ILEDCLOUD SDK interface instruction

SDK instruction :

一、 *ProgramManager*

This function is mainly for sending the main implementation of the user's designated file to the designated screen designated partition function;

Method :

Name	parameters	Return value	Instruction
sendProgram()	String apiUrl: API address necessary String appKey : APPKey String screens: The comma split string screen Map<String,File[]> data: key: interface partition code , "0"= default partition value:put into the partition file list Value =null, the default material for the interface partition is used Value =[] (empty array) clears the material. SendTask: send the callback control object, the caller can in another thread, read the send process and control the cancel	SendResult	Send the specified file to the specified screen.Send via default interface partition. This method blocks the current thread and does not return until the sending error or the sending is complete.

	send etc		

二、 SendTask

This class is used to allow the user to check its sending process in another thread, or to control the cancellation of sending;This class needs to be passed in when the program is sent;

Property

Name	Type	Empty	Instruction
name	Stirng	Yes	Task name (input and output)
status	Int	yes	Task progress (output) 0: check the sending conditions 1: sending 2: successful delivery (completed) 3: delivery failed (completed)
progress	float	yes	Task progress (0-100), which is set in sendPrograms and can be read by users in their own programs.
message	String	yes	Task progress status message (output)

cancel	Boolean	yes	Cancel task (enter) If cancel=true, cancel the send, and the sendProgram function is internally responsible for checking, breaking the send and returning at the appropriate time
reTryTimes	Integer	yes	Upload file retry number, default is 3, maximum 10 times

三、 SendResult

This class is used for the operation result of sending the program after the user sends the program.

Property

Name	Type	Empty	Instruction
name	Stirng	yes	Task name
result	Int	yes	Task progress status 0: successfully sent 1: failed to send
message	String	yes	Send the result message (error reason, etc.)
data	String	yes	Each screen sends a status list The same number of screens parameters as sendProgram passes in, separated by commas

Calling

In secondary development, it is necessary to send the file specified by the user to the specified partition of the specified screen, pass corresponding parameters to it, and call the method code of SDK and corresponding parameters as follows:

```
public static void main(String[] args){

        // request the address of the API
String apiUrl = "https://www.iledcloud.com/cloudapi";
//APPKEY value, source :iLEDCloud multimedia information publishing platform -> advanced function -> secondary development - basic information APPKEY under > application management
String appKey = "484 c1b676ce54f549fd0d027a14ee39b";
// comma separated screen barcode list, screen barcode source :iLEDCloud multimedia information publishing platform -> resource management -> screen management - device number in the list (select valid: online status, package status: package valid, package traffic is not 0)
String screens = "C0Y0401807210047, C0Y0401807210022";
// select the material to upload
File File = new File("/home/ HZK/picture/screenshot /zg.jpg");
File[] fs = new File[10];
Fs [0] = file;
```

// Map<String,File[]> data to be sent

// key: interface partition code, "0" = default interface partition, other values can be obtained from iLEDCloud multimedia information publishing platform -> advanced function -> secondary development -> interface partition management

// value: to be placed in the File list of interface partition, if value=null represents the default material of using interface partition and value=new File[0] (empty array), then clean up the interface partition material

Map < String, the File [] > data = new HashMap < > ();

Data. The put (" 0 ", the fs);// need to upload material to default interface partition

/ / data. The put (" 073 e8afe65d74722ad4b40c6f48c3fdb ", fs);

//SendTask send callback control object, the caller can in another thread, read send process and control cancel send etc

SendTask = new SendTask();

Try {

// calls the interface

SendResult SendResult = ProgramManager. GetInstance (). SendProgram (apiUrl, appKey, screens, data, task);

// task status 0: send successfully, 1: send failed

Int result = sendResult. GetResult ();

If (result == 0){

// sent successfully

// send the result message

String message = sendResult. GetMessage ();

// screens send the same number of screens parameters as sendProgram passes in

```
String rData = sendResult. GetData ();

System.out.println(" send result message (check if there is invalid bar code in yes) :"+message);

System.out.println(" send status list of each screen after successful sending :"+rData);

}else if (result == 1){

//1: failed to send

// send the result message

String message = sendResult. GetMessage ();

// screens send the same number of screens parameters as sendProgram passes in

String rData = sendResult. GetData ();

System.out.println(" error reason for sending result message :"+message);

System.out.println(" send status list of each screen after sending failure :"+rData);

}

} catch (Exception e) {

E.p rintStackTrace ();

        }

    }
```

