opentext™

Administration Guide

OpenText™ Blazon™ Enterprise

Install and configure the Blazon Enterprise components, including the Blazon Redaction module.

CLBRVW160612-ABZ-EN-02

Administration Guide
OpenText™ Blazon™ Enterprise
CLBRVW160612-ABZ-EN-02
Rev.: 2025-May-21

This documentation has been created for OpenText™ Blazon™ Enterprise 16.6.12.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Blazon Enterprise Administration Guide

This guide provides an overview of the OpenText™ Blazon™ Enterprise product, information about installation and configuration, as well as troubleshooting tips to assist administrators, integrators, and other IT professionals with advanced configuration Blazon Enterprise features. A certain amount of advanced technical knowledge is needed to effectively use this information guide.

For the latest documentation updates be sure to visit our product page on OpenText My Support (https://support.opentext.com).

1.1 Quick table references

Chapter 2

Planning considerations

Blazon Enterprise is a web-based component that allows integrators to publish files and monitor directory structures for publishing in an automated way.

Integrators interact with Blazon Enterprise using URL and they are notified of success using HTTP.

Publishing files: Files can be published as XDL (a multi-file, proprietary format), CSF (a single file, secure proprietary format), PDF, or TIFF file types.

Monitoring directories: Blazon Enterprise can be configured to monitor directory structures and publish all files as they are created or changed.

Redact. The optional Blazon Redaction module can be used with our markup technology to create renditions of those files that have markups burned in. Markup technology provides the ability to process files and redact sensitive strings to create redacted output as CSF, TIFF, or PDF file types.

2.1 Component overview

Blazon Enterprise consists of two major components; the Queue Server and the Job Processor. The Queue Server holds jobs for publishing and serves those jobs to awaiting Job Processors. The Job Processor does the actual work of publication, and is responsible for calling a notification URL upon job completion. A typical production installation consists of a single Queue Server installation on a server and one or more Job Processor installations, each on their own dedicated server.

2.1.1 The Blazon Enterprise Queue Server

The Queue Server consists of two parts: a set of IIS ASPX pages (ASP.NET) and a Win32 Service (written in .NET 4.6.2). This guide will refer to the IIS Pages as the “Queue Server”, and the Win32 Service as the “Queue Service”. Both parts are installed with the Queue Server component.

2.1.2 The Job Processor

The Job Processor connects to a Queue Server using a port (default, 8890) that is opened by the Queue Service. It requests jobs according to the configuration specified in `jobprocessor.config`. After publishing, it calls the Notification URL (specified in the job) to indicate that the publishing job is complete.

2.2 System requirements

Installing Blazon Enterprise software on certified operating systems allows installers to optimize performance.

IIS requirement: The following ASP.NET Role Services must be enabled for the IIS Version that is the default for the Operating System. ASP.NET roles can be found under the **Web server > Application Development** category for roles.

Refer to the release notes document (Blazon Enterprise Release Notes version 16.6.12.pdf) for detailed and the most up to date information on supported platforms, operating systems, and versions. https://support.opentext.com

2.2.1 Job Processor requirements

The Job Processor installation includes installation of the **CSF Writer**, which is necessary for converting some file types.

Microsoft .NET Framework 4.6.2 is a requirement of the Blazon Job Processor.

Note: .NET Framework is not bundled in the Blazon Enterprise installer. If not found, you must install it before installing the Blazon Enterprise Job Processor.

Service Account Requirements – The service account used to run the Job Processor must have the following:

- **Folder/Network permissions:** Read and access to the source and target destinations for jobs (this is the displaylistcache in Blazon), as well as permission to call the notification URL.
- **Privileged Executable Launching permissions:** The Job Processor service launches instances of separate executables to handle multiple threads of execution. These child processes must inherit access to the printing subsystem of Windows to successfully launch and interact.
- **Printer Configuration:**
  - CSF Writer must be installed and configured.
  - Administrative access to the CSF Writer printer driver is required.
  - The CSF Writer requires that Print Spooler services be set to **Automatic** or **Started**. The installation cannot start the Print Spooler service, required for CSF Writer, if the service is **Disabled**.

  - Permissions to manage the default printer.

- **Microsoft Office:** Access to the installed and activated version of Microsoft Office. To render Office formats (DOC, DOCX, XLS, XLSX, PPT, and PPTX), Office is required to be installed and licensed on the Job Processor machines, and applications used for publishing initialized - before running the Job Processor installation. We offer alternative technology that does not require Office as an additional option. Contact your Account Executive for additional information.

  Note: See *OpenText Brava! - CSF Writer Publishing Guide (CLBRVW-UCW)* for additional Office setup information.

- **Performance Counter permissions:** The Job Processor accesses Windows performance counters to track job status. The service account might need to be added as a member of the **Performance Monitor Users** group if the service account does not have full administrator privileges.

Notes

- The exact set of required permissions can vary, depending on which features are configured, which version of Windows is used, and which version of Office is used. Newer versions tend to be more restrictive and updates to Windows, Windows components, or Office can sometimes change the permissions required.
- We strongly recommend running the Job Processor with full administration privileges if possible, which is the scenario that is used for product testing.
- Microsoft Office requirements are important to avoid failure when publishing using either Office Automation or CSF Writer publishing. Office Automation processes Office documents through the automation API that is provided as part of Office to either print to the CSF Writer printer driver or to save the Office files to XPS format, and then converts the result. CSF Writer publishing processes documents through an external application that prints them to the CSF Writer printer driver, and then converts the captured data from that print operation.
- The requirements listed under printer configuration are important to avoid problems when using CSF Writer publishing.
- CSF Writer publishing sometimes causes third-party applications to display confirmation messages. The CSF Writer sub-system attempts to automatically handle the common messages, but does not suppress User Account Control (UAC) messages. Disabling of UAC is recommended for the Job Processor account to avoid job processing interruptions.

2.3 Scaling options

Blazon Enterprise can be scaled to handle additional publishing loads in two ways. Each Job Processor can be configured to take advantage of more powerful hardware and multiple Job Processors can be installed to handle additional load using more hardware.

2.3.1 Simultaneous publishing

Each Job Processor can be configured to perform a certain number of simultaneous conversions for each Queue for which it is responsible. The Job Processor can be configured to take advantage of additional CPU speed, cores and memory. The Job Processor scales almost linearly with number of CPUs/Cores. Additional memory or faster CPU will increase publish speed. Generally speaking, the number of simultaneous conversions should be set to no more than twice the number of processors (or cores) on the Job Processor machine. You can have more simultaneous processing than number of cores so that the conversion process doesn't become I/O bound and cause unused cycles.

2.3.2 Adding more Job Processors

Adding more Job Processors (on dedicated hardware) will increase total throughput somewhat linearly, until network saturation is reached between the Job Processors and the source and target destinations. To handle additional load, more Job Processors can be installed to talk to a single Queue Server. Each Job Processor can be configured to handle the same types of conversion jobs. Multiple Job Processors can each be configured to handle different kinds of publishing jobs. Up to 10 additional Job Processors can be added to a single Queue Server.

See "Setting up multiple Job Processors for a single Queue Server" on page 132.

2.3.3 Fault tolerance

The Job Processor is designed to perform each job in isolation from all other jobs. Thus, if the job has an unrecoverable error (due to malformed input for instance), then that error does not interrupt processing of other jobs. The Job processor reports the error to the integrator on occurrence.

Installation of multiple Job Processors on multiple machines provides additional fault tolerance in the case of network failures or hardware failure.

2.3.4 Bandwidth

If the files to be published are not on the Job Processor machine, then the loader reads the entire file once across the network. All published output is written locally and then copied to the destination specified in the published job. Therefore, bandwidth is directly proportionate to output file size.

The Job Processor will notify a URL, as directed, when publication is complete. These notifications are HTTP calls that are typically less than 4 kb.

2.4 Supported formats

To view the current list of supported input formats, refer to the Blazon Enterprise formats document available from:

https://www.opentext.com/file_source/OpenText/en_US/PDF/opentext-article-blazon-formats-en.pdf

Blazon Enterprise will read a supported input file type and publish it to an output format valid for the input you specified.

Blazon Enterprise supports conversion of a large number of file types. It can use a file's default application print output to convert virtually any file type (meaning that the Job Processor machine must have a `Print to` command available in the native application for that extension) . To enable conversion of additional formats not on this list, you must install the format's corresponding application on the Job Processor machine. For example, Office formats require that Microsoft Office be installed. In addition, the file's extension should be added to the `publish.request.extensions.doc, pdf, drw` property in the `server.properties` file located in the Blazon Enterprise base installation folder (for example, `C:\Program Files (x86)\OpenText\Blazon Enterprise`).

If a new extension is added to a `publish.request.extension.<format>` list, stop and restart the Blazon Job Processor Service and restart IIS to acknowledge the new entry.

2.4.1 Input file types (pre-published formats)

- Drawing File Extensions:

```
publish.request.extensions.drw=000,906,907,afp,ans,arx,asm,bmp,
cal,ccz,cg4,cgm,cgmct,cgmt,cit,cmi,csf,mi,dc,dgn,dgn7,dls,dsf,
dsn,dwf,dwfx,dwg,dxf,edc,emf,eps,fax,g3,g4,gif,gp4,grp,hdp,hgl,hpgl,
iam,ica,icd,icf,iges,igs,img,ipt,iso,jp2,jpg,jpeg,jpm,m3r,mcs,
mil,mrk,mvp,mvs,par,pcd,pcx,plt,png,prt,ps,psd,pub,ref,res,rle,
rnd,rnl,rtl,sid,slddrw,tg4,tif,tiff,txt,wdp,wmf,xdl,xml,xps,zip
```

- Document File Extensions:

```
publish.request.extensions.doc=doc,dochtml,docx,
eml,htm,html,mht,mhtml,msg,pps,ppsx,ppt,pptx,rtf,vdx,vsd,vsdx,vsx,xls,xlsx,
xlsm,xltx,xlw
```

```
publish.request.extensions.pdf=pdf,key,numbers,pages
```

2.4.2 Output file types (published formats)

CSF
XDL
PDF
TIFF

Chapter 3

Security considerations

3.1 Firewall, port, and endpoints

3.1.1 Firewalls and Blazon Enterprise

The Queue Server and the Job Processor components of Blazon Enterprise communicate through the HTTP and TCP protocols. Each component has specific requirements for access to the other component. Generally, certain configurable ports must be manually opened on any firewall running on the Queue Server or Job Processor machines.

3.1.2 Queue Server

The Queue Server accepts incoming API interaction through the IIS web application on port 8090 (by default). This port is controlled by the IIS settings for the Queue Server IIS virtual directory. This port is the primary point of access for integrations.

The Queue Server accepts inbound connections through TCP on port 8890 (by default) from the Job Processors. This inbound port is configured through the *pop.processor.port* parameter in the `server.properties` file. The configured port must match the corresponding port on each Job Processor. The Job Processor and the Queue Server employ handshaking on communication through this port to ensure that the contents are valid.

The Queue Server accepts inbound connections through HTTP on port 9900 (by default) from the Job Processor upon the completion of Enhanced Directory Monitoring jobs, if this feature is enabled. This inbound port is configured through the *enhanced.directory.monitor.listener.url* parameter in the `Server.properties` file. See "Enhanced Directory Monitoring" on page 38 for more detail.

The Queue Server makes outbound HTTP connections to each Job Processor on port 7070 (by default) to monitor its status. This outbound port is configured through the *jobprocessor.monitor.port* in the `server.properties` file. The endpoint queried is `http://jobprocessor:port/events`.

Queue Server accessible endpoints

These endpoints are all serviced by the IIS web application. The configured port defaults to 8090.

```
http://queueserver:port/QueueServer/Push.aspx
http://queueserver:port/QueueServer/IsEnqueued.aspx
http://queueserver:port/QueueServer/Cancel.aspx
http://queueserver:port/QueueServer/EDMSettings.aspx
http://queueserver:port/QueueServer/EDMCreateProject.aspx
```

```
http://queueserver:port/QueueServer/EDMCreateTask.aspx
http://queueserver:port/QueueServer/Status.aspx
```

Which uses

```
http://queueserver:port/QueueServer/CurrentlyPublishing.aspx
    http://queueserver:port/QueueServer/DirectoryMonitorSettings.aspx
```

Because these endpoints allow the viewing of system information and the changing of settings, it is recommended that access be restricted in accordance with your security policies and usage needs. Basic common usage patterns include authentication for access, and restrictions based on IP, but these are not the only options. Your security team should advise you on the specific usage pattern to use.

Authentication

IIS allows restricting access to a web application based on authentication. By restricting access to the full web application and then allowing anonymous access to only `Push.aspx`, you can allow modifications to be made only by authenticated users, but allow jobs to be submitted from an integration located anywhere on the network. With more effort, you can setup your integration to provide credentials when accessing `Push.aspx`, and require authentication on all access.

For more information about enabling basic authentication in IIS, reference this Microsoft documentation:

https://docs.microsoft.com/en-us/iis/configuration/system.webServer/security/authentication/basicAuthentication

IP restrictions

If your integration will run on the same server as Blazon Enterprise, you can configure IIS with a *Deny* rule for all IPs to access the Queue Server web application, and a higher priority *Allow* rule for the single IP that should have access. This prevents other computers in the network from submitting jobs, and allows the integration computer full access. This configuration works well for EDM, where only the local services need access to the web application.

For more information about configuring IP security in IIS, reference this Microsoft documentation:

https://docs.microsoft.com/en-us/iis/configuration/system.webserver/security/ipsecurity/

3.1.3 Job Processor

Each Job Processor contains a light-weight web server that is used to report configuration, status and logs. This runs on HTTP port 7070 (by default). This inbound port is configured through the *DataPort* parameter in the `JobProcessor.config` file. The entire web server can be disabled by setting `dataport=-1`. Disabling this will prevent the Queue Server from being able to display Job Processor status on the Queue Server status page, but will not otherwise impact job handling.

Each Job Processor makes outbound connections through TCP on port 8890 (by default) to request jobs from the Queue Server. This outbound port is configured through the *queue.server.pop.port.0* parameter in the `JobProcessor.config` file. The configured port must match the corresponding port on the Queue Server. The Job Processor and the Queue Server employ handshaking on communication through this port to ensure that the contents are valid.

Each Job Processor makes outbound connections through HTTP to notify job submitters of the results of a submitted job. The address and port used for this connection is sent for each job using the *NotificationUrl* job parameter as described later in Receiving Results using a Notifier.

Job Processor accessible endpoints

These are all serviced by the built-in light-weight web server. The configured port defaults to 7070. Because the information reported can be sensitive in some situations (such as with filenames of processed files, server configuration information), it can be desirable to setup the Job Processor machines on a network isolated from general users, or to disable the web server as previously described.

```
http://jobprocessor:port/status (Aliased by /events and /stats)
http://jobprocessor:port/config
http://jobprocessor:port/log
http://jobprocessor:port/servicelog
http://jobprocessor:port/legal
```

3.2 Job Processor Security

The following security measures must be configured on the Job Processor machines in the described circumstances.

3.2.1 Administration privileges

The as-installed Job Processor service must be configured to run using an account that belongs to the local machine's Administrator user group. This is because the Job Processor service requires elevated privileges to provide CSF Writer publishing features, which generally provide the best available fidelity to original source documents.

If your organization prefers not to run services using accounts with elevated system privileges, and can accept a reduced level of fidelity in published documents, it is possible to configure the Job Processor to run using a service executable that does not require Administrator privileges.

To run the Job Processor service using an account without local system Administrator privileges:

1. Stop the Job Processor service.
2. Browse to the installed directory for the Job Processor, and locate the `jpservice.exe`, `jpservice-nonadmin.exe`, and `jpservice-admin.exe` files.
3. Delete `jpservice.exe`.
4. Copy and paste `jpservice-nonadmin.exe` into the same directory.
5. Rename the copied file from `jpservice-nonadmin - Copy.exe` to `jpservice.exe`.
6. Change the account used to run the Job Processor service to the desired non-Administrator account.
7. Ensure that all file shares that the Job Processor will access have read and write access for the new non-Administrator account.
8. Run a **Command Prompt** as an administrator and execute the following commands, changing to the actual Windows user account you are using (for example, `.\MyLocalNonAdmSvcAcct`):

```
netsh http add urlacl url= http://*:7070/ user=<domain|local\user>

netsh http add urlacl url= http://*:8090/ user=<domain|local\user>

netsh http add urlacl url= http://*:8890/ user=<domain|local\user>

netsh http add urlacl url= http://*:9900/ user=<domain|local\user>
```

   Note: If Blazon Server is installed on a separate machine from one or more Job Processors, each machine requires these commands to be executed for the user account specified.

9. Browse to the `Job Processor\Igc.Loaders` folder and run the `loaders.configuration.exe` utility.
10. Expand the **OutsideIn2dl** loader and expand the **Associated Extensions**. For each extension that is a Microsoft Office format extension, click and change the loader to **OTF2DL**.

11. When complete, click **Done**, then click **Yes** to close the dialog and save all changes.

12. Start the Job Processor service.

3.2.2 Cr2dl Chrome security installation

Cr2dl (Chrome to DL) is the loader that is used for viewing local HTML files. It converts HTML files to PDF using the Chrome rendering engine, then loads the PDF using the Pdf2dl loader. Because arbitrary HTML loaded into Chrome has the potential to access data that it should not, Chrome must be secured prior to use by Cr2dl. The file restrictions required by Cr2dl are to apply a block list to Chrome that prevents loading any links from file, http, or https protocols. (See https://chromeenterprise.google/policies/#URLBlocklist.) As a result, the HTML files loaded by Cr2dl are passed directly into Chrome without loading from a link. This allows local HTML files to load, while preventing those files from accessing or referencing any other files. Note that many HTML files are designed to reference other files and can have an unexpected appearance under this restriction.

If using Cr2dl for processing HTML files, install the latest version of Chrome on your Job Processor(s) and follow the appropriate procedure below to apply the required block list.

To apply the block list on Windows – Method 1

1. Sign in with the account that the Job Processor will be running under.

2. Edit the Windows Registry:

   a. Navigate to the following key, creating any keys in the path that are missing: `HKEY_CURRENT_USER\SOFTWARE\Policies\Google\Chrome\URLBlocklist`

   b. Add a string value of “1” with the data of “`file://*`”

   c. Add a string value of “2” with the data of “`http://*`”

   d. Add a string value of “3” with the data of “`https://*`”

To apply the block list on Windows – Method 2

1. Sign in with the account that the Job Processor will be running under.

2. Using a text editor, put the following text into a new file. Save it as a registry file type with the name `cr2dl_install.reg` to a temporary directory.

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\SOFTWARE\Policies\Google\Chrome\URLBlocklist]
"1"="file://*"
"2"="http://*"
"3"="https://*"
```

3. Right-click the new `cr2dl_install.reg` file and click **Merge** to install its contents into your Windows Registry.

To apply the block list on Linux

1. Sign in with a root account.
2. Navigate to the following directory, creating any directories in the path that are missing: `/etc/opt/chrome/policies/managed`
3. Create a new text file named `URLBlocklist`.
4. Put the following text into the new file:

```
{
    "URLBlocklist": ["file://*",
     "http://*",
     "https://*"]
}
```

5. Save the file to the `managed` directory.

Configure Cr2dl:

1. In the Blazon default installation, Cr2dl is not the default loader for publishing HTM and HTML files and you must use the "Loader configuration" on page 131 to configure this option. See *OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)*.
2. Run the `loaders.configuration.exe` file from `<install dir>\Job Processor\Igc.Loaders\` to launch the Loader Configuration tool.
3. Expand the **Cr2dl > Parameters** section and set *`CromeLocation`* to the complete path to the local Chrome executable (`Chrome.exe` or `google-chrome`). Typically, this would be `C:\Program Files (x86)\Google\Chrome\Application` on Windows, or `/usr/bin` on Linux. Do not include the filename itself.
4. In the Cr2dl **Associated Extensions** section, add or change the htm and html extensions to use Cr2dl.

Note: If Cr2dl is configured for htm and html files and the Chrome security instructions are skipped, the default configuration of the loader will post a failure message as "Chrome not configured with secure file restrictions enabled. Please enable block-list security."

Chapter 4

Installation

This section covers the basic installation steps to install the Queue Server, CSF Writer, and Job Processor. Not all installations require all components, and exceptions are detailed here.

The installation steps are outlined in an InstallShield Wizard for convenience and ease of installation. However, a complete understanding of the installation choices affords optimal setup.

4.1 Installing the Enterprise Queue Server

Run this installation on one Queue Server machine. Double-click the self-extracting zip file that you downloaded.

Run the Blazon Enterprise Installation

1. When the InstallShield Wizard begins, on the **Welcome** page, click **Next**.
2. Read the **License Agreement** and select **I accept** if you have read, understand, and agree to the terms of the Licensing Agreement. Click **Next** to continue.

   Click **Cancel** at any point during the installation to stop the installation.

   Click **Back** at any point to return to the previous window.
3. In the **Choose Destination Location** page, click **Browse** to open Windows Explorer and navigate to a desired location for the Blazon Enterprise files, or click **Next** to accept the default location and continue.
4. In the **Custom Setup** page, select the components you want to install. The Queue Server needs to be installed on only one machine. You must run the installer again on each Job Processor machine to install the **JobProcessor** components. To install only the Queue Server, select **This feature will not be available** from the **JobProcessor** component list.
5. Services are installed by default. If you don't want to install services at this time, select **This feature will not be available** from the **Install Service** list.

   Note: You must install services before you can use Blazon Enterprise.
6. If you want the install to setup and configure the Enhanced Directory Monitoring feature, select **This feature and all subfeatures will be installed to run from the network** from the **QueueServer > Enhanced Directory Monitoring** install list. The installer creates a default Enhanced Directory Monitoring setup on the Queue Server which can be used immediately. If you select not to install, you can manually configure this feature at any time after installation completes. Click **Next**.

7. If you have selected to install this feature, the **Enhanced Directory Monitoring Share** page displays. Enter the UNC path to a unique shared directory that you want to use for the Enhanced Directory Monitoring feature. Click**Next**.

   Important
   Projects names (the final directory in the full path) MUST BE UNIQUE across the entire system.

8. In the **Website information** page, if you don't want to accept the **Default Web Site**, **Virtual Directory name**, and **Port**, enter your own unique name and port of the web site that you want Blazon Enterprise to use and click **Next**.

   Caution
   Use of a restricted name, such as “DefaultWebSite” or the computer's hostname, will result in a “-1603” fatal error and the installation will fail.

9. If you have elected to install services, the next page will prompt you for your Windows services sign on and password credentials. Enter valid domain and user sign on for the Queue Service, enter the username’s password in the **Password** field. Services will be installed and startup automatically.

10. Click **Install** on the **Ready to Install the Program** page and the program files are installed.

Queue Service account information

The following account information is needed for the Queue Service installation:

The Queue Service should be installed to run as an administrator account. Security settings in the `web.config` and `QueueService.exe.config` are pre-configured to allow the Queue Service and the ASP pages to communicate properly. You should select or create a domain user for your Queue Service sign on.

Directories

By default, the installation creates the following directories:

- `C:\Program Files\OpenText\Blazon Enterprise`, which contains the common configuration file, `Server.Properties`, shared by the Queue Server and the Queue Service.
- `C:\Program Files\OpenText\Blazon Enterprise\IgcWebAppRoot\QueueServer`, which contains the ASP.NET based Queue Server.
- `C:\Program Files\OpenText\Blazon Enterprise\QueueService`, which contains the Queue Service.

4.2 Installing the Job Processor

It is strongly recommended that the Queue Server and Job Processors be installed on separate servers. Each Job Processor machine must have all of the applications it will use for publishing specific formats (such as MS Office) installed and initialized before running the Job Processor installation.

Before running the Blazon Enterprise Installation:

1. In Control Panel, click **Services**.
2. Find the **Print Spooler** service.
3. If not already started, ensure that the **Startup type** for the service is set to **Automatic** and the service status is **Started**.

Run the Blazon Enterprise Installation:

1. Run the Installation on one or more separate Job Processor machines. When the InstallShield Wizard begins, on the **Welcome** page, click **Next**.
2. Read the **License Agreement** and select **I accept** if you have read, understand, and agree to the terms of the Licensing Agreement. Click **Next** to continue.
3. In the **Choose Destination Location** page, click **Browse** to open Windows Explorer and navigate to a desired location for the Blazon Enterprise files or click **Next** to accept the default location and continue.
4. From the **Custom Setup** page, select to install the **JobProcessor** and all subcomponents:

   CSF Writer notes:

   - By default, the CSF Writer installs as the user running the install. If you are running the Job Processor under a different account (the same domain user as specified earlier), you might have to manually set the CSF Writer properties for CSF Writer publishing to work correctly.
   - Before installing CSF Writer, any previous version of "IGC Writer" (if one exists) should first be uninstalled and the machine rebooted.
   - The print spooler service needs to be enabled and running in order to install the printer driver. If the print spooler service is disabled, the installer cannot start it.
5. In the **Enter QueueServer Host Information** page, enter the host name and port of the machine that the Enterprise QueueServer is or will be installed to.
6. If you have elected to install services at this time to run the Job Processor as a service, the next page will prompt you for your Windows services sign on and password credentials that will be used to start the service. Enter valid domain and user sign on for the Blazon Job Processor service, enter the username's Password in the **Password** field. Services will be installed and startup automatically.

- If you install the Job Processor as a service (called **BlazonEnterprise**) YOU MUST ENTER A USER ACCOUNT THAT HAS ADMINISTRATOR RIGHTS ON YOUR MACHINE in order to avoid permissions errors after the installation. The Blazon Enterprise service cannot run correctly as the Local System Account because the service must interact with other processes and components. Attempting to run the Blazon Enterprise service as the Local System Account will cause unexpected and undesirable behaviors with Blazon Enterprise.
- Both the Blazon Job Processor Service and the Job Processor Console are installed. After installation you are given the opportunity to start the service. As long as the service is not running, the console can run. The Job Processor console must be started manually through your Windows Start menu (**>Start Blazon Job Processor**) and be running on the desktop to allow publishing. Be aware that if you exit out of the Command window where the console was started, the console will stop.

7. Click **Install** on the **Ready to Install the Program** page and the program files and prerequisite installations are installed.
8. **Prerequisite Installations:** Microsoft .NET Framework 4.6.2 is a requirement of the Blazon Enterprise Server and Job Processor.

   Note: .NET Framework is not bundled in the Blazon Enterprise installer. If not found, you are required to install the .NET Framework before installing Blazon Enterprise.

9. On the **Installation Complete** page, click **Finish** to complete the Blazon Enterprise installation.

Job Processor account information

The following account information is needed for the Job Processor installation:

You should select or create a domain user for your Job Processor logon. This account must have access to the source and destination directories that are referenced in the jobs submitted for publishing.

Directory

By default, the installation creates the following directory:

`C:\Program Files\OpenText\Blazon Enterprise\JobProcessor\`, which contains the Job Processor and associated support files.

4.3 Evaluation installations

You must install one Queue Server and at least one Job Processor. When installing the Job Processor, the CSF Writer Printer Driver is also installed to support print-capture based publishing. The simplest installation for evaluation purposes is to install the Queue Server and Job processor on the same machine at the same time.

4.4 Installation verification

After completing the initial installation, you can verify the installation by locating a new set of menu items under the Windows **Start** menu. Launch **Blazon Enterprise > Blazon Enterprise Status** to open the Blazon Enterprise start page in your default web browser.

Note: If your default browser is the Windows UI version of Internet Explorer, you might need to switch to desktop Internet Explorer, Firefox, or Chrome to open the status page.

4.4.1 Job Processor list

Following the Queue Statistic block of information, a list of all of your installed Job Processors should display. If you don't see an installed Job Processor in this list, then it is not communicating with the Queue Service.

When the status of each Job Processor displays as **UP** on the Blazon Enterprise **Status** page, the system is ready for use.

Notes

- There might be a 1 or 2 minute delay before the status of the Job Processor shows **UP**. Click **Refresh** or **F5** as needed.
- If the Job Processor is installed on a server other than Queue Server, the Queue Service might require a restart before the new Job Processor appears on the list.

Clicking on the link to Job Processor provides additional detail. Alternatively, the following sample URL will provide the same detail of the Job Processor:

```
http://yourJobProcessorServer:7070/status
```

Follow these Configuration check steps if you are still unable to determine the status:

Configuration check

1. Verify that the Job Processor is running.
2. Check the configuration in the `JobProcessor.config` file.

3. Verify that the value for *queue.server.address* points to the Queue Server machine and that the port specified by *queue.server.pop.port* is accessible across your network.

4.5 Licensing

By default, the Blazon Enterprise product installs a 30-day evaluation key (or publishing of 1000 documents, whichever occurs first). You will receive a license file called `IGCKey.lic` when you license your installation. When you obtain this permanent license key, you must copy the file to the following installation directories to replace the 30 day evaluation license.

