AXIS 700 Application Integration

Introduction

Scanning using your AXIS 700 can be divided into two types:

- Pull Type Scanning, which is initiated from an application and performed by using the HTTP protocol
- Push Type Scanning, which is initiated from the control panel of your AXIS 700 and performed by using the FTP protocol.

This document describes how to integrate your AXIS 700 using both of these methods. It also describes how you can use FTP to obtain, and possibly modify, a list of available profiles. It contains the following information:

- Scanning
  - HTTP Commands and Headers Required
  - Pull Type Scanning
  - Push Type Scanning
  - Using Microsoft's Foundation Classes.
  - Using Microsoft's Internet API
- Using the *config.ini* file to add, modify and remove profiles.

Axis Communications AB

September 1st, 1998

Table of Contents

Scanning

Once you have decided which one of the available profiles you want to use, you can proceed with the scanning.

Pull Type Scanning

Pull type scanning uses the HTTP protocol. To do this, you need to emulate Scan-to-Web from your application.

HTTP Command Sequence

When scanning from the **Scan Document** page in your Web browser:

1. Select a profile from the profile list.
2. Click either the **Scan** button or the **Reserve** button.

A **POST** request is sent to your AXIS 700. It contains the name of the page, the name of the selected profile, and the name of the button you selected.

Your AXIS 700 makes sure that:

- A scanner is connected and supported.
- The scanner is not busy.
- The selected profile exists.
- Users are allowed to do Scan-to-Web.

When all of these conditions are met you will receive an HTTP response stating that the document has moved temporarily, along with the URL to the image itself.

However, if some of the conditions are not met, e.g. the scanner is busy, the response will contain the complete **Scan Document** page, including an error message at the top.

If you received the **Moved Temporarily** response, you should send a **GET** request to get the URL you received in the response.

Follow the steps below to scan an image to your application using your AXIS 700.

1. Send a HTTP request with, at least, the following headers:

```
POST /user/scan/scan.shtml  HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 40

page=scan&profileName=Color+low&scanNow=
```

The last row contains the parameters to your AXIS 700.

page=scan	- necessary for your AXIS 700 to process your request.
profileName=Color+low	- tells your AXIS 700 which profile to use.
scanNow	- immediate scan. No value is needed.

If you replace *scanNow* with *reserve* then the scanner is reserved, but the user is required to push the **Send** button on the front panel of your AXIS 700. This is the same as clicking the **Reserve** button on the **Scan Document** page.

Your AXIS 700 will respond with either status **302 Moved temporarily** or **200 OK**.

If you receive the **200 OK** response then something has gone wrong, e.g. the scanner could be busy. The content of the response is a complete **Scan Document** page, including an error message.

The **302 Moved Temporarily** response means that the scanning is initiated.

2. The **302 Moved Temporarily** response includes the URL to the scanned image. It is on the form /image.xxx where xxx is one of pdf, jpg, or tif. The table below lists the image name and MIME type for the different image types.

Image Type	**Image Name**	**MIME Type**
PDF	image.pdf	application/pdf
TIFF	image.tif	image/tiff
JPEG	image.jpg	image/jpeg

```
Http/1.0 302 Moved Temporarily
Content-Length: 169
Content-Type: text/html
Location: /image.jpg

<head><title>Moved Temporarily</title></head>
<body><h2>Moved Temporarily!</h2>
<p>The requested resource has been temporarily moved
to a new location.
</p>
</body>
```

The image data can now be retrieved by sending a **GET** request to your AXIS 700 for the URL. The request would look something like this:

```
GET /image.jpg HTTP/1.0
Accept: image/jpeg, */*
```

Your AXIS 700 will now respond by sending you a **200 OK** response and image data.

Scanning Using MFC

Microsoft Foundation Classes (MFC) offers good support for Internet connections, especially when it comes to HTTP and FTP. As described above, scanning is performed by requesting a **POST** using HTTP. If the operation is successful, the image is returned via a redirect response. The steps below describe how a scanning is performed using MFC. If you need more information about the classes described, see the MFC documentation. The following code is not checked for errors.

1. Create a CInternetSession object:

   ```
   CInternetSession aSession("ScanSDK");
   ```

   The parameter is a string identifying the application. An application needs only one instance of this class.

