

ISAPI 协议报警布防介绍

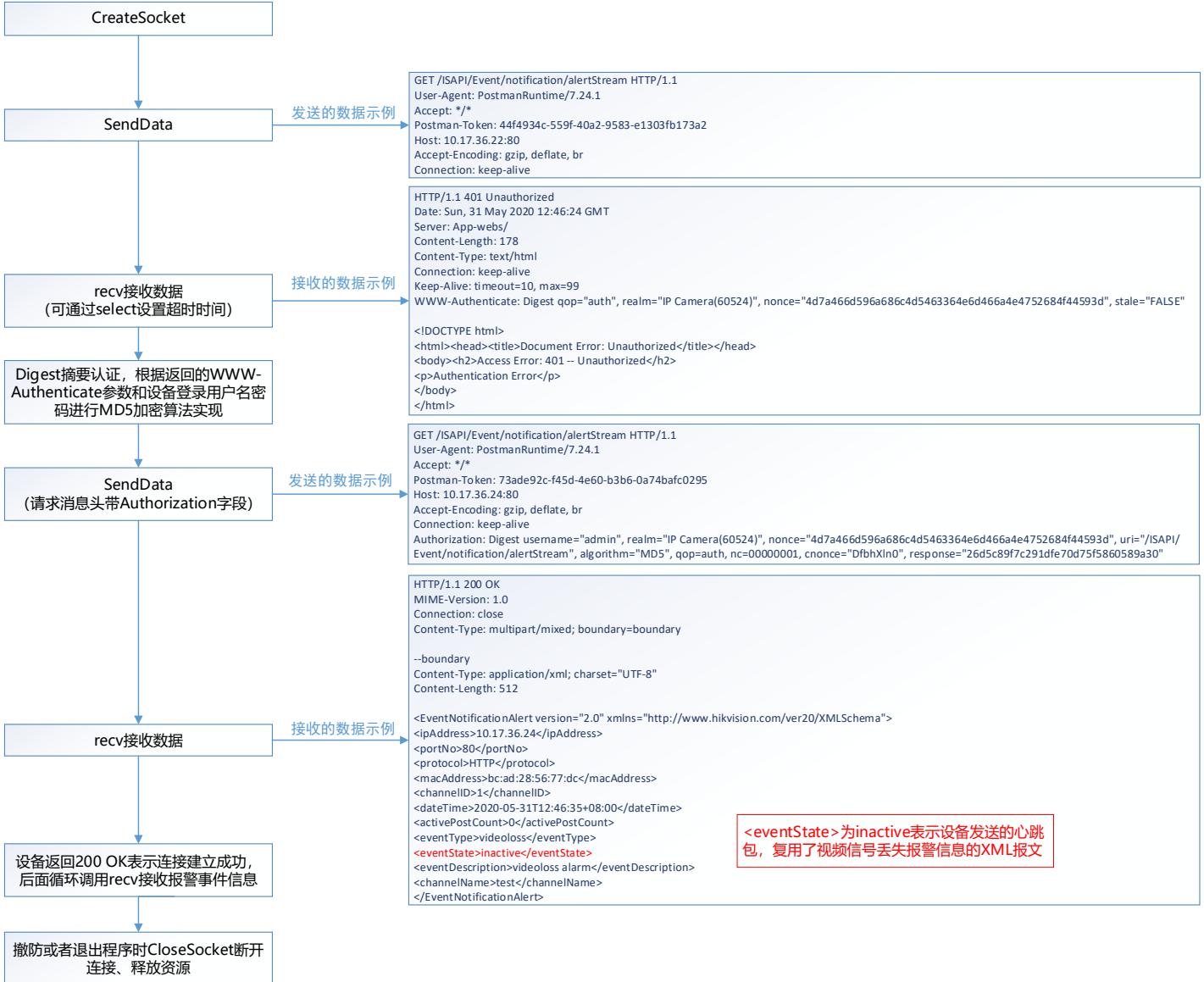
1 流程介绍(以 C++为例)

典型的 HTTP 协议处理过程如下所示:

- (1) 客户与服务器建立连接;
- (2) 客户向服务器提出请求;
- (3) 服务器接受请求，并根据请求返回相应的文件作为应答;
- (4) 客户与服务器关闭连接。