```
C:\Program Files\OpenText\Blazon Enterprise\JobProcessor\
```

Note: The default location might be different in your environment if the installation location is changed.

A restart of the Job Processor is not necessary after updating the license file.

4.5.1 OCR licensing

The OCR (Image Recognition Processing (or Intelligent Character Recognition) of image data in source documents) feature of Blazon Enterprise requires an additional Add-on purchase and will only be available when licensed.

See "Image text recognition processing" on page 54 for information about this feature.

4.5.2 Blazon Redaction module

The Blazon Redaction module requires an additional Add-on purchase and will only be available when licensed. This optional module allows integrators to securely redact information from files in an automated way.

Redact The presence of this image in this document identifies features that are only available when the Blazon Redaction module is enabled.

Chapter 5

Blazon Enterprise Usage

Information presented in this section will help you properly set up and use the Blazon Enterprise product to publish files as intended, to monitor directories as desired, and to more efficiently address security needs using search strings and various scripts. You will be able to identify efficiencies and manage your jobs successfully using this information.

5.1 Publishing a file

To publish files with Blazon Enterprise , jobs are submitted to the Queue Server. Jobs are the basic unit of work for Blazon Enterprise and each job represents a request to publish a file, to monitor a directory for files to publish.

For jobs that publish a file, the job must contain the following information:

Source: The full path to the source file.

For multi-doc publishing, you can set multiple source files as *sourceN* to publish as one export file.

Target: The target directory where the publishing rendition of the file will be placed after publishing is complete.

Note: Only network UNC paths are supported for *target* and *source* parameter path values. Local paths (but not mapped drives) can be used for single box testing only.

OutputFormat: The type of output you want. This format (value) can be one of `XDL`, `CSF`, `PDF`, `TIFF`, `TXT`, or `None` if the desired outcome is only for Thumbnails or artifacts.

NotificationUrl: A URL (in the format `http://Server:port`) that will be called upon completion of the publishing job. It will contain details of any errors that might have occurred, along with all of the original parameters sent with the job.

In addition, there are many optional job options which can be added to the basic job and are detailed in the Job Options section.

See "Enhanced Directory Monitoring" on page 38 for information on how end users can easily configure and publish jobs through the Windows file system without any programming knowledge.

5.1.1 Sample job

To submit a job, call the `push.aspx` page on the Queue Server, and pass the parameters as Query String Arguments

Given a sample job with the following parameters:

```
source=\\server\sourcefiles\netit\906\AIRPLANE.906
target=\\server\outputfiles\destinationdirectory
outputformat=pdf
notificationurl=http://localhost:9999
```

The example will appear when pushed as:

```
http://servername/queueserver/push.aspx?Source=\\server\sourcefiles\netit\906\AIRPLANE.
906&target=\\server\outputfiles
\destinationdirectory&outputformat=pdf&notificationurl=http://localhost:9999
```

Upon completion of this job, the output published by Blazon Enterprise will be a file named `airplane.pdf`, located in the directory `\\server\outputfiles\destinationdirectory`

The notification URL (`http://localhost:9999`) will be called with the following items on the QueryString:

```
source=\\server\sourcefiles\netit\906\AIRPLANE.906
target=\\server\outputfiles\destinationdirectory
outputformat=pdf
notificationurl=http://localhost:9999
threadid=drw5
type=drw
jobid=1278684777549834871
filename=AIRPLANE.906
starttime=3/14/2006 3:09:35 PM
tempdir=<InstallPath>\JobProcessor\bin\release\tempdir\1278684777556084752
ext=906
targetfile= <InstallPath>\JobProcessor\bin\release\tempdir
\1278684777556084752\AIRPLANE.pdf
endtime=3/14/2006 3:09:37 PM
totaltime=00:00:02.3264386
mainfile=AIRPLANE.pdf
```

This query contains information about how long the job took (the *`TotalTime`* parameter), and some information about the temporary files used by Job Processor. The presence of the *`MainFile`* parameter is the key indication of success, and is found in the directory pointed to by the *`target`* parameter. If errors or warnings are generated, then they will be described in the additional error details parameters (*`error 0`*, *`error 1`*, *`warning 0`*, *`warning1`*, and so on), if available.

5.1.2 Canceling a job on the Queue Server

If you want to submit a job that you want to cancel, submit a job to the queue server with a *requestid* that is set to a unique string. You will need to keep track of these unique string values in order to cancel the job successfully. For example:

```
requestid=<unique string>
```

requestid is not required, however, the value must be unique among all the jobs in the queue. The same request ID holds unique ID's if set by the (deprecated with v. 7.3) *cancelid* job parameter. If a job is submitted with a duplicate *requestid* value, the queue server will return the error: "Unable to submit job: Failed to queue publishing request. Attempt to submit two jobs with the same requestid"

In order to cancel a job, use the `cancel.aspx` entry point. Call with the *requestid* parameter set to the value established when the job was enqueued. For example;

```
http://server/queueserver/cancel.aspx?requestid=<unique string>
```

`Cancel.aspx` returns "HTTP OK" if the job is successfully cancelled or it returns an error message if the *requestid* does not exist. When a job is cancelled, it is completely removed from the queue server. A job can only be cancelled while it is in the queue. A job that is in process cannot be cancelled.

Note: `cancelid=<unique string>` is deprecated as of v.7.3 and replaced with the new property `requestid=<unique string>`. If the deprecated parameter *cancelid* is used, `Cancel.aspx` returns "HTTP OK" if the job is successfully cancelled, followed by a deprecation warning. An error message is returned if the *requestid* does not exist in the queue, followed by the deprecation warning.

5.1.3 Checking a job presence in the Queue Server

To submit a job that you want to check later to see if it is still in the queue, submit a job to the queue server with a *requestid* that is set to a unique string. You will need to keep track of these unique string values in order to check the job presence successfully. For example:

```
requestid=<unique string>
```

requestid is not required, however, the value must be unique among all the jobs in the queue. The same request ID dictionary holds unique ids set by the deprecated *cancelid* property. If a job is submitted with a duplicate *requestid*, the queue server returns the error: "Unable to submit job: Failed to queue publishing request. Attempt to submit two jobs with the same requestid"

To check a job presence in the queue, use the `isenqueued.aspx` entry point. Call with the *requestid* parameter set to the value established when the job was enqueued. For example;

```
http://server/queueserver/isenqueued.aspx?requestid=<unique string>
```

`IsEnqueued.aspx` returns “HTTP OK” if the job is still present in the queue or returns “NO” if the job is not yet in the queue, is cancelled and removed from the queue, or if it is out of the queue and is already being processed by a Job Processor. Additionally, unexpected error conditions are reported in the return of `IsEnqueued.aspx.page`. Once a job starts processing, it is removed from the queue server and the job status can be checked through the heartbeat progress.

Note: If both job parameters *`cancelid`* (deprecated as of v.7.3) and *`requestid`* are set in the publishing job request, preference is given to *`requestid`*, and *`cancelid`* is ignored.

5.2 Monitoring directory changes

You can configure Blazon Enterprise to monitor one or more directories for changes to files. When a directory is monitored, any file that is created or changed will be submitted for publishing. To monitor a directory, submit a job to the `push.aspx` page with the following parameters:

```
sourcedir=<a directory to monitor>
outputdir=<a directory (that exists) where output is placed)
outputformat=pdf
notificationurl=http://localhost:9999
register_dir=True
```

You can add any of the optional job parameters and they will be applied to all documents “submitted” to the monitored directory. This list shows the minimum needed for a job.

To stop monitoring a directory, submit a job with the same *`SourceDir`* and *`OutputDir`* values and `register_dir=False`.

Because the *`SourceDir`* and *`OutputDir`* must be accessible to all Job Processors in the system, they are typically specified as UNC Paths, not absolute drives.

5.2.1 Read and Write access

The account used by the Queue Service must have read and write access to the directory specified by the *`SourceDir`* parameter. The account used by the Job Processor must have read access to the directory specified by the *`SourceDir`* parameter and it must have read and write access to the directory specified by the *`OutputDir`* parameter.

Directory monitoring is recursive. If you monitor a top level directory, then changes and creations in subdirectories will be published. The output directory will contain a mirror of the same directory structure.

Note: You shouldn’t publish to XDL when monitoring a directory unless you configure a unique output directory (see next section). XDL is a multi-file format, and name collisions can happen if you publish multiple files in the same directory. You can use XDL as an output file type if you monitor a directory where each file exists exclusively in its own directory.

Since the Queue Service performs directory monitoring, changes will result in files inserted into the queue and published by the next available Job Processor.

When monitoring a directory, the Job Processor will send the notification URL for each file that is published.

5.2.2 Configuring unique output directories

You can configure the Queue Server, when in Directory Monitor mode, to allow generation of a unique output directory for each published file. This allows directory monitoring to work with all features, including thumbnails and text files, without overwriting published files.

To use this feature, submit the directory monitor job request with the parameter `dirmonuniquedirectory=true`. If this parameter is set to `true`, then the Queue Server will create a unique directory for each job it publishes, while still maintaining the hierarchy of the monitored directory tree.

Example 5-1: Output directory

If monitoring `\\mydirectory\monitor` and publishing to `\\mydirectory\output` and a file named `motor.dwg` is created in `\\mydirectory\monitor`, then the output directory might look like:

```
\\mydirectory\output\motor.dwg <time>\motor.pdf
```

If a file named `lawnmower.dwg` is created in `\\mydirectory\monitor\subdir`, then the output directory might look like:

```
\\mydirectory\output\subdir\lawnmower.dwg <time>\lawnmower.pdf
```

<*Time*> is UTC and formatted as <*Year-Month-Day Hour_Minute_Second*>, for example:

```
2009-12-18 15_41_33Z
```

If collisions occur, an additional -1, -2, and so on, will be appended to the time string.

Related Topics

- “Enhanced Directory Monitoring” on page 38

5.3 Receiving results using a notifier

Using the same example job we discussed earlier, this section will discuss what the Job Processor does when a job is completed.

The Job Processor calls the *NotificationUrl* with the http verb GET, POST, or MULTIPARTPOST (as determined by the value of parameter *NotificationVerb*). The results of the job in the query string, just like the call to push. The results are always *name=value* pairs, and all of the parameters submitted to push.aspx are included in the results, along with other information about the job, such as how long it took and what, if any, errors occurred.

The *NotificationVerb* can be set to either GET, POST, or MULTIPARTPOST. If you specify MULTIPARTPOST, then the content type is multipart/form-data, and each parameter is a separate data element. If you specify POST, then the content is application/x-www-form-urlencoded and the parameters of the reply are in the post data stream.

Because HTTP Post calls can often have many more parameters than GET calls, the Queue Server can remove non-job related parameters from the job before submitting it to the Job Processors. These parameters are called noise words and are defined in web.config using the *JOB_NOISE_WORDS* parameter.

The default list of parameters removed is:

ASP.NET_SessionId	*ALL_HTTP*	*ALL_RAW*
APPL_MD_PATH	*APPL_PHYSICAL_PATH*	*CONTENT_LENGTH*
GATEWAY_INTERFACE	*HTTPS*	*INSTANCE_ID*
INSTANCE_META_PATH	*LOCAL_ADDR*	*PATH_INFO*
PATH_TRANSLATED	*QUERY_STRING*	*REMOTE_ADDR*
REMOTE_HOST	*REMOTE_PORT*	*REQUEST_METHOD*
SCRIPT_NAME	*SERVER_NAME*	*SERVER_PORT*
SERVER_PORT_SECURE	*SERVER_PROTOCOL*	*URL*
HTTP_CONNECTION	*HTTP_HOST*	*SERVER_SOFTWARE*
AUTH_TYPE	*AUTH_USER*	*AUTH_PASSWORD*
LOGON_USER	*REMOTE_USER*	*CERT_COOKIE*
CERT_FLAGS	*CERT_ISSUER*	*CERT_KEYSIZE*
CERT_SECRETKEYSIZE	*CERT_SERIALNUMBER*	*CERT_SERVER_ISSUER*
CERT_SERVER_SUBJECT	*CERT_SUBJECT*	*CONTENT_TYPE*
HTTPS_KEYSIZE	*HTTPS_SECRETKEYSIZE*	*HTTPS_SERVER_ISSUER*
HTTPS_SERVER_SUBJECT	*HTTP_CONTENT_LENGTH*	*HTTP_CONTENT_TYPE*
HTTP_EXPECT		

As an integrator, you must write code to handle this call to use the parameters appropriately.

5.3.1 Using ResultFilterSettings.txt

An event system is in place on the Job Processor that allows administrators to filter completed jobs and take certain actions when the jobs meet specified criteria.

General usage notes

- Filters are set up on the Job Processor using a file called `resultfiltersettings.txt`.
- Administrators can match job data names or values using Regular Expression. If any of a job's names (keys) or values match a filters search pattern, the notification action is taken.
- Notifications are sent to events in the application log on the Job Processor machine.
- Notifications can be configured to be sent once in a certain time period (specified in seconds).
- By default, only licensing warnings are configured to be sent to the application log, and they are configured to do this once a day.

The configuration file, `resultsfiltersettings.txt`, contains:

- `FilterNameN` - the unique name of the filter.
- `FrequencyN` - the frequency, in seconds, to set this event.
- Either `NameN`, `ValueN`, or both - the regular expressions used to match the event.

Example 5-2: Sample filter entry in resultfiltersettings.txt

```
# Entry 0.  Put all errors in the Application event log.
FilterName0=Error in Publish Job
Name1=error.*
Frequency0=0
```

Line by line, this sample means:

`# Entry 0.Put all errors in the Application event log.`	This is a comment. Comments begin with a hash mark (#).
`FilterName0=Error in Publish Job`	This is the name of the filter. Filter names must be unique.
`Name0=error.*`	This line means match any line of job data with a name that begins with “error”.
`Frequency0=0`	This means put an entry in the application log every second.

Example 5-3: Sample with multiple filters configured

```
# Entry 0.  Log license warnings to the Application event log
FilterName0=License Warning
Name0=LicenseWarning
Frequency0=1440

# Entry 1.  Put all errors in the Application event log.
FilterName1=Error in Publish Job
Name1=error.*
Frequency1=0

# Entry 2.  If we publish a file named Airplane, note it.
FilterName2=Airplanes
Value2=.*[Aa]irplane.*
Frequency2=0
```

Entries must be numbered starting at zero and increasing one number at a time.

5.4 Document security settings

Security settings for published `CSF` and `XDL` documents are configured using XML and specified in jobs using the *`SecurityXmlFilename`* input parameter. An example document security file is provided in the Job Processor installation directory (`security.xml`) which can be reviewed for setting descriptions and usage details. If any of these settings are specified for jobs that export to formats other than CSF or XDL, an error occurs and no documents are exported. Thumbnails can be requested in such jobs, though the generated thumbnail image for each page will display the default CSF icon  rather than a rendering of that page's content.

5.4.1 Specific attribute notes

There are two attributes that you can specify for the *DateExpired* attribute element of published CSF files. Be aware that if you set the *`RelativeDays`* attribute, it then takes precedence over the year, month, and day (absolute) date attributes. The *`RelativeDays`* attribute takes the current date (at the time of the publishing) and adds the specified number of days.

Within the `<RightFlags>` element, the right corresponding to the element *`AuthorAndReviewMarkups`* encompasses the *`ReviewMarkups`* right. In other words, if *`AuthorAndReviewMarkups`* is set to `false`, it means the settings takes precedence if the *`ReviewMarkups`* right is also set to `false`.

5.4.2 ISO Banner settings

The ISO banners are strings of specific information (such as date, time, page number, and user name) assigned to a location on the document header and footer margins. The watermark is a semi transparent character string that stretches from the lower left corner to the upper right corner of the printed or on screen document.

Watermarks and banners can be used to support ISO 9000 and QS 9000 quality standards. They can contain text strings, token values, or, optionally (through integrations API), metadata fields from document management. With 12 possible banner locations allowing 10 lines of data for each location, you can add as much text as needed to provide sufficient traceability and status.

The CSF and XDL formats support the configuration of banners and watermarks that are visible only when printed or while viewing the document on a graphical display device. These are referred to as “print” and “screen” banners or watermarks, and cannot be specified for use on exported formats other than CSF or XDL. In addition to “print” and “screen” banners, “publish” banners can be configured that are visible when exporting to any rendered format.

Banner and watermark definitions are configured using XML and specified in jobs using the *BannerXmlFilename* input parameter. An example banner and watermarks configuration file is provided in the Job Processor installation directory with the file name `banners-watermarks.xml`

An XML schema file describing the format of a valid banners and watermarks configuration file is provided in the same directory with the file name `banners-watermarks-schema-1.0.xsd` and `banners-watermarks-schema-1.1.xsd` . Review these files for a detailed description of the possible settings. The 1.1 schema includes support for “last” and “-x” syntax allowing reference to the last page, and pages before the last page, in either the start or end position of a banner block.

Notes

- In `16.3` and earlier releases, banners and watermarks were configured using Document Security XML files. Starting with release 16.4, support for ISO banner configuration in these files is deprecated, and will eventually be removed in a future release. Support for Document Security XML used with earlier versions continues, but it is highly recommended to migrate banner/watermark configuration to the new XML model illustrated in the sample `banners-watermarks.xml` file.
- When running the product under a Demo license, the watermark cannot be configured and will only display and publish the string “Evaluation”.

5.4.2.1 Banner Macro commands

Blazon Enterprise supports the Macro Commands listed here for the watermark and banner text strings. Insert one of these commands in the banner or watermark string value and the string will be dynamically inserted at print or publish time. If a banner or watermark is editable, end-users can enter a percent symbol (%) in the edit field of a print/publish banner or watermark line to summon a list of available tokens.

Note: When the output format is set to produce CSF, banner macros are resolved during viewing sessions only and are ignored during document creation.

Banners and the watermark can contain text strings or one of the available token values:

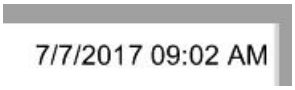
Table 5-1: Macro commands

Macro command	Description
`%Date (%daydate, %d, or %D)`	Inserts the date the print was spooled. If the tags are viewed on screen, the date at which the screen was last refreshed displays.
`%Time(or %t)`	Inserts the time the print was spooled based on a 12 hour clock (AM/PM). If the tags are viewed on screen, the time on which the screen was last refreshed displays.
`%MilTime`	Inserts the time the print was spooled based on a 24 hour clock.
`%Title`	Inserts the name of the document. If the Title macro is used, the title must be set using the *`DocumentTitle`* parameter
`%Page`	Inserts the page number.
`%TotalPages`	Inserts the total number of pages in macro output.
`%TotalOriginalPages`	Inserts the total number of pages in the original file (as opposed to the number of pages you are currently exporting, which can be a subset of the source document)
`%OriginalPageNum`	Inserts the number of the page in the original document (as opposed to its position in the export).
`%Login (or %User)`	Inserts the user name of the person who issued the print. If the user or Login macro is used, the job must set the parameter *`PublishUserName`*.

Macro command	Description
`%batespgno(x)`	Bates Number This tag is used to indicate the starting page number and the number of digits to use. For example, `%batespgno(0002)` would place 0002 on the first page, 0003 on the next page, and so on. A Bates Number is considered to be both the prefix and the page number WITH leading zeros. To create a bates ID, you can add text preceding the tag. For example, `Smith vs. Jones%batespgno(00001)` would result in "Smith vs. Jones000001" being printed on the first page, "Smith vs. Jones000002" printed on the second page, and so on. **Note:** When using this macro with `exportfileperpage=true`, only one export type should be used at once (`outputformatN` should not be used, and thumbnails should not be created) to ensure consistent numbering.
`%SourceFilenameNoExtension`	Inserts the source filename (not including the path), without the extension. If used for a multi-document publishing request, the name of the `Source0` document is used.
`%SourceFilenameWithExtension`	Inserts the source filename (not including the path), including the extension. If used for a multi-document publishing request, the name of the `Source0` document is used.
`%TargetFilenameNoExtension`	Inserts the target filename (not including the path), without the extension.
`%TargetFilenameWithExtension`	Inserts the target filename (not including the path), with the extension.

Example 5-4: Banner macro example

```
<RightTop string="%Date %Time" editable="true"/>
```

Result in printed/published output: 

5.5 Enhanced Directory Monitoring

Blazon Enterprise can be configured to monitor directory structures and publish all files as they are created or changed. This feature allows end users to easily configure and publish jobs through the Windows file system without any programming knowledge.

5.5.1 Overview

An enhanced type of directory monitoring is introduced in Blazon Enterprise version 7. Enhanced Directory Monitoring (EDM) makes it easy to use Blazon Enterprise features without writing custom integrations and without having to submit "URL" job submissions. Administrative pages are available on the Queue Server to allow quick and easy configuration of directory monitoring.

The installer creates a default Enhanced Directory Monitoring setup at installation time (if you choose to install this feature) which can be used immediately. Workflows and integrations can use this feature to integrate with Blazon Enterprise .

Enhanced Directory Monitoring supports:

- File publishing to PDF, TIFF, secure CSF, XDL, or None (used for generation of publishing artifacts without converting the source document)
- Thumbnail creation
- Burn-in of annotation, stamp, and watermark/banners
- Apply security settings (for CSF)
- Apply redaction scripts, zone templates (with Redaction module)

Publishing is controlled by the existence of templates, annotation files, and security files that are applied to or created in a task subdirectory. These files can be applied to all files contained in the source directory or using pairing by file name association. Process scheduling is optional and users can immediately publish files upon installation.

Enhanced Directory Monitoring setup and administration is done using the Queue Server status pages. End users can create new tasks and easily publish files and specify job options using drag and drop through the familiar Windows file system. No programming knowledge is needed, and folder permissions and security is defined by Windows file system security.

5.5.1.1 Concepts

Project - A Project for Enhanced Directory Monitoring is the top level directory that is monitored by the Queue Server. Associated with the Project directory is:

- The output format of the files that are published
- An optional schedule of when to publish.
- An optional time window during which files are published.
- A unique directory for each task.

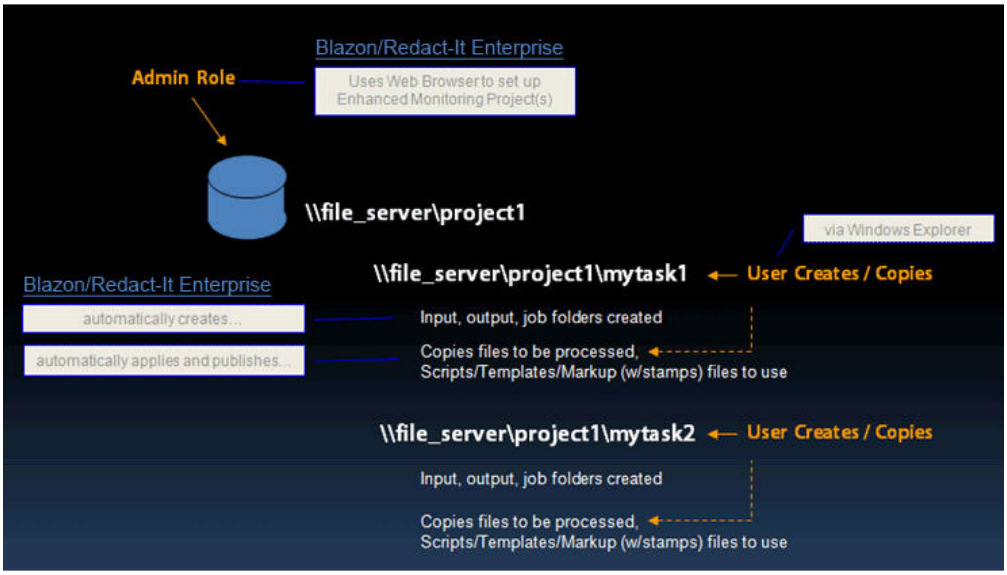
If the schedule is omitted, files are published as they appear or change in each categories Input directory. The schedule is a single time of day (such as 00:00:00 for midnight, 03:00:00 for 3 in the morning, 17:00:00 for 5 in the evening), local to the queue server, when files in the input directory are published.

Time window publishing defines a start and stop publishing time, using the same 24 hour format for defining the times. All files in the input directory at the start time will be published, and any files placed in the input directory after that time (up until the stop publishing time is reached) will be published. After the stop time, any files added won't be published until the next start time (24 hours later). The cycle repeats every 24 hours.

Notes

- Files must be removed from the directory after publishing to prevent daily republishing of these same files on the scheduled time.
- Projects names (the final directory in the full path) *MUST BE UNIQUE* across the entire system.
- Files are not placed in the Project's Directory. Rather, this directory merely contains zero or more `Task` directories.

Tasks - Tasks are directories in a `Project` directory where users configure publication options, place source files, and receive outputs. `Tasks` directories can be created ad-hoc and as users desire. When the Queue Server sees a directory appear in a `Project`, it then considers that directory a "Task" and creates, under the `Tasks` directory, a set of directories in which users manage publication options. All tasks in a given Project obey the same output type and schedule.



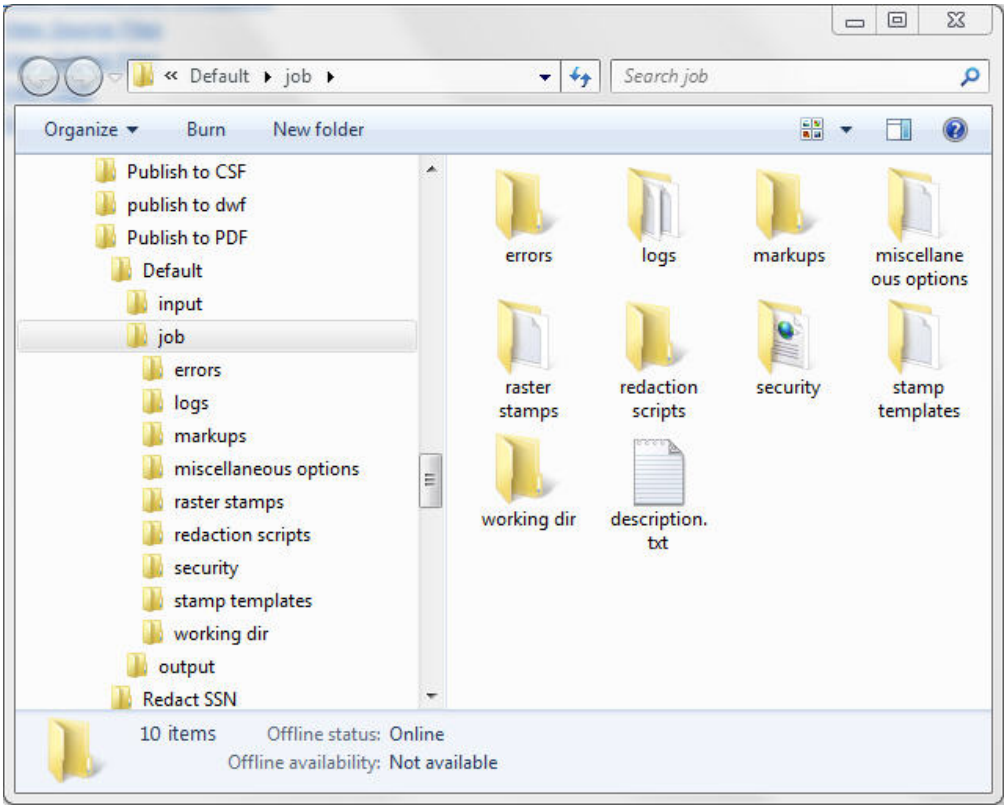
5.5.2 Anatomy of a Task

A Task's directory contains three folders: `input`, `output`, and `job`:

End-users can drag and drop files into a task's `input` folder and the file is instantly (or for each scheduled time) published with the job options specified in the `job` folder, and published to the `output` folder. Whenever a task directory is created, the directory structure is automatically created for the folder. The names of these folders cannot be changed. Default job settings files are provided with the installation that can be edited (for content) by users (`stampposition.txt`) and again, these filenames cannot be changed.

All files that are added to a task's input folder are published using the exact same settings, unless paired publication is being used (see "Paired files" on page 42). If users want to publish different files with different settings, he can create a new task directory and edit the job settings files located in the `job` folder. Files dropped into the new task directory's `input` folder will be published with the newly defined settings.

Tasks and projects provide very good security boundaries. Administrators should configure Windows file system permissions as appropriate, locking down projects or folders to meet their particular security needs. For example, locking down the `Security` folder will prevent users from editing banners and watermarks that are used for production of public documents.



- `input` - The folder where users place files for publication. It is also a location where users can place "paired" files (see "Paired files" on page 42).
- `output` - The folder where the final published output is placed.
- `job` - The job folder has several subdirectories. Users place files into these folders to affect publishing (such as burning in markups, applying stamps, and setting other miscellaneous job options). The sub folders of the job folder are:
  - `errors` - Contains the logs for any jobs that fail to publish (used by system).
  - `logs` - Contains the log files for all successful publish jobs and the heartbeat files for all publish requests (used by system).
  - `markups` - Users can place zero or more markup files in this directory to be burned in to the final publication.
  - `miscellaneous options` - Users can place zero or more text files in this directory. Each text file is expected to contain `name=value` pairs of additional options to apply at publish time. A sample file is provided in the folder.
  - `raster stamp` - Contains a file (called `stampposition.txt`) which can be edited to position a raster stamp during publication. Users place *one* additional PNG or JPG file into this directory to serve as the raster stamp. If more are added, no stamps are applied.

- `redaction scripts` - Users can place zero or more redactions scripts into this directory to apply redactions during publication. Redaction scripts require a Blazon Redaction module license and will fail if not supplied.
- `security` - This folder can contain one `security.xml` file (with any name) that is applied at publication time. A sample file is provided which can be modified and renamed.
- `stamp template` - Similar to `raster stamp` , this directory contains a `stampposition.txt` file and might contain *one* stamp template (XSP file designed in Brava! ActiveX or Brava! Desktop) to be applied at publication time.
- `working dir` - Used by the system for temporary files during job publication.

5.5.3 Paired files

In addition to setting up stamps, scripts, and templates in the `job` directories, a user can place files with specifically formatted names in the input directory that will then be applied to individual publish jobs.

The rules for precedence are as follows:

- A paired file can be one of one or more markup files, a single raster stamp, a raster stamp position, a single stamp template, a stamp template position, and/or a single security file. Miscellaneous job options can also be set in paired files.
- Paired files are files that have names similar to the input file they will apply to, and specific suffixes to indicate their purpose. For example:
  - If a user is publishing a file named `airplane.dwg`, then he can add a `security.xml` file to that file by creating a file named `airplane.dwg_security.xml` in the input directory.
  - If three markups files existed that needed to be applied to the `airplane.dwg` file during publish time, then those files would be named `airplane.dwg_markup_0.xrl`, `airplane.dwg_markup_1.xrl` and `airplane.dwg_markup_2.xrl`.
  - If only a single markup file needed to be applied to `airplane.dwg`, then the user could create `airplane.dwg_markup.xrl`.
- Paired files MUST be created before the target file (`airplane.dwg` in the example case) is placed in the input directory. Once the system sees `airplane.dwg` in the input directory, it will publish the job based on the state of the directory at that time, therefore, any paired files that appear after `airplane.dwg` will not be applied to the publish job.

Paired files precedence and combinations

Typically, job options (such as markups and stamps) that are in the job directory are combined with the paired files during publication. Thus, if a markup file is paired and placed in the input directory, it will be applied in addition to any markups in the `job/markups` directory.

The exception to this rule is the `security.xml` file, the raster stamp file, and the stamp template file. Since only ONE of these files can be applied at publication time, the file in the task's `job` directory will ALWAYS override any paired files.

5.5.3.1 Paired file suffixes

Following is a list of the suffixes that can apply to a file in the input directory.

Note: Any file that ends with these suffixes will NOT be published when it appears in the input directory, since it is assumed to be a pair to some other file.

Suffix	Purpose
`_markup.xrl`	A single markup file to be applied at publish time.
`_markup_N.xrl`	One of multiple markup files to be applied at publish time, start numbering at N=0.
`_security.xml`	A single `security.xml` file to be applied at publish time.
`_redaction.xrs`	Redact A single redaction script to be applied at publish time.
`_redaction_N.xrs`	Redact One of multiple redaction scripts to be applied at publish time, start numbering at N=0.
`_raster_stamp.png`	A single raster stamp file to be applied at publish time, PNG format.
`_raster_stamp.jpg`	A single raster stamp file to be applied at publish time, JPG format.
`_raster_stamp_position.txt`	A text file describing where to put the raster stamp (see "Positioning stamp templates and raster stamps" on page 44).
`_stamp_template.xsp`	A single stamp template to apply during publication.
`_stamp_template_position.txt`	A text file describing where to put the stamp template (see "Positioning stamp templates and raster stamps" on page 44).
`_misc_options.txt`	A text file of `name=value` pairs to set miscellaneous job options.

5.5.3.2 Brava! Desktop markup files

Paired publishing supports Brava! Desktop style markup files. By default, Brava! Desktop saves a markup file for a source file as follows:

Given a source file, such as `drawing.dwg`, the markup file will be named `drawing_dwg.xrl`.

Enhanced directory monitoring recognizes this pattern and will publish the file `drawing.dwg` with `drawing_dwg.xrl` if they are both in the input directory at job creation time.

If more than one markup file is required, the files are named as indicated in the previous table (see "Paired file suffixes" on page 43).

5.5.3.3 Positioning stamp templates and raster stamps

Stamp Templates and Raster Stamps are positioned by placing a text file along side the stamp template or raster stamp file to describe where to put the stamp. This file can contain comment lines beginning with a hash mark (#). It must contain a single line that is the stamp position parameter, as defined in the user's guide. This line is `Left|Top|Scale|Pages`.

In the case of stamps defined in the `job\Stamp template` or `job\Raster Stamp`, the file must live in the appropriate directory and be called `StampPosition.txt`. A default file is created when the task is created, so it can be easily edited.

In the case of paired templates, the position file must follow the suffix noted previously.

5.5.4 Setup and usage of Enhanced Directory Monitoring

Administrator steps:

1. Create a shared folder on a file server to hold all of the Projects.
2. Establish the project (top level file directory monitored on the Queue Server)
3. Set the output format (TIFF, PDF, CSF, TXT, or NONE).
4. Optional Set the time schedule for publishing.
5. Set appropriate folder securities.

User steps:

1. Create a task.
2. The system automatically sets up every `Task` directory to have three sub-directories (`input`, `output`, and `job`).
3. Copy desired templates, markups, miscellaneous options, and the security file xml into the appropriate `job` directories.

4. Copy desired source files into the `input` directory.
5. The system creates the resulting published files and saves them to the `output` directory.

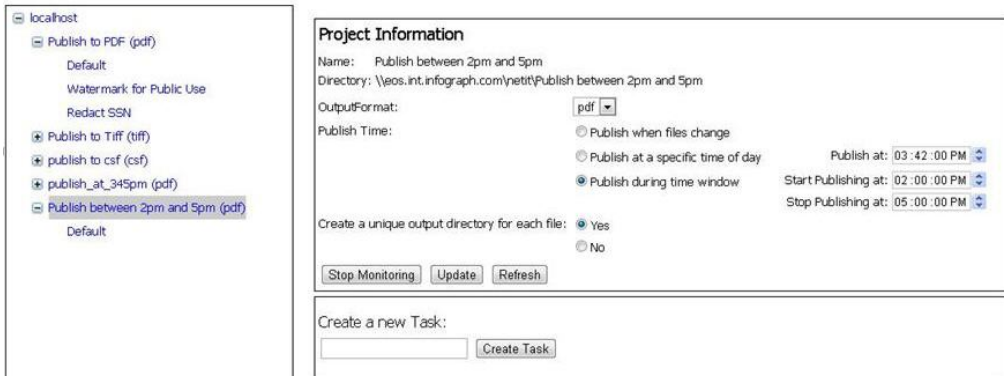
5.5.4.1 Administrative details

Administrators can use the web based management tool, `edmsettings.aspx`, to create new projects and add tasks to projects. Integrators can use the EDM API (see "EDM specific parameters" on page 45) to add projects and tasks to projects.

In typical usage, outlined previously, an administrator creates top level projects, and users or administrators create tasks for projects by creating sub-directories in the project directory. Once the Queue Server sees a new task directory under a project, it will populate it with all the required sub-directories.

Tasks can be deleted by deleting the directory from the project.

Administrators can stop monitoring a project using the Web UI (`edmsettings.aspx`). If an administrator stops monitoring a project, publishing will cease. The files and directories, however, are NOT deleted and remain on the file system until deleted manually.



5.5.4.2 EDM specific parameters

Project specific

- *`EnhDirAddsRcExtToOutputFileNames`* – To override the default project file-naming configuration, add this parameter, with a value of `false`, to a text file in the `/miscellaneous options` sub-folder within each EDM project task where you want to change the default naming convention.
- *`EnhDirPubTime`* – The time, in 24 hour format, when the system will publish all source files in the input directory. If omitted, then publishing happens as soon as a file is changed in the input directory.
- *`EnhDirPubWinStartTime`* – The time, in 24 hour format, when the system will begin publishing all source files in the input directory, and any files added to the

directory thereafter, until the time specified by *EnhDirPubWinStopTime* is reached. If omitted, then publishing happens as soon as a file is changed in the input directory (*EnhDirPubWinStopTime* will be ignored if set).

- *EnhDirPubWinStopTime* – The time, in 24 hour format, when the system will stop publishing any source files placed in the input directory, Files added to the input directory after this time will not be published until the time specified by *EnhDirPubWinStartTime* is reached. If omitted, and *EnhDirPubWinStartTime* is provided, then publishing stops 1 hour after the time specified by *EnhDirPubWinStartTime*.

Note: Files must be removed from the directory after publishing to prevent daily republishing of these same files on the scheduled time.

Project and Task specific

The following parameters can be configured for a project, or specifically for an individual task. To apply these to a specific task, enter them in a file in the `miscellaneous options` directory.

- *EnhDirSuccessDir* – A directory name, either a full path (for example, `\\FileShare\success`), or a path relative to the task directory (for example, `output\success`). If specified, the system will attempt to move source files to this directory upon successful job completion, removing the file from the input directory.
- *EnhDirFailureDir* – A directory name, either a full path, or a path relative to the task directory. If specified, the system will attempt to move source files to this directory upon job failure, removing the file from the input directory.
- *EnhDirUniqueOutputDir* - Boolean. If set to `true`, the system will create a unique directory under the job's output directory for each published file. Unique directory names are a combination of the source file name and the time of publication.

Redact Available with the optional Blazon Redaction module only

- *EnhDirEnableRedactionThreshold* – Boolean (default is `false`). If set to `true`, the system will inspect the value of *EnhDirEnableRedactionThreshold* and if, after the job is completed, the number of redactions performed was not greater than that value, then the published file will be placed in a `below threshold` directory. The name of this directory is `#igc# below redaction threshold`, and is located in the output directory of the task where the job was processed.
- *EnhDirRedactionThreshold* – An integer from 0-4294967294 (default is 0).
- *EnhDirRedactionReviewMode* – Boolean. If set to `true`, instead of generating a published file in the format configured for the project, the system will generate a consolidated markup file containing all markups, including redaction script results, and place the markup in a folder named “for review” at the same level as the `input` and `output` directories. Be aware that this parameter will have the effect of overriding any configured settings for:

- The *ExportMarkupFilename* job input parameter (the markup filename is automatically generated by the task).
- The *EnhDirSuccessDir* parameter (source files will not be placed in this directory after successful publication).
- The *EnhDirEnableRedactionThreshold* and *EnhDirRedactionThreshold* parameters (no files will be moved to the `#igc# below redaction threshold` directory, regardless of redaction results).

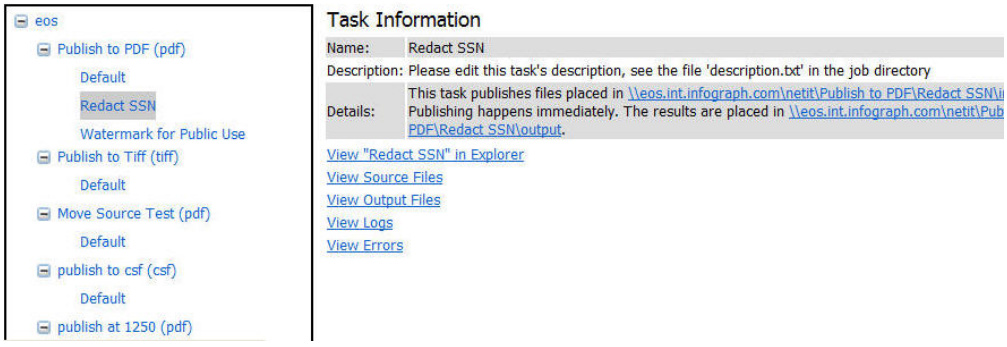
5.5.4.3 EDM API for creating projects and tasks

`EDMCreateProject.aspx` - This API allows integrators to programmatically add a project to the system. It has the following parameters (specified as query string or headers to the html call).

- *ProjectBaseDir* - Required. The full path to the project base directory. This should always be a UNC path.
- *OutputFormat* - Required. The type of output this project will generate. Either PDF, TIFF, CSF, XDL, TXT, or NONE.
- *CreateUniqueOuputDir* - Optional. If `true`, all publication in this project will create a unique directory. Required to be set to `true` if the *OutputFormat* is XDL.
- *PublishTime* - Optional. If set to a pars-able 24 hour time, create a project that publishes on a schedule. Otherwise, publish as soon as source files appear or change.

`EDMCreateTask.aspx` - This API allows integrators to programmatically add tasks to already created projects. It has the following parameters:

- *ProjectName* - Required. The name of the project in which to create the task. (Not the full path, only the project name.)
- *TaskName* - Required. The name of the task to create in the project.



Note: The server where the Queue Server is installed (example: `http://<queueserver>`) must be added to the Internet Explorer Trusted Sites list, and the checkbox for **Require server verification (HTTPS:) for all sites in this zone** must be cleared. When the server is added, the links on this page will open.

5.5.4.4 EDM setup default

The Blazon Enterprise Installer creates default Projects named `Publish to PDF`, `Publish to CSF`, and `Publish to Tiff`. The installer also creates a default Task, called **Default** in each of those directories (enabled by the `server.properties` setting; *`enhanced.directory.monitor.create.default`*). These projects are set up to publish as soon as a file changes and to NOT publish to a separate directory for each file.

5.6 Redaction usage

Redact This section is applicable if you are licensed to use the optional Blazon Enterprise Redaction module.

5.6.1 Redaction script details

See *OpenText Blazon Enterprise - Redaction Module Advanced User Guide (CLBRVW-SBZ)* for detailed scripting information.

You can create redaction scripts for phrases that you typically redact from your documents. To create a specifically formatted text file containing a list of phrases that you normally redact from your documents, along with a reason or any applicable exemption codes, follow the example shown here for formatting TXT or XRS file content.

