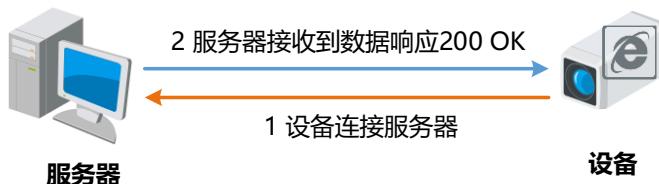


ISAPI 协议报警监听介绍

报警监听，是所需要集成的设备作为客户端，第三方服务器运行的程序作为服务端。设备不需要有固定 IP 地址，不需要登录设备，而服务器需要有固定 IP 地址，设备触发报警或者事件的时候主动连接服务器，连接成功之后上传报警或者事件数据，上传结束之后即断开连接，下次触发再重新连接和上传，是为短连接。



该集成方式，需要事先将服务器监听的 IP 和端口配置给设备，这个配置是需要登录设备的。并且，设备所在的网络也必须可以连接到服务器。

1 设备配置

因为是设备主动连接服务器，所以需要先将服务器 IP 和端口配置给设备，这样设备才能知道去连接哪个服务器。因为需要登录设备才能配置，所以需要知道设备 IP 地址、端口、用户名密码，可以在设备所在局域网进行登录配置。除了服务器监听 IP、端口的配置，其他相关事件参数比如检测区域、布防事件、联动方式（必须勾选上传中心）等也是必须配置的，本文章不做详细介绍，设备配置调试请参考设备操作手册或者咨询技术支持。

监听服务器 IP 和端口配置，这里介绍两种方式：一种是 Web 网页登录设备进配置界面手动设置，一种是直接使用 ISAPI 协议命令进行配置。

1.1 Web 网页配置

不同设备的网页配置界面不同，如下所示是典型的几种设备配置截图，IP 或者域名、端口都是配置的监听接收报警或者事件数据的服务器地址和端口，URL 是自定义的，测试按键可以测试服务器是否已经启动监听以及设备连接到服务器的网络通不通。

1. 某网络摄像机配置

高级配置

目的IP或域名	URL	协议类型	端口	测试
10.17.36.38	/test/data	HTTP	80	测试
0.0.0.0	/	HTTP	80	测试
0.0.0.0	/	HTTP	80	测试

保存

2. 某车牌识别抓拍机配置

系统设置

ANPR IP: 172.31.156.12
ANPR端口: 8090

3. 某客流量摄像机配置

数据上传

目的IP或域名	URL	端口	测试
10.17.35.33	/alarm	7203	测试
0.0.0.0	/	80	测试
0.0.0.0	/	80	测试

保存

4. 某脸谱配置



1.2 ISAPI 协议命令配置

首先，通过 GET /ISAPI/Event/notification/httpHosts/capabilities 获取能力集判断设备是否支持该功能以及支持的参数能力，其中<parameterFormatType>是判断数据格式是 XML 还是 JSON。

如下所示是 Postman 测试方法：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <HttpHostNotificationCap version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3      <hostNumber>1</hostNumber>
4      <curlen min="0" max="128"/>
5      <protocolType opt="HTTP"></protocolType>
6      <parameterFormatType opt="XML"/></parameterFormatType>
7      <addressingFormatType opt="ipaddress,hostname"></addressingFormatType>
8      <ipAddress opt="ipv4, ipv6"></ipAddress>
9      <portNo min="1" max="65535"/>
10     <userNameLen min="1" max="32"/>
11     <passwordLen min="1" max="16"/>
12     <httpAuthenticationMethod opt="none"></httpAuthenticationMethod>
13     <uploadImagesDataType opt="URL,binary"></uploadImagesDataType>
14 </HttpHostNotificationCap>

```

其次，通过 GET /ISAPI/Event/notification/httpHosts?format=json(设备支持的数据格式为 JSON 类型时需要使用这个 URL)或者 GET /ISAPI/Event/notification/httpHosts(设备支持的数据格式为 XML 类型时需要使用这个 URL)获取所有报警主机配置。

GET 10.17.36.241/ISAPI/Event/notification/httpHosts

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Co

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Status: 200 OK Time: 8 ms Size: 754 B Save Response

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <HttpHostNotificationList version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3   <HttpHostNotification version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
4     <id>1</id>
5     <url></url>
6     <protocolType>HTTP</protocolType>
7     <parameterFormatType>XML</parameterFormatType>
8     <addressingFormatType>ipaddress</addressingFormatType>
9     <ipAddress>0.0.0.0</ipAddress>
10    <portNo>80</portNo>
11    <userName></userName>
12    <httpAuthenticationMethod>none</httpAuthenticationMethod>
13  </HttpHostNotification>
14 </HttpHostNotificationList>

