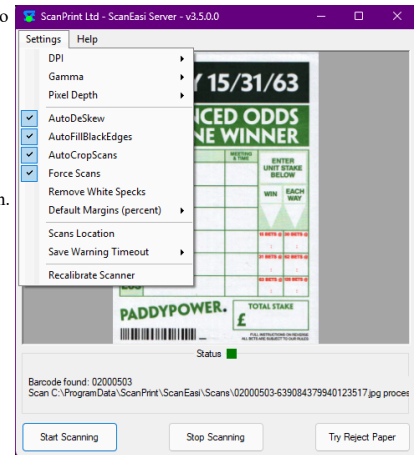


ScanPrint ScanEasi Server

The ScanEasi Server allows you to easily configure and retrieve scans for use in any application. All your application needs to do is to be able to read PNG or JPG files from a specified folder or alternatively make simple RESTful API calls.

FEATURES

- **Super fast real world scans** - From scan start to file save in less than 0.5 seconds typically (depending on the scan mode and resolution).
- **Auto saving of time stamped scans** - scans are time stamped to the nearest 100ns and saved into your specified folder location. File names are prefixed with the optional ScannerID and the scanned barcode, if one has been detected. This allows you to easily retrieve and sort scans based in barcode value and/or scannerID etc.
- **Easy configuration** from the provided GUI or a simple configuration file. I.e;
 - resolution,
 - scan colour depth,
 - gamma (overall brightness and contrast) and
 - File save location etc.
- **Push Scanning** - scans are triggered automatically as soon as paper is detected
- **Instant Status** notification via a traffic light themed ScanEasi server icon in the Windows Tasbar notification area. The Server Error colour theme is as follows:
 - Green: OK. No Errors found
 - Orange: Warning. A possible issue has been detected
 - Red: Error. An error has been detected Status messages are also shown in the Server GUI with the above colour theme.



- **Optional Background processing** via hiding of the server GUI (Critical errors are still displayed via Windows pop-up message boxes and via the ScanEasi server icon in the Windows Tasbar notification area)
- **Preprocessing options**
 - Auto De-skewing - straightening out of slanted forms
 - Auto Cropping – removal of black borders when scanning forms less than the scanner width
 - Auto-Filling of black edges - useful when a skewed form is scanned
 - Removal of White specks from Black / Dark areas of the scan
 - Barcode detection - optionally detect form barcodes in scanned images
- **Optional RESTful API server**

The RESTful API allows you to retrieve scans directly in any web app. Scans don't have to be saved locally first (however, they are saved for backup purposes). ScanEasi RESTful API Features summary;

 - Scan images can be remotely retrieved in any Web app via a simple HTTP GET 'scanImage' request
 - The configuration of the scanner can be remotely retrieved via a simple HTTP GET 'config' request
 - The resolution, Scan Colour Depth and Gama can be remotely set simple HTTP PUT requests
 - API Authentication is optionally via the use of Https, an APIkey and/or IP whitelists This means you can remotely set up and access the output from any number of scanners - all from your Web app written in Javascript or any modern language - no need for difficult C++ or C# development.
 - Current RESTful API endpoints;
 - Get BaseApiUrl:ApiPort/*scanImage/type='imageType'* ==> Retrieve the latest scanned image. Set *imageType* to 'png' or 'jpg' as appropriate. (E.g. For testing in the browser you can use a url like this - <http://localhost:3000/scanImage/type=png>). Note:
 - The custom response header "**ScanPrint-ImagesleftCount**" is the number of images scanned since this endpoint was last called. If the **ScanPrint-ImagesleftCount** value is more than 1 then this endpoint should be called repetitively until the value equals 1. This ensures that scans won't be missed. If the value is 1 or 0 then it can be assumed that the current scanImage is being retrieved.
 - The custom response header "**ScanPrint-ImageTimeStamp**" holds a 100ns resolution time stamp value for the image.
 - The custom response header "**ScanPrint-ScannerID**" holds the optional string identifier for the scanner if one has been set in the configuration file 'settings.json' (See the Configuration sections below).
 - The custom response header "**ScanPrint-BarcodeValue**" holds the value of any barcode detected in the scanned image if DoBarcodeSearch has been set to true (See the Pre-Processing Presets section below).
 - The last 3 custom header values are also used when saving the scan image to disk using the format sID-'ScanPrint-ScannerID'**BR**-'ScanPrint-BarcodeValue'**ScanPrint-ImageTimeStamp**'. (See the BASIC SETUP -> BASIC USAGE section below)
 - Note that Browsers (or other agents) may change the cases of custom headers (E.g. 'ScanPrint-ScannerID' may become 'scanprint-scannerid').
 - Get BaseApiUrl:ApiPort/*health* ==> Get the health status of the API server
 - Get BaseApiUrl:ApiPort/*version* ==> Get the server version
 - Get BaseApiUrl:ApiPort/*config* ==> get the current configuration settings
 - PUT "value" BaseApiUrl:ApiPort/*dPI* ==> Set the scanner resolution in DPI. Can be "200" or "300"
 - PUT "value" BaseApiUrl:ApiPort/*scanPixelDepth* ==> Set the scan colour depth. Can be '1', '8', '24' or '32'
 - PUT "value" BaseApiUrl:ApiPort/*gamma* ==> Sets the overall brightness and contrast. Can be 1 to 99.
 - PUT "value" BaseApiUrl:ApiPort/*autoCropScans* ==> Can be set to "true" or "false" - see below
 - PUT "value" BaseApiUrl:ApiPort/*autoDeskew* ==> Can be set to "true" or "false" - see below
 - PUT "value" BaseApiUrl:ApiPort/*autoFillBlackEdges* ==> Can be set to "true" or "false" - see below
 - PUT "value" BaseApiUrl:ApiPort/*removeWhiteSpecks* ==> Can be set to "true" or "false" - see below
 - PUT "value" BaseApiUrl:ApiPort/*forceScans* ==> Can be set to "true" or "false" - see below