```
Sample (txt):
     !redact!
     654321
     89089089
     9876543
     [:ssn:]
     [:email:]

Sample (xrs):
<?xml version="1.0" encoding="UTF-8"?>
<RedactionScript version="1">
<RedactionCommand comment="Redaction Reason"
hyperlink="www.infograph.com">
<SearchString string="John Doe" matchWholeWord="false" />
<SearchString string="111-222-3333" matchWholeWord="true" />
<SearchString string="[usphone]" matchWholeWord="false" />
...
</RedactionCommand>
<MacroDefinition name="ssn">
111-11-1111
</MacroDefinition>
<UnRedactionCommand>
<SearchString string="John Doe" matchWholeWord="true" />
...
</UnRedactionCommand>
<RedactAllRasters />
```

```
<SetRedactionColor red="255" blue="120" green="0"/>
<RedactionZone comment="Reason"
hyperlink="www.google.com"
left="0.1"
bottom="0.1"
right="0.3"
top="0.4"
pages="1,2-4"
isNDC="true" >
</RedactionZone>
<RedactionZonePeekThrough
left="10"
bottom="10"
right="100"
top="200"
pages="1,2-4"
isNDC="false" >
</RedactionZonePeekThrough>
...
</RedactionScript>
```

Supply the redaction script to Blazon Enterprise by including the following parameter to your call to `Push.aspx`. Redaction applies when the output type is CSF, PDF, or TIFF.

```
redactionscriptfilename=<full UNC Path to redaction script file>
```

The Brava! Desktop application can be used to model scripts for file sets to be redacted.

> **Important**
> Raster/graphic text is not handled by the **Find and Redact** function. It is strongly recommended to manually review any bulk redactions for accuracy as extra spacing and line wrapping might cause phrases not to be recognized.

5.6.2 Redaction script macro support

Redact In addition to the escape sequences listed earlier, you can use several macros for common search operations. The stringing of multiple search patterns together to form an abstracted concept defines a search macro. A concept ideally suited for use in redaction scripts since many related redaction strings can be grouped together and executed as one convenient macro command.

See *OpenText Blazon Enterprise - Redaction Module Advanced User Guide (CLBRVW-SBZ)* for detailed scripting information, search string rules, and examples, as well as information on using conditional macros.

5.6.2.1 Macro formatting

```
[:macro name:] (examples: [:usphone:], [:ssn:], [:email:])
```

Note: The old macro formatting style `\[macroname]` has been deprecated (as of v.7.0), but still remains functional for backwards compatibility. Search scripts that have previously been written in the old style will not be supported in future versions of Blazon Enterprise . Any new scripts should use the new macro syntax. If you need assistance updating your old scripts to use the new syntax, contact support.

Use one of the following standard macros provided for specific types of redaction searches.

Macro	Description
`[:ssn:]`	Finds and redacts representations of US Social Security Numbers such as 123-45-6789, including prefix.
`[:ssn_noprefix:]`	Finds and redacts representations of US Social Security Numbers such as 123-45-6789, excluding prefix.
`[:ssn_reveal4:]`	Finds US Social Security Numbers and redacts all but the last 4 digits (for example, 123-45). Does not include prefix.
`[:ocr_ssn:]`	Similar to [:ssn:] except allows both numbers and characters in the dash pattern string. For example, I23-45-67B9.
`[:usmoney:]`	Finds and redacts instances of US Currency, such as $100.42 or $1,588.
`[:email:]`	Finds and redacts email addresses in standard form.
`[:usphone:]`	Finds and redacts US style phone numbers.
`[:name:]`	Finds and redacts various combinations of a person's name. Used in scripts only and first requires the name to be defined.
`[:dob:]`	Finds and redacts instances of Date of Birth in standard forms.
`[:creditcard:]`	Redacts Visa, MasterCard, Discovery, Diners, American Express, FIGI.

The following form-based macros redact elements in a standard form.

Macro	Description
`[:zip:]`	Redacts "Zip: xxxxx[-xxxx]".
`[:usaddress:]`	Redacts Address: to end of \[Zip:].

Macro	Description
[:age:]	Redacts Age: x[x[x]].
[:gender:]	Redacts Sex: [M]|[F]|[Male]|[Female] or Gender [M]|[F]|[Male]|[Female].
[:race:]	Redacts Race: [any single word]|[African American]|[American Indian]|[Alaska Native]|[Pacific Islander]|[Native American].
[:acctno:]	Redacts "Account No: xxxxxx".
[:policyno:]	Redacts "Policy No: xxxxxx".
[:passportno:]	Redacts "Passport ID: xxxxxx".
[:drvlic:]	Redacts "Driver License: xxxxxx".
[:state:]	Redacts "State: XX OR XXXXXX XXXX".

5.6.3 Search string command support

Redact Search string commands can be used in any redaction script. See *OpenText Blazon Enterprise - Redaction Module Advanced User Guide (CLBRVW-SBZ)* for detailed scripting information, search string rules, and examples.

5.6.3.1 Text search modes

- **regex:**

  This is the most commonly used search method and indicates to perform a Regular Expression search. Macro references, within the string, are specified in the format "[:macroname:]", which works well along side the supported POSIX macro system. For example, in POSIX, "[:alnum:]" represents "[A-Za-z0-9]" (all the alpha and numeric characters). If you make your own macros, be sure to not use the name of one of the following that are already reserved by POSIX: `alnum`, `word`, `alpha`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, `xdigit`.

  > **Important**
  > All custom macros used in regular expressions must be written in regular expression format. *Do NOT reference an IGC wildcard macro from within a regex string or the reverse.*

- **phrase:**

  Find exact text match. This mode works similar to the **find text** command in most word processors, including Notepad. The string will be converted to a regular expression where spaces are allowed in-between characters. Spaces can be 0-to-many whitespace characters.

- **anyof:**

  Find any of the specified words or quoted phrases. This mode is similar to Google's default behavior regarding spaces and quotes. `"Hello there "John`

`Smith""` means to find “Hello” or “there” or “John Smith”. Each phrase string will be converted to a regular expression where spaces are allowed in-between characters and spaces can be 0-to-many whitespace characters.

- **macro:**

  Perform a find by looking up the named search string. For example, “email” would fetch the search string associated with the name email in the `search_macros.xml` file.

Note: The brackets should not be included. If you want to use the `ssn` macro, then your search string should be “ssn”. Not “[:ssn:]”.

5.6.3.2 The search string command

Redact Search string commands allow you to specify additional options when performing searches.

The search command string will be formatted as follows:

```
[mode] : [options] : [search string]
```

Example 5-5: Term-highlighted search

```
anyof:wiab-p-f-:this is the "search string"
```

If there are no options desired, then the command would look like this: (double colons)

```
anyof::this is the "search string"
```

Important

If using an XML editor rather than a text editor, the search string must obey the restrictions of XML which disallows the use of the following characters directly: `& " ' <>` (Instead you must use: `& &quot; &apos; < >`). For example, if editing in an XML editor, this example would be entered as: `anyof:wiab-p-f-:this is the &quot;search string&quot;` or `anyof::this is the &quot;search string&quot;`

The **`[mode]`** value can include any of the 5 previously listed (`phrase:`, `anyof:`, `macro:`, `igc:`, or `regex:`).

The **`[options]`** parameter of the string command can include combinations of the following. Each option character, except c, should be followed by either a **+** character or a **-** character to indicate whether or not the feature is being enabled or disabled (`true` or `false`). If neither is specified, it is assumed that the feature is being enabled (`true`).

Option	Description	Default
`w`	Only find answers on **w**hole word boundaries	false
`i`	Case **i**nsensitive	false
`a`	Highlight **a**ll results on this document	false
`b`	Look past paragraph **b**oundaries (try to step over headers, footers, and captions)	true
`p`	Search from current **p**osition	true
`f`	Search **f**orward	true
`s`	**s**ub-expression index	0
`m`	**m**acro definition. Allows you to specify the value associated with a macro	not set

Notes

- The option character flags can be separated by spaces, but is not a requirement.
- The sub expression index is not a flag; it is an integer value. Therefore, one or more digits is expected to be found after the “s”. Leaving it blank is invalid. The sub expression index parameter is only used for `regex` searches.
- Some text search modes don't use certain options as can be seen in their descriptions.
- If a string command is entered through the UI Search text field, the specified option flag value will override the defined defaults set for any of the Search menu items, such as Match Case and search direction. For example, if you search for “phrase::hello”, the state of the **Match Case** menu command is obeyed. If you search for `phrase:i:hello:`, then the “I” in the string is obeyed instead (`case insensitive=true`).
- By this same logic, when executing a redaction script command with the **Find and Redact** tool, you can set the value for **Find whole word only** option. If this option is specified in the find string command, the “w” flag overrides the state of the **Find whole word only** checkbox. For example, if you want the script command to search with find whole word and match case enabled, then you might enter `phrase:wi-:hello` in the **Find and Redact** search field, or in a redaction script text file.
- If a string passed into the API that does not begin with any of the reserved mode prefixes, then the search is assumed to be an older style search string (pre version 7.0). In this case, the string is examined to determine intent and looks for IGC wildcard escape sequences.

  If any wildcard escape sequences are found (aside from the “\<” and “\>” begin and end escape sequences), the string is interpreted as:

  ```
  igc:: [their string]
  ```

Otherwise, if the string starts with a "\[" and ends with a "]", it is interpreted as

```
macro:: [their string, minus those bits]
```

Otherwise, the string is interpreted as

```
phrase:: [their string]
```

If the phrase had the "\<" and "\>" begin and end escape sequences, then it will be flagged as wanting whole word boundary matching.

5.6.4 Redaction log processing

Redact The redaction audit log provides an audit trail that tracks various actions that occur when a file is published.

If you want to create a redaction audit log, when a file is published, specify `outputredactionauditlog=true` and provide a filename with `redactionauditlogfilename=<output xml file>`.

A log stylesheet is included in your installation `jobprocessor` directory. Whenever a log file is created, the log stylesheet, named `RedactionLogViewer.xslt`, is copied to the same directory as the log. This XSLT file is required for viewing the logs and if the log file is moved, the XSLT file must also be copied to the new location. Browsers for viewing the log stylesheet include supported versions of Internet Explorer and Firefox.

What is logged?

Events written to the log include the following:

- **Session summary** - logs the time when the redaction script is applied and includes the name of current user and name of the file.
- **Finalization** - logs the time taken for the redaction processing.
- **Run a script** - logs the time a redaction script is run and includes the name of the script file.

5.7 Image text recognition processing

The OCR (Image Text Recognition technology - or ICR: Intelligent Character Recognition) feature requires an additional Add-in purchase and will only be available when licensed.

When you submit one or more source documents of any format (not restricted to TIFF, BMP, PNG, PDF, or JPG) with `DoOCR=true`, the source documents are automatically processed and any image text in all of the documents becomes searchable.

Although OpenText Intelligent Character Recognition technology is highly accurate, all text in all image data with in all source formats is not guaranteed to be processed with 100% accuracy due to many aspects of an image's quality.

Usage Notes:

- OCR will only perform correctly by default on Western English (Latin alphabet) printed font characters. Additional upgrades can be purchased to support OCR of Asian Language characters.
- Our OCR processing supports character recognition for type characters only. Handwritten characters are not supported.
- The OCR process embeds recognized text to the internal format as hidden text which can then be exported as PDF/XDL/CSF documents with recognized, searchable, formatted, hidden text.
- For better text recognition results, the raster resolution should be at least 300 x 300 DPI before performing OCR.
- OCR processing can be performed on single- and multi-source document jobs of any format supported by Blazon Enterprise . Multi-source jobs and single-source jobs with documents (regardless of whether or not they are in a format natively supported by the Open Text character Recognition Engine) are first loaded into our internal format, analyzed for any image or drawable content, and if such content is found, a PNG temporary image is created prior to OCR processing. The processing result (in a form of hidden, formatted searchable image underlying text) is inserted back into the internal page content. Any markup operations supplied with the job are applied during this pre-OCR conversion step, as is markup export. If text exports are requested, or final publication to formats other than PDF is requested, these operations will be performed after OCR processing has completed. Note that this activity can significantly increase the total processing time for a given job compared to the same job with `DoOCR=false`, so job *`timeout`* parameters might need to be increased relative to jobs that do not perform OCR operations.

  Markup processing is not affected by OCR processing and therefore markup extents are not changed.
- Redact Redaction scripts, audit log, and other redaction-related processing on OCR jobs is always performed after OCR processing.

Chapter 6

Job Options

This section covers information about configuring the Blazon Enterprise optional parameters to publish jobs.

The job options covered in this section detail how to tell the system what you want it to do. They are to Blazon Enterprise what all of the menus, buttons and commands are to a desktop application. Information is presented in a series of tables for quick and convenient referencing.

Integrators use job options to tell the system what to do, such as publishing a file, and then what to do with the file after it is published, such as preferred output format and where to put the published results.

6.1 Optional job input parameters

This section describes the various job input parameters which can be supplied to each job. The parameters can be included in the URL of the publish request or other form of publish request to convert the content.

6.1.1 General job input parameters

SOURCE

Default value: *<path to file>*

Description: The complete path to the source file to publish. The Job Processor must have access to this file. Not valid for directory monitoring job.

Note: Only *network paths* are supported for source parameter path values. Local paths (but not mapped drives) can be used for single box testing only. The source parameter should be specified to the JP in *all lower case*. If mixed case is used, the JP status page might not display correctly.

SOURCEN

Default value: *<path to file>*

Description: With multi-doc publishing, you can combine multiple source documents to publish as one exported file. To do so, set multiple source file paths using *sourceN* instead of *source* (you cannot specify both). The various source files do not have to be of the same format and will all publish to the format specified by *OutputFormat*.

Examples:

```
source0=\\computer\share\Airplane.dwg

source1=\\computer\share\AirplaneSpecs.doc

source2=\\computer\share\AirplaneBrochure.pdf
```

Notes

- XDL files are supported as input types.
- Any CSF source files that have any security attributes set are not allowed.
- The *sourceN* parameter should be specified to the JP in *all lower case*. If mixed case is used, the JP status page might not display correctly.

MULTISOURCEMERGE

Default value: `true`

Description: If two or more source documents are provided using the *sourceN* input parameters, and *MultiSourceMerge* is set to `false`, then, rather than merging the source documents into a single result, all requested artifacts are individually processed for each source file. The resulting files are named with a prefix derived from the position of the source document in the list of sources, and the file name of the original source document.

In addition, all other 2D job results (such as thumbnails, exported markup files, redaction script execution (Redact with Blazon Redaction module), and reported loader information) are each processed and reported separately for each source document.

Examples:

A multi-source job with the following source list:

```
source0=\\path-1\some-file.tiff
source1=\\path-2\some-file.doc
source2=\\path-2\some-file.doc
```

and an *OutputFormat* of `PDF`, would produce the following files in the target directory:

```
\\target-path\0.some-file.tiff.pdf
\\target-path\1.some-file.doc.pdf
\\target-path\2.some-file.doc.pdf
```

Notes

- *driverinfo* results are reported using output parameters *driverinfo.0*, *driverinfo.1*, ... *driverinfo.N*.
- Redact redaction script results are reported using output parameters *redactionscriptresults.0*, *redactionscriptresults.1*, ... *redactionscriptresults.N*.
- Thumbnails are produced following the same name prefix convention as the converted output files (for example, `0.some-file.tiff_77x100_page_2.jpg`).
- Redact redaction audit logs are produced following the same name prefix convention as the converted output files.

- Items such as banners, watermarks, markups, and stamps are applied to each source document.
- Table of content entries are honored and only applied to the requested source documents.
- Multi-source markup requests are honored, so that only markup files specified as *sourceO_markupN* are applied to the document identified by *sourceO*.
- *DoOcr* supports a multi-source job.
- *ExportPageList* is applied identically to each source document, so warnings or errors can result if the page list contains one or more out-of-range page numbers.

CreateToc

Default value: `false`

Description: If set to `True`, this option creates a table of contents on the first page of the published document. TOC entries will contain the source file names for each source being published in the document. You can use the *TocEntry* parameters or the *StructuredToc* parameter to customize the TOC entries.

TocEntry

Default value: <*string*>; not set

Description: Use this parameter to specify table of contents and override entries when *CreateToc* is set to `true`.

Example: `tocentry=some text`.

TocEntryN

Default value: <*string*>; not set

Description: Use this parameter to specify table of contents and override entries in a multi-source job when *CreateToc* is set to `true`. When publishing a multi-source job, if *TocEntryN* is missing, the file name of the source is used.

Example:

```
tocentry0=some text

tocentry1=some other text

tocentry2=yet more text
```

StructuredToc

Default value: <*path to file*>; not set

Description: When *CreateToc* is set to `true`, you can use this parameter to specify the table of contents and source document ordering in a multi-source job . The file path provided must be a JSON file accessible by the Job Processor that adheres to the JSON schema found in `TOCSchema-1.0.0.json` (located in the Job Processor folder). See this schema file for details on the table of contents formatting available. Providing this file overrides any entries given in *TocEntryN* parameters.

BookmarkPerSource

Default value: `false`

Description: If set to `true` on a job that is merging multiple sources, this option creates a bookmark for each source file using the filename of the source file. This job option cannot be used with the *`CreateTOC`* job option, which creates its own bookmarks.

Example: `bookmarkpersource=false`

Header.ImageFile

Default value: <*path to file*>; not set

Description: Applies an image centered across the top of each page of the output document. This parameter must be a file path to an image file reachable by the Job Processor. This image should be a PNG or JPG file. If used, *`Header.Height`* and *`Header.Inset`* cannot be empty.

Example: `header.imagefile=\\server\sourcefiles\myheader.png`

Header.Height

Default value: <*integer in inches*>; not set

Description: This parameter is required if *`Header.ImageFile`* is non-empty. The height is the desired height of the image on the page, in inches. The width will be chosen based on the proportions of the image, given the height.

Header.Inset

Default value: <*integer in inches*>; not set

Description: This parameter is required if *`Header.ImageFile`* is non-empty. The inset is the margin between the top of the image and the top of the page.

Footer.ImageFile

Default value: <*path to file*>; not set

Description: Applies an image centered across the bottom of each page of the output document. This parameter must be a file path to an image file reachable by the Job Processor. This image should be a PNG or JPG file. If used, *`Footer.Height`* and *`Footer.Inset`* cannot be empty.

Example: `footer.imagefile=\\server\sourcefiles\myfooter.png`

Footer.Height

Default value: <*integer in inches*>; not set

Description: This parameter is required if *`Footer.ImageFile`* is non-empty. The height is the desired height of the image on the page, in inches. The width will be chosen based on the proportions of the image, given the height.

Footer.Inset

Default value: <*integer in inches*>; not set

Description: This parameter is required if *`Footer.ImageFile`* is non-empty. The inset is the margin between the top of the image and the top of the page.

Target

Default value: <*path to directory*>; not set

Description: The target directory, where the published artifacts are placed. With directory monitoring, published output is specified in the directory monitoring job.

Only mailto, printer URI's and network paths are supported for *target* parameter path values. Local paths (but not mapped drives) can be used for single box testing only.

Notes

- **Mailto path:** If the target value is a `mailto` URI, files produced by the job will be sent using e-mail as attachments addressed to the recipients specified in the URI. The URI must conform to RFC 1738 formatting, so while a single recipient address must be present in the `mailto` URI, multiple recipient addresses can be specified only through query parameters. Query parameters can be included directly in the target `mailto` URI, or they can be separately specified through additional input parameters that follow a naming convention of: `target.mailto.<query-parameter-name>=<query-parameter-value>`. For example, the following two forms of input target parameters are equivalent:

  Single-parameter form:

  ```
  target=mailto:somebody@somewhere.com?to=somebody-else@somewhere-else.com,nobody@nowhere.com&smtp.server=some-server.com&smtp.port=25
  ```

  Multiple-parameter form:

  ```
  target=mailto:somebody@somewhere.com target.mailto.to=somebody-else@somewhere-else.com,nobody@nowhere.com target.mailto.smtp.server=some-server.com target.mailto.smtp.port=25
  ```

  Both the *smtp.server* and *smtp.port* query parameters are required in order to deliver job results as email messages, and must be set to a valid SMTP relay server name and port. The complete set of supported query parameters and legal values are described under the following parameters `target.mailto.<query-parameter-name>`.

- **Printer path:** The Job Processor will accept a target value that is a `printer` URI of the form `printer://<printer-host>/<printer-name>`. For jobs that produce only a single artifact file, that file will be sent to the requested printer. If the job can produce multiple files (excluding the heartbeat log) and the target is set to a printer URI, the job will fail. The URI must conform to RFC 1738 formatting. Query parameters can be included directly in the target printer URI, or they can be separately specified through additional input parameters that follow a naming convention of: `target.printer.<query-parameter-name>=<query-parameter-value>`. For example, the following two forms of input target parameters are equivalent:

  Single-parameter form:

  ```
  target=printer://printer-host/printer-name?copies=2
  ```

  Multiple-parameter form:

```
target=printer://printer-host/printer-name

target.printer.copies=2
```

- If the printer-host and path separator are omitted, the Job Processor will attempt to send the request to a printer named "printer-name" on the host where the Job Processor is running. Otherwise, the requested printer-name must be available from the specified printer-host using Microsoft Windows UNC network printer naming syntax, For example, if `target=printer://printer-host/printer-name`, then `\\printer-host\printer-name` must be a valid network printer accessible from the Job Processor Microsoft Windows machine.
- If an output format of `None` or `Txt` is specified, the printed document will be a near-native rendering of the published document after all job options have been applied. Printing a text rendering of the published document is not supported.
- The Job Processor does not enforce any limits on the number of copies or pages that can be printed in one job or during any time period (see *`Target.Printer.Copies`*).
- The Job Processor can be configured to completely disable printing or to restrict printing to specific printers. To submit printers as publishing targets, see *`publish.to.printers.enabled`* and *`publish.to.printers.whitelist`*.

Target.MailTo.Smtp.Server
Default value: <*path to file*>: not set

Description: This parameter is required for all jobs whose target is a `mailto` URI and is expected to be a valid reachable SMTP relay server address. If not specified or invalid, the job will fail.

Example: `target.mailto.smtp.server=(`*`<smtp-server-dns-name>`*`|`*`<smtp-server-ip-address>`*`)`

Target.MailTo.Smtp.Port
Default value: <*path to file*>: not set

Description: This parameter is required for all jobs whose target is a `mailto` URI and is expected to be a valid SMTP relay server port used by the specified SMTP server. If not specified or invalid, the job will fail.

Example: `target.mailto.smtp.port=`*`<smtp-server-port>`*

Target.MailTo.UseSecureTransport
Default value: `false`; not set

Description: If set to `true`, the Job Processor attempts to connect to the SMTP server using either TLS or SSL.

Example: `target.mailto.smtp.useSecureTransport=(true|false)`

Target.MailTo.Smtp.Username
Default value: <*string*>; not set

Description: This parameter is required if `target.mailto.smtp.password` is non-empty.

Target.MailTo.Smtp.Password
Default value: <*string*>; not set

Description: This parameter is required if `target.mailto.smtp.username` is non-empty.

Target.MailTo.To
Default value: <*email address*>; not set

Description: Comma-separated list of one or more additional recipients for the **To:** line of the e-mail message to be sent.

Example: `target.mailto.to=username@domain,username2@domain, username3@domain`

Target.MailTo.From
Default value: <*email address*>; not set

Description: A single recipient address for the **From:** line of the e-mail message to be sent. If multiple addresses are specified, the job will fail. If not set, defaults to `job-processor@<jp-machine-name>.<jp-machine-domain>`

Example: `target.mailto.from=username@domain`

Note: Some relay servers might not deliver e-mail to recipients if the From: address doesn't correspond to a valid e-mail account. Other servers might forward the message, but some e-mail clients might filter these out or place the message in a spam or junk folder. To avoid this kind of issue, use a valid email account as the value for the *`target.mailto.from`* input parameter.

Target.MailTo.Cc
Default value: <*email address*>; not set

Description: Comma-separated list of one or more additional recipients for the **CC:** line of the e-mail message to be sent.

Example: `target.mailto.cc=username@domain,username2@domain, username3@domain`

Target.MailTo.Bcc
Default value: <*email address*>; not set

Description: Comma-separated list of one or more additional recipients for the **Bcc:** line of the e-mail message to be sent.

Example: `target.mailto.bcc=username@domain,username2@domain, username3@domain`

Target.MailTo.Subject
Default value: <*text*>; not set

Description: The subject of the e-mail message to be sent. If not set, defaults to artifacts from job: <*job-id*>.

Example: `target.mailto.subject=[text]`

Target.MailTo.Body
Default value: <*text*>; not set

Description: The text of the message body to be sent. If not set, defaults to artifacts from job: <*job-id*>. Use double pipe (`||`) to create line feeds in the body text.

Example: This example creates two lines as:

Please review and submit comments by the end of this week.

Final draft will be distributed at the end of the month.