```

然后，通过 PUT /ISAPI/Event/notification/httpHosts?format=json(设备支持的数据格式为 JSON 类型时需要使用这个 URL)或 PUT /ISAPI/Event/notification/httpHosts(设备支持的数据格式为 XML 类型时需要使用这个 URL)设置报警主机配置。

PUT 10.17.36.241/ISAPI/Event/notification/httpHosts

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Co

none form-urlencoded raw binary GraphQL Text

这里需要选择digest摘要认证，输入设备用户名密码

输入配置内容，根据GET获取的数据进行修改设置

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <HttpHostNotificationList version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3   <HttpHostNotification version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
4     <id>1</id>
5     <url>alarm/test</url>
6     <protocolType>HTTP</protocolType>
7     <parameterFormatType>XML</parameterFormatType>
8     <addressingFormatType>ipaddress</addressingFormatType>
9     <ipAddress>10.17.36.38</ipAddress>
10    <portNo>80</portNo>
11    <username></username>
12    <httpAuthenticationMethod>none</httpAuthenticationMethod>
13  </HttpHostNotification>
14 </HttpHostNotificationList>

```

需要接收报警或者事件信息的服务器地址、端口

Status: 200 OK Time: 43 ms Size: 474 B Save Response

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <ResponseStatus version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3   <requestURL>/ISAPI/Event/notification/httpHosts</requestURL>
4   <statusCode>1</statusCode>
5   <statusString>OK</statusString>
6   <subStatusCode>ok</subStatusCode>
7 </ResponseStatus>

```

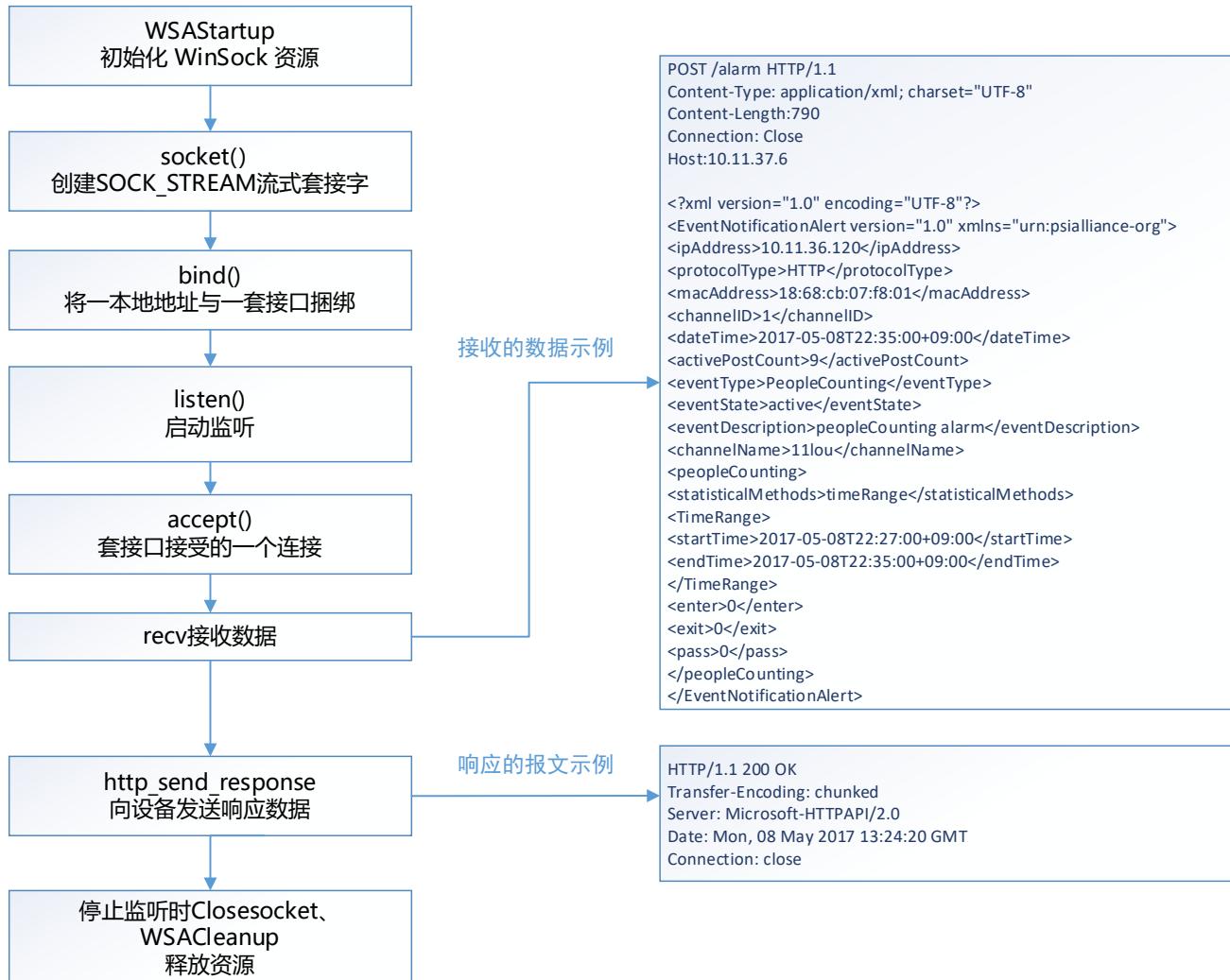
2 协议介绍(以 C++ 为例)

