

Linkus SDK Guide

Yeastar P-Series Appliance Edition

Version: 1.0

Date: 2024-04-22



Contents

Introduction.....	1
Yeastar Linkus SDK Introduction.....	1
Linkus SDK for Android.....	4
Overview.....	4
Release Notes.....	6
Integrate Linkus SDK for Android.....	6
Enable Linkus SDK and Bind Push Certificate.....	6
Integrate Linkus SDK for Android.....	8
Initialize Linkus SDK for Android.....	11
Obtain Login Signature for 'Linkus SDK for Android'.....	13
Use Linkus SDK for Android.....	18
Login and Connection.....	18
Call Features.....	22
Multi-party Call.....	26
Conference Call.....	28
Push Notifications.....	33
Call Detailed Record (CDR).....	34
Audio settings.....	35
Linkus SDK for iOS.....	37
Overview.....	37
Release Notes.....	38
Integrate Linkus SDK for iOS.....	38
Enable Linkus SDK and Bind APNs Certificate.....	38
Integrate Linkus SDK for iOS.....	40
Obtain Login Signature for 'Linkus SDK for iOS'.....	42
Use Linkus SDK for iOS.....	47
Configurations.....	47
Login and Logout.....	48
Call Features.....	49
Conference Call.....	50

Call Information.....	51
Complex Call Scenarios.....	53
Call Detailed Record (CDR).....	54
Linkus SDK for macOS.....	56
Overview.....	56
Release Notes.....	57
Integrate Linkus SDK for macOS.....	57
Enable Linkus SDK.....	57
Integrate Linkus SDK for macOS.....	58
Obtain Login Signature for 'Linkus SDK for macOS'.....	60
Use Linkus SDK for macOS.....	64
Configurations.....	64
Login and Logout.....	65
Call Features.....	66
Call Information.....	67
Complex Call Scenarios.....	69
Call Detailed Record (CDR).....	70
Linkus SDK for Windows.....	71
Overview.....	71
Release Notes.....	72
Enable Linkus SDK.....	72
Obtain Login Signature for 'Linkus SDK for Windows'.....	73
Linkus SDK for Windows Core.....	77
Integrate Linkus SDK for Windows Core.....	77
Use Linkus SDK for Windows Core.....	81
Linkus SDK for Windows UI.....	107
Integrate Linkus SDK for Windows UI.....	107
Linkus SDK for Web.....	113
Overview.....	113
Release Notes.....	114
Enable Linkus SDK.....	114
Obtain Login Signature for 'Linkus SDK for Web'.....	115
Linkus SDK for Web Core.....	119

Integrate Linkus SDK for Web Core.....	119
Use Linkus SDK for Web Core.....	123
Linkus SDK for Web UI.....	149
Integrate Linkus SDK for Web UI.....	149

Introduction

Yeastar Linkus SDK Introduction

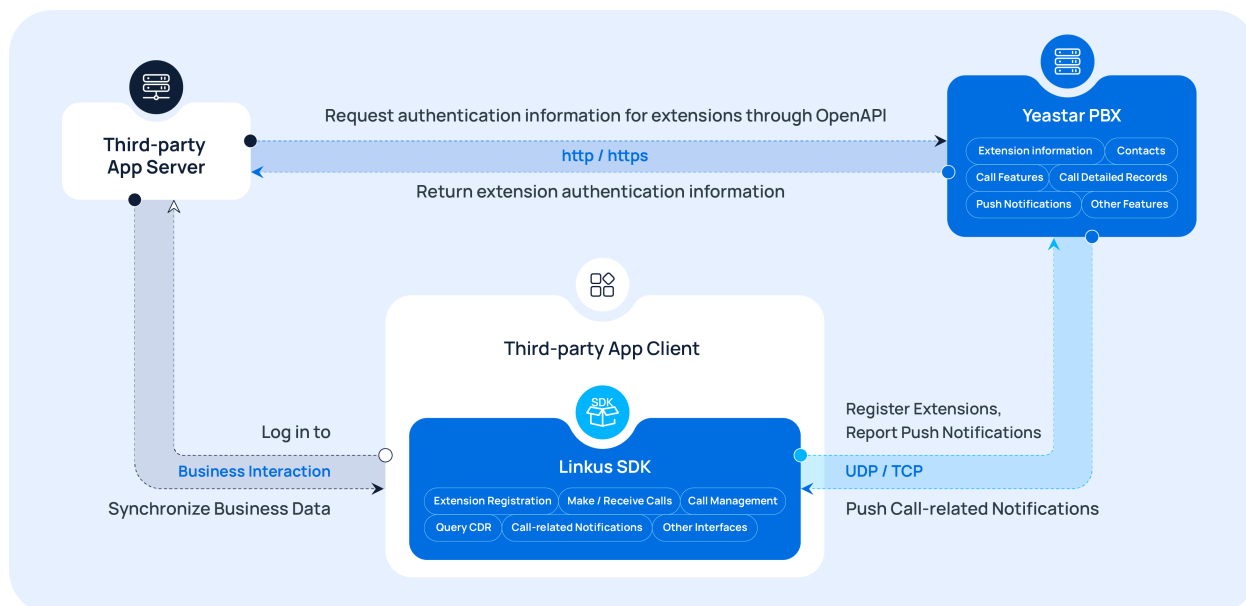
Yeastar P-Series PBX System provides Linkus Software Development Kit (SDK) for custom development, enabling developers to integrate the SDK into third-party projects to add calling, CDR, and more Linkus UC Clients' functionalities to third-party applications.

Requirements

Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Architecture



Supported platforms

Linkus SDK supports cross-platform development and can be integrated with existing applications, web pages, or platforms to streamline the development process and quickly implement call functionalities.

Platform	Resource Link
Android	<ul style="list-style-type: none"> • Key Features • Demo & Source Code • Linkus SDK Developer Guide
iOS	<ul style="list-style-type: none"> • Key Features • Demo & Source Code • Linkus SDK Developer Guide
macOS	<ul style="list-style-type: none"> • Key Features • Demo & Source Code • Linkus SDK Developer Guide
Windows	<ul style="list-style-type: none"> • Key Features • Demo & Source Code <ul style="list-style-type: none"> ◦ Core Call Functionalities ◦ UI Component • Linkus SDK Developer Guide
Web	<ul style="list-style-type: none"> • Key Features • Demo & Source Code <ul style="list-style-type: none"> ◦ Core Call Functionalities ◦ UI Component • Linkus SDK Developer Guide

Key features

Refer to the table below for the key features supported by Linkus SDK for different platforms.

		Android	iOS	macOS	Windows	Web
Basic Call Features	1:1 Voice Call	✓	✓	✓	✓	✓
	Make a Second Call	✗	✗	✓	✓	✓
	Call Transfer (Attended & Blind)	✓	✓	✓	✓	✓
	Call Hold & Resume	✓	✓	✓	✓	✓
	Call Mute	✓	✓	✓	✓	✓
	Call Recording	✓	✓	✓	✓	✓
	Call Waiting	✓	✓	✓	✓	✓
Advanced Call Features	Multi-party Call (Up to 5 parties)	✓	✓	✗	✗	✗


		Android	iOS	macOS	Windows	Web
	Conference Call (Up to 9 parties)	✓	✓	✗	✗	✗
	Audio Codec Selection	✓	✓	✗	✗	✗
	Audio Input Device Selection	✗	✗	✓	✓	✓
	Call Quality Monitoring	✓	✓	✓	✓	✓
CDR Query and Management	Query CDR Record	✓	✓	✓	✓	✓
	Delete CDR Record	✓	✓	✗	✗	✗
	Query the Number of Missed Calls	✓	✓	✗	✗	✗
System Push Notifications	Incoming Call Push Notification	✓	✓	✗	✗	✗
	Missed Call Push Notification	✓	✓	✗	✗	✗

Linkus SDK for Android

Linkus SDK for Android Overview

Yeastar P-Series PBX System supports Linkus SDK, enabling you to integrate calling, CDR, and more Linkus UC Clients' functionalities into 3rd-party Android applications. This topic describes the requirements, prerequisites, demo and source code, integration process, and features of 'Linkus SDK for Android'.

Requirements

Platform / Environment	Requirement
PBX Server	<ul style="list-style-type: none">• Firmware: 37.12.0.23 or later• Plan: Ultimate Plan (UP)
Development Environment	<ul style="list-style-type: none">• Java version: Java 11• Android version: 5.0 or later• Android Studio version: Arctic Fox or later• Gradle version: 6.5• Android Gradle Plugin version: 4.1.1 <div> Note: If you are using other versions of 'Gradle' and 'Gradle plugin', additional debugging may be required to ensure that the 'Linkus SDK for Android' works properly.</div>

Prerequisites

You have obtained the push certificate from the Android devices' operator, and noted down the required push certificate fields.



Note:

- Push certificate is used to ensure that you can receive incoming call notifications on your Android device after integrating Linkus SDK with your Android application.
- **Vivo Push** and **HONOR Push** are only available for version 37.14.0.24 or later.

The following table shows the required fields of push certificate for different push service platforms.

Push Service Platform	Required Field
Android Google Firebase Push	AppKey
Huawei Push	AppID, AppKey
Xiaomi Push	AppKey, AppPackageName
Vivo Push	App ID、 App Key、 App Secret
HONOR Push	App ID、 Client ID、 Client Secret

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for Android' to have an overview of the framework and workflow of 'Linkus SDK for Android'.

For more information, go to the [GitHub Repository of 'Linkus SDK for Android'](#).

Integration Process and Features

Integration Process

1. [Enable Linkus SDK and Bind Push Certificate](#)
2. [Integrate Linkus SDK for Android](#)
3. [Initialize Linkus SDK for Android](#)
4. [Obtain Login Signature for 'Linkus SDK for Android'](#)

Features

- [Login and Connection](#)
- [Call Features](#)
- [Multi-party Call](#)
- [Conference Call](#)
- [Push Notifications](#)
- [Call Detailed Record \(CDR\)](#)
- [Audio settings](#)

Release Notes - Linkus SDK for Android

Version 1.2.5

Release date: April 22, 2024

- Added compatibility with tablet devices, supporting simultaneous use of Linkus SDK on both tablets and mobile phones.
- Added support for push notification certificates: **Vivo Push & HONOR Push**.



Note:

This feature requires PBX version 37.14.0.24 or higher.

Version 1.1.2

Release date: October 11, 2023

- First release of **Linkus SDK for Android**. By integrating the SDK into your Android projects, you can quickly add calling, CDR, and more Linkus UC Clients' functionalities to your Android applications.

Integrate Linkus SDK for Android

Enable Linkus SDK and Bind Push Certificate

Before integrating 'Linkus SDK for Android' with your Android project, you need to enable Linkus SDK and bind push certificate on Yeastar P-Series PBX System, so that Android devices can receive incoming call notifications after the integration.

Requirements

Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Prerequisites

You have obtained the push certificate from the Android devices' operator, and noted down the required push certificate fields.

**Note:**

- Push certificate is used to ensure that you can receive incoming call notifications on your Android device after integrating Linkus SDK with your Android application.
- **Vivo Push** and **HONOR Push** are only available for version 37.14.0.24 or later.

The following table shows the required fields of push certificate for different push service platforms.

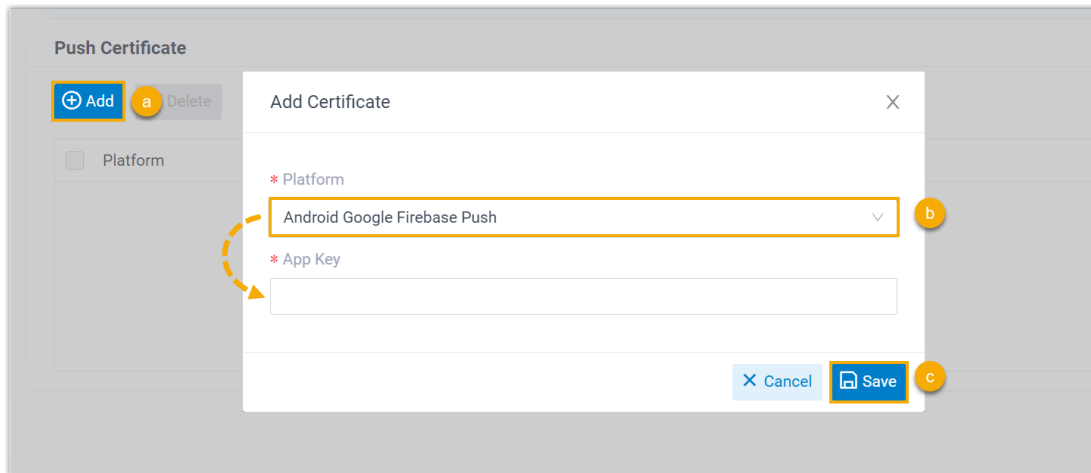
Push Service Platform	Required Field
Android Google Firebase Push	AppKey
Huawei Push	AppID, AppKey
Xiaomi Push	AppKey, AppPackageName
Vivo Push	App ID、 App Key、 App Secret
HONOR Push	App ID、 Client ID、 Client Secret

Procedure

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Enable **Linkus SDK**.



3. Bind the push certificate.



- a. In the **Push Certificate** section, click **Add**.
 - b. In the **Platform** drop-down list, select the platform of the push certificate, and then fill in the required information.
 - c. Click **Save**.
4. Click **Save**.

Result

You have enabled Linkus SDK and bound the push certificate, you can [Integrate Linkus SDK for Android](#).



Important:

After integrating 'Linkus SDK for Android', your Android application will use the bound certificate to send call-related push notifications to devices, and the Linkus's push certificate (Linkus Mobile Client's push notification) will no longer take effect.


Integrate Linkus SDK for Android

This topic describes how to integrate **Linkus SDK for Android** with your Android project by manually importing the Linkus SDK AAR file.

Requirements and Prerequisites

Requirements

Make sure that your development environment meets the following requirements:

Development Environment	Version Requirement
Java	Java 11
Android	5.0 or later
Android Studio	Arctic Fox or later
Gradle & Gradle Plugin	<ul style="list-style-type: none"> • Gradle: 6.5 • Android Gradle Plugin: 4.1.1 <div>  Note: If you are using other versions of 'Gradle' and 'Gradle Plugin', additional debugging may be required to ensure that the 'Linkus SDK for Android' works properly. </div>

Prerequisites

You have [enabled Linkus SDK and bound the push certificate](#).

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for Android' to have an overview of the framework and workflow of 'Linkus SDK for Android'.

For more information, go to the [GitHub Repository of 'Linkus SDK for Android'](#).

Procedure

1. Go to the [GitHub Repository of 'Linkus SDK for Android'](#), and download the latest version of **linkus-sdk-x.x.x.aar**.



Note:

x.x.x represents the version number of the Linkus SDK. Refer to [Release Notes - Linkus SDK for Android](#) to check the latest version number.

2. Copy the **linkus-sdk-x.x.x.aar** file to the directory **app > libs** of your Android project.
3. Import the AAR file of Linkus SDK to your project.



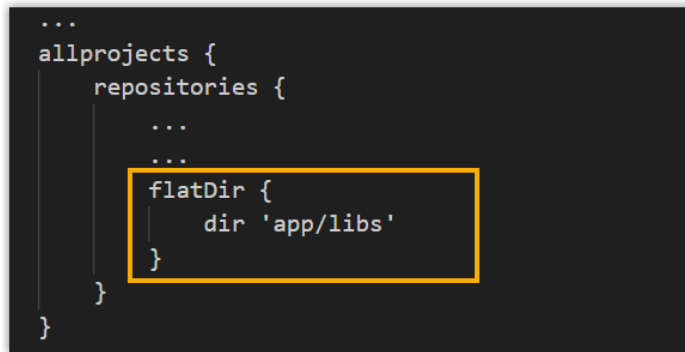
Note:



- The way to import AAR file varies depending on the development environment. Use the appropriate method based on your actual development environment.
- If your project files use Kotlin syntax (with a `.kts` extension), you need to convert the following codes into Kotlin syntax.

a. In the project root directory, open the **build.gradle** file, and add the following code in **allprojects > repositories**.

```
flatDir {
    dirs 'app/libs'
}
```



```
...
allprojects {
    repositories {
        ...
        flatDir {
            dir 'app/libs'
        }
    }
}
```

b. In the **app** directory, open the **build.gradle** file and add the following code in **dependencies**.

```
implementation(name: 'linkus-sdk-x.x.x', ext: 'aar')
```



Note:

linkus-sdk-x.x.x represents the file name (exclude the extension) of the Linkus SDK AAR file.

4. Synchronize the project.

Result

You have integrated 'Linkus SDK for Android' with your project, you can [Initialize Linkus SDK for Android](#).



Note:

You do NOT need to set the obfuscation configuration, as the obfuscation files have already been included in the AAR file.

Initialize Linkus SDK for Android

Before using the 'Linkus SDK for Android', you need to initialize it to launch the core services and components. 'Linkus SDK for Android' supports default initialization and custom initialization.

Prerequisites

You have [integrated 'Linkus SDK for Android' with your Android project](#).

Restriction

Initialization can only be performed once and must be performed in the main process.

Background information

Default configuration of 'Linkus SDK for Android'

The following table shows the general configuration items and their default values or settings for the 'Linkus SDK for Android'.

Configuration Item	Default Value / Setting
Audio Codec	iLBC
Automatic Gain Control (AGC)	Disabled
Echo Cancellation (EC)	Disabled
Audio Debugging	Disabled
Noise Cancellation (NC)	Enabled
Call Waiting	Enabled

Initialization options

'Linkus SDK for Android' supports default initialization and custom initialization. Choose either of the following ways to initialize 'Linkus SDK for Android'.

- **Default initialization:** Initialize 'Linkus SDK for Android' with the default configuration, no additional settings are required.

For more information, see [Initialize 'Linkus SDK for Android' with the default configuration](#).

- **Custom initialization:** Initialize 'Linkus SDK for Android' with the following custom configuration items.
 - Automatic Gain Control (AGC)

- Echo Cancellation (EC)
- Noise Cancellation (NC)
- Call Waiting
- Storage location for Linkus SDK data

For more information, see [Initialize 'Linkus SDK for Android' with custom configuration](#).

Initialize 'Linkus SDK for Android' with the default configuration

Method

Call the following method within `Application#onCreate` of the project to initialize 'Linkus SDK for Android' with the default configuration.

```
YlsBaseManager.getInstance().initYlsSDK(this, null);
```

Sample code

```
public class LinkusDemo extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        YlsBaseManager.getInstance().initYlsSDK(context: this, ylsInitConfig: null);
    }
}
```

Initialize 'Linkus SDK for Android' with custom configuration

Method

Call the following method within `Application#onCreate` of the project to customize the configuration items for 'Linkus SDK for Android' and initialize it.

```
YlsInitConfig config = new YlsInitConfig.Builder(projectPath)//
    Specify the storage location for Linkus SDK data (including logs)
        .supportCallWaiting(true)// Decide whether to enable Call Waiting. true: Enable; false: Disable
        .agc(true)// Decide whether to enable Automatic Gain Control (AGC). true: Enable; false: Disable
        .ec(true)// Decide whether to enable Echo Cancellation (EC). true: Enable; false: Disable
        .nc(true)// Decide whether to enable Noise Cancellation (NC). true: Enable; false: Disable
```



```

        .key("")// Optional. Specify the password for accessing
        database
        .build();// Initialize 'Linkus SDK for Android' with the
        configurations above
        YlsBaseManager.getInstance().initYlsSDK(this, config);

```

Sample code

```

public class LinkusDemo extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        String projectPath = YlsBaseManager.getInstance().getProjectPath(context: this);
        YlsInitConfig config = new YlsInitConfig.Builder(projectPath)
            .supportCallWaiting(b: false)
            .agc(b: true)
            .key(s: "12345")
            .build();
        YlsBaseManager.getInstance().initYlsSDK(context: this, config);
    }
}

```

What to do next

Request the SDK login signature from PBX server for authentication and login to 'Linkus SDK for Android'.

For more information, see [Obtain Login Signature for 'Linkus SDK for Android'](#).

Related information

[Initialize the conference call feature](#)

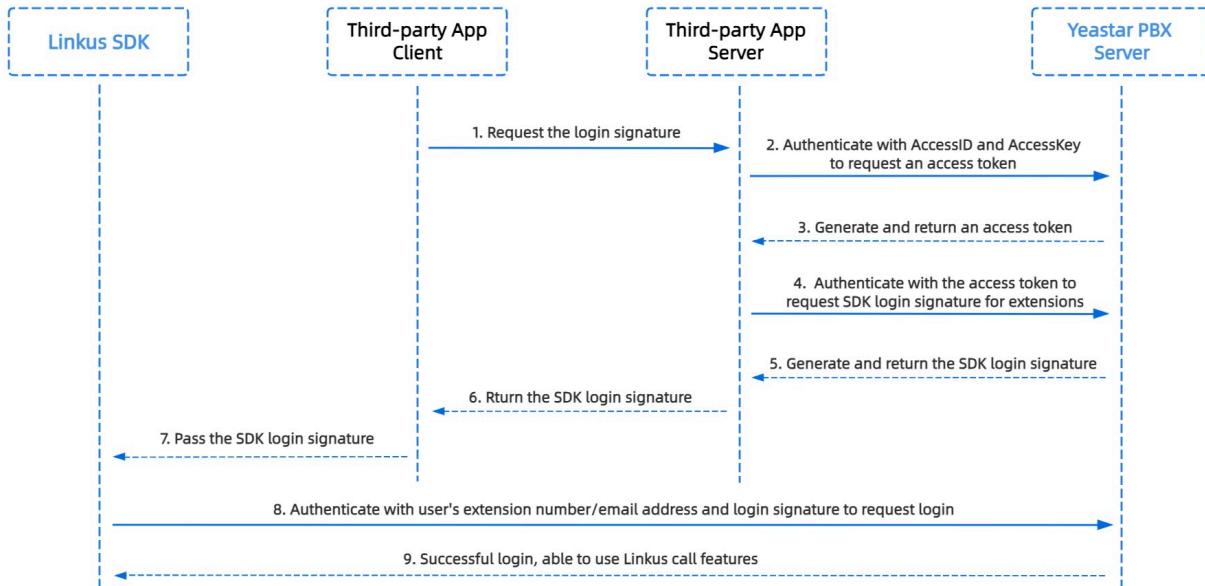
Obtain Login Signature for 'Linkus SDK for Android'

When logging into Linkus SDK, users need to use the SDK login signature for authentication instead of their login password. This topic describes how to request users' Linkus SDK login signatures from the PBX server via OpenAPI.

Prerequisites

- You have [enabled Linkus SDK and bound the push certificate](#).
- You have [integrated and initialized 'Linkus SDK for Android'](#).

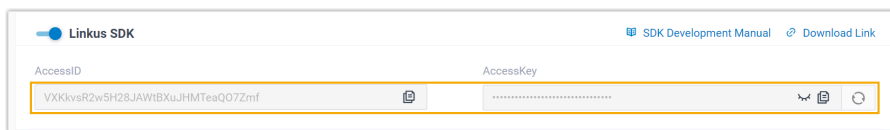
Linkus SDK login process



Step 1. Obtain the 'AccessID' and 'AccessKey' for Linkus SDK on PBX

Obtain the 'AccessID' and 'AccessKey' for Linkus SDK from Yeastar P-Series PBX System, which will be used for the third-party application to authenticate and connect with the PBX server.