```
target.mailto.body=Please review and submit comments by the end of this
week. ||Final draft will be distributed at the end of the month.
```

Target.MailTo.AttachmentName
Default value: <*text*>; not set

Description: If there is only one file produced by the job, then that one attachment is named using the value of this parameter. If more than one files are produced by the job, this parameter is ignored.

Example: `target.mailto.attachmentname=[text]`

Target.MailTo.CopySelf
Default value: `false`

Description: If set to `true`, the **From** recipient address is added to the **Bcc:** list for the message.

Example: `target.mailto.copyself=(true|false)`

Target.Printer.Copies
Default value: `1`

Description: The number of copies of the published document to print. Valid value is a positive integer greater than 1. If not set, the default is 1.

Example: `target.printer.copies=1`

Target.Printer.ScaleToFit
Default value: `true`

Description: Determines if pages are scaled to fit on the printable area, or are printed at the original size and cropped to the page. If not set, the default is `true`.

Example: `target.printer.scaletofit=(true|false)`

Target.Printer.RotateToFit
Default value: `true`

Description: Determines if pages are rotated to best fit on the printable area, or are printed with the original orientation. If not set, the default is `true`.

Example: `target.printer.rotatetofit=(true|false)`

RetainAttachments

Default value: `false`

Description: When `true`, a document's file attachments are retained as attached binary files rather than appending the content within the attachments as extra pages. Currently, this setting applies only to files published using the PDF2DL loader and output to PDF. When set to `false`, attachments are treated as additional documents that are appended as additional pages. In this case, if the attachments are unsupported file types, the job will fail.

Example: `retainattachments=(true|false)`

Notes

- Attachments linked to bookmarks are not currently retained as file attachments and are always appended as extra pages (if their file type is supported).
- The `retainattachments=true` option causes attachments to be retained in their original binary format without being opened or parsed.
- The `retainattachments=true` option will cause the job to fail if the requested PDF output compliance format does not support passing through the attachments. For example, the job will fail if the PDF/A-1a compliance formats (A-1a, A-1b) do not support the existence of attachments. If PDF/A-2a format is requested, any attachments must already be PDF/A-2a compliant PDF files because PDF/A-2a only allows PDF/A-2a attachments. If PDF/A-2b format is requested, any attachments must already be PDF/A-2b compliant PDF files because PDF/A-2b only allows PDF/A-2b attachments.

LoaderProfile

Default value: `default`

Description: Specifies the ID of a Loader Profile to use for the job. A Loader Profile collects a particular configuration of available loaders and loader parameters. The only installed Loader Profile is the "default" profile. In cases where several integrations share the same Job Processor but want to provide different loaders or loader behavior, each integration can define a Loader Profile, install it on the Job Processor, and select their profile by providing the ID in their jobs as a value for this parameter. For details, see *OpenText Blazon Enterprise - Blazon Enterprise SDK Guide (CLBRVW-RBZ)*. The default value of this parameter is `default`, which selects the base installed profile.

Example: `loaderprofile=<profile ID>`

OutputFormat

Default value: *<extension>*

Description: The format of the output file. Valid values are either `XDL`, `CSF`, `PDF`, `TIFF`, `TXT`, or `NONE`. When publishing to TIFF, or when markups are included in the publishing request, it is highly recommended that a page size is set through the *`OutputPageSize`* parameter to avoid huge file sizes.

If set to `NONE`, there will be no export of the published file, but other artifacts (such as thumbnails and text files) will be generated and copied to the target directory.

`TXT` is used to produce either a single file containing all of the text in the source documents, or a file for each page of text. When `exportfileperpage=true`, an *`OutputFormat`* value of `TXT` has the same effect as setting `gettext=true`, generating multiple files, each containing the text from one of the pages of the source documents. When `exportfileperpage=false`, a single file of text is produced. The existing input parameters *`TextFormatUtf8`* and *`TextFormatCrlf`* are both honored when `outputformat=txt`, and *`TextFilePrefix`* is honored only when `exportfileperpage=true`.

Example: `outputformat=none`

OutputFormatN

Default value: <*extensions*>

Description: If two or more output formats are desired for one or more source documents, the different formats can be requested by supplying multiple *`outputformatN`* input parameters, where N = 0, 1, 2, ...

Note: The Job Processor will fail any job for which PDF, TIF, CSF, XDL, or TXT appears more than once as a value of an *`OutputFormatN`* parameter (for example, if both `outputformat1=tif` and `outputformat3=tif` are listed, the job will fail). To establish file names to use for the different requested formats, see the next parameter, *`OutputFormatN.Filename`*.

Example: `outputformatN=(pdf|tif|csf|xdl|txt)`

```
outputformat0=xdl
outputformat1=tif
outputformat2=pdf
```

OutputFormatN.Filename

Default value: <*filename and extension*>

If not set, the value defaults to the set value of *`CsfName`* (without the extension), or the original source document file name (with an extension matching the corresponding *`OutputFormatN`* format).

Description: If two or more output formats are desired for one or more source documents, the names of the different output files can be individually specified using this parameter. If *`CsfName`* is set, then the filename from that parameter value (not including the extension) will be used as the filename for each requested output format. If neither *`OutputFormatN.Filename`* nor *`CsfName`* are provided, then the different output files will each use the name of the original source document with an extension matching that of the corresponding output format.

Example: `outputformatN.filename=myfile.pdf`

ForceSourceFormat, ForceSourceFormatN

Default value: <*extension*>

Description: These parameters allow users to define the specific source file format being used, rather than allowing the format to be detected through auto recognition. Use `forcesourceformat=<ext>` for single source requests and *`ForceSourceFormatN`* (N=0,1,2...) for multi-source requests (see *`Source`* and *`SourceN`*). Any leading "." character provided for values will be ignored. Unrecognized format values will fail during load. If these parameters are used, the format specified overrides autorec.

Example: `forcesourceformat=pdfforcesourceformatN=(pdf|tif|csf|xdl|txt)`

```
forcesourceformat0=xdl
forcesourceformat1=tif
forcesourceformat2=pdf
```

NotificationUrl

Default value: <*URL*>: not set

Description: The URL to call when the job is completed.

NotificationVerb

Default value: <*HTTP Method*>: not set

Description: Http method to use for making calls to the Job Processor. Valid values are `GET`, `POST`, or `MULTIPARTPOST`. The default value is GET. If *`NotificationVerb`* is set, then the Job Processor will call *`NotificationUrl`* with the set HTTP verb.

Example: `notificationverb=GET`

SourceN.MarkupN

Default value: <*Path to markup file*>: not set

Description: (Use in place of deprecated (as of v.16.0) *`SourceN_MarkupFilename`*) If two or more source documents are provided in a job, and one or more markup files are to be applied to one or more of the source documents, this parameter allows the submitter to specify to which source document each markup should be applied.

MarkupN

Default value: <*Path to markup file*>: not set

Description: (Use in place of deprecated (as of v.16.0) *`MarkupListFilename`*) If two or more markups are to be applied to the final document that is produced by a given job, that list of markup files can be specified by providing multiple *`MarkupN`* parameters.

Example: `markupN=\\share\directory\markup.xml`

```
markup0=\\some-share\some\markup.xrl
markup1=\\some-other-share\some\other\markup.xrl
markup2=\\yet-another-share\yet\another\markup.xrl
```

SourceN.PageList

Default value: <*page list*>: not set

If not set, all pages from the source document are included for publishing.

Description: If two or more source documents are provided in a job, this parameter allows the submitter to request that only specific pages should be included in the final document. Page numbering is zero-based. Page ranges are specified with two integers separated by a hyphen. If markups are applied to different sources in the request, only markup elements from the pages included in each source page list are applied to the corresponding pages in the corresponding source document.

Example: `sourcen.pagelist=<comma-separate list of zero-based page numbers or page ranges>`

```
source0=\\some-share\some\file.docx
source0.pagelist=1,3,5-10,13
source1=\\some-other-share\some\other\file.pdf
source1.pagelist=0-9
source2=\\yet-some-other-share\yet\some\other\file.ppt
```

In this example, assuming that `file.ppt` has 11 pages, the final document would have a total of 30 pages: 9 pages from `file.docx`, 10 pages from `file.pdf`, and all of the pages from `file.ppt`.

MarkupN.PageShift
Default value: `0`

Description: If it is known that the page indexes in a provided markup file need to be shifted forward or backward to align correctly with the supplied source document in a job, this parameter can be provided to specify that offset. Here N = 0, 1, 2, ..., corresponding to the markup file for which the shift is to be performed.

The shift value must be an integer, and can be negative, zero, or positive. The default value if not set, is 0.

Example:

```
source=\\path\to\a\source\file.pdf
markup0=\\path\to\a\markup\file.xrl
markup0.pageShift=-2
```

This request example subtracts 2 from all of the page indexes in the provided markup file, discards any indexes whose adjusted value is less than zero, and applies the markup entities according to the remaining adjusted indexes. If the `pageShift` value were +2, this would increase all of the markup page indexes by 2 and then apply its entities according to the adjusted indexes.

SourceN.MarkupN.PageShift
Default value: `0`

Description: If it is known that the page indexes in a provided markup file need to be shifted forward or backward in order to align correctly with one of the supplied source documents in a multi-source job, this parameter can be provided to specify that offset. Here M = 0, 1, 2, ... corresponding to the source

file to which the markup is to be applied, N = 0, 1, 2, ..., corresponding to markup file for which the shift is to be performed.

The shift value must be an integer, and can be negative, zero, or positive. The default value if not set, is 0.

Example: Multi-source markup page shift request example:

```
source0=\\path\to\a\source\file.pdf
source1=\\path\to\a\different\source\file.pdf
source1.markup0=\\path\to\a\markup\file.xrl
source1.markup0.pageShift=-2
```

This request subtracts 2 from all of the page indexes in the provided markup file, discards any indexes whose adjusted value is less than zero, and applies the markup entities according to the remaining adjusted indexes to the pages of the document referenced by the *source1* parameter. If the `pageShift` value were +2, this would increase all of the markup page indexes by 2 and then apply its entities according to the adjusted indexes to the referenced source document.

ExportMarkupFilename

Default value: <*filename*>: not set

Description: Specify the name of a file that will receive all of the markups created from redaction scripts Redact and markup files. If you use this option, the resulting published file WILL NOT have markups or redactions burned in. Rather, they will be placed in the file for review. The markup file applies to either the published XDL or CSF rendition of the file, or to the original file when opened using Brava! viewing technology.

SecurityXmlFilename

Default value: <*filename*>: not set

Description: Specify the name of the security settings for publication file. See "Document security settings" on page 34 document security settings.

BannerXmlFilename

Default value: <*filename*>: not set

Description: Specify the name of the banner and watermark settings for publication file. See "ISO Banner settings" on page 35.

Markup.FailOnNoEntitiesApplied

Default value: `true`

Description: If the application of any markup file to the source document in a job results in no entities added to any page, this parameter allows the submitter to specify whether the Job Processor should fail the job (`true`) or continue processing without error (`false`).

Example: `markup.failonnoentitiesapplied=(true|false)`

ExportChangemarkSummary

Default value: `false`

Description: If `true`, a Changemark summary page is appended to the PDF or TIFF output file.

Example: `exportchangemarksummary=(true|false)`

ExportRtfChangemarkSummary

Default value: `false`

Description: If `true`, and a markup file contains Changemark comments, a Changemark summary document is generated in RTF format, along with the output file. This file, named `ChangemarkSummary.rtf`, is generated in the target directory and contains a list of all Changemark comments, along with the author, date, and time of each comment. The RTF file also contains links to an image of the page on which they were added, together with a list of any Changemark replies to the comment.

If there are no Changemark entities applied, no comments are listed in the generated document.

If `true` on a multi-source job, and `MultisourceMerge=false`, then separate report documents are generated; one for each source document. Each generated report lists only the comments associated with the corresponding source document. The file name format for the generated documents is `<n>.changemarkSummary`, where <n> corresponds to the source document index in the submitted job.

Example: `exportrtfchangemarksummary=(true|false)`

DocxInsertChangemarks

Default value: `false`

Description: If `true`, and a Microsoft .docx format is provided as a source file in a single-source job, and one or more associated markup files contain **Text Highlight** Changemarks, this option will insert the **Text Highlight** Changemarks at the locations indicated in the markup file so that they appear as Microsoft Word Comments added to the original document.

Usage notes:

- This option only inserts **Text Highlight** Changemarks and no other Changemarks are inserted into a document.
- Only **Text Highlight** Changemarks created with Blazon/Brava! version 16.0.0 or later are supported.
- The modified version is copied to the requested target directory, given the same file name as the original. The original source document is not modified.
- This option is only supported for single-source jobs where the source document is in docx format. No other Word document or Office formats are supported.
- Multi-source jobs that also set this parameter to `true` are failed.
- Redaction markup elements and scripts Redact, stamps, banners/ watermarks can be provided, but they will not affect the contents of the docx source document.

- All other job options for exporting to different formats, thumbnails, text, and so on, are supported.
- All markup formats are supported (.xrl, .mrk) as long as the markup file is valid.
- Blazon supports highlighting some types of text that cannot be highlighted using Word. Examples are bullets, formatting characters, and text boxes. If these text types are used with a **Text Highlight** Changemark, they might not be highlighted correctly when exported to Word.

Example: `docxinsertchangemarks=(true|false)`

ExportBlockAttributes

Default value: `true`

Description: When `true`, all block attributes are included in the exported document. Use `false` to omit them.

Block attribute are text string that are attached to certain regions or elements of a document. They are usually displayed in a tooltip when the mouse is moved over the block. These attributes can be exported to formats that support this type of data. For complex documents with many small blocks, the presence of the block attributes can degrade viewing performance of the exported document. This setting provides an option for omitting them to improve performance.

Example: `exportblockattributes=(true|false)`

PdfNotesForRedactionBlocks

Default value: `false`

Description: Redact For publishing to PDF using the Blazon Redaction module only, when `true`, any redactions in the document will appear as PDF notes with the redaction reason code placed within the note. The underlying text is outlined but still visible for review.

Example: `pdfnotesforredactionblocks=(true|false)`

FitWithinBanners

Default value: `false`

Description: If `true`, the printed documents pages are scaled to fit within any banners that might be set.

Example: `fitwithinbanners=(true|false)`

ExpandToFitContent

Default value: `false`

Description: Use this parameter to expand page size to include source elements beyond the limits as defined by the source page. Where normal publishing would crop the images, `ExpandToFitContent=true` causes the page to expand to fit all content.

Example: `expandtofitcontent=(true|false)`

CsfName

Default value: <*filename.extension*>: not set

Description: Use this parameter to specify the output filename with extension. If specified, the full path to the output file is a concatenation of the Target and *CsfName* parameters. If omitted, the output file name is the source file name with a new extension based on output format. You must specify the file extension when using this parameter, and it must be the same as *OutputFormat*.

Example: `csfname=document.pdf`

ExportPageList

Default value: <*page list*>: not set

Description: Used to request export of individual pages and page ranges to publish to the requested target format. Page values are comma separated, zero-based, positive integers. Page ranges are indicated by two integers separated by a hyphen, where the integer on the left must be smaller than the one on the right. Leading and trailing whitespace and whitespace between values is legal. Explicit legal integer values must appear between every comma in the list. Any characters not described here appearing in the list, or any values not conforming to these constraints, results in a failed job. If not set, all available pages are published.

Usage notes:

- If multiple source documents are provided and merged into a single document, the page values provided in this parameter refer to valid pages of the final merged document. For example, if three 10-page documents are provided with only the first 3 pages requested for each document, the final merged document will have only 9 pages (rather than 30). Therefore, in this example, values specified in *ExportPageList* can only refer to pages numbered 0-9. Anything larger is out of range.
- If none of the values in the provided list refer to a valid page available for publishing, then the job is failed. If some values are valid while others are not, then a warning is issued indicating that some values are out of range, and only the valid page values are published.
- If multiple source documents are provided, but are not merged into a single document, then this list is applied equally to each resulting published document. If any one document has no pages that are within range of the requested page list, then the job is failed. If some documents have some out of range values but at least one other value from this list that are within range, then a warning is issued.

Example: `ExportPageList=0,0,15-19,1102`

This page list produces a document with 8 pages consisting of the 1st page repeated twice, followed by the 16th - 20th pages, followed by the 1,103rd page.

`ExportPageList=1,000`

This page list produces a document with 2 pages consisting of the 2nd page followed by the 1st page.

Rotate90

Default value: *<empty list>*

Description: Page list of output pages to rotate counterclockwise by 90 degrees during export. Only applies to PDF, TIFF, and thumbnail export.

Example: `rotate90=0,2,5-10`

Rotate180

Default value: *<empty list>*

Description: Page list of output pages to rotate counterclockwise by 180 degrees during export. Only applies to PDF, TIFF, and thumbnail export.

Example: `rotate180=0,2,5-10`

Rotate270

Default value: *<empty list>*

Description: Page list of output pages to rotate counterclockwise by 270 degrees during export. Only applies to PDF, TIFF, and thumbnail export.

Example: `rotate270=0,2,5-10`

PublishBlankPages

Default value: `true`

Description: If set to `false`, source pages that have no graphic or text data on them will not be included in the published document. The default behavior if this parameter is not set is to include blank pages.

Note: This parameter will not be honored if *`ExportPageList`* is specified.

Example: `publishblankpages=(true|false)`

PadOddPageDocuments

Default value: `false`

Description: Used for multi-source jobs, when this parameter is set to `true`, a blank page is added at the end of each document if that document has an odd number of pages. This can be used with duplex printing to ensure that the first page of each new document appears on a front-facing page.

If this property is set to `true`, and `PublishBlankPages=false`, the *`PublishBlankPages`* property is ignored, and blank pages are published.

Example: `padoddpagedocuments=(true|false)`

Export2Up

Default value: `false`

Description: When `true`, and the output format is PDF or TIFF, the Job Processor will export pages from the source document two at a time for each output document page (similar to printing 2 pages for each sheet). For example, with a 3 page source document, the output document will have 2 total pages. The first page will have page 1 and page 2 of the source document side by side

with page 1 on the left and page 2 on the right. The second page will have page 3 of the original source document on the left, and a blank page on the right.

Example: `export2up=(true|false)`

Export2UpLeadingBlankPage

Default value: `false`

Description: When `true` and *`Export2Up`* is also `true`, when the output format is PDF or TIFF, the Job Processor inserts a blank page into the output file before exporting pages from the source documents. For example, with a 3 page source document, the output document will have 2 total pages. The first page will be blank on the left side and contain the first page of the source document on the right side. The second page will contain pages 2 and 3 of the source document.

Example: `export2upleadingblankpage=(true|false)`

StepTimeout

Default value: `600`

Description: The maximum number of seconds allowed for any one step of the conversion process. It corresponds to the maximum amount of time allowed between heartbeat messages. The heartbeat is NOT required to be enabled in order to use this parameter. This value might need to be increased if you are publishing very large documents using CSF Writer publishing. The default can be set for all jobs using the `jpconsole.exe.config` file and `jpservice.exe.config` file, using the *`DefaultStepTimeoutInSeconds`* value. The value of *`StepTimeout`* in each job always overrides the value in the config file.

Example: `steptimeout=`*`<number of seconds>`*

HeartBeat

Default value: *<path to file>*: not set

Description: The value of this parameter is the full path to the file name. The path is not automatically created, and if not specified, heartbeat is disabled. When specified on a job, the Job Processor periodically writes to the specified file at points during the publish process. The file can be a unique file for each job (recommended - a file in the Target directory is a good choice here). Multiple jobs can use the same file, but they might incur performance penalties when trying to write to the same file simultaneously.

Each line in the file is formatted as:

An example line:

```
[08/30/17 03:15:41] [1ST FLOOR PLAN.DWG] Starting Job
```

```
[<Date Time>] [<Source File Name>] Msg
```

The file is encoded UTF-8, and the Job Processor must have access to the file. UNC paths are recommended.

Example: `heartbeat=\\share\directory\heartbeat file`

Following is the list of possible messaging information that might appear in a job:

- Starting Job

- Auto recognizing File <<source file>>
- Loading file <<source file full path>>
- Applying redaction script Redact
- Applying redaction zones Redact
- Applying markup <<markup file name>>
- Applying Banners and Watermarks
- Exporting as <<export type>>
- Generating text dump
- Publishing Complete
- Saving markups as file
- Generating thumbnails
- Job has an error
- Copying results
- Notifying <<URL>>
- Job Done
- Notification Response <<response from notification URL>>
- Final job parameters <<key pairs, one for each line>>
- Critical error in job <<error and stack information>>

StampImage

Default value: <*path to file*>: not set

Description: The **Stamp** feature allows users to specify a JPG or PNG image to apply as a stamp to all or certain pages of a 2D file when converted. This parameter contains the Job Processor accessible path to the image file to stamp. It must be a UNC path to a JPG or PNG file.

Example: `StampImage=\\mymachine\sampleimages\completed.png`

StampPositionAndSize

Default value: <*left*|*top*|*scale*|*page*[|*page*][|*page*]...[|*page*]>: not set

Description: This parameter is used in conjunction with *`StampImage`* and contains multiple values, separated by the pipe character (|), that specify the size and position of the image file (stamp) specified by the value of *`StampImage`*, and on which pages it will be applied.

The string parameter format is:

`Left`
Position of the image, as a percentage of page width.

`Top`
Position of the image, as a percentage of page height.

Scale
Size of the image, as a percentage of page width. This means that the image is resized to take up scale percentage of scale width. -1 can be specified to tell the system not to scale at all.

Page
Page on which to apply the stamp. It can be -1 to apply to all pages, or a sequence of page numbers, pipe character (|) separated and one-based on which to apply the stamp. To apply the stamp to only the first page, set this value to 1

Example:

```
Source=\\mymachine\files\DWG\layout.dwg
target=C:\temp\PublishedOutput
Outputformat=pdf
StampImage=\\mymachine\sampleimages\completed.png
StampPositionAndSize=10|10|-1|-1
notificationurl=http://mymachine:9999
```

This example would apply the image `completed.png` to all pages of the published PDF document, starting 10% of the width from the left of the page and 10% of the page height down from the top of the page. On each page, the image would not be scaled.

StampTemplateFilename

Default value: <*path to file*>: not set

Description: The **Stamp Template** feature allows users to specify a Stamp Template (created in Brava!) to apply to all or some pages of a 2D file when converted. This parameter contains the Job Processor accessible path to the Stamp Template file. It must be a UNC path to a Stamp Template file (XSP).

Example: `StampTemplateFileName=\\mymachine\samplefiles\mystamp.xsp`

StampTemplatePositionAndSize

Default value: <*left*|*top*|*scale*|*page*[|*page*][|*page*]...[|*page*]>: not set

Description: This parameter, used in conjunction with *`StampTemplateFilename`*, contains multiple values, separated by the pipe character (|), that specify the size and position of the stamp template specified by the value of *`StampTemplateFilename`*, and on which pages it will be applied

See *`StampPositionAndSize`* for the string parameter format descriptions.

Example:

```
Source=\\mymachine\files\DWG\layout.dwg
target=C:\temp\PublishedOutput
Outputformat=pdf
StampTemplateFileName=\\mymachine\samplefiles\mystamp.xsp
StampTemplatePositionAndSize=10|10|-1|-1
notificationurl=http://mymachine:9999
```

This example would apply the template `mystamp.xsp` to all pages of the published PDF document, starting 10% of the width from the left of the page and 10% of the page height down from the top of the page. On each page, the image would not be scaled. This feature is useful for applying stamp templates to a large volume of documents of the same type and size (for example, 8.5 x 11 PDF files, or same-size CAD files)

GetText

Default value: `false`

Description: A value of `true` instructs the server to produce a set of text files with all of the text from the published document extracted as plain text. This parameter is useful for indexing. If the *`ExportPageList`* value is specified, both values are honored and only pages listed in *`ExportPageList`* will have their text extracted.

Caution

All output will be named according to the `pagen.txt` naming schema, which could potentially result in overwriting existing files of the same name.

Example: `gettext=(true|false)`

TextFormatUtf8

Default value: `false`

Description: If set to `true`, then the output format produced from `GetText` is UTF-8 encoded. If omitted or set to `false`, the output format is UTF-16 encoded.

Example: `textformatutf8=(true|false)`

TextFormatClrf

Default value: `false`: not set

Description: By default, text files extracted by the Job Processor (when `gettext=true`) have their lines delimited by a single line feed (Unix style). If you want to have each line separated by a carriage return/line feed pair (DOS/Windows style), you can instruct the Job Processor to do so by adding the parameter `textformatcrlf=true` to the job.

Example: `textformatcrlf=(true|false)`

TextSortMode

Default value: `columns`

Description: Selects how text is arranged during source document loads. Valid values are:

`columns`: (default) Attempts to detect paragraphs that are arranged in logical columns and keep that text together. The relative arrangement of text affects text search and the order of text in text extraction by the *`GetText`* job option.

`topdown`: Attempts to arrange text in order from top to bottom of the page. This sorting causes column-based text to interleave lines from each column. For

tabular data, it keeps each row together in order instead of keeping each column together.

`none`: Keeps text in the order it was arranged in the source file, which might not be uniform in columns or topdown order.

Example: `textsortmode=(columns|topdown|none)`

TextFilePrefix

Default value: <*string*>: not set

Description: For use with *`GetText`*, use this parameter to specify a non-default prefix for the text or text with position output files. Each string might contain the **case sensitive** substitution string `${source}`. If that substitution string is present, it will be replaced with the source file's name (without the extension).

Examples: `textfileprefix=abcd`

produces text file output named `abcd0.txt`, `abcd1.txt`, `abcd2.txt`, and so on.

`textfileprefix=${source}-txt`

used with a source file called `fish.pdf`, produces text files named `fish-txt0.txt`, `fish-txt1.txt`, and so on.

DocumentTitle

Default value: <*string*>: not set

Description: Specifies the title of the document, used when the `%title` macro is specified for an ISOBanner in the XML provided to the *`BannerXmlFilename`* job option.

Examples: `documenttitle=mytitle`

AddMetadataToTextOutput

Default value: `false`: not set

Description: If both this and the *`GetText`* parameter are set to `true`, then any metadata present in the source file is included in the text output pages.

The values are inserted in the first page's text output and are available from DWG, PDF, DOCX, and XPS files only.

Metadata can include, but is not limited to:

```
Date created
Author of file
Date modified
Last updated
Title
Object
```

Examples: `addmetadatatotextoutput=(true|false)`

ExtractExIfData

Default value: `false`: not set

Description: This parameter adds the ability to output Exif data from JPG files (only) into the metadata output. If `true`, *`AddMetadataToTextOutput`* must also be `true`. Support for additional file types is planned for future releases.

Examples: `extractexifdata=(true|false)`

ExportFilePerPage

Default value: `false`: not set

Description: Blazon Enterprise can export a single file for each page in the source document if this parameter is `true`.

These pages naming format will be FILENAME_PAGE_NN.EXPORTTYPE. This feature is available for PDF and TIFF Export ONLY. For example, if you have an N page document, called `building.pdf`, then the output will be `building_page0.pdf`, `building_page1.pdf`, ... `building_pageN-1.pdf`, and so on.

When exporting to single file for each page, the total page count macro, the Bates number, and the page number macro will increment with each file. Therefore, file one will be page 1, file two page 2, and so on. Total pages will be set based on total pages in the export request (which equals all pages in the source file unless you are only exporting selected pages). For example, when publishing pages 2 and 3 of a six-page PDF file, one file for each page, the total page count will be 2 at export time.

Examples: `exportfileperpage=(true|false)`

ExportPageExtents

Default value: `false`: not set

Description: When `true`, the Job Processor will export a separate page extents file, for each page of the document, to the target folder. These files will be named `extents-page-<page number>.txt` by default. To customize these names, see the next parameter, *`ExportPageExtentsFile`*. The file for the first page will contain an additional header line describing each of the values.

Note: If the *`ExportPageList`* parameter is specified, then only the extents for those pages will be published into the file and available for export.

Examples: `exportpageextents=(true|false)`

ExportPageExtentsFile

Default value: `extents`

Description: When specified, the files produced by `exportpageextents=true` will be named *`<job option value>`*`-page-`*`<page number>`*`.txt` .

Note: If the *`ExportPageList`* parameter is specified, then only the extents for those pages are published into the file and available for export.

Examples: `exportpageextentsfile=`*`<file prefix>`*

PublishUsername

Default value: <*string*>: not set

Description: Specifies the user name who is publishing a file, used when the `%user` or `%login` macro is specified for an `ISOBanner` in the XML provided to *`BannerXmlFilename`* job option.

Examples: `publishusername=jdoe`

AllowFontSubstitution

Default value: `true`

Description: When `false`, any detected missing font that would normally be substituted with a similar font will instead cause the job to fail. Note that for certain loaders this detection process can be time-consuming and significantly slow down processing. When `true`, fonts used in a document that are not installed on the publishing machine (typically in `C:\Windows\Fonts\`) will be replaced by a similar font.

Examples: `allowfontsubstitution=(true|false)`

ExportContentAlignment

Default value: `Center`

Description: Used for source documents where dimensions might have unwanted white space in one or more areas when placed in the center of the target document's page. This parameter adjusts the placement of the content within the target page, anchoring it at one of the 9 possible locations.

This option is only supported for export to PDF and TIFF. If not set, defaults to `Center`.

Examples: `exportcontentalignment=TopLeft|TopCenter|TopRight|CenterLeft|Center|CenterRight|BottomLeft|BottomCenter|BottomRight)`

ExportScaleContentMode

Default value: `ScaleToFitPage`

Description: Used for source documents where more control over the layout of content within PDF or TIFF document pages is desired, this option selects one of 3 possible content scaling modes:

`ScaleToFitPage`
By default, the Job Processor scales all content to fit within the target document dimensions while retaining aspect ratio of the source. This corresponds to the `ScaleToFitPage` mode.

`ScaleToFitMargins`
This mode, like `ScaleToFitPage`, is an aspect-ratio-preserving mode that scales the content to fit the target page dimensions, but with an additional margin of whitespace around the original content. If `ScaleToFitMargins` is selected, additional parameters are required (see the following `ExportScaleMarginUnits` and `ExportScaleMarginTop/Bottom/Left/Right`).

`ScaleToValue`
This mode is also aspect-ratio-preserving, but scales the content up by a specified percentage of either the height or width of the content's original dimensions. If `ScaleToValue` is selected, additional parameters are required (see *`ExportScalePercentage`*).

This parameter only applies for export to PDF and TIFF formats. If not set, defaults to `ScaleToFitPage`.

Examples: `exportscalecontentmode=(ScaleToFitPage|ScaleToFitMargins|ScaleToValue)`

ExportScalePercentage
Default value: `1.0`

Description: If *`ExportScaleContentMode`* is set to `ScaleToValue`, and this parameter is set to a positive floating point number greater than 0.0, the Job Processor will scale content by this factor, preserving the aspect ratio by scaling both the height and width by the same percentage (1.0 leaves the original dimensions unchanged).

If a target page size is specified, and the custom-scaled content does not fit in the target page dimensions, clipping will occur. This parameter only applies for export to PDF and TIFF formats. If not set, default is 1.0.

Example: `exportscalepercentage=1.0`

ExportScaleMarginUnits
Default value: `in`

Description: If *`ExportScaleContentMode`* is set to `ScaleToFitMargins`, and this parameter is set to one of the legal values, then the values specified for `ExportScaleMarginTop/Bottom/Left/Right` will be taken as quantities of these units. cm = centimeters, in = inches, mm = millimeters, pt = typographic points, um = micrometers. If not set, default is `in`.

Examples: `exportscalemarginunits=(cm|in|mm|pt|um)`

ExportScaleMarginTop
Default value: `0`

Description: If *`ExportScaleContentMode`* is set to `ScaleToFitMargins`, and this parameter is set to a positive floating point number greater than or equal to 0.0, the Job Processor adds this amount of margin between the top edge of the target document page and the top edge of the original content. If not set, default is `0`.

Because `ScaleToFitMargins` is an aspect-ratio-preserving scaling operation, some margin widths can be larger than requested (but never smaller).

This parameter only applies for export to PDF and TIFF formats.

Examples: `exportscalemargintop=(positive floating point number>=0.0)`

ExportScaleMarginLeft
Default value: `0`

Description: If *`ExportScaleContentMode`* is set to `ScaleToFitMargins`, and this parameter is set to a positive floating point number greater than or equal to 0.0, the Job Processor adds this amount of margin between the left edge of the target document page and the left edge of the original content.

Because *`ScaleToFitMargins`* is an aspect-ratio-preserving scaling operation, some margin widths can be larger than requested (but never smaller).

This parameter only applies for export to PDF and TIFF formats.

Examples: `exportscalemarginleft=(positive floating point number>=0.0)`

ExportScaleMarginRight
Default value: `0`

Description: If *ExportScaleContentMode* is set to `ScaleToFitMargins`, and this parameter is set to a positive floating point number greater than or equal to 0.0, the Job Processor adds this amount of margin between the right edge of the target document page and the right edge of the original content.

Because *ScaleToFitMargins* is an aspect-ratio-preserving scaling operation, some margin widths can be larger than requested (but never smaller).

This parameter only applies for export to PDF and TIFF formats.

Examples: `exportscalemarginright=(positive floating point number>=0.0)`

ExportScaleMarginBottom

Default value: `0`

Description: If *ExportScaleContentMode* is set to `ScaleToFitMargins`, and this parameter is set to a positive floating point number greater than or equal to 0.0, the Job Processor adds this amount of margin between the bottom edge of the target document page and the bottom edge of the original content.

Because `ScaleToFitMargins` is an aspect-ratio-preserving scaling operation, some margin widths can be larger than requested (but never smaller).

This parameter only applies for export to PDF and TIFF formats.

Examples: `exportscalemarginbottom=(positive floating point number>=0.0)`

ForceTiffMonochrome

Default value: `false`

Description: If set to `true`, all markups, vectors, and rasters are rendered monochrome in the output TIFF file. If `true`, this parameter value overrides the value set by *OutputColorMode*.

Examples: `forcetiffmonochrome=(true|false)`

OutputColorMode

Default value: not set

Description: When set, all content in the document (rasters, vectors, and markups) are set to the chosen color mode. This parameter does not affect banner color, which can be set through the XML provided to the *BannerXmlFilename* job option as appropriate. Valid values are as follows:

`fullcolor`
All colors for all entities are exported as seen in the source documents and markups.

`grayscale`
or **greyscale**: All colors for text, rasters, vectors, and markups are rendered in 256 shades of grey.

`monochrome`
All text, rasters, vectors, and markups are rendered in 2-bits, either black or white.

Examples: `outputcolormode=(fullcolor|grayscale|monochrome)`

Note: If the output format is TIFF, and the color mode is set to `monochrome`, then the output compression type will be CCITT Group 4 (FAX).

CSF output only supports `fullcolor` mode. A warning message displays if set to any other value when publishing as CSF.

ExportLayerState

Default value: not set

Description: This parameter determines the layer state for exporting all document types. Valid values are:

`All`
Export all layers to PDF, CSF, or XDL (or visible layers flattened for TIFF).

`Visible`
Export all visible layers only to PDF, CSF, or XDL (or flattened for TIFF).

`None`
Export only visible layers, flattened.

If the original document does not contain layers, this option has no effect and a warning message displays. TIFF documents do not support layers and are recognized as single layer visible or flattened documents. When exporting any format to TIFF, you only get the visible layers flattened as output, regardless of this option setting.

Examples: `exportlayerstate=(all|visible|none)`

SetLayers.Hidden

Default value: not set; accepts a regular expression

Description: Any layers in the exported document with names matching the regular expression are set to hidden. This option is applied before the *`SetLayers.Visible`* option, so layers hidden by this option can still be set back to visible by the *`SetLayers.Visible`* option if also provided. This option is applied after the *`SetLayers.Toggled`* option.

Note: The *`SetLayers.Hidden`*, *`SetLayers.Visible`*, and *`SetLayers.Toggled`* parameters use the C++11 regex engine using ECMAScript grammar for matching.

Examples:

```
setlayers.hidden=(.)*CONSTRUCTION
```

This example hides all layers with names that end with “CONSTRUCTION”.