Name	Parameters	Return value	Instruction
sendProgram()	String apiUrl: requested API address must String appKey: appKey String screens: a comma - separated list of barcodes for	SendResult	Sends the specified file to the specified

	screens Map < String, the File [] > data: Key: interface partition code, "0"= default interface partition Value: list of files to be placed in the interface partition. Value =null, the default material for the interface partition is used Value =[] (empty array) clears the material. SendTask: send the callback control object, the caller can in another thread, read the send process and control the cancel send etc.		screen.Send via default interface partition. This method blocks the current thread and does not return until the sending error or the sending is complete

sample :

Description

Sample program yes in order to facilitate users to quickly get started and carry out secondary development of the package program, users can directly download the installation and use;This program mainly monitors the folder (task) under the directory specified in the configuration file. Once there is a new (new or modified) task.ini file, it parses it and automatically sends the content specified in task.ini to the screen according to the APPKey specified in the configuration file.Users should follow the following contents to understand and configure the sample program;

Firmware configuration parameters :

- DataRoot = monitor directory must, the root directory of the monitor
- AppKey = < appKey > necessary
- ApiUrl = requested API address necessary
- ClearTask [optional]=true/false;After sending, will yes delete task.ini file?Response: generate a task.ini file with a different name each time;Parallel production of task.ini files;Each task.ini sends a different screen, different content, etc.).The default false
- ClearResult [optional]=n(seconds);After the completion of transmission, how long will it take to automatically delete the result.txt file (mainly for the scenario where a new task file is generated each time) -1 or not set, which means no deletion;0 means immediate deletion;Other values are deleted after n seconds.The default -1;
- ClearMaterials = true/false;After sending, will yes delete the material files used in the task?(for applications that generate new material files every time).The default false
- Backup [optional]= backup directory, indicating that after each send, send material files, task.ini files, result.txt files to the specified directory of backup in a fixed way.(should trace management function).Default empty, no backup.
- BackupMode [optional]= copy/move/link (copy/move/ establish hard connection);Backup mode, setting the backup parameter will be effective.The default copy
-

Request

[DataRoot]

template/

task/task*.ini

result/result*.txt

Instruction：

File	Type	Content	Input/output
template	File folder	Store the template file of task.ini,result.txt (definition file, sample file).Content fixed. Each time the program starts, check and automatically generate.Used for manual testing, or programming reference.	output
task	File folder	Store the user's task.ini file (created if detection does not exist at startup)	Input
result	File folder	Store task send result file result.txt (created if detection does not exist at startup)	output
task*.ini	Ini file	Task definition file, can have more than one, fixed to start with task, do not automatically delete after completion. Content definition: # interface partition list,	input

		The left side of the equals sign is the interface partition code, 0 represents the default interface partition. On the right side of the equals sign is a list of files or folders, which requires absolute path, semicolon segmentation, non-newline, If the right side is clear, it means to clear the material. If the right side is blank, the interface partition default material is used [zone] 0 = < file1 >;The < file2 >;The < dir1-name >;The < dir2 >... # screen list , send the above interface partition and footage to the specified screen, one per line, 1 for send, 0 for no send [screen] < code > 1 = 1 < code > 2 = 1 Note that [zone] and [screen] do not have to be fixed.	
result*.txt	Test file	Send the result file.When task is finished, the program stores the result into this file. The content is defined as follows: Line 1: the progress value, the percentage of delivery progress. Second line: message value, sending a fact sheet. After that, each line has a screen bar code and sends the result Screen bar code = send result (success,message)	output

		Last line: end.When there is no end, the send is never completed.	

Request

- The user's material can be placed in a location accessible to any sample application, or under [DataRoot]/materials, without forcing this.Users manage the material files themselves.
- Users cannot edit task.ini directly under the [DataRoot]/task folder. They must copy task.ini in a folder other than [DataRoot]/task.

Update log

更新日志

- **版本：v2.1.0**

  发布日期：2019-07-26

  更新内容：

  1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  5. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  6. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  收起

