Variable Data Printing

DeskPack Workflow

ESKO

Contents

1. Introduction

Variable Data Printing provides you the ability to print jobs with variable elements on digital presses while keeping in mind the needs of the packaging industry. A variable element can either be text, a barcode or an image.

This document will explain only those topics related to Variable Data Printing. Knowledge of Esko BackStage Pilot, Esko FlexRip, Esko DeskPack and SmartMarks is required.

1.1 Copyright Notice

Correspondence regarding this publication should be forwarded to:

Esko Software BVBA

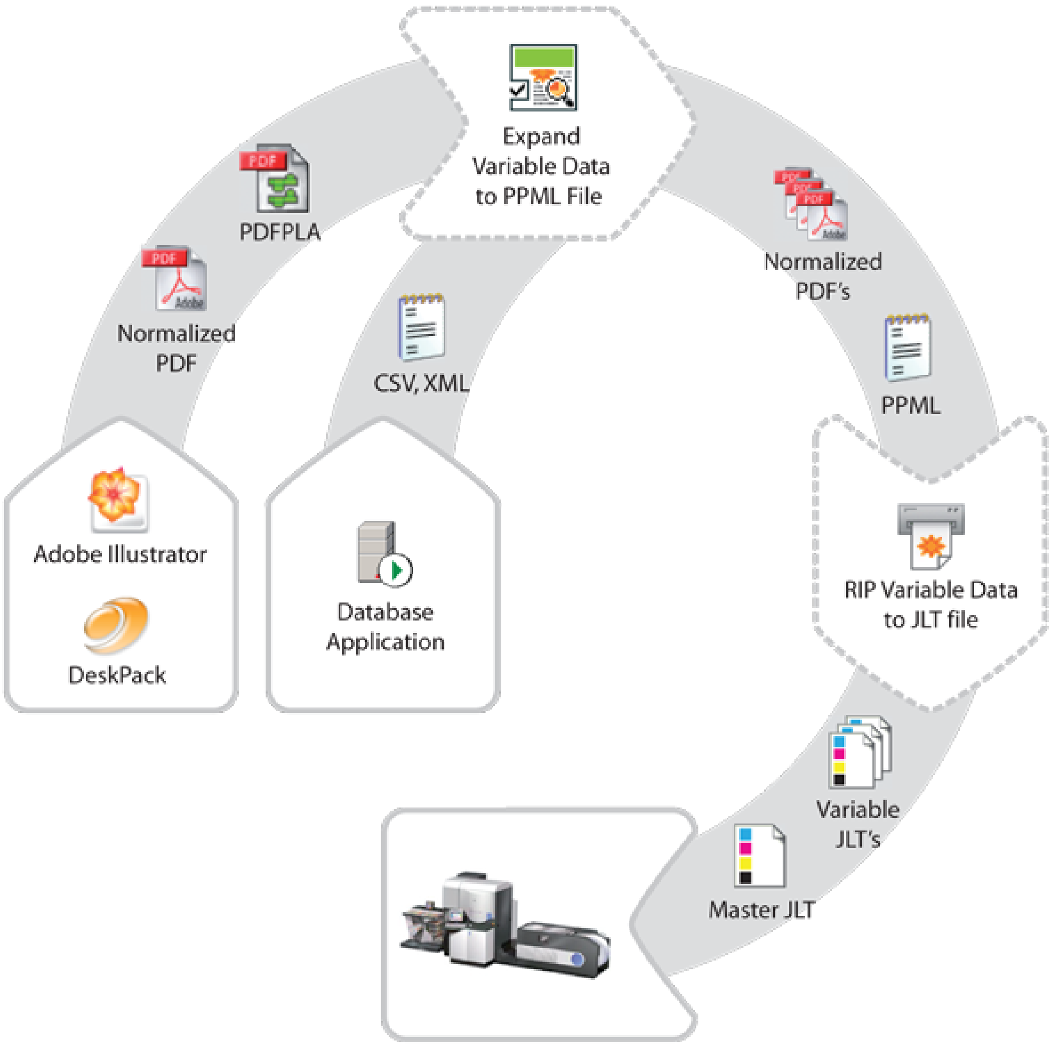
Kortrijksesteenweg 1095

B – 9051 Gent

info.eur@esko.com

2. Components

The Variable Data Printing module consists of a front end, a database and some BackStage server tasks.



2.1 Front End

The front end to the system consists of Adobe Illustrator and a set of DeskPack plug-ins. The Variable Data plug-in is used to define variable elements within an open Illustrator document. For this purpose, the concept of SmartMarks was introduced.