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Note down the **AccessID** and **AccessKey**.



Step 2. Request an access token from PBX

On the third-party application server, use the 'AccessID' and 'AccessKey' to request an access token from the PBX via OpenAPI. Access token is used to verify an authenticated API call, enabling you to request Linkus SDK login signatures for extensions.

Request URL

```
POST {base_url}/openapi/v1.0/get_token
```



Note:



To learn about the API request structure, see [Request Structure](#).

Request Parameters

Parameter	Required	Type	Description
username	Yes	String	User name. Use the AccessID of Linkus SDK as the username.
password	Yes	String	Password. Use the AccessKey of Linkus SDK as the password.

Request example

```
POST /openapi/v1.0/get_token
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "VXKkvsR2w5H28JAWtBXuJHMTeaQO7Zmf",
  "password": "Yq6yVsBceOZLhnuaGeMUG4U4qXXXXXXX"
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> 0: Succeed. Non-zero value: Failed.
errmsg	String	Returned message. <ul style="list-style-type: none"> SUCCESS: Succeed. FAILURE: Failed.
access_token_expire_time	Integer	Access token expire time. (Unit : second)
access_token	String	Credential of calling API interfaces. All requests to call API interfaces must carry an access token.
refresh_token_expire_time	Integer	Refresh token expire time. (Unit : second)
refresh_token	String	Refresh token.

Parameter	Type	Description
		<code>refresh_token</code> can be used to obtain new <code>access_token</code> and <code>refresh_token</code> .

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "access_token_expire_time": 1800,
  "access_token": "EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS",
  "refresh_token_expire_time": 86400,
  "refresh_token": "SCduGecwbG9jIusiS8FxFUVn3kf0Q9R8"
}
```

Step 3. Request login signatures for extensions

Use the access token to request login signatures for extensions from PBX server via Open-API.

Request URL

```
POST /openapi/v1.0/sign/create?access_token={access_token}
```

Request Parameters

Parameter	Required	Type	Description
username	Yes	string	Extension number or email address.
sign_type	Yes	string	Login signature type. Permitted value: <code>sdk</code>
expire_time	No	Integer	Expiration timestamp of the login signature. (Unit: second) <code>0</code> indicates no limitation on the validity duration of the login signature.

Request example

```
POST /openapi/v1.0/sign/create?access_token=EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "1000",
```

```

"sign_type": "sdk",
  "expire_time": 0
}

```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> • 0: Succeed. • Non-zero value: Failed
errmsg	String	Returned message. <ul style="list-style-type: none"> • SUCCESS: Succeed. • FAILURE: Failed.
data	Array < Ext_Sign >	Linkus SDK login signatures for extensions.

Ext_Sign

Parameter	Type	Description
sign	String	Extension's Linkus SDK login signature.

Response example

```

HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "data": {
    "sign": "ueb3rfcpsis8FxFUZMpZA086mbrKmVn3kf0Q9R8"
  }
}

```

Result

The third-party application server has obtained the Linkus SDK login signature for extension users, the login signature will be automatically returned to the third-party application client, and passed to Linkus SDK.

What to do next

Use the SDK login signature for authentication and [log in to 'Linkus SDK for Android'](#).

Use Linkus SDK for Android

Login and Connection

This topic introduces the functionalities and implementation methods related to login and connection of 'Linkus SDK for Android'.

Supported features

Feature	Description
Initial login (manual login)	Initial login after initializing Linkus SDK. The initial login requires login credentials and the address of PBX server to establish a connection with PBX and authenticate users.
Automatic login	After the initial login, Linkus SDK will cache the login credentials and PBX server information on the local device, enabling users to directly use it on subsequent logins without having to enter it again.
Query user's login status	Query if the user has logged in to Linkus SDK.
Query the connection status with PBX server	Query if Linkus SDK is connected with the PBX server.
SDK notification callbacks	Listen for and handle events such as user account logout, successful reconnection with PBX, CDR changes, etc.

Initial login (manual login)

This method is used when a user logs in to the Linkus SDK for the first time. This method requires the user to enter their extension numbers or email addresses and the login signature as the login credential, as well as the IP address of the PBX server to establish a connection with PBX.

```
/**
 * manual login
 *
 * @param context
 * @param userName: User's extension number or email address
 * @param passWord: Login signature for Linkus SDK
```

```

* @param localeIp: PBX's local IP address
* @param localePort: Port of PBX's local IP address
* @param remoteIp: PBX's public IP address
* @param remotePort: Port of PBX's public IP address
* Depending on users' actual usage environment, at least one set of
values, either 'localeIp, localePort' or 'remoteIp, remotePort' must be
filled in
* @param requestCallback: Login result callback
* @return: Return value of the login interface, indicating the reason for
login failure or abnormality. For more information, refer to the table
below
*/
public void loginBlock(Context context, String userName, String passWord,
    String localeIp, int localePort,
        String remoteIp, int remotePort, RequestCallback<Boolean>
requestCallback)
//Sample code for manual login
    YlsLoginManager.getInstance().loginBlock(this, userName, password,
    localeIp,
        localePortI, remoteIp, remotePortI, new RequestCallback<>() {

@Override // Login success callback
public void onSuccess(Boolean result) {
    closeProgressDialog();
    startActivity(new Intent(LoginActivity.this,
    DialPadActivity.class));
}

@Override// Login failure callback
public void onFailed(int code) {
    closeProgressDialog();
    Toast.makeText(LoginActivity.this, R.string.login_tip_login_failed,
    Toast.LENGTH_LONG).show();
}

@Override// Login exception callback
public void onException(Throwable exception) {
    closeProgressDialog();
    Toast.makeText(LoginActivity.this, R.string.login_tip_login_failed,
    Toast.LENGTH_LONG).show();
}
});

```

Login interface return value description

Return Value	Description
1	Failed to connect to the PBX server.
-5	No response to the login request.
403	Invalid user name or login signature.
405	Linkus UC client is disabled.
407	This account has been locked.
416	Access to the requested IP address is prohibited (Allowed Country / Region IP Access Protection is enabled).

Automatic login

After the initial login, Linkus SDK will cache login information. This method will automatically read the cached information for login, eliminating the need for users to enter it again.

```
int networkType = NetWorkUtil.getNetWorkType(this);
YlsLoginManager.getInstance().cacheLogin(networkType);
```

Query user's login status

```
/**
 * Query if a user has logged in to Linkus SDK
 * @return Return value, which indicates the user's login status. true:
 * Logged in; false: Not logged in
 */
public boolean isLoginEd()
// Example
    boolean isLoginEd = YlsLoginManager.getInstance().isLoginEd();
```

Query the connection status with PBX server

```
/**
 * Query if Linkus SDK is connected with PBX server
 * @return Return value, which indicates the connection status. true:
 * Connected; false: Not connected
 */
public synchronized boolean isConnected()
// Example
    YlsLoginManager.getInstance().isConnected();
```


SDK notification callbacks

The SDK notification callback can be used to listen for and handle events such as user account logout, reconnecting to the PBX, and changes of CDR.

```
YlsBaseManager.getInstance().setSdkCallback(new SdkCallback() {
// User account logout callback. Refer to the table below for account
  logout event types
@Override
public void onLogout(int type) {
    context.startActivity(new Intent(context, LoginActivity.class));
}

// Linus SDK reconnecting with PBX success callback
@Override
public void onReconnectSuccess() {}

// CDR records change callback
@Override
public void onCdrChange(int syncResult) {
    EventBus.getDefault().post(new CallLogChangeEvent(syncResult));
}
});
```

Account logout event types

Event	Code	Description
SdkEventCode.EVENT_USER_RELOGIN	1005	The account has logged in elsewhere.
SdkEventCode.P_EVENT_DISABLE_LINKUS_APP	20014	Linkus Mobile Client is disabled.
SdkEventCode.EVENT_LOGIN_LOCKED	1009	This account has been locked.
SdkEventCode.EVENT_LOGIN_INFO_ILLEGAL	1010	Invalid cached login information.
SdkEventCode.EVENT_CACHE_LOGIN_USER_NOTFOUND	1011	No available cached login information.
SdkEventCode.P_EVENT_LOGIN_MODE_CHANGE	20008	Login mode (manual login or automatic login) changes.
SdkEventCode.P_EVENT_COUNTRY_IP_LIMIT	20083	Access to the requested IP address is prohibited (Allowed Country / Region)

Event	Code	Description
		IP Access Protection is enabled).
SdkEventCode.P_EVENT_LICENSE_EXPIRE	20093	PBX plan is NOT Ultimate Plan (UP) .
SdkEventCode.P_EVENT_SDK_STATUS_CHANGE	20153	Linkus SDK is disabled on PBX.
SdkEventCode.P_EVENT_SDK_ACCESSKEY_CHANGE	20154	The AccessKey of Linkus SDK has been refreshed on PBX.

Call Features

This topic introduces the functionalities and implementation methods related to call features of 'Linkus SDK for Android'.

Make a call

```
/**
 *
 * @param callNumber: Callee number
 * @param netWorkAvailable: The network availability. true: Available; false: Unavailable
 * @return
 */
public void makeNewCall(String callNumber, boolean netWorkAvailable)
// Call method
    YlsCallManager.getInstance().makeNewCall(number, netWorkAvailable);
```

Answer a call

```
YlsCallManager.getInstance().answerCall(callId);
```

Reject a call

```
YlsCallManager.getInstance().answerBusy(context, callId);
```

Hang up a call

```
YlsCallManager.getInstance().hangUpCall(context, callId);
```

Hold a call

```
YlsCallManager.getInstance().holdCall(inCallVo);
```

Resume a call

```
YlsCallManager.getInstance().unHoldCall(getContext(), inCallVo);
```

Attended transfer

Perform an attended transfer

```
YlsCallManager.getInstance().makeTransferCall(context, calleeName, number, trunkName, route, object);
```

Confirm an attended transfer

```
YlsCallManager.getInstance().confirmTransfer(App.getInstance().getContext());
```

Blind transfer

```
YlsCallManager.getInstance().blindTransferCall(context, callOutNumber);
```

Mute or unmute a call

```
/**
 * @param inCallVo
 */
public void mute(InCallVo inCallVo)
```

Record a call

```
/**
 *
 * @param vo
 * @return
 */
public int record(InCallVo vo)
```

Send DTMF

```
/**
 *
```

```

* @param callId
* @param recordCode
* @return
*/
public int sendDtmf(int callId, String recordCode)

```

Query call quality

```

CallQualityVo callQualityVo =
    YlsCallManager.getInstance().getCallQuality();

```

SDK notification callbacks

```

YlsCallManager.getInstance().setCallStateCallback(new CallStateCallback() {
// Call status change callback
@Override
public void onCallStateChange(CallStateVo callStateVo) {
    EventBus.getDefault().post(new CallStateEvent(callStateVo));
}

// Call quality change callback
@Override
public void onNetWorkLevelChange(int callId, int networkLevel) {
    EventBus.getDefault().postSticky(new NetWorkLevelEvent(callId,
        networkLevel));
}

// Network connection change callback
@Override
public void onConnectChange() {
    EventBus.getDefault().postSticky(new ConnectionChangeEvent());
}

// Call recording status change callback
@Override
public void onRecordChange(boolean isRecording) {
    EventBus.getDefault().post(new RecordEvent(isRecording));
}

});

```

User Interface (UI) callback

```

YlsCallManager.getInstance().setActionCallback(new ActionCallback() {

```

```

// Call end callback
@Override
public void onFinishCall() {
    finishAllCall(context);
}

// Incoming call pop-up callback
@Override
public void onNewCall() {
    jump2CallActivity(context);
}

// Call waiting callback
@Override
public void onCallWaiting() {
    EventBus.getDefault().post(new CallWaitingEvent());
    SoundManager.getInstance().startPlay(context,
    YlsConstant.SOUND_CALL_WAITING_TYPE);
}

// Missed call callback
@Override
public void onMissCallClick() {

}

// Callback to stop the foreground service after the call ends (For Android
// 11 and above, calls running in the background require a foreground
// service)
@Override
public void onStopMicroPhoneService() {

}

// Audio routing popup disappearing callback
@Override
public void onDismissPopupView() {
    dismissPopupView();
}

// Audio routing change callback
@Override
public void onNotifyAudioChange() {
    notifyAudioChange();
}

```

```

    }
  });

```

Multi-party Call

This topic introduces the functionalities and implementation methods related to multi-party call of 'Linkus SDK for Android'.

Make a multi-party call

```

/**
 * Make a multi-party call
 */
public void makeMultipartyCall(String number, String trunkName, String
    route, Activity activity, Object obj)

```

Remove a member

```

/**
 * Remove a member from the current multi-party call
 *
 * @return
 */
public void hangUpSingleCall(Context context, int callId)

```

Mute or unmute a member

```

/**
 * Mute or unmute a member in a multi-party call
 */
public void muteSingleMember(InCallVo inCallVo)

```

Query information related to multi-party calls

```

/**
 * Retrieve the array of callID for all calls in a multi-party call.
 */
public int[] getCallIdArrays()

/**
 * Retrieve the array of callID for all the muted calls in a multi-party
    call.
 */

```

```

public int[] getMuteArrays()

/**
 * Retrieve the array of callID for all the held calls in a multi-party
 * call.
 */
public int[] getHoldArrays()

/**
 * Query whether the current call is a multi-party call.
 *
 * @return
 */
public boolean isInMultipartyCall()

/**
 * Set whether the current call is a multi-party call.
 *
 * @param isInMultipartyCall
 */
public void setInMultipartyCall(boolean isInMultipartyCall)

/**
 * Query whether all calls in a multi-party call are on hold.
 *
 * @return
 */
public boolean isInMultipartyHold()

/**
 * Retrieve the start time of a held call in a multi-party call.
 *
 * @return
 */
public long getMultipartyHoldStartTime()

/**
 * Query whether all calls in a multi-party call are muted.
 *
 * @return
 */
public boolean isMultipartyMute()

/**
 * Whether to mute all the members in a multi-party call.

```

```

*
* @param multipartyMute
*/
public void setMultipartyMute(boolean multipartyMute)

/**
 * Retrieve the start time of a multi-party call.
 *
 * @return
 */
public long getMultipartyCallStartTime()

/**
 * Query whether the multi-party call limit (4 calls) has been reached.
 *
 * @return
 */
public boolean reachMultiPartyCallsLimit()

/**
 * Query whether a multi-party call is being recorded.
 *
 * @return
 */
public boolean isMultipartyCallRecord(LinkedList<InCallVo> list)

/**
 * Query whether recording is available for a multi-party call.
 *
 * @return
 */
public boolean isMultiPartyCallRecordAble()

/**
 * Query whether recording is disabled for a multi-party call.
 *
 * @return
 */
public boolean isMultiPartyCallAlwaysRecordDisable()

```

Conference Call

This topic introduces the functionalities and implementation methods related to conference call of 'Linkus SDK for Android'.

Initialize the conference call feature

To use the conference call feature, you need to initialize it in the main process of your project's `Application` class by calling the following method.

```
YlsConferenceManager.getInstance().setConferenceCallback(context, new
    ConferenceCallback() {
@Override
public void onConferenceException(ConferenceVo conferenceVo) {
    // Callback for exceptional conference call
    EventBus.getDefault().postSticky(new
        ConferenceExceptionEvent(conferenceVo));
}

@Override
public void onConferenceStatusChange(String conferenceId, String number,
    int status) {
    // Callback for status of the conference call members
    EventBus.getDefault().post(new ConferenceStatusEvent(conferenceId,
        number, status));
}
});
```

Make a conference call

```
/**
 *
 * @param context
 * @param conferenceName: Conference name, which can NOT contain the following
    characters: ! $ ( ) / # ; , [ ] " = < > & \ ' `` ^ % @ { } | space, and
    can NOT exceed 63 characters
 * @param memberArray: Array of conference call members
 * @param requestCallback
 */
public void startConference(Context context, String conferenceName,
    String[] memberArray, RequestCallback requestCallback)
```

Manage conference call members

```
/**
 * When there are fewer than 9 members in a conference call, call this
    method to add the "Add Member" option
 *
 * @param memberList
 */
```

```

public void addNullMember(List<ConferenceMemberVo> memberList)

/**
 * This method is only available for the Host of a conference call
 * When there are fewer than 9 members in a conference call, call this
 * method to add the "Add Member" option
 * When there are more than 2 members in a conference call, call this
 * method to add the "Delete Member" option
 * @param memberList
 */
public void addNullMemberByAdmin(List<ConferenceMemberVo> memberList)

/**
 * When there are 9 members in a conference call, call this method to
 * remove the "Add Member" option
 *
 * @param memberList
 */
public void removeNullMember(List<ConferenceMemberVo> memberList)

```

Manage conference calls

```

/**
 *
 * During a conference call, the host mutes or unmutes all the members
 * @param conferenceId: ID of the conference call
 * @param member: Conference call members
 * @param isMute: Whether to mute all the members
 */
public ResultVo muteAllConferenceMemberBlock(String conferenceId, String
member, boolean isMute)

/**
 *
 * During a conference call, the host mutes or unmutes specific member(s)
 * @param conferenceId: ID of the conference call
 * @param number: ID of the specific member(s) to be muted or unmuted
 * @param isMute: Whether to mute the specific member(s)
 */
public ResultVo muteConferenceMemberBlock(String conferenceId, String
number, boolean isMute)

/**
 *
 * Host removes a specific member from the conference call

```

```

    * @param conferenceId: ID of the conference call
    * @param number: ID of the specific member to be removed
    */
public ResultVo kickConferenceMemberBlock(String conferenceId, String
    number)

/**
 *
 * Invite new members to the conference call
 * @param conferenceId: ID of the conference call
 * @param number: Number of the specific member to be invited
 */
public ResultVo inviteConferenceMemberBlock(String conferenceId, String
    number)

/**
 *
 * Re-invite the members who were previously invited but have not yet
    joined the conference call
 * @param conferenceId: ID of the conference call
 * @param number: Number of the specific member to be invited
 */
public ResultVo reInviteConferenceMemberBlock(String conferenceId, String
    number)

/**
 *
 * End the conference call
 * @param context
 * @param callId
 * @param conferenceVo
 * @param callback
 * @return
 */
public void endConferenceBlock(Context context, int callId, ConferenceVo
    conferenceVo, RequestCallback callback)

/**
 *
 * Reconnect to the conference call that was interrupted unexpectedly.
 * @param context
 * @param conferenceId: ID of the exceptional conference call
 * @param member: Conference call members
 * @return
 */

```

```
public ResultVo returnConferenceBlock(Context context, String conferenceId,
    String member)
```

Query conference call records

```
conferenceModelList =
    YlsConferenceManager.getInstance().getConferenceList();
```

Delete conference call records

```
/**
 * Delete specific conference call records
 * @param conferenceId: ID of the conference call record to be deleted
 */
public void deleteConferenceLog(String conferenceId)

/**
 * Delete all the conference call records
 */
public void deleteAllConferenceLog()
```

Set / Query the end time of a conference call

```
/**
 * Set the end time of a conference call
 * @param endConferenceTime
 */
public void setEndConferenceTime(long endConferenceTime)

/**
 * Query the countdown time of a conference call
 * You can make a new conference call only when the countdown is negative
 * @return
 */
public long getCountDownTime()
```

Set / Query the cached information of a conference call

```
/**
 * Set the cached information (name, member, etc) of the current conference
 * call
 * @param conferenceVo
 */
public void setConferenceVo(ConferenceVo conferenceVo)
```

```

/**
 * Query the cached information of the current conference call
 * @return
 */
public ConferenceVo getConferenceVo()

```

Reconnect to a conference call

```

/**
 * When a conference call was interrupted due to unstable network
 * condition, use this method to reconnect to the conference call
 * @param conferenceId: ID of the exceptional conference call
 * @param member: Conference call members
 * @return
 */
public ResultVo returnConferenceBlock(String conferenceId, String member)

```

Push Notifications

This topic introduces the functionalities and implementation methods related to push notifications of 'Linkus SDK for Android'.

Configure push notification

```

/**
 *
 * @param mode: Push service platform; huawei, xiaomi, firebase
 * @param token: Authentication information of the push certificate
 * @param requestCallback
 * @return
 */
public void setPushInfo(String mode, String token, RequestCallback
    requestCallback)
//Example
YlsBaseManager.getInstance().setPushInfo("GETUI", clientid, new
    RequestCallback() {

// Configuration success callback
@Override
public void onSuccess(Object result) {
    }

// Configuration failure callback

```

```

@Override
public void onFailed(int code) {
    }

// Configuration exception callback
@Override
public void onException(Throwable exception) {
    }
});

```

Handle push notifications

```

String data = new String(payload);
JSONObject jsonObject = null;
try {
    jsonObject = new JSONObject(data);
} catch (JSONException e) {
    e.printStackTrace();
}
YlsCallManager.getInstance().handlerPushMessage(context, jsonObject);

```

Call Detailed Record (CDR)

This topic introduces the functionalities and implementation methods related to Call Detailed Record (CDR) of 'Linkus SDK for Android'.

Retrieve CDR records

```

/**
 * Retrieve the specified number of CDR records
 * @param limit: Specify the number of CDR records that you want to retrieve
 * @return
 */
public List<CdrVo> getCdrList(int limit);
//Example
List<CdrVo> cdrVoList =
YlsCallLogManager.getInstance().getCdrList(1000);

```

Delete specific CDR records

```

/**
 *

```

```

* @param cdrIds: IDs of the CDR records that need to be deleted, separate
multiple IDs by ','
* @return
*/
public int deleteCdr(String cdrIds)

```

Delete all CDR records

```

/**
 *
 * @return
 */
public int deleteAllCdr()
//Example
    btnCdrClear.setOnClickListener(v ->
        YlsCallLogManager.getInstance().deleteAllCdr());

```

Query the number of missed calls

```

/**
 *
 * @return
 */
public int getMissCallCdrCount();

```

Mark all unread CDR records as read

```

/**
 *
 * @return
 */
public void readAllCdr()

```

Audio settings

This topic introduces the functionalities and implementation methods related to audio settings of 'Linkus SDK for Android'.

Enable or disable automatic gain control

```

public void agcSetting(boolean isOpen)

```

Enable or disable echo cancellation

```
public void echoSetting(boolean isOpen)
```

Enable or disable noise cancellation

```
public void ncSetting(boolean isOpen)
```


Linkus SDK for iOS

Linkus SDK for iOS Overview

Yeastar P-Series PBX System supports Linkus SDK, enabling you to integrate calling, CDR, and more Linkus UC Clients' functionalities into 3rd-party iOS applications. This topic describes the requirements, prerequisites, demo and source code, integration process, and features of 'Linkus SDK for iOS'.

Requirements and Prerequisites

Requirements

Platform / Environment	Requirement
PBX Server	<ul style="list-style-type: none">• Firmware: 37.12.0.23 or later• Plan: Ultimate Plan (UP)
Development Environment	<ul style="list-style-type: none">• iOS: Version 11 or later• Xcode: Version 14.3 or later

Prerequisites

You have obtained APNs (Apple Push Notification service) certificate.

