2024-10-15 17:47:24

Prerequisites

Successfully created: Cluster, Table Group, PLAYERONLINECNT Table
PLAYERONLINECNT Table description file table_test.xml as follows:

```
<?xml version="1.0" encoding="GBK" standalone="yes" ?>
<metalib name="tcaplus_tb" tagsetversion="1" version="1">
<struct name="PLAYERONLINECNT" version="1" primarykey="TimeStamp,GameSvrID"
splittablekey="TimeStamp">
    <entry name="TimeStamp"         type="uint32"     desc="Unit in
minutes" />
    <entry name="GameSvrID"         type="string"     size="64" />
    <entry name="GameAppID"         type="string"     size="64"
desc="gameapp id" />
    <entry name="OnlineCntIOS"      type="uint32"     defaultvalue="0"
desc="iOS Online User Count" />
    <entry name="OnlineCntAndroid"  type="uint32"     defaultvalue="0"
desc="Android Online User Count" />
    <entry name="BinaryLen"         type="smalluint" defaultvalue="1"
desc="Data source length; if length is 0, source check is ignored"/>
    <entry name="binary"            type="tinyint"     desc="Binary" count=
"1000" refer="BinaryLen" />
    <entry name="binary2"           type="tinyint"     desc="Binary2"
count= "1000" refer="BinaryLen" />
    <entry name="strstr"            type="string"     size="64"
desc="String"/>
    <index name="index_id"  column="TimeStamp"/>
</struct>
</metalib>
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Target business table name PLAYERONLINECNT
static const char * TABLE_NAME = "PLAYERONLINECNT";
```

Step 2: Initialize TcaplusAPI log handles

Log configuration file: tlogconf.xml

```
// TCaplus service log class
TcaplusService::TLogger* g_pstTlogger;
LPTLOGCATEGORYINST g_pstLogHandler;
LPTLOGCTX g_pstLogCtx;
int32_t InitLog()
{
    // Absolute path of the log configuration file
    const char* sLogConfFile = "tlogconf.xml";
    // Log class name
    const char* sCategoryName = "mytest";
    // Initialize the log handle using the configuration file
    g_pstLogCtx = tlog_init_from_file(sLogConfFile);
    if (NULL == g_pstLogCtx)
    {
        fprintf(stderr, "tlog_init_from_file failed.\n");
        return -1;
    }
    // Get the log class
    g_pstLogHandler = tlog_get_category(g_pstLogCtx, sCategoryName);
    if (NULL == g_pstLogHandler)
    {
        fprintf(stderr, "tlog_get_category(mytest) failed.\n");
        return -2;
    }
    // Initialize the log handle
    g_pstTlogger = new TcaplusService::TLogger(g_pstLogHandler);
    if (NULL == g_pstTlogger)
    {
        fprintf(stderr, "TcaplusService::TLogger failed.\n");
        return -3;
```

```
    }

    return 0;
}
```

Step 3: Initialize TcaplusAPI client

```
// The client main class of the TCaplus service API
TcaplusService::TcaplusServer g_stTcapSvr;
// Meta information for the table
extern unsigned char g_szMetalib_tcaplus_tb[];
LPTDRMETA g_szTableMeta = NULL;
int32_t InitServiceAPI()
{
    // Initialize
    int32_t iRet = g_stTcapSvr.Init(g_pstTlogger, /*module_id*/0, /*app
id*/APP_ID, /*zone id*/ZONE_ID, /*signature*/SIGNATURE);
    if (0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.Init failed, iRet:
%d.", iRet);
        return iRet;
    }

    // Add the directory server
    for (int32_t i = 0; i< DIR_URL_COUNT; i++)
    {
        iRet = g_stTcapSvr.AddDirServerAddress(DIR_URL_ARRAY[i]);
        if (0 != iRet)
        {
            tlog_error(g_pstLogHandler, 0, 0,
"g_stTcapSvr.AddDirServerAddress(%s) failed, iRet: %d.", DIR_URL_ARRAY[i],
iRet);
            return iRet;
        }
    }

    // Get the meta description of the table
    g_szTableMeta =
tdr_get_meta_by_name((LPTDRMETALIB)g_szMetalib_tcaplus_tb, TABLE_NAME);
    if(NULL == g_szTableMeta)
    {
        tlog_error(g_pstLogHandler, 0, 0,"tdr_get_meta_by_name(%s)
failed.", TABLE_NAME);
```

```
        return -1;
    }

    // Register the data table (connect to the directory server, verify the
password, and get the table routing) with 10-second timeout period
    iRet = g_stTcapSvr.RegistTable(TABLE_NAME, g_szTableMeta,
/*timeout_ms*/10000);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.RegistTable(%s)
failed, iRet: %d.", TABLE_NAME, iRet);
        return iRet;
    }

    // Connect to all TcaplusDB proxy servers corresponding to the table
    iRet = g_stTcapSvr.ConnectAll(/*timeout_ms*/10000, 0);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.ConnectAll failed,
iRet: %d.", iRet);
        return iRet;
    }

    return 0;
}
```