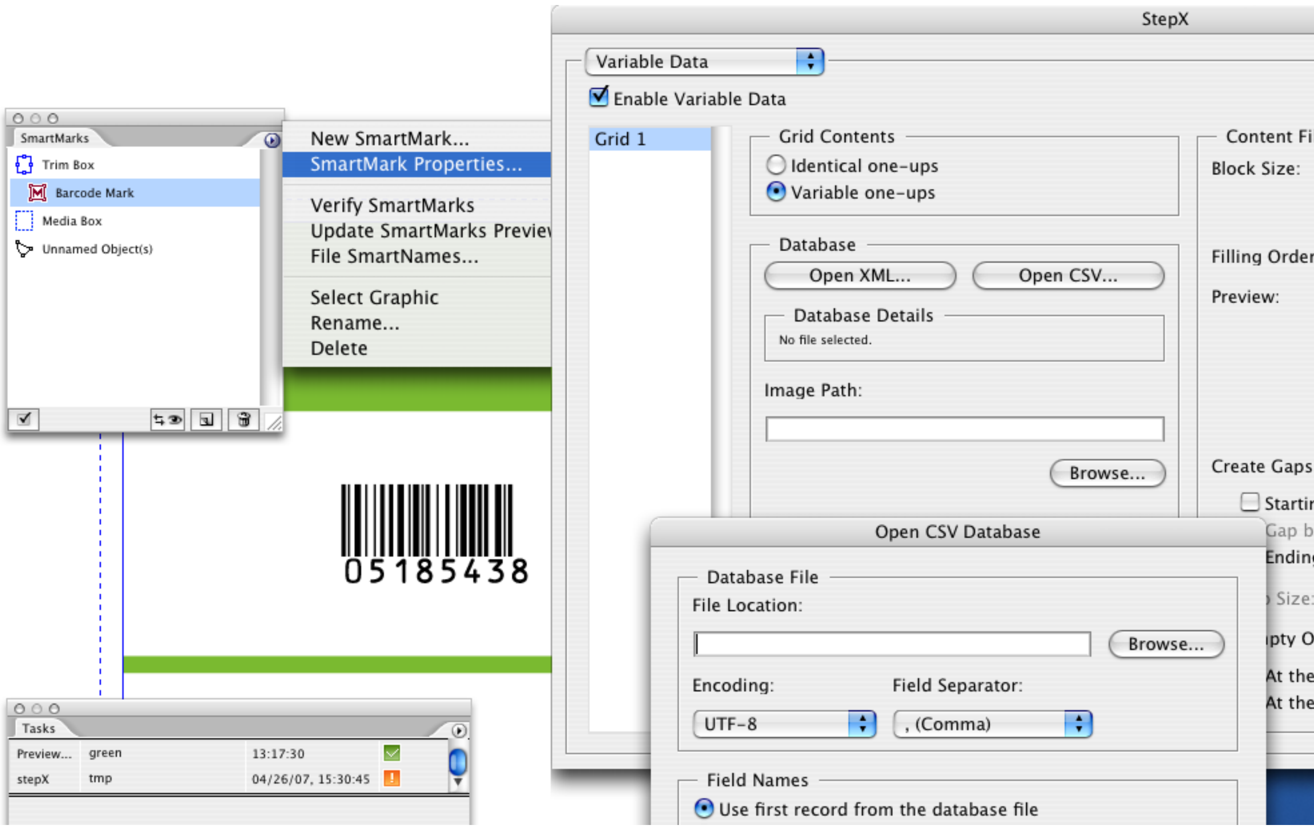
2.1.1 Introducing SmartMarks

SmartMarks are objects in the design that are linked to the database entries, and are generated based on their contents. In other words, for various database records various SmartMark objects are generated.

A SmartMark can be of three different types – a Text Mark, a Barcode Mark or an Image Mark:

- A Text Mark contains a text value and implements tools for its formatting;
- A Barcode Mark carries a bar code; and
- An Image Mark is used to place an image into the design, based on its file location.

As variable elements are generated on the server side, the plug-in needs to cooperate tightly with the server, where all the graphic objects represented by SmartMarks are created. For this reason, a SmartMarks Preview task was introduced; its main purpose is to update the SmartMark graphics objects and place them correctly into the design, to provide a comprehensive preview during the actual designing process.



Variable Data

[✓] Enable Variable Data

Identical one-ups [✓ Variable one-ups]

Encoding: UTF-8

Field Separator: , (Comma)

[✓ Use first record from the database file]

Preview...	green	13:17:30
stepX	tmp	04/26/07, 15:30:45

2.1.2 stepX for repetitions

For repetitions the stepX plug-in is used. In stepX, you can specify general repetition parameters, such as plate and sheet size, horizontal and vertical steps, as well as variable-data-specific parameters, such as the database parameters, ordering pattern, and gaps generation.

2.1.3 Front end output

The output of the front end consists of:

- A repetition file (PDFPLA file); and
- A one-up file (Normalized PDF file).

Server tasks are started via the stepX plug-in or the BackStage Pilot application. A more detailed description of the front end and its features can be found in the chapter on the DeskPack Workflow.

2.2 Database

CSV and XML database formats are supported. Both formats must be stored in UTF-8 encoding. CSV and XML files can be aquired from almost any database application, and as a consequence, other proprietary file formats are not supported.

2.2.1 Field name requirements

When composing database field names you are restricted to alphanumerical characters or underscores. The first character of a field name should not be a number. The interpretation of field names is case insensitive. Uppercase letters will always be converted to lowercase during import.

2.2.2 XML document structure requirements

The XML document structure should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<rootElem>
    <recordElem>
        <fieldName1>FieldValue-1-1</fieldName1>
        <_fieldName2> FieldValue "1-2"</fieldName2>
    </recordElem>
    <recordElem>
        <fieldName1></fieldName1>
        <_fieldName2>FieldValue-2-2</fieldName2>
    </recordElem>
</rootElem>
```

The names of the XML elements are arbitrary, but the following rules must be observed:

- Field name elements must conform to the rules for valid field names listed above.
- The amount and type of field name elements must be the same for each record.

Note:

The use of element attributes in the XML database is not forbidden, but they are not interpreted by our solution in any way.

A correct XML file can easily be exported from Microsoft Access. Microsoft Excel XML files are not supported.

2.2.3 CSV format requirements

An example of the CSV format for the same database is provided below:

```
fieldName1,_fieldname2
FieldValue-1-1,"FieldValue ""1-2"""
,FieldValue-2-2
```

Aside from the standard rules creating CSV formatted files, the following must be taken into account:

- Field names must conform to the rules for valid field names listed above.
- Each record (line) must have the same number of fields, equal to the number of field names.
- Newline characters in field values are not supported (if you want to use a newline character in Text Mark, use the standard '[nl]' control sequence).

Field names can be extracted either from the first record of the CSV file, or can be provided in a separate CSV file. The delimiter character can be a comma, semicolon, space or a tab character, with the comma being the default.

A valid CSV file can easily be exported from either Microsoft Access or from Microsoft Excel.

Note:

If a database contains regional characters, files exported from Microsoft Excel have to be converted to **UTF-8 encoding** (e.g. using Notepad), as Microsoft Excel does not support UTF-8 encoding for CSV files.

Microsoft Excel for Mac is not able to export some of the regional characters into CSV format, which might lead to loss of data if it is used for generating the CSV database file.

2.3 BackStage server tasks

The BackStage server offers three tasks to handle variable data: SmartMarks Preview, Expand Variable Data to PPML File, and RIP to WS4000 Series.

2.3.1 SmartMarks Preview

The SmartMarks Preview task generates the variable graphic objects at the stage of designing a variable data document in Illustrator. This task can be executed only from the Variable Data plug-in, and cannot be found in the BackStage Pilot application.

2.3.2 Expand Variable Data to PPML File (DeskPack only)

The Expand Variable Data to PPML File (DeskPack only) task performs expansion of the specified database entries into the corresponding graphical output.

The task takes the following input files:

- a design file containing variable elements (Normalized PDF file);
- a repetition file generated by the stepX plug-in (PDFPLA file);
- and a database file (XML or CSV).

It generates the following output files:

- one file with constant (unchanging) graphic elements (Normalized PDF file)
- several files with variable graphic elements (Normalized PDF files).

All these files are linked together by a standard PPML file.

2.3.3 The Rip to WS4000 Series task

The RIP to WS4000 Series task triggers FlexRip to create HP Indigo proprietary JLT files.These files can be read only by an HP Indigo press. The main difference compared to the generic RIP tasks is that this task supports PPML files, which is essential for correct variable data interpretation.

