Multiple Network Acceleration

Multiple Network Acceleration SDK

Product Documentation

Contents

Multiple Network Acceleration SDK
SDK Overview

Last updated：2026-05-21 11:38:13

Multiple Network Acceleration (MNA) SDK is a software development kit specially designed to improve data transmission efficiency and stability in network environments. It is intended to intercept accelerated traffic on the terminal by intelligently integrating multiple types of network resources (such as 4G/5G cellular networks, Wi-Fi, satellite links, etc.). Through multi-channel acceleration, it guarantees the transmission latency and transmission rate for users, and is widely applicable to scenarios such as gaming, video on demand (VOD) and live streaming. Based on different business network requirements, the SDK encapsulates original packets into tunnel packets, dynamically distributes them across multiple physical links, and resolves mobile network access issues in various scenarios through the reuse of multiple physical links. It provides developers with low-latency, high-reliability, and high-bandwidth network access capabilities, helping to quickly build high-performance and highly stable network communication applications.

SDK Download

Last updated：2026-05-21 16:46:58

Download SDK

Multiple Network Acceleration SDK is suitable for Windows/iOS/Linux/Android 64-bit systems and provides URL library format integration.
You can download the Multiple Network Acceleration SDK and get the Demo project on this page. Get the Demo App in Demo Download and Experience to experience the feature.

Android SDK

SDK Name	Android SDK
Version	2.9.8. For the release notes, see Release Notes.
SDK Introduction	Ensures user transmission latency and rate through multi-channel acceleration, and is widely applicable to scenarios such as online gaming, video-on-demand, and live streaming.
Developer	Tencent Cloud Computing (Beijing) Co., Ltd.
Access Documentation	SDK Download Demo Project Download Quick Integration

iOS SDK

SDK Name	iOS SDK
Version	2.9.8. For the release notes, see Release Notes.
SDK Introduction	Ensures user transmission latency and rate through multi-channel acceleration, and is widely applicable to scenarios such as online gaming, video-on-demand, and live streaming.
Developer	Tencent Cloud Computing (Beijing) Co., Ltd.
Access Documentation	SDK Download Swift Demo Project Download OC Demo Project Download Quick Integration

Linux SDK

SDK Name	Linux SDK
Version	0.24.2. For the release notes, see Release Notes.
SDK Introduction	Ensures user transmission latency and rate through multi-channel acceleration, and is widely applicable to scenarios such as online gaming, video-on-demand, and live streaming.
Developer	Tencent Cloud Computing (Beijing) Co., Ltd.
Access Documentation	0.24.2 SDK Download API Overview

Windows SDK

SDK Name	Windows SDK
Version	0.23.1. For the release notes, see Release Notes.
SDK Introduction	Ensures user transmission latency and rate through multi-channel acceleration, and is widely applicable to scenarios such as online gaming, video-on-demand, and live streaming.
Developer	Tencent Cloud Computing (Beijing) Co., Ltd.
Access Documentation	0.23.1 SDK Download 0.23.1 dll SDK Download API Overview

Android SDK
Quick Connection

Last updated：2026-05-21 16:49:19

Development Environment Requirement

- Android Studio 2.0+
- Android 7.0 (SDK API 24) and above

SDK Reference and Dependency Configuration

1. **Maven support for importing.**

```
//Multi-network acceleration SDK
implementation 'com.tencent.linkboost:mpacc:2.9.4'
```

2. **Add via AAR package**

> **Note:**
> If you add the SDK via Maven, you can skip this step.

- Copy the AAR file to the project directory (for example: app/libs).
- Add the following dependency configuration to the Gradle file:

```
dependencies {
//add SDK dependency
implementation fileTree(dir: 'libs', include: ['*.aar'])
//add gson dependency
implementation 'com.google.code.gson:gson:2.8.9'
//add encryption library dependency
implementation 'androidx.security:security-crypto:1.0.0'
}
```

International Site Configuration

The `setEnv` API of the `MpAccRegister` class defaults to the Mainland China site.

```
// `Context` refers to the `ApplicationContext`. `env` is an integer
value: 0 for the Mainland China site, 1 for the International site.
MpAccRegister.setEnv(context, env);
```

Sample Code

VPN Mode

Initializing MpAccClient

> **Note:**
> SDK authentication also supports application-based integration. You can call MpAccClient.setSign("appid","*****").
> You only need to configure either device-based integration or application-based integration.
> For the sign generation method, see Getting Started Guide in the console..

```
    private void initMpAcc() {
        // The datakey applied from Tencent Cloud needs to be imported
here (use device connectivity).
        MpAccClient.setDataKey("test-123456", "*");
        mpAccClient = MpAccClient.getInstance(this);
    }
```

Start acceleration

```
        try {
            mpAccClient.registerAccCallback(accCallback);
            //vpn acceleration requires user authorization
            Intent vpnIntent = MpAccClient.prepare(this);
            if (vpnIntent != null) {
                startActivityForResult(vpnIntent,
                VPN_REQUEST_CODE);
            } else {
                onActivityResult(VPN_REQUEST_CODE, RESULT_OK,
                null);
            }
```

```
        } catch (MpAccSDKException e) {
            e.printStackTrace();
        }
    }



    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intentdata) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == VPN_REQUEST_CODE && resultCode == RESULT_OK)
{
            try {
                AccConfig accConfig = new AccConfig();
                ArrayList<String> list = new ArrayList<>();
                accConfig.setAccMode(AccConfig.ACC_MOD_BONDING); //set
acceleration mode
                accConfig.setRoute("0.0.0.0", 0); //ip list for
acceleration
                //list.add("air.tv.douyu.android"); //input the package
name to accelerate, default if not passed accelerates ALL applications
                accConfig.setWhiteList(list, 0);
                mpAccClient.startAcc(accConfig);
            } catch (MpAccSDKException e) {
                e.printStackTrace();
            }
        }
    }
```

Releasing Resources

```
    //Free up resources when exiting, such as calling in the Activity's
onDestroy method
    private void stopAcc() {
        try {
            mpAccClient.unregisterAccCallback(accCallback);
            mpAccClient.stopAcc();
        } catch (MpAccSDKException e) {
```

```
            e.printStackTrace();
        }
    }
```

SOCKS Mode

> **Note:**
> SOCKS mode acceleration does not require user authorization. The initialization, authentication, and status callback logic of MpAccClient is the same as in VPN mode.

```
    private void initMpAcc() {
        MpAccClient.setDataKey("test-123456", "*");
        mpAccClient = MpAccClient.getInstance(this);
    }

    //Trigger acceleration (socks mode)
    private void startSocksAcc() {
        AccConfig accConfig = new AccConfig();
        accConfig.setAccMode(AccConfig.ACC_MOD_BONDING)
                .setPingInterval(3)
                .setEnableSocks(true) //Socks mode needs to be set to
true
                .setSocksPort(1080);//Set proxy port
        try {
            //Listen to acceleration status callback
            mpAccClient.registerAccCallback(accCallback);
            mpAccClient.startAcc(accConfig);
        } catch (MpAccSDKException e) {
            e.printStackTrace();
        }
    }

    private void stopSocksAcc() {
        try {
            mpAccClient.unregisterAccCallback(accCallback);
            mpAccClient.stopAcc();
        } catch (MpAccSDKException e) {
            e.printStackTrace();
        }
```

```
    }
```

> **Note:**
> After acceleration is successfully started in SOCKS mode (that is, after receiving the onAccSuccess callback), your service must send network requests through the SOCKS protocol.
> **The proxy IP is the local IP, and the proxy port must be the same as the port set through setSocksPort when starting acceleration.** Only then can the traffic enter the acceleration tunnel.

The following are common examples of switching network libraries to SOCKS5:

```
    //Example of sending packets with socks5 over UDP
    private void testUdpSocks() {
        try {
            Log.i(TAG, "start testUdp");
            //1. The port should be consistent with the setSocksPort
used to trigger acceleration.
            SocksProxy socksProxy = new Socks5(new
            InetSocketAddress("localhost", 1080));
            //2. Set socksProxy to DatagramSocket
            DatagramSocket clientSocket = new
            Socks5Datagram(socksProxy);
            //3. Perform normal packet sending business
            String message = "Hi, I am UDP client";
            byte[] sendBuffer = message.getBytes();
            DatagramPacket packet =
                    new DatagramPacket(sendBuffer, sendBuffer.length,
                    new InetSocketAddress("106.55.119.181", 8888));
            clientSocket.send(packet);
            //Received response message from UDP server.
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //HttpURLConnection uses socks Proxy
    private void testHttpSocks() {
        BufferedReader buff = null;
        HttpURLConnection urlConnection = null;
```

```
        try {
             Create a proxy agent
             Proxy socksProxy =  new Proxy(Proxy.Type.SOCKS, new
             InetSocketAddress("127.0.0.1", 1080));
            URL url = new URL("url");
            //Add Proxy to HttpURLConnection
            urlConnection = (HttpURLConnection)
            url.openConnection(socksProxy);
             //Perform normal packet sending business
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //OKHttp uses socks Proxy
    private void testOkHttpSocks(){
          Create a proxy agent
          Proxy socksProxy =new Proxy(Proxy.Type.SOCKS,
          new InetSocketAddress("127.0.0.1", 1080));
          //Add Proxy to OkHttpClient
          OkHttpClient client = new
          OkHttpClient.Builder().proxy(socksProxy).build();
          //Perform normal packet sending business
    }

   //TCP Socket uses socks Proxy
   private void testSocketSocks() {
        Proxy socksProxy =new Proxy(Proxy.Type.SOCKS,
        new  InetSocketAddress("127.0.0.1", 1080));
        Socket socket = new Socket(socksProxy);
        //...
    }
```

C++ SOCKS Mode

For TCP and UDP access to SOCKS, see:
GitHub – Logotipo/socks5-client-Linux: SOCKS5 client for Linux with supporting of CONNECT and UDP_ASSOCIATE

FAQs

1. **If I use the SDK in VPN mode for acceleration, will acceleration still take effect if another third-party app also enables VPN?**

Answer:No. By default, Android allows only one VPN service at a time. If another application also enables VPN, the current acceleration VPN service will be released, the onAccFail callback will be triggered with error code -5, and acceleration will stop.

2. **How do I use SOCKS mode for acceleration?**

Answer:To use SOCKS mode for acceleration, in addition to calling startAcc to configure the acceleration mode and SOCKS port, you must also encapsulate your service-related network interfaces to send traffic through a SOCKS5 proxy. For examples, see the SOCKS mode sample code.

3. **What is the difference between SOCKS mode and VPN mode acceleration?**

Answer:Both modes are designed to obtain service data traffic for accelerated forwarding.VPN mode requires the user to approve an authorization prompt, and a small key icon appears in the system UI. In VPN mode, any third-party application on the phone can be accelerated.SOCKS mode does not require user authorization, but you must modify the service network sending interface so that traffic is sent through a SOCKS proxy.

4. **Can dual Wi-Fi or a custom USB network adapter be used for multi-network acceleration?**

Currently, the Android SDK supports acceleration for three types of network adapters: Wi-Fi, MOBILE, and ETH. You can call setAccLinks to add the network adapters you want to use for acceleration.
If you want to add a network adapter type that is not supported by the SDK, you can dynamically add a custom network adapter through addNetworkPlugin. For plugin code, see:
MpNetworkPluginImpl.java

5. **What is the difference between the onStopMpAcc and onAccException callbacks in MeasureCallback?**

Answer:The SDK callback onStopMpAcc notifies the user to stop acceleration. It is mainly triggered when the device is in a single-network-adapter environment, when the default route network has returned to normal, or when there has been no obvious acceleration benefit for a long time. Whether to finally stop acceleration (stopAcc) is up to the user.
The SDK callback onAccException indicates that a major exception has occurred during acceleration, such as network disconnection after acceleration starts, excessively high latency, Negative optimization, or acceleration results that are far below expectations. In this case, the SDK proactively stops acceleration and speed testing so that service traffic returns to the origin path, preventing the acceleration exception from affecting normal service usage.

6. **How can I obtain the total traffic used in a single acceleration session and the bandwidth rate of each link during acceleration?**

Answer:AccCallback provides the onAccDataUpdate callback. Its parameters include the total accelerated uplink and downlink traffic, as well as detailed information about each link (PathDetail).

7. **The minSdkVersion of the SDK is 24, which is higher than the minSdkVersion configured in my app. How do I resolve the packaging/compilation error?**

Answer:Add the following configuration to the manifest file, and check the system version dynamically in code. If the current system version is lower than 24, do not call the SDK APIs.

```
<uses-sdk
tools:overrideLibrary="com.android.linkboost.multi,androidx.security"/>
```

8. **After acceleration is initiated, even without actual service requests, the onAccDataUpdate callback still reports accelerated traffic. What is the reason for this?**

Answer: Processes such as internal SDK handshake negotiation and protocol probing generate traffic. This traffic is minimal and negligible, and it is not included in acceleration billing.

API Overview

Last updated：2026-05-21 11:44:07

MpAccClient

Main entry point for SDK features:

API	Description
getInstance	Get the MpAccClient singleton instance
setDataKey	Set Device secret key
setSign	Set application signature authentication
setRegionalAccess	Set whether to use region-based access point assignment
setConnectTimeout	Sets connection timeout.
setLogInfoCallback	Set the log information callback interface
setLogLevel	Set the log level
prepare	Obtain the VPN authentication Intent
updateTCPDirect	Dynamically update TCP direct-connection status
updateUDPDirect	Dynamically update UDP direct-connection status
startAcc	Start acceleration
stopAcc	Stop acceleration
startMeasure	Enable network measurement
stopMeasure	Stop network measurement
registerAccCallback	Register a callback to listen for acceleration status
unregisterAccCallback	Unregister the callback to listen for acceleration status
registerMeasureCallback	Register a callback to listen for network measurement results

unregisterMeasureCallback	Unregister the callback to listen for network measurement results
requestNetwork	Request network
releaseNetwork	Release network
register	Pre-register
getSdkVersion	Obtaining the SDK Version Number

getInstance

Retrieve the MpAccClient instance.

```
public static MpAccClient getInstance(Context context)
```

Parameter description:

Parameter	Type	Description
context	Context	ApplicationContext

setDataKey

Set the Device secret key for authentication of Self-owned device

```
public static void setDataKey(String uuid, String dataKey)
```

Parameter description:

Parameter	Type	Description
uuid	String	Unique device identifier (can reuse the deviceId returned by the cloud API)
dataKey	String	Device key (each dataKey corresponds to one device and cannot be reused)

setSign

Set application signature authentication for application-created devices.a

```
public static void setSign(String appId, String sign)
```

Parameter description:

Parameter	Type	Description
appId	String	Application appId
sign	String	Signature string

setRegionalAccess

Set whether to use region-based access point assignment.

```
public static void setRegionalAccess(boolean isRegionalAccess)
```

Parameter description:

Parameter	Type	Description
isRegionalAccess	boolean	Whether to use region-based access point assignment • true: Yes • false: No • Default: true

setConnectTimeout

Set the registration timeout duration.

```
public static void setConnectTimeout(long timeout)
```

Parameter description:

Parameter	Type	Description
timeout	long	Timeout (seconds)

setLogInfoCallback

Set log callback.

```
public static void setLogInfoCallback(LogInfoCallback logInfoCallback)
```

Parameter description:

Parameter	Type	Description
logInfoCallback	LogInfoCallback	log callback instance

setLogLevel

Set the log level. For the Release version, it is recommended to set the log level to **Info**.

```
public static void setLogLevel(int level, boolean enablePrint)
```

Parameter description:

Parameter	Type	Description
level	int	Log level (0–6) • 0:PANIC • 1:FATAL • 2:ERROR • 3:WARN • 4:Info • 5:DEBUG • 6:TRACE
enablePrint	boolean	Whether to print logs

prepare

Get the VPN authorization Intent, mainly used to initiate VPN authorization, which the user confirms in the pop-up dialog
Return value: VPN authentication Intent.

```
public static Intent prepare(Activity activity)
```

Parameter description:

Parameter	Type	Description

activity	Activity	Current activity

startAcc

Initiate acceleration.

```
public void startAcc(AccConfig accConfig)
```

Parameter description:

Parameter	Type	Description
accConfig	AccConfig	Acceleration configuration object

stopAcc

Stop acceleration.

```
public boolean stopAcc()
```

startMeasure

Enable network measurement.

> **Note:**
> This API is mainly used for network measurement and fallback-to-origin coordination with acceleration, so as to optimize acceleration performance. If you use VPN acceleration for whitelisted applications, you must also add your own application package name to the whitelist. Only after this method is called will probing on the acceleration link take effect.

```
public void startMeasure(MeasureConfig measureConfig)
```

Parameter description:

Parameter	Type	Description
measureConfig	MeasureConfig	Network measurement configuration object

stopMeasure

Disable network measurement.

```
public void stopMeasure()
```

registerAccCallback

Signup acceleration status callback.

```
public void registerAccCallback(AccCallback accCallback)
```

Parameter description:

Parameter	Type	Description
accCallback	AccCallback	Acceleration status callback object

unregisterAccCallback

Cancel listening to acceleration status.

```
public void unregisterAccCallback(AccCallback accCallback)
```

Parameter description:

Parameter	Type	Description
accCallback	AccCallback	Acceleration status callback object

registerMeasureCallback

Trigger callback listening for network measurement.

```
public void registerMeasureCallback(MeasureCallback measureCallback)
```

Parameter description:

Parameter	Type	Description
measureCallback	MeasureCallback	Network measurement callback object

unregisterMeasureCallback

Cancel callback listening for network measurement.

```
public void unregisterMeasureCallback(MeasureCallback measureCallback)
```

Parameter description:

Parameter	Type	Description
measureCallback	MeasureCallback	Network measurement callback object

requestNetwork

Request a network. After this API is called, the activated network interface will participate in acceleration.

```
public void requestNetwork(int type)
```

Parameter description:

Parameter	Type	Description
type	int	Network Type • Cellular • 1:Wi-Fi

releaseNetwork

Release a network. After this API is called, the activated network interface will no longer participate in acceleration.

```
public void releaseNetwork(int type)
```

Parameter description:

Parameter	Type	Description
type	int	Network Type • Cellular • 1:Wi-Fi

register

Pre-register. Calling this API in advance after initialization can reduce the time from starting acceleration to successful acceleration.

```
public void register(boolean isActivateCellular, int mode,
            RegisterCallback registerCallback)
```

Parameter description:

Parameter	Type	Description
isActivateCellular	boolean	Whether to activate cellular. Activating cellular in advance under weak Wi-Fi conditions can reduce acceleration time and improve connection establishment success rate, but it also increases power consumption on the terminal device.
mode	int	Acceleration mode
registerCallback	RegisterCallback	Pre-registration callback object

getSdkVersion

Get SDK version number.

```
public String getSdkVersion()
```

LogInfoCallback

Log callback object

```
void onLogInfoUpdate(String tag, String msg)
```

API	Description
tag	Tag

msg	Message

registerCallback

Preregister callback object.

API	Description
onRegisterSuccess	Registration succeeded
onRegisterFail	Registration failed

onRegisterSuccess

register successfully.

```
void onRegisterSuccess()
```

onRegisterFail

registration failed.

```
void onRegisterFail(int code)
```

Parameter description:

Parameter	Type	Description
code	int	Error code. Refer to the error code module.
mode	int	Mode 0: capturing all packets; 1: capturing specific IP address packets (not supported yet)

AccConfig

Acceleration configuration object.

API	Description
setBlackList	Sets the blocklist applications.
setWhiteList	Sets the allowlist applications.
setAccMode	Sets the acceleration mode.

addAccNode	Adds acceleration nodes.
setPingInterval	Sets the callback interval.
setRoute	Sets the routing table.
setAccLinks	Sets the acceleration links.
setPriority	Sets the link priority.
setTCPDirect	Sets tcp direct connection.
setUDPDirect	Sets UDP direct connection.
setAllowBypass	Sets the bound NIC direct connection.
setVpnKeepAlive	Sets whether to keep the acceleration alive.
addNetworkPlugin	Inserts a custom network interface card.
setInt	Sets the int-type configuration parameters.
setString	Sets the String-type configuration parameters.
setBoolean	Sets the boolean-type configuration parameters.
setArrayList	Sets the ArrayList configuration parameters.
setEnableSocks	Sets whether to enable the socks proxy.
setSocksPort	Sets the socks proxy port number.
setSocksUsername	Sets the socks username.
setSocksPassword	Sets the socks password.
setEncryption	Sets whether to enable encryption.
setCongestionMode	Sets the congestion algorithm.
setNetworkConditions	Sets the conditions for network acceleration.
setStartForeground	Whether to start the foreground service.

setBlackList

Set blocklist for apps without acceleration.

```
public AccConfig setBlackList(ArrayList<String> apps)
```

Parameter description:

Parameter	Type	Description
apps	ArrayList<String>	List of application package names that do not require acceleration

Sample:

```
ArrayList<String> blackList = new ArrayList<>();
blackList.add("com.example.app1");
accConfig.setBlackList(blackList);
```

setWhiteList

Set application allowlist and mode for acceleration.

```
public AccConfig setWhiteList(ArrayList<String> apps, int mode)
```

Parameter description:

Parameter	Type	Description
apps	ArrayList<String>	List of application package names that require acceleration
mode	int	Mode 0: capturing all packets; 1: capturing specific IP packets (not supported yet)

setAccMode

Set global acceleration mode.

```
public AccConfig setAccMode(int mode)
```

Parameter description:

Parameter	Type	Description

mode	int	Acceleration mode • **AccConfig.ACC_MOD_BONDING** Aggregation (applicable to video streaming scenarios) • **AccConfig.ACC_MOD_REDUNDANT** Redundancy (applicable to gaming scenarios) • **AccConfig.ACC_MOD_FASTSWITCHING** Fast switching (applicable to meeting and cloud gaming scenarios)

addAccNode

Add CDN acceleration node.

> **Note:**
> Public cloud gateway not required, automatic connection to the best node.

```
public AccConfig addAccNode(String ip, int port)
```

Parameter description:

Parameter	Type	Description
ip	String	Acceleration node IP address
port	int	Acceleration node port number

setPingInterval

Set the callback interval (reporting frequency of AccCallback#onAccDataUpdate).

```
public AccConfig setPingInterval(int interval)
```

Parameter description:

Parameter	Type	Description
interval	int	Interval, unit: seconds

setRoute

Set route table and perform acceleration for ALL IPs: setRoute("0.0.0.0",0).

```
public AccConfig setRoute(String address, int prefixLength)
```

Parameter description:

Parameter	Type	Description
address	String	IP address
prefixLength	int	Subnet mask network bit length

setAccLinks

Set accelerated link and select network interface cards to perform acceleration.

```
public AccConfig setAccLinks(int links)
```

Parameter description:

Parameter	Type	Description
links	int	Acceleration link information Cellular link: AccConfig.MOBILE_LINK Wi-Fi link: AccConfig.WIFI_LINK Ethernet link: AccConfig.ETH_LINK Cellular + Wi-Fi: The SDK uses AccConfig.WIFI_LINK|AccConfig.MOBILE_LINK by default.

setPriority

Set link priority.

```
public AccConfig setPriority(HashMap<Integer, Integer> priority
```

Parameter description:

Parameter	Type	Description
priority	HashMap	Priority map key: link type 0: mobile 1: Wi-Fi 9: eth value: a number ranging from 0–255, where a lower number indicates a higher priority. The

		default value is 64. When the priority number is >= 128, it indicates that the link is a backup link. priority.put(0,64) sets the priority of the link with type=0 (mobile) to 64.

setUDPDirect

Set UDP direct connection.

```
public AccConfig setUDPDirect(boolean udpDirect)
```

Parameter description:

Parameter	Type	Description
udpDirect	boolean	udp direct connection • true: Direct connection • false: No direct connection Enables direct connection for this interface. After acceleration, all udp traffic will be sent directly to the business server via the default route.

setTCPDirect

Set TCP direct connection.

```
public AccConfig setTCPDirect(boolean tcpDirect)
```

Parameter description:

Parameter	Type	Description
tcpDirect	boolean	tcp direct connection • true: Direct connection • false: No direct connection Enables direct connection for this interface. After acceleration, all tcp traffic will be sent directly to the business server via the default route.

setAllowBypass

Set bind ENI direct connection. VPN mode takes effect.

```
public AccConfig setAllowBypass(boolean allowBypass)
```

Parameter description:

Parameter	Type	Description
allowBypass	boolean	tcp direct connection • true: Direct connection • false: No direct connection. The default value is false. In VPN mode, enabling direct connection for this interface causes traffic from the specified network card to bypass VPN interception.

addNetworkPlugin

Insert custom network interface card, such as com.example.EthNetworkPlugin.

```
public void addNetworkPlugin(String pluginClassName)
```

Parameter description:

Parameter	Type	Description
pluginClassName	String	Full class name of the NIC plugin (Refer to FAQ Q4 for details).

setInt

Set int type configuration parameter.

```
public AccConfig setInt(String key, String value)
```

Parameter description:

Parameter	Type	Description
key	String	Configuration name, key value: • **AccConfig.KEY_MAX_RTT_DISABLE_AGGREGATION** In aggregation mode, links exceeding this value are not included in the aggregation. It is generally

		not recommended for users to modify this value. • **AccConfig.KEY_MAX_DELAY_UNTIL_FAILED** In real-time mode, this is the threshold for determining available links. Links exceeding this value are considered faulty and will not be used until they recover. It is generally not recommended for users to modify this value. • **AccConfig.KEY_MIN_SWITCH_RTT** In real-time mode, this is the switching sensitivity. It is generally not recommended for users to modify this value. • **AccConfig.KEY_MAX_RTT_THRESHOLD** In aggregation and real-time modes, this is the threshold for enabling link fault detection. Links exceeding this value undergo fault detection. It is generally not recommended for users to modify this value. • **AccConfig.KEY_MAX_RETRY_TIMES** Setting the number of Quic connection establishment retries, with a default of 3.
value	String	Value

setString

Set String type configuration parameter.

```
public AccConfig setString(String key, String value)
```

Parameter description:

Parameter	Type	Description
key	String	Configuration name, key value: • **AccConfig.KEY_SERVER_DOMAIN: Configure domain name** • **AccConfig.KEY_SECRET_KEY_ID: Secret key ID** • **AccConfig.KEY_CIPHER_TEXT: Signature string** • **AccConfig.KEY_SIGNATURE: Digital signature**

		• **AccConfig.KEY_T2_OUT: Specify T2 segment acceleration drop-off point**
		Currently supported egress points: **frankfurt siliconvalley saopaulo tokyo bangkok singapore hongkong**
		If Hong Kong (China) is specified as the egress point:
		accConfig.setBoolean(AccConfig.KEY_ENABLE_T2_ACC, true);
		accConfig.setString(AccConfig.KEY_T2_OUT, "hongkong");

value	String	Value

setBoolean

Set Boolean type configuration parameter.

```
public AccConfig setBoolean(String key, boolean value)
```

Parameter description:

Parameter	Type	Description
key	String	Value for the configuration name key: • **AccConfig.KEY_ENABLE_T2_ACC: Enable T2 segment acceleration** • **AccConfig.KEY_ALLOW_PRIVATE_NETWORK: Private network interconnection**
value	String	Value

setArrayList

Set ArrayList<String> type configuration parameter.

```
public AccConfig setArrayList(String key, ArrayList<String> value)
```

Parameter description:

Parameter	Type	Description
key	String	Value for the configuration name key:

		AccConfig.KEY_APP_ID_LIST AppId list. Pass in the appId of the corresponding application, and the SDK will accelerate the application's allowlist domains in the T2 segment.
value	ArrayList	Value

setEnableSocks

Whether to enable socks proxy (use in socks mode).

```
public AccConfig setEnableSocks(boolean enableSocks)
```

Parameter description:

Parameter	Type	Description
enableSocks	boolean	Whether to enable the socks proxy: • true: Enable • false: Disable Default: false

setSocksPort

Set socks proxy port number.

```
public AccConfig setSocksPort(int socksPort)
```

Parameter description:

Parameter	Type	Description
socksPort	int	Sets the socks proxy port number.

setSocksUsername

Set socks username.

```
public AccConfig setSocksUsername(String socksUsername)
```

Parameter description:

Parameter	Type	Description
socksUsername	String	socks username

setSocksPassword

Set socks password.

```
public AccConfig setSocksPassword(String socksPassword)
```

Parameter description:

Parameter	Type	Description
socksPassword	String	socks password

setEncryption

Set whether to encrypt the message.

```
public AccConfig setEncryption(boolean encryption)
```

Parameter description:

Parameter	Type	Description
encryption	boolean	Whether the packet is encrypted: • true: Encrypt • false: Do not encrypt

setCongestionMode

Set congestion control algorithm.

```
public AccConfig setCongestionMode(int congestionMode)
```

Parameter description:

Parameter	Type	Description
congestionMode	int	Congestion algorithm parameter: • **0: BBR**

		• 1: CUBIC • 2: BBR-NORMAL Default value: 0

setNetworkConditions

Set network speed up. If the current terminal does not meet the set acceleration conditions, it will return error code -22.

```
public AccConfig setNetworkConditions(int conditions)
```

Parameter description:

Parameter	Type	Description
conditions	int	• **AccConfig.MUST_WIFI_ACTIVE**: The current terminal must have Wi-Fi activated. • **AccConfig.MUST_SIM_CARD_READY**: The current terminal must have a SIM card inserted. • **AccConfig.MUST_MOBILE_DATA_ENABLE**: The current terminal must have mobile data enabled. If you need to add multiple conditions, set them as follows: **AccConfig.MUST_WIFI_ACTIVE|AccConfig.MUST_SIM_CARD_READY**

setStartForeground

Set to pull up and speed up the foreground service.

```
public AccConfig setStartForeground(boolean startForeground)
```

Parameter description:

Parameter	Type	Description
startForeground	boolean	• false: Do not pull up • true: Pull up Default: true

MeasureConfig

Speed test configuration object. By setting this parameter, you can achieve dynamic acceleration effect.