Step 4. Send a get request via the API

```
int32_t SendGetRequest()
{
    // Request object class
    TcaplusService::TcaplusServiceRequest* pstRequest =
g_stTcapSvr.GetRequest(TABLE_NAME);
    if (NULL == pstRequest)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.GetRequest(%s)
failed.", TABLE_NAME);
        return -1;
    }

    // Initialize the request object
    int iRet = pstRequest->Init(TCAPLUS_API_GET_REQ, NULL, 0, 0, 0, 0);
    if(0 != iRet)
    {
```

```
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest-
>Init(TCAPLUS_API_GET_REQ) failed, iRet: %d.", iRet);
        return iRet;
    }

    // Add a record to the table according to the request
    TcaplusService::TcaplusServiceRecord* pstRecord = pstRequest-
>AddRecord();
    if (NULL == pstRecord)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest->AddRecord()
failed.");
        return -1;
    }

    PLAYERONLINECNT stPLAYERONLINECNT;
    memset(&stPLAYERONLINECNT, 0, sizeof(stPLAYERONLINECNT));

    // Specify the information of the key to be queried
    stPLAYERONLINECNT.dwTimeStamp = 1;
    snprintf(stPLAYERONLINECNT.szGameSvrID,
sizeof(stPLAYERONLINECNT.szGameSvrID), "%s", "mysvrid");

    // Set the record based on the TDR description
    iRet = pstRecord->SetData(&stPLAYERONLINECNT,
sizeof(stPLAYERONLINECNT));
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRecord->SetData() failed,
iRet: %d.", iRet);
        return iRet;
    }

    // Send the request packet
    iRet= g_stTcapSvr.SendRequest(pstRequest);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.SendRequest failed,
iRet: %d.", iRet);
        return iRet;
    }
    return 0;
}
```

Step 5. Receive the response via the API

```
int RecvResp(TcaplusServiceResponse*& response)
{
    // Block the reception of response packets 5,000 times with each block
lasting for 1 ms
    unsigned int sleep_us = 1000;
    unsigned int sleep_count = 5000;
    do
    {
        usleep(sleep_us);
        response = NULL;
        int ret = g_stTcapSvr.RecvResponse(response);
        if (ret < 0)
        {
            tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse
failed. ret:%d", ret);
            return ret;
        }

        // Receive a response packet
        if (1 == ret)
        {
             break;
        }
    } while ((--sleep_count) > 0);

    // Timeout period: 5 seconds
    if (0 == sleep_count)
    {
        tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse wait
timeout.");
        return -1;
    }
    return 0;
}
```