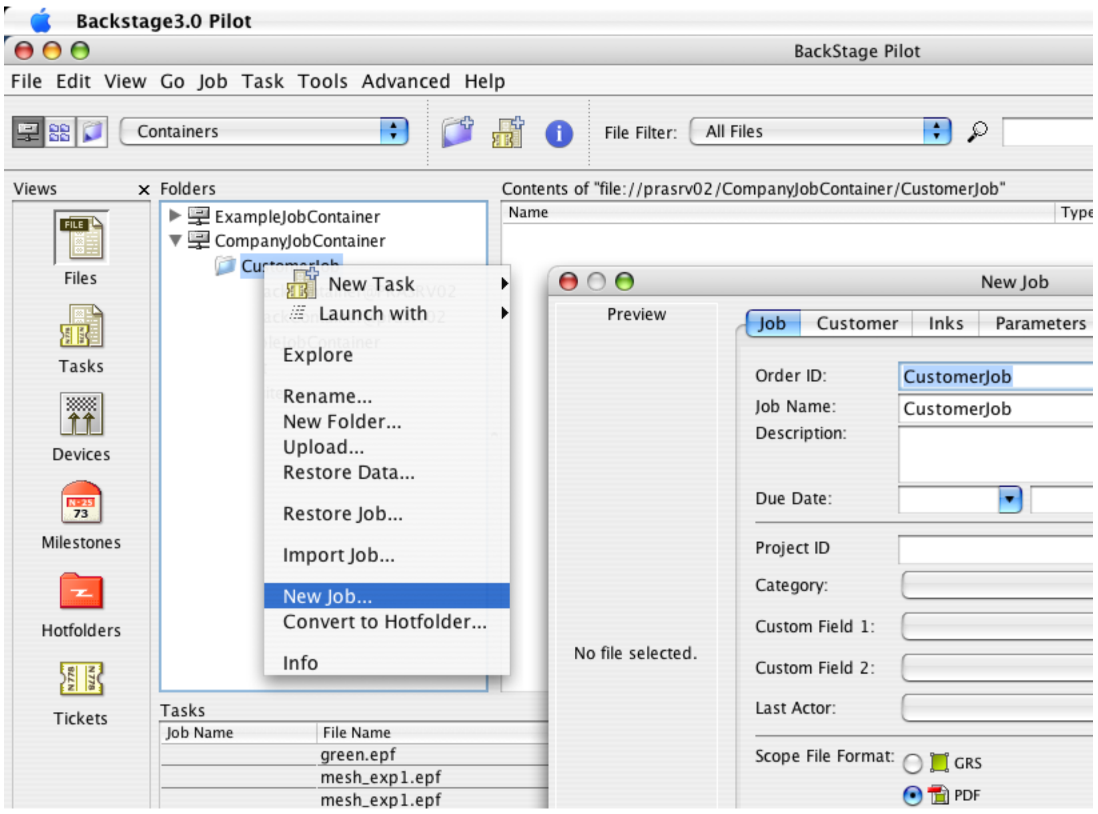
3. DeskPack Workflow

There are several ways to prepare documents and data for Variable Data Printing. However, it is recommended to follow this workflow:

1. *Create a job*.
2. *Prepare data*.
3. *Design a one-up in Adobe Illustrator*.
4. *Step & Repeat and launch server tasks*.

3.1 Create a Job

In the BackStage Pilot, create a new job or use an existing job, which will be used to encapsulate the variable data resources. This job should contain a one-up document containing the design with variable elements definitions, and all the documents which are created during the workflow process (such as a stepX preferences file, PDF files for both fixed and variable parts of the job and a linking PPML file). In addition, input data (database files and images) is also often stored in the Jobfolder.



File Filter: All Files

Contents of "file://prasrv02/CompanyJobContainer/CustomerJob"

Order ID: CustomerJob

Job Name: CustomerJob

Scope File Format: GRS [✓ PDF]

Tasks

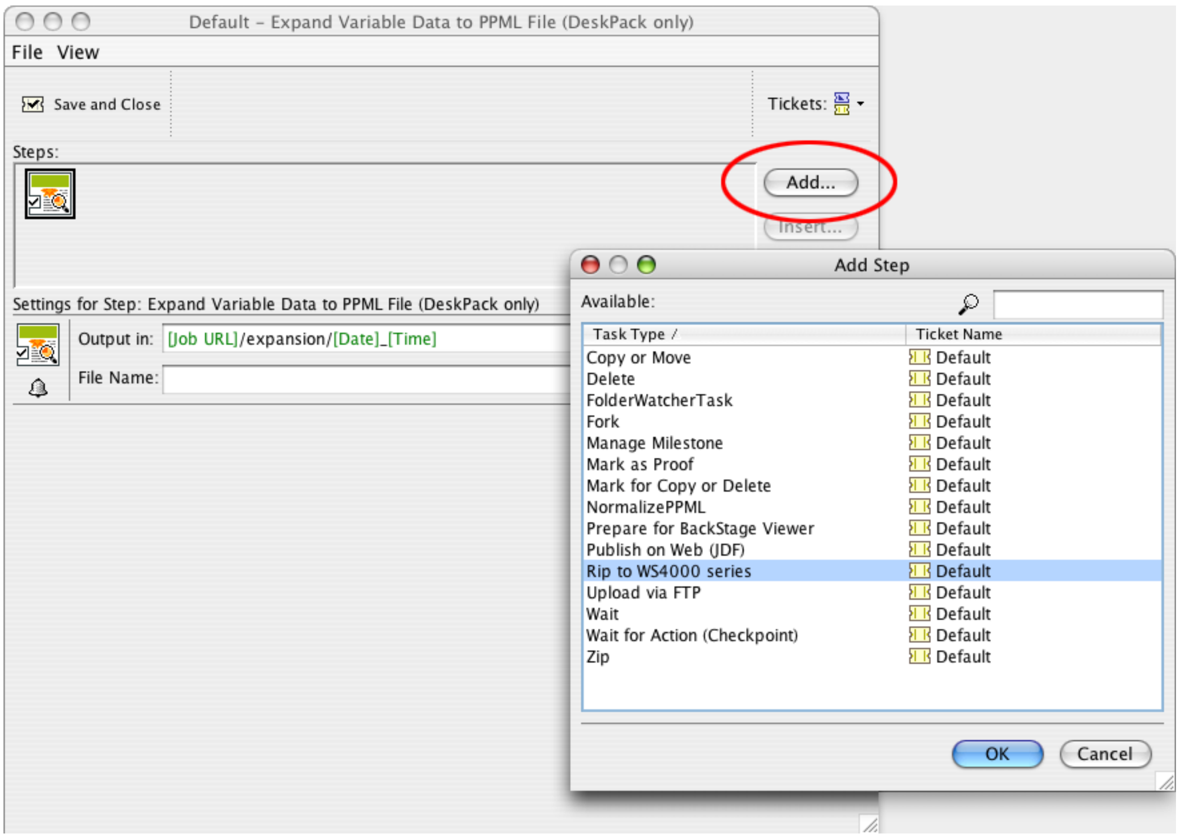
Job Name	File Name
	green.epf
	mesh_exp1.epf
	mesh_exp1.epf

3.2 Prepare the data and task tickets

The next step is to prepare the BackStage server task tickets that will be used for variable data expansion and consequential ripping.

3.2.1 Expand Variable Data to PPML File

The first task to consider is the 'Expand Variable Data to PPML File (DeskPack only)' task, which is the first task in the workflow chain. It is executed directly from within stepX. In most cases, the expansion task is followed directly by the ' RIP to WS4000 Series' task, so the use of a chained workflow ticket for both tasks is recommended. However, independent use of the expansion task is also possible (i.e. in case you only care about the expansion output, but do not want to run the RIP task afterwards).



Settings for Step: Expand Variable Data to PPML File (DeskPack only)

Output in: [Job URL]/expansion/[Date]_[Time]

File Name:

Task Type /	Ticket Name
Copy or Move	Default
Delete	Default
FolderWatcherTask	Default
Fork	Default
Manage Milestone	Default
Mark as Proof	Default
Mark for Copy or Delete	Default
NormalizePPML	Default
Prepare for BackStage Viewer	Default
Publish on Web (JDF)	Default
Rip to WS4000 series	Default
Upload via FTP	Default
Wait	Default
Wait for Action (Checkpoint)	Default
Zip	Default

You should specify an output folder for the 'Expand Variable Data to PPML File (DeskPack only)' task, otherwise the files are written into the DeskPack container and an error may occur.The Jobfolder is a suitable location for this, and since you can use SmartText in the folder location, you can create a path similar to this: [joburl]/expansion/[date]_[time].