```
setlayers.hidden=(TEMP)|(WORK(.)*)
```

This example hides the layer named “TEMP” and all layers starting with “WORK”.

SetLayers.Visible

Default value: not set; accepts a regular expression

Description: Any layers in the exported document with names matching the regular expression are set to visible. This option is applied after the

SetLayers.Hidden option. This option is applied after the *SetLayers.Toggled* option.

Example:

```
setlayers.visible=(.)*CONSTRUCTION
```

This example shows all layers with names that end with "CONSTRUCTION".

SetLayers.Toggled

Default value: not set; accepts a regular expression

Description: Any layers in the exported document with names matching the regular expression are switched to visible or hidden, depending on their initial state in the file. Note that this option is applied before the *SetLayers.Hidden* and *SetLayers.Visible* job options, so layers modified by this option can be modified again if either of these two job options are provided.

Example:

```
setlayers.toggled=(.)*CONSTRUCTION
```

This example switches the initial visibility state of all layers with names that end with "CONSTRUCTION" to visible or hidden.

OutputPageSize

Default value: not set

Description: You can set the page size by specifying the *OutputPageSize* for output of type PDF and TIFF. It is highly recommended to set this parameter when *OutputFormat* is set to `TIFF` or if markups are included in the publishing request to avoid creating a huge output file size. Refer to "Page size settings" on page 144.

Performance recommendation - when publishing CAD files (and other file types with very large page sizes) to TIFF, the output file can become quite large, since the page dimensions can be very large. Therefore, if you are publishing CAD files and are concerned with file size, it is strongly recommended that you set the page size to control file size.

Examples: `outputpagesize=(ISO A1 Portrait|Letter|ISO A4 Landscape|<other page size from table>)`

OutputPageSize.UseSourcePageOrientation

Default value: `false`

Description: When `false`, the dimensions set by *OutputPageSize* are applied to every page. If the size is set to `Letter`, which is 8.5 wide by 11 tall, and a source document page is wider than it is tall, the content of the source page will be scaled to fit into the 8.5 page width, with substantial blank space above and below on the page. When `true`, the dimensions set by *OutputPageSize* are applied as a size to use, but are oriented to match each source page. If the size is set to `Letter`, and a source document page is wider than it is tall, the resulting output size for that page will be 11 wide by 8.5 tall.

Notes

- When set to `true`, certain *`OutputPageSize`* values are redundant. For example, `Letter` and `LetterLandscape` will behave the same.
- When set to `true`, the job options *`PdfExportRotateToFit`* and *`TiffExportRotateToFit`* cannot be used and will cause the job to error.

Example: `outputpagesize.usesourcepageorientation=(true|false)`

CreatePageSizesTextFile

Default value: `false`

Description: For use with **XDL output** only, this parameter allows users to create a `PageSizes.txt` file. If set to `true`, then `PageSizes.txt` is created in the file's output directory.

Examples: `createpagesizestextfile=(true|false)`

CreatePageSizesTextMetric

Default value: `false`

Description: For use with **XDL output** only, this parameter determines the type of measurement information displayed in the `PageSizes.txt` file (when *`createpagesizestextfile`* is set to `true`). If this parameter is set to `true`, then the system outputs metric page size information.

Examples: `createpagesizestextmetric=(true|false)`

Timeout

Default value: `1200`

Description: The number of seconds to allow a job to process. The default value is 1200 (20 minutes). If a job takes longer than this set time to publish, processing is terminated and a time out error is returned.

Examples: `timeout=`*`<number of seconds>`*

PersistTimeout

Default value: `10`

Description: The amount of time (in seconds) a persistent document will be kept in memory before being recycled.

Example: `persisttimeout=`*`<# of seconds>`*

MaxMessageCount

Default value: `50`

Description: The maximum number of warnings or error messages written to the log file for each job.

Example: `maxmessagecount=`*`<# of messages>`*

ShowLineWeights

Default value: `false`

Description: When set to `true`, AutoCAD **Print Line Weights** are supported when publishing to PDF or TIFF.

Examples: `showlineweights=(true|false)`

ShowThinLinesOnly

Default value: `false`

Description: Use `true` to show only thin lines.

Examples: `showthinlinesonly=(true|false)`

ColorCollisionTolerance

Default value: `10`

Description: Sets the tolerance level for color collision on PDF output files only. Range value is 0 (least sensitive) to 255 (most sensitive). A lower number preserves more colors in the output while a value of 255 renders a black and white PDF

Examples: `ColorCollisionTolerance=<0-255>`

Priority

Default value: not set

Description: Priority indicates the order that jobs are given to a requested Job Processor thread.

This optional parameter is a string representation of a 32-bit signed integer. Jobs are sent to individual queues by their extension (DRW, DOC, PDF), and within those queues, they are ordered by the integer value set by Priority (priority is maintained for each queue, which will follow its own prioritized list of jobs). A higher number receives higher priority and equivalent numbers will be inserted in order of first in, first out. Jobs that do not have a priority value set will be processed after all jobs in the queue that have a priority value are processed.

The range of properties is any integer from -1000 to 1000, inclusive. If the range is out of bounds of -1000 to 1000, it is adjusted to the closest value. If the range is invalid (cannot be parsed as an integer), then the priority is ignored. In both cases, the server adds a warning to the job.

Examples: `priority=<-1000-1000>`

If 5 jobs are inserted into the DRW queue, for example, using the following calls in the following order:

```
push.aspx?name=job1,...,priority=10,...
push.aspx?name=job2,...,
push.aspx?name=job3,...,priority=0,...
push.aspx?name=job4,...,priority=10,...
push.aspx?name=job5,...,priority=-5,...
```

Then the jobs will be served up to the JPs in the following order:

```
Job1, Job4, Job3, Job5, job2
```

Be aware that jobs might not finish in that order if multiple Job Processors are running simultaneously.

CtbFile

Default value: not set

Description: Full path to the custom CTB file to use when publishing AutoCAD files.

To enable this feature, you must edit the loader configuration file (launch `loaders.configuration.exe` from your Blazon Enterprise loaders directory, typically …`Blazon Enterprise\JobProcessor\Igc.Loaders`). In the `Dwg2dl` section, change the value of the parameter *`UsePlotStyleColors`* to `true`.

Driver file `DWG2DL.dll` version 1.5.9.1 or higher is required to use this feature.

Examples: `ctbfile=`*`<path to file>`*

IntlOutputStringFile

Default value: not set (see `sample_intloutputstrings.txt`

Description: This parameter allows each job to override the strings used by the Job Processor during markup burn-in and export.

A sample file is provided in the installed Job Processor directory (called `sample_intloutputstrings.txt`) that can be modified or referenced. Refer to this sample file for a list of the default values for these strings.

You can override these parameters by creating a text file (using UTF-16 Little Endian byte order). Each line in the file can contain a definition. Each line must be one of the listed keys (such as MARKUP_GHOSTTEXT or CMCOPY_AT) followed by an equal sign, and then the string to use. The string to use cannot contain an equal sign. The file's first two bytes must contain a UTF-16LE byte order mark (0xFFFE).

Examples: `intloutputstringfile=`*`<path to file>`*

6.1.2 PDF input parameters

PdfExportA1b

Default value: `false`

Description: If `true`, then the PDF output will be PDF/A-1b archive compatible. It will be a larger file if it contains many rasters or transparencies.

> **Important**
>
> This job option and the following `PdfExportA*` and `PdfExportE` options are used with both the Job Processor and Linux Publishing Agent publishers. These options are mutually exclusive and cannot be combined in the same Blazon job call.

Examples: `pdfexporta1b=(true|false)`

PdfExportA2b

Default value: `false`

Description: If `true`, then the PDF output will be PDF/A-2b archive compatible.

Examples: `pdfexporta2b=(true|false)`

PdfExportA2u

Default value: `false`

Description: If `true`, then the PDF output will be PDF/A-2u archive compatible.

Examples: `pdfexporta2u=(true|false)`

PdfExportA3b

Default value: `false`

Description: If `true`, then the PDF output will be PDF/A-3b archive compatible.

Examples: `pdfexporta3b=(true|false)`

PdfExportA3u

Default value: `false`

Description: If `true`, then the PDF output will be PDF/A-3u archive compatible.

Examples: `pdfexporta3u=(true|false)`

PdfExportE

Default value: `false`

Description: If `true`, then the PDF output will be PDF/E.

Examples: `pdfexporte=(true|false)`

PdfExportA1aTag

Default value: `false`

Description: If `true`, a PDF/A-1a compliance tag is inserted in the PDF output file.

Note: The inserted tag indicates that the file is intended to be compliant with the PDF/A-[#]a standard. Although the presence of the tag will cause most PDF validators to recognize the file as a PDF/A-[#]a file, full compliance to the standard cannot be achieved by an automated conversion process. To be fully compliant, the generated output file would require editing in a PDF editor that supports the PDF/A-[#]a specification and have metadata manually inserted to meet the standard's requirements.

Examples: `pdfexporta1atag=(true|false)`

PdfExportA2aTag

Default value: `false`

Description: If `true`, a PDF/A-2a compliance tag is inserted in the PDF output file. See *`PdfExportA1aTag`* note.

Examples: `pdfexporta2atag=(true|false)`

PdfExportA3aTag

Default value: `false`

Description: If `true`, a PDF/A-3a compliance tag is inserted in the PDF output file. See *`PdfExportA1aTag`* note.

Examples: `pdfexporta3atag=(true|false)`

PdfFastWebView

Default value: `false`

Description: When `true`, Fast Web View is enabled. When `false` or omitted, Fast Web View is disabled.

Examples: `pdffastwebview=(true|false)`

PdfExportMarkupsAsAnnotations
Default value: `false`

Description: If `true`, markups on PDF output files are saved as PDF Annotations (note exceptions to follow). If `false`, markups are burned into the PDF output and cannot be removed by users.

Redact **Redaction Details**: Redaction markup entities are ALWAYS burned in by the server. This parameter (used with the Blazon Redaction module) affects the behavior of other markup entities when publishing to PDF. When `true`, entities other than redactions are saved as PDF Annotations and can be edited in Adobe Acrobat (but not Reader). When `false`, all Markup entities (redactions included) are "burned in" to the PDF and not saved as Annotations, so they can't be edited in Acrobat.

Changemarks Details: Regardless of the state of this flag, Changemark entities are ALWAYS annotations in PDF output files and Redaction entities are ALWAYS burned in and cannot be edited.

Example: `pdfexportmarkupsasannotations=(true|false)`

PdfExportRotateToFit
Default value: `false`

Description: If true, indicates that the pages of exported PDF files should be rotated to a consistent orientation. See *`PdfExportRotateToFit.Orientation`* for details.

Example: `pdfexportrotatetofit=(true|false)`

PdfExportRotateToFit.Direction
Default value: `ccw`

Description: Only used when *`PdfExportRotateToFit`* is set to `true`. This option controls the direction of rotation when a page is rotated. The default is a 90 degree rotation counter-clockwise (`ccw`). To change to a 90 degree rotation clockwise, specify `cw`.

Example: `pdfexportrotatetofit.direction=(ccw|cw)`

PdfExportRotateToFit.Orientation
Default value: not set

Description: Only used when *`PdfExportRotateToFit`* is set to `true`. This option controls the orientation that pages are rotated to agree with, either portrait or landscape, if they are not already in that orientation. When this value is not set, the resulting pages are portrait orientation, unless an *`OutputPageSize`* has been set that is wider than it is tall. In that case, the default is landscape. If a value is specified, that value is always used. When `portrait` is set, pages that are wider than they are tall are rotated 90 degrees during export. When `landscape` is set, pages that are taller than they are wide are rotated instead.

Example: `pdfexportrotatetofit.orientation=(portrait|landscape)`

PdfExportBookmarksInheritZoom
Default value: `false`

Description: When set to `true`, Adobe uses the zoom scale of the current page when switching to the bookmarked page.

Examples: `pdfexportbookmarksinheritzoom=(true|false)`

PdfExportBookmarkPanelState
Default value: `default`

Description: Sets the initial state of the bookmark panel in an exported PDF file. If set to `default` or if the job option is omitted, the bookmark panel is open if there are bookmarks in the file, otherwise the panel is closed. If set to `open`, the bookmark panel is open in the exported PDF. If set to `closed`, the bookmark panel is closed in the exported PDF.

Note: This parameter controls only the initial state of the bookmark panel. It can still be toggled open or closed after opening the file in a viewer that displays the panel. See also: *`PdfExportUseSourceBookmarkPanelState`*

Example: `pdfexportbookmarkpanelstate=(default|open|closed)`

PdfExportUseSourceBookmarkPanelState
Default value: `false`

Description: If this job option is provided and set to `true`, and the source file is a PDF, and the source PDF has set the bookmark panel initial state, that state will be respected in the exported PDF (either open or closed, as set). This case will override the setting from *`PdfExportBookmarkPanelState`*. If any of these conditions is not met, then the setting from *`PdfExportBookmarkPanelState`* applies.

Note: A PDF file might not set the panel state at all, but if it contains bookmarks, it will often open with the bookmark panel initially open. This type of file would not meet the conditions for this job option. The panel state must be specifically set in the file to be retained in the exported PDF

Example: `pdfexportusesourcebookmarkpanelstate=(true|false)`

PdfExportBookmarksExpandLevel
Default value: `0`

Description: Sets the level of the bookmark tree that will be expanded during PDF export. The default value, 0, indicates no expansion of the tree (top-level bookmarks will be visible). Setting a level will cause that number of levels to be expanded (Setting the value of 1 will cause all top-level and first-level bookmarks to be visible in the list upon opening the file).

Note: This does not affect the ability of the reader to expand or close bookmark levels while reading the file. This setting only controls the starting expansion upon opening the PDF file.

Example: `pdfexportbookmarksexpandlevel=3`

PdfExport.Magnification

Default value: `Default`

Description: This option allows integrators to customize the PDF Magnification setting for PDF export.

Examples: `pdfexport.magnification=Default`

Job option setting value	Adobe PDF properties value
Default	Default
ActualSize	Actual Size
FitPage	Fit Page
FitWidth	Fit Width
FitHeight	Fit Height
FitVisible	Fit Visible
One of: 25Percent | 50Percent | 75Percent | 100Percent | 125Percent | 150Percent | 200Percent | 400Percent | 800Percent | 1600Percent | 2400Percent | 3200Percent | 6400Percent	Magnification percentage (25% to 6400%)

Note: If you require magnification to persist when PDF bookmarks are clicked, set `PdfExportBookmarksInheritZoom=true`.

PDFExportAdjustLineWeightsForOutputSize

Default value: `false`

Description: CAD documents can set line weights in physical sizes. When exporting to PDF at a different page size than the original source size, these physical sizes will cause line widths that look out of proportion to the rest of the document. Setting this job option to `true` will cause the line weights to be scaled proportionally to the document scale, preserving the appearance of the full-size document. See also *`OutputPageSize`*.

Example: `pdfexportadjustlineweightsforoutputsize=(true|false)`

PdfExportDisablePrint

Default value: `false`

Description: When `true`, printing is disabled in an exported PDF file.

Requires setting a security password with *`PdfExportSecurityPassword`*.

Examples: `pdfexportdisableprint=(true|false)`

PdfExportDisableChange

Default value: `false`

Description: When `true`, content modification disabled in an exported PDF file.

Some types of modification might still be allowed unless *`PdfExportDisableAnnotation`*, *`PdfExportDisableFillFormFields`*, and

PdfExportDisableDocumentAssembly are also set `true`. Requires setting a security password with *PdfExportSecurityPassword*.

Examples: `pdfexportdisablechange=(true|false)`

PdfExportDisableSelectCopy
Default value: `false`

Description: When `true`, content copying is disabled in an exported PDF file.

Some types of content extraction might still be allowed for accessibility software unless *PdfExportDisableAccessibility* is also set `true`. Requires setting a security password with *PdfExportSecurityPassword*.

Examples: `pdfexportdisableselectcopy=(true|false)`

PdfExportDisableAnnotation
Default value: `false`

Description: When `true`, adding or modifying text annotations (also known as Commenting) is disabled in an exported PDF file. Requires setting a security password with *PdfExportSecurityPassword*.

Examples: `pdfexportdisableannotation=(true|false)`

PdfExportDisableFillFormFields
Default value: `false`

Description: When `true`, filling in interactive form fields is disabled in an exported PDF file.

If set to `true`, *PdfExportDisableAnnotation* and *PdfExportDisableChange* must also be set to `true` to allow Adobe to honor disabling form fields. In addition, you are required to set a security password with *PdfExportSecurityPassword*.

To use this option, *PreserveFormFields* must be set to `true` (the default value) in the Pdf2dl configuration file to ensure forms field are added to any output. See *OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)* for details.

Examples: `pdfexportdisablefillformfields=(true|false)`

PdfExportDisableAccessibility
Default value: `false`

Description: When `true`, extracting text and graphics for use by accessibility programs is disabled in an exported PDF file. Requires setting a security password with *PdfExportSecurityPassword*.

Examples: `pdfexportdisableaccessibility=(true|false)`

PdfExportDisableDocumentAssembly
Default value: `false`

Description: When `true`, insertion, rotation, and deletion of pages, and creation of bookmarks and thumbnails is disabled in an exported PDF file. Requires setting a security password with *PdfExportSecurityPassword*.

Examples: `pdfexportdisabledocumentassembly=(true|false)`

PdfExportDisableHighResPrint

Default value: `false`

Description: When `true`, high resolution printing is disabled in an exported PDF file. Low quality printing will still be available unless *`PdfExportDisablePrint`* is also set `true`. Requires setting a security password with *`PdfExportSecurityPassword`*.

Examples: `pdfexportdisablehighresprint=(true|false)`

PdfExportDisableAll

Default value: `false`

Description: If `true`, requires *`PdfExportSecurityPassword`* to be set.

Setting to `true` is equivalent to setting *`PdfExportDisablePrint`*, *`PdfExportDisableChange`*, *`PdfExportDisableSelectCopy`*, *`PdfExportDisableAnnotation`*, *`PdfExportDisableFillFormFields`*, *`PdfExportDisableAccessibility`*, *`PdfExportDisableDocumentAssembly`*, and *`PdfExportDisableHighResPrint`* all to `true`.

Examples: `pdfexportdisableall=(true|false)`

PdfExportEncryptionStrength

Default value: `128`

Description: Set to either `40` or `128` to set encryption key bit length in Adobe Acrobat Reader. This parameter is ignored if no password is supplied. If a password is supplied, the parameter can be:

- Omitted, in which case 128 is used.
- Set to 128.
- Set to 40.
- Set to some other value, in which case 128 is used and a warning message displays.

Examples: `pdfexportencryptionstrength=(128|40)`

PdfExportOpenPassword

Default value: <*string*>: not set

Description: Sets a password that will be required to open an exported PDF file. This password is also known as a user password. An open password does not prevent the file from being modified after it has been opened. To require a password before changing restricted features, use *`PdfExportSecurityPassword`* and optionally one or more *`PdfExportDisable<>`* job options to restrict feature sets. If both PDF export passwords are supplied, they cannot be set to the same password, but the document can be opened using either the open password or the security password.

Examples: `pdfexportopenpassword=documentpassword`

PdfExportSecurityPassword

Default value: <*string*>: not set

Description: Sets a password value for changing security permissions in an exported PDF file. This password is also known as a permissions password or a master password. Setting a security password is required to use any of the PDF export security permissions job options. A security password is not required to open the exported PDF file. To require a password to open the exported PDF, use *PdfExportOpenPassword*. If both PDF export passwords are supplied, they cannot be set to the same password, but the document can be opened using either the open password or the security password.

Examples: `pdfexportsecuritypassword=secretpassword`

PdfExportVersion

Default value: `1.4`

Description: Sets the desired version of exported PDF files. If other requested features require a newer version than the version requested with this parameter then the resulting PDF file might be a newer version than the version requested.

Examples: `pdfexportversion=(1.4|1.5|1.6|1.7)`

For example, if a job requires a PDF be A1a compatible, the PdfExportVersion will be set internally to 1.4 and any pdfexportversion passed in will be ignored. If a job requires a PDF be A2b compliant and the job passed PDF version 1.6, the PDF version would not be honored because A2b compliance requires PDF version 1.7.

PdfExportMaxFileSize

Default value: `B`

Description: Specifies maximum size of an exported PDF file. If a job would result in a PDF file that is larger than the specified size, multiple PDF files will instead be created, named with a suffix of _1, _2, and so on. The default result, if not provided, is to allow arbitrarily large single files. This parameter value must be an integer number without separator punctuation, followed immediately by an optional unit suffix. Allowed suffixes are:

`B` – value is bytes (default if no suffix is provided)

`KB` – value is a multiple of 1,000 bytes

`MB` – value is a multiple of 1,000,000 bytes

`KIB` – value is a multiple of 1,024 bytes

`MIB` – value is a multiple of 1,048,576 bytes

Example: `pdfexportmaxfilesize=5MB`

To limit the file size to 50,000,000 bytes, for example, the parameter value could be specified as `50mb`, `50000kb`, or `50000000b`.

PdfExportEnforceRelativeHyperlinks

Default value: `false`

Description: This option allows users to abort publishing if any source documents contain absolute paths within hyperlinks. When set to `true`, and any absolute paths are found in hyperlinks, the job is failed.

Examples: `pdfexportenforcerelativehyperlinks=(true|false)`

PdfExportHyperlinkBorder

Default value: `false`

Description: Controls added decoration for hyperlinks in an exported PDF. If `true`, hyperlinks are surrounded by a visible rectangle. If `false`, hyperlinks are left unchanged.

Examples: `pdfexporthyperlinkborder=(true|false)`

PdfExportDisableLinkText

Default value: `none`

Description: Controls if text links in a source document are disabled when exported to PDF. Valid values are:

`none`
All existing text links are left active.

`internal`
Any text that links to pages within the document is disabled.

`external`
Any text that links to other documents is disabled.

`all`
Both types of text links are disabled.

Note: Most PDF viewers automatically create an external link out of text that is formatted as an address, such as a web address (www.server.com) or an email address (name@server.com), when displaying the PDF, regardless of whether the text in the PDF was exported as an external link.

Examples: `pdfexportdisablelinktext=(none|internal|external|all)`

PdfExportColorOverrideLinkText

Default value: `none`

Description: Controls if text links in a source document are exported in a different color when exported to PDF. Valid values are:

`none`
All text links are exported in the color used in the source document.

`internal`
Any text that were linked to pages within the document are exported in the override color (see *`PdfExportColorOverrideLinkText.Color`*).

`external`
Any text links that were links to other documents are drawn in the override color.

`all`
Both types of text links are exported to the override color.

Examples: `pdfexportcoloroverridelinktext=(none|internal|external|all)`

PdfExportColorOverrideLinkText.Color

Default value: `0,0,0` (black)

Description: Sets the override color to use with *`PdfExportColorOverrideLinkText`*. This match can be a comma separated RGB value or a named color. The color name comes from the .Net `Known Colors` enumeration, which is documented here: http://msdn2.microsoft.com/en-us/library/system.drawing.knowncolor.aspx

Examples: To set to full red, for example, use `pdfexportcoloroverridelinktext.color=255,0,0`

PdfExportCmyk

Default value: `false`

Description: Controls the color space used in exported PDF files. If true, the resulting PDF uses the CMYK color space. If `false`, the resulting PDF uses RGB encoding.

Examples: `pdfexportcmyk=(true|false)`

PdfExport.ForceImageCompression.Color

Default value: `default`

Description: When exporting documents to PDF, by default the system will choose an image compression algorithm for embedded images based on the content of the image. When possible, the image compression algorithm of the source image is retained. This job option allows overriding this behavior and forcing a particular image compression algorithm for color images.

`jpeg`
Forces color images to be compressed with the lossy jpeg algorithm.

`flate`
Forces color images to be compressed with the lossless flate (zip) algorithm.

`default`
Result in the default behavior of selecting or retaining a compression algorithm based on the source image (recommended for best fidelity).

Examples: `pdfexport.forceimagecompression.color=(default|jpeg|flate)`

PdfExport.ForceImageCompression.Monochrome

Default value: `default`

Description: When exporting documents to PDF, by default the system will choose an image compression algorithm for embedded images based on the content of the image. When possible, the image compression algorithm of the source image is retained. This job option allows overriding this behavior and forcing a particular image compression algorithm for monochrome (1–bit) images.

`ccitt`
Forces monochrome images to be compressed with the CCITT Group4 algorithm.

`flate`
Forces monochrome images to be compressed with the lossless flate (zip) algorithm.

`default`
Result in the default behavior of selecting or retaining a compression algorithm based on the source image (recommended for best fidelity).

Examples: `pdfexport.forceimagecompression.monochrome=(default|ccitt|flate)`

PdfExportIncludeMetadata
Default value: `true`

Description: When exporting documents to PDF, by default metadata from the source is included in PDF metadata tags. This metadata is typically items such as author, creation date, and title, but can vary from format to format. Use `false` to prevent all such PDF metadata tags from being included in the resulting exported PDF.

Examples: `pdfexportincludemetadata=(true|false)`

PdfExport.PageLayout
Default value: `default`

Description: Setting this option controls what initial page layout display is saved into an exported PDF file. When the exported PDF is opened in a supporting reader, such as Adobe Acrobat, it will open with the specified layout. This does not prevent the reader from then changing to another layout and it only controls the opening layout. The default value of `default` does not save a layout preference, and leaves the initial layout up to the reader that opens the PDF. The other supported values correspond to Adobe PDF properties values as given below.

`default`
Adobe PDF properties value: Default

Description: No preference given.

`singlepage`
Adobe PDF properties value: Single Page

Description: One page shown at a time.

`singlepagecontinuous`
Adobe PDF properties value: Single Page Continuous

Description: One page shown, which can be scrolled smoothly through other pages.

`twoup`
Adobe PDF properties value: Two-Up

Description: Two pages displayed side-by-side.

`twoupcontinuous`
Adobe PDF properties value: Two-Up Continuous

Description: Two pages displayed side-by-side, which can be scrolled smoothly through other pages.

`twoupcoverpage`
Adobe PDF properties value: Two-up (Cover Page)

Description: First page shown as in `singlepage`, later pages shown as in `twoup`.

`twoupcontinuouscoverpage`
Adobe PDF properties value: Two-up Continuous (Cover Page)

Description: First page shown as in `singlepage`, later pages shown as in `twoup`, and can be scrolled.

Examples: `pdfexport.pagelayout=default`

PdfExport.FontEmbed.SingleByte
Default value: `partial`

Description: By default, or when this option is set to `partial`, during PDF export fonts used in a document are embedded in the resulting PDF as partial fonts, only including characters that are actually used in the document. Setting this option to `full` causes single-byte fonts (fonts that only use basic ASCII characters) to be embedded in the resulting PDF as a complete font, with all characters included. This makes the resulting PDF somewhat larger, but will enable editing of the resulting PDF in some PDF editors.

Setting this option to `none` causes single-byte fonts to not be embedded in the PDF at all. The resulting PDF will not display correctly on systems that do not have the excluded font installed.

Note: Embedding of multi-byte fonts (fonts with characters outside the basic ASCII range) is separately controlled by *`PDFExport.FontEmbed.MultiByte`*. Fonts with certain encodings (CID fonts) cannot be fully embedded or excluded and will always be partially embedded.

Examples: `pdfexport.fontembed.singlebyte=(partial|full|none)`

PdfExport.FontEmbed.MultiByte
Default value: `partial`

Description: By default, or when this option is set to `partial`, during PDF export fonts used in a document are embedded in the resulting PDF as partial fonts, only including characters that are actually used in the document. Setting this option to `full` causes multi-byte fonts (fonts with characters outside the basic ASCII range) to be embedded in the resulting PDF as complete fonts, with all characters included. This makes the resulting PDF larger (possibly considerably larger for large fonts), but will enable editing of the resulting PDF in some PDF editors. Setting this option to `none` causes multi-byte fonts to not be embedded in the PDF at all. The resulting PDF will not display correctly on systems that do not have the excluded font installed.

Note: Embedding of single-byte fonts (fonts that only use basic ASCII characters) is separately controlled by *PDFExport.FontEmbed.SingleByte*. Fonts with certain encodings (CID fonts) cannot be fully embedded or excluded and will always be partially embedded.

Examples: `pdfexport.fontembed.multibyte=(partial|full|none)`

PdfExport.FontEmbed.Excluded

Default value: not set

Description: Individual fonts can be excluded entirely from font embedding during PDF export.

The resulting PDF will not display correctly on systems that do not have the excluded font (or equivalent) installed. To exclude fonts, provide the font name(s) separated by the pipe symbol,|.

Note that fonts with certain encodings (CID fonts) cannot be excluded and will always be partially embedded.

Examples: `pdfexport.fontembed.excluded=Arial|Times New Roman|Courier New`

PdfExport.StripTagData

Default value: `false`

Description: PDF files can optionally tag data in a file to indicate its logical structure, or to textually describe visual content. By default, when exporting a PDF from a PDF source with tags, some of these tags (image tags only) are carried through. When this option is set to `true` all tag data is removed from the file during the export process.

Examples: `pdfexport.striptagdata=(true|false)`

6.1.3 TIFF input parameters

TiffDpi

Default value: `300`

Description: Specifies the DPI in the output TIFF file and is valid for use only when the output type of a conversion is TIFF. If the input type is also a TIFF and this parameter is not set, then the output DPI is set to the maximum DPI of any page in the source file. If this parameter is set, it overrides any source file DPI settings. Valid min/max values are 60–1200. A value set less than the minimum of 60 defaults to 60, and a value set more than maximum of 1200 defaults to 1200.

Example: `tiffdpi=<DPI>`

TiffBpp

Default value: `max24`

Description: Sets the bits-per-pixel to use for exported TIFF files. Valid values are 1, 4, max4, 8, max8, 24, and max24. Setting 1, 4, 8 or 24 forces each page of the exported TIFF to the chosen bits-per-pixel value. Setting max4, max8, or

max24 analyzes each page and uses the lowest bits-per-pixel it can, for each page, up to the given value. For example, a document with three black-and-white text pages and one full-color image, when exported with a max8 setting, will export three 1-bit text pages and will convert the full-color image page to an 8-bit image. The resulting file size is also affected by the compression used on each page, see *TiffCompression1Bit*, *TiffCompression4Bit*, *TiffCompression8Bit*, and *TiffCompression24Bit*.

Example: `tiffbpp=(1|4|8|24|max4|max8|max24)`

TiffCompression1Bit

Default value: `CCIT`

Description: Sets the type of image compression to use for TIFF export pages that are 1-bit-per-pixel (monochrome). Valid values are CCITT, LZW, PACKBITS, and ZIP. The default value of CCITT uses CCITT Group 4 (also called G4) compression, which gives excellent compression for most monochrome images.

Note: Depending on the bit-depth requested by *TiffBpp*, it is possible for the exported TIFF to have no pages that are 1-bit-per-pixel. In that case, the value of this setting has no affect on the exported TIFF. See also: *TiffBpp*

Example: `tiffcompression1bit=(CCITT|LZW|PACKBITS|ZIP)`

TiffCompression4Bit

Default value: `LZW`

Description: Sets the type of image compression to use for TIFF export pages that are 4-bits-per-pixel (16 colors). Valid values are LZW, PACKBITS, and ZIP.

Note: Depending on the bit-depth requested by *TiffBpp*, it is possible for the exported TIFF to have no pages that are 4-bits-per-pixel. In that case, the value of this setting has no affect on the exported TIFF.

Example: `tiffcompression4bit=(LZW|PACKBITS|ZIP)`

TiffCompression8Bit

Default value: `LZW`

Description: Sets the type of image compression to use for TIFF export pages that are 8-bits-per-pixel (256 colors). Valid values are LZW, PACKBITS, and ZIP.

Note: Depending on the bit-depth requested by *TiffBpp*, it is possible for the exported TIFF to have no pages that are 8-bits-per-pixel. In that case, the value of this setting has no affect on the exported TIFF.

Example: `tiffcompression8bit=(LZW|PACKBITS|ZIP)`

TiffCompression24Bit

Default value: `LZW`

Description: Sets the type of image compression to use for TIFF export pages that are 24-bits-per-pixel (full colors). Valid values are JPG, LZW, PACKBITS, and ZIP.

Note: Depending on the bit-depth requested by *TiffBpp*, it is possible for the exported TIFF to have no pages that are 24-bits-per-pixel. In that case, this value does not affect the exported TIFF.

Example: `tiffcompression24bit=(JPG|LZW|PACKBITS|ZIP)`

tiffexporttransform.rotate
Default value: `0`

Description: Specifies a number of degrees to rotate each page, counter-clockwise, when exporting to TIFF. Valid values are integer values from 0 to 360. See *tiffexporttransform.pagelist*

Example: `tiffexporttransform.rotate=90`

tiffexporttransform.mirrorx
Default value: `false`

Description: Specifies if the exported TIFF file should be mirrored in the x-axis (horizontally). See *tiffexporttransform.pagelist*.

Example: `tiffexporttransform.mirrorx=false`

tiffexporttransform.mirrory
Default value: `false`

Description: Specifies if the exported TIFF file should be mirrored in the y-axis (vertically). See *tiffexporttransform.pagelist*.

Example: `tiffexporttransform.mirrory=false`

tiffexporttransform.pagelist
Default value: *<empty>*

Description: Specifies a page list of pages to transform when exporting to TIFF. The list is in the same format as the *ExportPageList* parameter. If the page list is not provided or is empty, all exported pages will be transformed. The order of transforms is rotate, mirror in X, mirror in Y. It is not possible to apply different transforms to different pages within a single job, either all specified transforms are applied to a page, or no transforms are applied.

Example: `tiffexporttransform.pagelist=3,4,8-15,20`

TiffExportMaximumDimension
Default value: `18000`

Description: Specifies a maximum dimension in pixels for TIFF export. If the page size and DPI would result in a page being larger than this value in either height or width, the page is resized proportionally to be set at this maximum dimension. To disable this maximum value, set this job option to `0`. This is not recommended, as exporting very large TIFF files might cause crashes due to running out of memory. See also *OutputPageSize* and *TiffDPI*.

Example: `tiffexportmaximumdimension=18000`

TiffExportByteOrder
Default value: `littleendian`

Description: Sets the desired byte order for TIFF export. Most TIFF readers can read either byte order, so the default value of little-endian ordering should rarely require changing.

Example: `tiffexportbyteorder=<littleendian|bigendian>`

TiffExportRotateToFit
Default value: `false`

Description: If true, indicates that the pages of exported TIFF files should be rotated to a consistent orientation. See *`TiffExportRotateToFit.Orientation`* for details.

Example: `tiffexportrotatetofit=(true|false)`

TiffExportRotateToFit.Orientation
Default value: not set

Description: Only used when *`TiffExportRotateToFit`* is set to `true`. This option controls the orientation that pages are rotated to agree with, either portrait or landscape, if they are not already in that orientation. When this value is unset, the resulting pages are portrait orientation, unless *`OutputPageSize`* has been set that is wider than it is tall. In that case, the default is landscape. If a value is specified, that value is always used. When `portrait` is set, pages that are wider than they are tall are rotated 90 degrees during export. When `landscape` is set, pages that are taller than they are wide are rotated instead.

Example: `tiffexportrotatetofit.orientation=(portrait|landscape)`

TiffExportRotateToFit.Direction
Default value: `ccw`

Description: Only used when *`TiffExportRotateToFit`* is set to `true`. This option controls the direction of rotation when a page is rotated. The default is a 90 degree rotation counter-clockwise (`ccw`). To change to a 90 degree rotation clockwise, specify `cw`.

Example: `tiffexportrotatetofit.direction=(ccw|cw)`

6.1.4 OCR processing job input parameters

These parameters control **OCR processing** options.

DoOcr

Default value: `false`

Description: If license permits, any Blazon supported source document file type can be OCR-ed (processed through OCR text recognition) when this parameter is set to `true`. Files that have been preprocessed as OCR can be searched and redacted (Redact using the Blazon Redaction module) with any of the find and redact functions.

Limitations

- TIFF files should be at least 300 x 300 DPI before performing OCR.
- If GIF documents submitted for processing contain multiple files/pages, only the first GIF image is processed.
- When processing only raster images on document pages, the published documents preserve the source document's size and visual attributes and export any hidden text from OCR processing to the document's searchable text.

Examples: `doocr=(true|false)`

BypassOcrForFileFormat

Default value: <*list of extensions*> not set

Description: This setting provides an optional list of file types that will not be processed by OCR in this job, even though *`DoOcr`* has been set `true`. The file types should be listed by extension, separated by the pipe character (for example, `jpg|bmp`). If *`DoOcr`* is `false`, this option has no effect. The default value is an empty list and to apply *`DoOcr`* to all files found in the job.

Examples: `bypassocrforfileformat=<list|of|extensions>`

Ocr.Timeout

Default value: `1800`

Description: The longest time, in seconds, that the OCR engine will spend attempting to recognize characters in any one page. If this time elapses during recognition analysis of an image and recognition has not completed, then the OCR step will abort with an error. Note that regardless of the setting for this value, the JP *`timeout`* parameter and *`DefaultStepTimeoutInSeconds`* will be enforced on the OCR step as a whole, aborting with a failure if OCR has not completed within that time frame. Default if not set is 1800 seconds for each page.

Examples: `ocrtimeout=<number of seconds>`

Ocr.LanguageDictionaryCorrection

Default value: not set

Description: This is a ranked list of countries/languages representing the language content of the document, if known. If multiple countries/languages are provided, the first entry will be treated as being the dominant source of conventions considered during recognition, and each subsequent entry will be treated as less and less significant.

Note: Use of the Chinese, Korean, Japanese, or Thai language groups requires additional licensing.

For optimal processing, set only one country language item from the following list. If more than one country or language item from the list is set, the spelling and case (upper) must match exactly, and be delimited by either a comma (,) or semicolon (;) to avoid processing errors.

The complete list of supported country/language names by language group is:

Cyrillic: BELARUS or BELARUSIAN, BULGARIA or BULGARIAN, CYRILLIC, UKRAINE or UKRAINIAN, RUSSIA or RUSSIAN

Greek: GREECE or GREEK

Latin:
CENTRAL_EUROPE, WESTERN_EUROPE, TURKEY, BALTIC, AFRIKAANS, ALBANIAN, ANDORRA, ARGENTINA, AUSTRALIA, AUSTRIA, BALTIC, BASQUE, BELGIUM, BOSNIA_LATIN, BOSNIAN_LATIN, BRAZIL, CANADA, CATALAN, CENTRAL_AMERICA, CENTRAL_EUROPE, CHILE, COLOMBIA, CROATIA or CROATIAN, CZECH or CZECH_LANGUAGE, DANISH, DENMARK, DUTCH, ENGLISH, ESTONIA or ESTONIAN, FAROESE, FINLAND, FINNISH, FRANCE, FRENCH, FRISIAN, GERMAN or GERMANY, GREAT_BRITAIN, GUARANI, HANI, HUNGARIAN, HUNGARY, ICELAND or ICELANDIC, INDONESIAN, IRELAND, IRISH, ITALIAN, ITALY, JAPAN_LATIN_ONLY, KAZAKH_LATIN, KAZAKH_CYRILLIC, KIRGHIZ_ CYRILLIC , KIRUNDI, LATIN, LATVIA or LATVIAN, LIECHTENSTEIN, LITHUANIA or LITHUANIAN, LUXEMBOURG or LUXEMBOURGISH, MACEDONIAN, MALAY, MEXICO, NETHERLANDS, NEW_ZEALAND, NORWAY, NORWEGIAN, POLAND; POLISH, PORTUGAL, PORTUGUESE, QUECHUA, RHAETO_ROMANIC, ROMANIA or ROMANIAN, RWANDA, SCANDINAVIA, SERBIA_CYRILLIC, LATIN, SERBIA_LATIN, SERBIAN_LATIN,SHONA, SLOVAK or SLOVAKIA, SLOVENIA or SLOVENIAN, SOMALI, SORBIAN, SOUTH_AFRICA, SOUTH_AMERICA or SOUTH_AMERICA_SPANISH, SPAIN, SPANISH, SWAHILI, SWEDEN, SWEDISH, SWITZERLAND, TAJIK_CYRILLIC, TURKEY, TURKISH, TURKMEN_LATIN, TURKMEN_ CYRILLIC, USA, UZBEK_ LATIN, UZBEK_ CYRILLIC , VENEZUELA, WESTERN_EUROPE, WOLOF, XHOSA, ZULU

Azerjaijanian: AZERBAIJAN_CYRILLIC, AZERBAIJAN_LATIN , AZERBAIJANI_LATIN, or AZERBAIJANI_CYRILLIC

Chinese: CHINESE_SIMPLIFIED, CHINESE_TRADITIONAL, or CHINESE_TRADITIONAL_HK

Japanese: JAPAN or JAPANESE (cannot select both)

Korean: KOREAN

Thai: THAI or THAILAND

Examples: `ocr.languagedictionarycorrection=<language name>`

Notes

- If you work with Thai language, ensure that both Fonts `Angsana New` and `Tahoma` are installed.
- If you work with a Chinese, Japanese, or-Korean language, the following fonts must be installed: `SimHei`, `MingLiU`, `MS Mincho`, `Gulim`, `MS Gothic`.
- For PDF-generation, Font type `Arial`, `Courier New` and `Times New Roman` should be installed.
- OpenText strongly recommends the Font `Arial Unicode` is installed (especially for non-western languages).

Important

You must restart the server before the OCR engine can begin using ANY newly installed font.

Ocr.MinConfidence

Default value: `100`

Description: The minimum level of confidence for each page that recognition must achieve for any one character to be considered successfully recognized. A higher confidence level indicates more reliable recognition results. All characters with confidences below the specified minimum confidence will be marked in red and considered below-confidence. By default, below-confidence characters do not stop the recognition process, but the options *`ocr.BelowConfidenceThreshold`* and *`ocr.ErrorThreshold`* can be set together or individually to abort OCR with an error if a large enough number of such failures occur. Valid value is a number between 0-255 with the default being 100 if not set.

Examples: `ocr.minconficence=<0 to 255>`

Ocr.BelowConfidenceThreshold

Default value: `100`

Description: The maximum number of characters out of every 100 evaluated for each page that can fail recognition with a confidence less than *`ocr.MinConfidence`*, but greater than zero (meaning a possibly viable character was identified, but with lower confidence in its correctness). If this value is exceeded, OCR processing will stop with an error, failing the job. A value of 100 means that OCR processing will not be terminated regardless of the number of below-confidence recognitions. For western European languages, it is recommended that this parameter is omitted, or set to `100`. Valid value is a number between 0-100 with the default being 100 if not set.

Examples: `ocr.belowconficencethreshold=<0 to 100>`

Ocr.ErrorThreshold

Default value: `50`

Description: The maximum number of characters out of every 100 evaluated for each page that can fail recognition with a confidence of 0 (meaning no viable characters were identified). If this value is exceeded, OCR processing will stop with an error, failing the job. A value of 100 means that OCR processing is not terminated regardless of the number of failed recognitions. A setting of 100 can improve recognition speed, while any setting less than 100 can reduce recognition speed. Valid value is a number between 0-100 with the default being 50 if not set.

Examples: `ocr.errorthreshold=<0 to 100>`

Ocr.IgnoreErrors
Default value: `false`

Description: When set to `false`, if OCR processing errors occur, they are reported and the publishing job is stopped. When `true` and *`DoOcr`* is also `true`, OCR processing errors are ignored and are prevented from failing the publishing job.

Examples: `ocr.ignoreerrors=(true|false)`

Ocr.ProcessingInformationFileRspInput
Default value: `false`

Description: The path (absolute or UNC) to a valid Recostar project .rsp file (containing pending OCR processing) instead of (not in addition to) the default OCR processing on the input source files. If this option is provided, then all of the *`ocr.*`* options are ignored.

Examples: `ocr.processinginformationfilerspinput=<path to file>`

6.1.5 Comparison job input parameters

These parameters control comparison processing options.

GenerateTextComparison
Default value: `false`

Description: If set to `true`, this parameter generates a PDF file text comparison report showing differences between the textual content (only) of 2 source documents. The report is the same as the text comparison report for 2 documents generated by Brava! Desktop. The source documents must be provided in the *`Source0`* and *`Source1`* input parameters. No other publishing parameters are processed (redactions Redact, stamps, watermarks, or other items, are not applied to the source documents). The report is generated and saved in the specified target directory.

Examples: `generatetextcomparison=(true|false)`

TextComparisonReportName
Default value: not set

Description: When *`GenerateTextComparison`* is `true`, this parameter defines a name to be given to the text comparison report document. If not specified, the

name for the report is generated from the names of the source documents as follows:

Examples: `textcomparisonreportname=<source0-name>_<source0-extension>_<source1-name>_<source1-extension>_TextCompare.pdf`

TextComparisonOldFileName
Default value: not set

Description: When *`GenerateTextComparison`* is `true`, this property specifies the document name to use for the document in the "old" role (corresponding to the source document provided with input parameter *`source0`*) in the generated report. If not specified, the report will use the file name of the *`source0`* input parameter.

Examples: `textcomparisonoldfilename=<source0-name>`

TextComparisonNewFileName
Default value: not set

Description: When *`GenerateTextComparison`* is `true`, this property specifies the document name to use for the document in the "new" role (corresponding to the source document provided with input parameter *`source1`*) in the generated report. If not specified, the report will use the file name of the *`source1`* input parameter.

Examples: `textcomparisonnewfilename=<source1-name>`

GenerateTextComparison.Color.Addition
Default value: `0,255,128`

Description: Sets the color of the text used to denote additions in the text comparison report generated when *`GenerateTextComparison`* is true. The color can be a comma separated RGB value or a named color. Allowed color names come from the .Net Known Colors enumeration (see http://msdn2.microsoft.com/en-us/library/system.drawing.knowncolor.aspx.

Examples: `generatetextcomparison.color.addition=127,127,127`

`generatetextcomparison.color.addition=RoyalBlue`

GenerateTextComparison.Color.Deletion
Default value: `255,0,0`

Description: Like *`GenerateTextComparison.Color.Addition`*, but sets the color of the text used to denote deletions.

Examples: `generatetextcomparison.color.deletion=127,127,127`

`generatetextcomparison.color.deletion=RoyalBlue`

GenerateTextComparison.Color.Change
Default value: `255,224,0`

Description: Like *`GenerateTextComparison.Color.Addition`*, but sets the color of the text used to denote changes.

Examples: `generatetextcomparison.color.change=127,127,127`