Example

Please refer to main.cpp file

```
int main(void) {
    // Initialize the log
    int ret = InitLog();
```

```
    if (ret != 0)
    {
        printf("init log failed\n");
        return -1;
    }

    // Initialize the API client
    ret = InitServiceAPI();
    if ( ret != 0)
    {
        printf("init InitServiceAPI failed\n");
        return -1;
    }

    // Send the request
    ret = SendGetRequest();
    if (0 != ret)
    {
        printf("SendGetRequest failed\n");
        return -1;
    }

    // Receive the response
    TcaplusServiceResponse* response = NULL;
    ret = RecvResp(response);
    if (0 != ret)
    {
        printf("RecvResp failed\n");
        return -1;
    }

    //Get the operation result, 0 indicates success
    int32_t result = response->GetResult();
    if (0 != result)
    {
        printf("the result is %d\n", result);
        return -1;
    }

    // Get the record in the response
    //The batch command may contain multiple records in one response
    int count = 0;
    const TcaplusService::TcaplusServiceRecord* const_record = NULL;
    while((count++) < response->GetRecordCount())
    {
```

```
        // Read the record
        const_record = NULL;
        printf("---   ---   %3d   ---   ---\n", count - 1);
        ret = response->FetchRecord(const_record);
        if(0 != ret)
        {
            printf("FetchRecord() ret:%d\n", ret);
            continue;
        }

        // Assign the record to the structure
        PLAYERONLINECNT stPLAYERONLINECNT;
        memset(&stPLAYERONLINECNT, 0, sizeof(stPLAYERONLINECNT));
        ret = const_record->GetData(&stPLAYERONLINECNT,
sizeof(stPLAYERONLINECNT));
        if(0 != ret)
        {
            printf("const_record->GetData() failed, ret: %d.", ret);
            continue;
        }

        // Print the values in the structure
        printf("struct dwTimeStamp %d szGameSvrID %s szGameAppID %s
dwOnlineCntIOS %d dwOnlineCntAndroid %d \n",
            stPLAYERONLINECNT.dwTimeStamp,
            stPLAYERONLINECNT.szGameSvrID,
            stPLAYERONLINECNT.szGameAppID,
            stPLAYERONLINECNT.dwOnlineCntIOS,
            stPLAYERONLINECNT.dwOnlineCntAndroid);
    }
    return 0;
}
```

Updating Data

Last updated: 2024-10-15 17:47:45

Prerequisites

Successfully created: Cluster, Table Group, PLAYERONLINECNT Table
PLAYERONLINECNT Table description file table_test.xml as follows:

```
<?xml version="1.0" encoding="GBK" standalone="yes" ?>
<metalib name="tcaplus_tb" tagsetversion="1" version="1">
<struct name="PLAYERONLINECNT" version="1" primarykey="TimeStamp,GameSvrID"
splittablekey="TimeStamp">
    <entry name="TimeStamp"         type="uint32"     desc="Unit in
minutes" />
    <entry name="GameSvrID"         type="string"     size="64" />
    <entry name="GameAppID"         type="string"     size="64"
desc="gameapp id" />
    <entry name="OnlineCntIOS"      type="uint32"     defaultvalue="0"
desc="iOS Online User Count" />
    <entry name="OnlineCntAndroid"  type="uint32"     defaultvalue="0"
desc="Android Online User Count" />
    <entry name="BinaryLen"         type="smalluint" defaultvalue="1"
desc="Data source length; if length is 0, source check is ignored"/>
    <entry name="binary"            type="tinyint"     desc="Binary" count=
"1000" refer="BinaryLen" />
    <entry name="binary2"           type="tinyint"     desc="Binary2"
count= "1000" refer="BinaryLen" />
    <entry name="strstr"            type="string"     size="64"
desc="String"/>
    <index name="index_id"  column="TimeStamp"/>
</struct>
</metalib>
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Target business table name PLAYERONLINECNT
static const char * TABLE_NAME = "PLAYERONLINECNT";
```

