Matter Devices

Version: 20240927

Online Version

Contents

1. What is Matter?

The Matter standard, organized by the Connectivity Standards Alliance (CSA, formerly the Zigbee Alliance), is jointly promoted by Amazon, Google, Apple, and the CSA. Matter aims to allow all smart devices to become interoperable and implement device control simply on top of a single protocol. This topic describes how to implement management and control of Matter devices after they are integrated into your project.

2. Preparation

Before a Matter device comes into use, you must finish the **configuration of the development project** and **implement pairing with the Matter device**. For more information, see the documentation on pairing Matter devices .

Then, the Matter device can be added to your app for further use.

3. Determine Matter device

API description

```
DeviceBean.java

boolean isMatter();
```

Example

Java:

```
DeviceBean deviceBean = ThingHomeSdk.getDataInstance().getDeviceBean(devId);
if(deviceBean != null) {
  boolean isMatter = deviceBean.isMatter();
}
```

4. Control Matter device

Similarly to a generic type of device, call `publishDps` to send a device control data point (DP).

```
IThingDevice iThingDevice = ThingHomeSdk.newDeviceInstance(devId);
iThingDevice.publishDps(dps, new IResultCallback() {
  @Override
  public void onError(String code, String error) {

  }

  @Override
  public void onSuccess() {

  }
});
```

The following table lists the channels supported by Matter devices.

Device type	Supported channel	Remarks
Tuya-enabled Matter over Wi-Fi device	• Matter local area network (LAN) channel • MQTT channel	Tuya's LAN channel requires support by hardware firmware in the near future.
Tuya-enabled Matter gateway	• Matter LAN channel • Tuya's LAN channel • MQTT channel	None.
Tuya-enabled Matter over Thread device	• Matter LAN channel • Tuya's LAN channel • MQTT channel	None.
Third-party Matter device	Matter LAN channel	None.

5. Determine whether Matter device is online

Similarly to other types of devices, use the `isOnline` field of `DeviceBean` to determine whether a Matter device is online or offline.

API description

```
DeviceBean.java

boolean getIsOnline();
```

Example

Java:

```
DeviceBean deviceBean = ThingHomeSdk.getDataInstance().getDeviceBean(devId);
if(deviceBean != null) {
  boolean isMatter = deviceBean.getIsOnline();
}
```

6. Multiple Fabrics

The Multiple Fabrics feature is exclusive to Matter devices. This feature allows a Matter device to be interoperable among different manufacturers and ecosystems. Therefore, this device can be activated and used seamlessly on multiple Matter-enabled apps. For example, the Matter device can be used on a Tuya-enabled app, such as the **Smart Life** app. It can also be accessed with Apple Home, Google Home, and Amazon Alexa.

> *i*
>
> A fabric is a group of networked devices (also known as nodes) that share the same security domain. This enables secure communications among these nodes within the fabric. Nodes in the same fabric share the same Certificate Authority’s (CA) top-level certificate (Root of Trust) and, within the context of the CA, a unique 64-bit identifier named Fabric ID.

The Multiple Fabrics feature includes two parts: **share among fabrics** and **manage fabrics** .

7. Check channel availability

Check if the mobile app can communicate with the device before you invoke **sharing among fabrics** and **fabrics management** .

- HomeKit SDK v5.2.0 or later supports multiple channels for **sharing among fabrics** and **fabrics management**, with a new method for checking the availability of channels.
- Multichannel communication requires the respective support from the device firmware. The new API method is compatible with legacy devices, so you are recommended to migrate to it.