2. Using this object you can now retrieve an HTTP connection:

   ```
   CHttpConnection* anHttpConnection =
         aSession.GetHttpConnection("somescanserver",
                                    (INTERNET_PORT) 80,
                                    NULL,
                                    NULL);
   ```

3. Using this connection, you can now open a request. By default, redirect responses are handled automatically:

   ```
   CHttpFile* anHttpFile =
         anHttpConnection->OpenRequest(0,
                                       "/user/scan/scan.shtml");
   ```

   The first parameter defines that a POST request shall be opened, the second is the request target.

4. Create a query string containing the parameters you want to send:

   ```
   CString aQueryString =
         "\n\npage=scan&profileName=Color+low&scanNow=";
   ```

5. Add an HTTP header which informs the HTTP server that the request contains encoded data:

   ```
   anHttpFile->AddRequestHeaders(
         "Content-type: application/x-www-form-urlencoded");
   ```

6. Send the request:

```
anHttpFile->SendRequest(NULL,
                        0,
                        (void*)(const char*)aQueryString,
                        aQueryString.GetLength());
```

   The method will not return ????? until a response is received from the server or after a time out.

7. If the request is successful, image data will be sent back. Otherwise an HTML page containing error information will be sent. One way of testing what was sent back is to look at the Content-type. Its value can be retrieved by the following call:

```
anHttpFile->QueryInfo(HTTP_QUERY_CONTENT_TYPE,aContType);
```

   If all went well, aContType will have the value “image/jpeg” (if JPEG is the chosen format. See table above.), otherwise “text/html”.

8. Finally we are ready to receive the data:

```
UINT bytesRead = anHttpFile->Read(buffer, MAXBUF - 1);
while( nBytesRead > 0)
{
      //Save data somewhere
      .
      .
      .
      //Read next chunk
      nBytesRead = anHttpFile->Read(buffer,MAXBUF -1);
}
```

Scanning Using the WinInet API

If you do not want to use MFC to create an Internet application you can use the WinInet API instead. It provides basically the same functionality as MFC does and the functions in the WinInet API are very similar to those in MFC. In the steps below we describe how the API is used. For more information, see the WinInet documentation. The following code is not checked for errors.

1. Call InternetOpen to enable an Internet connection:

```
HINTERNET anInetHandle =
    InternetOpen("ScanSDK",
                 INTERNET_OPEN_TYPE_DIRECT,
                 NULL,
                 NULL,
                 0);
```

2. Create an HTTP handle by calling the InternetConnect function with the necessary parameters:

```
HINTERNET anHTTPHandle =
      InternetConnect(anInetHandle,
                      "171.16.1.185",
                      INTERNET_DEFAULT_HTTP_PORT,
                      "root",
                      "pass",
                      INTERNET_SERVICE_HTTP,
                      0,
                      0);
```

3. Use the HTTP handle to open a request. The second parameter indicates that it is a **POST** request and the third indicates that the target is the *'/user/scan/scan.shtml'* page. By default, redirect responses are handled automatically:

```
HINTERNET anHttpRequest =
      HttpOpenRequest(anHTTPHandle,
                      "POST",
                      "/user/scan/scan.shtml",
                      NULL,
                      NULL,
                      NULL,
                      0,
                      0);
```

4. Add an HTTP header which informs the HTTP server that the request contains encoded data:

```
char aReqStr[] =
      "Content-type: application/x-www-form-urlencoded";

HttpAddRequestHeaders(anHttpRequest,
                      aReqStr,
                      strlen(aReqStr),
                      0)
```

5. Create a query string containing the parameters you want to send and send the request:

```
char aQeryStr[] =
      "\n\npage=scan&profileName=Color+low&scanNow=";

HttpSendRequest(anHttpRequest,
                0,
                0,
                aQeryStr,
                strlen(aQeryStr))
```

6. If the request is successful, image data will be sent back. Otherwise an HTML page containing error information will be sent. One way of testing what was sent back is to look at the Content-type. Its value can be retrieved by the following call:

```
HttpQueryInfo(anHttpRequest,
              0,
              &aBuffer,
              &aBufferSize,
              HTTP_QUERY_CONTENT_TYPE);
```

   The third parameter is the address to a buffer that receives the information, and the fourth is the address to a variable containing the buffer's length. After the call it will contain the size of the information written to the buffer.

   If all went well, aContType will have the value "image/jpeg" (if JPEG was the chosen format (see table above). Otherwise the value will be "text/html".