Step 2: Initialize TcaplusAPI log handles

Log configuration file: tlogconf.xml

```
// TCaplus service log class
TcaplusService::TLogger* g_pstTlogger;
LPTLOGCATEGORYINST g_pstLogHandler;
LPTLOGCTX g_pstLogCtx;
int32_t InitLog()
{
    // Absolute path of the log configuration file
    const char* sLogConfFile = "tlogconf.xml";
    // Log class name
    const char* sCategoryName = "mytest";
    // Initialize the log handle using the configuration file
    g_pstLogCtx = tlog_init_from_file(sLogConfFile);
    if (NULL == g_pstLogCtx)
    {
        fprintf(stderr, "tlog_init_from_file failed.\n");
        return -1;
    }
    // Get the log class
    g_pstLogHandler = tlog_get_category(g_pstLogCtx, sCategoryName);
    if (NULL == g_pstLogHandler)
    {
        fprintf(stderr, "tlog_get_category(mytest) failed.\n");
        return -2;
    }
    // Initialize the log handle
    g_pstTlogger = new TcaplusService::TLogger(g_pstLogHandler);
    if (NULL == g_pstTlogger)
    {
        fprintf(stderr, "TcaplusService::TLogger failed.\n");
        return -3;
```

```
    }

    return 0;
}
```

Step 3: Initialize TcaplusAPI client

```
// The client main class of the TCaplus service API
TcaplusService::TcaplusServer g_stTcapSvr;
// Meta information for the table
extern unsigned char g_szMetalib_tcaplus_tb[];
LPTDRMETA g_szTableMeta = NULL;
int32_t InitServiceAPI()
{
    // Initialize
    int32_t iRet = g_stTcapSvr.Init(g_pstTlogger, /*module_id*/0, /*app
id*/APP_ID, /*zone id*/ZONE_ID, /*signature*/SIGNATURE);
    if (0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.Init failed, iRet:
%d.", iRet);
        return iRet;
    }

    // Add the directory server
    for (int32_t i = 0; i< DIR_URL_COUNT; i++)
    {
        iRet = g_stTcapSvr.AddDirServerAddress(DIR_URL_ARRAY[i]);
        if (0 != iRet)
        {
            tlog_error(g_pstLogHandler, 0, 0,
"g_stTcapSvr.AddDirServerAddress(%s) failed, iRet: %d.", DIR_URL_ARRAY[i],
iRet);
            return iRet;
        }
    }

    // Get the meta description of the table
    g_szTableMeta =
tdr_get_meta_by_name((LPTDRMETALIB)g_szMetalib_tcaplus_tb, TABLE_NAME);
    if(NULL == g_szTableMeta)
    {
        tlog_error(g_pstLogHandler, 0, 0,"tdr_get_meta_by_name(%s)
failed.", TABLE_NAME);
```

```
        return -1;
    }

    // Register the data table (connect to the directory server, verify the
password, and get the table routing) with 10-second timeout period
    iRet = g_stTcapSvr.RegistTable(TABLE_NAME, g_szTableMeta,
/*timeout_ms*/10000);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.RegistTable(%s)
failed, iRet: %d.", TABLE_NAME, iRet);
        return iRet;
    }

    // Connect to all TcaplusDB proxy servers corresponding to the table
    iRet = g_stTcapSvr.ConnectAll(/*timeout_ms*/10000, 0);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.ConnectAll failed,
iRet: %d.", iRet);
        return iRet;
    }

    return 0;
}
```