API	Description
setRTT	Sets the latency threshold.
setLoss	Sets the packet loss rate threshold.
setJitter	Sets the jitter threshold.
setAddr	Sets the speed test address.
setMode	Sets the mode.
setTime	Sets the sliding window measurement time.
setQuickTime	Sets the quick start window time.
setQuickRtt	Sets the quick start latency threshold.
setInterval	Sets the measurement callback frequency.
setPingTimeout	Sets the ping packet timeout.
setT2AccProbeAddr	Sets the T2 segment speed test IP address.
setDisableSlaveLossDetect	Disables slave path packet loss detection.
setAdditionalOptions	Sets the fallback origin custom options.
setStartForeground	Starts the foreground service.

setRTT

Set the latency threshold. When the actual network latency of the terminal in the sliding window is above the set latency threshold, the onStartMpAcc API will be triggered.

```
public MeasureConfig setRTT(int RTT)
```

Parameter description:

Parameter	Type	Description
RTT	int	Latency threshold (ms)

setLoss

Set the packet loss threshold. When the actual network packet loss of the terminal in the sliding window is above the set threshold, the onStartMpAcc API will be triggered.

```
public MeasureConfig setLoss(int loss)
```

Parameter description:

Parameter	Type	Description
loss	int	Packet loss percentage threshold (0–100)

setJitter

Set the jitter threshold. When the actual network jitter of the terminal in the sliding window is above the set threshold, the onStartMpAcc API will be triggered.

```
public MeasureConfig setJitter(int jitter)
```

Parameter description:

Parameter	Type	Description
jitter	int	Network jitter threshold (ms)

setMode

Set the mode and keep it consistent with the mode triggered for acceleration.

```
public MeasureConfig setJitter(int jitter)
```

Parameter description:

Parameter	Type	Description
mode	int	Modes are as follows: • Aggregation: AccConfig.ACC_MOD_BONDING • Dual-send: AccConfig.ACC_MOD_REDUNDANT • Fast switching: AccConfig.ACC_MOD_FASTSWITCHING

setTime

Set sliding window measurement time.

```
public MeasureConfig setTime(int time)
```

Parameter description:

Parameter	Type	Description
time	int	Measurement duration, in ms

setQuickRtt

Set the quick startup latency threshold. This API is mainly for scenarios with sudden latency spikes. When the actual network latency of the terminal within the quick startup window is above the set quick startup latency threshold, the onStartMpAcc API will be triggered.

```
public MeasureConfig setQuickRtt(int quickRtt)
```

Parameter description:

Parameter	Type	Description
quickRtt	int	Quick start latency threshold, unit: ms

setQuickTime

Set quick startup window measurement time.

```
public MeasureConfig setQuickTime(int quickTime)
```

Parameter description:

Parameter	Type	Description
quickTime	int	Quick start window time, unit: ms

setInterval

Set measurement callback frequency (onRttChanged reporting frequency).

```
public MeasureConfig setInterval(int interval)
```

Parameter description:

Parameter	Type	Description
interval	int	Callback frequency, in ms

setPingTimeout

Set ping packet timeout.

```
public MeasureConfig setPingTimeout(int pingTimeout)
```

Parameter description:

Parameter	Type	Description
pingTimeout	int	Timeout, unit: ms

setT2AccProbeAddr

Set T2 speed test IP. Setting this value will enable the SDK to start speed test for T2 drop-off points.

```
public MeasureConfig setT2AccProbeAddr(String addr)
```

Parameter description:

Parameter	Type	Description
addr	String	T2 speed test IP. Europe server: **MeasureConfig.FRANKFURT_ADDRESS** US server: **MeasureConfig.SILICON_VALLEY_ADDRESS** Brazil server: **MeasureConfig.SAO_PAULO_ADDRESS** Japan server: **MeasureConfig.TOKYO_ADDRESS** Thailand server: **MeasureConfig.BANGKOK_ADDRESS** Singapore server: **MeasureConfig.SINGAPORE_ADDRESS**

		Hong Kong (China): **MeasureConfig.HONG_KONG_ADDRESS**

setDisableSlaveLossDetect

Disable secondary path detection for fast switch mode. It is disabled by default. If 100% packet loss occurs on the secondary path within the measurement window, the onStopMpAcc callback will be triggered to remind you to disable acceleration.

```
public MeasureConfig setDisableSlaveLossDetect(boolean
disableSlaveLossDetect)
```

Parameter description:

Parameter	Type	Description
disableSlaveLossDetect	boolean	• true: Disable • false: Do not disable

setAdditionalOptions

Set custom options for emergency origin retrieval.

```
public MeasureConfig setAdditionalOptions(AdditionalOptions
additionalOptions)
```

Parameter description:

Parameter	Type	Description
additionalOptions	Additional Options	Fallback origin custom object

setStartForeground

Set to pull up the speed test foreground service.

```
public MeasureConfig setStartForeground(boolean startForeground)
```

Parameter description:

Parameter	Type	Description

startForeground	boolean	• false: Do not pull up • true: Pull up Default: true

AccCallback

Acceleration status callback object.

API	Description
onAccDataUpdate	Network speed callback
onSummaryInfoUpdate	Acceleration summary information update
onNetworkStateChanged	Network adapter link state change callback
onAccSuccess	Acceleration success callback
onAccFail	Acceleration failure callback

onAccDataUpdate

Network rate callback. The callback frequency is set by {@link AccConfig#setPingInterval(int)}.

```
void onAccDataUpdate(long tRx, long tTx, ArrayList<PathDetail>
pathDetails)
```

Parameter description:

Parameter	Type	Description
tRx	long	Total received data volume (bytes)
tTx	long	Total transmitted data volume (bytes)
pathDetails	ArrayList	Detailed data for each transmission path

onSummaryInfoUpdate

Speed up information update aggregation. Trigger every time acceleration is complete.

```
void onSummaryInfoUpdate(String summaryInfo)
```

Parameter description:

Parameter	Type	Description
summaryInfo	String	Acceleration summary information, which is JSON-formatted summary data containing key metrics such as network quality and acceleration effectiveness.

onNetworkStateChanged

Network interface status change callback.

```
void onNetworkStateChanged(int type, boolean available, String ip)
```

Parameter description:

Parameter	Type	Description
type	int	NIC type • 0:mobile • 1:Wi-Fi
available	boolean	Available or not
ip	String	NIC ip address (IPv4), left blank if unavailable.

onAccSuccess

Success callback.

```
void onAccSuccess(ArrayList<String> packageNames, String ip, int port)
```

Parameter description:

Parameter	Type	Description
packageNames	ArrayList	List of successfully accelerated application package names
ip	String	Acceleration server IP address
port	int	Acceleration server port

onAccFail

Failure callback.

```
void onAccFail(ArrayList<String> packageNames, int errorCode)
```

Parameter description:

Parameter	Type	Description
packageNames	ArrayList	List of application package names for which acceleration failed
errorCode	int	Error code (see the error code table for details)

Trigger timing: Triggered when the acceleration channel establishment fails.

MeasureCallback

Network measurement callback.

API	Description
onStartMpAcc	Enable acceleration (recommended)
onStopMpAcc	Disable acceleration (recommended)
onStartQos	Enable Qos (recommended)
onNoPolicy	No policy is available for the current network state.
onAccException	The acceleration link is abnormal.
onRttChanged	Callback for bypass udping rtt latency
onAccStateChanged	Acceleration state change callback
onDetectServerUpdate	Callback for speed test ip change

onStartMpAcc

It is advisable to initiate an acceleration callback, which will only be triggered when both Wi-Fi and cellular are online simultaneously.

```
void onStartMpAcc(int code)
```

Parameter description:

Parameter	Type	Description
code	int	trigger code • 1: Latency Exception Trigger • 2: Packet Loss Rate Exception Trigger • 3: Jitter exception trigger

onStopMpAcc

Disable acceleration callback.

```
void onStopMpAcc(int code)
```

Parameter description:

Parameter	Type	Description
code	int	trigger code • 1: Single NIC Trigger • 2: No Optimization Trigger • 3: No Network Trigger

onStartQos

Enable QoS, only be triggered in single-cell scenario.

```
void onStartQos(int code)
```

Parameter description:

Parameter	Type	Description
code	int	trigger code • 1: Latency Exception Trigger • 2: Packet Loss Rate Exception Trigger • 3: Jitter exception trigger

onNoPolicy

Current network status: no policy available.

```
void onNoPolicy(int code)
```

Parameter description:

Parameter	Type	Description
code	int	Trigger code • 2: Single Wi-Fi • 3: No network available • 4: Internal Error • 5: Speed test service abnormally disconnected • -12: Authentication timeout • -13: Authentication Exception • -19: Speed test service failure • -23: Account authentication failed.

onAccException

Accelerated link exception (in this case, the SDK will proactively close acceleration and speed test, switching back to normal mode).

```
void onAccException(int errorCode)
```

Parameter description:

Parameter	Type	Description
errorCode	int	Error Code • -6: acc speed test sustained packet loss exception. • -7: acc speed test maximum latency exception. • -8: acc speed test latency anomaly compared to default route. • -9: acc speed test average latency exceeds maximum threshold.

onRttChanged

Bypass udping rtt latency callback, min interval 500ms.

```
void onRttChanged(int type, int rtt)
```

Parameter description:

Parameter	Type	Description
type	int	Type Speed test from the terminal to the acceleration gateway. • **0:mobile** • **1:Wi-Fi** • **2:acc** Speed test from the terminal to the T2 offloading point. • **1001:default** • **1002:acc**
rtt	int	Latency (ms), default packet loss value: 60000.

onAccStateChanged

Acceleration status change callback.

```
void onAccStateChanged(int state, int code)
```

Parameter description:

Parameter	Type	Description
state	int	• 0: Disabled. • 1: Enabled.
code	int	• 0: Enabled • 1: Disabled • 2: Abnormal Disable • 3: Disable the onRevoke callback.

onDetectServerUpdate

Speed test ip change callback.

```
void onDetectServerUpdate(int type, String ip, int port)
```

Parameter description:

Parameter	Type	Description
type	int	Type (0: mobile, 1: Wi-Fi, 2: acc)
ip	String	Speed test IP address
port	int	Speed test port

Error Code

Error Code	Description
-1	VPN fd null pointer exception
-2	MPQUIC exception
-3	Accelerated connection establishment exception
-4	Invalid package name or route address
-5	Acceleration service abnormal disconnection
-6	acc speed test packet loss rate abnormal
-7	acc acceleration maximum latency abnormal
-8	acc speed test latency abnormal compared to Wi-Fi
-9	acc speed test average latency exceeds maximum threshold
-11	Acceleration authentication failure
-12	Authentication connection timeout
-13	Authentication connection exception
-17	Acceleration parameters invalid
-18	Acceleration service startup failure
-19	Speed test service startup failure
-21	Account authentication failure

–22	Network adapter status does not meet acceleration requirements.
–23	Invalid speed test address
–24	Invalid custom network interface card plugin
–25	Invalid acceleration plugin
–26	No active network interface card on the current device.
–28	Terminal UDP ping failure
–29	Acceleration not supported for this device.
–30	Speed test parameters invalid.
–31	Acceleration has been canceled.
–32	HarmonyOS cellular activation restriction is abnormal. It is recommended to restart the machine to recover.
100	Multi–channel acceleration has been started.
101	Multi–channel acceleration internal error
102	Acceleration proxy enabling failure.
103	Invalid acceleration node IP address
104	Invalid acceleration mode
105	Acceleration token authentication failure.
106	The acceleration link notification is disabled. (Check whether multiple devices are reusing the dataKey.)
107	Channel handshake establishment timeout. (Check whether the dataKey is passed in public cloud mode.)
108	Acceleration reconnection timeout
109	SOCKS5 proxy failure
110	Maximum concurrent connections to the current gateway reached.

Advanced Features
TRTC Plug-In Integration

Last updated：2026-05-21 11:45:04

Dependency Configuration

```
//Introduce the new version SDK of TRTC
implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'
//Multiple Network Acceleration SDK
implementation 'com.tencent.linkboost:mpacc:2.9.4'
//TRTC acceleration plug-in
implementation 'com.tencent.linkboost:trtc-acc-plugin:1.0.2'
```

Sample Code

- Enable/disable acceleration services after TRTC room entry and exit.

```
//Upon entering the room successfully, enable acceleration
protected class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onEnterRoom(long result) {
        //trigger acceleration
        MpAccManager.getSingleInstance(context).startMpAcc();
        ...
     }
}

private void exitRoom() {
    //disable acceleration when leaving the room
    MpAccManager.getSingleInstance(context).stopMpAcc();
    ...
}
```

- Use the TRTC proxy plugin to perform acceleration.

```
//Multiple Network Acceleration management class
```

```
public class MpAccManager {

private MpAccManager(Context context) {
      initMpAcc(context);
 }

private void initMpAcc(Context context) {
    // The datakey applied in the Tencent Cloud console
    MpAccClient.setDataKey("test-123456", "*");
    mpAccClient = MpAccClient.getInstance(context);
    //Get the plugin instance
     TRTCAccPlugin accProxy = new TRTCAccPlugin(context);
    //Add the TRTC acceleration plug-in
    AccPluginManager.getInstance().setAccProxyPlugin(accProxy);
}

 //Start acceleration
public void startMpAcc() {
    //Refer to the AccConfig class description for updating
initialization parameters
    AccConfig accConfig = new AccConfig();
    accConfig.setAccMode(3)  //1: aggregation acceleration  2: dual
acceleration  3: fast switch acceleration
             .setPingInterval(3)
             .setEnableSocks(true)
    try {
        //Register acceleration result callback
        mpAccClient.registerAccCallback(accCallback);
        mpAccClient.startAcc(accConfig);
    } catch (MpAccSDKException e) {
        e.printStackTrace();
    }
}

//stop acceleration
public void stopMpAcc() {
    try {
        mpAccClient.unregisterAccCallback(accCallback);
        mpAccClient.stopAcc();
    } catch (MpAccSDKException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Dynamic Acceleration

Last updated：2026-05-21 11:46:31

The SDK provides a range of network measurement APIs and callback APIs. Businesses can set measurement thresholds to achieve acceleration as needed. Meanwhile, the SDK also offers a fallback strategy. If significant anomalies occur during acceleration, such as network disconnection, excessive latency, negative optimization, or acceleration performance far below expectations, the SDK will proactively disable acceleration and speed tests, directing traffic to the origin server to prevent abnormal acceleration from impacting normal business operation.

> **Note:**
> When the SDK triggers the escape onAccException callback, it will directly terminate the acceleration channel. If the acceleration was initiated using a SOCKS5 proxy, the business must switch the appropriate network requests back to direct connection (no longer using a proxy).

Access Process

1. **Initialization and Sign-up Phase**
   - SDK Initialization: First, call setDataKey to complete the basic configuration.
   - Interface Registration: Initiate registration with the server via the register API.
   - Registration failed: The system directly enters the "End" process.
   - Registration successful: The system proceeds to the next phase via the onRegisterSuccess callback.
2. **Speed Test and Evaluation Phase**
   - Initiate speed test: After registering the listener event, set thresholds for RTT (latency), packet loss rate, jitter, and other metrics, and then start evaluating the current network quality via startMeasure.
   - Trigger acceleration: The onStartMpAcc callback is triggered when the network environment meets the preset acceleration conditions.
3. **Acceleration Configuration and Execution Phase**
   - Initiate acceleration: Configure the acceleration mode, application allowlist, and routing table. Then, the business side executes startAcc.
   - Secondary verification: The system performs a final verification of permissions, network conditions, and device compatibility.
   - Verification failed: The system returns the acceleration failure onAccFail.
   - Verification passed: The system establishes a QUIC channel, implements traffic diversion and acceleration, and returns the onAccSuccess callback.
4. **Effect Monitoring and Dynamic Adjustment Phase**

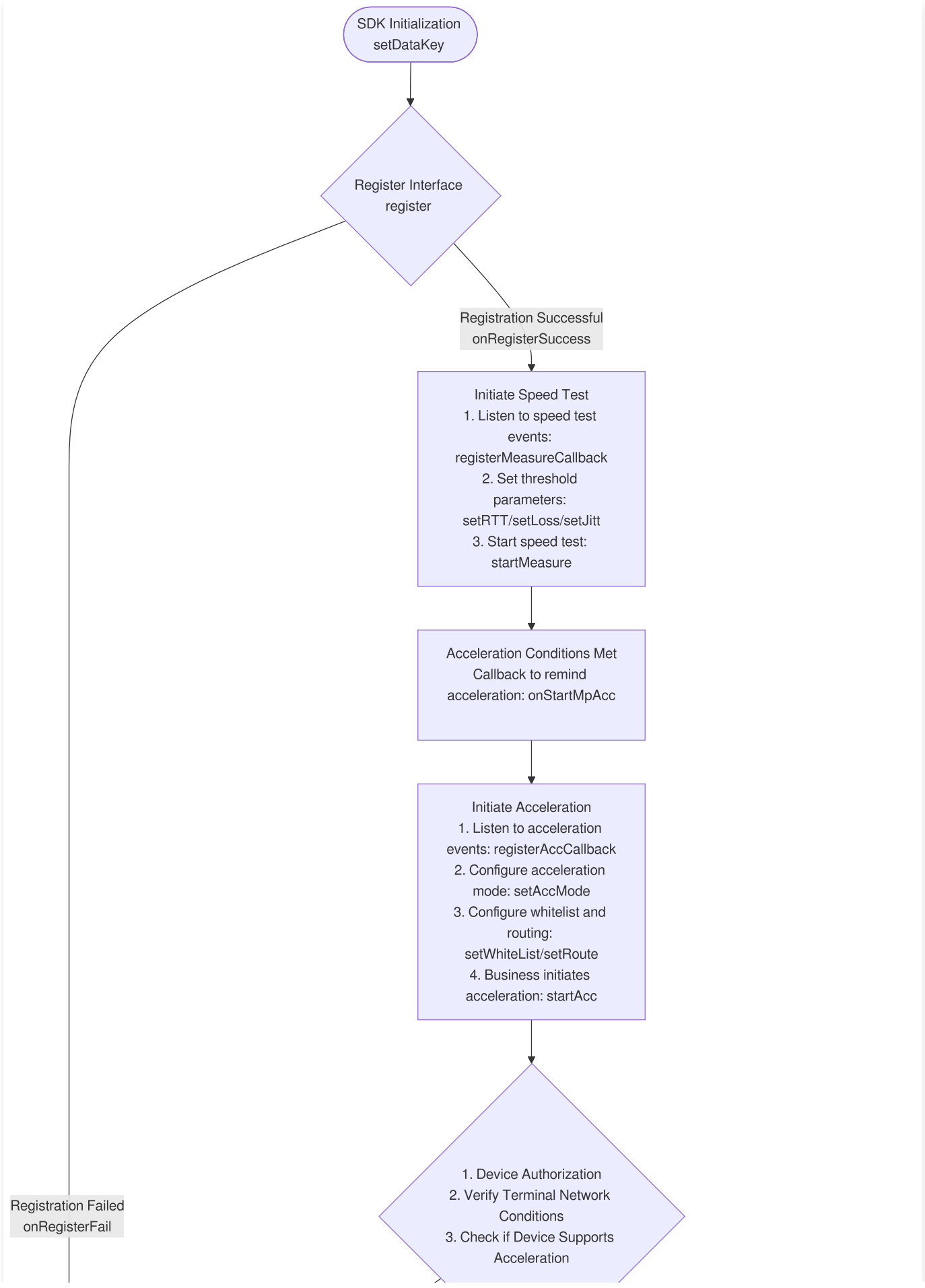
   After acceleration is enabled, the SDK continuously monitors network performance in real time:

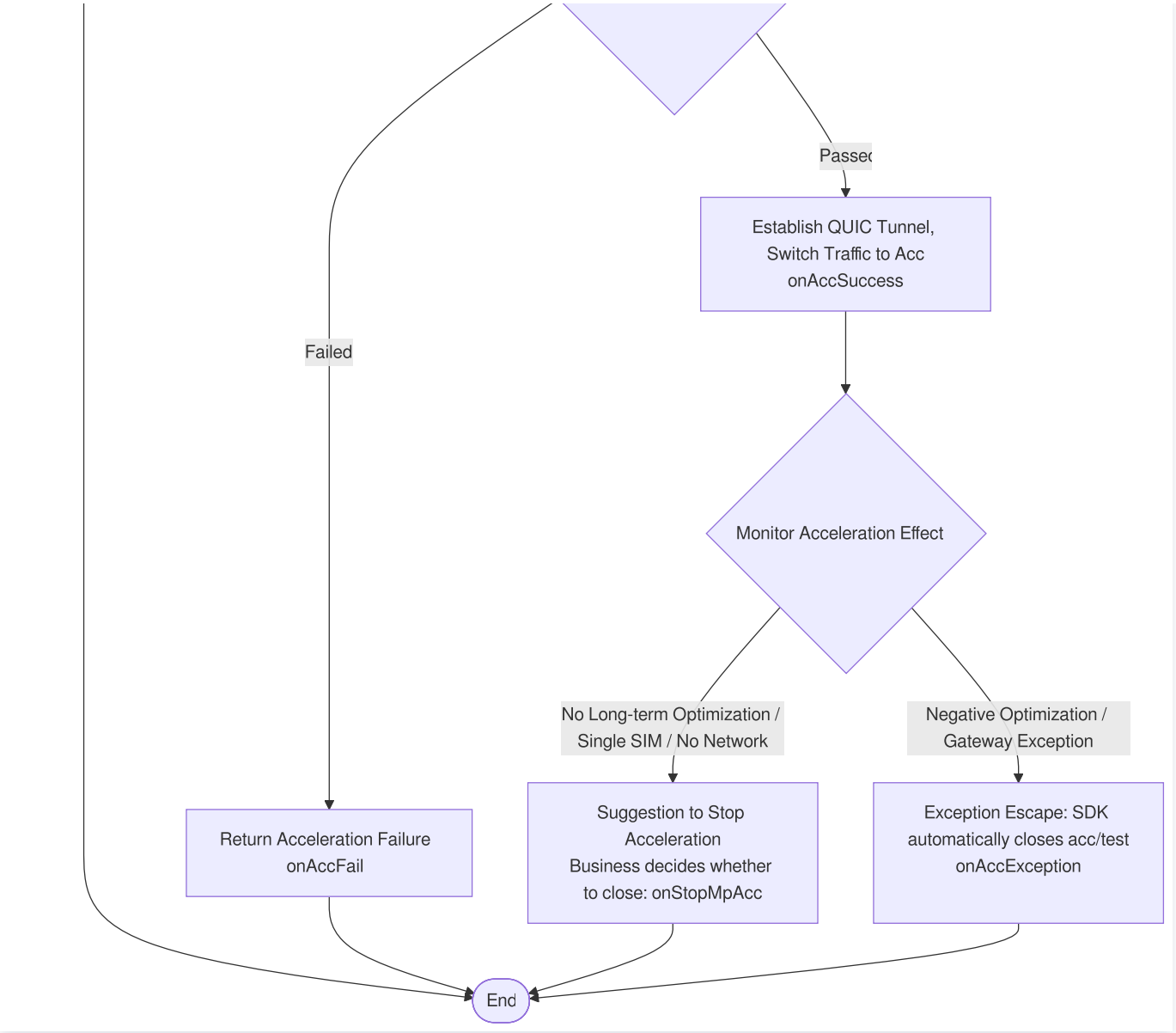
- Normal operation: Acceleration continues.
- Poor performance: If conditions such as prolonged lack of optimization, single NIC, or no network persist, it is recommended to stop acceleration (the business side decides whether to call onStopMpAcc).

5. **Exception Handling and Exit Phase**

- Exception escape: If "negative optimization" or a gateway exception occurs, the SDK triggers the onAccException callback and proactively disables acceleration and speed testing.
- End: All paths ultimately lead to the end of the process.

Flowchart





Example Code

```
private MeasureCallback measureCallback = new MeasureCallbackImpl();

 private void initMpAcc() {
        // The datakey applied from Tencent Cloud needs to be imported
        MpAccClient.setDataKey("test-123456", "*");
        mpAccClient = MpAccClient.getInstance(this);
}

private void startMeasure() throws MpAccSDKException {
       //Trigger callback listening
        mpAccClient.registerMeasureCallback(measureCallback);
        MeasureConfig measureConfig = new MeasureConfig();
```

```
        //Set measurement configuration threshold
        measureConfig.setJitter(15)
                .setLoss(5)
                .setRTT(60)
                .setTime(8000)
                .setQuickTime(2000)
                .setQuickRtt(80)
                .setInterval(1000)
                .setMode(2);
        //Start network measurement
        mpAccClient.startMeasure(measureConfig);
}

private void stopMeasure() throws MpAccSDKException {
    //Unregister network measurement listener
    mpAccClient.unregisterMeasureCallback(measureCallback);
    //Stop network measurement
    mpAccClient.stopMeasure();
}

class MeasureCallbackImpl implements MeasureCallback {
        @Override
        public void onStartMpAcc(int code, int links) {
          //Trigger acceleration, see example code for each mode
          startAcc();
        }

        @Override
        public void onStopMpAcc(int code) {
          //Disable acceleration, see example code for disabling
acceleration
          stopAcc();
        }

        @Override
        public void onNoPolicy(int code) {
         //No valid acceleration strategy currently
        }

        @Override
```

```
        public void onAccException(int code) {
         //Acceleration anomaly, SDK triggers fallback
        }
}
```

Plug-In Network Interface Card Acceleration

Last updated：2026-05-21 11:47:04

Currently, Multiple Network Acceleration Android SDK supports acceleration for three types of network interface cards: WIFI, MOBILE, and ETH. You can add the network interface card you want to accelerate by calling the **setAccLinks** method. If you want to add an unsupported network interface card type, you can dynamically add a custom network interface card by using the **addNetworkPlugin** function to add a network interface card plug-in.

Example code:

```
package com.android.tencentvpn.plugin;

import com.android.linkboost.multi.AccConfig;
import com.android.linkboost.multi.MpNetworkPlugin;
import com.android.linkboost.multi.MpNetworkPluginManager;
import com.android.linkboost.multi.log.MpAccLog;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.LinkProperties;
import android.net.Network;
import android.net.NetworkCapabilities;
import android.net.NetworkRequest;
import androidx.annotation.NonNull;

/**
 * Network interface card plugin implementation class
 * When acceleration needs to be triggered, add via {@link
AccConfig#addNetworkPlugin(String)}

accConfig.addNetworkPlugin("com.android.tencentvpn.plugin.MpNetworkPlugi
nImpl");
 */
public class MpNetworkPluginImpl extends MpNetworkPlugin {
    private static final String TAG = "MpNetworkPluginImpl";
    private static final String ETH0 = "eth0";
```

```
    private final ConnectivityManager mConnectivityManager;
    private final MpNetworkPluginManager mMpNetworkPluginManager;
    private final InnerCallback mInnerCallback;
    private Network mNetwork;

    public MpNetworkPluginImpl(Context context) {
        super(context);
        mConnectivityManager =
                (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        mMpNetworkPluginManager = new MpNetworkPluginManager(context);
        mInnerCallback = new InnerCallback();
    }

    //Business needs to implement the specific activation logic of the
network interface card itself
    @Override public void requestNetwork() {
        MpAccLog.i(TAG, "requestNetwork");
        NetworkRequest.Builder req = new NetworkRequest.Builder();
        req.addCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET);
        req.addTransportType(NetworkCapabilities.TRANSPORT_CELLULAR);
        mConnectivityManager.requestNetwork(req.build(),
mInnerCallback);
    }

    @Override public void releaseNetwork() {
        MpAccLog.i(TAG, "releaseNetwork");
        mConnectivityManager.unregisterNetworkCallback(mInnerCallback);
        mMpNetworkPluginManager.onLost(mNetwork);
    }

    class InnerCallback extends ConnectivityManager.NetworkCallback {
        @Override public void onLinkPropertiesChanged(@NonNull Network
network,
                @NonNull LinkProperties linkProperties) {
            super.onLinkPropertiesChanged(network, linkProperties);
            int type = 101;
            // Different type values are passed for different network
interface card names in linkProperties.getInterfaceName()
            // if (ETH0.equals(linkProperties.getInterfaceName())) {
```

```
            // type = 101;// type() Custom type greater than 100 and
less than 1000 (100-1000), avoid conflict with other system network
interface cards
            // }
            // if (ETH1.equals(linkProperties.getInterfaceName())) {
            // type = 102;
            // }
            mNetwork = network;
            MpAccLog.i(TAG, "onLinkPropertiesChanged:"
                    + network
                    + " ifname:"
                    + linkProperties.getInterfaceName());
            mMpNetworkPluginManager.onLinkPropertiesChanged(type,
network, linkProperties);
        }

        @Override public void onLost(@NonNull Network network) {
            super.onLost(network);
            MpAccLog.i(TAG, "onLost:" + network);
            mMpNetworkPluginManager.onLost(network);
        }
    }
}
```

Note:
Trigger acceleration by adding the plug-in via addNetworkPlugin.