- **版本：v2.0.2**

  发布日期：2019-05-26

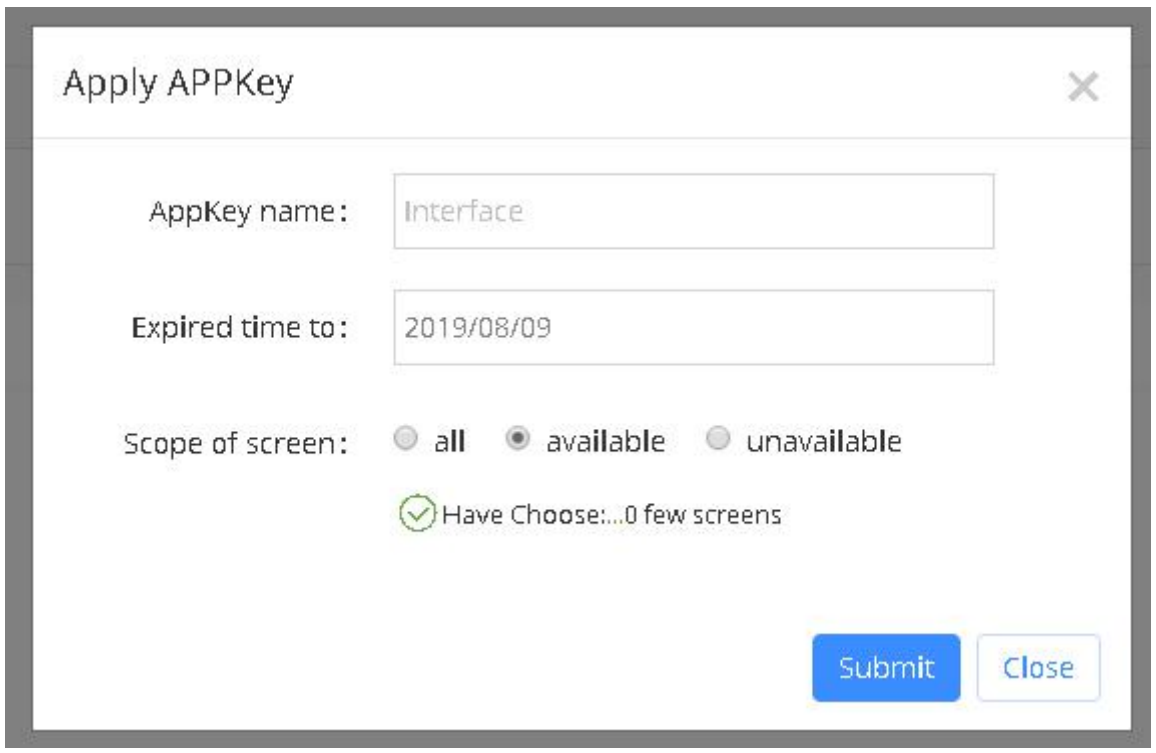
  更新内容：

  1. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  查看更多

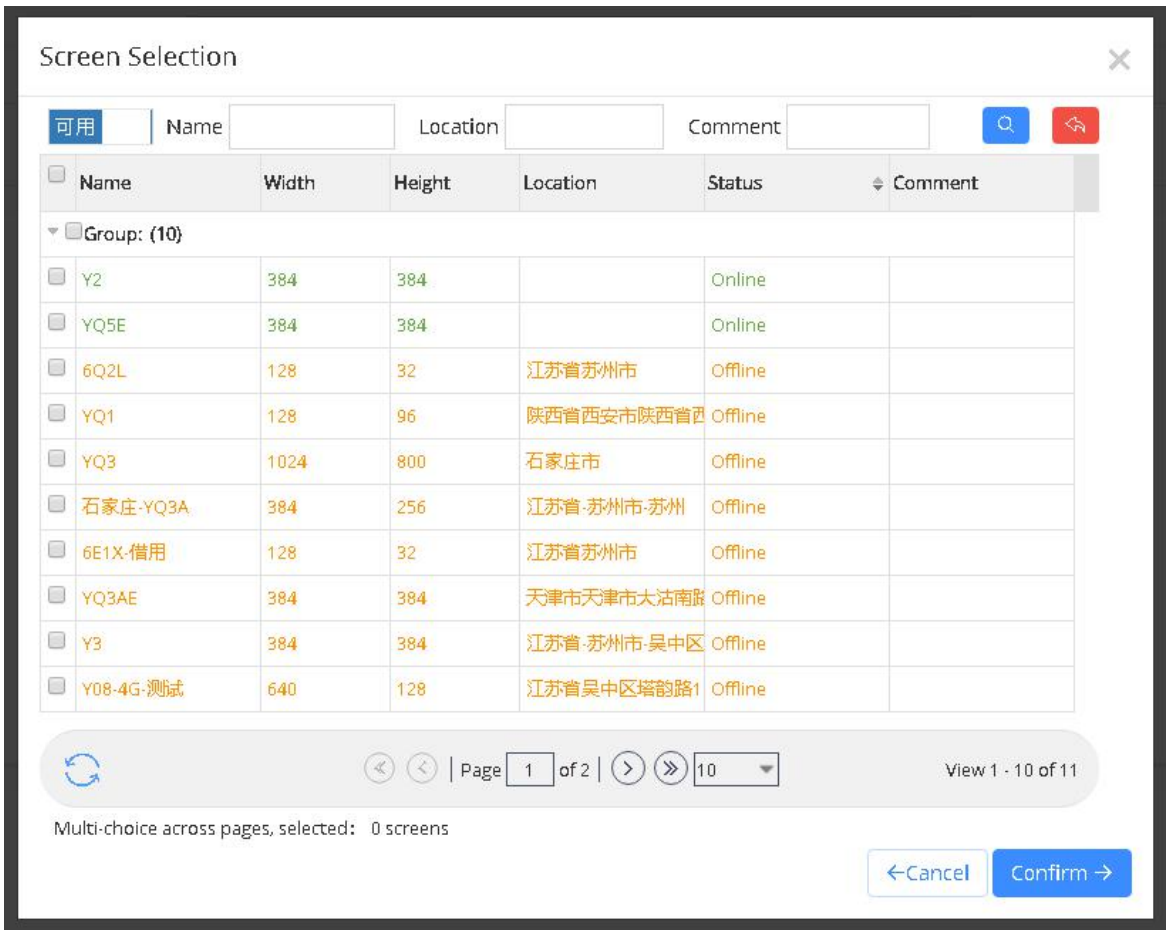
Example as the above：if more than 4 programs,then need，Then fold and click for more to expand the rest；