BASIC SETUP

INSTALLATION

1. Install the main ScanPrint Windows drivers via the provided ScanPrint_Setup.exe
2. Uninstall any existing 'ScanEasi' application from Windows 'Add or Remove Programs'.
3. Unzip the ScanEasi-XXbit.zip into a folder of your choice
4. Run ScanEasiSetup.exe.

This will install the ScanEasi executable (ScanEasiServer.exe) by default into the '*C:\Program Files (x86)\ScanPrint\ScanEasi*' folder. ScanEasi can be run by typing 'ScanEasi' in the Windows search bar.

CALIBRATION

At initial installation re-calibration should be done via the **Settings --> Recalibrate Scanner** menu option. This quick and easy step typically only needs to be done once in the lifetime of a scanner.

Note re-calibration will also be required if a scanner is replaced or has updated hardware components. If the re-calibration procedure is not completed the scanning quality may not be optimal especially in full colour scan modes.

BASIC USAGE

1. After calibration run ScanEasi and use the GUI to set the basic desired scan settings:
 - DPI
 - Gamma (this sets contrast and brightness). Around 4 to 9 typically works well for 1 bit B/W and 7 to 9 for gray scale / 24 bit colour scans.
 - Pixel Depth
2. Turn on the desired pre-processing options. E.g.
 - De-skew
 - Auto fill Black edges
 - Auto Crop
 - Force Scan etc

Note that all settings are automatically stored in the **Settings.json* configuration file (see below)

3. Set the location where your scans will be saved (defaults to "ProgramData\ScanPrint\ScanEasi\Scans"). Scans are saved using the following file format:

*sID-ScannerID__BR-BarcodeValue__ImageTimeStamp**

where 'sID-' is the Scanner ID prefix and 'BR-' is the barcode value prefix and note the use of 2 underscores ' __ ' to separate the scannerID, barcode value and file timestamp in the file name. This avoids issues with the use of '-' in the header values themselves. and allows you to easily sort and retrieve scans based on the scanner ID and/or barcode value etc.

At it's simplest, all your app needs to do is poll the scan location for new .jpg files, retrieve and then delete them once you have safely backed them up somewhere. Alternatively, you can use the SxanEasi's optional RESTful server for more flexibility (see below)

The GUID for ScanEasiServer.exe is shown in the GUI Help menu pop-up. (*It is currently "43697451-78f3-4d8e-9302-11d9390a6529"*)

CONFIGURATION

All ScanEasi configuration settings are saved in a JSON file (settings.json - this is created automatically if it does not exist and is located in the '*C:\programData\ScanPrint\ScanEasi*' folder typically). **Some** settings can be set via the server GUI or remotely via the optional API endpoints if the optional API server is enabled. Note that settings changed in the GUI or via the web server are saved as they changed. However if you manually edit Settings.json then you will have to stop and restart ScanEasi for the changes to be applied.