Please note that the contents of this folder must be maintained manually as the files are not deleted automatically.

3.2.2 Rip to WS4000 Series: Configuration

When the expansion task finishes, the 'RIP to WS4000 Series' task can be started to launch FlexRip. There are only two minor differences in the ticket dialog compared to a generic rip or proof task.

Note:

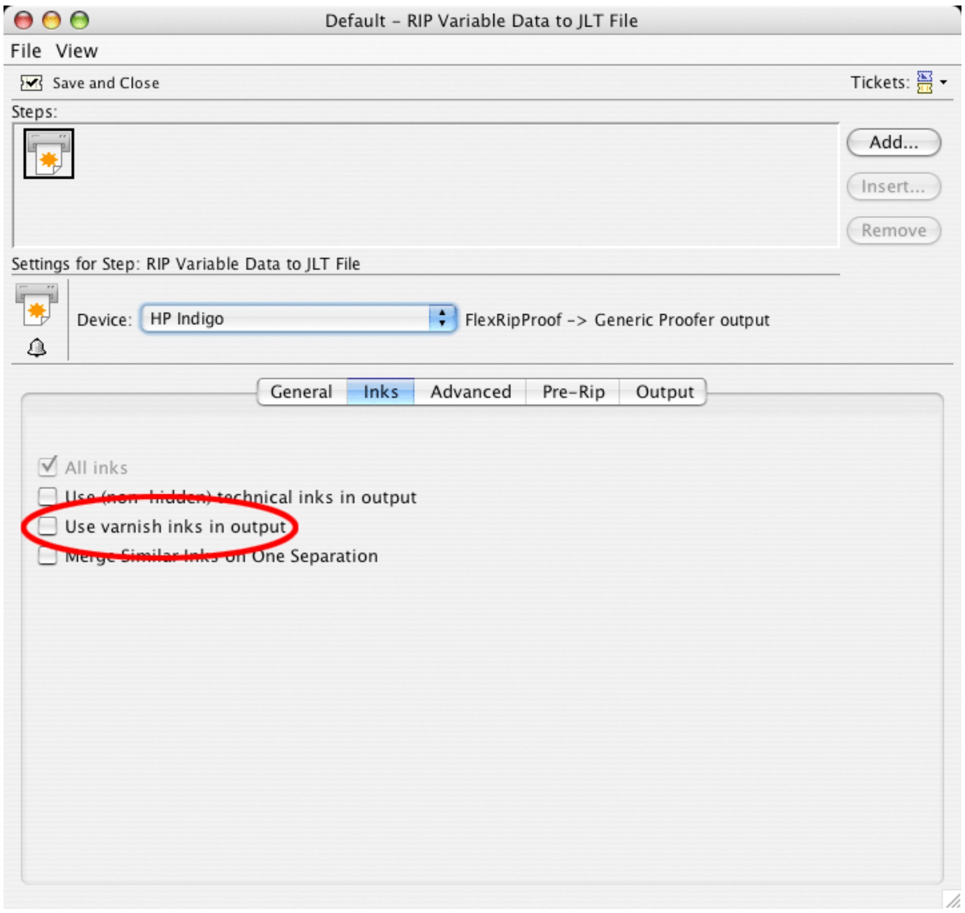
For a description on all the options of the RIP task, please refer to the Esko BackStage documentation.

If the RIP to WS4000 Series task finishes successfully, the temporary files from the expansion task can be deleted. These files are not deleted automatically.

Inks Tab

The Inks tab of the Rip to WS4000 Series task ticket contains a variable data printing-specific option, **Use varnish inks in output**.

The **Use varnish inks in output** option enables or disables interpretation of varnish-type inks by the RIP.


Default – RIP Variable Data to JLT File

Settings for Step: RIP Variable Data to JLT File

Device: HP Indigo FlexRipProof -> Generic Proofer output

[✓] All inks
[ ] Use (non-hidden) technical inks in output
[ ] Use varnish inks in output
[ ] Merge Similar Inks on One Separation

Output Tab

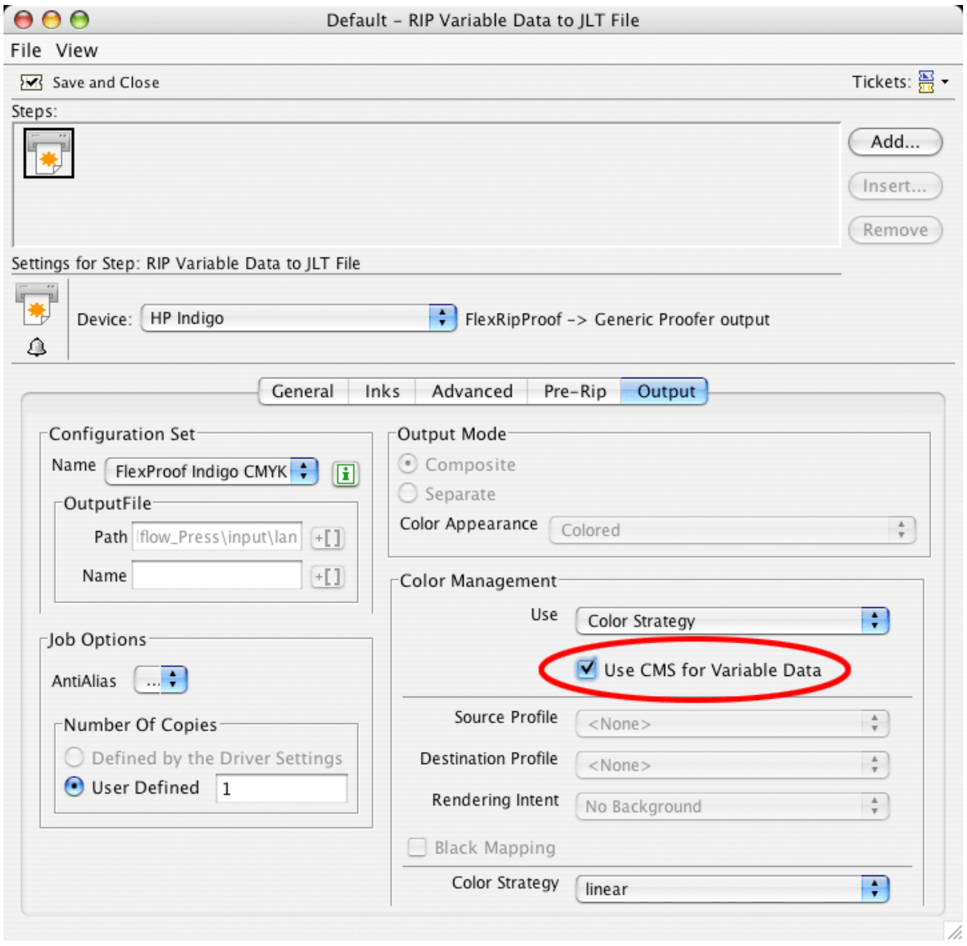
The Output tab of the Rip to WS4000 Series task contains the variable data printing-specific option **Use CMS for Variable Data.**

The **Use CMS for Variable Data** option enables the use of color management for the variable data elements in a design, instead of only color managing the fixed background.

If the variable data are put in a Pantone or a designer ink and the option is turned off, the inks will be converted by using the color strategy 'linear'.

Note:

Turning on color management for variable data slows down the performance significantly.


Default – RIP Variable Data to JLT File

Settings for Step: RIP Variable Data to JLT File

Device: HP Indigo FlexRipProof -> Generic Proofer output

Configuration Set Name: FlexProof Indigo CMYK

OutputFile Path: flow_Press\input\lan

OutputFile Name:

Job Options AntiAlias: ...

Number Of Copies: Defined by the Driver Settings [✓ User Defined] 1

Output Mode: Composite / Separate

Color Appearance: Colored

Color Management Use: Color Strategy

✓ Use CMS for Variable Data

Source Profile: <None>

Destination Profile: <None>

Rendering Intent: No Background

Black Mapping

Color Strategy: linear

Known Issues in the RIP