```
accConfig.addNetworkPlugin(MpNetworkPluginImpl.class.getName());
//...acceleration mode, routing rule settings
mMpAccClient.startAcc(accConfig);
```

Log Plugin

Last updated：2026-05-21 11:48:23

The SDK provides the ability for log reporting. If your application does not have a log reporting function, you can add a log reporting plug-in.

> **Note:**
> After integrating the log plugin, inform the Tencent Side for allowlisting.

Project Configuration

```
implementation 'com.tencent.linkboost:log-plugin:1.1.0'
```

Example Code

```
LogPlugin logPlugin = new TXLogPlugin(this);
//Device connectivity can also be authenticated by calling setDataKey
logPlugin.setSign(appId, deviceName, sign);
AccPluginManager.getInstance().setLogPlugin(logPlugin);
```

Dynamic Acceleration SO

Last updated：2026-05-21 11:48:50

Project Configuration

1. Exclude the cloud gathering so in build.gradle.
2. Introduce dynamic download so plug-in.

```
android {
      packagingOptions {
        //Exclude Multiple Network Acceleration so
        exclude 'lib/arm64-v8a/libgojni.so'
        exclude 'lib/armeabi-v7a/libgojni.so'
      }
}
dependencies {
  implementation 'com.tencent.linkboost:mpacc:2.9.3'
  //Introduce dynamic download so plug-in
  implementation 'com.tencent.linkboost:dynamic-so-plugin:1.0.3'
}
```

Example Code

```
DynamicSoPlugin dynamicSoPlugin = new TXDynamicSoPlugin(this);
//Dynamic loading so plugin initialization
dynamicSoPlugin.init(new DynamicSoPlugin.LoadListener() {
    //Dynamic loading so successfully
    @Override public void success() {
      mpAccClient = MpAccClient.getInstance(MpAccActivity.this);
      //Subsequently, normal acceleration can proceed
      ....
     }
     //Dynamic loading so failed
    @Override public void failure(String msg) {
        Log.e(TAG, "DynamicSo failure:" + msg);
     }
  });
//Add the plugin instance to AccPluginManager
```

```
AccPluginManager.getInstance().setDynamicSoPlugin(dynamicSoPlugin);
```

iOS SDK
Quick Connection

Last updated：2026-05-21 16:44:41

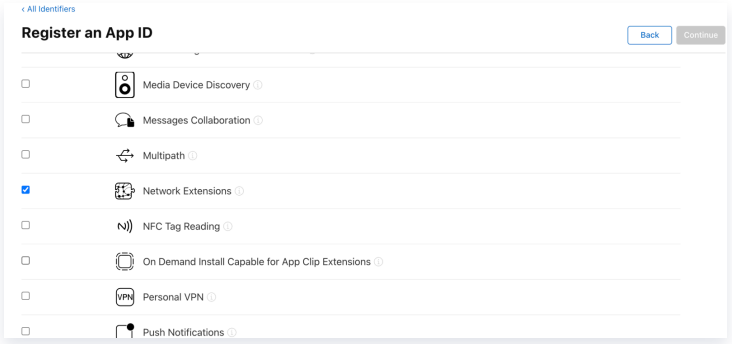
Configuration Instructions

1. VPN Access Configuration

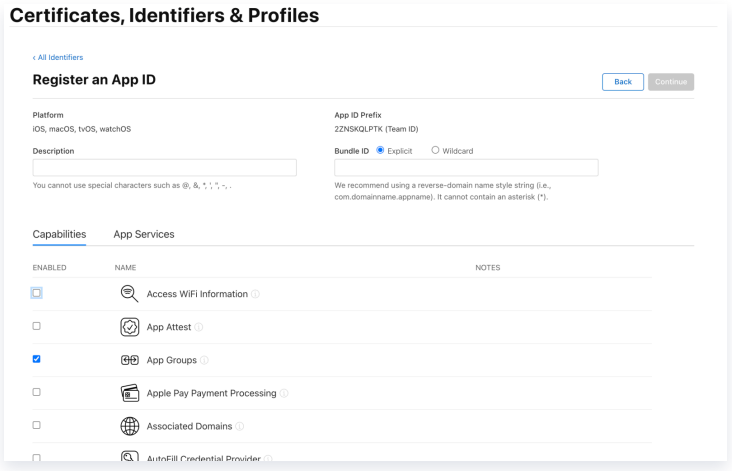
1.1 Developer Certificate Configuration

Adding Network Extensions and AppGroup configuration is required.

- Add Network Extensions to the App ID configuration.

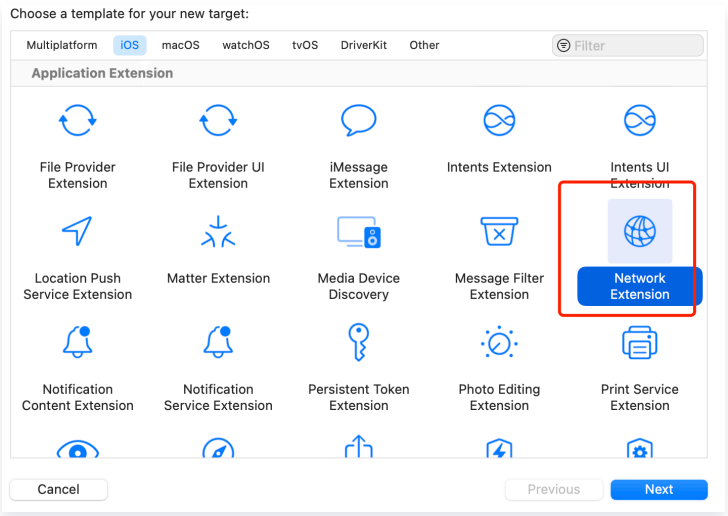


- Create an AppGroup (AppGroup is used for communication between the Extension process and the main process. If one exists in the project, it can be reused).

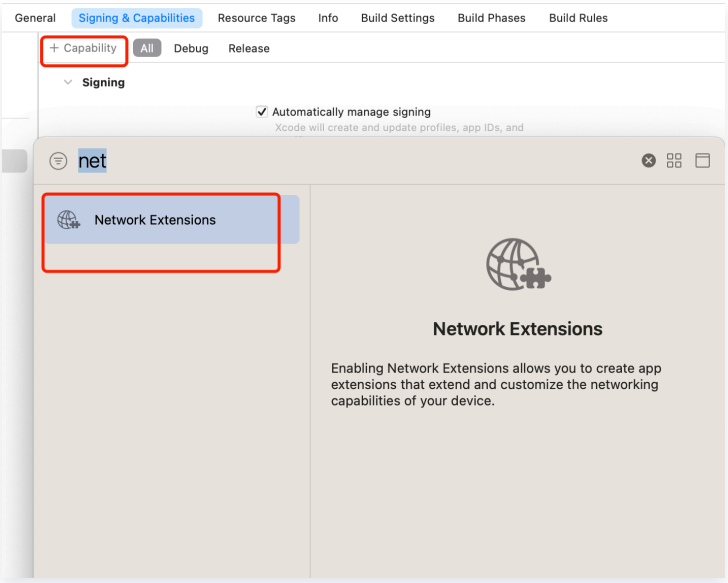


1.2 Xcode Configuration

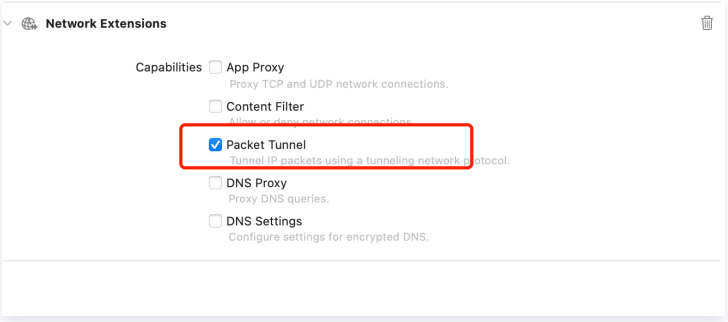
- Create a NetworkExtension target.



- Add **Capability** to both **main target** and **NetworkExtension target**, then select **Network Extension**.



- Then, check **Packet Tunnel**.



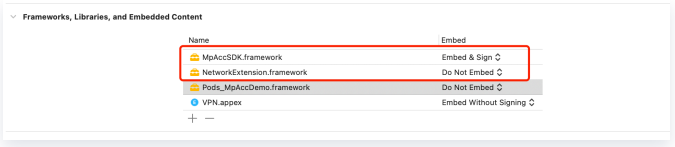
1.3 framework Reference Configuration

- Manually import xcframework

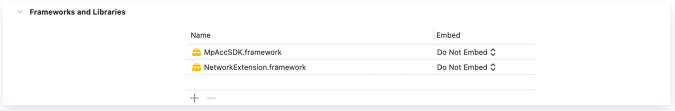
  MpAccSDK.framework: Integrate with the main target and Extension target. **Select "Embed & Sign" for the main target.**

NetworkExtension.framework: A system framework integrated with the main target and Extension target.

- Configuration of the main target as follows:



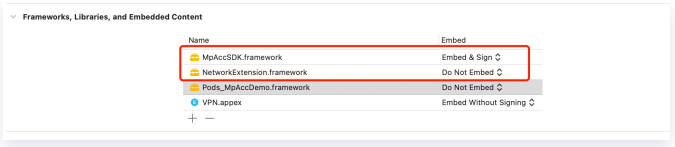
- Configuration of the Extension target as follows:



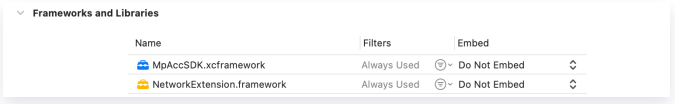
- Integrate xcframework via pod
  - Podfile file:

```
target 'XXX' do
pod 'MpAccSDK', :podspec => 'https://mpspeedr-online-1258344699.cos.ap-guangzhou.myqcloud.com/go/ios/2.9.8.0/MpAccSDK.podspec'
end
```

  - Configuration of the main target as follows:



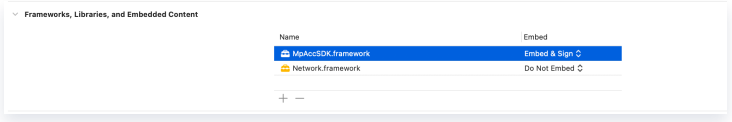
  - Configuration of the Extension target as follows:



2. 4 Adding the Following Environment Variables to the App'S Scheme

2.1 Importing an xcframework

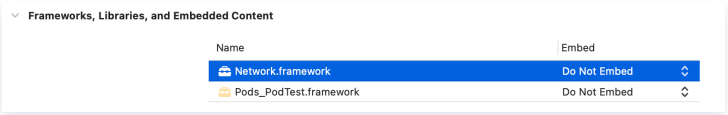
- Manually import xcframework: Integrate the following two frameworks in the App target with configuration as follows:



- Integrate xcframework via pod

```
target 'XXX' do
pod 'MpAccSDK', :podspec => 'https://mpspeedr-online-
1258344699.cos.ap-
guangzhou.myqcloud.com/go/ios/2.9.8.0/MpAccSDK.podspec'
end
```

Integrate only Network.framework in the App target with the configuration as follows:



2.2 Adding Privacy – Local Network Usage Description

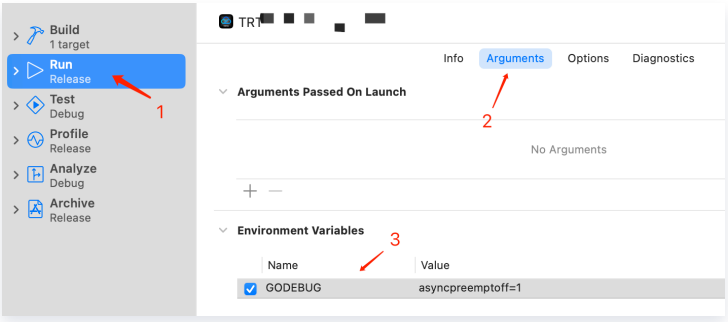
For SOCKS5 proxy, the App needs to perform data read/write on the local port, which can cause some system models to pop up a local network ask pop-up. It is necessary to configure this permission.

2.3 Adding the Following Environment Variables to the App'S Scheme

Name: GODEBUG

Value: asyncpreemptoff=1

> **Note:**
> This variable is valid only when DEBUG. The operation mechanism of the network library has conflicts with the Xcode DEBUG signal (SIGURG), leading to deteriorating performance in Xcode debugging. You can use the environment variable above to disable the response to this signal. Archive packages are unaffected.



Example Code

> **Note:**
> During initialization, SDK authentication can also be performed via application-based integration by calling the API MpAccClient.setSign("appid","*****"). Only one of device-based integration or

application-based integration needs to be configured.
For the sign generation method, see Console Getting Started Guide in the console.

1. VPN Acceleration Example

1.1 App-Side Startup Code

Swift Code

```
class VPNViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // If the SDK is in single vpn mode, you do not need to
configure this parameter.
        MpAccClient.shared.accType = .vpn
        // You only need to configure either device-based integration or
application-based integration.
        // Device-based integration
        MpAccClient.shared.setupDatakey(datakey: "xxxxx",
                                        deviceId: "xxxxx")
        // Application-based integration
        // MpAccClient.shared.setSign(appId:"xxx", sign:"xxx")
        // Register callback
        MpAccClient.shared.registerAccCallback(self)
        // Attempt to restore VPN
        MpAccClient.shared.resumeVpn()
    }
    //Start acceleration
    func startAcc() {
        let config = AccConfig()
        config.accMode = .Redundant // Acceleration mode 1: Aggregation
2: Dual-send 3: rtc
        config.pingInterval = 2
        config.groupId = "com.xxxx" // AppGroup
        config.vpnName = "XXXXXX" // VPN name
        // Start VPN
        MpAccClient.shared.start(config: config)
    }
    //Stop acceleration
    func stopAcc() {
```

```
            MpAccClient.shared.stop()
        }
    }

    extension VPNViewController: AccCallback {
        func onAccSuccess(ip: String, port: Int) -> Void { }
        func onAccFail(_ error: NSError?) -> Void { }
        func onAccDataUpdate(tRx: Int64, tTx: Int64, pathDetails:
    [MpPathDetail]) -> Void { }
        func onSummaryInfoUpdate(_ summaryInfo: String) -> Void { }
        func onNetworkStateChanged(_ type: MpInterfaceType, available: Bool,
    ip: String) -> Void { }
    }
```

OC Code

```
    @interface ViewController () <AccCallback>
    @end

    @implementation ViewController

    - (void)viewDidLoad {
        [super viewDidLoad];
        // If it is a single vpn mode SDK, this parameter does not need to
    be configured
        MpAccClient.shared.accType = AccTypeVpn;
        // Do any additional setup after loading the view.
        [MpAccClient.shared registerAccCallback:self];
        // You only need to configure either device-based integration or
    application-based integration.
        // Device-based integration
        [MpAccClient.shared setupDatakey:@"xxxxxxx"
                                deviceId:@"xxxxxxx"];

        // Application-based integration
        // MpAccClient.shared.setSign(appId:"xxx", sign:"xxx")

        [MpAccClient.shared resumeVpn];
    }
```

```
- (void)startVpn {
    AccConfig *config = [AccConfig new];
    /// Acceleration mode 1: Aggregation 2: Dual-send 3: rtc
    config.accMode = AccModeRedundant;
    config.pingInterval = 2;
    config.congrestionMode = MpCongrestionModeBBR;
    config.groupId = @"group.xxxxxxxx";
    config.vpnName = @"Test_name3";
    config.whiteList = @[];
    config.blackList = @[];
    [MpAccClient.shared startWithConfig:config];
}
- (void)stopAcc {
    [MpAccClient.shared stop];
}

// AccCallback
- (void)onAccSuccessWithIp:(NSString *)ip port:(NSInteger)port {
}
- (void)onAccFail:(NSError *)error {
}
- (void)onAccDataUpdateWithTRx:(int64_t)tRx tTx:(int64_t)tTx
pathDetails:(NSArray<MpPathDetail *> *)pathDetails {
}
- (void)onSummaryInfoUpdate:(NSString *)summaryInfo {
}
- (void)onNetworkStateChanged:(enum MpInterfaceType)type available:
(BOOL)available ip:(NSString *)ip {
}
@end
```

1.2 VPN Extension

Add the following code to the automatically generated PacketTunnelProvider in the Extension target of Xcode:

Swift Code

```
class PacketTunnelProvider: NEPacketTunnelProvider {
```

```
    var tunnelManager: MpPacketTunnelManager = MpPacketTunnelManager()

    override func startTunnel(options: [String : NSObject]?,
completionHandler: @escaping (Error?) -> Void) {
        //Call the startTunnel method of the SDK
        tunnelManager.startTunnel(packetTunnel: self, options: options,
completionHandler: completionHandler);
    }

    override func stopTunnel(with reason: NEProviderStopReason,
completionHandler: @escaping () -> Void) {
        //Call the stopTunnel method of the SDK
        tunnelManager.stopTunnel()
        completionHandler()
    }

    override func handleAppMessage(_ messageData: Data,
completionHandler: ((Data?) -> Void)?) {
        //Call the handleAppMessage method of the SDK
        tunnelManager.handleAppMessage(messageData, completionHandler:
completionHandler);
    }
}
```

OC Code

```
@interface PacketTunnelProvider() {
    MpPacketTunnelManager *tunnelManager;
}
@end
@implementation PacketTunnelProvider

- (id)init {
    self = [super init];
    if (self != NULL) {
        tunnelManager = [[MpPacketTunnelManager alloc] init];
    }
    return self;
}
```

```
- (void)startTunnelWithOptions:(NSDictionary *)options
completionHandler:(void (^)(NSError *))completionHandler {
    //Call the startTunnel method of the SDK
    [tunnelManager startTunnelWithPacketTunnel:self options:options
completionHandler:completionHandler];
}

- (void)stopTunnelWithReason:(NEProviderStopReason)reason
completionHandler:(void (^)(void))completionHandler {
    //Call the stopTunnel method of the SDK
    [tunnelManager stopTunnel];
    completionHandler();
}

- (void)handleAppMessage:(NSData *)messageData completionHandler:(void
(^)(NSData *))completionHandler {
    //Call the handleAppMessage method of the SDK
    [tunnelManager handleAppMessage:messageData
completionHandler:completionHandler];
}
@end
```

2. SOCKS5 Acceleration Example

> **Note:**
> After acceleration is successfully started in SOCKS mode (that is, after receiving the onAccSuccess callback), your service must send network requests through the SOCKS protocol.
> **The proxy IP address is the local IP, and the proxy port must be the same as the port set through config.socksProxyPort when starting acceleration.** Only then can the traffic enter the acceleration tunnel.

2.1 Swift Code (Using Alamofire as an Example)

```
import Alamofire
class Socks5ViewController: UIViewController {
    private var afs: Session = AF
    //Start acceleration
    func startAcc() {
```

```
        //Acceleration method. If it's a single SOCKS5_SDK, ignore it.
        MpAccClient.shared.accType = .socks5;
        let config = AccConfig()
        config.socksProxyPort = xxx
        // Start socks5
        MpAccClient.shared.start(config: config)
    }
}
extension Socks5ViewController: AccCallback {
    func onAccSuccess(ip: String, port: Int) -> Void {
        // Traffic switchover required
        let sessionConfig = URLSessionConfiguration.default
        sessionConfig.connectionProxyDictionary = [
            kCFStreamPropertySOCKSProxyHost : "127.0.0.1",
            // config.socksProxyPort
            kCFStreamPropertySOCKSProxyPort : "xxxxxx",
        ]
        afs = Session(configuration: sessionConfig)
    }
}
```

2.2 OC Code (Startup Only, Refer to Swift Code for HTTP Requests)

```
@interface ViewController () <AccCallback>
@end

@implementation ViewController
- (void)startVpn {
    // If it's a single SOCKS5_SDK, skip this field
    MpAccClient.shared.accType = AccTypeSocks5;
    AccConfig *config = [AccConfig new];
    // Set parameters
    [MpAccClient.shared startWithConfig:config];
}

// AccCallback
- (void)onAccSuccessWithIp:(NSString *)ip port:(NSInteger)port {
    // Traffic switchover required
    // AFNetworking settings are similar to Alamofire
```

```
}
@end
```

FAQ

1. **Swift Compilation Error**
   - Since the SDK is written in Swift, a Swift runtime environment is required. For Objective-C SDK integration, create a Swift empty file in the application project or CocoaPods repository.
   - SDK is compiled with Xcode 15. Using SDK with Xcode 14 will report Swift compatibility issues. Upgrade Xcode.
2. **Signature issue**

   Xcode does not sign nested SDKs by default. Therefore, do not nest MpAccSDK within your SDK. Alternatively, you can write a script to sign it.
3. **TRTC plug-in integration notes**
   - The TRTC xcframework reports an error. Upgrade CocoaPods, as versions prior to 12.1 contain a Bug.
   - Acceleration should be initiated after the room is entered. Otherwise, the acceleration component cannot obtain TRTC-related information.
4. **SOCKS5 mode acceleration, Xcode debugging lag issue**

   Reason: The operation mechanism of the network library has conflicts with the Xcode DEBUG signal (SIGURG), leading to deteriorating performance in Xcode debugging.
   Solution: Add the GODEBUG environment variable with the value asyncpreemptoff=1. You can use this environment variable to disable the response to this signal. Archive packages remain unaffected.
   For specific setting methods, see configuration instructions and SOCKS5 access configuration.
5. **SOCKS5 mode Privacy – Local Network Usage Description pop-up issue**

   Reason: During acceleration, the App needs to perform data read/write on the local port network, which may cause pop-up queries on some system models.
   Solution: Need to configure Privacy – Local Network Usage Description permission description.
6. **VPN Mode APP Store Rejection Issue**

   When submitting an APP for store listing, you must also upload the VPN qualification document. Otherwise, the submission will be rejected.

API Overview

Last updated：2026-05-21 16:50:39

MpAccClient

MpAccClient Properties

Required	Description
shared	The MpAccClient singleton
accType	Acceleration mode: `.vpn / .socks5 / .unowned`
isStart	Whether acceleration is active, which is true when status == .started.
status	Acceleration status: `.initialize / .launching / .launched / .started`
accConfig	The Config used by the current acceleration

MpAccClient Methods

Methodology	Description
register(mode:completion:)	Registration (accelerates the start process)
start(config:)	Start acceleration
stop	Stop acceleration.
getVpnStatus	Obtains the VPN status asynchronously.
resumeVpn	Call this API after the APP restarts to resume VPN event listening.
updateUDPDirect	Update UDP direct-connection status.
updateTCPDirect	Update TCP direct-connection status.
setupLogDelegate	Set up a log delegate to take over log processing.
setupLogLevel	Set the log level.

setupCallbackQueue	Sets the callback queue, which defaults to a sub-thread.
registerAccCallback	Register acceleration status listener.
unRegisterAccCallback	Unregister acceleration status listener.
setupDatakey	Set public cloud registration parameters.
setSign	Set signature and App ID.
setGatewayId	Set gateway ID.
enableOverseas	**Switches to the international site. The default is false (domestic site).**
setupRegisterConnectTimeout	Sets the registration timeout duration. The default is 10s.
startMeasure	Enable network measurement.
stopMeasure	Stop network measurement.
addEventInfo	Add custom reporting event.
registerMeasureCallback	Register network measurement callback listener.
unRegisterMeasureCallback	Unregister the callback to listen for network measurement results.
getSdkVersion	Obtaining the SDK Version Number

register

Pre-registration (to improve speed at start)

```
func register(mode: AccMode, completion: ((_ error: NSError?) -> Void)?)
```

Parameter	Description
mode	Acceleration mode: enum AccMode
completion	Pre-registration result

start(config:)

Initiate acceleration.

```
func start(config: AccConfig)
```

Parameter	Description
config	Acceleration parameter configuration. For details, see AccConfig.

stop

Stop acceleration.

```
func stop()
```

getVpnStatus

Get VPN status, return asynchronously.