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for iOS' to have an overview of the framework and workflow of 'Linkus SDK for iOS'.

For more information, go to the [GitHub Repository of 'Linkus SDK for iOS'](#).

Integration Process and Features

Integration Process

1. [Enable Linkus SDK and Bind APNs Certificate](#)
2. [Integrate Linkus SDK for iOS](#)
3. [Obtain Login Signature for 'Linkus SDK for iOS'](#)

Features

- [Configurations](#)
- [Login and Logout](#)
- [Call Features](#)
- [Conference Call](#)
- [Call Information](#)
- [Complex Call Scenarios](#)
- [Call Detailed Record \(CDR\)](#)

Release Notes - Linkus SDK for iOS

Version 1.2.0

Release date: April 22, 2024

- Added compatibility with iPads, supporting simultaneous use of Linkus SDK on both iPhones and iPads.

Version 1.1.1

Release date: October 11, 2023

- First release of **Linkus SDK for iOS**. By integrating the SDK into your iOS projects, you can quickly add calling, CDR, and more Linkus UC Clients' functionalities to your iOS applications.

Integrate Linkus SDK for iOS

Enable Linkus SDK and Bind APNs Certificate

Before integrating 'Linkus SDK for iOS' with your iOS project, you need to enable Linkus SDK and bind APNs (Apple Push Notification service) certificate on Yeastar P-Series PBX System, so that iOS devices can receive incoming call notifications after the integration.

Requirements and Prerequisites

Requirements

Make sure that the PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later

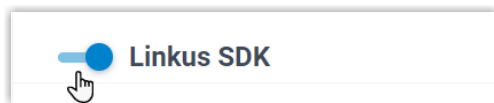
- **Plan:** Ultimate Plan (UP)

Prerequisites

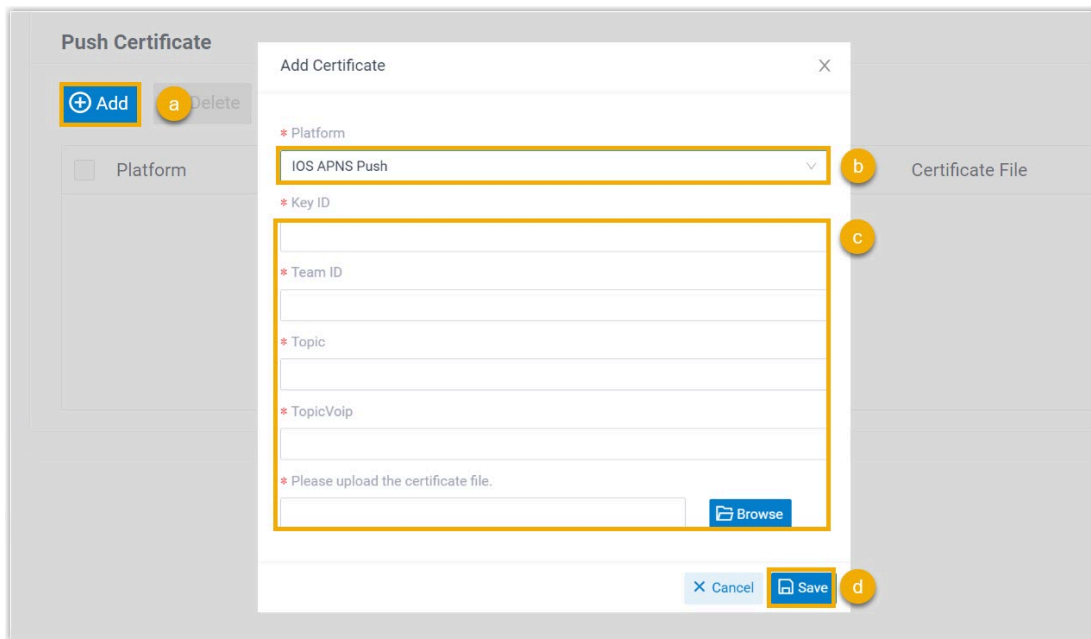
You have obtained APNs (Apple Push Notification service) certificate.

Procedure

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Enable **Linkus SDK**.



3. Bind the APNs certificate.



- a. In the **Push Certificate** section, click **Add**.
 - b. In the **Platform** drop-down list, select **IOS APNS Push**.
 - c. Fill in the required information and upload the APNs certificate.
 - d. Click **Save**.
4. Click **Save**.

Result

You have enabled Linkus SDK and bound the APNs certificate, you can [Integrate Linkus SDK for iOS](#).

**Important:**

After integrating 'Linkus SDK for iOS', your iOS application will use the APNs certificate to send call-related push notifications to devices, and the Linkus's push certificate (Linkus Mobile Client's push notification) will no longer take effect.

Integrate Linkus SDK for iOS

To integrate 'Linkus SDK for iOS', you need to import 'Linkus SDK for iOS' to your iOS project and initialize it.

Requirements and Prerequisites

Requirements

Make sure that your development environment meets the following requirements:

- **iOS:** Version 11 or later
- **Xcode:** Version 14.3 or later

Prerequisites

You have [enabled Linkus SDK and bound APNs certificate](#).

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for iOS' to have an overview of the framework and workflow of 'Linkus SDK for iOS'.

For more information, go to the [GitHub Repository of 'Linkus SDK for iOS'](#).

Step 1. Import 'Linkus SDK for iOS'

Use either of the following methods to import 'Linkus SDK for iOS':

- [Use CocoaPods to integrate 'Linkus SDK for iOS'](#)
- [Manually integrate 'Linkus SDK for iOS'](#)

Use CocoaPods to integrate 'Linkus SDK for iOS'

**Note:**



Before the integration, make sure that you have installed CocoaPods. For more information, see [Getting Started with CocoaPods](#).

1. In the iOS project, open **Podfile** and add the following code, then save it.



Note:

If there is no **Podfile**, you can enter the project root directory in Terminal and run the `pod init` command to generate it.

```
pod 'linkus-sdk'
```

2. In Terminal, run the following command to install 'Linkus SDK for iOS'.

```
pod install
```

After the installation is completed, the Terminal will display **Pod installation complete!**, and a new file with a suffix of **.xcworkspace** is generated under the project folder.

3. Use **Xcode** to open the file with the **.xcworkspace** suffix.

Manually integrate 'Linkus SDK for iOS'

1. Go to the [GitHub Repository of 'Linkus SDK for iOS'](#), and download 'Linkus SDK for iOS'.
2. In **Xcode**, add the **linkus_sdk_iOS.framework** to the corresponding **Target**, and select **Copy items if needed** in the pop-up window.
3. In the iOS project, go to **Build Phases > Link Binary With Libraries** and add the following libraries:

```
libz.dylib
libc++.dylib
libxml2.dylib
libresolv.dylib
```

Step 2. Initialize 'Linkus SDK for iOS'

1. In the iOS project, open **PrefixHeader.pch** file and import header files.

```
#import <linkus_sdk_iOS/linkus_sdk.h>
```

2. Use the following code snippet to initialize 'Linkus SDK for iOS' within the **application:didFinishLaunchingWithOptions:** method of **AppDelegate.m** file.

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [[YLSSDK sharedYLSSDK] initApp];
    return YES;
}
```

What to do next

Request the SDK login signature from PBX server for authentication and login to 'Linkus SDK for iOS'.

For more information, see [Obtain Login Signature for 'Linkus SDK for iOS'](#).

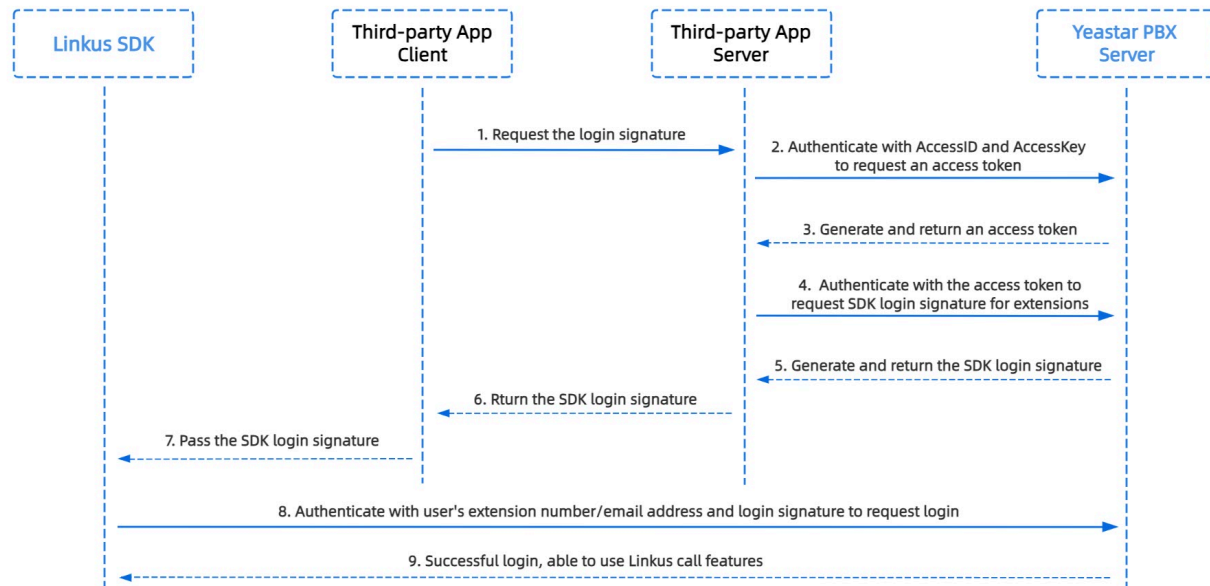
Obtain Login Signature for 'Linkus SDK for iOS'

When logging into Linkus SDK, users need to use the SDK login signature for authentication instead of their login password. This topic describes how to request users' Linkus SDK login signatures from the PBX server via OpenAPI.

Prerequisites

- You have [enabled Linkus SDK and bound the APNs certificate](#).
- You have [integrated 'Linkus SDK for iOS'](#).

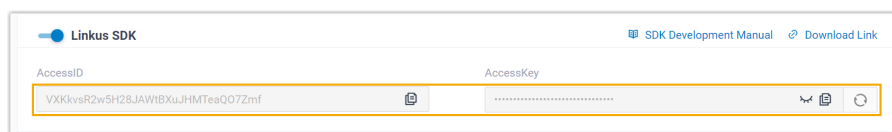
Linkus SDK login process



Step 1. Obtain the 'AccessID' and 'AccessKey' for Linkus SDK on PBX

Obtain the 'AccessID' and 'AccessKey' for Linkus SDK from Yeastar P-Series PBX System, which will be used for the third-party application to authenticate and connect with the PBX server.

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Note down the **AccessID** and **AccessKey**.



Step 2. Request an access token from PBX

On the third-party application server, use the 'AccessID' and 'AccessKey' to request an access token from the PBX via OpenAPI. Access token is used to verify an authenticated API call, enabling you to request Linkus SDK login signatures for extensions.

Request URL

```
POST {base_url}/openapi/v1.0/get_token
```



Note:



To learn about the API request structure, see [Request Structure](#).

Request Parameters

Parameter	Required	Type	Description
username	Yes	String	User name. Use the AccessID of Linkus SDK as the username.
password	Yes	String	Password. Use the AccessKey of Linkus SDK as the password.

Request example

```
POST /openapi/v1.0/get_token
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "VXKkvsR2w5H28JAWtBXuJHMTeaQO7Zmf",
  "password": "Yq6yVsBceOZLhnuaGeMUG4U4qXXXXXXX"
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> 0: Succeed. Non-zero value: Failed.
errmsg	String	Returned message. <ul style="list-style-type: none"> SUCCESS: Succeed. FAILURE: Failed.
access_token_expire_time	Integer	Access token expire time. (Unit : second)
access_token	String	Credential of calling API interfaces. All requests to call API interfaces must carry an access token.
refresh_token_expire_time	Integer	Refresh token expire time. (Unit : second)
refresh_token	String	Refresh token.

Parameter	Type	Description
		<code>refresh_token</code> can be used to obtain new <code>access_token</code> and <code>refresh_token</code> .

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "access_token_expire_time": 1800,
  "access_token": "EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS",
  "refresh_token_expire_time": 86400,
  "refresh_token": "SCduGecwbG9jIusiS8FxFUVn3kf0Q9R8"
}
```

Step 3. Request login signatures for extensions

Use the access token to request login signatures for extensions from PBX server via Open-API.

Request URL

```
POST /openapi/v1.0/sign/create?access_token={access_token}
```

Request Parameters

Parameter	Required	Type	Description
username	Yes	string	Extension number or email address.
sign_type	Yes	string	Login signature type. Permitted value: <code>sdk</code>
expire_time	No	Integer	Expiration timestamp of the login signature. (Unit: second) <code>0</code> indicates no limitation on the validity duration of the login signature.

Request example

```
POST /openapi/v1.0/sign/create?access_token=EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "1000",
```

```

"sign_type": "sdk",
  "expire_time": 0
}

```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> • 0: Succeed. • Non-zero value: Failed
errmsg	String	Returned message. <ul style="list-style-type: none"> • SUCCESS: Succeed. • FAILURE: Failed.
data	Array <Ext_Sign>	Linkus SDK login signatures for extensions.

Ext_Sign

Parameter	Type	Description
sign	String	Extension's Linkus SDK login signature.

Response example

```

HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "data": {
    "sign": "ueb3rfcpsis8FxFUZMpZAO86mbrKmVn3kf0Q9R8"
  }
}

```

Result

The third-party application server has obtained the Linkus SDK login signature for extension users, the login signature will be automatically returned to the third-party application client, and passed to Linkus SDK.

What to do next

Use the SDK login signature for authentication and [log in to 'Linkus SDK for iOS'](#).

Use Linkus SDK for iOS

Configurations

This topic introduces the functionalities and implementation methods related to configurations of 'Linkus SDK for iOS' .

```
/**
 * Configuration items
 */
+ (instancetype)sharedConfig;

/// Set the app name
@property (nonatomic,copy,nullable) NSString *localizedName;

/// Set the app icon with a size of 40*40 points
@property (nonatomic,copy,nullable) NSData *iconTemplateImageData;

/// Set the file path for logs, the corresponding folder needs to be
    created manually
@property (nonatomic,copy) NSString *logPath;

/// Set the file path for data, the corresponding folder needs to be
    created manually
@property (nonatomic,copy) NSString *dataPath;

/// Set the incoming call ringtone
@property (nonatomic,copy) NSString *comeAudioFileName
    API_AVAILABLE(macos(10.13));

/// Set the call end tone
@property (nonatomic,copy) NSString *hangupAudioFileName;

/// Set the call waiting tone
@property (nonatomic,copy) NSString *alertAudioFileName;
```

Login and Logout

This topic introduces the functionalities and implementation methods related to login and logout of 'Linkus SDK for iOS'.

Manual Login

```
/**
 * Refer to the table below for descriptions of the values returned by the
 * login interface
 */
- (void)login:(NSString *)account token:(NSString *)token localIP:(NSString
*)localIP localPort:(NSString *)localPort
    remoteIP:(NSString *)remoteIP remotePort:(NSString *)remotePort
completion:(void (^)(NSError * _Nullable error))completion;
```

Login interface return value description

Return Value	Description
1	Failed to connect to the PBX server.
-5	No response to the login request.
403	Invalid user name or login signature.
405	Linkus UC client is disabled.
407	This account has been locked.
416	Access to the requested IP address is prohibited (Allowed Country / Region IP Access Protection is enabled).

Automatic login

```
- (void)autoLogin API_AVAILABLE(ios(11.0));
```

Logout

```
- (void)logout:(void (^)(NSError * _Nullable error))completion;
```

SDK notification callbacks

```
/**
 * Login callback
 */
- (void)onLoginStep:(LoginStep)step;
```

```
/**
 * Forced logout callback
 */
- (void)onKickStep:(KickReason)code;
```

Call Features

This topic introduces the functionalities and implementation methods related to call features of 'Linkus SDK for iOS'.

Make a call

```
- (void)startCall:(YLSSipCall *)sipCall completion:(void (^)(NSError
*error))completion;
```

Hang up a call

```
- (void)endCall:(YLSSipCall *)sipCall;
```

Hold / Resume a call

```
- (void)setHeld:(YLSSipCall *)sipCall;
```

Mute / Unmute a call

```
- (void)setMute:(YLSSipCall *)sipCall;
```

Record a call

```
- (BOOL)setRecord:(YLSSipCall *)sipCall;
```

Perform an attended transfer

```
- (void)transferConsultation:(YLSSipCall *)sipCall;
```

Perform a blind transfer

```
- (void)tranforBlind:(YLSSipCall *)sipCall;
```

Query call quality

```
- (NSString *)callQuality;
```

Conference Call

This topic introduces the functionalities and implementation methods related to conference call of 'Linkus SDK for iOS'.

Make a conference call

```
- (void)createConference:(YLSConfCall *)confCall
    complete:(void (^)(NSError * _Nullable error, NSString
*confid))complete;
```

Manage conference call members

```
- (void)operationConferenceMember:(NSString *)member
    confid:(NSString *)confid
    operationType:(int)type
    complete:(void (^)(NSError * _Nullable
error))complete;
```

Invite members to a conference call

```
- (void)inviteConferenceMembers:(NSArray<NSString *> *)contacts
    confid:(NSString *)confid
    complete:(void (^)(NSError * _Nullable
error))complete;
```

Join a conference call

```
- (void)conferenceManager:(YLSConfManager *)manager
    callStatus:(YLSSipCall *)sipCall
    reportIncomingCall:(void (^)(void (^controllerBlock)(void), void
(^errorBlock)(NSError * _Nullable error)))completion;
```

Query conference call status

```
- (void)conferenceManager:(YLSConfManager *)manager callStatus:(YLSSipCall
*)sipCall;
```

Query information of the current conference call

```
- (YLSSipCall *)currentConfSipCall;
```

Query the status of conference call members

```
- (void)conferenceManager:(YLSCnfManager *)manager  
conferenceInfo:(YLSCnfCall *)confCall;
```

Rejoin a conference call

This method is used to handle unexpected situations in conference calls, such as interruptions due to unstable network. You can use this method to rejoin the conference call.

```
- (void)conferenceManager:(YLSCnfManager *)manager abnormal:(nullable  
YLSCnfCall *)confCall;
```

Set incoming call delegate for conference call

```
- (void)setIncomingCallDelegate:(id<YLSCnfManagerDelegate>)delegate;
```

Add delegate for conference call

```
- (void)addDelegate:(id<YLSCnfManagerDelegate>)delegate;
```

Remove delegate for conference call

```
- (void)removeDelegate:(id<YLSCnfManagerDelegate>)delegate;
```

Call Information

This topic introduces the functionalities and implementation methods related to call information of 'Linkus SDK for iOS'.

Handle VoIP push notifications

```
- (void)receiveIncomingPushWithPayload:(NSDictionary *)dictionaryPayload;
```

Handle missed calls

```
- (BOOL)didReceiveRemoteNotification:(NSDictionary *)userInfo;
```

Query information of the current call

```
- (YLSSipCall *)currentSipCall;
```

Query information of all the calls

```
- (NSArray<YLSSipCall *> *)currentSipCalls;
```

Query SIP registration status

```
- (BOOL)sipRegister;
```

Query recording availability

```
- (BOOL)enableRecord;
```

Query users' recording permissions

```
- (BOOL)adminRecord;
```

Set incoming call delegate

```
- (void)setIncomingCallDelegate:(id<YLSCallManagerDelegate>)delegate;
```

Add delegate

```
- (void)addDelegate:(id<YLSCallManagerDelegate>)delegate;
- (void)addDelegate:(id<YLSCallStatusManagerDelegate>)delegate;
```

Remove delegate

```
- (void)removeDelegate:(id<YLSCallManagerDelegate>)delegate;
```

SDK notification callbacks

```
/**
 * Incoming call callback
 */
- (void)callManager:(YLSCallManager *)callManager contact:(void
(^)(id<YLSCallProtocol> (^block)(NSString *number)))contact
completion:(void (^)(void (^controllerBlock)(void),void
(^errorBlock)(NSError *error)))completion;
```



```

/**
 * Call status change callback
 */
- (void)callManager:(YLSCallManager *)callManager
callInfoStatus:(NSMutableArray<YLSSipCall *> *)currentCallArr;

/**
 * SIP error code callback
 */
- (void)callManager:(YLSCallManager *)callManager callFailed:(NSError
*)error;

/**
 * Call recording status callback
 */
- (void)callManagerRecordType:(YLSCallManager *)callManager;

/**
 * Current call quality callback
 */
- (void)callManager:(YLSCallManager *)callManager
callQuality:(BOOL)quality;

/**
 * Call waiting callback
 */
- (BOOL)callWaitingSupport;

```

Complex Call Scenarios

This topic introduces the functionalities and implementation methods related to complex call scenarios (call waiting, call transfer, multi-party call, etc.) in 'Linkus SDK for iOS'.

Switch calls during call waiting

```

- (void)callChange:(YLSSipCall *)waitingCall;

```

Add delegate

When there are complex call scenarios (such as call transfer, call waiting, or multi-party call), use this method to add delegate.

```

- (void)addDelegate:(id<YLSCallStatusManagerDelegate>)delegate;

```

Remove delegate

When there are complex call scenarios (such as call transfer, call waiting, or multi-party call), use this method to remove delegate.

```
- (void)removeDelegate:(id<YLSCallStatusManagerDelegate>)delegate;
```

Hang up all the calls

Hang up all the calls in a multi-party call.

```
- (void)callStatusManagerDismiss:(YLSCallStatusManager
 *)callStatusManager;
```

Handle incoming calls while on a call

When receiving a new incoming call while on a call, use this method to call back the current call's status (answer, ringing, hang up, mute, etc.).

```
- (void)callStatusManager:(YLSCallStatusManager *)callStatusManager
currentCall:(YLSSipCall *)currentCall;
```

Callback for call status in complex call scenarios

When there are complex call scenarios (such as call transfer, call waiting, or multi-party call), use this method to call back the current call's status.