7.1. Channel availability check for SDK v5.2.0 or later

API description

```
boolean checkPipelineAvailable();
```

Example

Java:

```
IThingMatterMultipleFabricDevice mThingMatterFabricDevice;
...
private void preCheck(String devId){
   mThingMatterFabricDevice =
ThingHomeSdk.newMatterMultipleFabricDeviceInstance(devId);
   boolean pipelineAvailable = mThingMatterFabricDevice.checkPipelineAvailable();
}
```

7.2. Channel availability check for SDKs earlier than v5.2.0

API description

```
boolean isMatterOnline();
```

Example

Java:

```
private void preCheck(String devId){
```

```
    mThingMatterFabricDevice =
ThingHomeSdk.newMatterMultipleFabricDeviceInstance(devId);
    boolean isMatterOnline = mThingMatterFabricDevice.isMatterOnline();
}
```

8. Share among fabrics

```
%%{init: { "sequence": { "wrap": true} } }%%

sequenceDiagram
title: Process of sharing among fabrics

participant app1 as Tuya-enabled app
participant device as Matter device
participant cloud as Cloud
participant app2 as Third-party app

note over app1, cloud: Generate Multiple Fabrics sharing code
rect rgb(206, 235, 252)
  app1 ->> app1: Check channel availability

  app1 ->> device: Request the maximum number of supported fabrics
  device -->> app1: Return maxNum, such as 5

  app1 ->> device: Request the number of used fabrics
  device -->> app1: Return usedNum, such as 2

  alt maxNum - usedNum ≤ 0
    app1 ->> app1: Prompt: sharing quota reached
  else maxNum - usedNum > 0
    app1 ->> device: (Optional) Request SSID of Wi-Fi network connected by
device, only available to Matter over Wi-Fi device
    device -->> app1: Return SSID of Wi-Fi network
    app1 ->> app1: Display SSID of Wi-Fi network connected by device

      app1 ->> device: Request closing window for sharing and pairing
      device -->> app1: Return result of closing window
      alt Failed to close window
          app1 ->> app1: Prompt: unable to share
      else Window closed successfully
          app1 ->> cloud: Request data model of passcodeModel
          cloud -->> app1: Return data model of passcodeModel
          app1 ->> device: Request opening window of sharing and pairing
          device -->> device: Handle request
          device -->> app1: Return result of handling request
          app1 ->> app1: Display sharing QR code and setup code
      end
  end
end

note over device, app2: Sharing and pairing based on Multiple Fabrics
rect rgb(255, 233, 220)
  note over app2: Confirm running on the same LAN with device
  app2 ->> app2: Scan sharing QR code or enter setup code
  app2 ->> device: Share and pair device over Matter
  device -->> app2: Pairing ended
  app2 ->> app2: Use device
end
```

HomeKit SDK v5.2.0 or later simplifies the API used to generate the Multiple Fabrics sharing code and supports multiple communication channels for improved pairing rate.

8.1. Sharing among fabrics for SDK v5.2.0 or later

Before using the API, you can invoke a **channel availability check for SDK v5.2.0 or later** .

8.1.1. Open window for sharing and pairing

This API incorporates all the necessary methods to share devices across fabrics. You can request **the maximum number of supported fabrics**, **the number of used fabrics**, and **the SSID of the Wi-Fi network connected by the device**, and **open** and **close the window for sharing and pairing**.

API description

```
void sendEnhancedCommissioningCommand(boolean forceRefresh,
IThingMultipleFabricCallback callback);
```

Example

Java:

```
private void openECM() {
mThingMatterMultipleFabricDevice.sendEnhancedCommissioningCommand(true, new
IThingMultipleFabricCallback() {
@Override
public void onNetworkInfo(String ssid) {
    Log.d(TAG,"device ssId: "+ssid);
}

@Override
public void onSuccess(IThingMatterMultipleFabricDevice.SetupCodePayload
setupCodePayload) {
    Log.d(TAG,"open ecm success: "+setupCodePayload.toString());
}

@Override
public void onError(String errorCode, String errorMessage) {
    Log.d(TAG,"open ecm fail: "+errorCode);
}
});
```

8.2. Sharing among fabrics for SDKs earlier than v5.2.0

Implement the APIs used in the process of sharing devices across fabrics, as shown in the diagram above. Before using the API, you can invoke a **channel availability check for SDKs earlier than v5.2.0** .

8.2.1. Request maximum number of supported fabrics

The maximum number of supported fabrics can vary, depending on the performance of different Matter devices.