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

1. 报警布防方式：设备作为服务端，程序作为客户端，客户端主动与设备建立 TCP 连接，客户端通过建立的长连接持续接收设备主动上传的报警或事件信息。从 HTTP/1.1 起默认都开启了 Keep-Alive（http 头加入或者不加入"Connection: Keep-Alive"都可以），可以保持连接特性，即我们说的长连接。Keep-Alive 不会永久保持连接，它有一个保持时间，这个时间由服务器即设备决定。另外，我们设备也会自己定时发送应用层的心跳包的方式来判断连接是客户端创建 socket 连接。
2. 客户端向设备发送 GET /ISAPI/Event/notification/alertStream 的布防连接请求。
3. 客户端接收到设备返回 401 认证失败的信息。
4. 客户端需要使用 digest 摘要认证方式，根据设备返回的 WWW-Authenticate 参数和设备登录用户名密码进行 MD5 加密算法实现。
5. 重新发送 GET /ISAPI/Event/notification/alertStream 的布防连接请求，请求消息头需要带正确的 Authorization 认证信息。可以基于前面创建的 Socket 连接发送，也可以 CloseSocket 关闭前面的连接，再重新 CreateSocket。
6. 客户端接收到设备返回 200 OK 的响应，表示连接建立成功，后面客户端一直等待接收设备主动上传的报警事件信息（包含心跳包）即可，接收到报警信息或者心跳包之后客户端需要回复设备一个 ACK 数据。
7. 客户端如果不需要再接收报警信息，直接 CloseSocket 关闭 socket 连接。设备端上传数据之后一直接收不到客户端 ACK 响应（一般是连续 3 次上传没有收到响应）即认为连接断开，从而释放设备端的布防连接资源。



注：流程里面的接口详见 socket 通信系统头文件 WinSock2.h。

2 C#语言实现

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

```
public int StartHttpLongLink(string strUserName, string strPassword, string strUrl, string strparam, ProcessLongLinkData processLongLinkData, ref string strResponse, {  
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(strUrl);  
    request.Credentials = GetCredentialCache(strUrl, strUserName, strPassword);  
    request.Method = strHttpMethod;  
  
    if (!string.IsNullOrEmpty(strparam))  
    {  
        byte[] bs = Encoding.ASCII.GetBytes(strparam);  
        request.ContentType = "application/x-www-form-urlencoded";  
        request.ContentLength = bs.Length;  
        using (Stream reqStream = request.GetRequestStream())  
        {  
            reqStream.Write(bs, 0, bs.Length);  
        }  
    }  
  
    try  
    {  
        RequestState myRequestState = new RequestState();  
        myRequestState.request = request;  
        myRequestState.processLongLinkData = processLongLinkData;  
        IAsyncResult ret = request.BeginGetResponse(new AsyncCallback(RespCallback), myRequestState);  
  
        if (bBlock)  
        {  
            int nTimeoutLimit = m_iHttpTimeOut / 100;  
            int nTimeoutCount = 0;  
            while (!ret.IsCompleted && nTimeoutCount < nTimeoutLimit)  
            {  
                Thread.Sleep(100);  
                nTimeoutCount++;  
            }  
  
            if (nTimeoutCount == nTimeoutLimit)  
            {  
                request.Abort();  
            }  
  
            if (myRequestState.response != null && myRequestState.response.StatusCode == HttpStatusCode.OK)  
        }  
    }  
}
```

摘要认证

下发布防请求

设置回调函数接收报警事件

摘要认证可以直接使用 CredentialCache 类。

```
private CredentialCache GetCredentialCache(string sUrl, string strUserName, string strPassword)  
{  
    if (_credentialCache == null)  
    {  
        _credentialCache = new CredentialCache();  
        _credentialCache.Add(new Uri(sUrl), "Digest", new NetworkCredential(strUserName, strPassword));  
        strURL = sUrl;  
    }  
    if (strURL != sUrl)  
    {  
        _credentialCache.Add(new Uri(sUrl), "Digest", new NetworkCredential(strUserName, strPassword));  
        strURL = sUrl;  
    }  
  
    return _credentialCache;  
}
```