Step 4. Send an update request via the API

```
int32_t SendUpdateRequest()
{
    // Request object class
    TcaplusService::TcaplusServiceRequest* pstRequest =
g_stTcapSvr.GetRequest(TABLE_NAME);
    if (NULL == pstRequest)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.GetRequest(%s)
failed.", TABLE_NAME);
        return -1;
    }

    // Initialize the request object
    int iRet = pstRequest->Init(TCAPLUS_API_UPDATE_REQ, NULL, 0, 0, 0, 0);
    if(0 != iRet)
    {
```

```
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest-
>Init(TCAPLUS_API_UPDATE_REQ) failed, iRet: %d.", iRet);
        return iRet;
    }

    // Add a record to the table according to the request
    TcaplusService::TcaplusServiceRecord* pstRecord = pstRequest-
>AddRecord();
    if (NULL == pstRecord)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest->AddRecord()
failed.");
        return -1;
    }

    PLAYERONLINECNT stPLAYERONLINECNT;
    memset(&stPLAYERONLINECNT, 0, sizeof(stPLAYERONLINECNT));

    // Specify the information of the key to be updated
    stPLAYERONLINECNT.dwTimeStamp = 1;
    snprintf(stPLAYERONLINECNT.szGameSvrID,
sizeof(stPLAYERONLINECNT.szGameSvrID), "%s", "mysvrid");

    // Specify the information of the value to be updated
    snprintf(stPLAYERONLINECNT.szGameAppID,
sizeof(stPLAYERONLINECNT.szGameAppID), "%s", "myappid2");
    stPLAYERONLINECNT.dwOnlineCntIOS = 2;
    stPLAYERONLINECNT.dwOnlineCntAndroid = 2;

    // Set the record based on the TDR description
    iRet = pstRecord->SetData(&stPLAYERONLINECNT,
sizeof(stPLAYERONLINECNT));
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRecord->SetData() failed,
iRet: %d.", iRet);
        return iRet;
    }

    // Send the request packet
    iRet= g_stTcapSvr.SendRequest(pstRequest);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.SendRequest failed,
iRet: %d.", iRet);
```

```
        return iRet;
    }
    return 0;
}
```

Step 5. Receive the response via the API

```
int RecvResp(TcaplusServiceResponse*& response)
{
    // Block the reception of response packets 5,000 times with each block
lasting for 1 ms
    unsigned int sleep_us = 1000;
    unsigned int sleep_count = 5000;
    do
    {
        usleep(sleep_us);
        response = NULL;
        int ret = g_stTcapSvr.RecvResponse(response);
        if (ret < 0)
        {
            tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse
failed. ret:%d", ret);
            return ret;
        }

        // Receive a response packet
        if (1 == ret)
        {
             break;
        }
    } while ((--sleep_count) > 0);

    // Timeout period: 5 seconds
    if (0 == sleep_count)
    {
        tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse wait
timeout.");
        return -1;
    }
    return 0;
}
```

Example

Please refer to main.cpp file

```
int main(void) {
    // Initialize the log
    int ret = InitLog();
    if ( ret != 0)
    {
        printf("init log failed\n");
        return -1;
    }

    // Initialize the API client
    ret = InitServiceAPI();
    if ( ret != 0)
    {
        printf("init InitServiceAPI failed\n");
        return -1;
    }

    // Send the request
    ret = SendUpdateRequest();
    if (0 != ret)
    {
        printf("SendUpdateRequest failed\n");
        return -1;
    }

    // Receive the response
    TcaplusServiceResponse* response = NULL;
    ret = RecvResp(response);
    if (0 != ret)
    {
        printf("RecvResp failed\n");
        return -1;
    }

    //Get the operation result, 0 indicates success
    int32_t result = response->GetResult();
    if (0 != result)
    {
        printf("the result is %d\n", result);
        return -1;
    }

    return 0;
```

```
}
```

Deleting Data

Last updated: 2024-10-15 17:48:10

Prerequisites

Successfully created: Cluster, Table Group, PLAYERONLINECNT Table
PLAYERONLINECNT Table description file table_test.xml as follows:

```
<?xml version="1.0" encoding="GBK" standalone="yes" ?>
<metalib name="tcaplus_tb" tagsetversion="1" version="1">
<struct name="PLAYERONLINECNT" version="1" primarykey="TimeStamp,GameSvrID"
splittablekey="TimeStamp">
    <entry name="TimeStamp"         type="uint32"     desc="Unit in
minutes" />
    <entry name="GameSvrID"         type="string"     size="64" />
    <entry name="GameAppID"         type="string"     size="64"
desc="gameapp id" />
    <entry name="OnlineCntIOS"      type="uint32"     defaultvalue="0"
desc="iOS Online User Count" />
    <entry name="OnlineCntAndroid"  type="uint32"     defaultvalue="0"
desc="Android Online User Count" />
    <entry name="BinaryLen"         type="smalluint" defaultvalue="1"
desc="Data source length; if length is 0, source check is ignored"/>
    <entry name="binary"            type="tinyint"     desc="Binary" count=
"1000" refer="BinaryLen" />
    <entry name="binary2"           type="tinyint"     desc="Binary2"
count= "1000" refer="BinaryLen" />
    <entry name="strstr"            type="string"     size="64"
desc="String"/>
    <index name="index_id"  column="TimeStamp"/>
</struct>
</metalib>
```