典型的 HTTP 协议处理过程如下所示：

- (1) 设备与服务器建立连接，TCP 的 3 次握手；
- (2) 设备触发事件并且上传，上传 HTTP 协议的报文数据，JSON 或者 XML 格式，如果有图片数据则以表单方式在 XML 或者 JSON 数据后面上传。
- (3) 服务器接收到数据之后需要响应设备，响应状态为 200 OK。

(4) 设备接收到响应报文之后断开连接。

ISAPI 协议基于标准 HTTP REST 架构，而 HTTP 协议使用的是 TCP 协议，因此设备创建连接也是直接使用套接字(socket)，使用 SOCK_STREAM 流式套接字。



注：流程里面的接口详见 socket 通信系统头文件 winsock2.h。

3 C#语言实现

C#开发，可以参考我司 C#AppsDemo 的 Demo 示例，直接使用 HttpListener 的类。

1) 启动监听：

```

public void ISAPI_StartListen(string[] prefixes)
{
    try
    {

```

```

    {
        if (!HttpListener.IsSupported)
        {
            Console.WriteLine("Windows XP SP2 or Server 2003 is required to use the HttpListener class.");
            return;
        }
        // URI prefixes are required,
        if (prefixes == null || prefixes.Length == 0)
            throw new ArgumentException("prefixes");

        // Add the prefixes.
        foreach (string s in prefixes)
        {
            listener.Prefixes.Add(s);
        }
        listener.Start();
        AlarmInforTBox.Text = "Listening...";

        listener.BeginGetContext(new AsyncCallback(ListenerCallback), listener);
        MessageBox.Show("Start Listen Success!", "Information", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Start Listen Failed!" + ex.Message.ToString(), "Information", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
}

```

2) 接收数据和响应：

```

public void ListenerCallback(IAsyncResult result)
{
    try
    {
        HttpListener listener = (HttpListener)result.AsyncState;
        // Call EndGetContext to complete the asynchronous operation.
        HttpListenerContext context = listener.EndGetContext(result);
        HttpListenerRequest request = context.Request;
        // && request.RawUrl.Contains(request.UserHostAddress)
        if (request.HttpMethod == "POST")
        {
            // Obtain a response object.
            context.Response.StatusCode = 200;

            //deal post
            Stream stream = context.Request.InputStream;
            //System.IO.StreamReader reader = new System.IO.StreamReader(stream, Encoding.UTF8);
        }
    }
}

```

```

//String body = reader.ReadToEnd();

List<byte> byteList = new List<byte>();
if (request.ContentType.ToLower().Contains("multipart"))
{
    //表单类型 造个假HTTP头，便于后面解析boundary
    byteList.AddRange(Encoding.Default.GetBytes("Content-Type: " + request.ContentType +
"\r\n\r\n"));
}

byte[] buffer = new byte[10240];
using (MemoryStream ms = new MemoryStream())
{
    while (true)
    {
        int read = stream.Read(buffer, 0, buffer.Length);
        if (read <= 0)
        {
            break;
        }
        ms.Write(buffer, 0, read);
    }
    byteList.AddRange(ms.ToArray());
}

//stream.Read(streamBytes, 0, streamBytes.Length);

// 设置当前流的位置为流的开始
//stream.Seek(0, SeekOrigin.Begin);

//string alarmInfo = body.ToString();
byte[] streamBytes = byteList.ToArray();
m_SyncContext.Post(SetTextSafePost, streamBytes);
context.Response.Close();
}

listener.BeginGetContext(new AsyncCallback(ListenerCallback), listener);
}
catch (Exception ex)
{
    string test = ex.Message.ToString();
}
}