```
generatetextcomparison.color.change=RoyalBlue
```

6.1.6 Thumbnail publishing job input parameters

These parameters control **Thumbnail publishing** options.

Note: One of the following parameters are **required** for thumbnail publishing:

ThumbSizes, *ThumbWidth*, or *ThumbHeight*

ThumbFormat

Default value: `JPG`

Description: Specifies the file type to use for exported thumbnails. Valid values are:

`JPG`
Forces thumbnails to JPG format as the default output file type.

`PNG`
Forces thumbnails to PNG format as the default output file type.

`smallest`
Thumbnails are output as either JPG or PNG for each page, depending on which format should be smaller based on the content of the page. If the page has no raster images, or only one monochrome raster, then the system exports PNG. In all other cases (more than one raster, or one raster that is not monochrome), then the system exports thumbnails as JPG.

Example: `thumbformat=(JPG|PNG|smallest)`

ThumbName

Default value: `thumbnail`

Description: The name of the thumbnail file, without an extension. The ".jpg" or ".png" will be appended to the file name. Thumbnails are placed in the directory specified by the target parameter.

Example: `thumbname=thumbnail`

ThumbName.SupressDimensions

Default value: `false`

Description: When `true`, thumbnails generated have filenames of the form *<thumbname>*`_page_`*<pageindex>*`.`*<ext>*. When `false`, thumbnails generated include the dimensions of the image in the filename, and follow the form *<thumbname>*`_`*<width>*`_`*<height>*`_`*<pageindex>*`.`*<ext>*. If this value is set to `true`, only one size of thumbnail can be specified by the thumbnail size job options or the job will fail

Example: `thumbname.supressdimensions=(true|false)`

ThumbSizes

Default value: not set

Description: The size of the thumbnail (in pixels) to create, as x,y. More than one size thumbnail can be exported in a given print job.

If *ThumbSizes* is set, *ThumbWidths* and *ThumbHeights* parameters are ignored. The precedence for thumb sizes is *ThumbSizes*, *ThumbWidths*, *ThumbHeights*.

Example: `thumbsizes=300,200`

`thumbsizes=W1,H1,W2,H2,W3,H3,…,Wn,Hn` results in "n" thumbnails for each page requested.

ThumbWidths, ThumbHeights

Default value: not set

Description: Allows thumbnail requests by width or height only. When set, the system calculates the other dimension for each page being published. Both of these parameters can be a list of values. If specified as a list, the number of values entered will display for each page.

You can only specify either heights or widths. If both are specified, then heights is ignored (according to the precedence stated in *ThumbSizes*.

Example: `thumbwidths=200,300,1000`

Results in three thumbnails being created for each page. One at 200 pixels wide, a second at 300 pixels, and the third at 1000 pixels. Height is determined by constrained proportion.

ThumbPages

Default value: `A`

Description: Specifies the pages to create thumbnails for. Valid values are:

A – Creates a thumbnail for all the pages in the document

F – Creates a thumbnail for the first page only

<*page number*> – Creates a thumbnail for the specified page. Only one page number can be specified.

See "Thumbnail output" on page 124 for information on the filename format.

Example: `thumbpages=(A|F|<page number>)`

ThumbQuality

Default value: `75`

Description: The quality of the thumbnail JPEG if applicable. Enter a number from 1 (lowest quality) to 100 (highest quality).

Example: `thumbquality=<1-100>`

ThumbRotateToFit

Default value: `false`

Description: If true, indicates that thumbnails should be rotated to a consistent orientation. See *ThumbRotateToFit.Orientation* for details.

Example: `thumbrotatetofit=(true|false)`

ThumbRotateToFit.Orientation
Default value: not set

Description: Only used when *ThumbRotateToFit* is set to `true`. This option controls the orientation that pages are rotated to agree with, either portrait or landscape, if they are not already in that orientation. When this value is unset, the resulting pages are portrait orientation, unless both a height and a width has been set for the thumbnail and the set width is larger than the height. In that case, the default is landscape. If a value is specified, it is used in all cases. When `portrait` is set, pages that are wider than they are tall are rotated 90 degrees during export. When `landscape` is set, pages that are taller than they are wide are rotated instead.

Example: `thumbrotatetofit.orientation=(portrait|landscape)`

ThumbRotateToFit.Direction
Default value: `ccw`

Description: Only used when *ThumbRotateToFit* is set to `true`. This option controls the direction of rotation when a page is rotated. The default is a 90 degree rotation counter-clockwise (`ccw`). To change to a 90 degree rotation clockwise, specify `cw`.

Example: `thumbrotatetofit.direction=(ccw|cw)`

6.1.7 Redaction publishing job input parameters

Redact These parameters control **Redaction publishing** options and are used with the optional Blazon Redaction module only:

AuthorName
Default value: not set

Description: If a value is specified, that name is used as the author name for all markups created by redaction scripts and zones. If this parameter is not specified, then the author name will be set to *RedactionServer*.

Brava! companion products will not allow editing of redaction markups without the Brava! *username* matching the *AuthorName*. The **Verify Redactions** panel will not navigate non-matching redactions. The **Consolidate** mode of the Brava! client does allow a switch of ownership of markups objects.

Example: `authorname=JohnDoe`

RedactionScriptFilename
Default value: not set

Description: The path and file name of a redaction script to apply when publishing a file.

Example: `redactionscriptfilename=<path to script file>`

RedactionScriptListFilename
Default value: not set

Description: The name of a text file containing a list of multiple redaction scripts to apply when publishing a file.

To use this feature, set this parameter to a text file that contains the full path to each redaction script file (one for each line) to be applied.

Example: `redactionscriptlistfilename=redactionscriptlist.txt`

RedactionColor

Default value: not set

Description: Set the color of the redaction blocks used to block out items matching the commands in *`RedactionScriptFileName`*. This match can be a comma separated RGB value or a named color. The color name comes from the .Net `Known Colors` enumeration, which is documented here: http://msdn2.microsoft.com/en-us/library/system.drawing.knowncolor.aspx

Examples: `redactioncolor=127,127,127`

`redactioncolor=RoyalBlue`

Note: *`RedactionColor`* only applies to terms redacted in redaction scripts and not redaction zones.

RedactionReasonFontName

Default value: `Arial`

Description: Sets the font of the redaction reason codes placed over redaction blocks.

The value can be any font name available on the system where the JP is installed (for example, `Courier New`, `Bauhaus 93`, or `Tahoma`).

Example: `redactionreasonfontname=Courier New`

RedactionReasonColor

Default value: not set

The default, if not specified, is for the color to be auto-selected based on contrast with the redaction block color.

Description: Sets the color of the redaction reason codes placed over redaction blocks. The value can either be a comma-separated RGB value (such as "127,127,127"), or it can be a named color from the .Net `Known Colors` enumeration, which is documented here: http://msdn2.microsoft.com/en-us/library/system.drawing.knowncolor.aspx

Examples: `redactionreasoncolor=127,127,127`

`redactionreasoncolor=RoyalBlue`

RedactionReasonSize

Default value: not set

The default value is to auto-size according to the size of the redaction block.

Description: Sets the desired font size of the redaction reason codes placed over redaction blocks.

Example: `redactionreasonsize=12`

RedactionReasonMinSize
Default value: `8`

Description: Sets the smallest acceptable font size of the redaction reason codes placed over redaction blocks.

Example: `redactionreasonminsize=8`

RedactionReasonBold
Default value: `false`

Description: Sets whether to display reason codes in bold text.

Example: `redactionreasonbold=(true|false)`

RedactionReasonItalic
Default value: `true`

Description: Sets whether to display reason codes in italicized text.

Example: `redactionreasonitalic=(true|false)`

RedactionReasonUnderline
Default value: `false`

Description: Sets whether to display reason codes in underlined text.

Example: `redactionreasonunderline=(true|false)`

RedactionZonesFilename
Default value: not set

Description: This option lets you specify a zone file name for redaction. You can use the Brava! Desktop product to create markup files that will be applied as redaction zones to the output file. Use the **Blockout (For Redaction)** tool to create a markup file with one or more blackout zones for the source file.

Note: The coordinate system of the markup must match the file type being published. Markups created for a CAD drawing will not appear properly if applied to a PDF, for instance.

Example: `redactionzonesfilename=RedactionZones`

OutputRedactionInfo
Default value: `false`

Description: If set to `true`, and the output file type is PDF or TIFF, a redaction summary page is appended to the end of the output. See also: `RedactionInfoLegend`

Example: `outputredactioninfo=(true|false)`

OutputRedactionInfoAsFile
Default value: `false`

Description: If set to `true`, this option outputs a redaction summary page as a separate text file uniquely named `<target file name>.redactioninfo.txt`.

Example: `outputredactioninfoAsFile=(true|false)`

RedactionInfoLegend
Default value: not set

Description: If set, a legend is applied to the output summary page determined by the *OutputRedactionInfoAsFile* or *OutputRedactionInfo* parameter. The value of this parameter is the text of the legend and should be a string of less than 50 characters which will be presented at the top of the redaction summary output.

Example: `redactioninfolegend=<character string>`

OutputRedactionAuditLog
Default value: `false`

Description: If set to `true`, and a filename is provided by *RedactionAuditLogFilename*, this option outputs a redaction audit log file as a separate XML file. See also "Redaction log processing" on page 54.

Example: `outputredactionauditlog=(true|false)`

RedactionAuditLogFilename
Default value: not set

Description: Specifies a filename to use for the redaction audit log. This job option must be provided if *OutputRedactionAuditLog* is set to `true`.

Example: `redactionauditlogfilename=MyAuditLog.xml`

DisableRedactionBorders
Default value: `false`

Description: If set to `true`, borders are removed on all redaction entities created by scripts or loaded in markups. Useful when you don't want a redacted area to be obvious to the end user. For example, to create the appearance of blank white space, create white block out entities with borders turned off (on a document with white background).

Example: `disableredactionborders=(true|false)`

ShowRedactionReasonsOnBlockouts
Default value: `true`

Description: If set to `false`, the output file will not have redaction reasons labeled on blockout entities.

Example: `showredactionreasonsonblockouts=(true|false)`

ShowRedactionReasonsOnSummary
Default value: `true`

Description: If set to `false`, the redaction summary page (either as part of the file or as a separate file) will not contain redaction reasons

Example: `showredactionreasonsonsummary=(true|false)`

GeneratePrivilegeData
Default value: `false`

Description: If set to `true`, an XML file is generated containing a summary of the number of redactions performed, the associated reason codes, the Bates numbering range (if applicable), and other descriptive information. The file will be named after the target published file name appended with `-privilege-data`. For example, if the published file is named `briefing.pdf`, the privilege data file would be named `briefing-pdf-privilege-data.xml`.

Example: `generateprivilegedata=(true|false)`

ReasonCodeDescriptionFilename

Default value: not set

Description: The path and file name of a Brava! Desktop formatted redaction reason code description file. The long and short descriptions contained in this csv file will be included in the XML file produced by the *`GeneratePrivilegeData`* setting. For more information on the format of this file, see the `Creating a Reason CSV File` section of *OpenText Brava! Desktop - User Guide (CLBRVW-UBD)*.

Example: `reasoncodedescriptionfilename=<path to file>`

PrivilegeDataDirectory

Default value: not set

Description: The path to a directory where the Job Processor has access rights to create the XML files produced by the *`GeneratePrivilegeData`* setting. By directing multiple publishing requests to the same directory, the resulting XML files can be consolidated into a single report. If not set, the default is to place the privilege data file in the same directory as the published file.

Example: `privilegedatadirectory=<path to directory>`

PublishAsReviewDraft

Default value: `false`

Description: If set to `true`, redactions will not be burned in, but will have their blockout regions made transparent so the underlying text can be viewed.

Note: This option is not supported for publication to CSF format.

Example: `publishasreviewdraft=(true|false)`

6.1.8 CSF Writer publishing job input parameters

These parameters control CSF Writer publishing options on a per-job basis:

For the input parameters in this section to take effect, the `JobProcessor.config` file and CSF Writer's `BIPrint.ini` file must both be modified as follows:

1. In the `JobProcessor.config` file, add the following line:

   ```
   BIPrintJobIniFile=<full path>\BIPrintJob.ini
   ```

2. In CSF Writer's `BIPrint.ini` file, add the following section:

   ```
   [BIPrintJobIniFile]

   value=<full path>\BIPrintJob.ini
   ```

The path specified must be the same in both `JobProcessor.config` and `BIPrint.ini`. The path must point to a location that can be read and written to by the account used to run the Job Processor.

The content of `BIPrint.ini` can be empty, or it can contain any of the options described in this section. If any of these options are included, those options will be applied to all CSF Writer publishing requests by default. If any of the options included in `BIPrint.ini` are also provided as input parameters in a job request, then the input parameters provided with the job will override the settings found in `BIPrint.ini`.

Notes

- For detailed information about using the parameters listed in `BIPrint.ini`, see *OpenText Brava! - CSF Writer Publishing Guide (CLBRVW-UCW)*.
- Parameters listed in `BIPrint.ini` are **case-sensitive**.
- If the directory path/full path contains spaces, the path is not required to be placed in quotes. For example:`BIPrintJobIniFile= C:\Program Files\OpenText\Blazon Enterprise\CSFWriter\BIPrintJob.ini`

PrintJobTimeout

Default value: `120`

Description: The amount of time, in seconds, to allow the CSF Writer publishing step to complete. Must be less than the *`Timeout`* and *`StepTimeout`* parameters. The default value, if not set, is 2 minutes.

Example: `PrintJobTimeout=<number of seconds>`

LinksTimeout

Default value: `120`

Description: The amount of time, in seconds, to allow CSF Writer publishing to extract links. Must be less than *`Timeout`* and *`StepTimeout`* . For best results, *`PrintJobTimeout`* and *`LinksTimeout`* should, together, be less than *`StepTimeout`*.

Example: `LinksTimeout=<number of seconds>`

OutlookMessageFormat

Default value: `0`

Description: Determines the file format to which Outlook messages will be converted. The default, if not set, is to keep the existing MSG format.

Valid settings are:

0 – Keep existing MSG format

1 – Convert to plain text format

2 – Convert to rich text format

Example: `OutlookMessageFormat=(0|1|2)`

OutlookAttachments

Default value: `2`

Description: Determines the level of Outlook attachments to be published. By default the value is set to `2`, indicating publish all attachments.

Valid settings are:

0 – Print only message.

1 – Print message with first level attachments.

2 – Print all nested level attachments.

Example: `OutlookAttachments=(0|1|2)`

Note: To control attachments using this option, the .msg file extension must be associated with the `OutsideIn2dl` driver.

WordRevisionsMode

Default value: `0`

Description: Determines whether Microsoft Word displays revisions in balloons in the margin, or inline with the document's text. The default, if not set, is to display balloons.

Valid settings are:

0 – Balloon revisions

1 – Inline revisions

Example: `WordRevisionsMode=(0|1)`

ExcelResetPrintRegion

Default value: `false`

Description: If set to `true`, overrides any print regions defined in Excel documents and uses the default print region for publishing.

Example: `ExcelResetPrintRegion=(true|false)`

ExcelShowHiddenRows

Default value: `false`

Description: If set to `true`, hidden rows from Excel documents will be published as visible.

Note: If *ExcelShowHiddenSheets* is `true`, this setting is ignored and all hidden rows will be visible.

Example: `ExcelShowHiddenRows=(true|false)`

ExcelShowHiddenColumns

Default value: `false`

Description: If set to `true`, hidden columns from Excel documents will be published as visible.

Note: If *ExcelShowHiddenSheets* is `true`, this setting is ignored and all hidden columns will be visible.

Example: `ExcelShowHiddenColumns=(true|false)`

ExcelShowHiddenSheets

Default value: `false`

Description: If set to `true`, all hidden sheets, hidden rows, and hidden columns from Excel documents will be published as visible.

Example: `ExcelShowHiddenSheets=(true|false)`

ExcelPageOrientation

Default value: not set

The default, if not set, is given by the page orientation setting in the source document.

Description: Controls whether Excel pages will be published in portrait or landscape orientation.

Valid settings are:

`Landscape`
Publish pages with landscape orientation.

`Portrait`
Publish pages with portrait orientations.

Example: `ExcelPageOrientation=(Landscape|Portrait)`

ExcelPageOrder

Default value: not set

The default, if not set, is given by the print order setting in the source document.

Description: Controls whether pages within a given sheet are published across the sheet first and then down, or down first, then across.

Valid settings are:

`OverThenDown`
Publish pages with landscape orientation.

`DownThenOver`
Publish pages with portrait orientations.

Example: `ExcelPageOrder=(OverThenDown|DownThenOver)`

ExcelFitToPage
Default value: `false`

Description: If set to `true`, data from published pages are shrunk to fit the published page dimensions.

This parameter, when set to `true`, takes precedence over *`ExcelFitToPageWidth`* and *`ExcelFitToPageHeight`*.

Example: `ExcelFitToPage=(true|false)`

ExcelFitToPageWidth
Default value: not set

Description: If set to `true` and *`ExcelFitToPage`* is `false`, columns from published pages are shrunk to fit the published page width. See additional notes in *`ExcelFitToPageHeight`*.

Example: `ExcelFitToPageWidth=(true|false)`

ExcelFitToPageHeight
Default value: not set

Description: If set to `true` and *`ExcelFitToPage`* is `false`, rows from published pages are shrunk to fit the published page height. Setting both *`ExcelFitToPageHeight`* and *`ExcelFitToPageWidth`* to `true` produces the same result as setting *`ExcelFitToPage`* to `true`.

Set *`ExcelFitToPage`* to `false` and enable only one of these two parameters when you want to specify fitting either the height or width of an Excel sheet within the published page dimensions.

Example: `ExcelFitToPageHeight=(true|false)`

6.1.9 User defined parameters

You can send additional input options (set as `name=value` pairs in the query string of the push) to the Queue Server and they will be echoed, verbatim, by the Job Processor when it notifies you of the completed job. With this option, you can set additional information in the job to track progress, store results, and coordinate sent and received jobs. The only requirement is that the name of the parameter does not conflict with the names listed in the previous table, or with the list of reserved parameter names shown here:

igc_*	mainfile	tempdir
jobid	targetfile	endtime
type	filename	starttime
threaded	ext	totaltime

6.1.10 Deprecated and removed input parameters

6.1.10.1 Deprecated input parameters

The parameters listed in this table are currently deprecated and should not be used. They are listed here for reference purposes only.

Note: Options deprecated in previous releases and have been removed can be viewed in "Removed input parameters" on page 120.

6.1.10.2 Deprecated OCR processing job input parameters

Starting with the 16.2 release, OCR processing is done on an image-by-image basis during document load rather than as a publishing step producing an intermediate PDF. The only output used from the OCR engine is the resulting text data from its analysis. All source images are left as-is in the final published document in order to minimize size increases due to differences in image processing introduced during intermediate PDF generation. Options relating to the processing of an intermediate PDF by the OCR engine have no relevance and are ignored by the Job Processor.

The following descriptions summarize the revised implementation of these deprecated parameters:

Ocr.Orientation
DEPRECATED as of v.16.2

Description: This parameter was previously used as a hint to the OCR analysis engine when the orientation of the entire source document was known in advance to be, for example, portrait vs landscape. It also had the effect of potentially leaving some source images incorrectly rotated in the intermediate PDF document. The fact that this parameter is now ignored will have no measurable effect on OCR analysis time compared to the previous implementation, and eliminates the problem of incorrectly-oriented images in the final published result.

Ocr.OnErrorOutputAsImageOnly
DEPRECATED as of v.16.2

Description: This parameter was previously used to produce an image-only intermediate PDF document in the case where an OCR error occurred at any point during the OCR analysis. Now there is no such intermediate document. If any critical OCR errors occur during the loading of any page of a source document, they will cause the page load to fail, and the job, in turn, will fail.

Ocr.PdfAsInputImage
DEPRECATED as of v.16.2

Description: This parameter was previously used to produce an intermediate PDF that either a) retained the full color palette of the original document or b) left some images in a black/white color space that OCR analysis uses in some cases. Now there is no manipulation of the original source image, so there will

be no documents generated that contain images in a black/white color space used for OCR.

Ocr.ProcessingInformationFileRspPdf
DEPRECATED as of v.16.2

Description: This parameter was previously used to affect intermediate PDF generation. Now there is no intermediate PDF to alter, so any processing that was being performed in these project files will have no effect on published output.

MixedContentOcr, ContinueWithOcr
DEPRECATED as of v.16.2

Description: These parameters were previously used to prevent job failure when source documents with mixed content were submitted for OCR and publishing, because previous implementations could not perform OCR analysis on such documents, and integrations do not usually know whether a given source file has mixed content or not. The parameters had the effect of bypassing OCR processing altogether and only performing the publishing work. Now there are no formats that can't be analyzed by OCR, so the parameters no longer have any meaning. Behavioral differences for jobs that had these parameters both set to `false` when submitting mixed content source documents: previous implementations would fail these jobs, the new implementation will not fail the job, and will perform OCR analysis on the documents.

6.1.10.3 Removed input parameters

These previously deprecated parameters are removed and can no longer be used. Where provided, you can use the replacement parameter instead.

PublishPageList
Removed as of v.16.6.5

Description: This parameter is replaced with *`ExportPageList`*. Use *`ExportPageList`* to avoid problems in future releases.

This parameter specifies a list of page numbers to publish. Only the specified pages will be published into the final format. This can be a list of single pages (separated by the pipe character, `|`) or a set of page ranges (also separated by a pipe).

Examples: `Publishpagelist=0|2|5-10|15-20`

This example would publish the 1st, 3rd, 6th – 11th and 16th – 20th page in the file.

If XDL is the output type, then only single pages are supported (not arbitrary pages).

To use this parameter with multiple source publishing (combining multiple source documents into 1 document), the page numbers entered will apply to the combined document. For example, if combining three 10-page documents and you want to publish only the first page of each document, set: `publishpagelist=0|10|20`

MarkupFileName

Removed as of v.16.6.5. Use *MarkupN*.

Description: A single file name (XRL) that contains a proprietary formatted markup to be burned into the published file. It must be appropriate to the file specified in source. Supported markup types include our XRL format, Acrobat XFDF, and Autovue markups. See `MCU_AVReader_ReadMe.pdf` and `MCU_AdobeXFDFReader_ReadMe.pdf` for instructions on converting Acrobat and AutoVue markups. This parameter does nothing if the output format is XDL.

MarkupListFileName

Removed as of v.16.6.5. Use *MarkupN*.

Description: A text file containing a list of markup files, including path to file. One entry for each line.

SourceN_MarkupFileNameM

Removed as of v.16.6.5. Use *SourceN.MarkupN*.

Description: For multi-doc publishing, you can apply multiple markup files to specific source documents (and only those source documents). To do so, set multiple markup file paths using *SourceN_MarkupFilenameM*, where *SourceN* matches a *SourceN* parameter provided with the multi-doc job, and *MarkupFilenameM* identifies the path to a markup file to apply to that specific source document (and not to the others). "M" values must start at 0, and increase by 1 for each markup to be applied to the same source document. Note that this markup is applied in addition to any markup files provided through *MarkupFilename* or *MarkupListFilename*.

Examples:

