FAQs

Panel Development > Panel SDK Development > 5 Min > FAQs

Version: 20200108

Online Version

Contents

1 How to change device status

Changing the status of the device is that we need to `publish` a command to the device, and the device will`report` the command after receiving it successfully. In simple terms, if we need to change the status of the device, we can use the following code.

```
import { TYSdk } from "tuya-panel-kit";

const TYDevice = TYSdk.device;

const data = { [dpCode]: dpValue };

TYDevice.putDeviceData(data);
```

The above method is the simplest operation method, but we recommend to issue instructions through the action defined in redux. Through a unified processing method, we can more easily deal with subsequent interface name changes.

```
import { updateDp } from "path/redux/modules/common";

const data = { [dpCode]: dpValue };

dispatch(updateDp(data));
```

2 Common Event Series

1. Specific usage:

```
import { TYSdk } from "tuya-panel-kit";

const TYEvent = TYSdk.event;

TYEvent.on(yourEventName, yourHandler);
```

2. Methods for monitoring events:

- TYEvent.on: start listening for events
- TYEvent.off: cancel listening event
- TYEvent.emit: active event

3. Common event names:

- `deviceDataChange`: The core event, which is divided into three blocks in total, is distinguished by the type field in the return value: When type is`dpData`, it represents the status of the dp point, that is, the device reports the status of the dp point; When devInfo, `it represents device information change notification`, `such as device name change; when type is`deviceOnline', it represents device online status change
- `networkStateChange`: App network state change notification
- `bluetoothChange`: Bluetooth online status change notification
- `onLinkageTimeUpdate`: Timed state change notification
- `NAVIGATOR_ON_WILL_FOCUS`: Route change event encapsulated inside NavigatorLayout
- `NAVIGATOR_ON_DID_FOCUS`: Route change event encapsulated inside NavigatorLayout

> Be careful not to forget to **unlisten** events when **unmount**

3 Common method series

1. Specific usage:

```
import { TYSdk } from "tuya-panel-kit";

const TYDevice = TYSdk.device;
const TYNative = TYSdk.native;
```

2. Dp related method:

```
/**
 * @desc issues dp (core methods for interacting with the hardware side
    )
 * @param {Object} data-dp point data
 */
TYDevice.putDeviceData(data);
```

```
/**
 * @desc Get dpId according to dpCode
 * @param {String} dpCode
 * returns {Number|String} dpId
 */
TYDevice.getDpIdByCode(dpCode);
```

```
/**
 * @desc Get dpCode according to dpId
 * @param {String} dpId
 * @returns {String} dpCode
 */
TYDevice.getDpCodeById(dpId);
```

```
/**
 * @desc Check if dp exists
 * @param {String} dpId|dpCode
 * @returns {Bool}
 */
TYDevice.checkDpExist(dpId | dpCode);
```

```
/**
 * @desc Get schema information of dp according to code
 * @param {String} dpCode
 * @returns {Object}
 */
TYNative.getDpSchema(dpCode);
```

** 3. Data related methods: **

```
/**
* @desc Request Cloud Interface
*/
TYNative.apiRequest(
  {
    a: apiName,
    v: apiVersion
    postData: params,
  },
  d => successHandle(d),
  e => errorHandle(e),
);
```

4. TYNative related methods:

```
/**
 * @desc Jump to cloud timing page
 */
TYNative.gotoDpAlarm({
  category: category,
  repeat: 0, // 0 means repeat, 1 means not needed
  data: [
    {
      dpId: dpId,
      dpName: dpName,
      selected: 0,
      rangeKeys: [true, false],
      rangeValues: [dpValue1, dpValue2]
    }
  ]
});
```

```
/**
 * @desc lightweight conversation
 * @param {String} confirmText - confirmation text
 * @param {String} cancelText - cancel text
 * @param {String} title - dialog title
 * @param {String} message - text
 * @param {String} defaultValue - default value
 * @param {Function} onConfirmed - confirmation function
 * @param {Function} onCanceled - cancel function
 */
TYNative.showPromptDialog(
  confirmText,
  cancelText,
  title,
  message,
  defaultValue,
  onConfirmed,
  onCanceled
);
```

```
/**
 * @desc edit dialog
 * @param {String} title - title
 * @param {String} editString - edit the content of the message
 * @param {Function} onConfirmed - confirmation function
 * @param {Function} onCanceled - cancel function
 */
TYNative.showEditDialog(title, editString, onConfirmed, onCanceled);
```

```
/**
 * @desc bottom conversation list
 * @param {Array} itemList - list
 * @param {Number} selected - the index of the selected list
 * @param {Function} onConfirmed - confirmation function
 */
TYNative.bottomListDialog(itemList, selected, onConfirmed);
```

```
/**
 * @desc Easy confirmation dialog
 * @param {String} title - title
 * @param {String} msg - dialog information
 * @param {Function} onConfirmed - confirmation function
 * @param {Function} onCanceled - cancel function
 */
TYNative.simpleConfirmDialog(title, msg, onConfirmed, onCanceled);
```

```
/**
 * @desc display loading
 * Remarks: In IOS, displaying the dialog in the upper layer of the
    modal will cause an abnormal life cycle that cannot be controlled.
 */
TYNative.showLoading();
```

```
/**
 * @desc hide loading
 */
TYNative.hideLoading();
```

```
/**
 * @desc Whether it is 24-hour
 * @returns {Bool}
 */
TYNative.is24Hour();
```

```
/**
 * @desc Jump to an existing scene that has not been unmounted
 * @param {String} url - path
 */
TYNative.jumpTo(url);
```

```
/**
 * @desc IOS disable gesture full screen return
 */
TYNative.disablePopGesture();
```

```
/**
 * @desc ios open gesture full screen return
 */
TYNative.enablePopGesture();
```