回调函数里面解析接收到的数据：

```
private static void RespCallback(IAsyncResult asynchronousResult)
{
    // State of request is asynchronous.
    RequestState myRequestState = (RequestState)asynchronousResult.AsyncState;
    try
    {
        HttpWebRequest myHttpWebRequest = myRequestState.request;
        myRequestState.response = (HttpWebResponse)myHttpWebRequest.EndGetResponse(asynchronousResult);

        string strBoundary = myRequestState.response.ContentType;
        int nIndex = strBoundary.IndexOf("boundary=");
        if (nIndex >= 0)
        {
            strBoundary = strBoundary.Substring(nIndex + "boundary=".Length);
            myRequestState.strBoundary = strBoundary;
        }

        // Read the response into a Stream object.
        Stream responseStream = myRequestState.response.GetResponseStream();
        myRequestState.streamResponse = responseStream;

        // Begin the Reading of the contents of the HTML page and print it to the console.
        IAsyncResult asynchronousInputRead = responseStream.BeginRead(myRequestState.BufferRead, 0, BUFFER_SIZE, new AsyncCallback(ReceiveData), myRequestState);
        return;
    }
    catch (WebException e)
    {
        myRequestState.eStatus = e;
    }
}
```

3 Java 语言实现

Java 开发，可以参考我司 JFinal 的 JavaDemo，直接使用 CloseableHttpClient 的类。

```
package communicationCom;

import java.io.IOException;
import java.nio.CharBuffer;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Future;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.concurrent.FutureCallback;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.CloseableHttpAsyncClient;
import org.apache.http.impl.nio.client.HttpAsyncClients;
import org.apache.http.nio.IOControl;
import org.apache.http.nio.client.methods.AsyncCharConsumer;
import org.apache.http.nio.client.methods.HttpAsyncMethods;
import org.apache.http.nio.protocol.HttpAsyncRequestProducer;
import org.apache.http.protocol.HttpContext;
import alarm.ParseAlarmData;

public class HttpAysncClientUtil {
```

```
public static int iPort = 0;
public static String strIP="";
public static CloseableHttpClient httpAsyncclient;

private static int reconnect=3;
private static int timeout=10000;
private static boolean stoplink=false;

private static boolean DataRecv=false;
private static List<Character>chBuffer=new ArrayList<Character>();
private static ParseAlarmData AlarmData=new ParseAlarmData();

//Initializes a long connection communication object
public static void HttpAysncInit(String user,String password)
{
    //摘要认证
    CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
    credentialsProvider.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials(user, password));
    httpAsyncclient = HttpAsyncClients.custom()
        .setDefaultCredentialsProvider(credentialsProvider)
        .build();

}

//Long connection function
public static void LonLink(String Url ,String event, boolean subscribe)
{
    stoplink=false;
    chBuffer.clear();
    try {
        //设置回调函数
        FutureCallback<Boolean> callback = new FutureCallback<Boolean>() {
            @Override
            public void cancelled() {
                // TODO Auto-generated method stub
                System.out.println("cancelled");
            }
            @Override
            public void completed(Boolean arg0) {
                // TODO Auto-generated method stub
                System.out.println("completed");
            }
            @Override
            public void failed(Exception arg0) {
                // TODO Auto-generated method stub
                System.out.println("failed");
            }
        };
        httpAsyncclient.start();
    }
}
```

```

//Reconnect the query thread with a timeout on
ReConnect recn=new ReConnect();
Thread Rethread =new Thread(rexn);
Rethread.start();

//创建连接，设置接收报警事件的回调函数
// Url="http://"+ip+":"+port+"/ISAPI/Event/notification/alertStream";
Future<Boolean> future = httpAsyncclient.execute(
    HttpAsyncMethods.createGet(Url),
    new ResponseConsumer(), callback);

Boolean result = future.get();
if (result != null && result.booleanValue()) {
    System.out.println("Request successfully executed");
} else {
    System.out.println("Request failed");
}
System.out.println("Shutting down");

} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

return;
}

public static void StopLink()
{
    stoplink=true;
    DataRecv=false;
}

//接收和解析报警事件信息
static class ResponseConsumer extends AsyncCharConsumer<Boolean> {

    //Message type
    public String type;
    @Override
    protected void onResponseReceived(final HttpResponse response) {
        //Determine the message type
        System.out.println("onResponseReceived");
        String tbuf=response.toString();
        if(tbuf.contains("multipart"))
        {
            type="multipart";
        }else if(tbuf.contains("xml"))
        {
            type="xml";
        }else if(tbuf.contains("json"))
        {
            type="json";
        }
    }
}