```
source0=\\computer\share\Airplane.dwg
source1=\\computer\share\AirplaneSpecs.doc
source2=\\computer\share\AirplaneBrochure.pdf
source1_markupfilename0=\\computer\share\Markup1.xrl
source1_markupfilename1=\\computer\share\Markup2.xrl
```

PdfExportA1a

Removed as of v.16.6.4. Use *PdfExportA1aTag*.

Description: If `true`, then the PDF output will be PDF/A-1a archive compatible.

PdfExportA3a

Removed as of v.16.6.4. Use *PdfExportA3aTag*.

Description: If `true`, then the PDF output will be PDF/A-3a archive compatible.

EmailPublishAttachments

Removed as of v.16.6.6. See *AppendAttachments* in the `Eml2dl` section of the loader parameters guide for replacement option.*OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)*

Description: If any of the source documents in a job is a supported e-mail message format and this parameter is set to `true`, then any attachments in the email message are also published as additional pages in the final document. If `false`, then no attachments are published.

Note: This parameter and *emailpublishnestedattachments* are applicable only when publishing MSG files using the EML2dl loader and are ignored if using CSF Writer publishing. By default, all email attachments are published. If you want to use these job options to prevent attachments from printing, use the loaders.configuration.exe and change the .msg file extension association from OutsideIn2dl to EML2dl.

EmailPublishNestedAttachments

Removed as of v.16.6.6. See *EnableNestedAttachments* in the Eml2dl section of the loader parameters guide for replacement option.*OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)*

Description: If *EmailPublishAttachments* is true, and this parameter is also true, then any attachments which are also email messages with attachments are published as additional pages in the final document. If false, then only attachments appearing in the top-level e-mail message are published as additional pages in the final document. See note in the previous setting *EmailPublishAttachments*.

Example: emailpublishnestedattachments=(true|false)

TiffCompressionType

Removed as of v.16.6.7.

Description: Determines the TIFF compression algorithm type to use in the output TIFF file.

Example: tiffcompressiontype=(jpg|LZW|CCITT|PACKBITS)

CCITT is also known as “G4”.

6.2 Job output parameters

The output parameters listed in this section are returned from Blazon Enterprise after a job is complete. The output parameters let you deal with the results of the job, including errors.

MainFile

Description: The main output file. This parameter will be present only if publishing was successful. It will be the first file name for a multi-file output, or the name of the single file for the output of a single file (such as PDF). If you choose to publish each page of a document to a separate file, this will be the name of the first file.

Return: MainFile=*<file name>*

PublishedPageCount

Description: Returns the number of pages successfully published, if publish succeeds.

Return: PublishedPageCount=*<# of pages>*

SourcePageCount

Description: Returns the total number of pages in the source file.

Return: `SourcePageCount=<# of pages>`

Warning

Description: Any warnings that occur during the conversion are presented here.

Return: `Warning=<string>`

Error

Description: Any errors during conversion are detailed here. See the Errors table in the next section for code descriptions.

Return: `Error=<error code>`

Error/Warning 0,1, ..N

Description: Provides additional information, as required, relating to errors or warnings presented in the *Error* parameter.

Return example: `error 1= mismatched widgets`

Ext

Description: The original extension of the published file (such as PDF or PPT)

Return: `ext=<extension>`

CdlVersion

Description: The version of CDL (internal library) used during conversion.

Return: `CdlVersion=<version number>`

DlfcVersion

Description: The version of the DL file converter `igc.dlfileconverter.dll` used to process the job.

Return: `DlfcVersion=<version number>`

DlfcPlatform

Description: The platform of the DL file converter `igc.dlfileconverter.dll` that is used to process the job. Set to `x86` for the 32–bit version or `x64` for the 64–bit version of the DLL.

Return: `DlfcPlatform=(x86|x64)`

DriverInfo

Description: The specific driver or drivers used in publishing the file including the version number.

Return: `DriverInfo=<filenames+version number>`

StartTime

Description: The time the job started getting processed by the Job Processor.

Return: `StartTime=<time>`

EndTime

Description: The time the job finished processing by the Job Processor.

Return: `EndTime=<time>`

TotalTime

Description: The calculated time to publish the document (*starttime - endtime*).

Return: `Totalime=<time>`

PublisherName

Description: The Job Processor that published the job, along with the Job Processor version.

Return: `PublisherName=<JP name + version>`

OcrResults

Description: Information about the OCR process. Errors are reported in the typical Error 0, Error 1 scheme.

Return: `OcrResults=<ocr info>`

6.3 Thumbnail output

If you request thumbnail files during publishing, they are placed in the directory that you specify using the *Target* parameter.

Note: If *any* Visual Right is set on a published CSF file, including expiration dates, the file's thumbnail image will display with a default CSF icon and not a viewable thumbnail image of the document.

When you publish files and request one or more thumbnails, the thumbnail file names that the system assigns to the thumbnail files are formatted as follows:

```
<ThumbName>_XxY_page_N.jpg
```

Where:

<*ThumbName*> – is the value you specify in the *ThumbName* parameter

X – is the width of the thumbnail in pixels

Y – is the height of the thumbnail in pixels

N – is the page number of the thumbnail

If you request more than one thumbnail for each page, of different sizes, that distinction is reflected in the XxY portion of the file name.

6.4 Loader messages

During conversion, loaders might provide messages about the conversion. These messages are reported by the Job Processor as either:

```
Loader Error ## (description): detail
```

```
Loader Warning ## (description): detail
```

The number codes and their descriptions are listed in the following table:

Code	Type	Description
1	Error	Out of memory
3	Error	Out of stack
4	Error	Can’t write output
5	Error	Can’t open file
6	Error	Can’t find file
7	Error	Can’t create file
8	Error	Can’t read file
9	Error	Can’t write file
10	Error	Corrupt file
11	Error	Unsupported format
12	Error	Unlicensed format
13	Error	Can’t create image
14	Error	Can’t insert entity
15	Error	Can’t register layer
16	Error	Can’t register linestyle
17	Error	Invalid limits
18	Error	Invalid page
19	Error	Invalid version
20	Error	Invalid layer
21	Error	Invalid entity
22	Error	Callback error
23	Error	3rd party library error
24	Error	Exception thrown
25	Error	No 2D entities
26	Error	Can’t load loader

Code	Type	Description
27	Warning	Can't find support file
28	Warning	Can't find Xref
29	Error	Previous loader is still active
30	Error	Page is out of range
31	Error	No document open
32	Error	Function not implemented
33	Error	Output directory is not set
34	Error	Document summary allocation error
35	Error	DLGenerator allocation error
36	Error	Semaphore allocation error
37	Error	Total pages not known
38	Error	Thread failure
39	Error	File access error
40	Error	Loader error
41	Error	Can't load DLGenerator
42	Error	Invalid argument
43	Error	AutoRec failed
44	Error	Path is too long
50	Error	Can't load page
51	Error	Can't open document
52	Error	Software error
60	Error	CSF conversion failed
61	Error	CSF parameters are missing
62	Warning	Found 3D entities
63	Error	Invalid password
67	Error	User cancel
68	Error	.NET 3.0 is not installed
69	Error	Native file is protected
70	Error	Loader not licensed
71	Error	Print publishing: license error
72	Error	Print publishing: unspecified parameters

Code	Type	Description
73	Error	Print publishing: Module location not found The location of `BiPrint.dll` is expected to be found in the registry path specified by `HKEY_CURRENT_USER\Software\IGC\Net-It Now\<version>\<Install Directory>`.
74	Error	Print publishing: Unknown extension When a source document has an invalid extension (dll, exe, ncp, zip, mcf), or if set to print using command/DDE printing, the file extension cannot be looked up in the Registry.
75	Error	Print publishing: Drag and drop printing A print/printto action is not defined for this file format.
77	Error	Print publishing: DDE conversation User has canceled DDE conversion
78	Error	Print publishing: DDE connect An error has occurred in DDE connection
79	Error	Print publishing: DDE conversion Generic error in DDE conversion
80	Error	Print publishing: Perform shell exec Some applications, such as Office, use DDE to do print/printto actions. Other applications provide only command line printing with no DDE. This error indicates only command line printing failure.
81	Error	Print publishing: Register application An error occurred finding the registered application to print.
82	Error	Print publishing: Register command An error occurred finding the registered print/printto command for the current file format.
83	Error	Print publishing: Main exception As with Office automation errors, this thread execution error occurs when there is a problem writing into `NINFile.xml`. The `NINFile.xml` is a temporary file created to communicate the source document information to CSF Writer, launched after printing. CSF Writer updates this file with GRP and EMF files and BIPrint returns it to BI2DL.

Code	Type	Description
84	Error	Print publishing: Timeout This timeout error displays when any of the following actions exceed their maximum time allowance: printing, exporting to XPS, SaveAs MSG to local temp, or waiting on `NINFile.xml` to be updated with all GRP and EMF files.
85	Error	Print publishing: File read Error reading `NINFile.xml`
86	Error	Print publishing: Matching source doc Unable to match the source document name with the job title in the printed GRP file.
87	Error	Print publishing: Print attachment enabled This error occurs if Outlook **Print attached Files** option is set to `true`. For MSG CSF Writer publishing, this option must be disabled.
88	Error	Print publishing: Create mutex failed Mutex is used to implement mutual exclusion so that no more than one process is accessing a common resource at the same time.
89	Error	Print publishing: Office automation failed
90	Error	Print publishing: Abort security
91	Error	Print publishing: Temp folder not found Unable to find the TEMP folder (typically `%StystemRoot%\temp`). The system TEMP folder path is read from the registry entry: `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\Environment\TEMP` and if *`ControlSet001`* is missing from the registry and any key in the path is missing, this error is returned.
92	Error	Print publishing: Temp folder write failed Unable to write to the system TEMP folder
93	Warning	Print publishing: Item contains delegates Unable to process Outlook message/meeting request because it contains delegates and scheduling (meeting request is sent on behalf of someone).
94	Warning	Automation error: Export to XPS failed Unable to process Outlook message/meeting request because it contains delegates and scheduling (meeting request is sent on behalf of someone).

Code	Type	Description
95	Warning	Automation error: Can't publish MSG File, read receipt requested This print error occurs when unable to process Outlook messages when a read receipt is requested.
96	Warning	Automation error: Can't publish MSG File, item contains non-printable properties This print error occurs when processing Outlook messages using MAPI API, and MAPI API fails.
97	Error	.NET 4.0 is not installed
98	Error	iWork file does not contain PDF
99	Error	Max page count reached
100	Error	Cannot find loader configuration
101	Warning	Cannot find loader parameter configuration
102	Warning	Cannot write loader parameter configuration
103	Warning	Email: Ignored errors on attachments
104	Error	Email: Failed to read main messages
105	Error	Email: Failed to read attachments
106	Error	Email: igc.rdmapi.automation not registered
107	Error	Email: Outlook MAPI not installed
108	Error	Email: igc.rdmapi.automation not created
109	Error	Email: Exception in igc.rdmapi.automation
110	Error	Email: Timout in igc.rdmapi.automation
9999	Error	Unknown error

Publishing error note:

The BiPrint/CSF Writer publishing module performs publishing using either Office automation or DDE. Unless mentioned specifically, the error messages apply to both.

BIPrint.ini: The CSF Writer configuration file is installed to `C:\Program Files\OpenText\Blazon Enterprise\CSFWriter\BiPrint.ini` by default. Refer to the CSF Writer Publishing Guide *OpenText Brava! - CSF Writer Publishing Guide (CLBRVW-UCW)* for detailed information about the default publishing settings contained in this file.

Chapter 7

Job Processor configuration

This chapter covers the Job Processor configuration options. Several tables are presented here to provide you with the details you will need when configuring your Job Processors for Blazon Enterprise .

Job Processor configuration allows the administrator to configure the Job Processor performance and its communication with the Queue Server. Proper configuration is key to optimizing performance.

7.1 Loader configuration

Blazon Enterprise uses a set of DLLs known as “loaders” to read the different supported file formats. The specific loader that is used for which specific file extensions can be customized. Additionally, some loaders have format-specific options that can be customized.

The Loader Configuration Tool (`loaders.configuration.exe`) is a standalone executable that provides a simple user-friendly interface to the multiple XML configuration files that control loader behavior. This executable is located off the product's install directory in the `..\JobProcessor\Igc.Loaders\` folder.

Loader XML is grouped by Loader Profiles. A single Profile, named **default**, is installed with reasonable default values for most users. The Loader XML files installed beneath the `JobProcessor` directory (In `Igc.Loaders\Profiles\<profileid>`) are read-only copies. If changes are made with the Loader Configuration Tool, the changed copies are stored in `%ProgramData%\OpenText\Common\JobProcessor\Profiles\<profileid>`. This allows customization changes to be preserved when upgrading the Job Processor. The Loader Configuration Tool automatically detects when only a single Loader Profile is installed and selects that profile to configure. If multiple Loader Profiles are installed, the Tool will prompt on startup for which profile to customize. It is possible to add additional Loader Profiles and select between them on a per-job basis. See the *`LoaderProfile`* job parameter and the Blazon SDK documentation for details.

For usage, loader updating, and more information, see *OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)*.

7.2 Setting up multiple Job Processors for a single Queue Server

It's possible (and often preferred) to have multiple Job Processors set up to provide publishing capabilities to a single installation of the Queue Server. This section will help you to successfully distribute publishing responsibility.

7.2.1 Background

Each Queue Server manages publish requests, and any number of Job Processors query the Queue Server for available publishing jobs they can handle. This architecture allows publishing power to be increased to accommodate very small to very large systems with very little administrative responsibility.

An added feature of the Job Processor component is that it can be configured to handle only specific types of jobs. This can allow more complicated arrangements where, for example, a few very fast computers with very fast network connections are set up to handle PDF publishing, while a few slower computers can handle the relaxed requirements of other file types. Or, it can be used to direct types that require Microsoft Office to be installed to only a small subset of computers for which that product has been purchased, while using other machines to handle other types.

CSF Writer note:

Publishing with CSF Writer is designed to process one document at a time, for each machine, even when multiple publishing threads are configured.

To improve throughput with CSF Writer publishing methods (such as using the Bi2dl loader), additional Job Processor machines can be added. Alternatively, some formats can be configured with other loaders (such as Otf2dl) that can be run in parallel on a single Job Processor machine. See *OpenText Brava! - Loader Configuration User Guide (CLBRVW-ULC)*.

7.2.2 Prerequisites

Before creating multiple Job Processors to work with one Queue Server, the Queue Server must be set up. After a Queue Server is installed and running, a Job Processor can be installed elsewhere and directed to connect to the Queue Server.

Important

All Job Processors must be UNC-accessible to and from the Queue Server. Although requests are handled using TCP, actual file transfers occur using the UNC mechanism.

7.2.3 Setup

Perform the following steps for each Job Processor that will communicate with a single Queue Server.

Job Processor Setup

1. Using the installation program, install the Job Processor as described in "Installing the Job Processor" on page 23. This can be on any computer system that has both TCP and UNC access to the machine on which the Queue Server is installed.
2. Verify the installation by checking to see that the program now resides on the machine. `C:\Program Files\OpenText\Blazon Enterprise\JobProcessor\`
3. In the `JobProcessor` directory, locate the `JobProcessor.config` file. Open it and acquaint yourself with the following sections:

   ```
   thread.drw,thread.pdf,thread.doc

   queue.server.address
   ```

4. To direct the new Job Processor to connect to the existing Queue Server, change the *`queue.server.address`* property.
5. To define which types of files this Job Processor can handle, change the drw, pdf, and doc thread properties:

   - To allow only a certain group of file types, list only that group in the `jobprocessor.config` file. Comment out all the other thread properties (start the line with a hash mark (#)), leaving the ones that match the thread properties you want to keep.

     Example 7-1:

     If you want this Job Processor to handle only PDF and DRW file groups, the thread properties would look like this:

     ```
     thread.drw=4

     thread.pdf=4

     #thread.doc=2
     ```

     Note: Your specific license restrictions might prevent the option of configuring threads.

7.3 Job processor parameter settings

This section details the Job Processor configuration settings which can be modified in the installed `JobProcessor.config` file.

7.3.1 Job Processor logging configuration

The Job Processor can write to several different logs. Each of these logs can be accessed through the built-in web server:

- The *Status* Log records the current and previous job for each job retrieval thread. This log is always enabled, but is only written to disk when viewed through the built-in web server or when the Job Processor is shut down.
- The *Job* Log records recent jobs processed. This log can be enabled or disabled. The Job Log is written to disk from its own thread as jobs are processed.
- The *Service* Log records information about the working of the Job Processor itself. This log is always enabled, but is only written to disk when viewed through the built-in web server or when the Job Processor is shut down. The level of detail provided is set through Trace Flags that are individually set.

The `Trace Flag` settings in this section determine what types of messages are included in the service log. Certain messages are always included, and cannot be disabled. These include startup information about the Job Processor's runtime environment. The default for all optional flags is `false`, and you can enable an optional flag by setting its value to `true`.

log.folder

Default value: `%programdata%\OpenText\JobProcessor\logs`

Description: The folder where all three of the log record files are written (Status, Job, and Service logs). This setting can be configured to another folder as needed.

Example: `log.folder=<path to file>`

log

Default value: `Off`

Description: Enables or disabled job logging by the Job Processor. The Job Processor can retain the job descriptions of the most recent jobs it has handled. This job information is the same information returned in the job notification upon job completion. The Job Log can be accessed remotely through the built-in web server if enabled. The log file location is displayed in the Job Processor Console window and can be viewed by visiting `http://JPMachine:7070/log`.

Example: `log=(on|off)`

log.size

Default value: `200`

Description: The maximum number of jobs that are written to the log file at any time. Older jobs are removed from the log as newer jobs are processed. If

performance is an issue, logging can be turned off by setting the `log=off` parameter.

Example: `log.size=<# of jobs>`

TraceFlags.Startup
Default value: `false`

Description: Use `true` to include startup messages in the service log. Startup messages are generally related to the initialization of the Job Processor.

Example: `TraceFlags.Startup=(true|false)`

TraceFlags.Running
Default value: `false`

Description: Use `true` to include running messages in the service log. Running messages are related to on-going access to Job Processor files or resources during the course of processing.

Example: `TraceFlags.Running=(true|false)`

TraceFlags.JobRetrieval
Default value: `false`

Description: Use `true` to include JobRetrieval messages in the service log. JobRetrieval messages are generated by the JobGetters during job processing. Enabling this flag will generate significant log activity as each thread requests new jobs.

Example: `TraceFlags.JobRetrieval=(true|false)`

TraceFlags.JobSetup
Default value: `false`

Description: Use `true` to include JobSetup messages in the service log. JobSetup messages are generated as each retrieved job is started. Enabling this flag will generate messages for each job processed.

Example: `TraceFlags.JobSetup=(true|false)`

TraceFlags.JobCleanup
Default value: `false`

Description: Use `true` to include JobCleanup messages in the service log. JobCleanup messages are generated on errors while completing a job, such as sending a done notification.

Example: `TraceFlags.JobCleanup=(true|false)`

TraceFlags.Shutdown
Default value: `false`

Description: Use `true` to include Shutdown messages in the service log. Shutdown messages are generated during the tear-down process when the Job Processor is exiting.

Example: `TraceFlags.Shutdown=(true|false)`

TraceShowThreadId
Default value: `1`

Description: This property configures whether the thread ID is included with the service log messages (1 = include, 0 = exclude)

Example: `TraceShowThreadId=(1|0)`

TraceFileCount
Default value: `4`

Description: This property configures the number of archived trace log files to retain (oldest is deleted as the most recent file reaches its maximum message count).

Example: `TraceFileCount=4`

TraceMessagesPerFile
Default value: `1000`

Description: This property configures the maximum number of trace messages to include in a single log file.

Example: `TraceMessagesPerFile=1000`

7.3.2 Job retrieval configuration

The Job Processor utilizes one or more JobGetters. For each one of these JobGetters, several retrieval threads will be created and will run independently. These threads are separated by the name of the queue they will request jobs from. Multiple threads can be created to service the same queue. The general format to create these threads is `thread.<queue name>=<threads for that queue>`.

As a rule of thumb starting point, the total number of threads configured should be 2-3 times the number of logical cores on the Job Processor machine. So, for example, if the machine runs a quad-core processor and has hyperthreading or similar technology enabled, you might start with a total # of 2-3 x (4 x 2) = 16-24 threads. If hyperthreading is unavailable or disabled, then you might start with a total of 2-3 x 4 = 8-12 threads.

Notes

- This number is only a starting point, and will vary depending on number of different kinds of files, their sizes, file I/O bandwidth and the variability of the load on the Job Processor.
- Each JobGetter instantiated will create the configured number of threads for each configured queue as defined here. If multiple JobGetters will be employed then thread values should be set correspondingly lower.

thread.single
Default value: `3`

Description: The number of jobs that will simultaneously publish single page requests. A single page publishing request (as opposed to the standard full

document publishing request) is sent for any file type (that it has a native driver for) that can be more than one page in length.

Example: `thread.single=3`

thread.drw

Default value: `3`

Description: The number of jobs from the DRW queue to concurrently publish.

Example: `thread.drw=<# of jobs>`

thread.pdf

Default value: `3`

Description: The number of jobs from the PDF queue to concurrently publish.

Example: `thread.pdf=<# of jobs>`

thread.doc

Default value: `2`

Description: The number of jobs from the DOC queue to concurrently publish.

Example: `thread.doc=<# of jobs>`

PersistentQueues

Default value: `single`

Description: A comma separated list of (thread.*) queues that you want to be persisted. This setting allows a document, after it is initially loaded, to be re-used for multiple jobs if that document is to be processed multiple times. These are jobs that can be back-to-back jobs on the same file. The Job Processor will attempt to optimize these types of jobs on the named queues.

Example: `persistentqueues=doc,pdf`

Job Getters

JobGetters are used to retrieve jobs from a job source (typically a job queue). Custom JobGetters are supported. See the SDK documentation for more information. The Blazon JobGetter connects to the Blazon Queue Server through a TCP connection to retrieve jobs.

Notes

- The default JobGetter configuration properties use index values that are used by Open Text product installers to support configuration updates during installation. To configure custom or additional JobGetter entries, use indices with values of 4 or higher. Custom JobGetters with index values of 3 and lower can be used, but might prevent OpenText installers making configuration changes during product installation.

  Indices must be unique integer values, but they are not required to be ordered. For example, since JobGetter indices 0-3 are used by OpenText product installers, a custom `DirectoryWatcherJobGetter` can be configured by adding the following entry:

```
jobgetter.classname.4=igc.jobprocessor.DirectoryWatcherJobGetter
job.file.dir.4=\\myServer\myShare\myJobDirectory
dir.watcher.sleep.time.4=250
```

Although an index value of 4 is shown in this example, any index value equal to or greater than 4 could have been used. Custom JobGetters with index values of 3 and lower can be used, but might prevent Open Text installers making configuration changes during product installation.

jobgetter.classname

Default value: not set

Description: The name of the Job Getter used to retrieve jobs for the Job Processor to process. The `JobProcessor.config` file supports multiple JobGetters.

Examples: `jobgetter.classname=igc.jobprocessor.PopProcessorJobGetter`

To add multiple JobGetters, use the following syntax:

```
jobgetter.classname.0=igc.jobprocessor.PopProcessorJobGetter
jobgetter.classname.1=igc.jobprocessor.DirectoryWatcherJobGetter
jobgetter.classname.2=igc.jobprocessor.PopProcessorJobGetter
```

Notes

- If using only one JobGetter, a `.0` should always follow the `jobgetter.classname`.
- If using multiple JobGetters, the format used is `jobgetter.classname.<index>`, but the index number used is up to you. When a number is chosen, the associated options must also use that same number. For example, if you set the `jobgetter.classname.3=igc.jobprocessor.DirectoryWatcherJobGetter`, you must set the parameter for the `DirectoryWatcherJobGetter` as `job.file.dir.3`.

Example 7-2:

```
jobgetter.classname.1=igc.jobprocessor.DirectoryWatcherJobGetter
job.file.dir.1=\\<<SERVER>>\<<SHARE>>\<<JOBDIRECTORY ONE>>
dir.watcher.sleep.time.1=250
jobgetter.classname.2=igc.jobprocessor.DirectoryWatcherJobGetter
job.file.dir.2=\\<<SERVER>>\<<SHARE>>\<<JOBDIRECTORY TWO>>
dir.watcher.sleep.time.2=150
```

This example configuration sets up two different `DirectoryWatcherJobGetters`, on two different directories, with two different sleep times.

queue.server.address

Default value: <*Blazon application server*> :not set

Description: The hostname of the Server which has an internal publish request queue. Used by the Blazon `PopProcessorJobGetter`.

Example: `queue.server.address.0=<machinename.domainname>`

queue.server.pop.port
Default value: `8890`: not set

Description: The port number of the server which has an internal publish request queue. Used by the `igc.jobprocessor.PollingHttpJobGetter`.

Example: `queue.server.pop.port.0=<port number>`

queue.server.pop.timeout
Default value: 35

Description: The maximum time in seconds this JobGetter will wait for a Queue Server response before recycling the TCP socket. This allows recovery if the Queue Server goes down or is briefly unreachable. Changes to this value should be coordinated with the value of `queue.server.pop.wait.time.in.seconds` on the Queue Server, so that this timeout is not reached before the Queue Server returns an empty queue message.

Example: `queue.server.pop.timeout.0=<# in seconds>`

job.sleep
Default value: `1500`: not set

Description: The maximum time (in milliseconds) between intervals querying the Job Processor queue for jobs awaiting publishing.

Example: `job.sleep.0=<# in seconds>`

job.file.dir
Default value: `\\<SERVER>\<SHARE>\<JOBDIRECTORY>`: not set

Description: The directory where the `DirectoryWatcherJobGetter` retrieves jobs. The `DirectoryWatcherJobGetter` monitors a configured network share directly and reads job files placed in that share. See *OpenText Blazon Enterprise - Blazon Enterprise SDK Guide (CLBRVW-RBZ)* for more information.

Example: `job.file.dir=\\mymachine\jobs`

dir.watcher.sleep.time
Default value: `250`: not set

Description: The amount of time (in milliseconds) that the `DirectoryWatcherJobGetter` spends looking for new jobs before returning "no job present". See *OpenText Blazon Enterprise - Blazon Enterprise SDK Guide (CLBRVW-RBZ)* for more information.

Example: `dir.watcher.sleep.time.0=<# of milliseconds>`

7.3.3 Job Processor access configuration

The parameters in this section control how the Job Processor accesses resources, and how it can be accessed itself.

DataPort

Default value: `7070`

Description: The port that the Job Processor opens to provide configuration and log web pages. The Job Processor runs a built-in light-weight web server which is used to provide remote access to the status of the Job Processor. Point your browser at `http://JPMachine:7070/config` to see configuration information. The server is accessed through the port configured here. To disable the server, set the port to `-1`.

Example: `dataport=7070`

NotificationVerb

Default value: `POST`

Description: For job completion notifications, this parameter establishes the HTTP verb used to communicate with submitters when a request has completed. Valid values are `GET`, `POST`, or `MULTIPARTPOST`. The default value, if not set, is POST. Use of GET might encounter limitations in the URL notification length for jobs with many parameters, or parameters with very long values. The value set here can be overridden by the job options for each job.

Example: `notificationverb=(GET|POST|MULTIPARTPOST)`

publish.to.printers.enabled

Default value: `false`: not set

Description: The Job Processor can send publishing results directly to a printer. To use this advanced feature, set to `true`.

Example: `publish.to.printers.enabled=(true|false)`

publish.to.printers.whitelist

Default value: not set

Description: This parameter allows specification of a white list of acceptable printers to which published results can be submitted. If `publish.to.printers.enabled` is set to `true`, this property should either be left commented out, or set to a non-empty list of comma-separated uri paths in the format `printer://<printer-host>/<printer-name>`

To refer to a printer on the Job Processor machine use `localhost` for the value of *<printer-host>*: `printer://localhost/<printer-name>`.

For example, if the Windows unc path to a printer is `\\print-server\HP LaserJet P2050 Series PS`, its URI would be: `printer://print-server/HP LaserJet P2050 Series PS`

If this property is commented out, this means that job results can be published to any reachable printer on the network. If a whitelist is supplied, then only jobs that specify a matching printer target will be allowed to start. If any of the

values configured for the whitelist are incorrectly formatted, no printer jobs will be allowed to start, and errors will be written to the Windows event log on each failure.

Example: `publish.to.printers.whitelist=<comma separated printer list>`

7.3.4 Process Monitor configuration

The Job Processor runs a Process Monitor thread that runs clean-up tasks on a regular basis. If jobs do not complete successfully, they can leave behind child processes or temporary files. The Process Monitor cleans up these orphaned items.

EnableProcessMonitor

Default value: `true`

Description: This setting controls if the Process Monitor is used. The monitor is enabled by default.

Example: `EnableProcessMonitor=(true|false)`

ProcessMonitorLogFile

Default value: `%programdata%\OpenText\JobProcessor\logs\ProcessMonitorLog.txt`: not set

Description: This defines the Process Monitor log file location. This must be a location that the Job Processor service has write access to.

Example: `ProcessMonitorLogFile=<path to file>`

ProcessMonitorLogFileMaxSizeInMB

Default value: `1`

Description: Determines the Process Monitor log file maximum size in megabytes. Each time a log entry is added, if the existing log is larger than the size provided here, the current log is moved to a `.backup` copy (overwriting the previous backup) and a new log is started.

Example: `ProcessMonitorLogFileMaxSizeInMB=<# of megabytes>`

ProcessMonitorPollingFrequencyInSeconds

Default value: `300`

Description: Polling frequency, in seconds. This setting determines how often to check for orphaned or stale automation processes.

Example: `ProcessMonitorPollingFrequencyInSeconds=<# of seconds>`

ProcessesToMonitor

Default value: `excel.exe, winword.exe, powerpnt.exe, excelcnv.exe, EXCEL.EXE, WINWORD.EXE, POWERPNT.EXE, EXCELCNV.EXE, VISIO.EXE, WINPROJ.EXE, Net-ItNow.exe, BuEAppNT.exe, BuEAppTS.exe, chrome.exe`

Description: Determines the processes to monitor. Only processes named in this comma-separated list are candidates for monitoring. This list IS CASE SENSITIVE. The default list includes the Office programs launched through automation from `Bi2dl.dll` and `Bi2dl` support processes.

Example: `ProcessesToMonitor=<comma separated list of processes>`

StaleProcessTimeout
Default value: `14400`

Description: Cutoff threshold for process age. Only monitored processes that have been running longer than this setting are eligible for termination. The default value translates to 4 hours: 60 (seconds) * 60 (minutes) * 4 (hours) =14,400 seconds).

Example: `StaleProcessTimeout=<# of seconds>`

ProcessMonitorParentProcessName
Default value: `DCOM Server Process Launcher`

Description: Sets the parent service for processes to monitor. Any children of a service process with this `DisplayName` (listed in `ProcessesToMonitor`) are considered. The default value is the service that launches Office automation used by `Bi2dl`. This setting should not be changed unless a future version of Windows requires changes.

Example: `ProcessMonitorParentProcessName=DCOM Server Process Launcher`

ProcessMonitorKillOrphans
Default value: `true`

Description: This additional option monitors processes that have a parent process that no longer exists. This can happen if the automation launcher has exited but was not able to clean up its child processes.

Example: `ProcessMonitorKillOrphans=(true|false)`

TempFileMaxAgeInSeconds
Default value: `14400`

Description: Cutoff threshold for temporary file age. Only matching files that have been on disk longer than this setting are eligible for deletion. The default value translates to 4 hours: 60 (seconds) * 60 (minutes) * 4 (hours) =14,400 seconds)

Example: `TempFileMaxAgeInSeconds=<# of seconds>`

ProcessMonitorCleanupDir
Default value: `see example`

Description: During file conversion, temporary files are sometimes created. These files are normally automatically deleted. In the event of an error, the automatic deletion can be skipped. The Process Monitor thread also monitors the temporary file folders and cleans up certain types of files that the Job Processor creates. These settings customize that cleanup. Directories to monitor for cleanup are listed in sequence. Additional entries can be added by increasing the number at the end of the property. You cannot skip entries, and the first entry is 0. The default 5 entries are common temporary file locations. Entries that do not exist will be quietly ignored.

Example: `ProcessMonitorCleanupDir0=c:\windows\temp`

`ProcessMonitorCleanupDir1=%temp%`

```
ProcessMonitorCleanupDir2=%appdata%

ProcessMonitorCleanupDir3=%localappdata%

ProcessMonitorCleanupDir4=c:\windows\temp\xdltemp

ProcessMonitorCleanupDir5=%programdata%\OpenText\JobProcessor
\.requests
```

ProcessMonitorFileDeletePattern

Default value: `see example`

Description: In each monitored directory, files matching any of the following sequence of file descriptions will be considered for deletion. Additional entries can be added by increasing the number at the end of the property. You cannot skip entries, and the first entry is 0. The default 6 entries are files the Job Processor is known to create during conversions.

Example:

```
ProcessMonitorFileDeletePattern0=DL_*

ProcessMonitorFileDeletePattern1=*.EMF

ProcessMonitorFileDeletePattern2=*.GRP

ProcessMonitorFileDeletePattern3=*.GRPXML

ProcessMonitorFileDeletePattern4=*.XPS

ProcessMonitorFileDeletePattern5=*.PDF
```

ProcessMonitorDirectoryDeletePattern

Default value: `ProcessMonitorDirectoryDeletePattern0=DL_*, ProcessMonitorDirectoryDeletePattern1=JPJOB_*`

Description: In each monitored directory, folders matching any of the following sequence of folder descriptions will be considered for deletion. Additional entries can be added by increasing the number at the end of the property. You cannot skip entries, and the first entry is 0. When a folder is considered for deletion, EVERY file in the folder (and any child folders) is checked against the maximum file age, and deleted if it is older. If this process results in the folder being empty, the folder itself is then deleted. The default entry is a folder name pattern the Job Processor is known to create during conversions.

Example: `ProcessMonitorDirectoryDeletePattern<index>=<directoryname pattern>`

7.4 Page size settings

The Job Processor page size configuration settings are set in `PageSizes.config`.

The file can be extended if desired. After changing the file, you must restart the Job Processor before the changes will take effect. The page sizes are listed without respect to orientation for JIS, ARCH, and ANSI where sizes are listed with smaller dimension first. Letter through Executive are listed in width by height order for the expected (portrait/landscape) orientation. For ISO, the orientation is provided and should be used when specifying this size. Both the page size and orientation can be combined onto a single line.

Example 7-3: Page size settings