```
func getVpnStatus(_ completion: @escaping (_ status: NEVPNStatus) ->
Void)
```

Parameter	Description
completion	Asynchronous callback status. The reference values are as follows: • Not configured: NEVPNStatus.invalid • Connected: NEVPNStatus.connected • Connecting: NEVPNStatus.connecting • Disconnected: NEVPNStatus.disconnected • Disconnecting: NEVPNStatus.disconnecting • Reasserting: NEVPNStatus.reasserting

resumeVpn

After restart, call this API to restore VPN event monitoring.

```
func resumeVpn()
```

updateUDPDirect

Update UDP direct connection status.

```
// asynchronous method
func updateUDPDirect(_ direct: Bool, completion: @escaping (_ result:
Bool) -> Void)
// synchronized method (cannot be called in the main thread in vpn mode)
func updateUDPDirect(_ direct: Bool) -> Bool
```

Parameter	Description
direct	Whether to connect directly. When set to false, the system queries the access gateway load status. This is a time-consuming operation and is recommended to be executed in a sub-thread.
completion	Completion callback. A result of true indicates a successful update.

updateTCPDirect

Update TCP direct connection status.

```
// async mode
func updateTCPDirect(_ direct: Bool, completion: @escaping (_ result:
Bool) -> Void)
// sync mode (cannot be called in the main thread in vpn mode)
func updateTCPDirect(_ direct: Bool) -> Bool
```

Parameter	Description
direct	Direct or Not
completion	Completion callback. A result of true indicates a successful update.

setupLogDelegate

Set log agent and take over log processing.

```
func setupLogDelegate(_ delegate: MpLoggerDelegate)
```

Parameter	Description

delegate	Log delegate

setupLogLevel

Set log level.

```
func setupLogLevel(_ logLevel: LogLevel)
```

Parameter	Description
logLevel	Log level enumeration

setupCallbackQueue

Set callback queue.

```
func setupCallbackQueue(_ queue: DispatchQueue)
```

Parameter	Description
queue	Callback queue, which defaults to the main queue.

registerAccCallback

Signup speed up listen for status.

```
func registerAccCallback(_ callback: AccCallback)
```

Parameter	Description
callback	Acceleration status callback interface

unRegisterAccCallback

Cancel speed up listen for status.

```
func unRegisterAccCallback(_ callback: AccCallback)
```

Parameter	Description
callback	Acceleration status callback interface

setupDataKey

Set public cloud registration parameters.

```
func setupDatakey(_ datakey: String, deviceId: String)
```

Parameter	Description
datakey	Device secret key
deviceId	Unique device identifier

setSign

Set application signature authentication.

```
func setSign(appId: String, sign: String)
```

Parameter	Description
appId	Application ID
sign	Signature

setGatewayId

Set Gateway ID.

```
func setGatewayId(_ gwId: String)
```

Parameter	Description
gwId	Gateway ID

enableOverseas

Switch to international site, default to false (domestic site).

```
func enableOverseas(enabled: Bool)
```

Parameter	Description
enabled	Whether to switch to the international site • false (domestic site) • true (international site)

setupRegisterConnectTimeout

Set registration timeout duration, default 10s.

```
func setupRegisterConnectTimeout(_ connectTimeout: Int)
```

Parameter	Description
connectTimeout	Timeout time

startMeasure

Enable network measurement.

```
func startMeasure(measureConfig: MpMeasureConfig)
```

Parameter	Description
measureConfig	Speed test configuration

stopMeasure

Terminate network measurement.

```
func stopMeasure()
```

addEventInfo

Add custom event reporting.

```
func addEventInfo(event: Int, msg: String)
```

Parameter	Description
event	Event ID
msg	Event Description

registerMeasureCallback

Register network measurement callback listener

```
func registerMeasureCallback(_ callback: MpMeasureCallback)
```

Parameter	Description
callback	Speed test callback interface

unRegisterMeasureCallback

Cancel network measurement callback listening.

```
func unRegisterMeasureCallback(_ callback: MpMeasureCallback)
```

Parameter	Description
callback	Speed test callback interface

getSdkVersion

Get SDK version number.

```
func getSdkVersion() -> String
```

MpPacketTunnelManager

VPN process API class.

Methodology	Description
startTunnel	Starts the VPN Tunnel and passes parameters to the SDK.
stopTunnel	Stops the VPN Tunnel.

handleAppMessage	Receives messages from the main program and passes parameters to the SDK.
getSdkVersion	Obtaining the version number.
setupLogDelegate	Setting up a log delegate.
registerMeasureCallback	Registering a network speed test callback listener.
unRegisterMeasureCallback	Unregistering the network speed test callback listener.

startTunnel

Enable VPN Tunnel, parameter passthrough to SDK.

```
public func startTunnel(packetTunnel: NEPacketTunnelProvider, options:
[String : NSObject]?, completionHandler: @escaping (Error?) -> Void)
```

Parameter	Description
packetTunnel	An instance object of the NEPacketTunnelProvider VPN Provider
options	Parameters passed through to the SDK, which include Config data.
completionHandler	Completion callback

stopTunnel

Disable VPN Tunnel.

```
func stopTunnel()
```

handleAppMessage

Receive main program messages, parameter passthrough to SDK

```
func handleAppMessage(_ messageData: Data, completionHandler: ((Data?) -
> Void)?)
```

Parameter	Description
messageData	Binary data, with parameters passed through to the SDK.
completionHandler	Completion callback, with parameters passed through to the SDK.

getSdkVersion

Get version number.

```
func getSdkVersion() -> String
```

setupLogDelegate

Set log agent.

```
func setupLogDelegate(_ delegate:MpLoggerDelegate?)
```

Parameter	Description
delegate	Log delegate instance

registerMeasureCallback

Register network speed test callback listener

```
func registerMeasureCallback(_ callback: MpMeasureCallback)
```

Parameter	Description
callback	Speed test callback interface

unRegisterMeasureCallback

Cancel network speed test callback listening.

```
func unRegisterMeasureCallback(_ callback: MpMeasureCallback)
```

Parameter	Description

callback	Speed test callback interface

AccCallback

Acceleration status and event callback.

Methodology	Description
onAccSuccess	Acceleration success callback
onAccFail	Acceleration startup failure
onAccDataUpdate	Network speed callback
onSummaryInfoUpdate	Acceleration summary information update
onNetworkStateChanged	Network adapter link state change callback

onAccSuccess

Success callback.

```
func onAccSuccess(ip: String, port: Int) -> Void
```

Parameter	Description
ip	Acceleration node IP address
port	Acceleration node port

onAccFail

Start acceleration failed.

```
func onAccFail(_ error: NSError) -> Void
```

Parameter	Description
error	Determines whether acceleration is terminated normally or abnormally based on the error code.

onAccDataUpdate

Network rate callback. The callback frequency is set by AccConfig.pingInterval.

```
func onAccDataUpdate(tRx: Int64, tTx: Int64, pathDetails:
[MpPathDetail]) -> Void
```

Parameter	Description
tRx	Total received data volume (unit: byte)
tTx	Total transmitted data volume (unit: byte)
pathDetails	Detailed link information

onSummaryInfoUpdate

Acceleration summary update.

```
func onSummaryInfoUpdate(_ summaryInfo: String) -> Void
```

Parameter	Description
summaryInfo	Acceleration summary information

onNetworkStateChanged

Network interface card link status change callback.

```
func onNetworkStateChanged(type: MpInterfaceType, available: Bool, ip:
String) -> Void
```

Parameter	Description
type	• 0:mobile • 1:Wi-Fi
available	NIC availability: • true: Available • false: Unavailable
ip	NIC IP (IPv4), left blank if unavailable.

MpMeasureCallback

Speed test callback.

Methodology	Description
onStartMpAcc	Enable acceleration (recommended)
onStopMpAcc	Disable acceleration (recommended)
onNoPolicy	No policy is available for the current network state.
onAccException	The acceleration link is abnormal. (In this case, the SDK automatically disables acceleration and switches back to normal mode.)
onRttChanged	Callback for bypass udping rtt latency
onAccStateChanged	Acceleration state change callback

onStartMpAcc

Enable acceleration.

```
func onStartMpAcc(code: Int) -> Void
```

Parameter	Description
code	trigger code • 0: Aggregation mode trigger (Aggregation mode is always triggered.) • 1: Latency exception trigger • 2: Packet loss rate exception trigger • 3: Jitter exception trigger

onStopMpAcc

Disable acceleration.

```
func onStopMpAcc(code: Int, msg: String) -> Void
```

Parameter	Description
code	Trigger code • 1: Single NIC trigger • 2: No optimization trigger

	• 3: No network trigger • 4: The acceleration NIC and the actual active NIC are inconsistent.
msg	Trigger message

onNoPolicy

Current network status: No policy available.

```
func onNoPolicy(code: Int, message: String?) -> Void
```

Parameter	Description
code	Trigger code • 2: Single Wi-Fi • 3: No network available • 4: Internal error • -12: Authentication timeout • -13: Authentication exception • -19: Speed test service failure • -23: Account authentication failed.
message	Trigger message. If the message is not empty, the message information takes precedence.

onAccException

The accelerated link encounters an exception (in this case, the SDK will proactively disable acceleration and switch back to normal mode).

```
func onAccException(errorCode: Int, msg: String) -> Void
```

Parameter	Description
errorCode	Error Code: • -6: acc speed test sustained packet loss anomaly • -7: acc speed test maximum number of delays anomaly • -8: acc speed test vs. Wi-Fi delay anomaly

	• -9: acc speed test average latency exceeds maximum threshold
msg	Error Message

onRttChanged

Bypass udping rtt delay callback.

```
func onRttChanged(type: Int, rtt: Int) -> Void
```

Parameter	Description
type	NIC type: • 0:mobile • 1:Wi-Fi • 2:acc
rtt	Latency (ms), packet loss value: 460.

onAccStateChanged

Aggregate acceleration status change callback.

```
func onAccStateChanged(state: Bool, code: Int) -> Void
```

Parameter	Description
state	Acceleration status • false: Disable • true: Enable
code	• 0: Enabled • 1: Disabled • 2: Abnormal Disable

AccPluginManager

Plugin Management (Currently used only in SOCKS5 acceleration).

Methodology	Description
shared	Singleton method
setAccProxyPlugin	Sets the SOCKS5 plugin.
getAccProxyPlugin	Obtains the SOCKS5 plugin instance.
setLogPlugin	Setting up a log plugin
getLogPlugin	Obtains the log plugin instance.

shared

Singleton method.

```
shared
```

setAccProxyPlugin

Set SOCKS5 plug-in.

```
func setAccProxyPlugin(_ accProxyPlugin: AccProxyPlugin?)
```

Parameter	Description
accProxyPlugin	SOCKS5 plugin instance For example: setAccProxyPlugin(TRTCAccPlugin()). TRTCAccPlugin can be obtained by importing the plugin package provided by Tencent.

getAccProxyPlugin

Retrieve the SOCKS5 plugin instance.

```
func getAccProxyPlugin() -> AccProxyPlugin?
```

setLogPlugin

Set log plugin

```
func setLogPlugin(_ logUploadPlugin: AccLogPlugin?)
```

Parameter	Description
logUploadPlugin	Log plugin instance For example: setLogPlugin(LogPlugin()). LogPlugin can be obtained by importing the plugin package provided by Tencent.

getLogPlugin

Obtain the log plugin instance.

```
func getLogPlugin() -> AccLogPluginModel
```

Model

AccConfig

Required	Description
node	Acceleration nodes. Mandatory in private cloud deployments. Optional in public cloud deployments. If they are configured, they are prioritized for use.
accMode	Acceleration mode: • Enumeration (AccMode) • Aggregation (Bonding) • Dual-transmit (Redundant) • Real-time fast switching (FastSwitching)
pingInterval	Interval for the onAccDataUpdate callback.
congestionMode	Congestion algorithm (BBR / CUBIC)
encryption	Whether encryption is enabled. Default: true.
tcpDirect	Whether TCP direct connection is enabled. Default: false.
udpDirect	Whether UDP direct connection is enabled. Default: false.

udpTimeout	UDP attempt duration
maxReconnectTimeout	Maximum connection timeout. No reconnection attempts are made after this duration is exceeded. Unit: seconds.
accRules	Acceleration rules
priority	Link priority • key: 0: mobile, 1: Wi-Fi. The priority.put(0,64) call sets the link priority for type=0 (mobile) to 64. • value: The priority is a number ranging from 0–255, where a lower number indicates a higher priority, with a default value of 64. When the priority number is >= 128, it indicates that the link is a backup link.
keepAlive	Configures whether the acc connection is kept alive. If kept alive, the acceleration status is determined by whether UDP acceleration is enabled.
networkConditions	Network conditions required to start acceleration. No conditions are required by default.
groupId	AppGroup
vpnName	VPN name
dnsServerIp	VPN DNS Server IP address. Default: 119.29.29.29.
whiteList	Allowlist. All IP addresses are blocked by default.
blackList	Sets the blocklist.
socksProxyPort	socks proxy port number. The default is 3000. If Acc is used.
socksUsername	Preset username for socks5 proxy.
socksPassword	Preset password for socks5 proxy.
extraParams	Extended parameters, such as keyId, which are passed via extended parameters.

enableMultiNetwork	Whether to enable multiple NICs in EO scenarios. Default: false. • false: Default network adapter • true: Enumerates available network adapters based on the configured priority. If the priority is not configured, enumerates all available network adapters.
addLineType(:)	Adding acceleration links in EO scenarios. 0: Direct connection, 1: EO.

Setting extraParams

Method	Description
setBool(_:,forKey:)	Sets the bool-type extraParams parameter.
setString(_:,forKey:)	Sets the String-type extraParams parameter.
setStringList(_:,forKey:)	Sets the [String]-type extraParams parameter.
setInt(_:,forKey:)	Sets the Int-type extraParams parameter.
setFloat(_:,forKey:)	Sets the Float-type extraParams parameter.

Parameters Supported by extraParams

KEY	Description
Private cloud-related parameters	
AccConfig.KEY_SECRET_KEY_ID	[String] Secret key ID
AccConfig.KEY_CIPHERTEXT	[String] Signature string
AccConfig.KEY_SIGNATURE	[String] Digital signature
Parameters related to the link disabling algorithm	
AccConfig.KEY_MAX_RTT_THRESHOLD	[Int] The CheckLatestRTT judgment is performed only when the link latency exceeds this value. Unit: ms. Default: 150 ms.
AccConfig.KEY_MIN_RTT_FACTOR	[Float64] The amplification factor for the minimum link latency. If the current link latency >

	minimum link latency * factor, the link is considered to have high latency. Default: 15.
AccConfig.KEY_LATEST_RTT_FACTOR	[Float64] The amplification factor for the last sampled latency. If the current link latency > last sampled latency * factor, the link is considered to have high latency. Default: 5.
AccConfig.KEY_MIN_WAITING_WHEN_PATH_FAILED	[Int] Minimum waiting duration for prohibiting link retention. Unit: s. Default: 3s. // Fast switchover.
Fast switch–related parameters	
AccConfig.KEY_MIN_SWITCH_RTT	[Int] For fast switchover mode, link switching is not triggered when the latency is below this value. Unit: ms. Default: 20 ms.
AccConfig.KEY_SWITCH_TOLERANCE	[Float64] For fast switchover mode, the tolerance coefficient. The current path latency is amplified and then compared with other paths to avoid frequent switching. Default: 1.125.
AccConfig.KEY_DISABLE_AUTO_SWITCH	[Bool] For fast switchover mode, whether to disable the dynamic switchover feature. Default: false.
AccConfig.KEY_MIN_DELAY_UNTIL_LOST	[Int] For fast switchover mode, the minimum latency for determining packet loss. Link disabling is considered only when the latency exceeds this value. Unit: ms. Default: 150 ms.
AccConfig.KEY_MAX_DELAY_UNTIL_FAILED	[Int] For fast switchover mode, the maximum tolerable time threshold. Packet transmission on the link is disabled when the latency exceeds this threshold. Unit: ms. Default: 460 ms.
AccConfig.KEY_TIME_REORDERING_FRACTION	[Float64] For fast switchover mode, the amplification factor for link latency, used to determine whether a link failure has occurred. Default: 2.25.
Parameters related to aggregation mode	
AccConfig.KEY_MAX_RTT_DISABLE_AGGREGATION	[Int] For aggregation mode, the link is excluded from aggregation when its latency exceeds this value. Unit: ms. Default: 460 ms.

UDP Enhanced Aggregation	
AccConfig.KEY_JITTER_DELAY	[Int] For UDP enhanced aggregation mode, the compensation latency. Unit: ms. Default: 0 ms.
AccConfig.KEY_PATH_IDLE_TIMEOUT	[Int] For UDP enhanced aggregation mode, the path idle duration. This duration is excluded from compensation. Unit: s. Default: 1 s.
AccConfig.KEY_ENABLE_SINGLE_PATH_SMOOTHING	[Bool] For UDP enhanced aggregation parameters, latency smoothing is also performed on a single path after enabling. Default: false.
AccConfig.KEY_DELAY_FACTOR	[Float64] For UDP enhanced aggregation parameters, used to calculate the one-way smoothed latency. Default: 0.125.
AccConfig.KEY_OUTER_CWND_GAIN	[Float64] quic congestion window gain factor
T2 acceleration 1. AccConfig.KEY_ENABLE_T2ACC must be set to true. 2. Set AccConfig.KEY_T2OUT for cross-domain operations. Otherwise, access may be blocked. 3. AccConfig.KEY_T2_APP_ID_LIST must be configured. Otherwise, rules cannot be matched. 4. AccConfig.dnsServerIp must be set to a non-encrypted resolution server, for example: "162.14.21.56" (Tencent International dns server).	
AccConfig.KEY_ENABLE_T2ACC	[Bool] Whether to enable T2 acceleration. Default: false.
AccConfig.KEY_T2OUT	[String] Specifies the T2 segment egress. Default: empty (automatic routing).
AccConfig.KEY_T2_APP_ID_LIST	[String] Specifies the T2 acceleration applications: App ID List.
AccConfig.KEY_T2ACC_IP_PREFIX_BLACKLIST	[String] T2 acceleration IP address CIDR blocklist (Direct).
AccConfig.KEY_T2ACC_IP_PREFIX_WHITELIST	[String] T2 acceleration IP CIDR allowlist (ACC).
AccConfig.KEY_T2ACC_DOMAIN_BLACKLIST	[String] T2 acceleration domain blocklist (Direct).
AccConfig.KEY_T2ACC_DOMAIN_WHITELIST	[String] T2 acceleration domain allowlist (ACC).
AccConfig.KEY_T2ACC_DOMAIN_SUFFIX_BLACKLIST	[String] T2 acceleration domain suffix blocklist (Direct).

AccConfig.KEY_T2ACC_DOMAIN_SUFFIX_WHITELIST	[String] T2 acceleration domain suffix allowlist (ACC).
Others	
AccConfig.KEY_SERVER_DOMAIN	[String] Certificate domain name
AccConfig.KEY_ALLOW_PRIVATE	[Bool] Allows private network interconnection.
AccConfig.KEY_MAX_RETRY_TIMES	[Int] The maximum number of retries. Default: 3. Avoid setting it to 0, as this may cause issues when there is no network connectivity.
AccConfig.KEY_ENABLE_OBFUSCATED	[Bool] Whether to enable origin IP address obfuscation.

AccNode

Required	Description
ipList	Node IP list, empty by default.
port	Node port, with a default value of –1.

AccRule

Required	Description
priority	Set the priority. A lower value indicates a higher priority. The priority is a number from 0 to 255. A lower number indicates a higher priority. The default is 64.
dstIPCIDR	Set the destination IP address. Supports CIDR notation. For example: 10.8.0.0/8. An empty value indicates matching any destination IP address.
dstPortRange	Set the destination port. Supports a range. For example: 5002–5003. An empty value indicates matching any destination port.
proto	Set the protocol type. Supported values are "TCP" and "UDP". An empty value indicates matching any protocol.
mode	Setting the acceleration mode. Supported values:

	1: bonding, 2: redundant, 3: RTC.

MpMeasureConfig

Required	Description
addr	The speed test address (for private cloud setup) is in the format "IP address:PORT". Multiple probe points are separated by semicolons, for example, "172.17.0.4:9000;172.17.0.4:9001".
t2AccProbeAddr	T2 segment speed test address
time	Sliding window measurement time (ms), default 30000
rtt	Latency threshold, default 90
loss	Packet loss rate threshold, default 5
jitter	Jitter threshold, default 15
quickTime	Quick start window time, default 10000
quickRtt	Quick start latency threshold, default 100
interval	Measurement callback frequency, default 1000
mode	Acceleration mode. Options: 1: bonding, 2: redundant, 3: RTC. Default: 2.
delayRemindTime	Delay reminder time, unit: milliseconds, default 10000
pingTimeout	ping packet timeout, unit: ms
disableQuickDetect	Disables quick detection. • true: Disable • false: Do not disable
additionalOptions	Fallback attachment options
disableSlaveLossDetect	Disables packet loss detection on the secondary path. • true: Disable • false: Do not disable

Common speed test addresses: usable as T1 and T2 segment speed test addresses.	
FRANKFURT_ADDRESS	162.62.215.101:8888
SILICON_VALLEY_ADDRESS	43.130.30.222:8888
SAO_PAULO_ADDRESS	43.135.207.42:8888
TOKYO_ADDRESS	43.128.254.208:8888
BANGKOK_ADDRESS	43.128.200.25:8888
SINGAPORE_ADDRESS	101.33.46.84:8888
HONG_KONG_ADDRESS	43.159.234.5:8888

MpAdditionalOptions

Required	Description
packetLossEscape	The acceleration link is disconnected for [packetLossCount] consecutive times.
maxRttEscape	The ping value of the acceleration link is greater than or equal to [rttThreshold] ms for [maxRttCount] consecutive times.
rttExceptionEscape	The average ping value of the acceleration link over [detectWindowTime] is greater than that of the primary link by [rttDiffThreshold]%, and the average latency of the acc link is greater than [accBenchmarkRtt] ms.
stopAccRemind	If the average latency of the acceleration link within [detectWindowTime] is greater than that of the primary link by [rttDiffRate]%, and the jitter is greater than that of the primary link by [jitterRate]%, it is recommended to disable acceleration.

MpPathDetail

Required	Description
rxBytes	Downlink data

txBytes	Uplink data
lost	Packet loss rate.
RTT	Data latency
type	NIC type • case mobile = 0 • case Wi-Fi = 1
InterfaceName	NIC name
TotalRxBytes	Total downlink traffic
TotalTxBytes	Total uplink traffic
peakRxBytes	Peak downlink traffic
peakTxBytes	Peak uplink traffic

NetworkConditions

Required	Description
MUST_NONE	No restrictions
MUST_WIFI_ACTIVE	Wi-Fi must be available.
MUST_SIM_CARD_READY	SIM card must be inserted.
MUST_MOBILE_DATA_ENABLE	Cellular data must be enabled.

Methodology	Description
combine	Combine multiple conditions

Error Code Definition

```
public enum AccErrorType: Int {
    /// vpn fd null pointer exception
    case vpnInterfaceNullPoint = -1
    /// mpquic exception
    case mpQuicLibraryException = -2
    /// vpn connection establishment exception
```

```
    case establishVpnException = -3

    /// **⚠andriod code**  Invalid package name (allowlist) exception
    @available(*, deprecated, message: "andriod code")
    case invalidPackageName = -4

    /// System exception
    case vpnDisconnectException = -5

    /// acc speed test packet loss rate anomaly
    case accContinuePackageLossException = -6
    /// acc acceleration maximum delay count anomaly
    case accMaxRttCountException = -7
    /// acc speed test contrast Wi-Fi delay anomaly
    case accRttContrastException = -8
    /// acc speed test average latency exceeds maximum threshold
    case accAvgRttException = -9
    /// acc speed test average packet loss rate anomaly
    case accAvgPackageLossException = -10

    /// acc authentication fail
    case accAuthenticationFailed = -11
    /// Authentication connection timeout
    case accAuthenticationTimeOut = -12
    /// Authentication connection exception
    case accAuthenticationException = -13
    /// SLA normal negative optimization
    case slaNormalAccException = -16
    /// Invalid acceleration parameter
    case accIllegalParamter = -17

    /// **⚠andriod code** Failed to pull up vpn service
    @available(*, deprecated, message: "andriod code")
    case setUpVpnServiceFailed = -18

    /// Failed to pull up speed test service
    case setUpMeasureFailed = -19
    /// No valid access point
    case noValidAccessPoint = -20
    /// Account authentication fail
```

```
    case accountAuthenticationError = -21

    /// Network interface status fails to satisfy acceleration
conditions
    case notMeetAccConditions = -22

    /// Invalid speed test address
    case invalidSpeedTestAddress = -23

    /// Invalid custom network interface card ⚠ Android error code
    @available(*, deprecated, message: "andriod code")
    case invalidReflectObject = -24

    /// Invalid acceleration plug-in
    case invalidAccProxyPlugin = -25

    /// No active network interface card on current device
    case noConnectedNetworkCard = -26

    /// Node udping Unreachable
    case accessPointUdpingFailed = -28

    /// Unsupported acceleration for this device
    case unsupportAccDevice = -29

    /// MpAccClient.shared.status incorrect status
    case statusError = -50

    /// Multi-channel acceleration has been pulled up
    case mulitpathStarted = 100
    /// Multi-channel acceleration internal error
    case mulitpathCoreFailed = 101
    /// Failed to pull up acceleration proxy
    case mulitpathTunDevFailed = 102
    /// Invalid acceleration node IP
    case mulitpathCoreAccIpInvalid = 103
    /// Invalid acceleration mode
    case mulitpathCoreAccModeInvalid = 104
    /// Acceleration token authentication failure
    case mulitpathCoreAccAuthFailed = 105
```

```
    /// Accelerated link notification is disabled
    case mulitpathCoreAccControllerClose = 106
    /// Channel handshake establishment timeout
    case mulitpathCoreAccHandshakeTimeout = 107
    /// Acceleration reconnection timeout
    case mulitpathCoreAccReconnectTimeout = 108
    /// socks5 proxy failure
    case mulitpathCoreSocks5Failed = 109
    /// Reached the maximum gateway access count
    case reachedMaximumNumberGratewayAccesses = 110
}
```

Advanced Features
Dynamic Acceleration

Last updated：2026-05-21 16:14:31

The SDK provides a range of network measurement APIs and policy callback APIs. By setting measurement thresholds, businesses can achieve on-demand acceleration. At the same time, the SDK also offers a fallback strategy. If significant anomalies occur during acceleration, such as network disconnection, excessive latency, negative optimization, or acceleration performance far below expectations, the SDK proactively disables acceleration and speed tests, redirecting traffic to the origin server to prevent abnormal acceleration from affecting normal business operations.

> **Note:**
> When the SDK triggers the escape onAccException callback, it will directly terminate the acceleration channel. If the acceleration was initiated using a socks5 proxy for business operations, the business must switch the appropriate network requests back to direct connection (no longer using the proxy).