```

```

    {
        type="json";
    }
}

//Callback function to receive a message
@Override
protected void onCharReceived(final CharBuffer buf, final IOControl ioctl) throws IOException {

    DataRecv=true;
    //Parsing by message type
    if(type.equals("multipart"))
    {
        int length=buf.length();
        for(int i=0;i<buf.length();i++)
        {
            //Fill buffer
            chBuffer.add(buf.charAt(i));
        }
        //Form data parsing s
        AlarmData.parseMultiData(chBuffer);
    }else if(type.equals("xml"))
    {
        int length=buf.length();
        for(int i=0;i<buf.length();i++)
        {
            //Fill buffer
            chBuffer.add(buf.charAt(i));
        }
        //Form data parsing s
        AlarmData.parseMultiData(chBuffer);
    }else if(type.equals("json"))
    {
        int length=buf.length();
        for(int i=0;i<buf.length();i++)
        {
            //Fill buffer
            chBuffer.add(buf.charAt(i));
        }
        //Form data parsing s
        AlarmData.parseMultiData(chBuffer);
    }

    if(stoplink)
    {
        buf.clear();
        chBuffer.clear();
        this.close();
        stoplink=false;
    }
}

```

```

    }

    @Override
    protected Boolean buildResult(final HttpContext context) {
        return Boolean.TRUE;
    }
}

```

4 Postman 测试

首先，根据布防协议文档说明，输入对接设备的 IP、端口以及布防 URI 命令，Authorization 里面选择 Digest 摘要认证方式，输入设备的用户名、密码。

The screenshot shows the Postman interface with a red box highlighting the 'GET方法, 输入URL' (Method, input URL) section. The URL field contains 'http://10.17.36.30:80/ISAPI/Event/notification/alertStream'. Below the URL, the 'Authorization' tab is selected in the Params dropdown, and a yellow box highlights the '选择Digest摘要认证类型, 输入设备登录用户名和密码' (Select Digest authentication type, enter device login username and password) instruction. The 'Username' field is filled with 'admin' and the 'Password' field with 'hik123456'. A checkbox for 'Show Password' is checked.

然后，如果是非订阅方式，URI 为/ISAPI/Event/notification/alertStream，没有输入参数，直接“Send”发送请求即可。

发送请求之后建立成功可以参考如下图的 Status 状态，200 OK 表示建立成功。报警事件信息和心跳包等数据在 Postman 界面查看不到，可以同时运行 Wireshark 抓包，查看具体内容。

Postman

File Edit View Help

New Import Runner

My Workspace Invite

No Environment

Comments

Sign In

Filter

History Collections APIs + New API

Sign in to create APIs

APIs define related collections and environments under a consistent schema.

Sign in to create APIs

Untitled Request

GET http://10.17.36.30:80/lSAPI/Events/notification/alertStream

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

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

布防非订阅模式，不需要输入参数

Send Save Cookies Code

Status: 200 OK Time: 4.22s Size: 117 B Cancel

建立连接成功

Postman没有显示接收到的报警信息，界面上不能直接查看，因此可以同时运行wireshark抓包工具，抓包查看报警事件数据内容。

*以太网

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

ip.addr==10.17.36.30&tcp.port==80 过滤设备IP、端口

No.	Time	Source	Destination	Protocol	Length	Info
20	2020-05-31 14:37:28.286868	10.17.36.38	10.17.36.30	TCP	55	15837 → 80
21	2020-05-31 14:37:28.287236	10.17.36.30	10.17.36.38	TCP	66	80 → 15837
32	2020-05-31 14:37:29.287765	10.17.36.38	10.17.36.30	TCP	55	[TCP Keep-Alive]
33	2020-05-31 14:37:29.288168	10.17.36.30	10.17.36.38	TCP	66	[TCP Keep-Alive]
40	2020-05-31 14:37:30.213946	10.17.36.30	10.17.36.38	TCP	388	80 → 15837
41	2020-05-31 14:37:30.253841	10.17.36.38	10.17.36.30	TCP	54	15837 → 80
58	2020-05-31 14:37:31.213702	10.17.36.38	10.17.36.30	TCP	55	[TCP Keep-Alive]
59	2020-05-31 14:37:31.213998	10.17.36.30	10.17.36.38	TCP	66	[TCP Keep-Alive]
62	2020-05-31 14:37:32.214113	10.17.36.38	10.17.36.30	TCP	55	[TCP Keep-Alive]
63	2020-05-31 14:37:32.214426	10.17.36.30	10.17.36.38	TCP	66	[TCP Keep-Alive]
83	2020-05-31 14:37:33.214297	10.17.36.38	10.17.36.30	TCP	55	[TCP Keep-Alive]