```
- (void)callStatusManager:(YLSCallStatusManager *)callStatusManager
currentCall:(YLSSipCall *)currentCall
    callWaiting:(nullable YLSSipCall *)callWaitingCall
transferCall:(nullable YLSSipCall *)transferCall;
```

Call Detailed Record (CDR)

This topic introduces the functionalities and implementation methods related to Call Detailed Record (CDR) of 'Linkus SDK for iOS'.

Retrieve CDR records

```
- (NSArray<YLSHistory *> *)historys;
```

Delete specific CDR records

```
- (void)historyManagerRemove:(NSArray<YLSHistory *> *)historys;
```

Delete all the CDR records

```
- (void)historyManagerRemoveAll;
```

Mark unread CDR records as read

```
- (void)checkMissedCalls;
```

SDK notification callbacks

```
/**
 * CDR records change callback
 */
- (void)historyReload:(NSMutableArray<YLSHistory *> *)historys;

/**
 * Missed call count change callback
 */
- (void)historyMissCallCount:(NSInteger)count;
```

Linkus SDK for macOS

Linkus SDK for macOS Overview

Yeastar P-Series PBX System supports Linkus SDK, enabling you to integrate calling, CDR, and more Linkus UC Clients' functionalities into 3rd-party macOS applications. This topic describes the requirements, demo and source code, integration process, and features of 'Linkus SDK for macOS'.

Requirements

Platform / Environment	Requirement
PBX Server	<ul style="list-style-type: none">• Firmware: 37.12.0.23 or later• Plan: Ultimate Plan (UP)
Development Environment	<ul style="list-style-type: none">• macOS: Version 10.13 or later• Xcode: Version 14.3 or later

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for macOS' to have an overview of the framework and workflow of 'Linkus SDK for macOS'.

For more information, go to the [GitHub Repository of 'Linkus SDK for macOS'](#).

Integration Process and Features

Integration Process

1. [Enable Linkus SDK](#)
2. [Integrate Linkus SDK for macOS](#)
3. [Obtain Login Signature for 'Linkus SDK for macOS'](#)

Features

- [Configurations](#)
- [Login and Logout](#)
- [Call Features](#)
- [Call Information](#)

- [Complex Call Scenarios](#)
- [Call Detailed Record \(CDR\)](#)

Release Notes - Linkus SDK for macOS

Version 1.0.12

Release date: October 11, 2023

- First release of **Linkus SDK for macOS**. By integrating the SDK into your macOS projects, you can quickly add calling, CDR, and more Linkus UC Clients' functionalities to your macOS applications.

Integrate Linkus SDK for macOS

Enable Linkus SDK

Before integrating 'Linkus SDK for macOS' with your macOS project, you need to enable Linkus SDK on Yeastar P-Series PBX System.

**Note:**

After enabling Linkus SDK, Linkus Mobile Client's push notification will no longer take effect. If you want to continue receiving call-related push notifications on mobile devices, see '[Linkus SDK for Android Overview](#)' or '[Linkus SDK for iOS Overview](#)'.

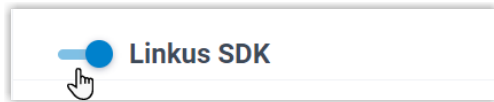
Requirements

Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Procedure

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Enable **Linkus SDK**.



3. Click **Save**.

Result

You have enabled Linkus SDK, and you can [Integrate Linkus SDK for macOS](#).

Integrate Linkus SDK for macOS

To integrate 'Linkus SDK for macOS', you need to import 'Linkus SDK for macOS' to your macOS project and initialize it.

Requirements and Prerequisites

Requirements

Make sure that your development environment meets the following requirements:

- **macOS:** Version 10.13 or later
- **Xcode:** Version 14.3 or later

Prerequisites

You have [enabled Linkus SDK on PBX](#).

Demo and Source code

Before the integration, we recommend that you try out the Demo and review the source code of 'Linkus SDK for macOS' to have an overview of the framework and workflow of 'Linkus SDK for macOS'.

For more information, go to the [GitHub Repository of 'Linkus SDK for macOS'](#).

Step 1. Import 'Linkus SDK for macOS'

Use either of the following methods to import 'Linkus SDK for macOS':

- [Use CocoaPods to integrate 'Linkus SDK for macOS'](#)
- [Manually integrate 'Linkus SDK for macOS'](#)

Use CocoaPods to integrate 'Linkus SDK for macOS'

**Note:**

Before the integration, make sure that you have installed CocoaPods. For more information, see [Getting Started with CocoaPods](#).

1. In the macOS project, open **Podfile** and add the following code, then save it.

**Note:**

If there is no **Podfile**, you can enter the project root directory in Terminal and run the `pod init` command to generate it.

```
pod 'linkus-sdk-macOS'
```

2. In Terminal, run the following command to install 'Linkus SDK for macOS'.

```
pod install
```

After the installation is completed, the Terminal will display **Pod installation complete!**, and a new file with a suffix of **.xcworkspace** is generated under the project folder.

3. Use **Xcode** to open the file with the **.xcworkspace** suffix.

Manually integrate 'Linkus SDK for macOS'

1. Go to the [GitHub Repository of 'Linkus SDK for macOS'](#), and download 'Linkus SDK for macOS'.
2. In **Xcode**, add the **linkus_sdk_MacOS.framework** to the corresponding **Target**, and select **Copy items if needed** in the pop-up window.
3. In the macOS project, go to **Build Phases > Link Binary With Libraries** and add the following libraries:

```
libcurl.dylib  
libxml2.dylib  
libc++.dylib
```

Step 2. Initialize 'Linkus SDK for macOS'

1. In macOS project, open **PrefixHeader.pch** file and import header files.

```
#import <linkus_sdk_MacOS/linkus_sdk.h>
```

2. Refer to the [Demo](#) to initialize 'Linkus SDK for macOS'.

What to do next

Request the SDK login signature from PBX server for authentication and login to 'Linkus SDK for macOS'.

For more information, see [Obtain Login Signature for 'Linkus SDK for macOS'](#).

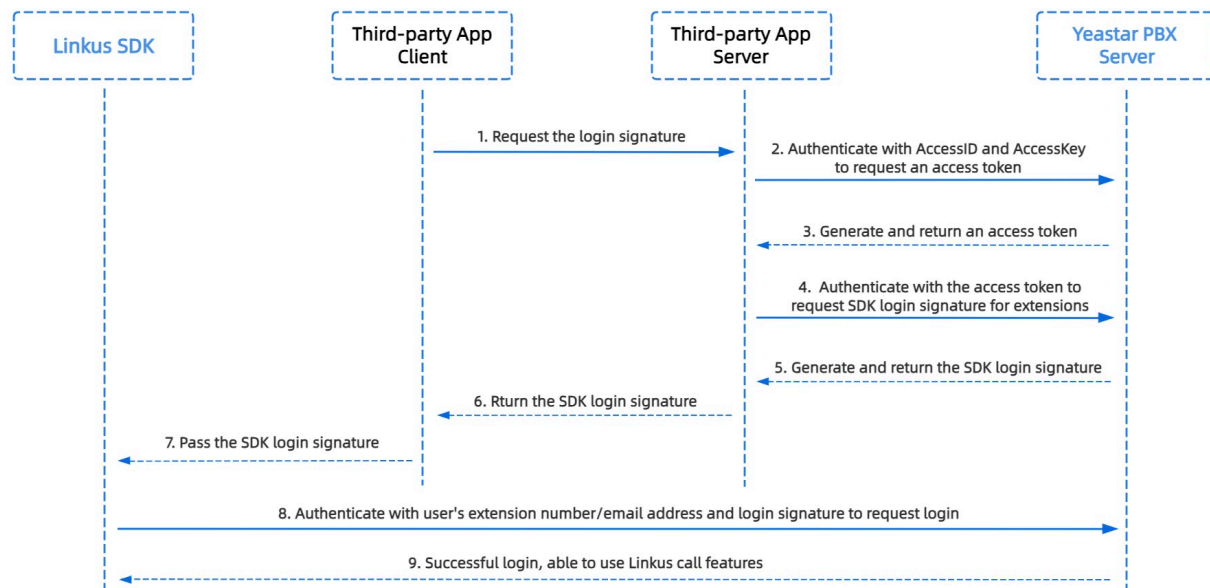
Obtain Login Signature for 'Linkus SDK for macOS'

When logging into Linkus SDK, users need to use the SDK login signature for authentication instead of their login password. This topic describes how to request users' Linkus SDK login signatures from the PBX server via OpenAPI.

Prerequisites

- You have [enabled Linkus SDK on Yeastar P-Series PBX System](#).
- You have [integrated 'Linkus SDK for macOS'](#).

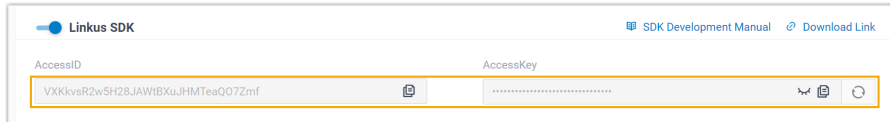
Linkus SDK login process



Step 1. Obtain the 'AccessID' and 'AccessKey' for Linkus SDK on PBX

Obtain the 'AccessID' and 'AccessKey' for Linkus SDK from Yeastar P-Series PBX System, which will be used for the third-party application to authenticate and connect with the PBX server.

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Note down the **AccessID** and **AccessKey**.



The screenshot shows the Linkus SDK web portal interface. At the top, there is a header with the Linkus SDK logo and links for 'SDK Development Manual' and 'Download Link'. Below the header, there are two input fields: 'AccessID' and 'AccessKey'. The 'AccessID' field contains the value 'VXKkvsR2w5H28JAWtBXuJHMTeeQO7Zmf'. The 'AccessKey' field is empty and has a masked input type (password field).

Step 2. Request an access token from PBX

On the third-party application server, use the 'AccessID' and 'AccessKey' to request an access token from the PBX via OpenAPI. Access token is used to verify an authenticated API call, enabling you to request Linkus SDK login signatures for extensions.

Request URL

```
POST {base_url}/openapi/v1.0/get_token
```



Note:

To learn about the API request structure, see [Request Structure](#).

Request Parameters

Parameter	Required	Type	Description
username	Yes	String	User name. Use the AccessID of Linkus SDK as the username.
password	Yes	String	Password. Use the AccessKey of Linkus SDK as the password.

Request example

```
POST /openapi/v1.0/get_token
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "VXKkvsR2w5H28JAWtBXuJHMTeeQO7Zmf",
```

```
"password": "Yq6yVsBceOZLhnuaGeMUG4U4qXXXXXX"
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> 0: Succeed. Non-zero value: Failed.
errmsg	String	Returned message. <ul style="list-style-type: none"> SUCCESS: Succeed. FAILURE: Failed.
access_token_expire_time	Integer	Access token expire time. (Unit: second)
access_token	String	Credential of calling API interfaces. All requests to call API interfaces must carry an access token.
refresh_token_expire_time	Integer	Refresh token expire time. (Unit: second)
refresh_token	String	Refresh token. <u>refresh_token</u> can be used to obtain new <u>access_token</u> and <u>refresh_token</u> .

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "access_token_expire_time": 1800,
  "access_token": "EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS",
  "refresh_token_expire_time": 86400,
  "refresh_token": "SCduGecwbG9jIusiS8FxFUVn3kf0Q9R8"
}
```

Step 3. Request login signatures for extensions

Use the access token to request login signatures for extensions from PBX server via Open-API.

Request URL

```
POST /openapi/v1.0/sign/create?access_token={access_token}
```

Request Parameters

Parameter	Required	Type	Description
username	Yes	string	Extension number or email address.
sign_type	Yes	string	Login signature type. Permitted value: <code>sdk</code>
expire_time	No	Integer	Expiration timestamp of the login signature. (Unit: second) <code>0</code> indicates no limitation on the validity duration of the login signature.

Request example

```
POST /openapi/v1.0/sign/create?access_token=EXZMpZA086mbrKm6rFtg
eb3rfcpC9uqS
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "1000",
  "sign_type": "sdk",
  "expire_time": 0
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"><code>0</code>: Succeed.Non-zero value: Failed
errmsg	String	Returned message. <ul style="list-style-type: none"><code>SUCCESS</code>: Succeed.<code>FAILURE</code>: Failed.
data	Array < Ext_Sign >	Linkus SDK login signatures for extensions.

Ext_Sign

Parameter	Type	Description
sign	String	Extension's Linkus SDK login signature.

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "data": {
    "sign": "ueb3rfcpsiS8FxFUZMpZAO86mbrKmVn3kf0Q9R8"
  }
}
```

Result

The third-party application server has obtained the Linkus SDK login signature for extension users, the login signature will be automatically returned to the third-party application client, and passed to Linkus SDK.

What to do next

Use the SDK login signature for authentication and [log in to 'Linkus SDK for macOS'](#).

Use Linkus SDK for macOS

Configurations

This topic introduces the functionalities and implementation methods related to configurations of 'Linkus SDK for macOS'.

```
/**
 * Configuration items
 */
+ (instancetype)sharedConfig;

/// Set the app name
@property (nonatomic,copy,nullable) NSString *localizedName;

/// Set the app icon with a size of 40*40 points
@property (nonatomic,copy,nullable) NSData *iconTemplateImageData;
```

```

/// Set the file path for logs, the corresponding folder needs to be
    created manually
@property (nonatomic,copy) NSString *logPath;

/// Set the file path for data, the corresponding folder needs to be
    created manually
@property (nonatomic,copy) NSString *dataPath;

/// Set the incoming call ringtone
@property (nonatomic,copy) NSString *comeAudioFileName
    API_AVAILABLE(macos(10.13));

/// Set the call end tone
@property (nonatomic,copy) NSString *hangupAudioFileName;

/// Set the call waiting tone
@property (nonatomic,copy) NSString *alertAudioFileName;

```

Login and Logout

This topic introduces the functionalities and implementation methods related to login and logout of 'Linkus SDK for macOS'.

Manual Login

```

/**
 * Refer to the table below for descriptions of the values returned by the
    login interface
 */
- (void)login:(NSString *)account token:(NSString *)token localIP:(NSString
    *)localIP localPort:(NSString *)localPort
    remoteIP:(NSString *)remoteIP remotePort:(NSString *)remotePort
    completion:(void (^)(NSError * _Nullable error))completion;

```

Login interface return value description

Return Value	Description
1	Failed to connect to the PBX server.
-5	No response to the login request.
403	Invalid user name or login signature.
405	Linkus UC client is disabled.

Return Value	Description
407	This account has been locked.
416	Access to the requested IP address is prohibited (Allowed Country / Region IP Access Protection is enabled).

Automatic login

```
- (void)autoLogin API_AVAILABLE(ios(11.0));
```

Logout

```
- (void)logout:(void (^)(NSError * _Nullable error))completion;
```

SDK notification callbacks

```
/**
 * Login callback
 */
- (void)onLoginStep:(LoginStep)step;

/**
 * Forced logout callback
 */
- (void)onKickStep:(KickReason)code;
```

Call Features

This topic introduces the functionalities and implementation methods related to call features of 'Linkus SDK for macOS'.

Make a call

```
- (void)startCall:(YLSSipCall *)sipCall completion:(void (^)(NSError *error))completion;
```

Hang up a call

```
- (void)endCall:(YLSSipCall *)sipCall;
```

Hold / Resume a call

```
- (void)setHeld:(YLSSipCall *)sipCall;
```

Mute / Unmute a call

```
- (void)setMute:(YLSSipCall *)sipCall;
```

Record a call

```
- (BOOL)setRecord:(YLSSipCall *)sipCall;
```

Perform an attended transfer

```
- (void)transferConsultation:(YLSSipCall *)sipCall;
```

Perform a blind transfer

```
- (void)tranforBlind:(YLSSipCall *)sipCall;
```

Query call quality

```
- (NSString *)callQuality;
```

Call Information

This topic introduces the functionalities and implementation methods related to call information of 'Linkus SDK for macOS'.

Handle missed calls

```
- (BOOL)didReceiveRemoteNotification:(NSDictionary *)userInfo;
```

Query information of the current call

```
- (YLSSipCall *)currentSipCall;
```

Query information of all the calls

```
- (NSArray<YLSSipCall *> *)currentSipCalls;
```

Query microphone and speaker information

```
- (NSArray<YLSCaptureDevice *> *)audioALLDevice
  API_AVAILABLE(macos(10.13));
```

Set microphone and speaker

```
- (void)audioSetDevice:(NSInteger)microphone speaker:(NSInteger)speaker
  API_AVAILABLE(macos(10.13));
```

Query SIP registration status

```
- (BOOL)sipRegister;
```

Query recording availability

```
- (BOOL)enableRecord;
```

Query users' recording permissions

```
- (BOOL)adminRecord;
```

Set incoming call delegate

```
- (void)setIncomingCallDelegate:(id<YLSCallManagerDelegate>)delegate;
```

Add delegate

```
- (void)addDelegate:(id<YLSCallManagerDelegate>)delegate;
- (void)addDelegate:(id<YLSCallStatusManagerDelegate>)delegate;
```

Remove delegate

```
- (void)removeDelegate:(id<YLSCallManagerDelegate>)delegate;
```

SDK notification callbacks

```
/**
 * Incoming call callback
 */
- (void)callManager:(YLSCallManager *)callManager contact:(void
(^)(id<YLSContactProtocol> (^block)(NSString *number)))contact
```



```

completion:(void (^)(void (^controllerBlock)(void),void
(^errorBlock)(NSError *error)))completion;

/**
 * Call status change callback
 */
- (void)callManager:(YLSCallManager *)callManager
callInfoStatus:(NSMutableArray<YLSSipCall *> *)currentCallArr;

/**
 * SIP error code callback
 */
- (void)callManager:(YLSCallManager *)callManager callFailed:(NSError
*)error;

/**
 * Call recording status callback
 */
- (void)callManagerRecordType:(YLSCallManager *)callManager;

/**
 * Current call quality callback
 */
- (void)callManager:(YLSCallManager *)callManager
callQuality:(BOOL)quality;

/**
 * Call waiting callback
 */
- (BOOL)callWaitingSupport;

```

Complex Call Scenarios

This topic introduces the functionalities and implementation methods related to complex call scenarios (call waiting, call transfer, etc.) in 'Linkus SDK for macOS'.

Switch calls during call waiting

```

- (void)callChange:(YLSSipCall *)waitingCall;

```

Add delegate

When there are complex call scenarios (such as call transfer or call waiting), use this method to add delegate.

```
- (void)addDelegate:(id<YLSCallStatusManagerDelegate>)delegate;
```

Remove delegate

When there are complex call scenarios (such as call transfer or call waiting), use this method to remove delegate.

```
- (void)removeDelegate:(id<YLSCallStatusManagerDelegate>)delegate;
```

Handle incoming calls while on a call

When receiving a new incoming call while on a call, use this method to call back the current call's status (answer, ringing, hang up, mute, etc.).

```
- (void)callStatusManager:(YLSCallStatusManager *)callStatusManager
currentCall:(YLSSipCall *)currentCall;
```

Callback for call status in complex call scenarios

When there are complex call scenarios (such as call transfer or call waiting), use this method to call back the current call's status.

```
- (void)callStatusManager:(YLSCallStatusManager *)callStatusManager
currentCall:(YLSSipCall *)currentCall
callWaiting:(nullable YLSSipCall *)callWaitingCall
transferCall:(nullable YLSSipCall *)transferCall;
```

Call Detailed Record (CDR)

This topic introduces the functionalities and implementation methods related to Call Detailed Record (CDR) of 'Linkus SDK for macOS'.

Retrieve CDR records

```
- (NSArray<YLSHistory *> *)historys;
```

SDK notification callbacks

```
/**
 * CDR records change callback
 */
- (void)historyReload:(NSMutableArray<YLSHistory *> *)historys;
```

Linkus SDK for Windows

Linkus SDK for Windows Overview

Yeastar P-Series PBX System supports Linkus SDK, enabling you to integrate calling, CDR, and more Linkus UC Clients' functionalities into 3rd-party Windows applications. This topic describes the requirements, prerequisites, modules, integration process and features of 'Linkus SDK for Windows'.

Requirements

Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Modules

Linkus SDK for Windows decouples call logic module and User Interface(UI) module from the PBX Web calling component. Refer to the following table for more information about the available modules.

Module	Description	Resource Link
Linkus SDK for Windows Core	Provide core call functionalities of Linkus UC clients.	<ul style="list-style-type: none">• React Demo• Source Code
Linkus SDK for Windows UI	Provides a pre-integrated UI component that requires no additional coding.	<ul style="list-style-type: none">• Source Code

Integration process and Features

Integration process

1. [Enable Linkus SDK on Yeastar P-Series PBX System](#)
2. [Obtain Login Signature for 'Linkus SDK for Windows'](#)
3. [Integrate the core call functionality of Linkus SDK for Windows](#)
4. Optional: [Integrate the UI component of Linkus SDK for Windows](#)

'Linkus SDK for Windows Core' features

- ['Linkus SDK for Windows Core' Usage Example](#)

- [PBX Features \(PBXOperator\)](#)
- [Call Features \(PhoneOperator\)](#)
- [Call Status and Call Features \(Session\)](#)
- [Return Result \(Result\)](#)
- [Types and Interface of other Objects](#)
- [Tone Resource](#)

Release Notes - Linkus SDK for Windows

Version 1.0.12

Release date: October 11, 2023

- First release of **Linkus SDK for Windows**. By integrating the SDK into your Windows projects, you can quickly add calling, CDR, and more Linkus UC Clients' functionalities to your Windows applications.

Enable Linkus SDK

Before integrating **Linkus SDK for Windows** with your Windows project, you need to enable Linkus SDK on Yeastar P-Series PBX System.



Note:

After enabling Linkus SDK, Linkus Mobile Client's push notification will no longer take effect. If you want to continue receiving call-related push notifications on mobile devices, see ['Linkus SDK for Android' Overview](#) or ['Linkus SDK for iOS' Overview](#).

Requirements

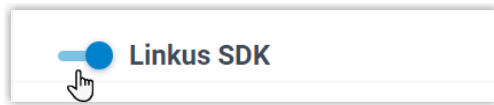
Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Procedure

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.

2. Enable **Linkus SDK**.



3. Click **Save**.

Result

You have enabled Linkus SDK, and you can [request the login signature from the PBX for authentication](#).

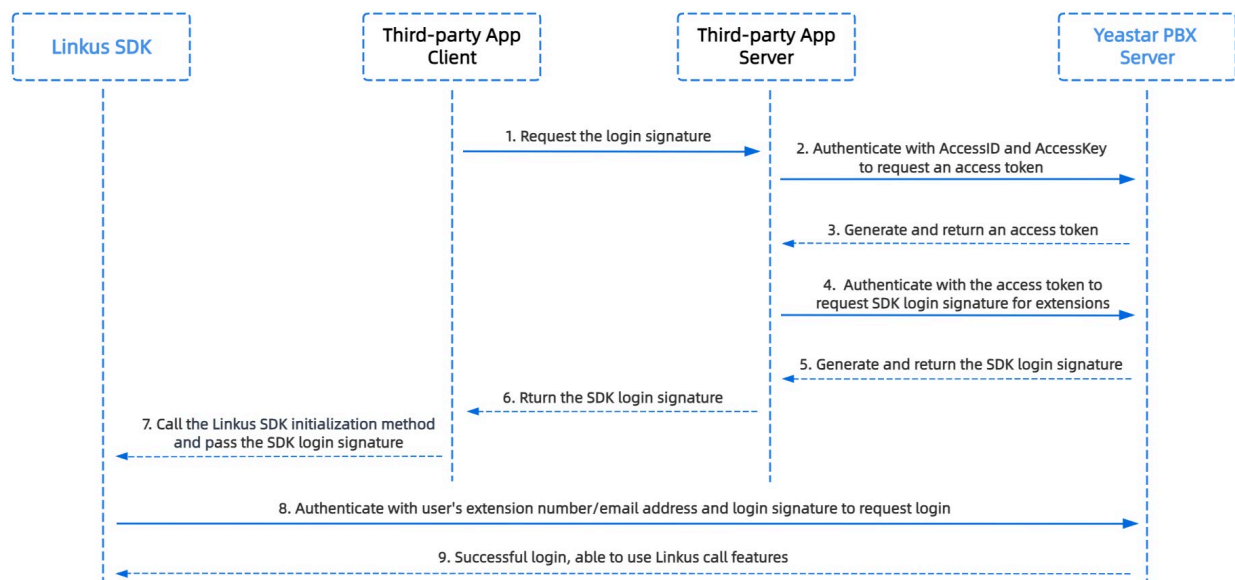
Obtain Login Signature for 'Linkus SDK for Windows'

When initializing 'Linkus SDK for Windows', users are required to authenticate with the SDK login signature. This topic describes how to request users' Linkus SDK login signatures from the PBX server via OpenAPI.

Prerequisites

You have [enabled Linkus SDK on Yeastar P-Series PBX System](#).

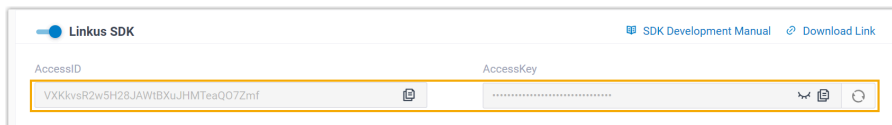
Authentication process



Step 1. Obtain the 'AccessID' and 'AccessKey' for Linkus SDK on PBX

Obtain the 'AccessID' and 'AccessKey' for Linkus SDK from Yeastar P-Series PBX System, which will be used for the third-party application to authenticate and connect with the PBX server.

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Note down the **AccessID** and **AccessKey**.



Step 2. Request an access token from PBX

On the third-party application server, use the 'AccessID' and 'AccessKey' to request an access token from the PBX via OpenAPI. Access token is used to verify an authenticated API call, enabling you to request Linkus SDK login signatures for extensions.

Request URL

```
POST {base_url}/openapi/v1.0/get_token
```



Note:

To learn about the API request structure, see [Request Structure](#).

Request Parameters

Parameter	Required	Type	Description
username	Yes	String	User name. Use the AccessID of Linkus SDK as the username.
password	Yes	String	Password. Use the AccessKey of Linkus SDK as the password.

Request example

```
POST /openapi/v1.0/get_token
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "VXKkvsR2w5H28JAWtBXuJHMTeeQO7Zmf",
```

```
"password": "Yq6yVsBceOZLhnuaGeMUG4U4qXXXXXX"
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> 0: Succeed. Non-zero value: Failed.
errmsg	String	Returned message. <ul style="list-style-type: none"> SUCCESS: Succeed. FAILURE: Failed.
access_token_expire_time	Integer	Access token expire time. (Unit : second)
access_token	String	Credential of calling API interfaces. All requests to call API interfaces must carry an access token.
refresh_token_expire_time	Integer	Refresh token expire time. (Unit : second)
refresh_token	String	Refresh token. <u>refresh_token</u> can be used to obtain new <u>access_token</u> and <u>refresh_token</u> .