Configuration settings:

- *ScannerID*: An optional string which may be used to uniquely identify the scanner
 - *DisbleSettingsMenu*: Default is *false*. if *true* no settings can be changed via the server GUI. All settings must be made via manual editing of the 'settings.json' file and restarting of the server application. If *false* a subset of the settings (see below) can be dynamically changed whilst the server is running.
-

*The settings below can only be set directly in the settings.json file irrespective of the **DisableSettingsMenu** value*

Background-Processing Presets

- *ShouldRunHidden*: Default is *false*. if *true* the UI is hidden at start up. Critical errors are still shown via pop-up Message boxes and once a critical error has occurred the user interface is enabled and shown. **Note**: If there are no critical errors, when the UI is hidden then, the server application can only be killed via the Windows Task Manager or the Windows 'taskkill' command.
- *ShowReHideWarning*: Default is *false*. If *true* show a pop-up messagebox when re-hiding the UI after a critical error. **Note**: The UI is hidden by design after a critical error when *ShouldRunHidden* is true, by just clicking on the UI minimise icon. This has no effect if *ShouldRunHidden* is false;

API server Presets

- *BaseApiUrl*: The base URL used to access the API. E.g. "localhost" or an explicit IP address "194.617.120.090" etc if this is null (the default) the API server is disabled.
- *ApiPort*: The port number used to access the API. E.g. 3000,
- *ApiPingInterval*: Set the time interval in milliseconds used internally to check that the API server is still running. If a problem is detected the server application status indicator turns orange and a status message is shown.
- *UseHttps*: Set to 'true' or 'false' as appropriate
- *APIKey*: An Optional secret key to be sent in a 'APIKey' header. If this is null or an empty string then no key will be used
- *IPWhitelist*: A list of allowed Client IPs for the API. if this is null the whitelist is not active. For the local machine any of ' '::1' , 'localhost' or 127.0.0.1 can be used
- *CORS_AllowedOrigins*: Specifies the allowed domains when the API is accessed from a web app running in a Browser from a different domain from this API server.Defaults to the 'Any domain' wildcard "*".

Pre-Processing Presets

- *CropNoiseFilterSize*: When detecting areas to crop this determines the size of the noise specks (in pixels) to ignore. The Default is 8 pixels. The filtering does not affect the saved scan images.
- *CropEdgeTolerance*: When detecting black areas to crop this sets the margins (in pixels) to ignore at the edge of the paper. The Default is 5 pixels. THIS IS CURRENTLY NOT USED.
- *MaxFillBlackEdgeWidth*: When filling in black edges this limits the maximum width (in pixels) of the fill to avoid over-filling. If set to zero or negative the value of 1 is used. The value is scaled based on DPI above 200 DPI (e.g. at 300 DPI = 1.5x this value)
- *HorizontalCropBorder*: Set the horizontal Crop Border (positive values) or horizontal Trim amount (negative values). If missing or null defaults to the negative of the Crop Noise Filter Size + 2 which gives minimal trimming to avoid over-cropping
- *VerticalCropBorder*: Set the vertical Crop Border (positive values) or vertical Trim amount (negative values). If missing or null defaults to the negative of the Crop Noise Filter Size + 2 which gives minimal trimming to avoid over-cropping
- *DoBarcodeSearch*: If true, scanned images are searched for barcodes after any cropping and de-skewing. The barcode value is then included in the saved file name (with the prefix "BR-") and in the custom response header 'ScanPrint-BarcodeValue' when retrieving scans via the API. Default is false. The detected barcode types are
 - EAN_8,
 - EAN_13,
 - UPC_E,
 - UPC_EAN_EXTENSION,
 - CODE_128,
 - CODE_39,
 - CODE_93,
 - CODABAR,
 - ITF,
 - RSS_14,
 - RSS_EXPANDED

The settings below can be dynamically set in the server GUI provided that **DisableSettingsMenu** has been set to true. The server application does not have to be restarted for changes to have an effect.

- *ScanSaveWarningTimeout*: A warning timeout period measured in milliseconds. If the scan is not saved to disk within that period the server status indicator turns orange and the total processing time is showed in the GUI. (Note this does not mean the scan was not saved. It just means it was not saved within the specified time.
- *DefaultMargins*: The left and right scan margins, as a percentage of the scanner width, to ignore in all cases. Default is 0%. This may be useful if there are issues with peristent noise at the the edges of scans. However, note that it reduces the maximum usable form width. Use with care.
- *RawScanFilesLocation*: The folder location where scans will be saved. By default this is in a 'Scans' sub-folder where this server application is run from.

These settings can be set in the server GIU AND via an API endpoint if the API server is enabled

- *Gamma*: Controls the overall brightness of the scanned image (Actually gamma optimizes the contrast and brightness in the midtones. This is particularly important for scanned documents because it can significantly improve pages readability). Values from 1 to 99 are valid. if not set a default of 4 is used.
- *DpiValue*: The resolution of the scanner. Currently either 200 or 300 DPI.
- *ScanModePixelsDepth*: Determines the number of bytes used for each pixel during the scan. Possible values
 - 1 : 1 bit black and white scans - Fastest scans
 - 8 : 8 bit grayscale scans
 - 24 : 24 bit RGB scans
 - 32 : assumes 24 bit RGB scans - Slowest scans note that the full colour scans are slower than the black and white scans
- *AutoDeskew*: An Optional pre-processing option which tries to vertically re-align any forms which have been scanned at an angle. By default this is false.
- *AutoFillBlackEdges*: An Optional pre-processing option which fills in any black edges arising from the deskeewing process. By default this is false.
- *AutoCropScans*: An Optional pre-processing option which crops black edges caused by scanning forms smaller then the width of the scanner. By default this is true.
- *ForceScans*: If true start scan as soon as any paper is detected rather than waiting for the full paper edge to be detected (I.e. rather than waiting for the form to be inserted correctly aligned straight with the scanner edge)
- *RemoveWhiteSpecks*: If true try remove white specks noise BEFORE any operations. This will cause some smoothing / blurring of the captured scan image and so should not be used unless required. Default is false.