```
ISO A1 Landscape|mm|841|594
ISO A1 Portrait|mm|594|841
Letter|in|8.5|11
LetterLandscape|in|11|8.5
```

The page settings are used by the *`OutputPageSize`* parameter.

Note: When exporting DGN files as PDF with a forced page size, fixed line widths will be scaled.

Chapter 8

Tips and troubleshooting

This chapter includes a collection of tips for addressing common issues you might encounter while installing and configuring your Blazon Enterprise software.

8.1 Troubleshooting

Every installation and configuration is unique based on your requirements, third-party software, and equipment. You will find the information in this section helpful when addressing setup, optimization and maintenance issues.

8.1.1 Updating loaders

You should check for availability of the most recent loaders from time to time, and download them to production as a best practice recommendation. Issues are often resolved in loaders, which are released separately from the main product release. You can download the latest from My Support (https://support.opentext.com):

Deployment instructions for your setup are provided in the Release Notes document distributed in each loader package. See also "Loader configuration" on page 131.

8.1.2 Office file formats

The Job Processor offers several loaders to process common Office documents which provide a varying range of benefits. The default BI2DL loader, which requires and uses installed and licensed native applications to provide the required content, offers the highest fidelity. Several other loaders are available with supporting libraries; some internal to OpenText and others from third-parties. These include LOTODL, OTF2DL, and the license fee based OutsideIn2DL loader.

For more information and to review their benefits, see the latest loader and Knowledge Base article for Office based formats on My Support: https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0720088

8.1.3 Microsoft Outlook

To publish MSG files, Microsoft Outlook needs to be installed and initially set up on the Job Processor machine with no personal mail account information set and no Exchange server information set. After this initial setup, close Outlook. Processing MSG files should now launch in Outlook and successfully print-publish.

If you do not perform this initial setup, MSG files do not publish and a critical error message is logged.

Since the CSF Writer might forcefully close Outlook during the CSF Writer publishing process, there should be **no** email accounts configured in the Outlook application on the Job Processor machine because those accounts can become corrupted. If the Job Processor is having problems converting MSG files, do the following:

1. Shut down the Job Processor.
2. Make sure CSF Writer and Outlook processes are not running (go to the Windows Task Manager and end the processes if they appear on the list).
3. Manually attempt to open Outlook and make sure it can launch without reporting errors.
4. Close Microsoft Outlook and restart the Job Processor.

See "Publishing tips and errors" on page 147 for error messages specific to publishing MSG files.

8.1.4 Antivirus exclusions

We highly recommend that you add any server directories that are actively managed by Blazon Enterprise and the Job Processor to the whitelist of your antivirus software. Excluding the directories that hold the display list cache and other Blazon Enterprise components from your antivirus scans can significantly improve performance.

Folders to exclude from antivirus scans:

- The Job Processor temporary directory. By default, this is the system temporary folder.
- The Job Processor data folder. By default, this is: `C:\ProgramData\OpenText\JobProcessor\`

  Specifically, the `.request` folder.

8.2 Publishing tips and errors

MSG files fail to publish when using the eml2dl loader

This issue was reported on Windows 2012 R2 server, but might not be limited to this server version. The security setup on Windows will vary depending on customer requirements. To resolve this issue, system administrators should make sure that the JP service domain has admin account rights for interactive services. This is because the JP is considered a sub-process in Office automation and in this case, `igc.rdmapi.automation.exe` is using MAPI to open and process MSG/EML files. Giving the JP rights to launch processes and applications, and to open files from the local network without being blocked should allow MSG files to publish using `eml2dl`. Refer to the Microsoft Windows 2012 R2, UAC and security details for additional information.

Old Office formats are failing

If attempting to publish older Office formats, print or XPS publishing can fail if the file block protection settings are enabled. This setting can be altered through your Office application file options:

To edit file block settings:

1. From your Office application, go to:

   File > Options > Trust Center > Trust Center Settings > File Block Settings

2. Any file checked as `open` will fail.

Macro-based documents are failing

You should adjust any application Trust Center settings (**File > Options > Trust Center > Trust Center Settings > Macro Settings**) to match your business rules, ensuring that you choose selections containing "Disable all macros without notification" as applicable. Without proper configuration, macro-based documents might not render.

Cyrillic text files are failing

Error message "Unable to read text index file". The encoding standards for input files that we support with the TXT2DL loader are ANSI, UTF8, UCS16LE, and Shift_JIS.

Excel 2010 Crash error

A crash can occur when CSF Writer publishing a file using Excel 2010 if screen updating is turned off. A Microsoft hotfix download is available for this issue and is described in this link:

http://support.microsoft.com/kb/2475875/

Can't open file or Can't find Temp folder error

If one of these publishing error messages is the result of trying to open an Office document, following are two possible causes and resolutions for this issue. Other indications specific to these issues are that the CSF writer will print Office formats from Office applications, but CSF Writer publishing components error logs are not produced (when the logging option is enabled for `<System TEMP>\NINLogs\BIPrintErrors.log`). This indicates that `BIPrint.dll` CSF Writer publishing component is not initializing.

- **Can't open file error:** System updates or regular maintenance of the machine might have removed dependent MFC libraries (that `BIPrint.dll` is linked to). If this component is not running, error logs cannot be produced. The run time dependences can be checked by using a dependency checker such as Dependency Walker `depends.exe`. To resolve this issue, re-distribute the missing MFC libraries into the `System32` folder.
- **Can't find Temp folder error**: This issue occurs when `BIPrint.dll` cannot find the system temp folder. Since the system temp folder is used for logging errors and writing into temporary files while CSF Writer publishing, the publishing is not initiated, and the errors are not logged. To resolve this issue:

  1. Verify the expandable string type on the TEMP environment variable in:

     `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment`

  2. Edit this value if needed. When changing settings, the account has to have full administrative rights.
  3. The TEMP value is usually `%SystemRoot%\TEMP`, but could also be set to concrete value, such as`c:\windows\Temp`.
  4. Verify for the TEMP, TMP, and Path environment variables that the type is `REG_EXPAND_SZ`.

Failed to publish document

After the install completes, the Black Ice install needs to complete installation. If a possible race condition occurs, the Print Spooler might not be started. When this happens, Microsoft PowerPoint, Word, and other MS Documents generate the following error:

```
Server Details:
Failed to publish document.
HTTP Response Code is 500, not 2XX.
Net-It Enterprise error: Critical CDL Error: Print Publishing: Printer not
installed (0x004C): C:\Program Files (x86)\OpenText\Blazon Enterprise\dlcache
\9\bd9a17b8-6341-41e9-bc31-6ca3460895c0\Stellar%20Newsletter.docx
```

Solution:

1. In Control Panel, click **Devices and Printers**.

2. If the **CSF Writer** printer icon does not show in the **Printers and Faxes** panel, but shows other printers, the printer driver for the CSF Writer will be located in the `C:\Windows\System32\spool\drivers\x64\3` directory as the following files:

```
BuEDRVNT.dll or BuEDRVTS.dll
BuEiniNT.ini or BuEiniTS.in
BuEResNT.dll or BuEResTS.dll
BuEUifNT.dll or BuEUifTS.dll
Jpeg32.dll
Tiff32.dll
```

3. If these files are not installed, contact the help desk.
4. If none of the printers, including the CSF Writer printer icon in the Printers and Faxes panel do not display, then do the following:
   a. Open **Services**
   b. The **Print Spooler** service should be set to **Automatic** and **Started**.

Tips

- Some instances of Blazon Enterprise ship with licenses that restrict the number of queues that the Job Processor can process. If more than the number of allowable threads are attempted to be created (as defined in the license), warnings will be created in the event log and the system will continue to run as licensed.
- Ensure that folder set in system TEMP variable, allows write permissions to all accounts using CSF Writer publishing and SaveAsXPS/PDF. Under <*System TEMP*>, publishing components are writing error report files, logging diagnostic reports (optional), and writing temporary files to this directory. To do this, check the registry for the system temp folder (in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session`) and give WRITE access to all the accounts that will do CSF Writer publishing.
- The presence of active RDP sessions and printing outside of our applications while CSF Writer publishing is used is strongly discouraged.
- If you have upgraded your Internet Explorer version on the Job Processor machine and are experiencing problems publishing HTML files, you might be required to edit your registry for CSF Writer publishing to use command line printing instead of DDE printing. (Be sure to back up your registry before performing any edits.) To do this:
  1. Through **regedit**, browse your registry to locate `HKEY_CLASSES_ROOT\htmlfile\shell`.
  2. There should only be one key (folder) listed in the `printto` folder called command (that is set with Type REG_SZ for IE 6,0 or REG_EXPAND_SZ for IE 7.0 to run `rundll32.exe`).
  3. If an additional key (folder) exists called `ddeexec`, you must delete this DDE key (folder) so that CSF Writer publishing will only use command line printing with HTML files.

Chapter 9

Appendix

9.1 Appendix 1 - Input options quick reference table

This table is provided as a quick reference for input job options, grouped in order of function. For full descriptions, see "Optional job input parameters" on page 57.

Publishing Parameter Name/ Description	
Source	Path to the source file to publish.
SourceN	Multiple source file paths.
MultiSourceMerge	Produces N documents in the requested target format from N source documents of a single publish job.
CreateTOC	Creates a table of contents on the first page of the published document.
TocEntry	Used with *`CreateTOC`*, to override default entries.
TocEntryN	Used with *`CreateTOC`* to override default entries in multi-source publishing.
StructuredToc	Used with *`CreateTOC`*, to specify TOC and source document ordering in multi-source publishing.
BookmarkPerSource	Used to create a bookmark for each source file in a job that is merging multiple sources. Cannot be used with *`CreateTOC`*.
header.imagefile/ footer.imagefile	Applies an image (PNG/JPG) centered across the top (header)/bottom (footer) of each page of the output document.
header.height/footer.height	The desired height of the image on the page, in inches. The width is based on the proportions of the image, given the height.
header.inset/footer.inset	The inset is the margin between the top/bottom of the image and the top/bottom of the page.
Target	Target directory, where the published artifacts will be placed. Path or mailto/printer URI.
target.mailto.<query parameter>	If `mailto` is used for the *`Target`* value, additional query parameters available include *`target.mailto<.smtp.server`*, *`.smtp.port`*, *`.useSecureTransport`*, *`.smtp.username`*, *`.smtp.password`*, *`.to`*, *`.from`*, *`.cc`*, *`.bcc`*, *`.subject`*, *`.body`*, *`.attachmentName`*, and *`.copySelf>`*.

target.printer.copies	Specifies number of copies to print of the published document.
target.printer.scaletofit	Determines if pages are scaled to fit on the printable area, or are printed at the original size and cropped to the page.
target.printer.rotatetofit	Determines if pages are rotated to best fit on the printable area, or are printed with the original orientation.
LoaderProfile	Specifies the ID of a loader profile to use for the job.
OutputFormat	The format of the output file. Valid values are either XDL, CSF, PDF, TIFF, TXT, or NONE.
OutputFormatN	If two or more output formats are desired for one or more source documents, the different format extensions can be requested by supplying multiple *`outputformatN`* input parameters, where N = 0, 1, 2.
OutputFormatN.Filename	If two or more output formats are desired for one or more source documents, the names of the different output files can be individually specified using this parameter.
ForceSourceFormat	Defines the specific single source file format being used, rather than allowing the format to be detected through auto recognition.
ForceSourceFormatN	Defines the specific mult-isource file (where N=0, 1, 2....) format being used, rather than allowing the format to be detected through auto recognition.
NotificationUrl	The URL to call when the job is completed.
NotificationVerb	Http method to use for making calls to the Job Processor. The default value is GET. Optional value is POST or MULTIPARTPOST.
SecurityXmlFilename	Security settings for publication.
BannerXmlFilename	Banners and watermarks for publication.
FitWithinBanners	If `true`, the printed documents pages are scaled to fit within any banners that might be set.
CSFName	Specifies the output filename with extension.
OutputColorMode	Specifies color mode to use for published output (excluding banners): `fullcolor`, `grayscale`, or `monochrome`.
ExportPageList	A list of pages and page ranges to publish to the requested target format.
Rotate90	Page list of output pages to rotate counterclockwise by 90 degrees during export.
Rotate180	Page list of output pages to rotate counterclockwise by 180 degrees during export.
Rotate270	Page list of output pages to rotate counterclockwise by 270 degrees during export.

PublishBlankPages	If `false`, source pages that have no graphic or text data on them will not be included in the published document.
RetainAttachments	If `true`, attachments are retained as attached binary files to PDF output. If `false` (default), attachments are appended as additional pages.
PadOddPageDocuments	For multi-source jobs, when this parameter is set to `true`, a blank page is added at the end of each document if that document has an odd number of pages.
Export2up	Stitches two pages together, side by side, in the output PDf/TIFF.
Export2upLeadingBlankPage	Used with *`Export2up`*, inserts a single blank page at the start of a published PDF/TIFF.
HeartBeat	Full path to the HeartBeat file name. Each line in the file is formatted as: `[<Date Time>] [<Source File Name>] Msg`
StepTimeout	The maximum number of seconds allowed for any one step of the conversion process.
DocumentTitle	Title of the document, used when the `%title` macro is specified for an ISOBanner in the XML provided to the *`BannerXmlFilename`* job option.
ExportFilePerPage	Exports a single file for each page in the source document if this parameter is set to `true`.
PublishUserName	Specifies the user name who is publishing a file, used when the `%user` or `%login` macro is specified for an ISOBanner in the XML provided to the *`BannerXmlFilename`* job option.
Timeout	The number of seconds to allow a job to process.
Priority	A string representation of a 32-bit signed integer. Jobs are queued by their extension and then ordered by the integer value set by Priority from high to low.
ctbfile	Path to the custom CTB file to use when publishing AutoCAD files.
intloutputstringfile	This parameter allows each job to override the strings used by the Job Processor during markup burn-in and export. Refer to the sample file `sample_intloutputstrings.txt` for a list of valid values to use.
Comparison Parameter / Description	
GenerateTextComparison	When `true`, this parameter generates a PDF-formatted text comparison report showing differences between the textual content (only) of 2 source documents.

TextComparisonOldFileName	When *GenerateTextComparison* is `true`, specifies the document name to use for the "old" role (corresponding to *source0*) in the generated report. If not specified, the report will use the file name of the *source0* input parameter.
TextComparisonNewFileName	When *GenerateTextComparison* is `true`, specifies the document name to use for the "new" role (corresponding to *source1*) in the generated report. If not specified, the report will use the file name of the *source1* input parameter.
TextComparisonReportName	Defines a name to be given to the text comparison report document.
GenerateTextComparison.Color.Addition	Sets the color of the text used to denote additions in the text comparison report when *GenerateTextComparison* is `true`.
GenerateTextComparison.Color.Deletion	Sets the color of the text used to denote deletions in the text comparison report when *GenerateTextComparison* is `true`.
GenerateTextComparison.Color.Change	Sets the color of the text used to denote changes in the text comparison report when *GenerateTextComparison* is `true`.
TIFF Parameter / Description	
ForceTiffMonochrome	If `true`, TIFF output will be monochrome.
TiffDpi	Sets the DPI in the output TIFF file.
TiffBpp	Sets the bits-per-pixel to use for exported TIFFs. Valid values are 1, 4, max4, 8, max8, 24, and max24.
TiffCompression1Bit	Sets the type of image compression (CCITT, LZW, PACKBITS, or ZIP)to use for TIFF export pages that are 1-bit-per-pixel (monochrome).
TiffCompression4Bit	Sets the type of image compression (LZW, PACKBITS, or ZIP)to use for TIFF export pages that are 4-bits-per-pixel (16 color).
TiffCompression8Bit	Sets the type of image compression (LZW, PACKBITS, or ZIP) to use for TIFF export pages that are 8-bits-per-pixel (256 color)
TiffCompression24Bit	Sets the type of image compression (JPG, LZW, PACKBITS, or ZIP)to use for TIFF export pages that are 24-bits-per-pixel (full color).
tiffexporttransform.mirrorx	Specifies if the exported TIFF file should be mirrored in the x-axis (horizontally).
tiffexporttransform.mirrory	Specifies if the exported TIFF file should be mirrored in the x-axis (vertically).
tiffexporttransform.rotate	Number of degrees to rotate each page, counter-clockwise.

tiffexporttransform.pagelist	Specifies a page list of pages to transform when exporting to TIFF.
TiffExportMaximumDimension	Specifies a maximum dimension in pixels for TIFF export.
TiffExportByteOrder	Sets the desired byte order for TIFF export.
TiffExportRotateToFit	True indicates that pages of exported TIFF files should be rotated to a consistent orientation.
TiffExportRotateToFit.Orientation	If TiffExportRotateToFit is true, sets the orientation of the exported TIFF pages to landscape or portrait.
TiffExportRotateToFit.Direction	If TiffExportRotateToFit is true, sets the direction of rotation counter-clockwise or clockwise.
PDF Parameter / Description	
PdfExportA1b	If `true`, PDF output will be PDF/A-1b archive compatible.
PdfExportA2b **PdfExportA2u** **PdfExportA3b** **PdfExportA3u**	If `true`, PDF output will be PDF/A-<type> archive compatible.
PdfExportA1aTag **PdfExportA2aTag** **PdfExportA3aTag**	If `true`, a PDF/A-[#]a compliance tag is inserted in the output file.
PdfExportE	If `true`, PDF output will be PDF/E.
PDFExportAdjustLineWeightsForOutputSize	If `true`, PDF line weights will be scaled proportionately to the document scale.
pdfExportDisablePrint	Use `true` to disable printing of the exported PDF file.
pdfExportDisableChange	Use `true` to disable content modification of the exported PDF file.
pdfExportDisableSelectCopy	Use `true` to disable content copying of the exported PDF file.
PdfExportDisableAnnotation	Use `true` to disable adding or modifying text annotations in an exported PDF file.
PdfExportDisableFillFormFields	Use `true` to disable filling in interactive form fields in an exported PDF file.
PdfExportDisableAccessibility	Use `true` to disable extracting text and graphics for use by accessibility programs in an exported PDF file.
PdfExportDisableDocumentAssembly	Use `true` to disable insertion, rotation, and deletion of pages, and creation of bookmarks and thumbnails in an exported PDF file.

PdfExportDisableHighResPrint	Use `true` to disable high resolution printing in an exported PDF file.
pdfExportDisableAll	Use `true` to disable all of the PDF export security permission job options.
pdfExportEncryptionStrength	Set to either 40 or 128 to set encryption key bit length in Adobe Acrobat Reader.
pdfExportOpenPassword	Set a password value to set a Document Open Password when viewing the exported PDF file.
pdfExportSecurityPassword	Set a password value for changing security settings when viewing the exported PDF file.
pdfExportVersion	Sets the desired version of exported PDF files.
pdfExportMaxFileSize	Specifies the maximum size of an exported PDF file.
pdfExportEnforceRelativeHyperlinks	Use `true` to abort publishing if any source documents contain absolute paths within hyperlinks.
pdfExportHyperlinkBorder	Use `true` to add a visible border around hyperlinks.
PdfExportDisableLinkText	Controls if text links in a source document are disabled when exported to PDF. Can be set to `none`, `internal`, `external`, or `all`.
PdfExportColorOverrideLinkText	Controls if text links in a source document are exported in a different color when exported to PDF.
PdfExportColorOverrideLinkText.Color	Controls the override color to use with *`PdfExportColorOverrideLinkText`*.
PdfExportCMYK	Use `true` to use CMYK color space rather than RGB encoding in the resulting PDF.
PdfExport.ForceImageCompression.Color	Determines compression algorithm of color images as either `jpeg`, `flate`, or `default` in the exported PDF.
PdfExport.ForceImageCompression.Monochrome	Determines compression algorithm of monochrome images as either `ccitt`, `flate`, or `default` in the exported PDF.
PdfExportIncludeMetadata	Use `false` to prevent PDF metadata tags from being included in the exported PDF.
PDFExport.PageLayout	Determines the initial page layout to apply to an exported PDF file.
PDFExport.FontEmbed.SingleByte	Use **full** to embed single-byte fonts in the resulting PDF as a complete font, with all characters included.
PDFExport.FontEmbed.MultiByte	Use **full** to embed multi-byte fonts in the resulting PDF as complete fonts, with all characters included.
PDFExport.FontEmbed.Excluded	Specifies fonts to exclude from font embedding.
PDFExport.StripTagData	Use `true` to remove all tag data from the file during the export process.

PdfExport.Magnification	Sets the Adobe PDF magnification setting for a PDF export job.
AllowFontSubstitution	When `true`, missing fonts are substituted with a similar font. When `false`, missing fonts are not substituted and will cause the job to fail.
PdfExportMarkupsAsAnnotations	If `true`, markups* on PDF output files are saved as annotations. If `false` (the default), markups are burned into the PDF output and cannot be removed by users. *Redactions and Changemark behavior are not affected by this setting.
ExportContentAlignment	This parameter adjusts the placement of the TIFF/PDF content within the target page, anchoring it at one of the 7 possible locations.
PdfExportBookmarksInheritZoom	If `true`, Adobe uses the zoom scale of the current page when switching to the bookmarked page.
PdfExportBookmarkPanelState	Sets the initial state of the bookmark panel in an exported PDF file as either open, closed, or default.
PdfExportUseSourceBookmarkPanelState	True indicates to respect the state of the bookmarks panel setting in the PDF source file. If a panel state is set in the source file, it overrides the value set in *`PdfExporBookmarksPanelState`*.
PdfExportBookmarksExpandLevel	Sets the initial level of the bookmark tree that will be expanded during PDF export.
PdfExportRotateToFit	True indicates that pages of exported PDF files should be rotated to a consistent orientation.
PdfExportRotateToFit.Orientation	If *`PdfExportRotateToFit`* is true, sets the orientation of the exported PDF pages to landscape or portrait.
PdfExportRotateToFit.Direction	If *`PdfExportRotateToFit`* is true, sets the direction of rotation counter-clockwise or clockwise.
ExportScaleContentMode	Used for source documents where more control over the layout of content within PDF or TIFF document pages is desired, this option selects one of 3 possible content scaling modes: `ScaleToFitPage`, `ScaleToFitMargins`, or `ScaleToValue`
ExportScalePercentage	If *`ExportScaleContentMode`* is set to `ScaleToValue`, and this parameter is set to a floating point number> 0.0, the Job Processor will scale PDF and TIFF content by this factor, preserving the aspect ratio (1.0 leaves the original dimensions unchanged).
ExportScaleMarginUnits	If *`ExportScaleContentMode`* is set to `ScaleToFitMargins`, and this parameter is set to one of the legal values, then the values specified for `ExportScaleMarginTop/Bottom/Left/Right` will be taken as quantities of these units. cm = centimeters, in = inches, mm = millimeters, pt = typographic points, um = micrometers.

ExportScaleMarginTop/ Bottom/Left/Right	If *ExportScaleContentMode* is set to `ScaleToFitMargins`, and this parameter is set to a positive floating point number>= 0.0, the Job Processor adds this amount of margin between the (top/bottom/left/right) edge of the target document page and the same edge of the original content.
ExportLayerState	Determines layer state to publish document. Value can be `All`, `Visible`, or `None`
SetLayers.Hidden	Any layers in the exported document with names matching the regular expression are set to `hidden`.
SetLayers.Visible	Any layers in the exported document with names matching the regular expression are set to `visible`.
SetLayers.Toggled	Any layers in the exported document with names matching the regular expression are switched to `visible` or `hidden`, depending on their initial state in the file.
PersistTimeout	Number of seconds a persistent document will be kept in memory before being recycled.
MaxMessageCount	Maximum number of warnings or error messages written to the log file for each job.
ShowLineWeights	When `true`, AutoCAD Print Line Weights are supported when publishing to PDF or TIFF.
ShowThinLinesOnly	Use `true` to show only thin lines.
PdfFastWebView	Use `true` to enable Fast Web View.
ColorCollisionTolerance	Sets the color collision tolerance for PDF output. Range is from 0 to 255.
Page Size Parameter / Description	
OutputPageSize	Sets the page size by specifying output of type PDF and TIFF.
OutputPageSize.UseSourcePageOrientation	When `false`, the dimensions set by *OutputPageSize* are applied to every output page. When `true`, *OutputPageSize* size dimensions are applied, but orientation will match each source page.
CreatePageSizesTextFile	When `true`, then `PageSizes.txt` is created in the XDL file's output directory.
CreatePageSizesTextMetric	When `true`, the system will output metric page size information for XDL files.
ExpandToFitContent	Expands page size to include source elements beyond the limits as defined by the source page.
Markup Parameter / Description	
ExportMarkupFilename	Specifies the name of a file that will receive all of the markups created from redaction scripts and markup files.

markupN.pageShift	Used to shift markup page indexes before applying to source documents.
sourceM.markupN.pageShift	Used to shift multi-source markup page indexes before applying to source documents.
sourceN.markupM	Used to specify which source document which markups are applied to.
markupN	Used to specify paths to multiple markup files to be applied to final document.
sourceN.pagelist	Used to request that only specific pages should be included in the final document.
markup.failOnNoEntitiesApplied	Used to fail a job if markup file added to a source document results in no entities added.
ExportChangemarkSummary	If `true`, a Changemark summary page is appended to the PDF or TIFF output file.
ExportRTFChangemarkSummary	If `true`, a Changemark summary page in RTF format is published along with the output file.
docxInsertChangemarks	If `true`, inserts Text Highlight Changemarks on .docx files as Microsoft Word Comments.
ExportBlockAttributes	When `true` (the default), all block attributes are included in the exported document. Use `false` to omit them.
PdfNotesForRedactionBlocks	Redact For publishing to PDF only, when `true`, any redactions in the document appear as PDF notes with the redaction reason code placed within the note. The underlying text is outlined but still visible for review.
Redact*Redaction Parameter / Description*	
AuthorName	Specifies the name to use as the author name for all markups created by redaction scripts and zones.
OutputRedactionInfo	If `true`, enables the output of a redaction summary page as either an additional page in the output, or as a separate text file.
OutputRedactionInfoAsFile	If `true`, outputs a redaction summary page as a separate text file uniquely named `<target file name>.redactioninfo.txt`.
RedactionInfoLegend	Applies a legend to the output summary page determined by the *`OutputRedactionInfoAsFile`* or *`OutputRedactionInfo`* parameter. The value of this parameter is the text of the legend.
RedactionZonesFilename	Specifies a zone file name for redaction.
RedactionScriptFilename	A redaction script to apply when publishing a file.
RedactionColor	Sets the color of the redaction block outs. Can be a comma separated RGB value or a named color.

RedactionReasonFontName	Sets the font of the redaction reason codes placed over redaction blocks.
RedactionReasonColor	Sets the color of the redaction reason codes placed over redaction blocks.
RedactionReasonSize	Sets the desired font size of the redaction reason codes placed over redaction blocks.
RedactionReasonMinSize	Sets the smallest acceptable font size of the redaction reason codes placed over redaction blocks.
RedactionReasonBold	Set to `true` or `false` to indicate whether reason codes will be bolded.
RedactionReasonItalic	Set to `true` or `false` to indicate whether reason codes will be italicized.
RedactionReasonUnderline	Set to `true` or `false` to indicate whether reason codes will be underlined.
OutputRedactionAuditLog	If `true`, when publishing with a redaction script, outputs a redaction audit log file, named `<target file name>.redactionauditlog.xml` as an additional page.
RedactionAuditLogFilename	Specifies a file name to use for the audit log as `<target file name>.<redactionauditlogfilename value>.`
ShowRedactionReasonsOnBlockouts	If `false`, the output file will not have redaction reasons labeled on blockout entities.
ShowRedactionReasonsOnSummary	If `false`, the redaction summary page (either as part of the file or as a separate file) will not contain redaction reasons.
GeneratePrivilegeData	If `true`, an XML file is generated containing a summary of the number of redactions performed, the associated reason codes, the Bates numbering range (if applicable), and other descriptive information.
ReasonCodeDescriptionFilename	The path and file name of a Brava! Desktop formatted redaction reason code description file.
PrivilegeDataDirectory	The path to a directory where the Job Processor has access rights to create the XML files produced by the `GeneratePrivilegeData` setting.
PublishAsReviewDraft	If `true`, redactions will not be burned in, but will have their blockout regions made transparent so the underlying text can be viewed.
Thumbnail Parameter / Description	
ThumbFormat	Specifies the format to use for exported thumbnails. Valid values are JPG, PNG, or smallest.
ThumbName	The name of the thumbnail file, without an extension.
ThumbName.SupressDimensions	Use `true` to exclude the dimensions of the image in the generated thumbnail filename.
ThumbSizes	The size of the thumbnail (in pixels) to create, as x,y.

ThumbWidths **ThumbHeights**	Allows thumbnail requests by width or height only.
ThumbPages	The pages to create thumbnails for. Valid options are: `A`, `F`, or *<page number>*
ThumbQuality	The quality (1-100) of the thumbnail JPEG .
ThumbRotateToFit	True indicates that thumbnails should be rotated to a consistent orientation.
ThumbRotateToFit.Orientation	If ThumbRotateToFit is true, sets the orientation of the thumbnails to landscape or portrait.
ThumbRotateToFit.Direction	If ThumbRotateToFit is true, sets the direction of thumbnail rotation counter-clockwise or clockwise.
Get Text Parameter / Description	
GetText	Instructs the server to produce a set of text files with all of the text from the published document extracted as plain text.
AddMetadataToTextOutput	If both this and the *`GetText`* parameter are `true`, then any metadata present in the source file will be included in the text output pages.
TextFormatUTF8	If `true`, then the output format produced from *`GetText`* is UTF-8 encoded.
ExtractExifData	If `true`, Exif data is output from JPG files into the metadata output file. *`AddMetadataToTextOutput`* must also be `true`.
TextFilePrefix	Use to specify a non-default prefix for the text or text with position output files. Each string can contain the **case sensitive** substitution string ${source}.
textformatcrlf	Set to `true` if you want text output to be formatted with carriage returns separating each line.
TextSortMode	Sets how text is arranged during source document loads. Use `columns`, `topdown`, or `none`.
OCR Parameter / Description	
DoOCR	If `true`, TIFF, JPG, BMP, GIF, PNG, and PDF files are processed through OCR.
BypassOCRForFileFormat	Provides an optional list of file formats that will not be processed through OCR when *`DoOCR`* is `true`.
ocr.Timeout	The longest time, in seconds, that the OCR engine will spend attempting to recognize characters in any one image.
ocr.LanguageDictionaryCorrection	A ranked list of countries/languages representing the language content of the document, if known.

ocr.MinConfidence	The level of confidence that recognition must achieve for any one character to be considered successfully recognized.
ocr.BelowconfidenceThreshold	The maximum number of characters out of every 100 evaluated that can fail recognition with a confidence less than *`ocr.MinConfidence`*, but greater than zero, before failing the job.
ocr.ErrorThreshold	The maximum number of characters out of every 100 evaluated that can fail recognition with a confidence of 0 before failing the job.
ocr.ignoreErrors	Determines whether ocr errors are ignored and do not fail the publishing job.
Stamp Parameter / Description	
StampImage	Path to the image file to stamp, using the Stamp feature. It must be a UNC path to a JPG or PNG file.
StampPositionAndSize	Multiple values that specify the size and position of the stamp image (XSP file). The string parameter format is: `StampPositionAndSize=left|top|scale|page[|page][|page]...[|page]`
StampTemplateFileName	The UNC path to a Stamp Template file.
StampTemplatePositionAndSize	Multiple values that specify the size and position of the stamp template. The string parameter format is: `StampTemplatePositionAndSize=left|top|scale|page[|page][|page]...[|page]`
CSF Writer Publishing Parameter / Description	
PrintJobTimeout	The amount of time, in seconds, to allow the CSF Writer publishing step to complete.
LinksTimeout	The amount of time, in seconds, to allow CSF Writer publishing to extract links.
OutlookMessageFormat	Determines the format to which Outlook messages will be converted. `0=MSG`, `1=Plain Text`, `2=Rich Text`
WordRevisionsMode	Determines whether Microsoft Word displays revisions in balloons in the margin, or inline with the document's text. `0=balloon`, `1=inline`
ExcelResetPrintRegion	If `true`, overrides any print regions defined in Excel documents and uses the default print region for publishing
ExcelShowHiddenRows	If `true`, hidden rows from Excel documents are published as visible.
ExcelShowHiddenColumns	If `true`, hidden columns from Excel documents are published as visible.
ExcelShowHiddenSheets	If `true`, hidden sheets, rows, and columns from Excel documents are published as visible.

ExcelPageOrientation	Controls whether Excel pages are published in portrait or landscape orientation.
ExcelPageOrder	Controls whether pages within a given sheet are published across the sheet first and then down, or down first, then across. Values are `OverThenDown` or `DownThenOver`.
ExcelFitToPage	If `true`, data from published pages are shrunk to fit the published page dimensions.
ExcelFitToPageHeight	If `true` and *`ExcelFitToPage`* is `false`, rows from published pages are shrunk to fit the published page height.
ExcelFitToPageWidth	If `true` and *`ExcelFitToPage`* is `false`, columns from published pages are shrunk to fit the published page width.

Chapter 10

Copyright Notices and Acknowledgements

This software includes third-party component software distributed by OpenText to you pursuant to specific third-party license agreements, whose terms and conditions are as set forth in your license agreement with OpenText and/or the Terms and Conditions of Embedded Products. Copies of such Embedded Software Licenses relating to the use and distribution of such Embedded Products can be found in the `\OpenText\Brava! <product>\Licenses` directory located in the product install directory. You agree to comply with all such Embedded Software Licenses which apply to the Software licensed to you by OpenText.

Please refer to the **LegalNotices.txt** document included in your product installation to view all third-party software copyright notices and acknowledgements that are associated with this product and its components.