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "access_token_expire_time": 1800,
  "access_token": "EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS",
  "refresh_token_expire_time": 86400,
  "refresh_token": "SCduGecwbG9jIusiS8FxFUVn3kf0Q9R8"
}
```

Step 3. Request login signatures for extensions

Use the access token to request login signatures for extensions from PBX server via Open-API.

Request URL

```
POST /openapi/v1.0/sign/create?access_token={access_token}
```

Request Parameters

Parameter	Required	Type	Description
username	Yes	string	Extension number or email address.
sign_type	Yes	string	Login signature type. Permitted value: <code>sdk</code>
expire_time	No	Integer	Expiration timestamp of the login signature. (Unit: second) <code>0</code> indicates no limitation on the validity duration of the login signature.

Request example

```
POST /openapi/v1.0/sign/create?access_token=EXZMpZA086mbrKm6rFtg
eb3rfcpC9uqS
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "1000",
  "sign_type": "sdk",
  "expire_time": 0
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"><code>0</code>: Succeed.Non-zero value: Failed
errmsg	String	Returned message. <ul style="list-style-type: none"><code>SUCCESS</code>: Succeed.<code>FAILURE</code>: Failed.
data	Array < Ext_Sign >	Linkus SDK login signatures for extensions.

Ext_Sign

Parameter	Type	Description
sign	String	Extension's Linkus SDK login signature.

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "data": {
    "sign": "ueb3rfcpsiS8FxFUZMpZAO86mbrKmVn3kf0Q9R8"
  }
}
```

Result

The third-party application server has obtained the Linkus SDK login signature for initializing 'Linkus SDK for Windows'. The login signature will be automatically returned to the third-party application client, and passed to 'Linkus SDK for Windows'.

What to do next

Use the login signature for authentication to [integrate and initialize 'Linkus SDK for Windows Core'](#).

Linkus SDK for Windows Core

Integrate Linkus SDK for Windows Core

Linkus SDK for Windows Core provides core call functionalities without UI components. This topic describes how to integrate **Linkus SDK for Windows Core** with your Windows project.

Prerequisites

- You have [enabled Linkus SDK on Yeastar P-Series PBX System](#).
- You have [obtained the Linkus SDK login signature](#).

Demo and Source code

Before the integration, we recommend that you try out the [React Demo of 'Linkus SDK for Windows Core'](#), and review the [source code](#) to have an overview of the framework and workflow of 'Linkus SDK for Windows Core'.

Supported module formats

'Linkus SDK for Windows Core' supports 4 module formats: **UMD**, **CJS**, **ESM**, and **IIFE**. You can choose a module format according to your needs.



Note:

If you prefer a smaller SDK bundle size or your Windows projects support **ESM**, we recommend that you import the **ESM** module.

Step 1. Import 'Linkus SDK for Windows Core'

1. Go to the [GitHub Repository of 'Linkus SDK for Windows Core'](#), and download 'Linkus SDK for Windows Core'.
2. Use either of the following methods to import 'Linkus SDK for Windows Core' to your project.
 - Use **npm** to install 'Linkus SDK for Windows Core'

```
npm install ys-webrtc-sdk-core
```

- Use **script** to import 'Linkus SDK for Windows Core'

```
<script src="./ys-webrtc.umd.js"></script>
```

Step 2. Initialize 'Linkus SDK for Windows Core'


Use the **init** method to initialize 'Linkus SDK for Windows Core'. After successful initialization, 2 instantiated Operator objects **PBXOperator** and **PhoneOperator**, and a method **destroy** are returned.

Name	Type	Description
PBXOperator	Object	This object contains methods and attributes related to the PBX, such as querying CDR records, logging out of user account, and more.
PhoneOperator	Object	This object contains methods and attributes related to calls, such as making a call, answering a call, ending a call, and more.
destroy	Method	This method is used to destroy 'Linkus SDK for Windows Core'.

Method

```
init({
    params //Refer to the following table for specific parameter
})
```

Parameters

Parameter	Type	Required	Description
username	string	Yes	Extension number or email address.
secret	string	Yes	Linkus SDK login signature.
pbxURL	URL string	Yes	The URL for accessing your PBX system, including the transfer protocol. For example, <code>https://192.168.1.1:8088</code> .
enableLog	boolean	No	Whether to enable log output and report error logs to PBX. Valid value: <ul style="list-style-type: none"> <code>true</code>: Enable <code>false</code>: Disable <div>  Note: This feature is enabled by default. </div>
reRegistryPhoneTimes	number	No	Define the number of retry attempts for SIP UA (User Agent) registration. By default, it is unlimited.
userAgent	WebPC" "WebClient	No	The User Agent in Asterisk, which indicates the type of Linkus client. The default value is WebClient .
deviceIds	{ cameraId?: string; microphoneId?: string; }	No	Define the IDs of the audio and video input devices, including the camera ID and microphone ID.
disableCallWaiting	boolean	No	Whether to disable call waiting. Valid value: <ul style="list-style-type: none"> <code>true</code>: Disable call waiting.

Parameter	Type	Required	Description
			<p>The call waiting time set on PBX will NOT take effect and the PBX only handles a single call.</p> <ul style="list-style-type: none"> • <code>false</code>: Enable call waiting.

Sample code

- Use **npm** to install and initialize 'Linkus SDK for Windows Core'.

```
import { init, on } from 'ys-webrtc-sdk-core';

init({
  username: '1000',
  secret: 'sdkshajgllliiaggskjh',
  pbxURL: 'https://192.168.1.1:8088'
})
  .then((operator) => {
    // Obtain the 'PhoneOperator' instance, 'PBXOperator'
    instance, and 'destroy' method
    const { phone, pbx, destroy } = operator;
  })
  .catch((error) => {
    console.log(error);
  });
```

- Use **script** to import and initialize 'Linkus SDK for Windows Core'.

```
<script src="./ys-webrtc.umd.js"></script>
<script>
  // After successful import, initialize 'Linkus SDK for We
  b Core' using the 'YSWebRTC' object
  YSWebRTC.init({
    username: '1000',
    secret: 'sdkshajgllliiaggskjh',
    pbxURL: 'https://192.168.1.1:8088',
  })
    .then((operator) => {
      // Obtain the 'PhoneOperator' instance, 'PBXOperator'
      instance, and 'destroy' method
      const { phone, pbx, destroy } = operator;
    })
    .catch((error) => {
      console.log(error);
    });
```

```
});  
</script>
```

Related information

['Linkus SDK for Windows Core' Usage Example](#)

Use Linkus SDK for Windows Core

'Linkus SDK for Windows Core' Usage Example

The simplified sample codes below demonstrate the general workflow for making and receiving calls via 'Linkus SDK for Windows Core'.

```
import { init } from 'ys-webrtc-sdk-core';

init({
  username: '1000',
  secret: 'sdkshajgllliiaggskjhf',
  pbxURL: 'https://192.168.1.1:8088'
})
  .then(operator => {
    // Obtain the 'PhoneOperator' instance, 'PBXOperator' instance, and
    'destroy' method.
    const { phone, pbx, destroy } = operator;

    // Create an RTC instance.
    phone.on('newRTCSession', ({callId,session})=>{
      const {status} = session

      // Listen for events in the session.
      session.on('confirmed', {callId,session})=>{
        // A call is successfully connected, the
        'session.status.callStatus' changes to 'talking'.
        // Update the user interface, start the call timer.
      })
    })

    // Listen for the 'startSession' events.
    phone.on('startSession',({callId,session})=>{
      const {status} = session
      if(status.communicationType === 'outbound') {
        // Outbound call.
      }
    })
  })
```

```

        // Update the user interface to display 'Calling',
        indicating that the callee side is ringing.
    }else{
        // Inbound call.
        // Update the user interface to display 'Connecting'.
    }

});

// Listen for Incoming call events.
phone.on('incoming', (callId,session)=>{
    const {status} = session
    // Pop up an incoming call dialog, displaying the caller's
    phone number and name.
    // ...
    // Click the 'Answer' button to trigger the 'answer' method and
    the 'startSession' event.
    phone.answer(status.number);
});

// After listening for events, start registering the SIP UA.
phone.start();

// ...
// Click the 'Call' button to call 1001.
phone.call('1001')

})
.catch(error => {
    console.log(error);
});

```

PBX Features (PBXOperator)

The **PBXOperator** is used to implement PBX-related features, such as querying CDR records and logging out. This topic describes the attributes, methods, and events related to PBX features (PBXOperator object).

Attributes

Attribute	Type	Description
username	string	Username (extension number or email address)
secret	string	Login signature requested from PBX.

Attribute	Type	Description
token	string	The access token requested from PBX.
url	URL	The URL for accessing your PBX system.
socket	WebSocker	The socket instance for monitoring PBX status changes and receiving notifications in real-time.
extensionNumber	string	Extension number.
extensionId	string	Extension ID.
extensionName	string	Extension name.

Methods

Methods overview

- [Initialization](#)
- [Listen for events](#)
- [Query CDR records](#)
- [Log out](#)
- [Destroy the PBXOperator object](#)

Initialization

The initialization method is used within Linkus SDK Core.



Note:

After successful initialization, any subsequent calls to this method will be invalid and a response `Promise.reject` will be returned.

Method	<code>init()</code>
Parameters	Null.
Return value	<code>Promise<Result></code>

Listen for events

Method	<code>on(eventName, listener)</code>
Parameters	<ul style="list-style-type: none"> • <code>eventName</code>: The event name. • <code>listener</code>: Callback function.

Return value	Null.
---------------------	-------

**Note:**

For more information about the events that you can listen for, see [Events](#).

Query CDR records

Method	<code>cdrQuery(params)</code>
Parameters	<pre> params: { page: number; //Required, define which page is displayed. size: number; //Required, define how many records per page. status?: number; // Optional, specify the call type. 0: All; 1: Inbound ca sortBy: 'time' 'id'; //Required, define the sorting field. orderBy?: 'desc' 'asc'; //Optional, define the display order. filter?: string null; //Optional, define the filtering criteria. } </pre>
Return value	Promise
Return value description	<pre> { errcode: 0, errmsg: "SUCCESS", personal_cdr_list: [//CDR list { id: 2546, //The sequence number of the record. date: "Today", //The date when the call was made or received. time: "17:21:39", //The time when the call was made or received. timestamp: 1686561699, //The timestamp of the time when the call w number: "1011", // Caller number number_type: "extension", extension: { //Extension information ext_id: 25, //Extension ID ext_num: "1011", //Extension number caller_id_name: "cwt1011", //Extension name photo: "", // Profile image ID status: 0, // Online status of the extension endpoints. 0: Off presence_status: "available", //Extension presence status presence_information: "", first_name: "wt1011", //First name last_name: "c", //Last name email_addr: "", //Email address mobile_number: "", //Mobile number enb_vm: 0, //Whether the voicemail is enabled. vm_total_count: 0, //The total number of voicemail messages vm_unread_count: 0, //The total number of unread voicemail mes visible: 0, </pre>


```

        im_id: "xxxxx",
        org_list_info: "",
        is_favorite: 0,
        title: "" //Job title
    },
    status: 3, //Call type. 0: All; 1: Inbound call; 2: Missed call; 3: Outbound call;
    talk_duration: 3, //The time between the call answered and the call ended.
    call_type: "Internal", // Communication type: 'Internal' or 'External'
    uniqueid: "1686561699.137", // The unique ID of the CDR.
    disposition: "ANSWERED", // Call status: 'ANSWERED', 'NO ANSWER', 'HANGUP', 'BUSY', 'NOT REACHED', 'UNREACHABLE', 'TRANSFER', 'TRANSFER TO VOICEMAIL', 'TRANSFER TO EXTENSION', 'TRANSFER TO EXTENSION AND VOICEMAIL', 'TRANSFER TO EXTENSION AND VOICEMAIL AND VOICEMAIL', 'TRANSFER TO EXTENSION AND VOICEMAIL AND VOICEMAIL AND VOICEMAIL',
    dcontext: "DLPN_DialPlan1010",
    caller_id: "1011", // Caller number
    clid: "\"cwt1010\" <1010>"
    }
  ],
  total_number: 63, //The total number of the queried CDR.
}

```

Log out

Method	<code>logout()</code>
Parameters	Null.
Return value	Promise

Destroy the PBXOperator object

Method	<code>destroy()</code>
Parameters	Null.
Return value	Null.

Events

Example code

```
pbx.on('eventName', (data) => {})
```

Runtime error event

Event name	<code>runtimeError</code>
Data	PBXResult
Report parameters	<pre>{ code: '',</pre>

```
msg: '',
}
```

**Note:**

Refer to the following table for reporting parameter descriptions of the event.

Report parameter description

code	msg	Description
-106	LINKUS_DISABLED	Linkus UC client is disabled.
-107	LOGGED_IN_ELSEWHERE	The extension has logged in elsewhere.
-108	EXTENSION_DELETED	The extension has been deleted.
-109	RE_LOGIN	The extension needs to log in again.
-110	SDK_PLAN_DISABLED	PBX plan is NOT Ultimate Plan (UP).

CDR changes event

When this event is triggered, you need to manually call the method to query CDR records.

Event name	<code>cdrChange</code>
Data	<code>cdrNotifyData</code>
Report parameters	<pre>{ ext_num: '1001', personal_cdr: { date:"", // The date when the call was made or received. time:"", // The time when the call was made or received. timestamp: 1693201380, // The timestamp of the time when the call was made or received. Accurate to the second. number: "1001", talk_duration:0 // The time between the call answered and the call ended. } }</pre>

Call Features (PhoneOperator)

The **PhoneOperator** object is used for managing calls. Each call is cached through the `sessions` attribute, which is a map containing different call instances (`Map<string, Session>`), and each call instance is represented as a `Session`. This topic describes the attributes, methods, and events related to call features (PhoneOperator object).

Attributes

Attributes	Type	Description
currentSessionID	string	ID of the current call (Session).
reRegistryPhoneTimes	number	Define the number of retry attempts for SIP UA (User Agent) registration.
deviceIds	{ cameraId?: string; microphoneId?: string;}	IDs of the audio and video input devices, including the camera ID and microphone ID.
sessions	Map<string, Session>	Caching <code>Map</code> object for calls (Session), with the <code>callId</code> as the key.
currentSession	Session	Current call.
isRegistered	boolean	Whether the SIP UA registration is successful.
recordPermissions	number	Call recording permission: <ul style="list-style-type: none"> • 0: No permission • 1: Permission to pause / resume call recording • 2: Permission to start / pause / resume call recording
incomingList	session[]	The list of the incoming calls.
isMaxCall	boolean	Whether the maximum number of concurrent calls has been reached.

Methods

Methods overview

• Listen for events	• Perform an attended transfer	• Terminate a call
-------------------------------------	--	------------------------------------

• Start registering SIP UA	• Hold a call	• Disconnect the SIP UA registration
• Re-register SIP UA	• Resume a call	• Retrieve all the calls (sessions attribute)
• Make a call	• Send DTMF	• Set the current call (currentSession)
• Reject a call	• Mute a call	• Retrieve the current call (currentSession)
• Answer a call	• Unmute a call	• Update the static properties of Session
• Hang up a call	• Start recording	• Destroy PhoneOperator object
• Perform a blind transfer	• Pause recording	

Listen for events

Method	<code>on(eventName:string,listener: (...args: any[]) => void)</code>
Parameters	<ul style="list-style-type: none"> • <code>eventName</code>: The event name. • <code>listener</code>: Callback function.
Return value	Null.



Note:

For more information about the events that you can listen for, see [Events](#).

Start registering SIP UA

After a successful SIP UA registration, users can make and receive calls.



Note:

You need to listen for desired events before registering SIP UA. Otherwise, some events may be missed.

Method	<code>start()</code>
Parameters	Null.

Return value	Null.
---------------------	-------

Re-register SIP UA




Note:

To avoid unexpected situations, use this method **ONLY** within the **PhoneOperator** object.

Method	<code>reRegister(authorizationUser: string, hal: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>authorizationUser</code>: User name. • <code>hal</code>: Password.
Return value	<code>this</code>

Make a call

Method	<code>call(number: string, option?: CallOptions, transferId?: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>number</code>: Callee number. • <code>option</code>: Optional. Specify userMedia constraints. • <code>transferId</code>: Optional. ID of the attended transfer call. <div style="border-left: 2px solid #007bff; padding-left: 10px; margin-top: 10px;">  <p>Note: If there is a value for <code>transferId</code>, it indicates that this is an attended transfer call, which is an asynchronous function.</p> </div>
Return value	<code>Promise<Result></code>

Reject a call

Method	<code>reject(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Answer a call

Method	<code>answer(callId: string, option?: CallOptions)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>option</code>: Optional. Specify userMedia constraints.
Return value	<code>Promise<Result></code>

Hang up a call

Method	<code>hangup(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Perform a blind transfer

Method	<code>blindTransfer(callId: string, number: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>number</code>: The number of the transfer target.
Return value type	<code>boolean</code>

Perform an attended transfer

Method	<code>attendedTransfer(callId: string, number: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>number</code>: The number of the transfer target.
Return value type	<code>boolean</code>

Hold a call

Method	<code>hold(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Resume a call

Method	<code>unhold(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Send DTMF

Method	<code>dtmf(callId: string, dtmf: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>dtmf</code>: String (0123456789*#).
Return value type	<code>boolean</code>

Mute a call

Method	<code>mute(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Unmute a call

Method	<code>unmute(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Start recording

Method	<code>startRecord(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Pause recording

Method	<code>pauseRecord(callId: string)</code>
---------------	--

Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Terminate a call

Method	<pre>terminate(callId: string, type: 'hangup' 'reject' 'terminate' = 'terminate')</pre>
Parameters	<ul style="list-style-type: none"> <code>callId</code>: The unique ID of the call. <code>type</code>: Types of call termination.
Return value type	<code>boolean</code>

Disconnect the SIP UA registration

Method	<pre>disconnect()</pre>
Parameters	Null.
Return value type	<code>boolean</code>

Retrieve all the calls (sessions attribute)

Retrieve the `sessions` attribute and return them as an array.

Method	<pre>getSession()</pre>
Parameters	Null.
Return value type	<code>boolean</code>

Set the current call (currentSession)

Method	<pre>setCurrentSession(callId: string)</pre>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Retrieve the current call (currentSession)

Method	<pre>getCurrentSession()</pre>
---------------	--------------------------------

Parameters	Null.
Return value	Session

Update the static properties of Session

Method	<pre>setSessionStaticStatus(callId: string, staticStatus: Partial<StaticCallStatus>, startManualModel?: boolean)</pre>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>staticStatus</code>: The static properties of Session include <code>name</code>, <code>avatar</code>, and <code>company</code>. • <code>startManualModel</code>: Optional. Whether to enable manual mode. If enabled, the static properties will NOT be automatically updated.
Return value type	<code>boolean</code>

Destroy PhoneOperator object

This method will cancel all the event subscriptions, stop the SIP UA instance, and delete all the sessions.

Method	<pre>destroy()</pre>
Parameters	Null.
Return value	Null.

Events

Event Name	Description	Report Parameter
connected	Connected to the SIP service.	Null
disconnected	Disconnected the SIP service.	Null
registered	SIP UA registration succeeded.	Null
registrationFailed	SIP UA registration failed.	<ul style="list-style-type: none"> • <code>code: number</code>: Error code. • <code>msg: string</code>: Error message.
newRTCSession	A new call instance (Session) is created.	<ul style="list-style-type: none"> • <code>callId: string</code>: The unique ID of the call.

Event Name	Description	Report Parameter
		<ul style="list-style-type: none"> • <code>session: Session</code>: The current call instance.
incoming	There is an incoming call.	<ul style="list-style-type: none"> • <code>callId: string</code>: The unique ID of the call. • <code>session: Session</code>: The current call instance.
startSession	A call instance (Session) is added to the map that stores calls (sessions).	<ul style="list-style-type: none"> • <code>callId: string</code>: The unique ID of the call. • <code>session: Session</code>: The current call instance.
recordPermissionsChange	The recording permission is changed.	<ul style="list-style-type: none"> • <code>0</code>: No permission. • <code>1</code>: Permission to pause / resume call recording. • <code>2</code>: Permission to start / pause / resume call recording.
deleteSession	A call instance (Session) is deleted.	<ul style="list-style-type: none"> • <code>callId: string</code>: The unique ID of the call. • <code>cause: string</code>: Reason for deleting the call instance.
isRegisteredChange	Whether the SIP UA registration status is changed.	<ul style="list-style-type: none"> • <code>true</code>: Registration status changed. • <code>false</code>: Registration status did NOT change.

Call Status and Call Features (Session)

The **Session** object is a call instance for managing call status and performing related operations during a call. This topic describes the attributes, methods, and events related to call status and call feature (Session object).

Attributes

Attributes	Type	Description
RTCSession	RTCSession	Call instance.
callReport	Report	Call quality report.
status	CallStatus	Call status, including name, number, profile image, status of call mute, etc.
incomingList	Session[]	Incoming call array.
timer	TimerType	Call timer.

Attributes	Type	Description
localStream	MediaStream	Local stream with video track only, no audio.
remoteStream	MediaStream	Remote stream with both audio and video tracks.

Method

Method Overview

• Stop the call timer	• Perform an attended transfer	• Start recording
• Listen for events	• Hold a call	• Pause recording
• Reject a call	• Resume a call	• Terminate a call
• Answer a call	• Send DTMF	• Update the call status
• Hang up a call	• Mute a call	• Update the static properties of Session
• Perform a blind transfer	• Unmute a call	• Destroy Session instance

Stop the call timer

Method	<code>stopTimer()</code>
Parameters	Null.
Return value	<code>this</code>

Listen for events

Method	<code>on(eventName:string,listener: (...args: any[]) => void)</code>
Parameters	<ul style="list-style-type: none"> <code>eventName</code>: The event name. <code>listener</code>: Callback function.
Return value	Null.



Note:

For more information about the events that you can listen for, see [Events](#).

Reject a call

This method is equivalent to the 'Reject a call' method of **PhoneOperator** object.

Method	<code>reject()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Answer a call

This method is equivalent to the 'Answer a call' method of **PhoneOperator** object.

Method	<code>answer(option?: CallOptions)</code>
Parameters	<code>option</code> : Optional. Specify userMedia constraints.
Return value	<code>Promise<Result></code>

Hang up a call

This method is equivalent to the 'Hang up a call' method of **PhoneOperator** object.

Method	<code>hangup()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Perform a blind transfer

This method is equivalent to the 'Perform a blind transfer' method of **PhoneOperator** object.

Method	<code>blindTransfer(number: string)</code>
Parameters	<code>number</code> : The number of the transfer target.
Return value type	<code>boolean</code>

Perform an attended transfer

This method is equivalent to the 'Perform an attended transfer' method of **PhoneOperator** object.

Method	<code>attendedTransfer(number: string)</code>
Parameters	<code>number</code> : The number of the transfer target.
Return value type	<code>boolean</code>

Hold a call

This method is equivalent to the 'Hold a call' method of **PhoneOperator** object.

Method	<code>hold()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Resume a call

This method is equivalent to the 'Resume a call' method of **PhoneOperator** object.

Method	<code>unhold()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Send DTMF

This method is equivalent to the 'Send DTMF' method of **PhoneOperator** object.

Method	<code>dtmf(dtmf: string)</code>
Parameters	<code>dtmf</code> : String (0123456789*#)
Return value type	<code>boolean</code>

Mute a call

This method is equivalent to the 'Mute a call' method of **PhoneOperator** object.

Method	<code>mute()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Unmute a call

This method is equivalent to the 'Unmute a call' method of **PhoneOperator** object.

Method	<code>unmute()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Start recording

This method is equivalent to the 'Start recording' method of **PhoneOperator** object.

Method	<code>startRecord()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Pause recording

This method is equivalent to the 'Pause recording' method of **PhoneOperator** object.

Method	<code>pauseRecord()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Terminate a call

This method is equivalent to the 'Terminate a call' method of **PhoneOperator** object.

Method	<code>terminate(type: 'hangup' 'reject' 'terminate' = 'terminate')</code>
Parameters	<code>type</code> : Type of call termination.
Return value type	<code>boolean</code>

Update the call status

Method	<code>setStatus(status: Partial<CallStatus>)</code>
Parameters	<code>status</code> : The call status object.
Return value	<code>this</code>

Update the static properties of Session

Method	<code>setStaticStatus(staticStatus: Partial<StaticCallStatus>, startManualModel?: boolean)</code>
Parameters	<ul style="list-style-type: none"> • <code>staticStatus</code>: The static properties of Session include <code>name</code>, <code>avatar</code>, and <code>company</code>. • <code>startManualModel</code>: Optional. Whether to enable manual mode. If enabled, the static properties will NOT be automatically updated.
Return value	<code>this</code>

Destroy Session instance

This method will cancel all the event subscriptions, and stop the RTCSession.

Method	<code>destroy()</code>
Parameters	Null.
Return value	Null.

Events

Event Name	Description	Report Parameter
callReport	The call quality report is updated, with a frequency of once every 3 seconds.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>callReport</code>: Report: Call quality report.
streamAdded	A media stream is added to the call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>communicationType</code>: "outbound" "inbound": Call type. <code>stream</code>: MediaStream: Media stream.
ended	The call is ended.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reason for ending the call.
failed	Failed to make a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reasons for the call failure. <code>code</code>: number: Error code.
clientError	Errors occurred on the client, resulting in the failure to make a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reason for the error.
reinvite	The other party performed attended transfer during a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
accepted	Received the success status response code 200 OK .	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
confirmed	The session is confirmed (received the response of ACK packet).	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
statusChange	Call status is changed.	<ul style="list-style-type: none"> <code>newStatus</code>: New call status. <code>oldStatus</code>: Previous call status.
staticStatusChange	The static properties of the call is changed.	<ul style="list-style-type: none"> <code>newStatus</code>: New static properties. <code>oldStatus</code>: Previous static properties.

Return Result (Result)

This topic introduces the structure and the subclasses of the 'Linkus SDK for Windows Core' return result (Result object).

Background information

Result structure