7. The data can now be received:

```
// All data has been read when (nrBytesRead == 0)
while(InternetReadFile(anHttpRequest,
                       aBuffer,
                       aBufferSize,
                       &nrBytesRead) &&
      nrBytesRead > 0)
{
      // Save data somewhere
}
```

8. Finally, it is important to close all opened handles:

```
InternetCloseHandle(anHttpRequest);
InternetCloseHandle(anHTTPHandle);
InternetCloseHandle(anInetHandle);
```

Push Type Scanning

We refer to push type scanning as scanning initiated from the control panel of your AXIS 700. The images are *pushed out* to a directory on an FTP server.

Your application can scan the directory on the FTP server. When new images arrive, the application can, depending on information sent with the image, take the appropriate actions.

For each image file written to the directory, an information file will also be written, unless you choose *Include Image Information: None* when defining the destination. The information file is in raw text format and it contains data about the image.

Information included in the information file is

- Name and size of the image file.
- Internet address of your AXIS 700.
- Currently connected scanner.
- The name of the destination (the name appearing in the destination list).

- Scanning profile used.
- Paper size.
- Number of pages.

If you selected *Include Image Information: Complete*, then additional information, such as resolutions and file format, is also included.

Your application can use values in the information file to trigger actions.

Example of information file: The following example is taken from a real scanning, with the pre-defined scanning profile Text. Here, *Image Information Included* is set to *Complete*. If it was set to *Basic*, the Image Information part would not be present.

```
; AXIS 700 Parameter List, V1.10 Jul 29 1998

[General]
File name            = branko_subasic001.tif
File size            = 71413
Host address         = 172.19.4.22
Scanner              = FCPA ScanPartner 600C
Destination          = Branko FTP
Profile              = Text low
Paper size           = Letter
Number of pages      = 2

[Image Information]
Width                = 1696
Height               = 2200
Resolution (DPI)     = 200
Bits per pixel       = 1
Data type            = Black & white
Paper orientation    = Portrait
Format               = CCITT G.4
Double-sided         = Off
Intensity            = 50
Contrast             = 50

; End of Parameter List
```

It is important to notice that the information file is *always* written *after* the image file. Thus when your application processes the information file the image file itself is guaranteed to be there already.

Example: A document distributing system may check the file name to see whom the image is targeted for. The file name may then be on the form *name_lastNameXXX.tif*, where XXX is the sequence number optionally appended by your AXIS 700. Your application can now decide whom to send the image to.

Example: A fax server application can use the image name to obtain the fax number, and the file format to decide how to convert the document to fax data.

See the *AXIS 700 Additional Installation Note, Setting Up Your AXIS 700 for Scan-to-File* for information on how to define Scan-to-File destinations.

Accessing the List of Scanning Profiles

The profiles list is stored in the config.ini file, along with all other parameters. It can be obtained using FTP (see *AXIS 700 Additional Installation Note, Reading and Writing the config.ini file*). There is one group stating the number of available profiles, along with their names. In addition, there is one parameter group for each profile.

To obtain all available profiles it is enough to read the file from your AXIS 700 and find the group [**Profiles**]. It looks like this:

```
[Profiles]
Clear List First      = no                    ; yes/no
Profiles              = 2
Profile0              = Profile-Text
Profile1              = Profile-Color low
```

The first parameter defines whether to remove all existing profiles before storing the new ones or not. The second parameter defines the number of available profiles, which are two in this case. The following two lines contain the names of the parameter groups defining the two profiles.

This is a parameter group for a profile:

```
[Profile-Text]
Description           = Text
X-Resolution          = 300
Y-Resolution          = 300
Data type             = Black & white
Paper orientation     = Portrait
Image compression     = CCITT G.4
File format           = TIFF
Double-sided          = Off
Document mode         = Multi page
Intensity             = 50
Contrast              = 50
```

The first parameter is the name of the profile. Then follows the resolutions, the data type etc.

To add a profile you have to create a parameter group for the profile. You also have to increase the *Profiles* parameter in the *[Profiles]* parameter group and add a parameter in that group defining the name of the new profile.

To remove a profile you have to remove the parameter group. Then you have to modify the *[Profiles]* group. Decrease the *Profiles* parameter and remove the line defining the profile's name.

Finally you must write the modified config.ini file to your AXIS 700.

Note: No validation of the correctness of the defined profiles will occur when writing the modified config.ini file to your AXIS 700. Scanning using profiles with incorrect values may result in corrupt images.