The second development function was added in the v2.1.0 version of the platform. When users enter this function for the first time, an APPKey application popup will pop up, as shown in figure 5.9-2. Users need to initially set the name, effective time and screen scope of the APPKey in the popover (as shown in figure 5.9-3) and submit it before they can use it in the next step.



Picture 5.9-2 Apply APPKey

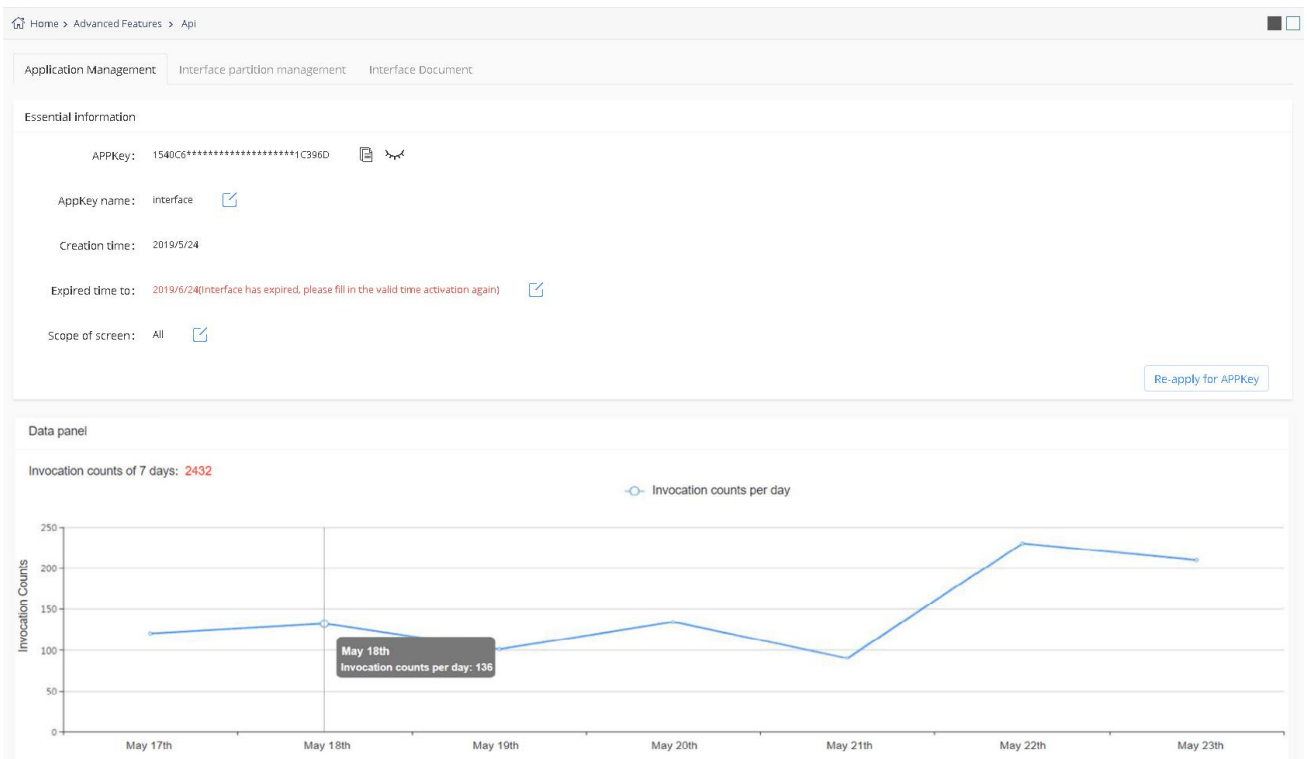
When selecting the applicable range of the screen, click the part that is available and the part that is not available, and the screen selection popup as shown in figure 5.9-3 will pop up. Users can choose which screens can or cannot use the secondary development function;