```
{
    code: '',
    msg: '',
}
```

Result subclass

Subclass	Description	Code Range
CommonResult	Common return result.	<ul style="list-style-type: none"> • Success code: 0 to 99 • Error code: -1 to -99
PBXResult	Return results related to the PBXOperator object.	<ul style="list-style-type: none"> • Success code: 100 to 199 • Error code: -100 to -199
PhoneResult	Return results related to the PhoneOperator object.	<ul style="list-style-type: none"> • Success code: 200 to 299 • Error code: -200 to -299

CommonResult

Error codes (COMMON_ERROR)

code	msg	Description
-1	UNKNOWN_ERROR	Unknown error.
-2	INVALID_PBX_URL	Invalid PBX URL.
-3	PBX_URL_NOT_HTTPS	The transfer protocol of the PBX URL is not HTTPS.
-4	GET_PRODUCT_FAILED	Failed to retrieve the PRODUCT interface of PBX .

Success code (COMMON_SUCCESS)

code	msg	Description
0	SUCCESS	Success.

PBXResult

Error codes (PBX_ERROR)

code	msg	Description
-100	UNKNOWN_ERROR	Unknown error.
-101	REGISTRY_FAILED	SIP UA registration failed.
-102	PBX_NETWORK_ERROR	API request failed due to the network error on the client-side.
-103	PBX_API_ERROR	API request failed due to the server-side error.
-104	GET_PERSONAL_NOT_FOUND_DATA	PBX did NOT return extension information.
-105	PBX_ALREADY_INITIALIZED	The PBX object has already been initialized and can NOT be initialized again.
-106	LINKUS_DISABLED	Linkus UC client is disabled.
-107	LOGGED_IN_ELSEWHERE	This extension has logged in elsewhere.
-108	EXTENSION_DELETED	The extension has been deleted.
-109	RE_LOGIN	The extension needs to log in again.
-110	SDK_PLAN_DISABLED	PBX plan is not Ultimate Plan (UP).

Success code (PBX_SUCCESS)

code	msg	Description
100	SUCCESS	Success.

PhoneResult

Error codes (PHONE_ERROR)

code	msg	Description
-200	UNKNOWN_ERROR	Unknown error.
-201	REGISTRY_FAILED	SIP UA registration failed.
-202	GET_AGREE_CHROME_USER_MEDIA_ROLE_ERROR	Failed to retrieve media stream (Browser authorization is not granted).
-203	GET_LOCAL_STREAM_ERROR	Failed to retrieve local media stream.
-204	RE_REGISTRY_MAX_LIMIT_TIMES	Reached the maximum number of the allowed SIP UA re-registration attempts.

code	msg	Description
-205	MAX_LIMIT_CALL	Reached the maximum number of concurrent calls.
-206	GET_LOCAL_MEDIA_INFO_ERROR	Failed to access local media device.
-207	ATTENDED_PARENT_NOT_FOUND	Failed to find the parent call of the attended transfer call.
-208	CALL_TOO_MANY_TIMES	Reached the limit of outgoing calls within one second.
-209	INVALID_NUMBER	Invalid number.
-210	CURRENT_CALL_HAS_NOT_CONNECTED	There is currently an unconnected call.
-211	NOT_FOUND_CALL_ID	Failed to find the call ID.
-290	NOT_FOUND_AUDIO_INPUT_DEVICE	Failed to find the audio input device.
-291	NOT_FOUND_VIDEO_INPUT_DEVICE	Failed to find the video input device.

Success code

code	msg	Description
200	SUCCESS	Success.

Types and Interface of other Objects

This topic introduces the types and interface of other objects used by 'Linkus SDK for Windows Core'.

Report object

Call quality report object.

```
interface Report {
    lksTimestamp: string; // The timestamp of the time when the call was made or received.
    lksDate: string; // The time when the call was made or received. Time format: YYYY-MM-DD hh:mm:ss
    callId: string; // The unique ID of the call.
    extension: string; // Extension number.
    iceResult: string; // The ICE candidate type. local: host; public: srflx or prflx; turn: relay.
```

```

    lksNetworkType: string; // Network type.
    lksDeviceType: string; // The client type. Currently only Web client is
    available.
    lksDuration: string; // Correspond to the 'totalSamplesDuration'
    attribute.
    lksAudioCodec: string; // Audio codec.
    lksAudioRx: string; // The audio packets received that correspond to
    the 'packetsReceived' attribute.
    lksAudioRxLost: string; // The audio packet loss rate that corresponds
    to the 'packetsLost' attribute. Formula: lksAudioRxLost=(packetsLost /
    (packetsLost + packetsReceived)).toFixed(6).
    lksAudioRxMaxjitter: string; // The maximum audio jitter on the
    receiver.
    lksAudioRxAvgjitter: string; // The average audio jitter that
    corresponds to "Jitter*1000".
    lksAudioTx: string; // The audio packet sent.
    lksAudioTxLost: string; // Audio packet loss rate on the sender.
    lksAudioTxMaxjitter: string; // The maximum audio jitter on the sender.
    lksAudioTxAvgjitter: string; // The average audio jitter on the sender.
    lksVideoCodec: string; // Video codec
    lksVideoRx: string; // The video packet received.
    lksVideoRxLost: string; // The video packet loss rate that corresponds
    to the 'packetsLost' attribute. Formula: lksVideoRxLost=(packetsLost /
    (packetsLost + packetsReceived)).toFixed(6).
    lksVideoRxMaxjitter: string; //The maximum video jitter on the
    receiver.
    lksVideoRxAvgjitter: string; // The average video jitter on the
    receiver.
    lksVideoTx: string; // The video packet sent.
    lksVideoTxLost: string; // Video packet loss rate on the sender.
    lksVideoTxMaxjitter: string; // The maximum video jitter on the sender.
    lksVideoTxAvgjitter: string; // The average video jitter on the sender.
}

```

StaticCallStatus object

Static properties of a call.

```

interface StaticCallStatus {
    name: string; // Caller name
    avatar: string; // Profile image
    company: string; // Company name
}

```

CallStatus object

The call status object.

```
interface CallStatus extends StaticCallStatus{
    number: string; // Phone number or extension number.
    callId: string; // The unique ID of the call.
    communicationType: 'outbound' | 'inbound'; // Call type
    isVideo: boolean; //Whether it is a video call.
    isRing: boolean;
    isHold: boolean;
    isMute: boolean;
    callStatus: 'ringing' | 'calling' | 'talking' | 'connecting';
    callStartTime: number; // The time when the call is answered.
    recordStatus: 'stop' | 'recording' | 'pause'; // Recording status.
    stop: Not in recording; recording: Recording,; pause: Recording is paused.
    isTransfer: boolean; // Whether it is an attended transfer call.
    transferParent?: TransferParentType; // When it's an attended transfer
    call, specify the parent call.
    isConference: boolean; // Whether it is an audio conference call.
}

type TransferParentType = {
    // Parent call in the attended transfer call.
    callId: string; // The unique ID of the call.
    avatar: string; // Profile image.
    name: string; // Name of the transfer target.
    number: string; // Number of the transfer target.
    callDuration: number; // The time between the call answered and the
    call ended.
    holdDuration: number; // The duration of the call being held.
}
```

TimerType type

Types of call timing.

```
type TimerType = {
    ringDuration: number; // The time between the call started and the call
    answered.
    callingDuration: number; // The time between the call dialed and the
    call answered.
    callDuration: number; // The time between the call answered and the
    call ended.
    holdDuration: number; // The duration of the call being held.
}
```

Tone Resource

This topic introduces the tone resources provided by 'Linkus SDK for Windows Core' and the usage methods.

Background information

The **assets** directory of 'Linkus SDK for Windows Core' project files contains tone resources in the following forms:

- **Original audio files:** Stored in the **sounds** folder.
- **base64-encoded strings:** Stored in the **sound.js** file.

Refer to the following table for the available tone resources.

Name	Description	Name	Description
Ring	Incoming call ringtone	DTMF04	Keypad 4 tone
Callend	Call end tone	DTMF05	Keypad 5 tone
Callwaiting	Call waiting tone	DTMF06	Keypad 6 tone
ringback	Ringback tone	DTMF07	Keypad 7 tone
DTMF00	Keypad 0 tone	DTMF08	Keypad 8 tone
DTMF01	Keypad 1 tone	DTMF09	Keypad 9 tone
DTMF02	Keypad 2 tone	DTMFStar	Keypad * tone
DTMF03	Keypad 3 tone	DTMFPound	Keypad # tone

Use tone resources

'Linkus SDK for Windows Core' supports to use tone resources in the following methods:

- Use the original audio files of tone resources.
- Use the base64-encoded strings of the tone resources, as shown in the sample code below.

```
import sounds from '/assets/sounds';
const { Ring, Callend } = sounds;
const audio = new Audio(Ring);
audio.play();

// Change the value of "audio.src" to play different tones.
audio.src = Callend;
audio.play();
```

Linkus SDK for Windows UI

Integrate Linkus SDK for Windows UI

Linkus SDK for Windows UI provides a pre-integrated UI component that requires no additional coding, you can integrate it with your Web projects to implement call functionalities with a user interface based on the integration of 'Linkus SDK for Windows Core'.

Prerequisites

- You have [obtained the Linkus SDK login signature](#).
- You have [integrated 'Linkus SDK for Windows Core'](#).

Step 1. Import 'Linkus SDK for Windows UI'

1. Go to the [GitHub Repository of 'Linkus SDK for Windows UI'](#), and download 'Linkus SDK for Windows UI'.
2. Use either of the following methods to import 'Linkus SDK for Windows UI' to your project.

- Use **npm** to install 'Linkus SDK for Windows UI'

```
npm install ys-webrtc-sdk-ui --save
```

- Use **script** to import 'Linkus SDK for Windows UI'

```
<script src="./ys-webrtc-sdk-ui.js"></script>
```

Step 2. Initialize 'Linkus SDK for Windows UI'


Use the `init` method to initialize 'Linkus SDK for Windows UI' and render the UI components.

Method

```
init (container, {rtcOption})
```

Parameters

- **container**: Used to render UI components, and the data type is `HTMLElement`.
- **rtcOption**: Initialization options. Refer to the following table for specific parameters.

Parameter	Type	Required	Description
username	string	Yes	Extension number or email address.
secret	string	Yes	Linkus SDK login signature.
pbxURL	URL string	Yes	The URL for accessing your PBX system, including the transfer protocol. For example, <code>https://192.168.1.1:8088</code> .
enableLog	boolean	No	Whether to enable log output and report error logs to PBX. Valid value: <ul style="list-style-type: none"> ◦ <code>true</code>: Enable ◦ <code>false</code>: Disable <div>  Note: This feature is enabled by default. </div>
reRegistryPhoneTimes	number	No	Define the number of retry attempts for SIP UA (User Agent) registration. By default, it is unlimited.
deviceIds	{ cameraId?: string; microphoneId?: string; }	No	Define the IDs of the audio and video input devices, including the camera ID and microphone ID.
incomingOption	{ style?: React.CSSProperties; class?: string; }	No	Adjust the styling of the incoming call component.
dialPanelOption	{ style?: React.CSSProperties; class?: string; }	No	Adjust the styling of the dialpad component.
sessionOption	SessionOption	No	Adjust the position and size of the call window component.
hiddenIncomingComponent	boolean	No	Hide the incoming call component.

Parameter	Type	Required	Description
hiddenDialPanelComponent	boolean	No	Hide the dialpad component.
disableCallWaiting	boolean	No	Whether to disable call waiting. Valid value: <ul style="list-style-type: none"> ◦ <code>true</code>: Disable call waiting. <p>The call waiting time set on PBX will NOT take effect and the PBX only handles a single call.</p> <ul style="list-style-type: none"> ◦ <code>false</code>: Enable call waiting.
intl	<code>{ local: string; messages: Record<string, string> }</code>	No	Internationalization (multilingual) settings.

SessionOption description

```

type SessionOption = {
  sessionSetting?: {
    width?: number;
    height?: number;
    minWidth?: number;
    minHeight?: number;
    x?: number;
    y?: number;
  };
  style?: React.CSSProperties;
  class?: string;
}

```

intl (Internationalization) description

- `local`: Name of the current language, connected with a `-`. For example, `en-US`, `zh-CN`.
- `message`: Configurable text content, which is presented in key-value pairs. Refer to the following content for specific configuration items and their default values.

```
{
```

```

    "common.cancel": "Cancel",
    "common.confirm": "Confirm",
    "dial_panel.input.placeholder":
    "Please input number",
    "dial_panel.tip.connect_failed":
    "Failed to connect to server, you cannot
    initiate or answer a call. Trying to
    reconnect to the server.",
    "incoming.btn.hang_up": "Hang up",
    "incoming.btn.audio": "Audio",
    "session.calling": "Calling...",
    "session.ringing": "Ringing...",
    "session.talking": "Talking...",
    "session.connecting": "Connecting...",
    "session.hang_up": "End Call",
    "session.new_call": "New call",
    "session.record": "Record",
    "session.mute": "Mute",
    "session.hold": "Hold",
    "session.resume": "Resume",
    "session.dialpad": "Dialpad",
    "session.transfer": "Transfer",
    "session.attended_transfer": "Attended
    Transfer",
    "session.blind_transfer": "Blind
    Transfer",
    "session.error.client_error": "Client
    Error: {0}",
    "session.tip.recording": "Recording
    the Audio...",
    "session.tip.pause": "The recording is
    paused.",
    "error.code_200": "Unknown Error.",
    "error.code_202": "No available
    communication device found.(no
    permissions)",
    "error.code_205": "Call failed. Cannot
    process more new calls now.",
    "error.code_206": "No available
    communication device found.",
    "error.code_207": "Attended Transfer
    Failed.",
    "error.code_208": "Call failed. Called
    too many times.",

```

```

    "error.code_209": "Call failed.
    Invalid Number.",
    "error.code_210": "Operation failed
    with pending calls.",
    "error.code_211": "Answer failed",
    "error.code_290": "No available
    microphone found.",
    "error.code_291": "No available camera
    found."
  }
}

```

Sample code

Use 'npm' to install and initialize 'Linkus SDK for Windows UI'

```

import { init } from 'ys-webrtc-sdk-ui';
import 'ys-webrtc-sdk-ui/lib/ys-webrtc-sdk-ui.css';
const container =
  document.getElementById('container');
// initialization
init(container, {
  username: '1000',
  secret: 'sdkshajgllliiaggskjhf',
  pbxURL: 'https://192.168.1.1:8088'
}).then(data => {
  // Obtain the exposed instances for additional
  business needs
  const { phone, pbx, destroy, on } = data;
  // ...
}).catch(err=>{
  console.log(err)
})

```

Use 'script' to import and initialize 'Linkus SDK for Web UI'

```

<!-- Loading styles -->
<link rel="stylesheet" href="ys-webrtc-sdk-ui.css">

<div id="test"></div>
<!-- Loading Yeastar WebRTC SDK UI-->
<script src="./ys-webrtc-sdk-ui.js"></script>
<script>
  const test = document.getElementById('test');
  // Initialize Yeastar WebRTC SDK UI with the
  'YSWebRTCUI' object.
  window.YSWebRTCUI.init(test, {

```