The following are known issues when ripping variable data printing:

Only PostScript and Opaque overprint modes

The overprint modes of the variable part on the fixed part are limited to PostScript overprint and Opaque. This is caused by the fact that the 2 parts are combined on the press, and not on the system. However inside the fixed part of the job all overprint or transparency settings are supported.

Unsupported transparacies in variable layer

Using unsupported transparencies in the variable layer may result in parts of the variable objects not being printed.

No JLT viewer available

Currently no JLT viewer is available. The only way to check ripped data is to view them at HP Indigo press, using the Job Editor.

Files not deleted automatically

Files stored in the 'images' folder at the press PC are not deleted automatically. Make sure that the total number of files located in this folder does exceed 3 000, otherwise operations within this folder may become slower.

CMS for variable data reduces performance

When CMS is involved for the variable part of job, performance is reduced significantly.

No Magenta1 layer for variable data

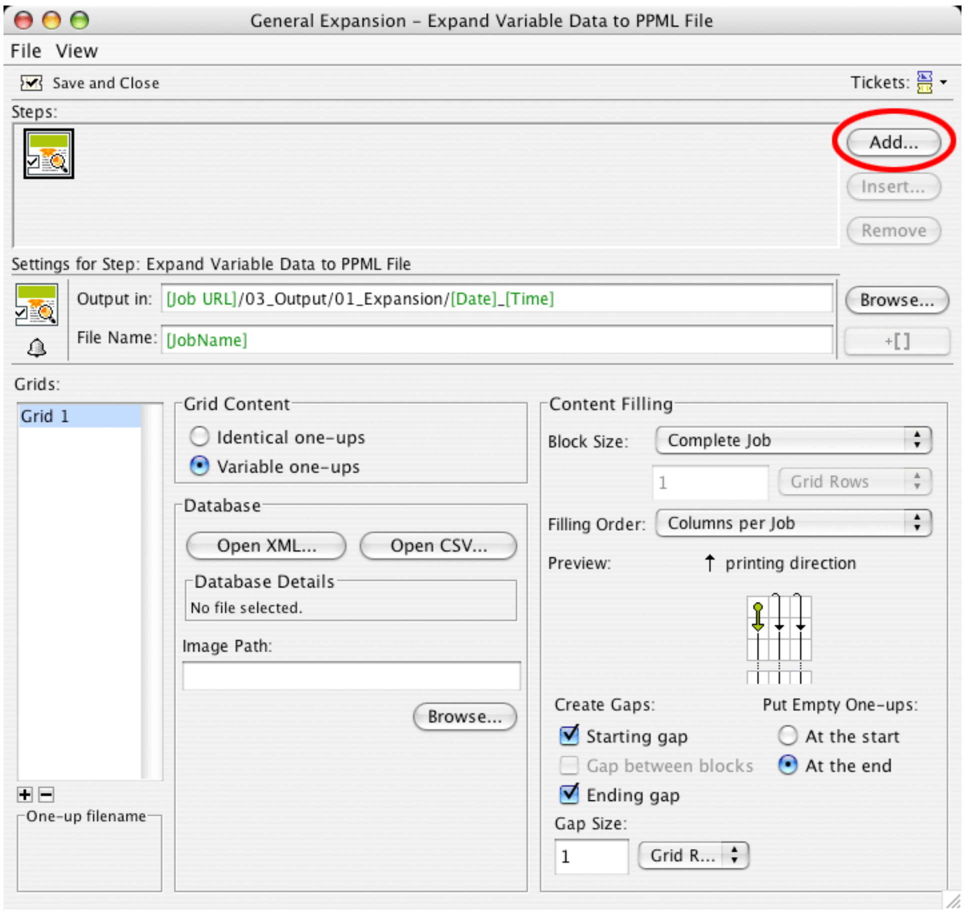
A Magenta1 separation (M1) is generated when Magenta overlaps with an Orange, Violet or Green separation. Using M1, however, is not supported for the variable layer. M1 will never be generated for the variable layer.

3.2.3 Chaining the Tasks

Two or more tickets can be connected in a chain. Consult the Esko BackStage 7.0 User Manual for more detailed instructions.

To chain the variable data printing tickets:

1. Open the Expand Variable Data to PPML File (DeskPack only) task ticket.


General Expansion – Expand Variable Data to PPML File

Settings for Step: Expand Variable Data to PPML File

Output in: [Job URL]/03_Output/01_Expansion/[Date]_[Time]

File Name: [JobName]

Grids: Grid 1

Grid Content: Identical one-ups [✓ Variable one-ups]

Database Details: No file selected.

Image Path:

Content Filling

Block Size: Complete Job

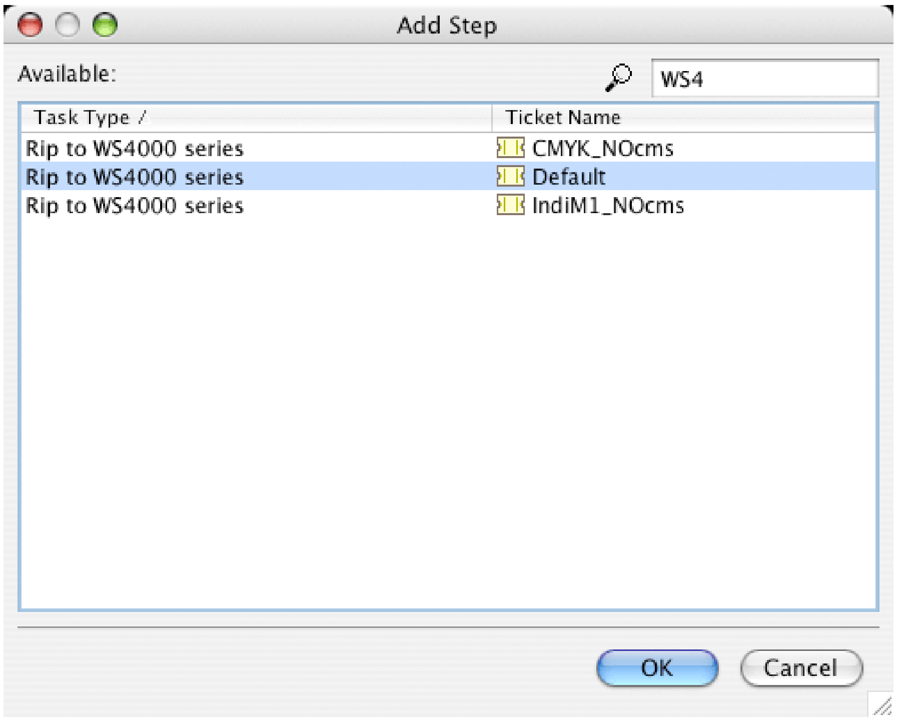
Filling Order: Columns per Job

Create Gaps: [✓ Starting gap] Gap between blocks [✓ Ending gap]

Put Empty One-ups: At the start [✓ At the end]

Gap Size: 1 Grid R...

2. Click the Add… button.
   The Add Step dialog box appears.
3. Select the Rip to WS4000 Series task ticket from the Task Type list in the Add Step dialog box.



Task Type /	Ticket Name
Rip to WS4000 series	CMYK_NOcms
Rip to WS4000 series	Default
Rip to WS4000 series	IndiM1_NOcms

4. Click OK.
The chain has been created and you can now save the workflow ticket.