> Frame 40: 388 bytes on wire (3104 bits), 388 bytes captured (3104 bits) on interface 0
 > Ethernet II, Src: 98:df:82:8b:f9:c7 (98:df:82:8b:f9:c7), Dst: Liteon_ab:b8:bf (6c:4b:90:ab:b8:bf)
 > Internet Protocol Version 4, Src: 10.17.36.30, Dst: 10.17.36.38
 > Transmission Control Protocol, Src Port: 80, Dst Port: 15837, Seq: 1, Ack: 2, Len: 334

No.	Time	Source	Destination	Protocol	Length	Info
0000	01 ea 0b a3 00 00	2d 2d	4d 49 4d 45 5f 62 6f 75		1K..... E-	
0010	01 76 24 e5 40 00	40 06	b8 37 0a 11 24 1e 0a 11		.v\$.@-@- . 7 - \$ - -	
0020	24 26 00 50 3d dd	df b7	ea 39 21 6c 6d e6 50 18		\$& - P= - - - 9!1m - P-	
0030	01 ea 0b a3 00 00	2d 2d	4d 49 4d 45 5f 62 6f 75		- - - - - MIME_bou	
0040	6e 64 61 72 79 0d	0a 43	6f 6e 74 65 6e 74 2d 54		n dary .. C ontent-T	
0050	79 70 65 3a 20 61	70 70	6c 69 63 61 74 69 6f 6e		ype: app lication	
0060	2f 6a 73 6f 6e 3b	20 63	68 61 72 73 65 74 3d 22		/json; c harset="	
0070	55 54 46 2d 38 22	0d 0a	43 6f 6e 74 65 6e 74 2d		UTF-8" .. Content-	
0080	4c 65 6e 67 74 68	3a 20	32 34 33 0d 0a 0d 0a 7b		Length: 243...{	
0090	0a 09 22 69 70 41	64 64	72 65 73 73 22 3a 09 22		.. "ipAdd ress": "	
00a0	31 30 2e 31 37 2e	33 36	2e 33 38 22 2c 0a 09 22		10.17.36 .38", .."	
00b0	70 6f 72 74 4e 6f	22 3a	09 38 30 2c 0a 09 22 70		portNo": .80, .."p	
00c0	72 6f 74 6f 63 6f	6c 22	3a 09 22 48 54 54 50 22		rotocol" :"HTTP"	
00d0	2c 0a 09 22 63 68	61 6e	6e 65 6c 49 44 22 3a 09		, .."chan nelID": .	
00e0	31 2c 0a 09 22 64	61 74	65 54 69 6d 65 22 3a 09		1, .."dat eTime": ..	

接收到的报警信息

wireshark_{120CD42-45A8-4924-9B2E-F9ECCAA3F5BF}_20200531143726_a08644.pcapng 分组: 4886 已显示: 827 (16.9%) 配置: Default

Follow TCP 查看报文内容:

Wireshark · 追踪 TCP 流 (tcp.stream eq 13) · 以太网

```

Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Authorization: Digest username="admin", realm="DS-2ED2A36F", nonce="NmVmZDIzMGM3MjR1OGZkYmM2NzV1OTF1NDBkYzk2MWQ=", uri="/ISAPI/Event/notification/alertStream", algorithm="MD5", qop=auth, nc=00000001, cnonce="jTrNBcAF", response="e336567eff42f918f00e37896a8144f1"

HTTP/1.1 200 OK
MIME-Version: 1.0
Connection: keep-alive
Content-Type: multipart/mixed; boundary=MIME_boundary

Content-Length: 334

--MIME_boundary
Content-Type: application/json; charset=UTF-8
Content-Length: 243

{
    "ipAddress": "10.17.36.38",
    "portNo": 80,
    "protocol": "HTTP",
    "channelID": 1,
    "dateTime": "2020-05-31T14:52:05+08:00",
    "activePostCount": 1,
    "eventType": "videoloss",
    "eventState": "inactive",
    "eventDescription": "videoloss alarm"
}

```

订阅方式报警布防：

POST 方法，URI 是 /ISAPI/Event/notification/subscribeEvent，同样选择 Digest 摘要认证，需要输入订阅参数。

POST http://10.17.36.20:80/ISAPI/Event/notification/subscribeEvent

```

<SubscribeEvent version="2.0" xmlns="http://www.isapi.org/ver20/XMLSchema">
<format>json</format>
<heartbeat>5</heartbeat>
<eventMode>all</eventMode>
</SubscribeEvent>

```