```
username: '1000',  
secret: 'sdkshajglllliiaggskjhf',  
pbxURL: 'https://192.168.1.1:8088'  
})  
  .then(data => {  
    const { phone, pbx } = data;  
  })  
  .catch(error => {  
    console.log(error);  
  });  
</script>
```

Result

You have integrated and initialized 'Linkus SDK for Windows UI', two instantiated Operator objects are returned:

- **PBXOperator**: Contains methods and attributes related to the PBX, such as querying CDR records, logging out, and more.
- **PhoneOperator**: Contains methods and attributes related to calls, such as making a call, answering a call, ending a call, and more.

Linkus SDK for Web

Linkus SDK for Web Overview

Yeastar P-Series PBX System supports Linkus SDK, enabling you to integrate calling, CDR, and more Linkus UC Clients' functionalities into 3rd-party Web applications. This topic describes the requirements, modules, integration process, and features of 'Linkus SDK for Web'.

Requirements

Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Modules

Linkus SDK for Web decouples call logic module and User Interface(UI) module from the PBX Web calling component. Refer to the following table for more information about the available modules.

Module	Description	Resource Link
Linkus SDK for Web Core	Provide core call functionalities of Linkus Web Client.	<ul style="list-style-type: none">• React Demo• Source Code
Linkus SDK for Web UI	Provides a pre-integrated UI component that requires no additional coding.	<ul style="list-style-type: none">• Source Code

Integration Process and Features

Integration Process

1. [Enable Linkus SDK on Yeastar P-Series PBX System](#)
2. [Obtain Login Signature for 'Linkus SDK for Web'](#)
3. [Integrate core call functionalities of 'Linkus SDK for Web'](#)
4. **Optional:** [Integrate the UI component of 'Linkus SDK for Web'](#)

'Linkus SDK for Web Core' features

- ['Linkus SDK for Web Core' Usage Example](#)

- [PBX Features \(PBXOperator\)](#)
- [Call Features \(PhoneOperator\)](#)
- [Call Status and Call Features \(Session\)](#)
- [Return Result \(Result\)](#)
- [Types and Interface of other Objects](#)
- [Tone Resource](#)

Release Notes - Linkus SDK for Web

Version 1.0.12

Release date: October 11, 2023

- First release of **Linkus SDK for Web**. By integrating the SDK into your Web projects, you can quickly add calling, CDR, and more Linkus UC Clients' functionalities to your Web applications.

Enable Linkus SDK

Before integrating 'Linkus SDK for Web' with your Web project, you need to enable Linkus SDK on Yeastar P-Series PBX System.



Note:

After enabling Linkus SDK, Linkus Mobile Client's push notification will no longer take effect. If you want to continue receiving call-related push notifications on mobile devices, see ['Linkus SDK for Android' Overview](#) or ['Linkus SDK for iOS' Overview](#).

Requirements

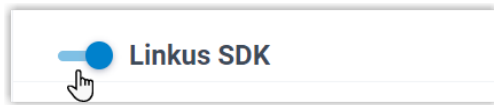
Make sure that your PBX server meets the following requirements:

- **Firmware:** 37.12.0.23 or later
- **Plan:** Ultimate Plan (UP)

Procedure

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.

2. Enable **Linkus SDK**.



3. Click **Save**.

Result

You have enabled Linkus SDK, and you can [request the login signature from the PBX for authentication](#).

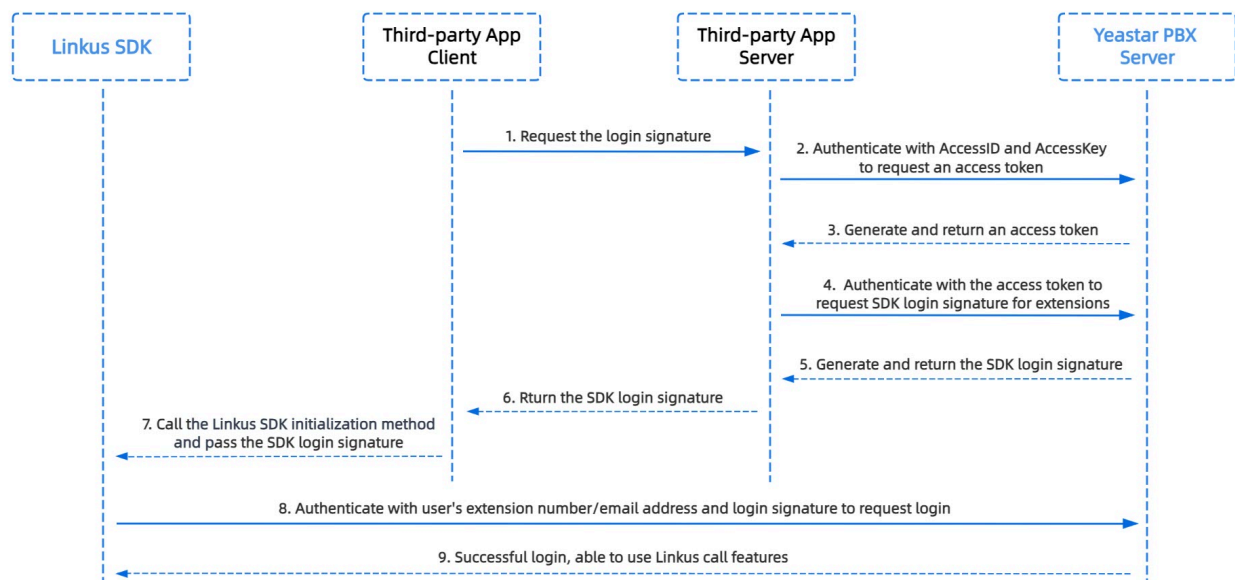
Obtain Login Signature for 'Linkus SDK for Web'

When initializing 'Linkus SDK for Web', users are required to authenticate with the SDK login signature. This topic describes how to request users' Linkus SDK login signatures from the PBX server via OpenAPI.

Prerequisites

You have [enabled Linkus SDK on Yeastar P-Series PBX System](#).

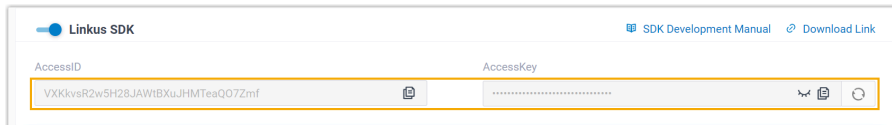
Authentication process



Step 1. Obtain the 'AccessID' and 'AccessKey' for Linkus SDK on PBX

Obtain the 'AccessID' and 'AccessKey' for Linkus SDK from Yeastar P-Series PBX System, which will be used for the third-party application to authenticate and connect with the PBX server.

1. Log in to PBX web portal, go to **Integrations > Linkus SDK**.
2. Note down the **AccessID** and **AccessKey**.



Step 2. Request an access token from PBX

On the third-party application server, use the 'AccessID' and 'AccessKey' to request an access token from the PBX via OpenAPI. Access token is used to verify an authenticated API call, enabling you to request Linkus SDK login signatures for extensions.

Request URL

```
POST {base_url}/openapi/v1.0/get_token
```



Note:

To learn about the API request structure, see [Request Structure](#).

Request Parameters

Parameter	Required	Type	Description
username	Yes	String	User name. Use the AccessID of Linkus SDK as the username.
password	Yes	String	Password. Use the AccessKey of Linkus SDK as the password.

Request example

```
POST /openapi/v1.0/get_token
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "VXKkvsR2w5H28JAWtBXuJHMTeeQO7Zmf",
```



```

"password": "Yq6yVsBceOZLhnuaGeMUG4U4qXXXXXX"
}

```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"> 0: Succeed. Non-zero value: Failed.
errmsg	String	Returned message. <ul style="list-style-type: none"> SUCCESS: Succeed. FAILURE: Failed.
access_token_expire_time	Integer	Access token expire time. (Unit: second)
access_token	String	Credential of calling API interfaces. All requests to call API interfaces must carry an access token.
refresh_token_expire_time	Integer	Refresh token expire time. (Unit: second)
refresh_token	String	Refresh token. <u>refresh_token</u> can be used to obtain new <u>access_token</u> and <u>refresh_token</u> .

Response example

```

HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "access_token_expire_time": 1800,
  "access_token": "EXZMpZA086mbrKm6rFtgeb3rfcpC9uqS",
  "refresh_token_expire_time": 86400,
  "refresh_token": "SCduGecwbG9jIusiS8FxFUVn3kf0Q9R8"
}

```

Step 3. Request login signatures for extensions

Use the access token to request login signatures for extensions from PBX server via Open-API.

Request URL

```
POST /openapi/v1.0/sign/create?access_token={access_token}
```

Request Parameters

Parameter	Required	Type	Description
username	Yes	string	Extension number or email address.
sign_type	Yes	string	Login signature type. Permitted value: <code>sdk</code>
expire_time	No	Integer	Expiration timestamp of the login signature. (Unit: second) <code>0</code> indicates no limitation on the validity duration of the login signature.

Request example

```
POST /openapi/v1.0/sign/create?access_token=EXZMpZA086mbrKm6rFtg
eb3rfcpC9uqS
Host: 192.168.5.150:8088
Content-Type: application/json
{
  "username": "1000",
  "sign_type": "sdk",
  "expire_time": 0
}
```

Response parameters

Parameter	Type	Description
errcode	Integer	Returned error code. <ul style="list-style-type: none"><code>0</code>: Succeed.Non-zero value: Failed
errmsg	String	Returned message. <ul style="list-style-type: none"><code>SUCCESS</code>: Succeed.<code>FAILURE</code>: Failed.
data	Array < Ext_Sign >	Linkus SDK login signatures for extensions.

Ext_Sign

Parameter	Type	Description
sign	String	Extension's Linkus SDK login signature.

Response example

```
HTTP/1.1 200 OK
{
  "errcode": 0,
  "errmsg": "SUCCESS",
  "data": {
    "sign": "ueb3rfcpsiS8FxFUZMpZAO86mbrKmVn3kf0Q9R8"
  }
}
```

Result

The third-party application server has obtained the Linkus SDK login signature for initializing 'Linkus SDK for Web'. The login signature will be automatically returned to the third-party application client, and passed to 'Linkus SDK for Web'.

What to do next

Use the login signature for authentication to [integrate and initialize 'Linkus SDK for Web Core'](#).

Linkus SDK for Web Core

Integrate Linkus SDK for Web Core

'Linkus SDK for Web Core' provides core call functionalities without UI components. This topic describes how to integrate 'Linkus SDK for Web Core' with your Web project.

Prerequisites

- You have [enabled Linkus SDK on Yeastar P-Series PBX System](#).
- You have [obtained the Linkus SDK login signature](#).

Demo and Source code

Before the integration, we recommend that you try out the [React Demo of 'Linkus SDK for Web Core'](#), and review the [source code](#) to have an overview of the framework and workflow of 'Linkus SDK for Web Core'.

Supported module formats

'Linkus SDK for Web Core' supports 4 module formats: **UMD**, **CJS**, **ESM**, and **IIFE**. You can choose a module format according to your needs.



Note:

If you prefer a smaller SDK bundle size or your Web project supports **ESM**, we recommend that you import the **ESM** module.

Step 1. Import 'Linkus SDK for Web Core'

1. Go to the [GitHub Repository of 'Linkus SDK for Web Core'](#), and download 'Linkus SDK for Web Core'.
2. Use either of the following methods to import 'Linkus SDK for Web Core' to your Web project.
 - Use **npm** to install 'Linkus SDK for Web Core'

```
npm install ys-webrtc-sdk-core
```

- Use **script** to import 'Linkus SDK for Web Core'

```
<script src="./ys-webrtc.umd.js"></script>
```

Step 2. Initialize 'Linkus SDK for Web Core'


Use the **init** method to initialize 'Linkus SDK for Web Core'. After successful initialization, 2 instantiated Operator objects **PBXOperator** and **PhoneOperator**, and a method **destroy** are returned.

Name	Type	Description
PBXOperator	Object	This object contains methods and attributes related to the PBX, such as querying CDR records, logging out of user account, and more.
PhoneOperator	Object	This object contains methods and attributes related to calls, such as making a call, answering a call, ending a call, and more.
destroy	Method	This method is used to destroy 'Linkus SDK for Web Core'.

Method

```
init({
  params //Refer to the following table for specific parameter
})
```

Parameters

Parameter	Type	Required	Description
username	string	Yes	Extension number or email address.
secret	string	Yes	Linkus SDK login signature.
pbxURL	URL string	Yes	The URL for accessing your PBX system, including the transfer protocol. For example, <code>https://192.168.1.1:8088</code> .
enableLog	boolean	No	Whether to enable log output and report error logs to PBX. Valid value: <ul style="list-style-type: none"> • <code>true</code>: Enable • <code>false</code>: Disable <div>  Note: This feature is enabled by default. </div>
reRegistryPhoneTimes	number	No	Define the number of retry attempts for SIP UA (User Agent) registration. By default, it is unlimited.
userAgent	WebPC" "WebClient	No	The User Agent in Asterisk, which indicates the type of Linkus client. The default value is WebClient .
deviceIds	{ cameraId?: string; microphoneId?: string; }	No	Define the IDs of the audio and video input devices, including the camera ID and microphone ID.
disableCallWaiting	boolean	No	Whether to disable call waiting. Valid value: <ul style="list-style-type: none"> • <code>true</code>: Disable call waiting.

Parameter	Type	Required	Description
			<p>The call waiting time set on PBX will NOT take effect and the PBX only handles a single call.</p> <ul style="list-style-type: none"> • <code>false</code>: Enable call waiting.

Sample code

- Use **npm** to install and initialize 'Linkus SDK for Web Core'.

```
import { init, on } from 'ys-webrtc-sdk-core';

init({
  username: '1000',
  secret: 'sdkshajgllliiaggskjh',
  pbxURL: 'https://192.168.1.1:8088'
})
  .then((operator) => {
    // Obtain the 'PhoneOperator' instance, 'PBXOperator'
    instance, and 'destroy' method
    const { phone, pbx, destroy } = operator;
  })
  .catch((error) => {
    console.log(error);
  });
```

- Use **script** to import and initialize 'Linkus SDK for Web Core'.

```
<script src="./ys-webrtc.umd.js"></script>
<script>
  // After successful import, initialize 'Linkus SDK for Web
  Core' using the 'YSWebRTC' object
  YSWebRTC.init({
    username: '1000',
    secret: 'sdkshajgllliiaggskjh',
    pbxURL: 'https://192.168.1.1:8088',
  })
    .then((operator) => {
      // Obtain the 'PhoneOperator' instance, 'PBXOperator'
      instance, and 'destroy' method
      const { phone, pbx, destroy } = operator;
    })
    .catch((error) => {
      console.log(error);
    });
```

```
});  
</script>
```

Related information

['Linkus SDK for Web Core' Usage Example](#)

Use Linkus SDK for Web Core

'Linkus SDK for Web Core' Usage Example

The simplified sample codes below demonstrate the general workflow for making and receiving calls via 'Linkus SDK for Web Core'.

```
import { init } from 'ys-webrtc-sdk-core';

init({
  username: '1000',
  secret: 'sdkshajgllliiaggskjhf',
  pbxURL: 'https://192.168.1.1:8088'
})
  .then(operator => {
    // Obtain the 'PhoneOperator' instance, 'PBXOperator' instance, and
    'destroy' method.
    const { phone, pbx, destroy } = operator;

    // Create an RTC instance.
    phone.on('newRTCSession', ({callId,session})=>{
      const {status} = session

      // Listen for events in the session.
      session.on('confirmed', {callId,session})=>{
        // A call is successfully connected, the
        'session.status.callStatus' changes to 'talking'.
        // Update the user interface, start the call timer.
      })
    })

    // Listen for the 'startSession' events.
    phone.on('startSession',({callId,session})=>{
      const {status} = session
      if(status.communicationType === 'outbound') {
        // Outbound call.
      }
    })
  })
```

```

        // Update the user interface to display 'Calling',
        indicating that the callee side is ringing.
    }else{
        // Inbound call.
        // Update the user interface to display 'Connecting'.
    }

});

// Listen for Incoming call events.
phone.on('incoming', (callId,session)=>{
    const {status} = session
    // Pop up an incoming call dialog, displaying the caller's
    phone number and name.
    // ...
    // Click the 'Answer' button to trigger the 'answer' method and
    the 'startSession' event.
    phone.answer(status.number);
});

// After listening for events, start registering the SIP UA.
phone.start();

// ...
// Click the 'Call' button to call 1001.
phone.call('1001')

})
.catch(error => {
    console.log(error);
});

```

PBX Features (PBXOperator)

The **PBXOperator** is used to implement PBX-related features, such as querying CDR records and logging out. This topic describes the attributes, methods, and events related to PBX features (PBXOperator object).

Attributes

Attribute	Type	Description
username	string	Username (extension number or email address)
secret	string	Login signature requested from PBX.

Attribute	Type	Description
token	string	The access token requested from PBX.
url	URL	The URL for accessing your PBX system.
socket	WebSocker	The socket instance for monitoring PBX status changes and receiving notifications in real-time.
extensionNumber	string	Extension number.
extensionId	string	Extension ID.
extensionName	string	Extension name.

Methods

Methods overview

- [Initialization](#)
- [Listen for events](#)
- [Query CDR records](#)
- [Log out](#)
- [Destroy the PBXOperator object](#)

Initialization

The initialization method is used within Linkus SDK Core.



Note:

After successful initialization, any subsequent calls to this method will be invalid and a response `Promise.reject` will be returned.

Method	<code>init()</code>
Parameters	Null.
Return value	<code>Promise<Result></code>

Listen for events

Method	<code>on(eventName, listener)</code>
Parameters	<ul style="list-style-type: none"> • <code>eventName</code>: The event name. • <code>listener</code>: Callback function.

Return value	Null.
---------------------	-------

**Note:**

For more information about the events that you can listen for, see [Events](#).

Query CDR records

Method	<code>cdrQuery(params)</code>
Parameters	<pre> params: { page: number; //Required, define which page is displayed. size: number; //Required, define how many records per page. status?: number; // Optional, specify the call type. 0: All; 1: Inbound ca sortBy: 'time' 'id'; //Required, define the sorting field. orderBy?: 'desc' 'asc'; //Optional, define the display order. filter?: string null; //Optional, define the filtering criteria. } </pre>
Return value	Promise
Return value description	<pre> { errcode: 0, errmsg: "SUCCESS", personal_cdr_list: [//CDR list { id: 2546, //The sequence number of the record. date: "Today", //The date when the call was made or received. time: "17:21:39", //The time when the call was made or received. timestamp: 1686561699, //The timestamp of the time when the call w number: "1011", // Caller number number_type: "extension", extension: { //Extension information ext_id: 25, //Extension ID ext_num: "1011", //Extension number caller_id_name: "cwt1011", //Extension name photo: "", // Profile image ID status: 0, // Online status of the extension endpoints. 0: Off presence_status: "available", //Extension presence status presence_information: "", first_name: "wt1011", //First name last_name: "c", //Last name email_addr: "", //Email address mobile_number: "", //Mobile number enb_vm: 0, //Whether the voicemail is enabled. vm_total_count: 0, //The total number of voicemail messages vm_unread_count: 0, //The total number of unread voicemail mes visible: 0, </pre>

Method	<code>logout()</code>
Parameters	Null.
Return value	<code>Promise</code>

Method	<code>destroy()</code>
Parameters	Null.
Return value	Null.

```
pbx.on('eventName', (data) => {})
```

Event name	<code>runtimeError</code>
Data	<code>PBXResult</code>
Report parameters	<pre>{ code: '',</pre>