```

3) 停止监听:

```

public void ISAPI_StopListen()
{

```

```
        try
        {
            listener.Stop();
            MainFormHandler.SetStatusString(MainFormHandler.Level.Info, "ISAPI_StopListen", "Stop Listen Success!");
        }
        catch (Exception ex)
        {
            MainFormHandler.SetStatusString(MainFormHandler.Level.Error, "ISAPI_StopListen", "Stop Listen Failed!", ex.Message);
        }
    }
```

4 Java 语言实现

Java 开发，可以参考我司 JavaDemo，直接使用 HttpServer 的类。

```
package ISAPI;

import java.awt.BorderLayout;
import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import com.sun.net.httpserver.spi.HttpServerProvider;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;

import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetSocketAddress;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import java.awt.Color;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JLabel;
import java.awt.Window.Type;
```

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class ListenServer_Recv extends JDialog {

    private final JPanel contentPanel = new JPanel();
    private JTextField ListenServerPort;

    JTextArea textListenServerPane = new JTextArea();
    JsonFormatTool JsonFormatTool = new JsonFormatTool();

    public HttpServer httpserver = null;
    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        try {
            ListenServer_Recv dialog = new ListenServer_Recv();
            dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
            dialog.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Create the dialog.
     */
    public ListenServer_Recv() {
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent arg0) {
                httpserver.stop(1);
            }
        });
        setType(Type.UTILITY);
        setTitle("ListenServer_Recv");
        setBounds(100, 100, 436, 629);
        getContentPane().setLayout(new BorderLayout());
        contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
        getContentPane().add(contentPanel, BorderLayout.CENTER);
        contentPanel.setLayout(null);

        JLabel lblNewLabel_1 = new JLabel("Listen Server Port");
        lblNewLabel_1.setBounds(10, 557, 115, 15);
        contentPanel.add(lblNewLabel_1);
```

```
ListenServerPort = new JTextField();
ListenServerPort.setBounds(135, 554, 66, 21);
ListenServerPort.setText("8080");
contentPanel.add(ListenServerPort);
ListenServerPort.setColumns(10);

 JButton btnStartListen = new JButton("Start Listen");
btnStartListen.setBounds(225, 553, 115, 23);
btnStartListen.addActionListener(new ActionListener() {

    class MyHttpHandler implements HttpHandler{

        public void handle(HttpExchange httpExchange) throws IOException {

            String requestMethod=httpExchange.getRequestMethod();
            if (requestMethod.equalsIgnoreCase("POST")){
                InputStreamReader ISR=new
InputStreamReader(httpExchange.getRequestBody(),"utf-8");
                BufferedReader br =new BufferedReader(ISR);

//处理接收到的数据

                String strout="";
                String temp="";
                while((temp = br.readLine()) != null) {
                    strout=strout+temp;
                }

httpExchange.sendResponseHeaders(200, 0); //响应200 OK
                httpExchange.close();

                textListenServerPane.setText(JsonFormatTool.formatJson(strout));
            }
        }
    }
}

public void Listener(){
    try {

        if(btnStartListen.getText()=="Start Listen")
        {
            HttpServiceProvider provider = HttpServiceProvider.provider();
```

```
        httpserver = provider.createHttpServer(new
InetSocketAddress(Integer.parseInt(ListenServerPort.getText()), 100);
        httpserver.createContext("/alarm", new MyHttpHandler());
        httpserver.setExecutor(null);

        httpserver.start(); //启动监听

        textListenServerPane.setText("Listen Start success");

        btnStartListen.setText("Stop Listen");
    }
else
{
    httpserver.stop(1);
    textListenServerPane.setText("Listen Stop success");

    btnStartListen.setText("Start Listen");
}

} catch (IOException e) {
    // TODO Auto-generated catch blocks
    e.printStackTrace();

    textListenServerPane.setText("Listen error");
}//监听端口8080,能同时接 受100个请求

}

public void actionPerformed(ActionEvent arg0) {
    Listener();
}

});
contentPanel.add(btnStartListen);