API description

```
void readSupportedFabrics(IThingDataCallback<Integer> callback);
```

Example

Java:

```
mThingMatterFabricDevice.readSupportedFabrics(new IThingDataCallback<Integer>()
{
  @Override
  public void onSuccess(Integer result) {
    Log.i("readSupportedFabrics", " SupportedFabrics:" + result);
  }

  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("readSupportedFabrics", "matter share errorCode:" + errorCode + "
errorMsg:" + errorMessage);
  }
});
```

8.2.2. Request number of used fabrics

API description

```
void readCommissionedFabrics(IThingDataCallback<Integer> callback);
```

Example

Java:

```
mThingMatterFabricDevice.readCommissionedFabrics(new
IThingDataCallback<Integer>() {
  @Override
  public void onSuccess(Integer result) {
    Log.i("readCommissionedFabrics", " currentCount:" + result);
  }

```

```
  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("readCommissionedFabrics", "matter share errorCode:" + errorCode + "
errorMsg:" + errorMessage);
  }
});
```

8.2.3. Get SSID of Wi-Fi network connected by device

Returns the SSID of the Wi-Fi network to which a Matter over Wi-Fi device is connected. This API method is unavailable to **wired gateways** and **Thread devices**.

API description

```
void getWifiDeviceSsid(IThingDataCallback<String> callback);
```

Example

Java:

```
mThingMatterFabricDevice.getWifiDeviceSsid(new IThingDataCallback<String>() {
  @Override
  public void onSuccess(String result) {
    Log.d("getWifiDeviceSsid", "--getWifiName:--=" + result);
  }

  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("getWifiDeviceSsid", "--getWifiName--error--=" + errorMessage);
  }
});
```

8.2.4. Open window for sharing and pairing

Returns the model object of `MultipleFabricPasscode` .

API description

```
// Get the object of MultipleFabricPasscode.
void getMultipleFabricPasscode(IThingDataCallback<MultipleFabricPasscode>
callback);

// Force refresh to get the object of MultipleFabricPasscode.
void
getMultipleFabricPasscodeForceRefresh(IThingDataCallback<MultipleFabricPasscode>
callback);
```

Example

Java:

```
mThingMatterFabricDevice.getMultipleFabricPasscode(new IThingDataCallback<IThingMatterMultipleFabricDevice.MultipleFabricPasscode>() {
  @Override
  public void onSuccess(IThingMatterMultipleFabricDevice.MultipleFabricPasscode result) {
    if (result == null) {
      return;
    }
    L.d("getMultipleFabricPasscode", "getMultipleFabricPasscode success");
    revokeCommissioningCommand(result);
  }

  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("getMultipleFabricPasscode", "matter share errorCode:" + errorCode + " errorMsg:" + errorMessage);
  }
});

mThingMatterFabricDevice.getMultipleFabricPasscodeForceRefresh(new IThingDataCallback<IThingMatterMultipleFabricDevice.MultipleFabricPasscode>() {
  @Override
  public void onSuccess(IThingMatterMultipleFabricDevice.MultipleFabricPasscode result) {
    if (result == null) {
      return;
    }
    Log.d("getMultipleFabricPasscodeForceRefresh", "getMultipleFabricPasscodeForceRefresh success");
  }

  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("getMultipleFabricPasscodeForceRefresh", "matter share errorCode:" + errorCode + " errorMsg:" + errorMessage);
  }
});
```

Uses the value of `passcodeModel` obtained in the previous step to open the window for sharing and pairing a device.

API description

```
void sendEnhancedCommissioningCommand(MultipleFabricPasscode multipleFabricPasscode, IThingDataCallback<SetupCodePayload> callback);
```

Example

Java:

```
mThingMatterFabricDevice.sendEnhancedCommissioningCommand(multipleFabricPasscode, new IThingDataCallback<IThingMatterMultipleFabricDevice.SetupCodePayload>() {
  @Override
```

```
    public void onSuccess(IThingMatterMultipleFabricDevice.SetupCodePayload
result) {
      Log.i("sendEnhancedCommissioningCommand", "matter share SetupCodePayload:" +
result);
    }

    @Override
    public void onError(String errorCode, String errorMessage) {

      Log.e("sendEnhancedCommissioningCommand", "matter share errorCode:" +
errorCode + " errorMsg:" + errorMessage);
    }
});
```

8.2.5. Close window for sharing and pairing

The window for sharing and pairing a device cannot be opened again when it is already opened. This window must be closed before it can be opened again.