```
msg: '',
}
```

**Note:**

Refer to the following table for reporting parameter descriptions of the event.

Report parameter description

code	msg	Description
-106	LINKUS_DISABLED	Linkus UC client is disabled.
-107	LOGGED_IN_ELSEWHERE	The extension has logged in elsewhere.
-108	EXTENSION_DELETED	The extension has been deleted.
-109	RE_LOGIN	The extension needs to log in again.
-110	SDK_PLAN_DISABLED	PBX plan is NOT Ultimate Plan (UP).

CDR changes event

When this event is triggered, you need to manually call the method to query CDR records.

Event name	<code>cdrChange</code>
Data	<code>cdrNotifyData</code>
Report parameters	<pre>{ ext_num: '1001', personal_cdr: { date:"", // The date when the call was made or received. time:"", // The time when the call was made or received. timestamp: 1693201380, // The timestamp of the time when the call was made or received. Accurate to the second. number: "1001", talk_duration:0 // The time between the call answered and the call ended. } }</pre>

Call Features (PhoneOperator)

The **PhoneOperator** object is used for managing calls. Each call is cached through the `sessions` attribute, which is a map containing different call instances (`Map<string, Session>`), and each call instance is represented as a `Session`. This topic describes the attributes, methods, and events related to call features (PhoneOperator object).

Attributes

Attributes	Type	Description
<code>currentSessionID</code>	string	ID of the current call (Session).
<code>reRegistryPhoneTimes</code>	number	Define the number of retry attempts for SIP UA (User Agent) registration.
<code>deviceIds</code>	{ cameraId?: string; microphoneId?: string; }	IDs of the audio and video input devices, including the camera ID and microphone ID.
<code>sessions</code>	Map<string, Session>	Caching Map object for calls (Session), with the <code>callId</code> as the key.
<code>currentSession</code>	Session	Current call.
<code>isRegistered</code>	boolean	Whether the SIP UA registration is successful.
<code>recordPermissions</code>	number	Call recording permission: <ul style="list-style-type: none"> • 0: No permission • 1: Permission to pause / resume call recording • 2: Permission to start / pause / resume call recording
<code>incomingList</code>	session[]	The list of the incoming calls.
<code>isMaxCall</code>	boolean	Whether the maximum number of concurrent calls has been reached.

Methods

Methods overview

• Listen for events	• Perform an attended transfer	• Terminate a call
-------------------------------------	--	------------------------------------

• Start registering SIP UA	• Hold a call	• Disconnect the SIP UA registration
• Re-register SIP UA	• Resume a call	• Retrieve all the calls (sessions attribute)
• Make a call	• Send DTMF	• Set the current call (currentSession)
• Reject a call	• Mute a call	• Retrieve the current call (currentSession)
• Answer a call	• Unmute a call	• Update the static properties of Session
• Hang up a call	• Start recording	• Destroy PhoneOperator object
• Perform a blind transfer	• Pause recording	

Listen for events

Method	<code>on(eventName:string,listener: (...args: any[]) => void)</code>
Parameters	<ul style="list-style-type: none"> • <code>eventName</code>: The event name. • <code>listener</code>: Callback function.
Return value	Null.



Note:

For more information about the events that you can listen for, see [Events](#).

Start registering SIP UA

After a successful SIP UA registration, users can make and receive calls.



Note:

You need to listen for desired events before registering SIP UA. Otherwise, some events may be missed.

Method	<code>start()</code>
Parameters	Null.

Return value	Null.
---------------------	-------

Re-register SIP UA




Note:

To avoid unexpected situations, use this method **ONLY** within the **PhoneOperator** object.

Method	<code>reRegister(authorizationUser: string, hal: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>authorizationUser</code>: User name. • <code>hal</code>: Password.
Return value	<code>this</code>

Make a call

Method	<code>call(number: string, option?: CallOptions, transferId?: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>number</code>: Callee number. • <code>option</code>: Optional. Specify userMedia constraints. • <code>transferId</code>: Optional. ID of the attended transfer call. <div style="border-left: 2px solid #007bff; padding-left: 10px; margin-top: 10px;">  <p>Note: If there is a value for <code>transferId</code>, it indicates that this is an attended transfer call, which is an asynchronous function.</p> </div>
Return value	<code>Promise<Result></code>

Reject a call

Method	<code>reject(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Answer a call

Method	<code>answer(callId: string, option?: CallOptions)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>option</code>: Optional. Specify userMedia constraints.
Return value	<code>Promise<Result></code>

Hang up a call

Method	<code>hangup(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Perform a blind transfer

Method	<code>blindTransfer(callId: string, number: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>number</code>: The number of the transfer target.
Return value type	<code>boolean</code>

Perform an attended transfer

Method	<code>attendedTransfer(callId: string, number: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>number</code>: The number of the transfer target.
Return value type	<code>boolean</code>

Hold a call

Method	<code>hold(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Resume a call

Method	<code>unhold(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Send DTMF

Method	<code>dtmf(callId: string, dtmf: string)</code>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>dtmf</code>: String (0123456789*#).
Return value type	<code>boolean</code>

Mute a call

Method	<code>mute(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Unmute a call

Method	<code>unmute(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Start recording

Method	<code>startRecord(callId: string)</code>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Pause recording

Method	<code>pauseRecord(callId: string)</code>
---------------	--

Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Terminate a call

Method	<pre>terminate(callId: string, type: 'hangup' 'reject' 'terminate' = 'terminate')</pre>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>type</code>: Types of call termination.
Return value type	<code>boolean</code>

Disconnect the SIP UA registration

Method	<pre>disconnect()</pre>
Parameters	Null.
Return value type	<code>boolean</code>

Retrieve all the calls (sessions attribute)

Retrieve the `sessions` attribute and return them as an array.

Method	<pre>getSession()</pre>
Parameters	Null.
Return value type	<code>boolean</code>

Set the current call (currentSession)

Method	<pre>setCurrentSession(callId: string)</pre>
Parameters	<code>callId</code> : The unique ID of the call.
Return value type	<code>boolean</code>

Retrieve the current call (currentSession)

Method	<pre>getCurrentSession()</pre>
---------------	--------------------------------

Parameters	Null.
Return value	Session

Update the static properties of Session

Method	<pre>setSessionStaticStatus(callId: string, staticStatus: Partial<StaticCallStatus>, startManualModel?: boolean)</pre>
Parameters	<ul style="list-style-type: none"> • <code>callId</code>: The unique ID of the call. • <code>staticStatus</code>: The static properties of Session include <code>name</code>, <code>avatar</code>, and <code>company</code>. • <code>startManualModel</code>: Optional. Whether to enable manual mode. If enabled, the static properties will NOT be automatically updated.
Return value type	<code>boolean</code>

Destroy PhoneOperator object

This method will cancel all the event subscriptions, stop the SIP UA instance, and delete all the sessions.

Method	<pre>destroy()</pre>
Parameters	Null.
Return value	Null.

Events

Event Name	Description	Report Parameter
connected	Connected to the SIP service.	Null
disconnected	Disconnected the SIP service.	Null
registered	SIP UA registration succeeded.	Null
registrationFailed	SIP UA registration failed.	<ul style="list-style-type: none"> • <code>code: number</code>: Error code. • <code>msg: string</code>: Error message.
newRTCSession	A new call instance (Session) is created.	<ul style="list-style-type: none"> • <code>callId: string</code>: The unique ID of the call.

Event Name	Description	Report Parameter
		<ul style="list-style-type: none"> <code>session: Session</code>: The current call instance.
incoming	There is an incoming call.	<ul style="list-style-type: none"> <code>callId: string</code>: The unique ID of the call. <code>session: Session</code>: The current call instance.
startSession	A call instance (Session) is added to the map that stores calls (sessions).	<ul style="list-style-type: none"> <code>callId: string</code>: The unique ID of the call. <code>session: Session</code>: The current call instance.
recordPermissionsChange	The recording permission is changed.	<ul style="list-style-type: none"> <code>0</code>: No permission. <code>1</code>: Permission to pause / resume call recording. <code>2</code>: Permission to start / pause / resume call recording.
deleteSession	A call instance (Session) is deleted.	<ul style="list-style-type: none"> <code>callId: string</code>: The unique ID of the call. <code>cause: string</code>: Reason for deleting the call instance.
isRegisteredChange	Whether the SIP UA registration status is changed.	<ul style="list-style-type: none"> <code>true</code>: Registration status changed. <code>false</code>: Registration status did NOT change.

Call Status and Call Features (Session)

The **Session** object is a call instance for managing call status and performing related operations during a call. This topic describes the attributes, methods, and events related to call status and call feature (Session object).

Attributes

Attributes	Type	Description
RTCSession	RTCSession	Call instance.
callReport	Report	Call quality report.
status	CallStatus	Call status, including name, number, profile image, status of call mute, etc.
incomingList	Session[]	Incoming call array.
timer	TimerType	Call timer.

Attributes	Type	Description
localStream	MediaStream	Local stream with video track only, no audio.
remoteStream	MediaStream	Remote stream with both audio and video tracks.

Method

Method Overview

• Stop the call timer	• Perform an attended transfer	• Start recording
• Listen for events	• Hold a call	• Pause recording
• Reject a call	• Resume a call	• Terminate a call
• Answer a call	• Send DTMF	• Update the call status
• Hang up a call	• Mute a call	• Update the static properties of Session
• Perform a blind transfer	• Unmute a call	• Destroy Session instance

Stop the call timer

Method	<code>stopTimer()</code>
Parameters	Null.
Return value	<code>this</code>

Listen for events

Method	<code>on(eventName:string,listener: (...args: any[]) => void)</code>
Parameters	<ul style="list-style-type: none"> <code>eventName</code>: The event name. <code>listener</code>: Callback function.
Return value	Null.



Note:

For more information about the events that you can listen for, see [Events](#).

Reject a call

This method is equivalent to the 'Reject a call' method of **PhoneOperator** object.

Method	<code>reject()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Answer a call

This method is equivalent to the 'Answer a call' method of **PhoneOperator** object.

Method	<code>answer(option?: CallOptions)</code>
Parameters	<code>option</code> : Optional. Specify userMedia constraints.
Return value	<code>Promise<Result></code>

Hang up a call

This method is equivalent to the 'Hang up a call' method of **PhoneOperator** object.

Method	<code>hangup()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Perform a blind transfer

This method is equivalent to the 'Perform a blind transfer' method of **PhoneOperator** object.

Method	<code>blindTransfer(number: string)</code>
Parameters	<code>number</code> : The number of the transfer target.
Return value type	<code>boolean</code>

Perform an attended transfer

This method is equivalent to the 'Perform an attended transfer' method of **PhoneOperator** object.

Method	<code>attendedTransfer(number: string)</code>
Parameters	<code>number</code> : The number of the transfer target.
Return value type	<code>boolean</code>

Hold a call

This method is equivalent to the 'Hold a call' method of **PhoneOperator** object.

Method	<code>hold()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Resume a call

This method is equivalent to the 'Resume a call' method of **PhoneOperator** object.

Method	<code>unhold()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Send DTMF

This method is equivalent to the 'Send DTMF' method of **PhoneOperator** object.

Method	<code>dtmf(dtmf: string)</code>
Parameters	<code>dtmf</code> : String (0123456789*#)
Return value type	<code>boolean</code>

Mute a call

This method is equivalent to the 'Mute a call' method of **PhoneOperator** object.

Method	<code>mute()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Unmute a call

This method is equivalent to the 'Unmute a call' method of **PhoneOperator** object.

Method	<code>unmute()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Start recording

This method is equivalent to the 'Start recording' method of **PhoneOperator** object.

Method	<code>startRecord()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Pause recording

This method is equivalent to the 'Pause recording' method of **PhoneOperator** object.

Method	<code>pauseRecord()</code>
Parameters	Null.
Return value type	<code>boolean</code>

Terminate a call

This method is equivalent to the 'Terminate a call' method of **PhoneOperator** object.

Method	<code>terminate(type: 'hangup' 'reject' 'terminate' = 'terminate')</code>
Parameters	<code>type</code> : Type of call termination.
Return value type	<code>boolean</code>

Update the call status

Method	<code>setStatus(status: Partial<CallStatus>)</code>
Parameters	<code>status</code> : The call status object.
Return value	<code>this</code>

Update the static properties of Session

Method	<code>setStaticStatus(staticStatus: Partial<StaticCallStatus>, startManualModel?: boolean)</code>
Parameters	<ul style="list-style-type: none"> • <code>staticStatus</code>: The static properties of Session include <code>name</code>, <code>avatar</code>, and <code>company</code>. • <code>startManualModel</code>: Optional. Whether to enable manual mode. If enabled, the static properties will NOT be automatically updated.
Return value	<code>this</code>

Destroy Session instance

This method will cancel all the event subscriptions, and stop the RTCSession.

Method	<code>destroy()</code>
Parameters	Null.
Return value	Null.

Events

Event Name	Description	Report Parameter
callReport	The call quality report is updated, with a frequency of once every 3 seconds.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>callReport</code>: Report: Call quality report.
streamAdded	A media stream is added to the call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>communicationType</code>: "outbound" "inbound": Call type. <code>stream</code>: MediaStream: Media stream.
ended	The call is ended.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reason for ending the call.
failed	Failed to make a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reasons for the call failure. <code>code</code>: number: Error code.
clientError	Errors occurred on the client, resulting in the failure to make a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>cause</code>: string: Reason for the error.
reinvite	The other party performed attended transfer during a call.	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
accepted	Received the success status response code 200 OK .	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
confirmed	The session is confirmed (received the response of ACK packet).	<ul style="list-style-type: none"> <code>callId</code>: string: The unique ID of the call. <code>session</code>: Session: The current call instance.
statusChange	Call status is changed.	<ul style="list-style-type: none"> <code>newStatus</code>: New call status. <code>oldStatus</code>: Previous call status.
staticStatusChange	The static properties of the call is changed.	<ul style="list-style-type: none"> <code>newStatus</code>: New static properties. <code>oldStatus</code>: Previous static properties.

Return Result (Result)

This topic introduces the structure and the subclasses of the 'Linkus SDK for Web Core' return result (Result object).

Background information

Result structure

```
{
  code: '',
  msg: '',
}
```

Result subclasses

Subclass	Description	Code Range
CommonResult	Common return result.	<ul style="list-style-type: none"> • Success code: 0 to 99 • Error code: -1 to -99
PBXResult	Return results related to the PBXOperator object.	<ul style="list-style-type: none"> • Success code: 100 to 199 • Error code: -100 to -199
PhoneResult	Return results related to the PhoneOperator object.	<ul style="list-style-type: none"> • Success code: 200 to 299 • Error code: -200 to -299

CommonResult

Error codes (COMMON_ERROR)

code	msg	Description
-1	UNKNOWN_ERROR	Unknown error.
-2	INVALID_PBX_URL	Invalid PBX URL.
-3	PBX_URL_NOT_HTTPS	The transfer protocol of the PBX URL is not HTTPS.
-4	GET_PRODUCT_FAILED	Failed to retrieve the PRODUCT interface of PBX .

Success code (COMMON_SUCCESS)

code	msg	Description
0	SUCCESS	Success.

PBXResult

Error codes (PBX_ERROR)

code	msg	Description
-100	UNKNOWN_ERROR	Unknown error.
-101	REGISTRY_FAILED	SIP UA registration failed.
-102	PBX_NETWORK_ERROR	API request failed due to the network error on the client-side.
-103	PBX_API_ERROR	API request failed due to the server-side error.
-104	GET_PERSONAL_NOT_FOUND_DATA	PBX did NOT return extension information.
-105	PBX_ALREADY_INITIALIZED	The PBX object has already been initialized and can NOT be initialized again.
-106	LINKUS_DISABLED	Linkus UC client is disabled.
-107	LOGGED_IN_ELSEWHERE	This extension has logged in elsewhere.
-108	EXTENSION_DELETED	The extension has been deleted.
-109	RE_LOGIN	The extension needs to log in again.
-110	SDK_PLAN_DISABLED	PBX plan is not Ultimate Plan (UP).

Success code (PBX_SUCCESS)

code	msg	Description
100	SUCCESS	Success.

PhoneResult

Error codes (PHONE_ERROR)

code	msg	Description
-200	UNKNOWN_ERROR	Unknown error.
-201	REGISTRY_FAILED	SIP UA registration failed.
-202	GET_AGREE_CHROME_USER_MEDIA_ROLE_ERROR	Failed to retrieve media stream (Browser authorization is not granted).
-203	GET_LOCAL_STREAM_ERROR	Failed to retrieve local media stream.
-204	RE_REGISTRY_MAX_LIMIT_TIMES	Reached the maximum number of the allowed SIP UA re-registration attempts.

code	msg	Description
-205	MAX_LIMIT_CALL	Reached the maximum number of concurrent calls.
-206	GET_LOCAL_MEDIA_INFO_ERROR	Failed to access local media device.
-207	ATTENDED_PARENT_NOT_FOUND	Failed to find the parent call of the attended transfer call.
-208	CALL_TOO_MANY_TIMES	Reached the limit of outgoing calls within one second.
-209	INVALID_NUMBER	Invalid number.
-210	CURRENT_CALL_HAS_NOT_CONNECTED	There is currently an unconnected call.
-211	NOT_FOUND_CALL_ID	Failed to find the call ID.
-290	NOT_FOUND_AUDIO_INPUT_DEVICE	Failed to find the audio input device.
-291	NOT_FOUND_VIDEO_INPUT_DEVICE	Failed to find the video input device.

Success code

code	msg	Description
200	SUCCESS	Success.

Types and Interface of other Objects

This topic introduces the types and interface of other objects used by 'Linkus SDK for Web Core'.

Report object

Call quality report object.

```
interface Report {
    lksTimestamp: string; // The timestamp of the time when the call was
    made or received.
    lksDate: string; // The time when the call was made or received. Time
    format: YYYY-MM-DD hh:mm:ss
    callId: string; // The unique ID of the call.
    extension: string; // Extension number.
    iceResult: string; // The ICE candidate type. local: host; public:
    srflx or prflx; turn: relay.
```

```

    lksNetworkType: string; // Network type.
    lksDeviceType: string; // The client type. Currently only Web client is
    available.
    lksDuration: string; // Correspond to the 'totalSamplesDuration'
    attribute.
    lksAudioCodec: string; // Audio codec.
    lksAudioRx: string; // The audio packets received that correspond to
    the 'packetsReceived' attribute.
    lksAudioRxLost: string; // The audio packet loss rate that corresponds
    to the 'packetsLost' attribute. Formula: lksAudioRxLost=(packetsLost /
    (packetsLost + packetsReceived)).toFixed(6).
    lksAudioRxMaxjitter: string; // The maximum audio jitter on the
    receiver.
    lksAudioRxAvgjitter: string; // The average audio jitter that
    corresponds to "Jitter*1000".
    lksAudioTx: string; // The audio packet sent.
    lksAudioTxLost: string; // Audio packet loss rate on the sender.
    lksAudioTxMaxjitter: string; // The maximum audio jitter on the sender.
    lksAudioTxAvgjitter: string; // The average audio jitter on the sender.
    lksVideoCodec: string; // Video codec
    lksVideoRx: string; // The video packet received.
    lksVideoRxLost: string; // The video packet loss rate that corresponds
    to the 'packetsLost' attribute. Formula: lksVideoRxLost=(packetsLost /
    (packetsLost + packetsReceived)).toFixed(6).
    lksVideoRxMaxjitter: string; //The maximum video jitter on the
    receiver.
    lksVideoRxAvgjitter: string; // The average video jitter on the
    receiver.
    lksVideoTx: string; // The video packet sent.
    lksVideoTxLost: string; // Video packet loss rate on the sender.
    lksVideoTxMaxjitter: string; // The maximum video jitter on the sender.
    lksVideoTxAvgjitter: string; // The average video jitter on the sender.
}

```

StaticCallStatus object

Static properties of a call.

```

interface StaticCallStatus {
    name: string; // Caller name
    avatar: string; // Profile image
    company: string; // Company name
}

```

CallStatus object

The call status object.

```
interface CallStatus extends StaticCallStatus{
    number: string; // Phone number or extension number.
    callId: string; // The unique ID of the call.
    communicationType: 'outbound' | 'inbound'; // Call type
    isVideo: boolean; //Whether it is a video call.
    isRing: boolean;
    isHold: boolean;
    isMute: boolean;
    callStatus: 'ringing' | 'calling' | 'talking' | 'connecting';
    callStartTime: number; // The time when the call is answered.
    recordStatus: 'stop' | 'recording' | 'pause'; // Recording status.
    stop: Not in recording; recording: Recording,; pause: Recording is paused.
    isTransfer: boolean; // Whether it is an attended transfer call.
    transferParent?: TransferParentType; // When it's an attended transfer
    call, specify the parent call.
    isConference: boolean; // Whether it is an audio conference call.
}

type TransferParentType = {
    // Parent call in the attended transfer call.
    callId: string; // The unique ID of the call.
    avatar: string; // Profile image.
    name: string; // Name of the transfer target.
    number: string; // Number of the transfer target.
    callDuration: number; // The time between the call answered and the
    call ended.
    holdDuration: number; // The duration of the call being held.
}
```

TimerType type

Types of call timing.

```
type TimerType = {
    ringDuration: number; // The time between the call started and the call
    answered.
    callingDuration: number; // The time between the call dialed and the
    call answered.
    callDuration: number; // The time between the call answered and the
    call ended.
    holdDuration: number; // The duration of the call being held.
}
```

Tone Resource

This topic introduces the tone resources provided by 'Linkus SDK for Web Core' and the usage methods.

Background information

The **assets** directory of 'Linkus SDK for Web Core' project files contains tone resources in the following forms:

- **Original audio files:** Stored in the **sounds** folder.
- **base64-encoded strings:** Stored in the **sound.js** file.

Refer to the following table for the available tone resources.

Name	Description	Name	Description
Ring	Incoming call ringtone	DTMF04	Keypad 4 tone
Callend	Call end tone	DTMF05	Keypad 5 tone
Callwaiting	Call waiting tone	DTMF06	Keypad 6 tone
ringback	Ringback tone	DTMF07	Keypad 7 tone
DTMF00	Keypad 0 tone	DTMF08	Keypad 8 tone
DTMF01	Keypad 1 tone	DTMF09	Keypad 9 tone
DTMF02	Keypad 2 tone	DTMFStar	Keypad * tone
DTMF03	Keypad 3 tone	DTMFPound	Keypad # tone

Use tone resources

'Linkus SDK for Web Core' supports to use tone resources in the following methods:

- Use the original audio files of tone resources.
- Use the base64-encoded strings of the tone resources, as shown in the sample code below.

```
import sounds from '/assets/sounds';
const { Ring, Callend } = sounds;
const audio = new Audio(Ring);
audio.play();

// Change the value of "audio.src" to play different tones.
audio.src = Callend;
audio.play();
```


Linkus SDK for Web UI

Integrate Linkus SDK for Web UI

Linkus SDK for Web UI provides a pre-integrated UI component that requires no additional coding, you can integrate it with your Web projects to implement call functionalities with a user interface based on the integration of 'Linkus SDK for Web Core'.

Prerequisites

- You have [obtained the Linkus SDK login signature](#).
- You have [integrated 'Linkus SDK for Web Core'](#).

Step 1. Import 'Linkus SDK for Web UI'

1. Go to the [GitHub Repository of 'Linkus SDK for Web UI'](#), and download 'Linkus SDK for Web UI'.
2. Use either of the following methods to import 'Linkus SDK for Web UI' to your project.
 - Use **npm** to install 'Linkus SDK for Web UI'

```
npm install ys-webrtc-sdk-ui --save
```

- Use **script** to import 'Linkus SDK for Web UI'

```
<script src="./ys-webrtc-sdk-ui.js"></script>
```

Step 2. Initialize 'Linkus SDK for Web UI'


Use the `init` method to initialize 'Linkus SDK for Web UI' and render the UI components.

Method

```
init (container, {rtcOption})
```

Parameters

- `container`: Used to render UI components, and the data type is `HTMLElement`.
- `rtcOption`: Initialization options. Refer to the following table for specific parameters.

Parameter	Type	Required	Description
username	string	Yes	Extension number or email address.
secret	string	Yes	Linkus SDK login signature.
pbxURL	URL string	Yes	The URL for accessing your PBX system, including the transfer protocol. For example, <code>https://192.168.1.1:8088</code> .
enableLog	boolean	No	Whether to enable log output and report error logs to PBX. Valid value: <ul style="list-style-type: none"> ◦ <code>true</code>: Enable ◦ <code>false</code>: Disable <div>  Note: This feature is enabled by default. </div>
reRegistryPhoneTimes	number	No	Define the number of retry attempts for SIP UA (User Agent) registration. By default, it is unlimited.
deviceIds	{ cameraId?: string; microphoneId?: string; }	No	Define the IDs of the audio and video input devices, including the camera ID and microphone ID.
incomingOption	{ style?: React.CSSProperties; class?: string; }	No	Adjust the styling of the incoming call component.
dialPanelOption	{ style?: React.CSSProperties; class?: string; }	No	Adjust the styling of the dialpad component.
sessionOption	SessionOption	No	Adjust the position and size of the call window component.

Parameter	Type	Required	Description
hiddenIncomingComponent	boolean	No	Hide the incoming call component.
hiddenDialPanelComponent	boolean	No	Hide the dialpad component.
disableCallWaiting	boolean	No	Whether to disable call waiting. Valid value: <ul style="list-style-type: none"> ◦ <code>true</code>: Disable call waiting. <p>The call waiting time set on PBX will NOT take effect and the PBX only handles a single call.</p> <ul style="list-style-type: none"> ◦ <code>false</code>: Enable call waiting.
intl	{ local: string; messages: Record<string, string> }	No	Internationalization (multilingual) settings.

SessionOption description

```

type SessionOption = {
  sessionSetting?: {
    width?: number;
    height?: number;
    minWidth?: number;
    minHeight?: number;
    x?: number;
    y?: number;
  };
  style?: React.CSSProperties;
  class?: string;
}

```

intl (Internationalization) description

- `local`: Name of the current language, connected with a `-`. For example, `en-US`、`zh-CN`.
- `message`: Configurable text content, which is presented in key-value pairs. Refer to the following content

for specific configuration items and their default values.

```
{
  "common.cancel": "Cancel",
  "common.confirm": "Confirm",
  "dial_panel.input.placeholder":
  "Please input number",
  "dial_panel.tip.connect_failed":
  "Failed to connect to server, you cannot
  initiate or answer a call. Trying to
  reconnect to the server.",
  "incoming.btn.hang_up": "Hang up",
  "incoming.btn.audio": "Audio",
  "session.calling": "Calling...",
  "session.ringing": "Ringing...",
  "session.talking": "Talking...",
  "session.connecting": "Connecting...",
  "session.hang_up": "End Call",
  "session.new_call": "New call",
  "session.record": "Record",
  "session.mute": "Mute",
  "session.hold": "Hold",
  "session.resume": "Resume",
  "session.dialpad": "Dialpad",
  "session.transfer": "Transfer",
  "session.attended_transfer": "Attended
  Transfer",
  "session.blind_transfer": "Blind
  Transfer",
  "session.error.client_error": "Client
  Error: {0}",
  "session.tip.recording": "Recording
  the Audio...",
  "session.tip.pause": "The recording is
  paused.",
  "error.code_200": "Unknown Error.",
  "error.code_202": "No available
  communication device found.(no
  permissions)",
  "error.code_205": "Call failed. Cannot
  process more new calls now.",
  "error.code_206": "No available
  communication device found.",
```

```

    "error.code_207": "Attended Transfer
Failed.",
    "error.code_208": "Call failed. Called
too many times.",
    "error.code_209": "Call failed.
Invalid Number.",
    "error.code_210": "Operation failed
with pending calls.",
    "error.code_211": "Answer failed",
    "error.code_290": "No available
microphone found.",
    "error.code_291": "No available camera
found."
}

```

Sample code

Use 'npm' to install and initialize 'Linkus SDK for Web UI'

```

import { init } from 'ys-webrtc-sdk-ui';
import 'ys-webrtc-sdk-ui/lib/ys-webrtc-sdk-ui.css';
const container =
  document.getElementById('container');
// initialization
init(container, {
  username: '1000',
  secret: 'sdkshajgllliiaggsckjh',
  pbxURL: 'https://192.168.1.1:8088'
}).then(data => {
  // Obtain the exposed instances for additional
  business needs
  const { phone, pbx, destroy, on } = data;
  // ...
}).catch(err=>{
  console.log(err)
})

```

Use 'script' to import and initialize 'Linkus SDK for Web UI'

```

<!-- Loading styles -->
<link rel="stylesheet" href="ys-webrtc-sdk-ui.css">

<div id="test"></div>
<!-- Loading Yeastar WebRTC SDK UI-->
<script src="./ys-webrtc-sdk-ui.js"></script>
<script>

```

```

const test = document.getElementById('test');
// Initialize Yeastar WebRTC SDK UI with the
'YSWebRTCUI' object.
window.YSWebRTCUI.init(test, {
  username: '1000',
  secret: 'sdkshajgllliiaggskjhf',
  pbxURL: 'https://192.168.1.1:8088'
})
  .then(data => {
    const { phone, pbx } = data;
  })
  .catch(error => {
    console.log(error);
  });
</script>

```

Result

You have integrated and initialized 'Linkus SDK for Web UI', two instantiated Operator objects are returned:

- **PBXOperator**: Contains methods and attributes related to the PBX, such as querying CDR records, logging out, and more.
- **PhoneOperator**: Contains methods and attributes related to calls, such as making a call, answering a call, ending a call, and more.