```
extension TestViewController {
    func initMpAcc {
      MpAccClient.shared.setupDatakey("xxx", uuid: "xxx")
    }
    func startMeasure() {
      //Register speed test listener
      MpAccClient.shared.registerMeasureCallback(self)
      let measureConfig = MpMeasureConfig()
      /// Sliding window measurement time (ms)
      config.time = 8000
      /// Latency threshold
      config.rtt = 60
      /// Packet loss rate threshold
      config.loss = 5
      /// Jitter threshold
      config.jitter = 15
      /// Quick startup window time
      config.quickTime = 2000
      /// Quick startup latency threshold
      config.quickRtt = 80
```

```
        /// Measure callback frequency
        config.interval = 1000
        /// Acceleration mode
        config.mode = .Redundant
        /// Enable acceleration
        MpAccClient.shared.startMeasure(measureConfig: measureConfig)
    }

    func stopMeasure() {
        // Unregister network measurement listener
        MpAccClient.shared.unRegisterAccCallback(self);
        //stop speed measurement
        MpAccClient.shared.stopMeasure()
    }

}
extension TestViewController: MpMeasureCallback {
    /**
     * enable dual send
     * @param code  Trigger code 1: Latency anomaly trigger 2: Packet
loss anomaly trigger 3: Jitter anomaly trigger
     */
    func onStartMpAcc(code: Int) {
        appendLog("code:\(code)")
    }

    /**
     * disable dual send
     * @param code  Trigger code  1: ENI trigger  2: No optimization
trigger  3: No network trigger  4: Accelerated network interface and
actual active network interface not the same
     * @msg trigger information
     */
    func onStopMpAcc(code: Int, msg: String) {
        appendLog("code:\(code)")
    }

    /**
     * Current network status has no policy available
```

```
     * @param code Trigger code 1: Mobile only 2: wifi only 3: No
network 4: Internal error -19: Failed to pull up speed test service -23:
Test URL unavailable -25: plugin error -23: Invalid Speed Test -11:
Authentication error
     * @param message trigger information, if message is not empty,
mainly use message info
     */
    func onNoPolicy(code: Int, message: String?) {
        appendLog("code:\(code) msg: \(message ?? "")")
    }

    /**
     * Exception in accelerated link (in this case the sdk will
proactively disable acceleration and switch back to normal mode)
     * @param errorCode error code
     * @param msg error information
     */
    func onAccException(errorCode: Int, msg: String) {
        appendLog("errorCode:\(errorCode) msg: \(msg)")
    }

    /**
     * Bypass udping rtt delay callback
     * @param type type (0: mobile 1: wifi 2: acc)
     * @param rtt  Delay (ms) Packet loss value is 460
     */
    func onRttChanged(type: Int, rtt: Int) {
        appendLog("type:\(type) rtt:\(rtt)")
    }

    /**
     * Aggregate acceleration status change callback
     * @param state  false: off     true: on
     * @param code   0: normally enabled 1: normal shutdown 2: abnormal
shutdown
     */
    func onAccStateChanged(state: Bool, code: Int) {
        appendLog("state:\(state) code:\(code)")
    }
```

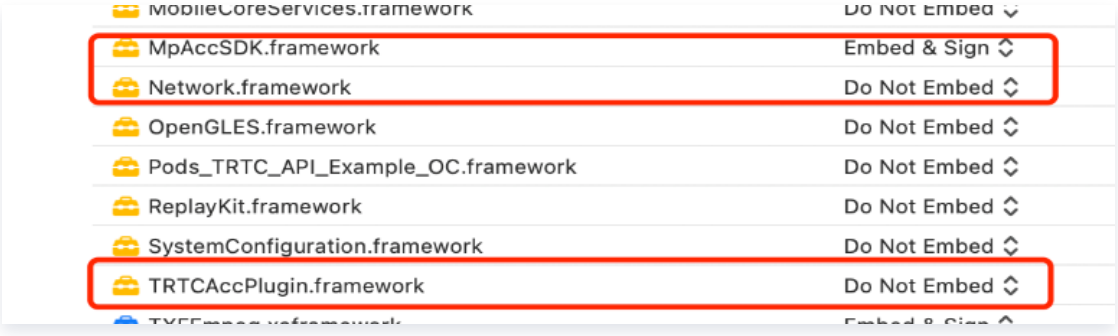
```
}
```

TRTC Plug-In Integration

Last updated：2026-05-21 16:16:48

Configuring Dependencies

First, introduce TRTCSDK. See TRTC documentation (SOCKS5 acceleration is only supported in TRTC 11.6 and later).
Then, configure acceleration and the TRTC plug-in SDK: MpAccSDK and TRTCAccPlugin. Among them, Network.framework is the system database.



Example Code

```
class TRTCPluginDemoViewController: UIViewController {
    private let accClient: MpAccClient = MpAccClient.shared

    override func viewDidLoad() {
        super.viewDidLoad()
        AccPluginManager.shared.setAccProxyPlugin(TRTCAccPlugin())
        accClient.setupDatakey("xxxx", deviceId: "xxxx")
    }

    deinit {
        AccPluginManager.shared.setAccProxyPlugin(nil)
        accClient.stop()
    }
    // Enable TRTC
   func startTRTCSDK() { /*enable VPN*/ }
}
extension TRTCPluginDemoViewController: TRTCCloudDelegate {
    func onEnterRoom(_ result: Int) {
        guard result > 0 else { return }
```

```
        accClient.registerAccCallback(self)
        let config = AccConfig()
        config.accMode = .FastSwitching  //1: Aggregation acceleration
2: Dual transmission acceleration 3: Fast switching acceleration
        config.pingInterval = 3
        accClient.start(config: config)
    }
    func onExitRoom(_ reason: Int) {
        accClient.stop()
        accClient.unRegisterAccCallback(self)
    }
}

extension TRTCPluginDemoViewController: AccCallback {
    configure callback
}
```

Log Plugin

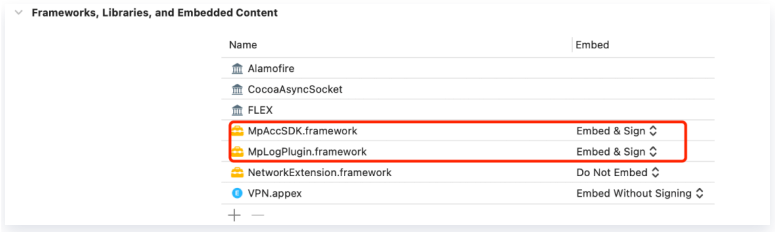
Last updated：2026-05-21 16:57:16

The SDK provides the ability for log reporting. If your application does not have the log reporting function, you can add a log reporting plug-in.

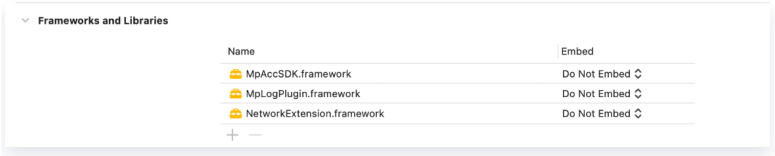
> **Note:**
> After integrating the log plug-in, inform the Tencent Side to perform allowlist processing.

1. Dependency Integration

Configure MpAccSDK and MpLogPlugin under App Target, as shown below:



If the mode is VPN, make a similar configuration under VPN Target, as shown in the figure below (the difference is in the Embed option).



> **Note:**
> The system database introduced in VPN mode is NetworkExtension.framework. In Socks5 mode, only Network.framework is required, without NetworkExtension.

2. Example Code

> **Note:**
> The plug-in needs to be set before speed up.

2.1 VPN Access

```
// App Process
let logPlugin = MpLogPlugin(
```

```
    setting: .vpnApp(dataKey: "xxxxxxxxxxxx",
                     deviceId: "xxxxxxxxxxxx",
                     groupId: "xxxxxxxxxxxxx",
                     consoleEnabled: true))
AccPluginManager.shared.setLogUploadPlugin(logPlugin)
```

```
// VPN process
class PacketTunnelProvider: NEPacketTunnelProvider {

    var mTunnelManager: MpPacketTunnelManager = MpPacketTunnelManager()

    override func startTunnel(options: [String : NSObject]?,
completionHandler: @escaping (Error?) -> Void) {
        // Add code here to start the process of connecting the tunnel.

        let logPlugin = MpLogPlugin(setting: .vpnExtension(groupId:
"xxxxxxxxxxxxx", consoleEnabled: true))
        AccPluginManager.shared.setLogUploadPlugin(logPlugin)

        mTunnelManager.startTunnel(packetTunnel: self, options: options,
completionHandler: completionHandler)
    }
    // .......
}
```

2.2 Socks5 Integration

```
//datakey registration method
let logPlugin = MpLogPlugin(
    setting: .socks5(dataKey: "xxxxxxxxxxxxxxxxxx",
                     deviceId: "xxxxxxxxxxxxxxxxxx",
                     consoleEnabled: true,
                     justAutoUploadInWiFi: true))

//application signature registration method
let plugin = LogPlugin(setting: .socks5(appId: "xxx",
                                        sign: "xxx",
                                        consoleEnabled: true,
```

```
                                        justAutoUploadInWiFi: true))

AccPluginManager.shared.setLogUploadPlugin(logPlugin)
```

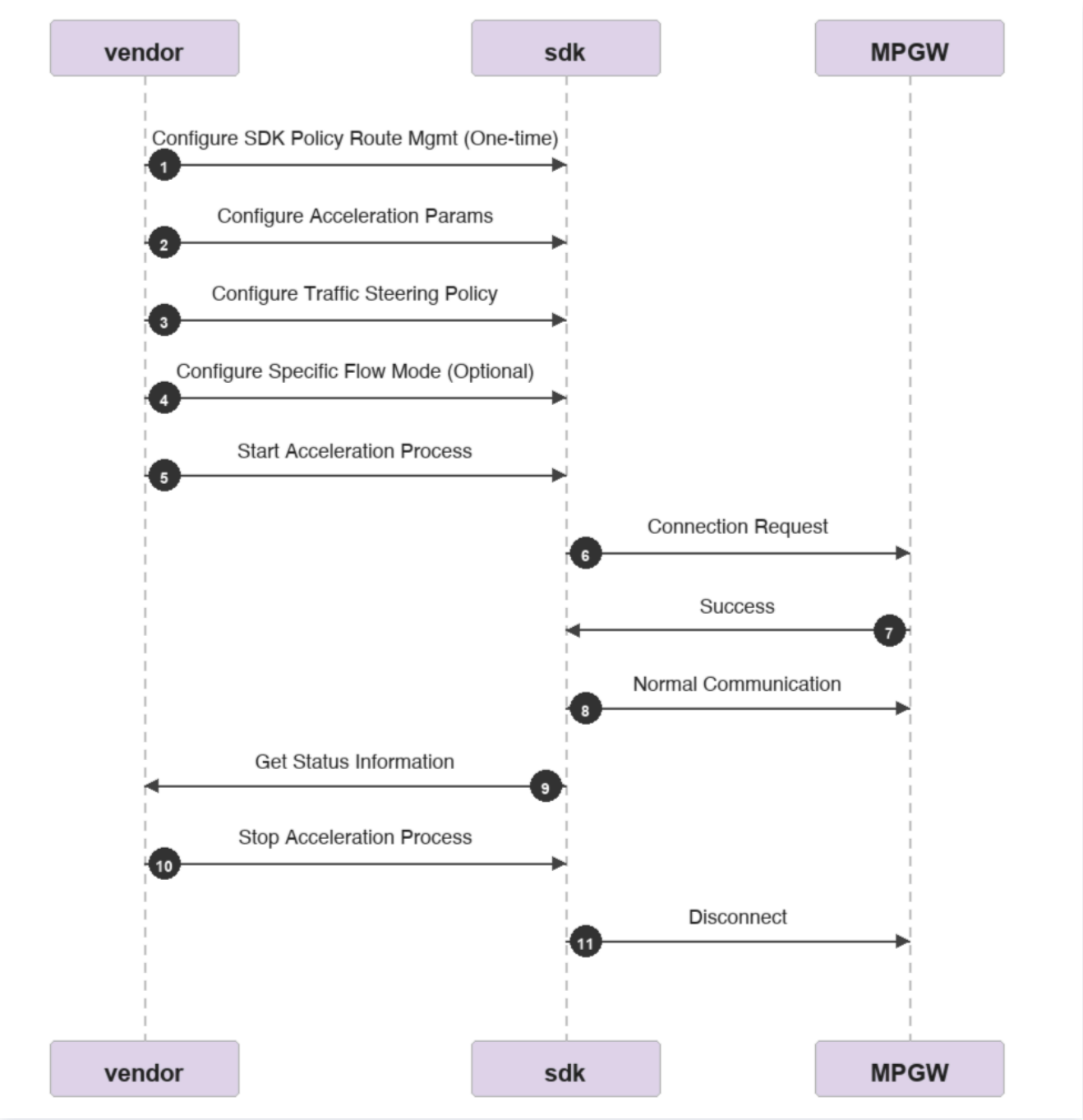
Linux SDK
API Overview

Last updated：2026-05-22 15:08:08

Access Process

The steps for integrating the Cloud Jutong Linux SDK are as follows:

1. Determine whether to enable the SDK's built-in policy routing management module (It is recommended to enable it, as configuration is required only once).
2. Configure acceleration parameters (including dataKey, interfaces, acceleration mode).
3. Configure the business traffic diversion policy to determine which traffic requires acceleration.
4. Configure flow-based multi-mode rules to direct segmented traffic to specific modes (optional).
5. Start up the acceleration process.
6. Check the status of various SDKs.
7. Stop the acceleration process.



Note:

- Business traffic diversion policy: It determines which traffic can be accelerated. Typically, it is a five-tuple rule (all or specific).
- Multi-mode rule: It determines the acceleration mode for traffic entering the acceleration process. The rule can be configured as bonding / redundant / rtc mode, corresponding to different business scenarios. For example: services with high latency requirements can use redundant mode, while audio/video services can use rtc or bonding mode.
- Relationship between the multi-mode rule and the mode in acceleration parameter configuration:

- If no multi-mode rule is configured, all services are accelerated uniformly using the mode specified in the acceleration parameters. This may not be the optimal choice for certain specific business scenarios.
- If a multi-mode rule is configured, it takes precedence.

SDK Built-In Policy Routing Management

Multiple Network Acceleration SDK includes a built-in policy routing management module. The feature is selectable. Please let the vendor manufacturer determine whether to enable it based on their own situation. Generally, we recommend using it.

Module overview of the built-in policy-based routing in the SDK

The policy routing management module built into the Multiple Network Acceleration SDK implements policy-based routing using the source IP addresses of WAN interfaces. This ensures that packets with a specified source IP address are sent out through the interface corresponding to that IP address. Without such policy routing, all packets are sent via the default route, which prevents the SDK from sending and receiving packets through multiple network interface cards.

For example, if a device has two WAN interfaces, ath0 and rmnet_mhi0.1, with IP addresses 192.168.81.80 and 10.221.7.233 respectively, and next-hop gateways 192.168.81.1 and 10.21.5.17, the SDK will generate such policy-based routing based on changes in the interface IP and default route:

```
/tmp # ip rule
0: from all lookup 128
1: from all lookup local
79: from 198.18.0.1 lookup 180
79: from all fwmark 0x1/0xf lookup 180
80: from 192.168.81.80 lookup 100
81: from 10.221.7.233 lookup 101
```

The SDK adds the policy with a priority of 79 and a routing table ID of 180 by default. This addition occurs regardless of whether the SDK's policy routing management feature is enabled.

Here, 198.18.0.1 is the IP address configured by the SDK for the mp_tun0 network interface. The firewall mark 0x1/0xf is a tag applied to the traffic that requires acceleration. The default egress for routing table 180 is the mp_tun0 interface, which ultimately directs the traffic to the mp_tun0 network card. The SDK's multi-network protocol then accelerates all traffic originating from the mp_tun0 network card.

The SDK's policy routing management adds two rules with priorities 80 and 81. Simultaneously, it adds the corresponding default routes to the routing tables with IDs 100 and 101, as follows:

```
/tmp # ip route ls table 100
default via 192.168.81.1 dev ath0
```

```
unreachable default metric 50000
/tmp # ip route ls table 101
default via 10.21.5.17 dev rmnet_mhi0.1
unreachable default metric 50000
```

Therefore, the manufacturer can determine whether to enable the built-in policy-based routing module in the SDK based on its OS circumstances. If the built-in policy-based routing module in the SDK is not enabled, the above work requirement needs to be implemented by the vendor itself.

Enable SDK policy-based routing management

```
curl -X POST 'http://127.0.0.1:9801/api/v2/route/policyRouteManagment' -
-header 'enable:true'
```

> **Note:**
> This configuration is automatically saved with persistent storage. It is recommended to execute it only once when installing the SDK. Enabling and disabling it again every start may lead to route exceptions.

Disable SDK policy-based routing management

```
curl -X POST 'http://127.0.0.1:9801/api/v2/route/policyRouteManagment' -
-header 'enable:false'
```

Accelerating Process Control

Configuring Acceleration Parameters

Before starting the acceleration process, the vendor manufacturer needs to collect the required parameters, which must be carried as parameters when launching the SDK acceleration.

Required parameter

1. Activate either of the two parameters:
   - vendor and sn: vendor is the device vendor model, sn is the serial number provided by the device vendor.
   - dataKey: device License key.
2. Interfaces: List of WAN physical interfaces for equipment participating in multi-network aggregation.
3. scheduleMode: Default work modes: bonding, redundant, rtc.

> **Note:**

Multiple Network Acceleration feature activation has two methods. The vendor can determine which one to select based on their own situation:

1. vendor+sn: Activate with the device serial number. The device to be activated must be created in the Tencent Cloud console in advance.
2. dataKey: Activated by a License key. Contact Tencent Cloud sales or business to obtain it.

scheduleMode Mode Description:

Mode	Scenario	Description
bonding	Live streaming, high bandwidth	Distributes service traffic across multiple links to fully utilize the bandwidth of all links. Benefits: high bandwidth, reliable transmission, and weak network resistance.
redundant	Gaming, control signaling	Copies and sends service packets over all links, and the receiver forwards the packet that arrives first. Effect: consumes extra bandwidth to guarantee the lowest latency.
rtc	Real-time audio and video communication	Prioritizes transmitting service traffic over the link with the lowest latency, while utilizing other links as backups. Effect: consumes no extra bandwidth, ensures smooth and non-lagging service, and provides strong weak network resistance.

Call the following API to configure acceleration parameters (only configure parameters without starting acceleration):

```
curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder'  -H
'accept: */*' -H 'Content-Type: application/json'   -d '{
        "dataKey":"get this devicekey from tencent and replace it here",
        "interfaces": ["usb0","usb1","eth0"],
        "registerEnv": -2,
        "scheduleMode": "bonding"
}'
```

This interface imports a Client entity, defined as follows:

name	type	Description
dataKey	string	Required: Choose either vendor+sn or dataKey. The device dataKey.

vendor	string	Required: Choose either vendor+sn or dataKey. The device vendor and model.
sn	string	Required: Choose either vendor+sn or dataKey. The unique device serial number provided by the device vendor.
registerEnv	integer	Required. Value: 1.
interfaces	[string]	Required. The list of interfaces participating in multi-network aggregation. Supports specifying a priority, ranging from 0 to 255. A smaller value indicates a higher priority. The priority is separated from the network interface card by a colon. If no priority is provided, it defaults to 64. Note: Priority specification is not supported in redundant mode. ["eth1:100", "eth2"].
scheduleMode	string	Required. The default acceleration mode. "bonding", "redundant", "rtc".
accGateway	string	Optional. Specify the acceleration gateway IP address. Multiple IP addresses should be separated by commas. If this parameter is not specified, the SDK will automatically connect to the nearest gateway. Otherwise, it will forcibly connect to the specified acceleration gateway, for example: "120.30.39.129".
gwPort	string	Optional. The gateway port. Default: "443".
UUID	string	Optional. The hardware fingerprint. If this parameter is not specified, the SDK will automatically generate it based on the hardware.
disableCrypto	integer	Optional. Allows the customer to choose whether to enable encryption when acceleration is started. Traffic encryption is enabled by default. Disabling encryption can reduce traffic consumption. 0: Enable traffic encryption (default). 1: Disable traffic encryption.
flowStatisticsInterval	integer	Optional. The link acceleration traffic statistics frequency. Default: 3 seconds.
maxRttDisableAggregation	integer	Optional. The latency threshold for link participation in aggregation. Default: 460 ms.

maxRttThreshold	integer	Optional. The latency for initiating link failure detection. Default: 460 ms.
minSwitchRTT	integer	Optional. The sensitivity for fast link switching. Default: 20 ms.
maxDelayUntilFailed	integer	Optional. The latency threshold for link participation in switching. Default: 460 ms.
enableRecreatePathPconn	bool	Whether to enable path recreation.

Note:

- The above configuration requires **restarting** the acceleration process to make the configuration take effect.
- Before the first acceleration, identify a network card whose mac address will not change, and write the network card name into the `/usr/local/etc/mp-speeder/mp_client_ifname.conf` file.

  You must configure this file; otherwise, acceleration will not work. Configuration command example:

```
  echo '{"ifName":"eth0"}' > /usr/local/etc/mp-
speeder/mp_client_ifname.conf
```

Note:

Call the following API to start acceleration. The SDK will then create a virtual network interface named mp_tun0. Business traffic will be guided to this virtual interface for acceleration based on the traffic diversion policy. (Before calling this API, ensure the acceleration parameters are configured.)

```
 curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder/start' -
H 'accept: */*' -H 'Content-Type: application/json'
```

After starting acceleration, you need to call the "Query Acceleration Status API" to determine whether acceleration has started successfully.

Stopping Acceleration (Asynchronous)

After multi-network acceleration is stopped, all traffic will not be accelerated, acceleration is terminated, and the mp_tun0 virtual network interface is terminated.

```
curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder/stop'  -H
'accept: */*' -H 'Content-Type: application/json'
```

After stopping acceleration, you need to call the "Query Acceleration Status API" to determine whether acceleration has been stopped.

Restarting Acceleration (Asynchronous)

Restart the acceleration process and reload configuration. This API is generally used to make the configuration take effect after modification.

```
curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder/restart'
-H 'accept: */*' -H 'Content-Type: application/json'
```

After acceleration is restarted, you need to call the "Query Acceleration Status API".

Query Acceleration Status

This API will return a Status entity containing SDK status info.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/mp-speeder" -H "accept:
*/*" -H "Content-Type: application/json"
```

Status definition

name	type	Description
ready	boolean	Whether the acceleration process is started normally.
UUID	string	The hardware fingerprint of the device.
dataKey	string	The device dataKey.
swVersion	string	The SDK software version.
accGateway	string	The IP address of the acceleration gateway.
gatewayPort	string	The port of the acceleration gateway.
interfaces	[string]	The list of interfaces participating in multi-link aggregation.

scheduleMode	string	The default acceleration mode. "bonding", "redundant", "rtc".

Query Acceleration Traffic Information

This API will return a Statistics entity, containing the cumulative accelerated traffic consumption based on the network interface card and the real-time speed of the acceleration channel. The statistical information is periodically refreshed at 10-second intervals.

```
curl -X 'GET' 'http://127.0.0.1:9801/api/v2/client/flowStatistics' -H
'accept: application/json' -H 'all: true'
```

Definition of Statistics

name	type	Description
interface	string	Name of the link NIC.
state	integer	Current link operational status: • -2: Link unavailable. • 60: Link temporarily disabled. • 100: Link normal.
totalReceivedBytes	integer	Cumulative accelerated traffic in the receive direction, in Bytes.
totalSendBytes	integer	Cumulative accelerated traffic in the send direction, in Bytes.
receivedRate	float	Receive rate of the current NIC acceleration channel, in bit/s.
sendRate	float	Send rate of the current NIC acceleration channel, in bit/s.
loss	float	Packet loss rate of the current NIC acceleration channel. The packet loss rate is expressed as a decimal, for example, 0.1 indicates a packet loss rate of 10%, and 1 indicates a packet loss rate of 100%.
rtt	integer	rtt of the current NIC acceleration channel, in ms. Note: -1 indicates that the current link is unavailable.

Business Traffic Diversion

The primary function of traffic diversion is to guide traffic of interest into the mp_tun virtual interface for acceleration. It works by tagging the configured five-tuple traffic, performing policy-based routing, and

directing the traffic through the mp_tun interface to the SDK for processing. The diverted traffic will use the default acceleration mode (configured in the "Required Acceleration Parameters").

> **Note:**
> Once you perform the traffic diversion config operation, you must **restart** the acceleration process to make the configuration take effect.
> `curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder/restart' -H 'accept: */*' -H 'Content-Type: application/json'`

Adding Full Traffic Diversion Rule

Intercept all TCP/UDP traffic from local machine–attached devices accessing the Internet. This traffic will be diverted into the SDK.

Note:

1. Full traffic diversion rules only apply to devices connected to the local machine and do not support traffic from the machine itself. To speed up local machine traffic, configure specific diversion rules with the source IP set to 0.0.0.0.
2. Full traffic diversion is configured on the lan (or LAN) interface. Ensure that the network interface card name contains the string "lan".
3. If you configure full traffic diversion without a lan (or LAN) interface, SSH traffic may be diverted, which can prevent remote access to the device.

```
curl -X 'POST' 'http://127.0.0.1:9801/api/v2/route/businessRoute' -H
'accept: */*' -H 'all: true'
```

Adding Specific Traffic Diversion Rule

Configure acceleration for specific business quintuples. Set –H 'all: false'.

```
curl -X 'POST' \
  'http://127.0.0.1:9801/api/v2/route/businessRoute' \
-H 'accept: */*' \
-H 'all: false' \
-H 'Content-Type: application/json' \
-d '[
{
"dstIP": "9.134.246.109",
"protocol": "tcp"
}
```

```
]'
```

In this example, only TCP traffic with destination IP 124.220.191.156 is accelerated.
This API carries the businessRoute entity. Parameter description:

name	type	Description
srcIP	string	Optional: The source IP address. Supports subnet masks. For example: "192.168.3.3/30".
dstIP	string	Optional: The destination IP address. Supports subnet masks. For example: "192.168.3.3/30".
protocol	string	Optional: "TCP", "UDP".
srcPorts	string	Optional: The source port. Supports a range. For example: "3303–3330".
dstPorts	string	Optional: The destination port. Supports a range. For example: "3303–3330".
isBlack	bool	Optional: Whether blocklist or not. Default: false.

Deleting Full Traffic Diversion Rule

Delete all acceleration traffic diversion routes. After deletion, traffic will no longer be accelerated, but the acceleration process remains active.

```
curl -X DELETE 'http://127.0.0.1:9801/api/v2/route/businessRoute' --
header 'all: true'
```

Deleting Specific Traffic Diversion Rule

Delete specific traffic diversion routes. After deletion, traffic under the corresponding rules will no longer be accelerated.

```
curl -X 'DELETE' \
  'http://127.0.0.1:9801/api/v2/route/businessRoute' \
-H 'accept: */*' \
-H 'all: false' \
-H 'Content-Type: application/json' \
-d '[
{
```

```
"dstIP": "124.220.191.156",
"protocol": "tcp"
}
]'
```

Querying Traffic Diversion Rules

This API will get ALL traffic diversion rules. For the definition of businessRoute, please refer to the description in the previous section.

```
curl -X 'GET' 'http://127.0.0.1:9801/api/v2/route/businessRoute'
```

Segmentation Flow Specific Pattern

This feature is optional. It is primarily used to meet the requirement for directing targeted, specific traffic flows to different modes.
Typical case: Suppose in a robot scenario, control signaling needs to use redundant mode to ensure ultra-low latency, while the onboard camera uses RTC mode to ensure smooth video without consuming double the traffic. Therefore, this feature can be leveraged to configure match rules, accurately identify control flow and video stream based on IP quintuple, and then set different running modes for the two streams.
After the match rules are configured, traffic will be processed according to the following logic:

1. First, match traffic based on the IP quintuple. If a match is found, the traffic follows the configured rule and uses the specified acceleration mode.
2. If no rule is configured in the system or no rule is matched, traffic will use the default acceleration mode. The default acceleration mode is configured in the "required parameter for acceleration".
3. Multi-mode rules support priority configuration. A lower number indicates a higher priority. The default priority is 255.