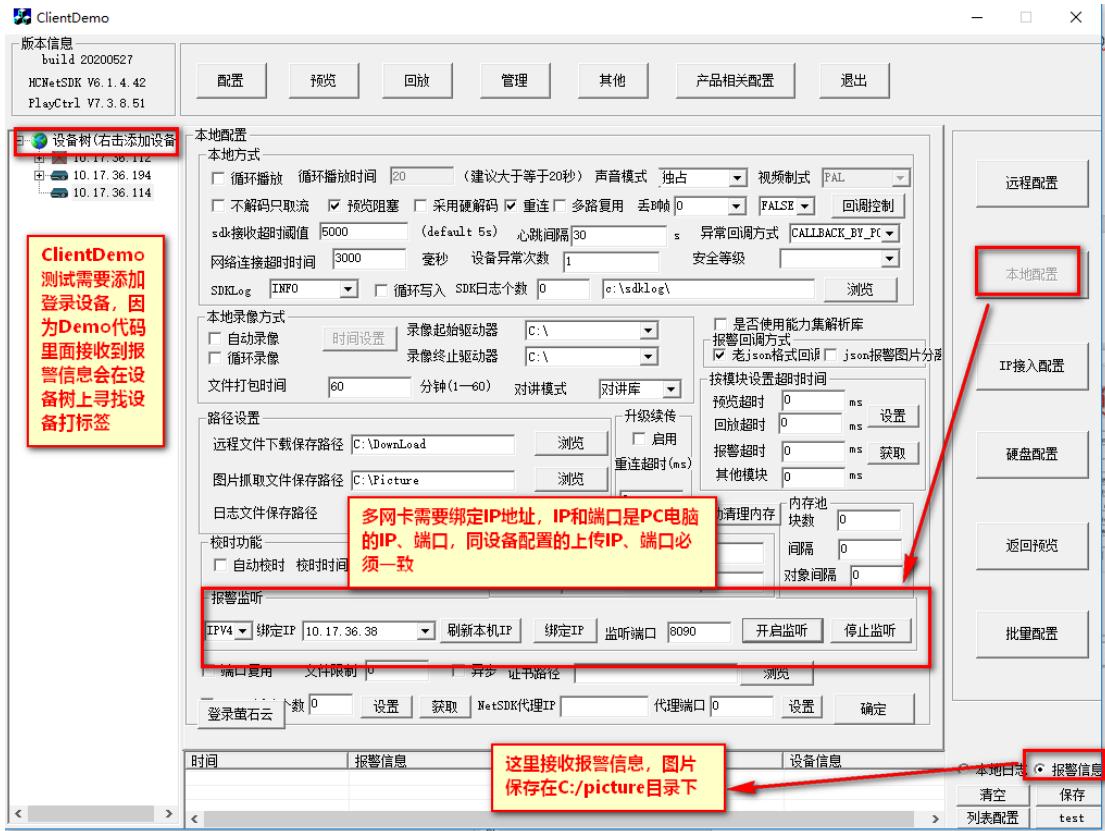
JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(10, 10, 404, 533);
contentPanel.add(scrollPane);
scrollPane.setViewportView(textListenServerPane);
{

JPanel buttonPane = new JPanel();
getContentPane().add(buttonPane, BorderLayout.SOUTH);
{
    JButton okButton = new JButton("OK");
    okButton.setBounds(316, 5, 45, 23);
    okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
        }
    });
    buttonPane.setLayout(null);
}
```

```
{  
    JLabel lblNewLabel = new JLabel("New label");  
    lblNewLabel.setBounds(257, 9, 54, 15);  
    buttonPane.add(lblNewLabel);  
}  
okButton.setActionCommand("OK");  
buttonPane.add(okButton);  
getRootPane().setDefaultButton(okButton);  
}  
{  
    JButton cancelButton = new JButton("Cancel");  
    cancelButton.setBounds(366, 5, 69, 23);  
    cancelButton.setActionCommand("Cancel");  
    buttonPane.add(cancelButton);  
}  
}  
}  
}
```

5 ClientDemo 测试

设备网络 SDK 也是集成了 HTTP 协议的,如果没有测试程序,也可以直接使用设备网络 SDK 的 ClientDemo 测试,如图所示,本地配置里面报警监听,选择本地网卡 IP 地址,然后监听端口也是 PC 电脑本地端口 (空闲的端口),然后“启动监听”即可,接收到的报警信息在报警信息列表里面, ClientDemo 的最右下角选择“报警信息”可以查看。



如果接收不到数据, 可以抓包分析, 设备触发事件之后, 先分析抓包里面的 TCP 三次握手连接是否正常, 再分析设备上传的 HTTP 报文 (XML 或者 JSON), 最后看看 200 OK 的响应。如果服务器端接收到设备报文数据没有响应 200 OK, 设备就不能正常发送后面的事件数据。