Here is a typical Settings.json file contents

```
{
  "ShouldRunHidden": false,
  "ShowReHideWarning": true,
  "DisbleSettingsMenu": false,
  "ScannerID": "AR2R37660",
  "BaseApiUrl": "localhost",
  "ApiPort": 3000,
  "ApiPingInterval": 0,
  "UseHttps": false,
  "APIKey": "fjoaud&adsFF6!G7834ii_$",
  "IPWhitelist": [
    "localhost",
    "127.0.0.1"
  ],
  "CORS_AllowedOrigins": "",
  "CropNoiseFilterSize": 8,
  "CropEdgeTolerance": 0,
  "MaxFillBlackEdgeWidth": 50,
  "HorizontalCropBorder": null,
  "VerticalCropBorder": null,
  "ScanSaveWarningTimeout": 40,
  "DefaultMargins": 0.0,
  "RawScanFilesLocation": "C:\\scanprint\\ScanEasiServer\\Scans",
  "Gamma": 9,
  "DpiValue": 200,
  "ScanModePixelsDepth": 1,
  "AutoDeskew": false,
  "AutoFillBlackEdges": true,
  "AutoCropScans": true,
  "RemoveWhiteSpecks": false,
  "ForceScans": false
}
```

Optional API Server Set-Up

The API server is enabled if the **BaseApiUrl** value in Settings.json is not set to the value "**null**" and is a valid URL.

However, note that Windows also requires some additional steps to configure the API server: Step 1 (and 2 for https if used) below need to be done before running ScanEasiServer.exe**

Hint - For quick testing in the browser after setup you can use a url like - <http://localhost:3000/scanImage?type=png>.

1. Restful API server URL authorisation

In Windows the hosted URLs needs to be authorised. This means one of the following need to be done :

A. Always running The Web server process (ScanEasiServer.exe) with admin privileges (not advised)

OR

B. Before starting ScanEasiServer.exe using non-admin privileges, creating a **namespace reservation for the provided URL**. This can be done via a admin level command console to reserve the URL

For **HTTP** usage (note the '+' wildcard format)

```
C:> netsh http add urlacl url=http://+:3000/ user="Everyone"
```

OR for secure **HTTPS**

```
C:> netsh http add urlacl url=https://localhost:3500/ user="Everyone"
```

Note that for https the wildcard pattern (E.g. url=https://+:3500/) does not appear to work

To see existing URL namespace reservations

```
C:> netsh http show urlacl
```

Existing reservations may clash (especially for https). To delete before trying again use something like

```
C:> netsh http delete urlacl url=http://localhost:3000/
```

2. HTTPS configuration

A. For HTTPS an SSL certificate is needed.

If OpenSSL is installed you can generate a self signed certificate via openssl :

```
C: > openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
```

and to get the thumbprint hash needed later

```
C:> openssl x509 -in certificate.pem -noout -thumbprint
```

OR via Powershell :

```
PS $cert = New-SelfSignedCertificate -DnsName "localhost" -CertStoreLocation "C:\Users\Yourname\" -FriendlyName "LocalhostCertificate"
```

to get the thumbprint (displayed in the console where you can cut and paste)

```
PS $cert.Thumbprint
```

and to save the certificate (needed later for adding to the Windows Trust Store)

```
PS Export-Certificate -Cert $cert -FilePath "C:\Users\YourName\sslTemp\certificate.cer"
```

B. Self-certified certs need to be trusted.

This means adding it to the Windows "Trusted Root Certification Authorities" store. This can be done by opening the .cer file and clicking on 'install Certificate'

C. For HTTPS the SSL certificate needs to bind to the web server assigned port

This can be done via the console (or Powershell) command

```
C:> netsh http add sslcert ipport=0.0.0.0:YOUR_PORT certhash=YOUR_CERT_THUMBPRINT appid={YOUR_APP_GUID}
```

E.g.

```
C:> netsh http add sslcert ipport=0.0.0.0:3000 certhash=eb8e73617b2e35fb5a1ad876d60eca144a2ce2bc appid="{43697451-78f3-4d8e-9302-11d9390a6529}"
```