3.3 Design a one-up in Adobe Illustrator

A design for Variable Data Printing consists of two main parts: fixed graphic elements in the background and variable elements placed on top of them. As merging variable and fixed parts is not performed by the BackStage server, placing fixed graphics on top of the variable graphics is not supported.

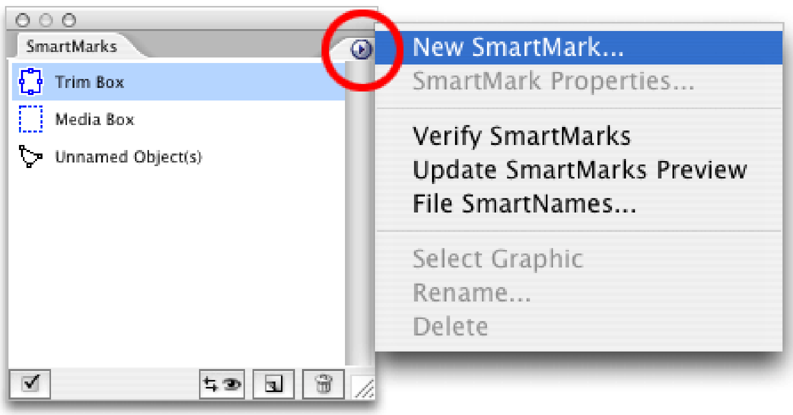
3.3.1 Design requirements

While designing the document, the following rules have to be obeyed:

- The fixed background elements must not be located in the layer named Variable_Data.
- The SmartMarks placed in the Variable_Data layer must not be edited using the built-in Illustrator tools (except of the selection tool which can be used to reposition SmartMarks).
- The Variable_Data layer always remains the top layer in the document.

3.3.2 Create a SmartMark

To create SmartMarks, go to Window > Esko > Variable Data to launch the Variable Data plug-in.The SmartMarks palette, which is the main working area of Variable Data appears.



You can only create a SmartMark object if the file contains trim and media box definitions. Trim and media box definitions can be added using the DeskPack Trim Box and Media Box plug-in.

Also, keep in mind that at least one ink should be used in the document's fixed part, so that this ink can be used by a SmartMark object.

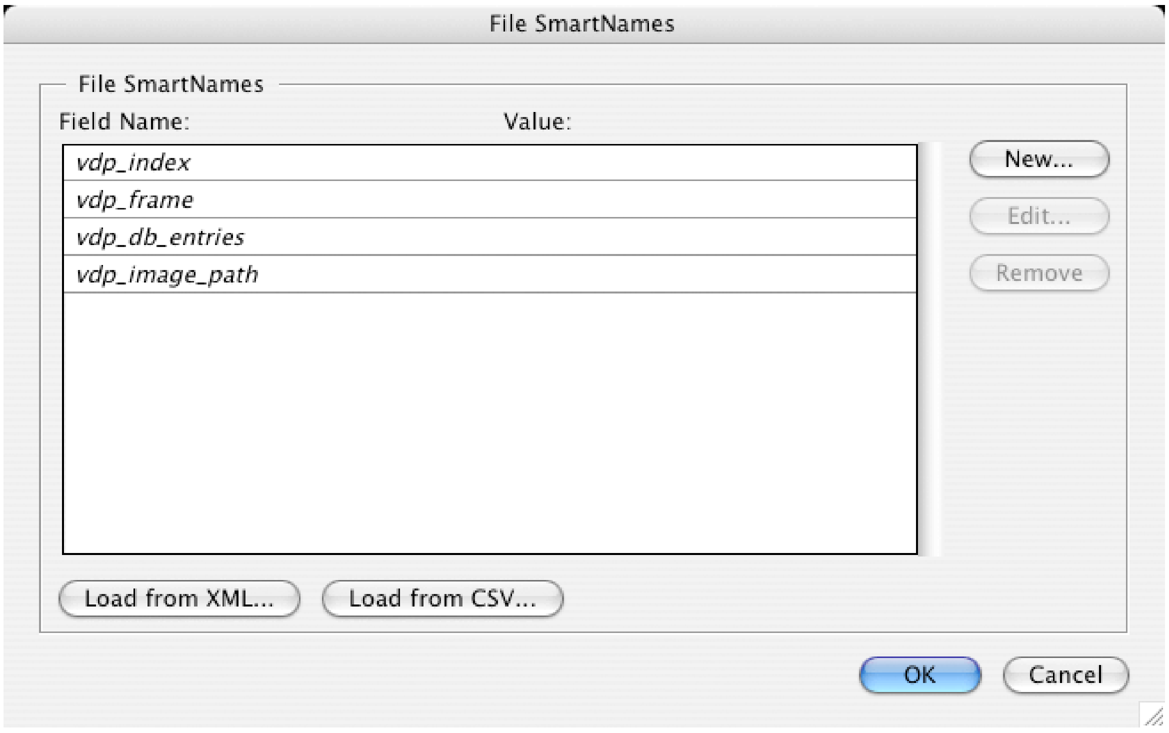
3.3.3 Defining File SmartNames

Select File SmartNames… from the SmartMarks palette's menu to define File SmartNames. File SmartNames provide a link between the SmartMarks' values and the actual database values that are used during the expansion. Designers do not need access to the real database used for printing. However, they must be able to obtain the correct field names, since these are essential when identifying the specific database columns.

You can either:

- Use a dummy database file; or
- Enter the field names and preview values manually.

Using a dummy database file

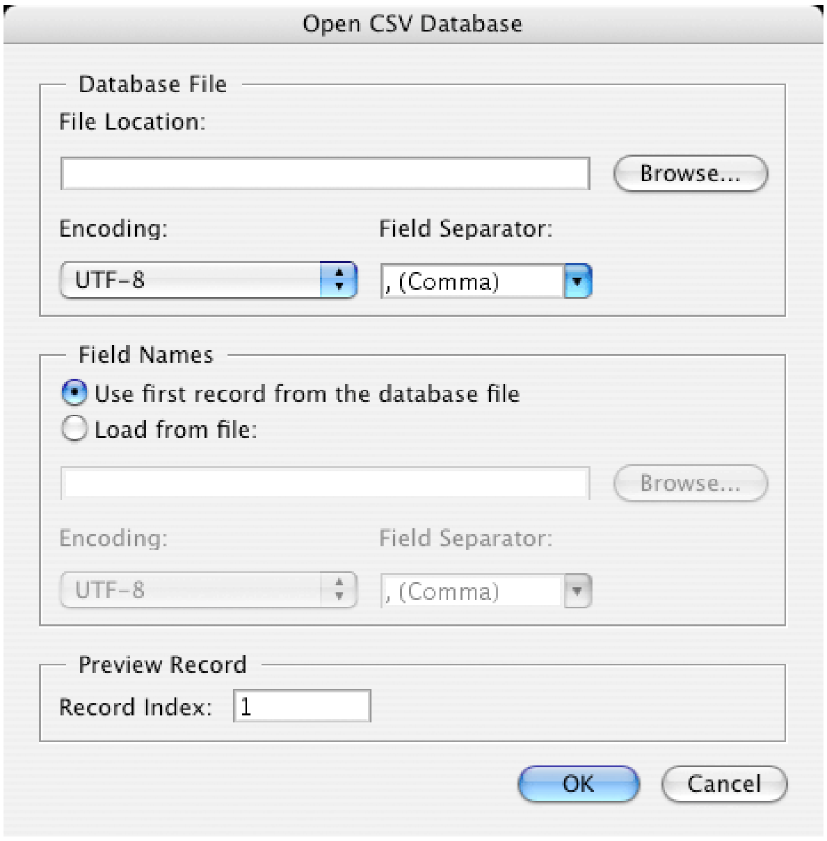


vdp_index
vdp_frame
vdp_db_entries
vdp_image_path

Typically, the field names along with some preview values (the values used while designing the one-up) are loaded from a database file:

- Click **Load from XML...** to load an XML database file.
- Click **Load from CSV...** to load a CSV formatted file.

For XML files, only the file location and record index are specified. CSV format files need additional information concerning the field separator, and the possibility of field names being defined in a separate CSV file.


Open CSV Database

Database File

File Location:

Encoding: UTF-8

Field Separator: , (Comma)