Picture 5.9-3 select the applied screen

After the creation of APPKey, it enters the main interface of secondary development function. The page is divided into three parts: interface management, interface partition management and interface document.

Figure 5.9-4 shows the interface management interface, displaying the basic information and data panel of the interface;In the basic information section, users can see the APPKey and its name, creation time, valid time and screen application scope, and reapply the APPKey button, and can modify its name, valid time and screen application scope separately.Click the button of reapplying APPKey, then the new APPKey and interface can be reapplied and the original interface will be invalid;In the data panel, users can view the total number of data calls and the number of calls per day within a week.

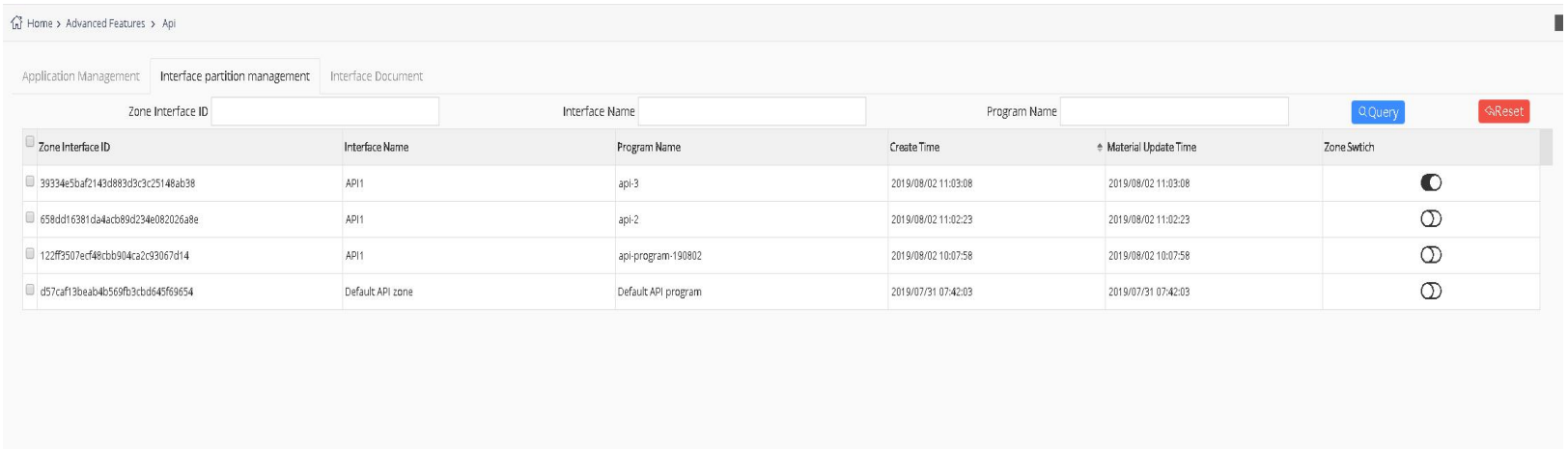


Picture 5.9-4 Interface management

Click the interface partition management page above to enter the interface partition management page. In this page, users can view the partition ID, partition name, program name, creation time, material update time and other contents of the interface partition created in the applicable interface, and can manage the switch of interface partition.

When each interface is created, a unique full-screen interface partition will be automatically generated immediately. Users can use this interface partition to publish full-screen programs according to the instructions in the interface document.

Meanwhile, users can create a new interface partition during program production (as shown in figure 5.9-6). After the program is saved, a new interface partition will be generated, and the partition information can be seen in the interface management office. At this time, users can send materials to this interface partition through interface development.In the interface partition management, the user can also control the interface partition on and off to limit the playback of the interface.



Picture 5.9-5

Picture 5.9-6

Click the interface document page above to enter the interface document page. In this page, users can view the instructions of secondary development function SDK and call methods, and download the SDK.In addition to SDK, users can download the sample program and check the instructions of the sample program to quickly get started and use the related functions of secondary development.