> **Note:**
> Once you perform the traffic subdivision config operation, you must restart the acceleration process to make the configuration take effect.
> `curl -X 'POST' 'http://127.0.0.1:9801/api/v2/client/mp-speeder/restart' -H 'accept: */*' -H 'Content-Type: application/json'`

Adding Multi-Mode Rule

The body of multi-mode carries the SpeedModeRule parameter:

```
curl -X 'POST' \
```

```
  'http://127.0.0.1:9801/api/v2/client/multi-mode' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"dstIP": "124.220.191.156/32",
"protocol": "UDP",
"speedMode": 3
}'
```

SpeedModeRule entity parameter description:

name	type	Description
priority	integer	Optional: 1–255. A lower value indicates a higher priority. The default priority is 255.
srcIP	string	Optional: The source IP address. Supports subnet masks. For example: "192.168.3.3/30".
dstIP	string	Optional: The destination IP address. Supports subnet masks. For example: "192.168.3.3/30".
protocol	string	Optional: "TCP", "UDP", "ANY".
srcPorts	string	Optional: The source port. Supports a range. For example: "3303–3330".
dstPorts	string	Optional: The destination port. Supports a range. For example: "3303–3330".
speedMode	integer	Required: ● 0 – "DEFAULT": Reuses the default acceleration mode. ● 1 – "DIRECT": No acceleration. Traffic uses the system's default NIC. ● 2 – "bonding": Aggregation mode. ● 3 – "rtc": Real–time audio and video mode. ● 4 – "redundant": Multi–send selective–receive mode.

Deleting Multi–Mode Rule

1. Delete a single rule.

```
curl -X 'DELETE' \
  'http://127.0.0.1:9801/api/v2/client/multi-mode' \
```

```
-H 'accept: application/json' \
-H 'all: false' \
-H 'Content-Type: application/json' \
-d '{
"priority":0,
"dstIP": "124.220.191.156/32",
"protocol": "UDP"
}'
```

Parameters as follows:

name	type	Description
priority	integer	Optional: 1–255. A lower value indicates a higher priority. The default priority is 255.
srcIP	string	Optional: The source IP address. Supports subnet masks. For example: "192.168.3.3/30".
dstIP	string	Optional: The destination IP address. Supports subnet masks. For example: "192.168.3.3/30".
protocol	string	Optional: "TCP", "UDP", "ANY".
srcPorts	string	Optional: The source port. Supports a range. For example: "3303–3306".
dstPorts	string	Optional: The destination port. Supports a range. For example: "3303–3306".

2. Delete all rules.

```
curl -X 'DELETE' \
  'http://127.0.0.1:9801/api/v2/client/multi-mode' \
-H 'accept: application/json' \
-H 'all: true' \
-H 'Content-Type: application/json' \
-d '{}'
```

Querying Multi-Mode Rules

This API returns a SpeedModeRule entity. For details about the returned parameters, refer to the SpeedModeRule definition in the previous section.

Carry –H 'all: true' in the header to query all multi–mode rules.

```
curl -X 'GET' \
  'http://127.0.0.1:9801/api/v2/client/multi-mode' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-H 'all: true' \
-d '{}'
```

Diagnostics

This feature is selectable and mainly used for product operation and maintenance.

Log Path

- The default output path for SDK logs is: /var/log, which includes mp–sdk.log and mp–speeder.log.
- If needed, you can specify the log output path. The modification method is as follows:

  Manually modify the logDir field in the /usr/local/etc/mp–speeder/log_config.json file, ensuring the format remains consistent with the file.

```
{
    "logLevel": "info",
    "autoUpload": false,
    "uploadInterval": 10,
    "logDir": "/var/log"
}
```

Note:

1. If the specified logDir path does not exist, it is created by default.
2. After making modifications, you must restart the SDK.

Log Reporting

Log reporting is selectable for problem localization. There are two methods for log reporting.

- Manual reporting: Execute once command to report multi–network logs saved locally on the device to the backend.
- Automatic reporting: After you set the reporting interval and enable the switch, logs are periodically reported to the backend. You can disable the switch at any time to stop reporting logs.

> **Note:**
> - Local logs occupy up to 50MB of storage space. After they are reported, locally stored logs are automatically cleaned up.
> - Modify log collection parameters, the SDK must be restarted for the changes to take effect.

1. Configure log reporting parameters.

```
curl -X POST 'http://127.0.0.1:9801/api/v2/diagnosis/log' -H 'accept:
application/json' -H 'Content-Type: application/json' -d '{
"logLevel": "debug",
"upload": false,
"uploadInterval": 10
}'
```

Parameter description:

name	type	Description
logLevel	string	Optional: The program log level. Defaults to "info". Supports configuration of "info", "debug", and "warn".
autoUpload	boolean	Optional: The automatic upload switch. Defaults to false. When set to true, logs are automatically uploaded.
uploadInterval	integer	Optional: The automatic upload interval. Defaults to 10 minutes. This parameter does not take effect when automatic upload is not enabled.

2. Manual log reporting

The one-time log reporting feature is recommended.

```
curl -X POST 'http://127.0.0.1:9801/api/v2/diagnosis/log'
```

Speed Test Escape Callback

Adding Callback API

```
curl -X POST
'http://127.0.0.1:9801/api/v2/client/setCallback'
  -H 'accept: application/json'
  -H 'Content-Type: application/json'
```

```
  -d '{
  "callback_url": "http://127.0.0.1:9801/api/v2/client/callBackMock"
}'
```

Callback message body

```
{
    "acc_mode": "1",
    "mandatory": true,
    "code": 2,
    "reason": "Acc RTT 496.634µs over Master 447.855µs * 1 in 5m0s"
}
```

Field description:

Field	Type	Meaning
acc_mode	string	Acceleration mode. Acceleration mode. • 1: Aggregation mode; • 2: Dual-send mode; • 3: Fast-switch mode.
mandatory	bool	Notifies the SDK whether to actively disable acceleration. When false, the notification event is only a suggestion.
code	int	Error code for disabling acceleration. Return value for the code field: • -6: ACC link sustained packet loss anomaly; • -7: ACC link maximum delay anomaly, where the ACC link delay continuously exceeds the specified threshold while the primary link remains normal; • -8: ACC link average delay exceeds the primary link average delay within the time window; • -9: ACC link average delay exceeds the configured threshold; • -100: A link anomaly may exist, applicable only to aggregation mode; • 2: Acceleration is ineffective; it is recommended to disable acceleration.
reason	string	Specific reason for disabling acceleration.

> **Note:**
> After the configuration is complete, you need to restart acceleration for it to take effect.

Speed Test Configuration

> Note: This feature requires manual configuration of the speed test configuration file. The file location is `/usr/local/etc/mp-speeder/metric.json`. You can use default configuration or customize it.

Configure according to the necessary acceleration mode. Here is an example for aggregation mode:

```
"bonding":{
        "probeCfg": {
            "fastProbeInterval": 200,
            "normalProbeInterval": 1000,
            "minTimeForSelectProbePoint": 3000
        },
        "checkCfg": {
            "quicDetectTime": 10000,
            "primaryDetectTime":60000,
            "secondaryDetectTime":300000,
            "iQRAlpha": 0.75,
            "minDetectCount": 4,
            "maxDetectRTT": 400,
            "detectTimeout": 1000
        },
        "enableAcc": {
            "lossRate": 5,
            "quicRTT": 100,
            "avgRTT": 110,
            "jitterRTT": 20,
            "mdevRTT": 20,
            "maxRTT": 300,
            "disableQuicWndDetect": true,
            "disableAvgRTTDetect": false,
            "disableJitterDetect": false,
            "disableLossDetect": false
        },
```

```
        "disableAcc": {
            "lossCount": 4,
            "maxRTTCount": 4,
            "avgRTT": 130,
            "toleranceRate": 1.1,
            "minAvgRTT": 66,
            "SecondaryToleranceRate":1.0,
            "SecondaryJitterRate":0.8,
            // Escape: ${lossCount} consecutive packet losses and no
consecutive packet loss on the primary link
            "EnableLossDetect": true,
            // Escape: ${maxRTTCount} consecutive RTTs above
${checkCfg.maxDetectRTT} and normal primary link
            "EnableMaxRTTDetect": false,
            // Escape: The avgRTT of the acc link exceeds
${disableAcc.avgRTT} within the ${checkCfg.primaryDetectTime} time
window
            "EnablePrimaryAvgRTTDetect": false,
            // Escape: The avgRTT of the acc link exceeds that of the
primary link within the ${checkCfg.primaryDetectTime} time window
            "EnablePrimaryDetect": false,
            // Recommended to disable: The avgRTT of the acc link
exceeds that of the primary link within the
${checkCfg.secondaryDetectTime} time window
            "EnableSecondaryDetect": false,
            // Recommended to disable: ALL secondary links with packet
loss in the time window ${checkCfg.primaryDetectTime}
            "EnableSlaveLossDetect": false
        }
    }
```

Note:
After the configuration is complete, you need to restart acceleration for it to take effect.

Viewing Callback API

```
curl -X 'GET'
'http://127.0.0.1:9801/api/v2/client/getCallback'
```

```
  -H 'accept: application/json'
  -H 'Content-Type: application/json'
```

FAQ

Last updated：2026-05-21 16:24:52

This document is for customer guidance on locating and troubleshooting problems when using Multiple Network Acceleration product.

1. Common Reasons for Business Performance Issues and Troubleshooting Methods

This section mainly introduces common causes and troubleshooting methods when business effects are not meeting expectations while users are using aggregation and real-time acceleration.
Business impact does not meet expectations. For example: video stuttering, screen glitch, or frame loss during push/pull streaming.

1.1 Common Reasons

1.1.1 Inappropriate Acceleration Mode Selection

The acceleration mode is configured inappropriately, for example, real-time mode should be used instead of aggregation mode. Choose an appropriate acceleration mode based on business type and performance requirement. For specific configuration, consult Tencent Operation and Maintenance Personnel.
For the acceleration mode query method, see Checking Acceleration Mode and Access Gateway IP address.

1.1.2 Inappropriate Business Parameter Setting

- Business-related parameters: Inappropriate business-related settings, commonly used:
  - Excessively large MSS leads to reduced performance in UDP packet transfer.
  - Setting the live streaming push tolerance delay lower than the link's working RTT causes the link to not participate in aggregation or to drop packets.
- Algorithm-related parameters of Multiple Network Acceleration. The default configuration can generally meet the requirements. For special requirements, consult Tencent Operation and Maintenance Personnel.

1.1.3 Poor Network Environment

Business runs normally in most scenarios and most time, but fails to meet expectations at specific physical locations or during peak hours, which might be due to poor current network environment unable to meet business needs.
In case of poor network quality in part or all links (such as excessively high latency, large latency jitter, or high packet loss rate), the issues may exist.

- **Aggregation mode**
  - The aggregated bandwidth of each link is less than the business bandwidth requirement. The current network environment does not meet business needs. Physical links need to be replaced or added.

- Some links are disabled. The aggregated bandwidth of the remaining links is less than the business bandwidth requirement. The current network environment does not meet business needs. Links need to be changed or physical links added.
- The aggregated link bandwidth meets the business bandwidth requirement, but occasional lags, packet loss, or out-of-order packets might occur. This could be due to a sudden decrease in current network quality (jitter, packet loss, latency).

- **Real-time mode**
  - The bandwidth of the selected transmission link is still less than the business requirement. The current network environment does not meet business needs. The terminal location needs to be changed or the physical link replaced.
  - The delay of the selected transmission link is too large, or the jitter is high, which does not meet business needs. The current network environment does not meet business needs. The terminal location needs to be changed or the physical link replaced.
  - Link switching is frequent, and the delay jitter does not meet business needs. The current network environment does not meet business needs. The terminal location needs to be changed or the physical link replaced.

> **Note:**
> To troubleshoot and confirm the existence of poor network environment, use one or more of the following methods for troubleshooting and confirmation.

1.2 Troubleshooting Method

1.2.1 Checking Acceleration Mode and Access Gateway IP address

```
curl -s -X 'GET' 'http://127.0.0.1:9801/api/v2/client/mp-speeder' -H
'accept: */*' -H 'Content-Type: application/json'
```

- scheduleMode (acceleration mode)
  - bonding: aggregation
  - fastSwitching: real-time
  - redundant: dual-transmit
- accGateway (access gateway IP)

1.2.2 Traffic Monitoring

Log in to the Tencent Cloud console to view the real-time network status of the specified device, including uplink traffic, downstream traffic, uplink rate, downstream rate, packet loss, and latency.

- Check the rate of each link for any abnormal phenomenon such as a severe rate drop or a fall to zero.

- Check the latency and packet loss rate of each link to determine whether there is a latency surge or significant jitter (for example: latency increases from 80ms to 300ms) or a packet loss rate surge (for example: packet loss rate rises from nearly zero to over 10%).

> **Note:**
> You can further verify this by analyzing the 1.2.3 SDK Log Analysis.

1.2.3 SDK Log Analysis

SDK log local file storage path: `/var/log/mp-sdk.log and /var/log/mp-speeder.log`.
Common analysis metrics for logs and fault judgment methods follow these steps:

- Registration failed, search keywords: "sdk register failed". The issue is caused by datakey invalidation.
- Trigger link disabled, search keywords: "potentiallyFailed".
- Restore link after disabling, search keywords: "potentiallyOK".
- datakey reuse by device causes accelerated connection interruption. Search keywords: "close session by controller".
- Activate path, search keywords: "Enable PathID".
- Close path, search keywords: "Close PathID".

1.2.4 udpping Speed Test

The udpping tool is an acceleration gateway service. It works by sending probe packets to designated probe points through specified network ports at assigned frequency, collecting link delay changes. It can be used to compare the delay between physical links and accelerated links.

```
 /usr/local/bin/mp-speeder/udping -h
Usage of /usr/local/bin/mp-speeder/udping:
  -C int
        ping count
  -I string
        bind interface name
  -c string
        probe address, IP:PORT
  -i duration
        ping interval (default 1s)
  -l int
        payload length
  -logLevel string
        log level, default info
  -o    output to shell
```

```
  -v    show version
  -w duration
        ping timeout (default 1s)
```

Parameter setting suggestion:

```
-c string
      probe address, IP:PORT
```

> **Note:**
> The IP address can be obtained from the "Acceleration Mode and Access Gateway IP Query" section, and the PORT is 8888.

UDP traffic generation usage example:

- Assess the RTT delay from the designated network card (such as eth0) to the specified probe point (for example: access gateway IP 49.7.248.202). The command meaning: specify to collect the delay of eth0 network interface card accessing gateway 49.7.248.202, detection message length 200 bytes, one probe every 0.1s, total 100 probes.

```
  Reference command: /usr/local/bin/mp-speeder/udping -C 100 -I eth0 -c
49.7.248.202:8888 -i 0.1s -l 200
```

- Assess the RTT delay of the accelerated link. The following command: collects the delay of the accelerated link when the SDK accesses the gateway at 49.7.248.202, using a probe packet length of 200 bytes, one probe every 0.1 seconds, for a total of 100 probes.

```
  Reference command: /usr/local/bin/mp-speeder/udping -C 100 -I mp_tun0
-c 49.7.248.202:8888 -i 0.1s -l 200
```

1.2.5 UDP Traffic Testing

UDP traffic generation uses the iperf3 service, a commonly used network speed test. UDP traffic generation can serve as an auxiliary tool to monitor link bandwidth and packet loss rate changes. Self-deployment of the iperf3 client and services is required.

UDP traffic generation usage example:

- Assess the UDP transmission bandwidth from the specified network interface (eth0) to the IP address (x.x.x.x). The IP address x.x.x.x hosts the Iperf3 service, and the service port is 5201.

```
Reference command: iperf3 -c x.x.x.x -p 5201 -u -b 100M -B eth0
```

- Assess the UDP bandwidth of the accelerated link. Among them, x.x.x.x is the IP provided by the Iperf3 service, and the server port is 5201.

```
Reference command: iperf3 -c x.x.x.x -p 5201 -u -b 100M -B 192.18.0.1
```

1.2.6 Network Measurement Callback Listener API

The network measurement callback listener API can obtain RTT, rate, and packet loss metrics for each in-band link. The measurement results synchronize with the business. User-saved API callback results are required.
Query the traffic statistics API.

```
curl -X 'GET' 'http://127.0.0.1:9801/api/v2/client/flowStatistics' -H
'accept: application/json' -H 'all: true'
```

2. Troubleshooting Methods for Common Issues with Linux SDK

2.1 Unable to Start Acceleration or Traffic Not Connected after Acceleration

2.1.1 Execute One-Click Detection Script

When encountering acceleration that cannot start or traffic not connected after acceleration, run the one-click detection script and make corrections based on the detection result. Until ALL check results normal.
The Linux one-click detection script supports the following detection capabilities:

- Detect acceleration status (key process, policy-based routing, traffic diversion rule).
- Detect rp_filter configuration.
- Detect L3 feature configuration.

The detection script of V0.16.1 and later versions has been placed in this directory: `/usr/local/bin/mp-speeder/mp_check.sh` (if there is no mp_check.sh in this directory, copy the following files to this directory).

```
Code explanation Code rewrite #!/bin/bash

# Set color output
YELLOW='\033[0;33m'
GREEN='\033[0;32m'
RED='\033[0;31m'
```

```
NC='\033[0m' # No Color

# Global variable used to store the list of ENIs
interfaces=""

# Function: Check whether the process is running
check_process() {
    local process_name=$1
    if pgrep -f "$process_name" > /dev/null; then
        echo -e "${GREEN}[✓] Process $process_name is running${NC}"
        return 0
    else
        echo -e "${RED}[✗] Process $process_name is not running${NC}"
        return 1
    fi
}

# Function: Check network interface card existence and start up
check_interface() {
    local interface=$1
    if ip link show "$interface" &> /dev/null; then
        if ip link show "$interface" | grep -q "UP"; then
            echo -e "${GREEN}[✓] Network interface card $interface is
started and status is normal${NC}"
            return 0
        else
            echo -e "${RED}[✗] Network interface card $interface exists
but not started${NC}"
            return 1
        fi
    else
        echo -e "${RED}[✗] Network interface card $interface does not
exist${NC}"
        return 1
    fi
}

# Function: Check rp_filter
check_rp_filter() {
```

```
    if sysctl net.ipv4.conf.mp_tun0.rp_filter | grep -q
"net.ipv4.conf.mp_tun0.rp_filter = 1"; then
        echo -e "${RED}[✗] rp_filter is not disabled${NC}"
        return 0
    else
        echo -e "${GREEN}[✓] rp_filter is closed${NC}"
        return 1
    fi
}

# Function: Check policy-based routing
check_policy_routing() {
    # Check policy rule existence
    if ip rule | grep -q "from 198.18.0.1 lookup"; then
        local table_num=$(ip rule | grep "from 198.18.0.1 lookup " | awk
'{print $NF}')
        echo -e "${GREEN}[✓] Find policy rule: from 198.18.0.1 lookup
$table_num${NC}"

        # Check default route in routing table
        if ip route show table "$table_num" | grep -q "default dev
mp_tun0"; then
            echo -e "${GREEN}[✓] Correct default route exists in route
table $table_num${NC}"
        else
            echo -e "${RED}[✗] Correct default route not found in route
table $table_num${NC}"
            return 1
        fi
    else
        echo -e "${RED}[✗] Policy rule not found: from 198.18.0.1${NC}"
        return 1
    fi

    # Check policy rule existence based on source ip

    for iface in $interfaces; do
        # Get the IP address of the network interface card
        local ip_addr=$(ip addr show $iface | awk '/inet / {print $2}' |
cut -d/ -f1|head -n 1)
```

```
        if [ -z "$ip_addr" ]; then
            echo -e "${YELLOW}[-] Network interface card $iface has no
IP address configuration and cannot be used to speed up. Suggest
checking${NC}"
            continue
        fi

        # Check policy rule existence
        if ip rule | grep -q "from $ip_addr lookup"; then
            local table_num=$(ip rule | grep "from $ip_addr lookup" |
awk '{print $NF}')
            echo -e "${GREEN}[✓] Find policy rule for network interface
card $iface ($ip_addr): lookup $table_num${NC}"

            # Check whether there is a route in routing table
            if ip route show table "$table_num" | grep -q "dev $iface";
then
                echo -e "${GREEN}[✓] Route table $table_num configured
correctly${NC}"
            else
                echo -e "${RED}[✗] Route not found in route table
$table_num${NC}"
                return 1
            fi
        else
            echo -e "${YELLOW}[-] Policy rule for network interface card
$iface ($ip_addr) not found, may affect acceleration performance, enable
policy-based routing management in SDK${NC}"
        fi
    done
    return 0
}

# Function: check speeder status
check_speeder_status() {
    local response
    response=$(curl -s -X 'GET' 'http://127.0.0.1:9801/api/v2/client/mp-
speeder' -H 'accept: */*' -H 'Content-Type: application/json')

    if echo "$response" | grep -q '"ready":true'; then
```

```
        echo -e "${GREEN}[✓] speeder is running and acceleration status
is normal${NC}"

        # Extract list of ENIs
        interfaces=$(echo "$response" | sed -n 's/.*"interfaces":\[\
([^]]*\)\].*/\1/p' | sed 's/"//g' | sed 's/:[0-9]*//g' | tr ',' ' ')
        echo -e "${GREEN}[✓] List of ENIs: $interfaces${NC}"
        return 0
    else
        echo -e "${RED}[✗] speeder is not running or acceleration status
is abnormal${NC}"
        return 1
    fi
}

# Function: Check traffic diversion rule
check_iptables_rules() {
    echo "Start checking traffic diversion rule..."

   # Check traffic diversion rule
    if iptables -L mp_route_mark -nv -t mangle | grep -q "0x1/0xf"; then
        echo -e "${GREEN}[✓] Traffic diversion rule check passed${NC}"
    else
        echo -e "${YELLOW}[-] No traffic diversion rule detected, please
confirm whether it meets expectations${NC}"
    fi
    return 0
}

# Function: Check L3 feature
check_l3_functionality() {
    echo "Start checking L3 feature..."

    # Check wireguard virtual port status
    # /usr/local/bin/mp-speeder/wg show
    # interface: 09yg7kk9di
    #   public key: ISh0Y01aQ3dig5gyuivFnQe8lxi8Sdp3epNReVIYVSI=
    #   private key: (hidden)
    #   listening port: 55368
```

```
    # peer: RzZ9m8lgMVeFzILbkfLmlM9I3ZBuGoP4hrfe78Vjzg0=
    #   endpoint: 198.18.0.5:10001
    #   allowed ips: 192.168.18.0/28
    #   latest handshake: 1 minute ago
    #   transfer: 3.66 MiB received, 17.26 MiB sent
    #   persistent keepalive: every 15 seconds
    local response
    response=$(/usr/local/bin/mp-speeder/wg show)
    if echo "$response" | grep -q "latest handshake"; then
        echo -e "${GREEN}[✓] wireguard virtual port status is
normal${NC}"
    else
        echo -e "${RED}[✗] wireguard virtual port status exception${NC}"
        return 1
    fi

    # Check the route table
    local interface_name=$(echo "$response" | grep "interface:" | awk
'{print $2}')
    local allowed_ips=$(echo "$response" | grep "allowed ips:" | awk
'{print $3}')

    if [ -z "$interface_name" ] || [ -z "$allowed_ips" ]; then
        echo -e "${RED}[✗] unable to get interface name or allow IP from
wireguard configuration${NC}"
        return 1
    fi

    # Check the route table
    local routing_tables=$(ip route)
    local expected_route="$allowed_ips dev $interface_name scope link"
    if echo "$routing_tables" | grep -q "$expected_route"; then
        echo -e "${GREEN}[✓] Correct route exists in routing table:
$expected_route${NC}"
    else
        echo -e "${RED}[✗] Correct route not found in routing table:
$expected_route${NC}"
        return 1
    fi
```

```
    # Check whether the endpoint conforms to the 198.18.x.x format
    local endpoint=$(echo "$response" | grep "endpoint:" | awk '{print
$2}')
    if echo "$endpoint" | grep -q "198.18."; then
        echo -e "${GREEN}[✓] endpoint complies with the 198.18.x.x
format${NC}"
    else
        echo -e "${RED}[✗] endpoint does not match the 198.18.x.x
format${NC}"
        return 1
    fi

    # Check traffic diversion rule
    # The following rules must exist in the output of iptables -L -nv -t
mangle
    # The OUTPUT chain has three chains: mp_route_bypass, mp_route_mark,
and mp_l3_route, and mp_route_bypass must be first.
    # 2. The mp_route_bypass chain must have one rule with destination
as allowed_ips and target as ACCEPT
    # 3. The mp_l3_route chain must have one rule with destination as
endpoint and target as MARK
    local mangle_rules=$(iptables -L OUTPUT -nv -t mangle)

    # 1. Check chain sequence in OUTPUT chain
    if ! echo "$mangle_rules" | grep -q "mp_route_bypass" || \
       ! echo "$mangle_rules" | grep -q "mp_route_mark" || \
       ! echo "$mangle_rules" | grep -q "mp_l3_route"; then
        echo -e "${RED}[✗] Necessary chain lacks in OUTPUT chain${NC}"
        return 1
    fi

    # Confirm mp_route_bypass is in the first position
    if ! iptables -L OUTPUT  1 -nv -t mangle | grep -q
"mp_route_bypass"; then
        echo -e "${RED}[✗] mp_route_bypass is not in the first position
of OUTPUT chain${NC}"
        return 1
    fi

    # 2. Check rules in mp_route_bypass chain
```

```
    if ! iptables -L mp_route_bypass -nv -t mangle | grep -q
"ACCEPT.*$allowed_ips"; then
        echo -e "${RED}[✗] The mp_route_bypass chain lacks an ACCEPT
rule for target $allowed_ips${NC}"
        return 1
    fi

    # 3. Check rules in mp_l3_route chain
    endpoint=$(echo "$endpoint" | awk -F ':' '{print $1}')
    if ! iptables -L mp_l3_route -nv -t mangle  | grep -q
"$endpoint.*MARK.*0x1/0xf"; then
        echo -e "${RED}[✗] The mp_l3_route chain lacks a MARK rule for
target $endpoint${NC}"
        return 1
    fi

    echo -e "${GREEN}[✓] iptables rules check passed${NC}"

    echo "L3 feature check done"
    return 0
}

# Parse command-line parameters.
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -h|--help)
            echo "Usage: $0 [options]"
            echo "Options:"
            echo "  -h, --help          Display help information"
            echo "  -l, --l3-check      Check L3 feature"
            exit 0
            ;;
        -l|--l3-check)
            l3_check=true
            shift
            ;;
        *)
            echo -e "${RED}[✗] Unknown parameter: $1${NC}"
            exit 1
            ;;
```

```
    esac
done



echo "Start system check..."
echo "----------------------------------------"

echo "1. Check process status:"
# Check mp-sdk process
check_process "mp-sdk"
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

# Check mp-speeder process
check_process "mp-speeder"
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

echo -e "\n2. Check network interface card status:"
check_interface "mp_tun0"
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

echo -e "\n3. Check acceleration status:"
check_speeder_status
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

echo -e "\n4. Check rp_filter configuration:"
check_rp_filter
if [ $? -ne 0 ]; then
```

```
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

echo -e "\n5. Check routing configuration:"
check_policy_routing
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

echo -e "\n6. Check traffic diversion rule configuration:"
check_iptables_rules
if [ $? -ne 0 ]; then
    echo -e "\n${RED}Check failed, stop${NC}"
    exit 1
fi

# If the --l3-check parameter is specified, then execute L3 feature
check
if [ "$l3_check" = true ]; then
    echo -e "\n5. Check L3 feature:"
    check_l3_functionality
    if [ $? -ne 0 ]; then
        echo -e "\n${RED}L3 feature check failed${NC}"
        exit 1
    fi
fi

echo "----------------------------------------"
echo -e "${GREEN}ALL check items are normal${NC}"
exit 0
```

2.1.2 One-Click Troubleshoot Anomaly Results and Solution

Check Item	Detailed Description	Abnormal Detection Result	Solution

Check Process Status	–	Process not running.	1. The SDK process is not running: It is configured for auto-start on boot. Check the service status, or you can use `Systemctl restart mp-sdk`. 2. The mp-speeder process is not running: Check whether the acceleration API has been invoked. If not, invoke the acceleration API.
Check NIC Status.	–	NIC exists but not started.	Check the "Check speeder status" item.
	–	NIC not present.	Check whether the tun kernel module exists (modinfo tun).
Check rp_filter.	–	rp_filter not disabled.	sysctl -w net.ipv4.conf.mp_tun0.rp_filter=0.
Check Policy Routing.	–	Correct default route not found in the routing table.	Restart the SDK.
	–	Policy rule not found.	Restart the SDK.
Check whether source IP address-based policy rules exist.	Obtain NIC IP address.	NIC not configured with an IP address, acceleration via this NIC unavailable, check recommended.	Monitor it and no action is required.
	Check policy rule status.	Route not found in the routing table.	Restart the SDK.
		Policy rule for the NIC not found, which may affect acceleration performance. Enable SDK policy routing management.	Enable SDK policy routing management.
Check speeder Status.	–	Not running or acceleration status abnormal.	1. Call the API to start acceleration. 2. Startup failed. Check for possible causes: datakey invalidation or network disconnection.