Step 1: Define configuration parameters

```
// Access address of the target cluster
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
   "tcp://10.191.***.99:9999",
   "tcp://10.191.***.88:9999"
};
// The number of target cluster addresses
```

```
static const int32_t DIR_URL_COUNT = 2;
// Cluster ID of the target business
static const int32_t APP_ID = 3;
// Table group ID of the target business
static const int32_t ZONE_ID = 1;
// Target business password
static const char * SIGNATURE = "*******";
// Target business table name PLAYERONLINECNT
static const char * TABLE_NAME = "PLAYERONLINECNT";
```

Step 2: Initialize TcaplusAPI log handles

Log configuration file: tlogconf.xml

```
// TCaplus service log class
TcaplusService::TLogger* g_pstTlogger;
LPTLOGCATEGORYINST g_pstLogHandler;
LPTLOGCTX g_pstLogCtx;
int32_t InitLog()
{
    // Absolute path of the log configuration file
    const char* sLogConfFile = "tlogconf.xml";
    // Log class name
    const char* sCategoryName = "mytest";
    // Initialize the log handle using the configuration file
    g_pstLogCtx = tlog_init_from_file(sLogConfFile);
    if (NULL == g_pstLogCtx)
    {
        fprintf(stderr, "tlog_init_from_file failed.\n");
        return -1;
    }
    // Get the log class
    g_pstLogHandler = tlog_get_category(g_pstLogCtx, sCategoryName);
    if (NULL == g_pstLogHandler)
    {
        fprintf(stderr, "tlog_get_category(mytest) failed.\n");
        return -2;
    }
    // Initialize the log handle
    g_pstTlogger = new TcaplusService::TLogger(g_pstLogHandler);
    if (NULL == g_pstTlogger)
    {
        fprintf(stderr, "TcaplusService::TLogger failed.\n");
        return -3;
```

```
    }

    return 0;
}
```

Step 3: Initialize TcaplusAPI client

```
// The client main class of the TCaplus service API
TcaplusService::TcaplusServer g_stTcapSvr;
// Meta information for the table
extern unsigned char g_szMetalib_tcaplus_tb[];
LPTDRMETA g_szTableMeta = NULL;
int32_t InitServiceAPI()
{
    // Initialize
    int32_t iRet = g_stTcapSvr.Init(g_pstTlogger, /*module_id*/0, /*app
id*/APP_ID, /*zone id*/ZONE_ID, /*signature*/SIGNATURE);
    if (0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.Init failed, iRet:
%d.", iRet);
        return iRet;
    }

    // Add the directory server
    for (int32_t i = 0; i< DIR_URL_COUNT; i++)
    {
        iRet = g_stTcapSvr.AddDirServerAddress(DIR_URL_ARRAY[i]);
        if (0 != iRet)
        {
            tlog_error(g_pstLogHandler, 0, 0,
"g_stTcapSvr.AddDirServerAddress(%s) failed, iRet: %d.", DIR_URL_ARRAY[i],
iRet);
            return iRet;
        }
    }

    // Get the meta description of the table
    g_szTableMeta =
tdr_get_meta_by_name((LPTDRMETALIB)g_szMetalib_tcaplus_tb, TABLE_NAME);
    if(NULL == g_szTableMeta)
    {
        tlog_error(g_pstLogHandler, 0, 0,"tdr_get_meta_by_name(%s)
failed.", TABLE_NAME);
```

```
        return -1;
    }

    // Register the data table (connect to the directory server, verify the
password, and get the table routing) with 10-second timeout period
    iRet = g_stTcapSvr.RegistTable(TABLE_NAME, g_szTableMeta,
/*timeout_ms*/10000);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.RegistTable(%s)
failed, iRet: %d.", TABLE_NAME, iRet);
        return iRet;
    }

    // Connect to all TcaplusDB proxy servers corresponding to the table
    iRet = g_stTcapSvr.ConnectAll(/*timeout_ms*/10000, 0);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.ConnectAll failed,
iRet: %d.", iRet);
        return iRet;
    }

    return 0;
}
```