API description

```
void revokeCommissioningCommand(IThingDataCallback<Void> callback);
```

Example

Java:

```
mThingMatterFabricDevice.revokeCommissioningCommand(new
IThingDataCallback<Void>() {
    @Override
    public void onSuccess(Void result) {
      Log.i("revokeCommissioningCommand", "success");
    }

    @Override
    public void onError(String errorCode, String errorMessage) {
      // API request error

      Log.e("revokeCommissioningCommand", "matter share errorCode:" + errorCode +
" errorMsg:" + errorMessage);
    }
});
```

9. Manage fabrics

Matter devices can be used on multiple Matter-enabled apps. The following sections describe how to get the list of fabrics among which a Matter device is shared and revoke the sharing permission.

> *i*
>
> HomeKit SDK v5.2.0 or later supports multiple communication channels for improved pairing success rate. The usage of fabrics management APIs remains unchanged.

9.1. Get list of fabrics

API description

```
void readFabrics(IThingDataCallback<List<OperationalFabricInfo>> callback);
```

Data model

The following table defines the fields in the model `OperationalFabricInfo` .

Property	Type	Description
vendorId	Integer	The ID of the manufacturer.
nodeId	Long	The value of `nodeId` provided by the specified fabric during pairing.
fabricId	Long	The value of `fabricId` provided by the specified fabric during pairing.
fabricIndex	String	The value of `index` provided by the specified fabric during pairing.
label	NSString	The value of `label` provided by the specified fabric during pairing.

Property	Type	Description
isCurrent	boolean	Indicates whether the current app is working with the specified fabric.

Example

Java:

```
public void readFabrics() {
  if (mThingMatterFabricDevice == null) {
    return;
  }
  mThingMatterFabricDevice.readFabrics(new IThingDataCallback<List<OperationalFabricInfo>>() {
    @Override
    public void onSuccess(List<OperationalFabricInfo> result) {
      if (result != null) {
        Log.d("readFabrics", "ReadFabrics success: Result = " + Arrays.toString(result.toArray()));
      }
    }

    @Override
    public void onError(String errorCode, String errorMessage) {
      Log.e("readFabrics", "ReadFabrics error: ErrorCode = " + errorCode + "; ErrorMessage = " + errorMessage);
    }
  });
}
```

9.2. Remove specified fabric

The `fabricIndex` of the fabric to be removed cannot be the one used by the current app. You can use the `isCurrent` field of `OperationalFabricInfo` to determine whether the current app is working with the target fabric.

API description

```
void removeFabric(Integer fabricIndex, IThingDataCallback<Integer> callback);
```

Example

Java:

```
mThingMatterFabricDevice.removeFabric(Integer.parseInt(fabricIndex), new IThingDataCallback<Integer>() {
  @Override
  public void onSuccess(Integer result) {
```

```
    if (result ==0){
      Log.i("removeFabric","RemoveFabric success");
    }
  }

  @Override
  public void onError(String errorCode, String errorMessage) {
    Log.e("removeFabric", "RemoveFabric error: ErrorCode = " + errorCode + ";
ErrorMessage = " + errorMessage);
  }
});
```

10. Error codes

Error codes	Meaning
3020	Failed to sign and issue the NOC chain.
3027	The device is offline on all channels.
3051	Failed to open the window for sharing and pairing a device.
3037	Failed to read device properties. Example: • Failed to get the number of times a device can be shared. • Failed to get the number of times a device has been used. • Failed to get the status of the device pairing window. • Failed to get `fabricList` . • Failed to get the basic information of a device.
3055	Failed to remove `fabricIndex` of a device.
3057	Failed to make the API request, for example, for fabric passcode.
3058	Failed to close the device pairing window.
3059	The number of available fabrics is zero.