			3. For other situations, contact Tencent Cloud Ops personnel.
Check Traffic Steering Rules	–	Traffic steering rules not detected, confirm whether this is as expected.	1. Confirm whether the API has been invoked to add the traffic diversion policy. 2. Addition failures are usually caused by traffic diversion rule configuration issues, such as spelling errors (such as TCP, UDP, and so on) or port numbers exceeding 65536. Check and resolve these issues yourself. 3. For other situations, contact Tencent Cloud Ops personnel.
Check L3 Feature.	Check wireguard virtual interface status.	wireguard virtual interface status abnormal.	1. If the wireguard virtual port is missing, which may be due to network issues or loss of connection with the controller, prioritize troubleshooting these causes. 2. For other situations, contact Tencent Cloud Ops personnel.
	Check routing table	Correct route not found in the routing table.	1. If a machine has a route identical to L3, an ECMP scenario should be formed, which is currently not supported. 2. For other situations, contact Tencent Cloud Ops personnel.
	Check whether the endpoint matches the 198.18.x.x format.	endpoint does not match the 198.18.x.x format.	Contact Tencent Cloud Ops personnel.
	Check traffic steering rules – check chain order in OUTPUT chain.	Necessary chain missing in the OUTPUT chain.	Contact Tencent Cloud Ops personnel.
		mp_route_bypass is not the first chain in the OUTPUT chain.	This issue is usually caused by users manually adding other rules. Please check it yourself.

	Check traffic steering rules – check rules in mp_route_bypass chain.	The mp_route_bypass chain lacks the target ACCEPT rule.	This issue is usually caused by users manually deleting the corresponding rules. Please check it yourself.
	Check rules in mp_l3_route chain.	The mp_l3_route chain lacks the target MARK rule.	This issue is usually caused by users manually deleting the corresponding rules. Please check it yourself.

2.2.1.2 Partial Links Disabled Due to High Packet Loss Rate

Common Issues:

- Multi-NIC acceleration performance is lower than the sum of link bandwidths.
- Single ENI acceleration performance is much lower than the link bandwidth without acceleration.

2.2.1 Common Reasons and Solution

2.2.1.1 Partial Links Disabled Due to Poor Network Quality

Common Reasons for Link Disabling: Excessive RTT latency or jitter, and packet loss on the link. You can confirm the link's network latency, packet loss, and disabled status by analyzing the 1.2.3 SDK logs.

Solution:

- Perform a speed test on the physical link with poor quality after network change.
- Adjust the link disable condition and increase the following parameters (such as higher than the actual link delay) to make ALL links participate in aggregation as much as possible.

  Note: The default parameter balances bandwidth and delay for comprehensive performance. This adjustment is only used for testing the limit of aggregate bandwidth.
  - MaxRttDisableAggregation: latency threshold for aggregation links.
  - MaxRttThreshold: threshold for linkage failures.

2.2.1.2 IoT SIM Card Issue

Scenario: One or multiple IoT SIM cards are involved in acceleration, but the IoT SIM cards have not whitelisted our server ip, which can cause poor performance. This stream will be disabled.

Solution:

- Replace the network interface card with an ordinary one.
- Whitelisting the server ip for IoT SIM card on Cloud Connect.

2.2.1.3 Limited CPU Capability of Endpoint Devices

Query the device CPU usage. If the CPU usage rate of each core is high (for example, 90%), it indicates that the performance is limited by the device capacity.

For performance of typical configuration, consult Tencent Operation and Maintenance Personnel.

2.2.1.4 Policy-Based Routing Not Configured Correctly

Monitor whether the traffic is sent from multiple acceleration NICs.
If not sent from multiple acceleration NICs, confirm whether policy-based routing is configured for each network interface card through following steps:

- Method 1: Execute the one-click detection script to check whether policy-based routing based on source IP is configured for each accelerated network interface card. Configure the correct policy-based routing according to the prompts.
- Method 2: Query IP rule and policy-based routing via command line.

A default route will be configured in the corresponding routing table.

2.2.1.5 Multiple Acceleration Network Interface Cards Use the Same Operator Exit

If multiple SIM cards in a cellular network belong to the same carrier, they may access the same base station community. Subject to the overall bandwidth capacity of the community, adding more SIM cards may not increase the total bandwidth. Consider switching to other ISP SIM cards to expand the total link bandwidth.
Query the public IP address of the network interface card.

```
Reference command: curl --interface eth0 ip.sb
```

For the carrier IP address attribute query platform, refer to this website.

2.2.1.6 Caused by Low socket buffer

Check the packet sending and receiving status during traffic testing. If udp packet sending and receiving buffer errors are detected, increase this value.

- Continuously monitor the iperf server socket buffer using `netstat -su` and check for buffer errors.
- Check the socket buffer size.
- Adjust the socket buffer size.
- Restart after adjustment to speed up.

2.2.1.7 Hardware Does Not Support Encryption and Decryption Commands

Solution:

- Disable encryption and decryption.

  `/usr/local/etc/mp-speeder/mp_client_extend.conf` file configuration **-disableCrypto** to disable data encryption/decryption functionality.
- Use hardware that supports encryption and decryption commands.

2.3 Iperf Traffic Test Experiences Flow Interruption

Problem manifestation: During iperf TCP traffic testing for aggregated link bandwidth measurement, traffic experiences a bottoming-out phenomenon.

2.3.1 All Links Participating in Aggregation Experience Packet Loss in Weak Network Conditions

Analyze SDK logs to confirm network delay, packet loss, and disabled status.
Log retrieval keyword: potentiallyFailed.

2.3.2 Frequent Handover Between 4G/5G Mobile Networks

The basic network performance of a single network interface card is poor. Without acceleration, it also shows large latency jitter and stream interruption during traffic testing.
You can use the 4/5G module AT Instruction Set to view the network the module stays on, signal strength, and signal quality.

2.4 Policy-Based Routing Issue

2.4.1 SDK Adds Multiple Duplicate Policy Routes

Check whether the SDK is pulled up repeatedly.
View method: Check `/var/log/mp-sdk.log` to see whether there is multiple occurrence of "Server started on ..." at the same time.

2.4.2 Detailed Route Error or Empty

Check whether the script pulls up the program with repeated deactivation and enable policy route operations.
View method: Check `/var/log/mp-sdk.log` to see whether `policyRouteManagement` quickly calls deactivation and enable when acceleration is turned on.

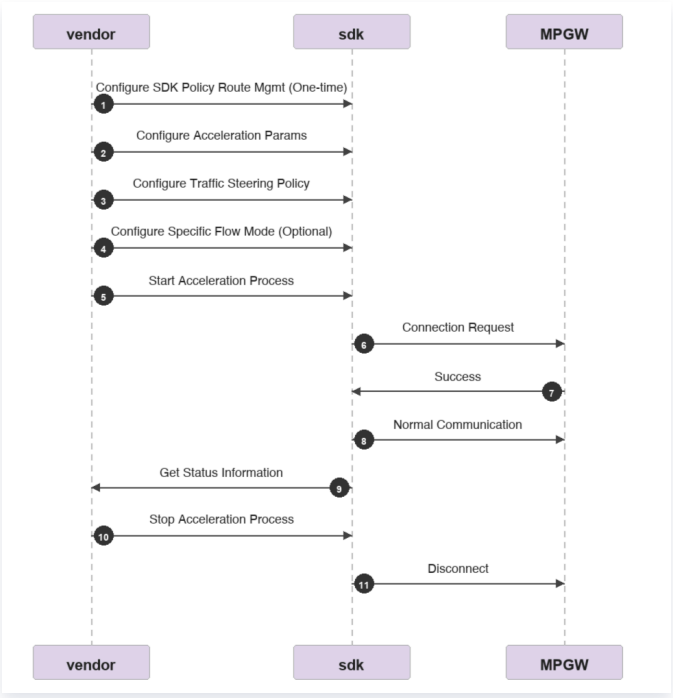
Windows SDK
API Overview

Last updated：2026-05-22 15:03:07

I. Deliverables Description

- linkboost.exe main program (subsequently abbreviated as SDK): Managed by customer app, mainly implements API layer and process management.
- linkboost-core.exe base package: Managed by SDK main program, mainly implements mpquic tunnel encapsulation and forwarding.
- helper kernel: mainly implements VPN block and allowlist filtering.

II. Access Process



vendor manufacturer integration with Cloud Jutong Windows SDK is as follows:

1. Policy routing management (Windows not supported).
2. Configure acceleration parameters (including dataKey, interfaces, acceleration mode).
3. Configure business traffic diversion policy to determine which traffic needs acceleration (socks mode integration not supported).
4. Configure flow-based multi-mode, with traffic segmentation routed to specific mode (selectable).
5. Start the acceleration process.
6. The SDK initiates a connection request to the MPGW.

7. MPGW returns a successful connection request.
8. The SDK and MPGW enter the normal communication stage.
9. Query SDK status (SDK synchronizes status with user).
10. Stop the acceleration process.
11. The SDK disconnects from the MPGW.

III. API Description

api Path	Description
/api/v2/client/mp-speeder	Configure acceleration parameters
/api/v2/client/mp-speeder/start	Start acceleration
/api/v2/client/mp-speeder/stop	Stop acceleration
/api/v2/client/mp-speeder/restart	Restart acceleration
/api/v2/client/mp-speeder	Query acceleration status
/api/v2/client/flowStatistics	Query acceleration traffic information
/api/v2/route/businessRoute	Configure traffic steering rules
/api/v2/client/multi-mode	Configure flow forwarding policy
/api/v2/diagnosis/log	Log Reporting
/api/v2/client/t2Statistics	Query drop-off point speed test information

Configure Acceleration Parameters

registerEnv	integer	Required. Value: 1.

Before starting up the acceleration process, the manufacturer needs to collect the required parameters, which must be carried as parameters when launching the sdk acceleration.

required parameter

1. Activate one of the following parameters:

- vendor and sn: vendor is the device vendor model, sn is the serial number provided by the device vendor.
- dataKey: device license key.
- appId and appSign: For app mode registration, obtain them from the console.
- appId, appSign, and gwId: For eo mode registration, obtain them from the console.

scheduleMode: The default working mode is rtc.

> Note:
> There are four types of methods to activate the Cloud Connect feature, and vendors can select based on their own circumstances.
> - vendor+sn: Activate using the device serial number. The device to be activated must be created in the Tencent Cloud console in advance.
> - dataKey: Activate with a license key. Contact Tencent Cloud sales or business to obtain it.
> - appId+appSign: Activate with appId. Please retrieve from the cloud console.
> - appId+appSign+gwId: For eo mode usage, toggle on the enableEoSch switch simultaneously.

Call the following API to configure acceleration parameters (only configure parameters without starting acceleration):

datakey mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
-d "
{\"serviceMode\":0,\"dataKey\":\"xxxxxxx\",\"scheduleMode\":\"rtc\",\"re
gisterEnv\":-2,\"tunInterfaceName\":\"mp_tun0\",\"t2Probe\":true}"
```

appId mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
-d "{\"serviceMode\": 0,\"appId\":\"app-
0eoo3vpctx\",\"appSign\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXZp
Y2VOYW1lIjoiamFja3ktdGVzdC0yMC0xIn0.TleKYrzBL1dyp2i8WTBgs8Y8UvBJQv-
n2A2BRMT1xuQ\",\"scheduleMode\":\"rtc\"}"
```

eo mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
```

```
-d "{\"serviceMode\": 0,\"gwId\":\"mpgw-n96i24ugbd\",\"appId\":\"zone-
359h792djt7h\",\"appSign\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZX
ZpY2VOYW1lIjoiamFja3ktdGVzdC0yMC0xIn0.RWGNKp_DXqnrg7SJ2yR5UV_MDs3_KWDccB
URQfBfiYg\",\"scheduleMode\":\"rtc\",\"enableEoSch\":true}"
```

This API imports a Client entity, defined as follows:

Name	Type	Description
dataKey	string	Required: The device dataKey when one of the four activation modes mentioned above is selected.
vendor	string	Required: The device vendor model when one of the four activation modes mentioned above is selected.
sn	string	Required: The unique device serial number provided by the device vendor when one of the four activation modes mentioned above is selected.
registerEnv	integer	Required. Value: 1.
appId	string	Optional: Used when the app mode + eo mode is selected from the four activation modes mentioned above. Fill in the corresponding zoneId in eo mode.
appSign	string	Optional: Used when the app mode + eo mode is selected from the four activation modes mentioned above.
enableEoSch	boolean	Optional: Must be true in eo mode.
gwId	string	Optional: Used in eo mode. Fill in the corresponding gatewayId.
lineType	[int]	Optional: Used when specified lines are enabled in eo mode. • 0: Direct connection. • 1:eo. • 2: Third-party.
enableObfuscated	string	Optional: Whether to enable the IP address obfuscation feature.
encryptSign	string	Optional: Required when IP address obfuscation is enabled. Used for IP address encryption and decryption. Must match the sign used for IP address encryption on the client side. Using appSign is recommended. For the encryption algorithm, see the appendix.
scheduleMode	string	Optional: The default acceleration mode for Windows, "rtc".

accGateway	string	Optional: Specify the acceleration gateway IP address. Multiple IP addresses should be separated by commas. If this parameter is not specified, the sdk will automatically connect to the nearest gateway. Otherwise, it will forcibly connect to the specified acceleration gateway, for example: "120.30.39.129".
gwPort	string	Optional: The gateway port. Default: "443".
UUID	string	Optional: The hardware fingerprint. If this parameter is not specified, the sdk will automatically generate it based on the hardware.
disableCrypto	integer	Optional: Allows the customer to choose whether to enable encryption when acceleration is started. Traffic encryption is enabled by default. Disabling encryption can reduce traffic consumption. • 0: Enable traffic encryption (default). • 1: Disable traffic encryption.
flowStatisticsInterval	integer	Optional: The frequency for link acceleration traffic statistics. Default: 3 seconds.
maxRttDisableAggregation	integer	Optional: The maximum latency for link aggregation. Default: 460 ms.
maxRttThreshold	integer	Optional: The latency for initiating link failure detection. Default: 460 ms.
minSwitchRTT	integer	Optional: The sensitivity for fast link switching. Default: 20 ms.
maxDelayUntilFailed	integer	Optional: The maximum latency for link switching. Default: 460 ms.
registerEnv	integer	**Required: The value must be 1.**
tunInterfaceName	string	Optional: The name of the tun interface. Default: mp_tun0.
t2Probe	boolean	Optional. • true: Enable t2 speed test. • false: Disable the t2 speed test.
authCode	string	Optional: Enabling traffic-free access.
accProcessName	string	Optional: The process name for the socks server. If you customize it, you must also rename the linkboost-core.exe file in the installation package accordingly, and the names must match.

		Example: "multipath-core.exe".

> **Note:**
> The above configuration requires **restarting** the acceleration process to make the configuration take effect.

Configuring a socks5 server

If you need to customize the speed-up socks5 server parameters, configure them via this API.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/socks5" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"enable\": true, \"port\": 12345, \"userName\": \"xxxxx\",
\"passWord\": \"xxxxx\"}"
```

> **Note:**
> The binding address of the socks5 server is "127.0.0.1" and is non-configurable.

Name	Type	Description
enable	bool	Required: Whether to customize the socks5 server.
port	int	Required: socks5 server port.
userName	string	Required: socks5 server username.
passWord	string	Required: socks5 server password.

If you need to query the socks5 server configuration, use this API for the query.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/socks5" ^
 -H "accept: application/json" ^
 -H "Content-Type: application/json"
```

Starting Acceleration

Call the following APIs to start acceleration. Then the SDK will create a virtual interface named mp_tun0, and business traffic will be guided to this virtual interface for acceleration based on the traffic diversion policy. (Before calling this API, please ensure the acceleration parameters are configured.)

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/start" -H
"accept: */*" -H "Content-Type: application/json"
```

Stop Acceleration

After stopping multiple networks, all traffic will not be accelerated, acceleration terminates, and the mp_tun0 interface is terminated.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/stop" -H
"accept: */*" -H "Content-Type: application/json"
```

Restarting Acceleration

Restart the acceleration process and reload the configuration.
This API is normally used to make the configuration take effect after modification.

```
 curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/restart" -
H "accept: */*" -H "Content-Type: application/json"
```

Querying Acceleration Status

This interface will return a Status entity containing SDK status info.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/mp-speeder" -H "accept:
*/*" -H "Content-Type: application/json"
```

Status definition:

Name	Type	Description
ready	boolean	Whether the acceleration process is started normally.
UUID	string	The hardware fingerprint of the device.
dataKey	string	The device dataKey.
swVersion	string	The SDK software version.
accGateway	string	The IP address of the acceleration gateway.
gatewayPort	string	The port of the acceleration gateway.

interfaces	[string]	The list of interfaces participating in multi-link aggregation.
scheduleMode	string	The default acceleration mode. "bonding", "redundant", "rtc".

Querying Acceleration Traffic Information

This API will return a Statistics entity, containing cumulative accelerated traffic consumption based on the network interface card and real-time speed of the acceleration channel. The statistical information is periodically updated at 10-second intervals.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/flowStatistics" -H
"accept: application/json" -H "all: true"
```

Statistics definition:

Name	Type	Description
interface	string	Name of the link NIC.
state	integer	Current link operational status: • -2: Link unavailable. • 60: Link temporarily disabled. • 100: Link normal.
totalReceivedBytes	integer	Cumulative accelerated traffic in the receive direction, in Bytes.
totalSendBytes	integer	Cumulative accelerated traffic in the send direction, in Bytes.
receivedRate	float	Receive rate of the current NIC acceleration channel, in bit/s.
sendRate	float	Send rate of the current NIC acceleration channel, in bit/s.
loss	float	Packet loss rate of the current NIC acceleration channel. The packet loss rate is expressed as a decimal, for example, 0.1 indicates a packet loss rate of 10%, and 1 indicates a packet loss rate of 100%.
rtt	integer	rtt of the current NIC acceleration channel, in ms. Note: -1 indicates that the current link is unavailable.

Querying Drop-Off Point Speed Test Info

When specifying T2 acceleration and a drop-off point, you can use this api to query speed test info for the drop-off point.

> **Note:**
> Speed test for drop-off points is supported only when manually specifying a drop-off point. When using auto to automatically select a drop-off point, speed test for drop-off points is not supported.

Name	Type	Description
area	string	Name of the drop-off point.
rttAccelerated	int	Latency from the sdk to the drop-off point.
rttDirect	int	Latency for the sdk directly connecting to the drop-off point.

Traffic Diversion Policy Configuration

The traffic diversion policy mainly specifies which traffic needs to introduce SDK processing, based on the allowlist feature. socks mode integration requires no attention.

```
curl -X POST "http://127.0.0.1:9801/api/v2/route/businessRoute" ^
 -H "accept: */*" ^
 -H "all: false" ^
 -H "whitelist: true" ^
 -H "Content-Type: application/json" ^
 -d "[{ \"appId\": \"12345\" }]"
```

Name	Type	Required	Description
ruleType	int	Yes	Traffic steering type, with the following values: • 1: Five-tuple rule; • 2: Route traffic based on domain. • 3: Route traffic based on gameId.
protocol	string	No	Protocol type, with the following values: • TCP • UDP

srcIP	string	No	Source IP address.
srcPorts	string	No	Source Port.
dstIP	string	No	Destination IP address.
dstPorts	string	No	Destination Port.
gameId	string	No	Game ID.
domain	string	No	Domain name.

Configuring Forwarding Policy

The forwarding policy mainly specifies how traffic should be forwarded, based on the game ID.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/multi-mode" ^
 -H "accept: application/json" ^
 -H "Content-Type: application/json" ^
 -d "{\"appId\": \"12345\",\"area\": \"hongkong\",\"speedMode\": 35}"
```

Name	Type	Description
appId	string	Optional: Accelerates traffic based on the specified appId rule. The socks mode does not need to be considered.
area	string	• auto: Automatically selects a drop-off point based on the region to which the dip belongs. • Frankfurt: Frankfurt • SiliconValley: United States – Silicon Valley • SaoPaulo: Brazil – Sao Paulo • Tokyo: Japan • Bangkok: Thailand (Southeast Asia Server) • Singapore: Singapore • Hong Kong: Hong Kong (China) • Seoul: Seoul, South Korea
speedMode	integer	• 0 – "DEFAULT": Reuses the default acceleration mode. • 1 – "DIRECT": No acceleration; traffic uses the system's default network card. • 2 – "bonding": Aggregation mode. • 3 – "rtc": Real-time audio and video mode.

		• 4 – "redundant": Multi-send selective-receive mode. • 25 – (Reserved configuration). • 35 – "T2 rtc": First performs rtc fast switching, then performs T2 full-path acceleration. • 45 – (Reserved configuration).

Mobile Rights and Interests Authentication API

Mobile rights and interests authentication. If necessary, obtain through the API to avoid traffic consumption during acceleration.

```
 curl -X POST
"http://127.0.0.1:9801/api/v2/client/enableCMCCRightsVerify" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"token\": \"xxxxx\"}"
```

Request Parameters:

Name	Type	Description
token	string	Required: Used to verify the token.
guid	string	Required: Used to verify the guid, and must be used together with the token.
salt	string	Required: The salt value, used for encryption.

Response Parameters:

Name	Type	Description
status	bool	Required: Whether the token is authenticated successfully.
authCode	string	Optional: Returns the traffic-free authentication code if the verification succeeds.
error	string	Optional: Returns an error via the error field when the verification fails.

Log Reporting

Selectable log reporting for problem localization. Log reporting has two methods:

1. Manual reporting: Execute a command once to report the multi-network logs saved locally on the device to the backend.
2. Automatic reporting: After setting the reporting interval and turning on the switch, logs will be reported to the backend periodically. You can turn off the switch at any time to pause upload.

> **Note:**
> Local logs occupy up to 50MB storage space. Stored logs will be deleted after reporting.

Configure log reporting parameters.

```
curl -X POST "http://127.0.0.1:9801/api/v2/diagnosis/log" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"logLevel\": \"debug\", \"upload\": false, \"uploadInterval\":
10}"
```

Parameters:

Name	Type	Description
logLevel	string	Optional: The program log level. Defaults to "info". Supports configuration of "info", "debug", and "warn".
autoUpload	boolean	Optional: The automatic upload switch. Defaults to false. When set to true, logs are automatically uploaded.
uploadInterval	integer	Optional: The automatic upload interval. Defaults to 10 minutes. Does not take effect when automatic upload is not enabled.

> **Note:**
> Modify log collection parameters. The SDK must be restarted to take effect.

Manual log reporting.
Single log reporting, recommended.

```
curl -X POST "http://127.0.0.1:9801/api/v2/diagnosis/log"
```

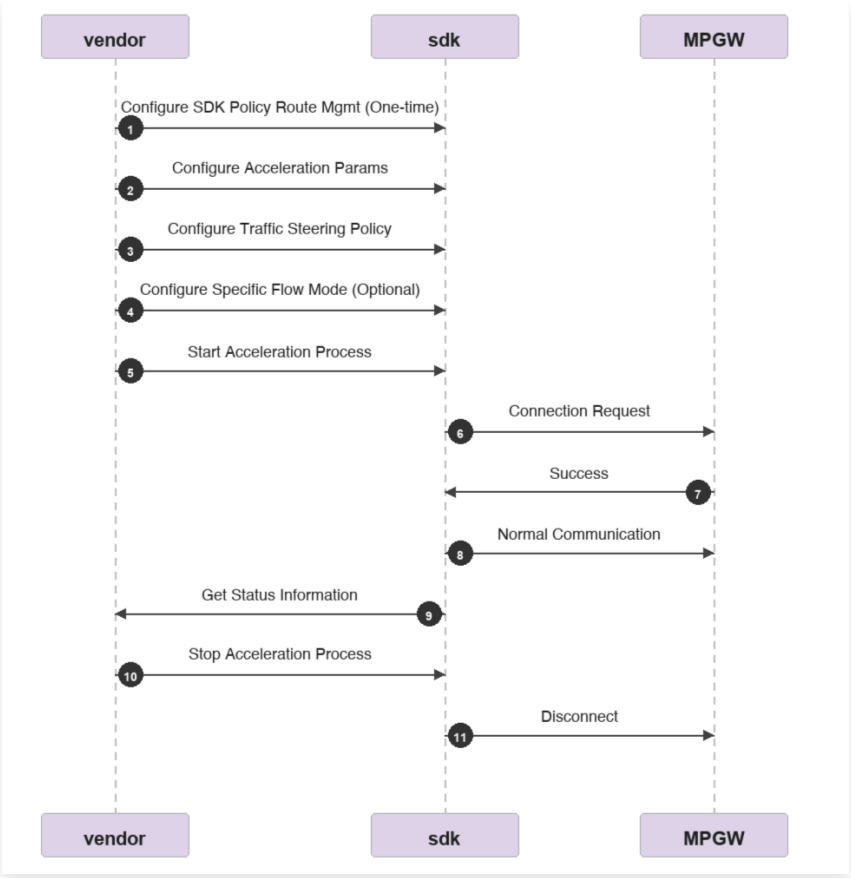
API Overview – dll

Last updated：2026-05-22 15:04:41

I. Deliverables Description

- The dll Dynamic Library linkboost.dll (abbreviation sdk): called by the customer app to mainly implement the api layer.
- The header file linkboost.h: called by the customer app.

II. Access Process



The steps for vendor manufacturer integration with Cloud Connect Windows SDK are as follows:

1. Policy routing management (Temporary not for Windows).
2. Configure acceleration parameters (including dataKey, interfaces, acceleration mode).
3. Configure business traffic diversion policy to determine which traffic needs acceleration (socks mode integration is unused).
4. Configure flow-based multi-mode to route segmented traffic through specific modes (optional).
5. Start the acceleration process.
6. The SDK initiates a connection request to the MPGW.

7. The MPGW returns that the connection request succeeded.
8. The SDK and MPGW enter the normal communication stage.
9. Query SDK status information (SDK synchronizes user status).
10. Stop the acceleration process.
11. The SDK disconnects from the MPGW.