Step 4. Send a delete request via the API

```
int32_t SendDeleteRequest()
{
    // Request object class
    TcaplusService::TcaplusServiceRequest* pstRequest =
g_stTcapSvr.GetRequest(TABLE_NAME);
    if (NULL == pstRequest)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.GetRequest(%s)
failed.", TABLE_NAME);
        return -1;
    }

    // Initialize the request object
    int iRet = pstRequest->Init(TCAPLUS_API_DELETE_REQ, NULL, 0, 0, 0, 0);
    if(0 != iRet)
    {
```

```
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest-
>Init(TCAPLUS_API_DELETE_REQ) failed, iRet: %d.", iRet);
        return iRet;
    }

    // Add a record to the table according to the request
    TcaplusService::TcaplusServiceRecord* pstRecord = pstRequest-
>AddRecord();
    if (NULL == pstRecord)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRequest->AddRecord()
failed.");
        return -1;
    }

    PLAYERONLINECNT stPLAYERONLINECNT;
    memset(&stPLAYERONLINECNT, 0, sizeof(stPLAYERONLINECNT));

    // Specify the information of the key to be deleted
    stPLAYERONLINECNT.dwTimeStamp = 1;
    snprintf(stPLAYERONLINECNT.szGameSvrID,
sizeof(stPLAYERONLINECNT.szGameSvrID), "%s", "mysvrid");


    // Set the record based on the TDR description
    iRet = pstRecord->SetData(&stPLAYERONLINECNT,
sizeof(stPLAYERONLINECNT));
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "pstRecord->SetData() failed,
iRet: %d.", iRet);
        return iRet;
    }

    // Send the request packet
    iRet= g_stTcapSvr.SendRequest(pstRequest);
    if(0 != iRet)
    {
        tlog_error(g_pstLogHandler, 0, 0, "g_stTcapSvr.SendRequest failed,
iRet: %d.", iRet);
        return iRet;
    }
    return 0;
}
```

Step 5. Receive the response via the API

```
int RecvResp(TcaplusServiceResponse*& response)
{
    // Block the reception of response packets 5,000 times with each block
lasting for 1 ms
    unsigned int sleep_us = 1000;
    unsigned int sleep_count = 5000;
    do
    {
        usleep(sleep_us);
        response = NULL;
        int ret = g_stTcapSvr.RecvResponse(response);
        if (ret < 0)
        {
            tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse
failed. ret:%d", ret);
            return ret;
        }

        // Receive a response packet
        if (1 == ret)
        {
             break;
        }
    } while ((--sleep_count) > 0);

    // Timeout period: 5 seconds
    if (0 == sleep_count)
    {
        tlog_error(g_pstLogHandler, 0, 0, "tcaplus_server.RecvResponse wait
timeout.");
        return -1;
    }
    return 0;
}
```

Example

Please refer to main.cpp file

```
int main(void) {
    // Initialize the log
    int ret = InitLog();
```

```
    if ( ret != 0)
    {
        printf("init log failed\n");
        return -1;
    }

    // Initialize the API client
    ret = InitServiceAPI();
    if ( ret != 0)
    {
        printf("init InitServiceAPI failed\n");
        return -1;
    }

    // Send the request
    ret = SendDeleteRequest();
    if (0 != ret)
    {
        printf("SendDeleteRequest failed\n");
        return -1;
    }

    // Receive the response
    TcaplusServiceResponse* response = NULL;
    ret = RecvResp(response);
    if (0 != ret)
    {
        printf("RecvResp failed\n");
        return -1;
    }

    //Get the operation result, 0 indicates success
    int32_t result = response->GetResult();
    if (0 != result)
    {
        printf("the result is %d\n", result);
        return -1;
    }

    return 0;
}
```