III. API Description

1. SDK Management APIs

- Initialize sdk

```
// InitSDK Initialize SDK
// Parameter:
//   - disableLogEncrypt: want to close log encryption, 1 means
disabling log encryption, 0 means enable log encryption
// Return: 0 indicates successful initialization (reentrant), -1
indicates failure
int InitSDK(int disableLogEncrypt);
```

Note:

1. Please ensure the client program has the config and log directories in the working directory at runtime or has permission to create these two directories.
2. Please ensure the sdk has file read/write permissions in the config and log directories.

- Start up the sdk

```
// StartSDK Start the SDK service
// Return: 0 indicates successful startup (reentrant), -1 indicates
startup failure, -2 indicates the SDK is uninitialized
int StartSDK();
```

- Stop the sdk

```
// StopSDK Stop the SDK service
// Return: 0 indicates stopped successfully, -1 indicates the SDK is not
started
int StopSDK();
```

```
```

- View sdk status

```
// IsSDKRunning Check whether the SDK is running
// Return: 1 means running, 0 means not running
int IsSDKRunning();
```

2. Acceleration Relevant APIs (REST API)

api	Description
/api/v2/client/mp-speeder	Configure acceleration parameters
/api/v2/client/mp-speeder/start	Start acceleration
/api/v2/client/mp-speeder/stop	Stop acceleration
/api/v2/client/mp-speeder/restart	Restart acceleration
/api/v2/client/mp-speeder	Query acceleration status
/api/v2/client/flowStatistics	Query acceleration traffic information
/api/v2/client/multi-mode	Configure flow forwarding policy
/api/v2/diagnosis/log	Log Reporting
/api/v2/client/t2Statistics	Query drop-off point speed test information

- **Configure acceleration parameters**

Before starting the acceleration process, the manufacturer needs to collect the required parameters, which must be carried as parameters when launching the sdk acceleration.
required parameter

1. Activate one of the four parameters:

   vendor and sn: vendor is the device vendor model, and sn is the serial number provided by the device vendor.

   b. dataKey: device license key.

   c. appId and appSign: Register in app mode. Get from console.

   d. appId, appSign, and gwId: Register in eo mode. Get from console.

scheduleMode: default work mode: rtc.

Note: Four types of activation methods for Cloud Connect. The vendor can determine which to select based on their situation.

- vendor+sn: Activate by device serial number. Create the device to be activated in the Tencent Cloud console in advance.
- dataKey: Activate by license key. Contact Tencent Cloud sales and business to get it.
- appId+appSign: Activate by appId. Please retrieve from console.
- appId+appSign+gwId: Use in eo mode. Enable the enableEoSch switch at the same time.

Call the following APIs to configure acceleration parameters (only configure parameters without starting acceleration):

datakey mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
-d "
{\"serviceMode\":0,\"dataKey\":\"xxxxxxx\",\"scheduleMode\":\"rtc\",\"re
gisterEnv\":-2,\"tunInterfaceName\":\"mp_tun0\",\"t2Probe\":true}"
```

appId mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
-d "{\"serviceMode\": 0,\"appId\":\"app-
0eoo3vpctx\",\"appSign\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXZp
Y2VOYW1lIjoiamFja3ktdGVzdC0yMC0xIn0.TleKYrzBL1dyp2i8WTBgs8Y8UvBJQv-
n2A2BRMT1xuQ\",\"scheduleMode\":\"rtc\"}"
```

eo mode

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder" ^
-H "accept: */*" ^
-H "Content-Type: application/json" ^
-d "{\"serviceMode\": 0,\"gwId\":\"mpgw-n96i24ugbd\",\"appId\":\"zone-
359h792djt7h\",\"appSign\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZX
```

```
ZpY2VOYW1lIjoiamFja3ktdGVzdC0yMC0xIn0.RWGNKp_DXqnrg7SJ2yR5UV_MDs3_KWDccB
URQfBfiYg\",\"scheduleMode\":\"rtc\",\"enableEoSch\":true}"
```

This interface imports the Client entity, defined as follows:

Name	Type	Description
dataKey	string	Required: The device dataKey for the activation mode selected from the three options above.
vendor	string	Required: The device vendor and model for the activation mode selected from the three options above.
sn	string	Required: The unique device serial number provided by the device vendor when one of the three activation modes mentioned above is selected.
registerEnv	integer	Required. Value: 1.
appId	string	Optional: Used when the app mode + eo mode is selected from the three activation modes mentioned above. Fill in the corresponding zoneId in eo mode.
appSign	string	Optional: Used when the app mode + eo mode is selected from the three activation modes mentioned above.
enableEoSch	boolean	Optional: Must be true in eo mode.
gwId	string	Optional: Used in eo mode. Fill in the corresponding gatewayId.
lineType	[int]	Optional: Used when specified lines are enabled in eo mode. • 0: Direct connection. • 1:eo. • 2: Third-party.
enableObfuscated	string	Optional: Whether to enable the IP address obfuscation feature.
encryptSign	string	Optional: Required when IP address obfuscation is enabled. Used for encrypting and decrypting IP addresses. Must match the sign used for IP address encryption on the client side. appSign is recommended. For the encryption algorithm, see the appendix.
scheduleMode	string	Optional: The default acceleration mode for Windows, "rtc".

accGateway	string	Optional: Specify the acceleration gateway IP address. Multiple IP addresses should be separated by commas. If this parameter is not specified, the SDK will automatically connect to the nearest gateway. Otherwise, it will forcibly connect to the specified acceleration gateway, for example: "120.30.39.129".
gwPort	string	Optional. The gateway port. Default: "443".
UUID	string	Optional: The hardware fingerprint. If this parameter is not specified, the SDK will automatically generate it based on the hardware.
disableCrypto	integer	Optional: Allows the customer to choose whether to enable encryption when starting acceleration. Traffic encryption is enabled by default. Disabling encryption can reduce traffic consumption. • 0: Enable traffic encryption (default). • 1: Disable traffic encryption.
flowStatisticsInterval	integer	Optional: The frequency for link acceleration traffic statistics. Default: 3 seconds.
maxRttDisableAggregation	integer	Optional: The maximum latency for link aggregation. Default: 460 ms.
maxRttThreshold	integer	Optional: The latency for initiating link failure detection. Default: 460 ms.
minSwitchRTT	integer	Optional: The sensitivity for fast link switching. Default: 20 ms.
maxDelayUntilFailed	integer	Optional: The maximum latency for link switching. Default: 460 ms.
t2Probe	boolean	Optional: true: enables t2 speed measurement; false: disables t2 speed measurement.
authCode	string	Optional: Enabling traffic-free access.

Note:

- RegisterEnv is optional, for use during testing only. Not required for commercial devices.
- The above configuration requires a restart to speed up the process and make the configuration take effect.

- **Configure socks5 server**

If you need to customize socks5 server parameters, configure them via this API.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/socks5" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"enable\": true, \"port\": 12345, \"userName\": \"xxxxx\",
\"passWord\": \"xxxxx\"}"
```

> **Note:**
> The listening address of the socks5 server is "127.0.0.1" and non-configurable.

Name	Type	Description
enable	bool	Required: Whether to customize the socks5 server.
port	int	Required: socks5 server port.
userName	string	Required: socks5 server username.
passWord	string	Required: socks5 server password.

If necessary, query the socks5 server configuration via this API.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/socks5" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json"
```

- **Start acceleration**

Call the following API to start acceleration. The SDK will then create a virtual interface named mp_tun0. Business traffic is directed to this virtual interface for acceleration based on the traffic diversion policy. (Before calling this API, ensure the acceleration parameters are configured.)

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/start" -H
"accept: */*" -H "Content-Type: application/json"
```

- **Stop acceleration**

After stopping multi-net, all traffic will not be accelerated, acceleration terminates, and mp_tun0 interface is terminated.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/stop"     -
H "accept: */*" -H "Content-Type: application/json"
```

- **Restart acceleration**

Restart the acceleration process and reload configuration. This API is normally used to make the configuration take effect after modification.

```
 curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/restart" -
H "accept: */*" -H "Content-Type: application/json"
```

- **Query acceleration status**

This API will return a Status entity containing SDK status info.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/mp-speeder" -H "accept:
*/*" -H "Content-Type: application/json"
```

Status definition:

Name	Type	Description
ready	boolean	Whether the acceleration process is started normally.
UUID	string	The hardware fingerprint of the device.
dataKey	string	The device dataKey.
swVersion	string	The sdk software version.
accGateway	string	The IP address of the acceleration gateway.
gatewayPort	string	The port of the acceleration gateway.
interfaces	[string]	The list of interfaces participating in multi-link aggregation.
scheduleMode	string	The default acceleration mode. "bonding", "redundant", "rtc".

- **Query acceleration traffic information**

This interface will be returned Statistics entity, containing cumulative accelerated traffic consumption based on network interface card and acceleration channel real-time speed. Statistical information is periodically refreshed at 10-second intervals.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/flowStatistics" -H
"accept: application/json" -H "all: true"
```

Statistics definition:

Name	Type	Description
interface	string	Name of the link NIC.
state	integer	Current link operational status: • –2: Link unavailable. • 60: Link temporarily disabled. • 100: Link normal.
totalReceivedBytes	integer	Cumulative accelerated traffic in the receive direction, in Bytes.
totalSendBytes	integer	Cumulative accelerated traffic in the send direction, in Bytes.
receivedRate	float	Receive rate of the current NIC acceleration channel, in bit/s.
sendRate	float	Send rate of the current NIC acceleration channel, in bit/s.
loss	float	Packet loss rate of the current NIC acceleration channel. The packet loss rate is expressed as a decimal, for example, 0.1 indicates a packet loss rate of 10%, and 1 indicates a packet loss rate of 100%.
rtt	integer	rtt of the current NIC acceleration channel, in ms. Note: –1 indicates that the current link is unavailable.

- **Query drop-off point speed test info**

When T2 acceleration is specified and a drop-off point is specified, you can query the speed test info of the drop-off point via this api.

> **Note:**
> Speed test for drop-off points is supported only when manually specifying a drop-off point. When using auto to automatically select a drop-off point, speed test for drop-off points is not supported.

```
curl -X GET "http://127.0.0.1:9801/api/v2/client/t2Statistics" -H
"accept: application/json" -H "all: true"
```

Name	Type	Description
area	string	Name of the drop-off point.
rttAccelerated	int	Latency from the sdk to the drop-off point.
rttDirect	int	Latency for the sdk directly connecting to the drop-off point.

- **Forwarding policy configuration**

The forwarding policy mainly specifies how traffic should be forwarded.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/multi-mode" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"area\": \"hongkong\",\"speedMode\": 35}"
```

Name	Type	Description
appId	string	Optional: Accelerates traffic based on the specified appId rule.
area	string	• auto: Automatically selects a drop-off point based on the region to which the dip belongs. • Frankfurt: Frankfurt. • SiliconValley: United States – Silicon Valley. • SaoPaulo: Brazil – Sao Paulo. • Tokyo: Japan. • Bangkok: Thailand (Southeast Asia Server). • Singapore: Singapore. • HongKong: Hong Kong (China). • Seoul: Seoul, South Korea.
speedMode	integer	• 0 – "DEFAULT": Reuses the default acceleration mode. • 1 – "DIRECT": No acceleration; traffic uses the system's default network card. • 2 – "bonding": Aggregation mode. • 3 – "rtc": Real-time audio and video mode. • 4 – "redundant": Multi-send selective-receive mode. • 25 – (Reserved configuration). • 35 – "T2 rtc": First performs rtc fast switching, then performs T2 full-path acceleration.

		• 45 – (Reserved configuration).

- **Mobile rights and interests authentication interface**

Mobile rights and interests authentication. If necessary, obtain through the API to avoid traffic consumption during acceleration.

```
 curl -X POST
"http://127.0.0.1:9801/api/v2/client/enableCMCCRightsVerify" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"token\": \"xxxxx\"}"
```

Request Parameters:

Name	Type	Description
token	string	Required: Used to verify the token.
guid	string	Required: Used to verify the guid, and must be used together with the token.
salt	string	Required: The salt value, used for encryption.

Response Parameters:

Name	Type	Description
status	bool	Required: Whether the token is authenticated successfully.
authCode	string	Optional: Returns the traffic-free authentication code if the verification succeeds.
error	string	Optional: Returns an error via the error field when the verification fails.

IV. Diagnostics

This chapter is optional and mainly used for product operation and maintenance.

Log Reporting

Optional: Report logs to the backend for problem localization. Log reporting has two methods:

1. Manual reporting: Execute a command to report the multi-network logs saved locally on the device to the backend.

2. Automatic reporting: After setting the reporting interval and turning on the switch, logs will be reported to the backend periodically. You can turn off the switch at any time to pause upload.

> **Note:**
> Local logs occupy up to 50MB storage space. Stored logs will be deleted after reporting.

- **Configure log reporting parameters**

```
curl -X POST "http://127.0.0.1:9801/api/v2/diagnosis/log" ^
  -H "accept: application/json" ^
  -H "Content-Type: application/json" ^
  -d "{\"logLevel\": \"debug\", \"upload\": false, \"uploadInterval\":
10}"
```

Parameters:

Name	Type	Description
logLevel	string	Optional: The program log level. Defaults to "info". Supports configuration of "info", "debug", and "warn".
autoUpload	boolean	Optional: The automatic upload switch. Defaults to false. When set to true, logs are automatically uploaded.
uploadInterval	integer	Optional: The automatic upload interval. Defaults to 10 minutes. Does not take effect when automatic upload is not enabled.

> **Note:**
> Modify the data collection parameters. The SDK must be restarted to take effect.

- **Manual reporting parameters**

Report log. Recommended.

```
curl -X POST "http://127.0.0.1:9801/api/v2/diagnosis/log"
```

V. Access Example

- **SDK Startup Example**

```
using System;
```

```
using System.Runtime.InteropServices;
using System.Threading;

class Program
{
    [DllImport("linkboost.dll", CallingConvention =
CallingConvention.Cdecl)]
    static extern int InitSDK(int disableLogEncrypt);

    [DllImport("linkboost.dll", CallingConvention =
CallingConvention.Cdecl)]
    static extern int StartSDK();

    [DllImport("linkboost.dll", CallingConvention =
CallingConvention.Cdecl)]
    static extern int StopSDK();

    [DllImport("linkboost.dll", CallingConvention =
CallingConvention.Cdecl)]
    static extern int IsSDKRunning();

    private static bool keepRunning = true;

    static void Main()
    {
         // Register Ctrl+C event handling
        Console.CancelKeyPress += (sender, e) =>
        {
            e.Cancel = true; // prevent exit immediately
            keepRunning = false;
            Console.WriteLine("\nPausing program...");
        };

        try
        {
            Console.WriteLine("Initializing SDK...");
            // Initialize sdk, unencrypted logs
            InitSDK(1);

            Console.WriteLine("Starting SDK...");
```

```
            // Start the sdk
            StartSDK();

            Console.WriteLine($"SDK Running: {IsSDKRunning()}");
            Console.WriteLine("\nSDK started, program will continuously
run");
            Console.WriteLine("Press Ctrl+C to manually end the
program\n");

            // Continuously run until Ctrl+C is pressed
            while (keepRunning)
            {
                Thread.Sleep(1000); // Check once per second
            }
        }
        finally
        {
            Console.WriteLine("Pausing SDK...");
            Stop the sdk
            StopSDK();
            Console.WriteLine("SDK stopped, program exit");
        }
    }
}
```

- **socks5 acceleration**

1. **Call mobile rights and interests authentication (Option)**

For customers supporting mobile rights and interests with free data traffic, first call /api/v2/client/enableCMCCRightsVerify to obtain the free data traffic authentication code.
Parameter configuration:

- token: token provided by mobile.
- guid: be used with token.
- salt: salt value, used for encryption.

> **Note:**
> - If successful, record the return value authCode and fill it in step 2.
> - If it fails, respond with based on requirements.

2. **Configure acceleration parameters**

First, retrieve the appId and app key from the Tencent Cloud console, then refer to the source code in the appendix to calculate the appSign. Next, call /api/v2/client/mp-speeder (parameter interface for acceleration configuration).
Configure the parameters as follows:

- appId: Enter the application id obtained from the Tencent Cloud console.
- appSign: calculated based on the app key. For the algorithm, please refer to the appendix.
- scheduleMode:rtc.
- authCode (Option): fill in the authCode returned by the mobile rights and interests authentication interface in step one.
- interfaces: not required.
- accGateway and gwPort: not required.
- UUID: not required.
- disableCrypto: not required, keep the default.
- Other optional parameters: not required.

3. **Configure forwarding rules**

Call /api/v2/client/multi-mode (stream forwarding policy configuration interface) and specify the forwarding policy.
Specified parameters:

- area: auto, automatically select drop-off points.
- speedMode: select 35.

4. **Enable the socks5 service.**

Call /api/v2/client/socks5 (socks5 server configuration API) to enable socks5.
Specified parameters:

- enable:true.
- port: customer specified.
- userName: customer specified.
- passWord: customer specified.

5. **Enable acceleration**

After the acceleration interface is called, the SDK first connects to the nearest Tencent Cloud gateway.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/start" -H
"accept: */*" -H "Content-Type: application/json"
```

6. **Direct traffic to socks5.**

Business app traffic is directed to the specified socks5 port and forwarded based on configured rules.

Appendix: appSign Calculation Method

python

```
import jwt
from datetime import datetime, timedelta
from typing import Tuple


class JWTGenerator:
    EXPIRE_HOURS = 72  # Expiration time, in hours
    SECRET_KEY = "shCrVLKq/Hp6UdtVpw8lBqu6ZDcLwOABf2Az67e/pjI="  # APP
key

    @classmethod
    def generate_sign(cls, device_name: str) -> Tuple[str, int]:
        Generate JWT signature
        now = datetime.utcnow()
        expire_time = now + timedelta(hours=cls.EXPIRE_HOURS)

        payload = {
            "nbf": now,
            "exp": expire_time,
            "deviceName": device_name
        }

        token = jwt.encode(
            payload,
            cls.SECRET_KEY,
            algorithm="HS256"
        )

        return token, int(expire_time.timestamp())


if __name__ == "__main__":
    try:
        sign, expire_ts = JWTGenerator.generate_sign("tencent-test")
        print(f"Signature: {sign}")
        print(f"Expire Time: {expire_ts}")
```

```
    except Exception as e:
        print(f"Error generating token: {str(e)}")
```

golang

```
package main

import (
    "fmt"
    "github.com/dgrijalva/jwt-go/v4"
    "time"
)

func main() {
    sign, expireTime, err := GenerateSign("tencent-test")
    if err != nil {
        panic(err)
    }
    println(sign, expireTime)
}

// SignClaims JwtCustomClaims
type SignClaims struct {
    jwt.StandardClaims
    DeviceName string `json:"deviceName"`
}

var (
    ExpireTime = 72
// Expiration time, in hours
    SecretKey  = []byte("shCrVLKq/Hp6UdtVpw8lBqu6ZDcLwOABf2Az67e/pjI=")
// APP key
)

// GenerateSign generate signature
func GenerateSign(deviceName string) (string, int64, error) {
    expireTime := time.Now().Add(time.Duration(ExpireTime) * time.Hour)
    claims := &SignClaims{
```

```
        StandardClaims: jwt.StandardClaims{
            NotBefore: jwt.Now(),
            ExpiresAt: jwt.At(expireTime),
        },
        DeviceName: deviceName,
    }
    println(fmt.Sprintf("%#v", claims))

    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    sign, err := token.SignedString(SecretKey)
    if err != nil {
        return "", 0, err
    }
    return sign, expireTime.Unix(), nil
}
```

java

```
package org.example;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

import java.security.Key;
import java.util.Date;
import javax.crypto.spec.SecretKeySpec;

/**
 * generate sign
 */
public class SignGenerate {

    private static final long EXPIRE_TIME = 72 * 60 * 60 * 1000; //
Expiration time, in milliseconds
    private static final String SECRET_KEY =
"shCrVLKq/Hp6UdtVpw8lBqu6ZDcLwOABf2Az67e/pjI="; // APP key
```

```
    public static void main(String[] args) {
        String deviceName = "tencent-test";
        String sign = generateSign(deviceName);
        System.out.println(sign);
    }

    public static String generateSign(String deviceName) {
        Date now = new Date();
        Date expireDate = new Date(now.getTime() + EXPIRE_TIME);
        Key key = new SecretKeySpec(SECRET_KEY.getBytes(),
SignatureAlgorithm.HS256.getJcaName());
        return Jwts.builder()
                .claim("deviceName", deviceName)
                .notBefore(now)
                .expiration(expireDate)
                .signWith(key)
                .compact();
    }

}
```

Appendix: encryptSign Calculation Method

golang

```
package main

import (
    "crypto/sha256"
    "fmt"
    "golang.org/x/crypto/chacha20"
    "net"
)

func main(ip net.IP, psk []byte) ([]byte, error) {
    if ip.To4() == nil {
        return nil, fmt.Errorf("invalid ipv4 address")
    }
```

```
    key := sha256.Sum256(psk)
    encryptKey := key[:32]
    nonce := make([]byte, 12)
    cipher, err := chacha20.NewUnauthenticatedCipher(encryptKey, nonce)
    if err != nil {
        return nil, err
    }

    // In-place encryption (32bit exact output)
    ciphertext := make([]byte, net.IPv4len)
    cipher.XORKeyStream(ciphertext, ip.To4())
    return ciphertext, nil
}
```

Quick Connection

Last updated：2026-05-21 16:55:57

socks5 Integration

1. Calling Mobile Rights and Interests Authentication (Optional)

For customers with mobile rights and interests supported for data-free access, first call `/api/v2/client/enableCMCCRightsVerify` to get the authentication code for data-free access.
Parameter configuration:

- token: token provided by mobile
- guid: used with token.
- salt: salt value, used for encryption.

If successful, record the return value authCode and fill it in step 2.
If it fails, respond with a message based on requirements.

2. Configure Acceleration Parameters (Required)

First, retrieve the appId and app key from the Tencent Cloud console, then refer to the source code in the appendix to calculate the appSign.
Call `/api/v2/client/mp-speeder` (parameter interface for acceleration configuration).
Configure with the following parameters:

- appId: fill in the application id retrieved from the Tencent Cloud console.
- appSign: calculated based on the app key. For the algorithm, see the appendix.
- scheduleMode:rtc
- authCode (optional): fill in the `authCode` returned by the first step mobile rights and interests authentication API.
- Other optional parameters can be filled in as needed.

3. Configure Forwarding Rules (Optional, Enable Only When T2 Acceleration Is Required)

Call `/api/v2/client/multi-mode` (stream forwarding policy configuration API), specify the forwarding policy:
Specified parameters:

- area: Select the drop-off point in section T2 as shown in the API overview.
- speedMode: Select 35.

4. Enable socks5 Service (Required)

Call `/api/v2/client/socks5` (configure socks5 server interface) to enable `socks5 server`.
Specified parameters:

- enable: true.
- port: Customer specified.
- userName: Customer specified.
- passWord: Customer specified.

5. Enable Acceleration

After calling (enable acceleration API), the SDK will first connect to the nearby Tencent Cloud gateway.

```
curl -X POST "http://127.0.0.1:9801/api/v2/client/mp-speeder/start" -H
"accept: */*" -H "Content-Type: application/json"
```

6. Directing Traffic to socks5

Business app traffic import Step 4 specifies the socks5 port, and traffic will be forwarded according to the configured rules.

Appendix: appSign Calculation Method

Python

```
import jwt
from datetime import datetime, timedelta
from typing import Tuple


class JWTGenerator:
    @classmethod
    def generate_sign(cls, secret_key, device_name: str) -> Tuple[str,
int]:
        Generate JWT signature
        now = datetime.utcnow()

        payload = {
            "deviceName": device_name
        }

        token = jwt.encode(
```

```
            payload,
            secret_key,
            algorithm="HS256"
        )

        return token

if __name__ == "__main__":
    try:
        # replace with the APP key retrieved from console
        secret_key = "QGZzL0M6L3dpbmRvd3Mvd2luLmluaT9yYXc/aW1wb3J0Jg=="
        # Replace with device name, each device please use a different
device_name
        device_name = "test"

        sign = JWTGenerator.generate_sign(secret_key, device_name)
        print(f"Signature: {sign}")
    except Exception as e:
        print(f"Error generating token: {str(e)}")
```

golang

```
package main

import (
    "fmt"
    "github.com/golang-jwt/jwt/v4"
)

type JWTGenerator struct{}

// GenerateSign generates JWT signature
func (g *JWTGenerator) GenerateSign(secretKey string, deviceName string)
(string, error) {
    // Create payload
    claims := jwt.MapClaims{
        "deviceName": deviceName,
    }
```

```
    // Create token
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)

    // Use key signature
    signedToken, err := token.SignedString([]byte(secretKey))
    if err != nil {
        return "", err
    }

    return signedToken, nil
}

func main() {
    // replace with the APP key retrieved from console
    secretKey := "QGZzL0M6L3dpbmRvd3Mvd2luLmluaT9yYXc/aW1wb3J0Jg=="
    // Replace with device name, each device please use a different
device_name
    deviceName := "test"

    generator := &JWTGenerator{}

    sign, err := generator.GenerateSign(secretKey, deviceName)
    if err != nil {
        fmt.Printf("Error generating token: %v\n", err)
        return
    }

    fmt.Printf("Signature: %s\n", sign)
}
```

Java

```
<dependencies>
    <!-- JWT Library -->
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.11.5</version>
    </dependency>
```

```
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-impl</artifactId>
        <version>0.11.5</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-jackson</artifactId>
        <version>0.11.5</version>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

The above configuration shows a Java project adding dependency for the JWT tool library via Maven.

```
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import javax.crypto.SecretKey;
import java.nio.charset.StandardCharsets;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class JWTGenerator {

    /**
     * Generate JWT signature
     * @param secretKey Key string
     * @param deviceName Device name
     * @return JWT token
     */
    public static String generateSign(String secretKey, String
deviceName) {
        // Convert the key string to SecretKey
        SecretKey key =
Keys.hmacShaKeyFor(secretKey.getBytes(StandardCharsets.UTF_8));

        // Create payload
```

```
        Map<String, Object> claims = new HashMap<>();
        claims.put("deviceName", deviceName);

        // Build JWT
        return Jwts.builder()
                .setClaims(claims)
                .signWith(key)
                .compact();
    }

    public static void main(String[] args) {
        try {
            // replace with the APP key retrieved from console
            String secretKey =
"QGZzL0M6L3dpbmRvd3Mvd2luLmluaT9yYXc/aW1wb3J0Jg==";
            // Replace with device name, each device please use a
different device_name
            String deviceName = "test";

            String sign = generateSign(secretKey, deviceName);
            System.out.println("Signature: " + sign);
        } catch (Exception e) {
            System.out.println("Error generating token: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Appendix: encryptSign Calculation Method

golang

```
package main

import (
    "crypto/sha256"
    "fmt"
    "golang.org/x/crypto/chacha20"
```

```
    "net"
)

// share encryption and decryption
func encryptOrdecrypt(ip net.IP, psk []byte) ([]byte, error) {
    if ip.To4() == nil {
        return nil, fmt.Errorf("invalid ipv4 address")
    }

    key := sha256.Sum256(psk)
    encryptKey := key[:32]
    nonce := make([]byte, 12)
    cipher, err := chacha20.NewUnauthenticatedCipher(encryptKey, nonce)
    if err != nil {
        return nil, err
    }

    // in-place encryption (32bit exact output)
    ciphertext := make([]byte, net.IPv4len)
    cipher.XORKeyStream(ciphertext, ip.To4())
    return ciphertext, nil
}
```