Field Names

[● Use first record from the database file]

○ Load from file:

Encoding: UTF-8

Field Separator: , (Comma)

Preview Record

Record Index: 1

Defining File SmartNames manually

You can also manually define File SmartNames by clicking the 'New...', 'Edit…' and 'Remove' buttons. However, be careful not to misspell the column names, because during expansion only the entries in the database that match these field names will be processed.

Very long SmartName values may be truncated in the dialog box, but this has no influence on reading SmartNames from database files and the expansion will handle long values correctly. Please read the chapter on the Database before you submit your first database.

Apart from the user-defined SmartNames, the following SmartNames are available:

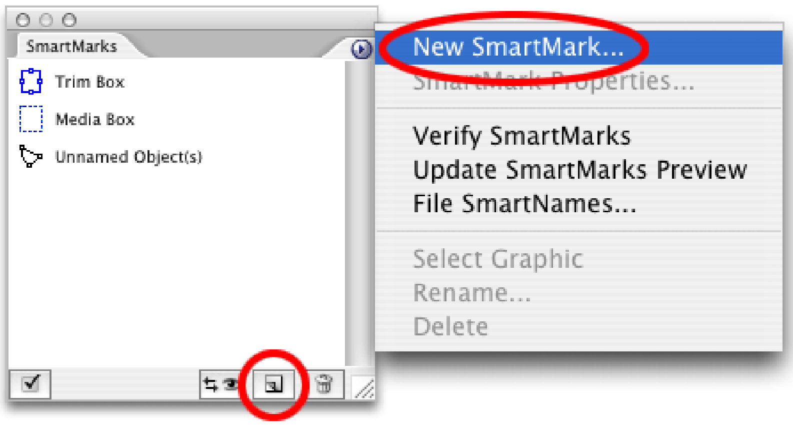
- vdp_index: returns the number of the used database record, starting with 1.
- vdp_frame: returns the number of the current frame generated by the expansion task.
- vdp_db_entries: returns the total count of all entries in the current database file.
- vdp_image_path: returns a path to a folder (see Image Mark for more information).

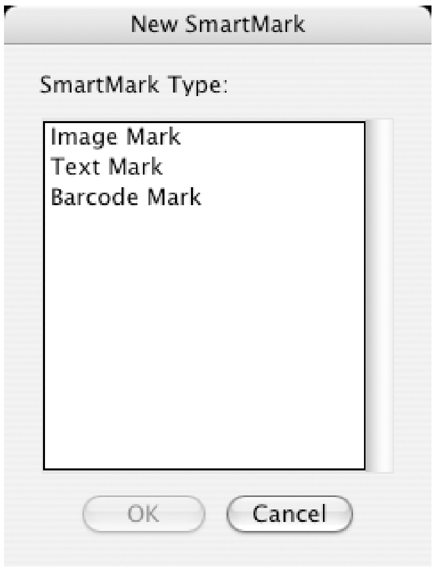
3.3.4 Inserting new SmartMarks

Once we have File SmartNames defined, we can create a first SmartMark.

To create a new SmartMark select the 'New SmartMark…' item from the fly-out menu of the palette, or click the shortcut icon in the bottom area of the palette.

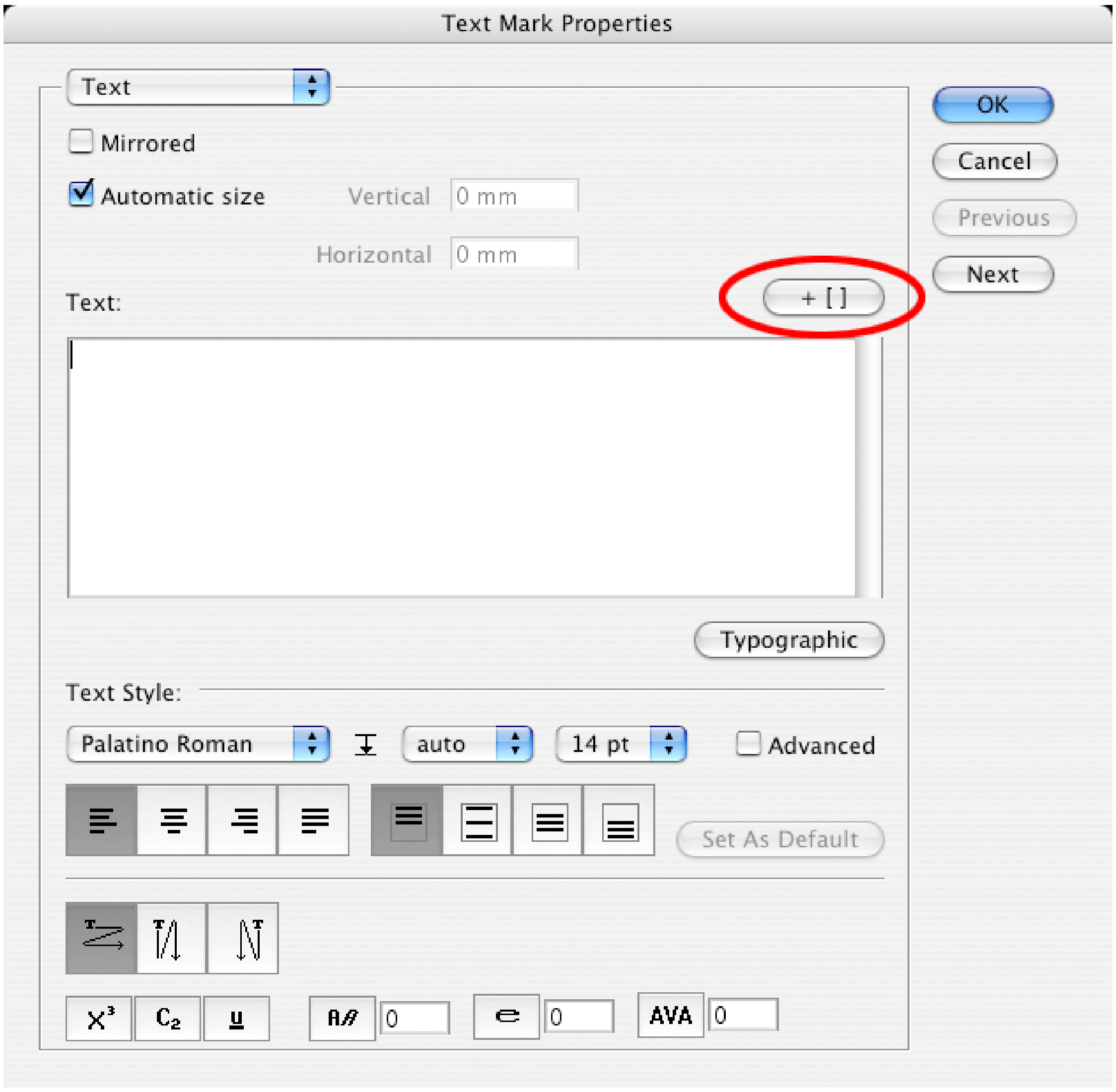
SmartMarks are divided into three types: Text Mark, Image Mark and Barcode Mark.





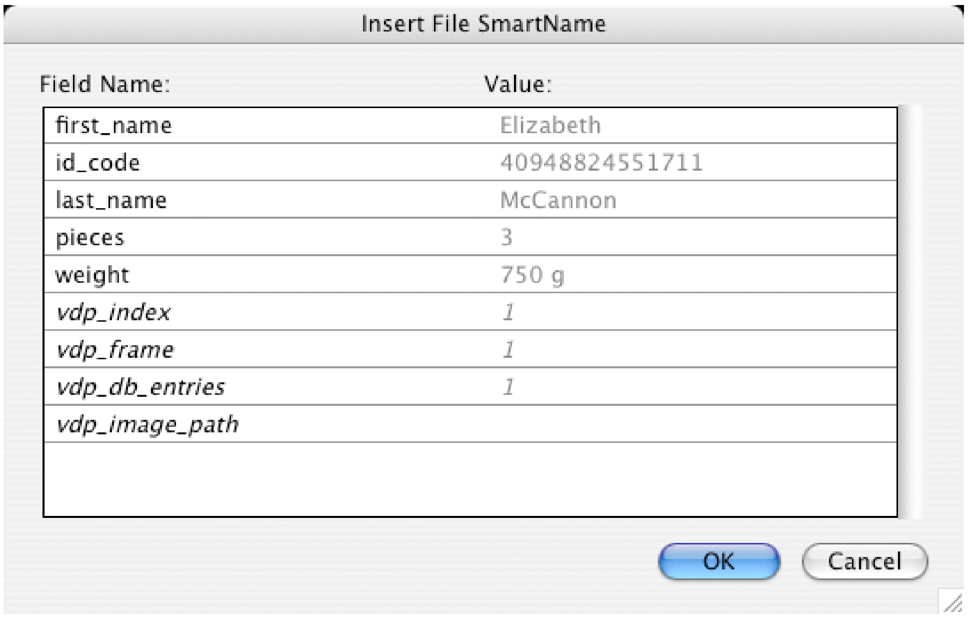
Inserting Text Marks

A Text Mark is used to generate variable text elements. The first page of the Text Mark Properties dialog allows you to define a text value for a variable text element and to choose formatting and aligning of the text within the specified text box.



You can:

- Use the button '+ [ ]' to **insert a File SmartName into the text value**. Once a File SmartName is used in the Text Mark, the Text Mark carries variable information – for each record of the database the Text Mark will be expanded into a different value. For design purposes, the value specified in the File SmartNames dialog will be used. The inserted File SmartName is displayed in the input field as [f.fieldname].



Field Name:	Value:
first_name	Elizabeth
id_code	40948824551711
last_name	McCannon
pieces	3
weight	750 g
vdp_index	*1*
vdp_frame	*1*
vdp_db_entries	*1*
vdp_image_path	

- **Format the text** in the 'Text' input field by selecting the appropriate options in the **Text Style area**:
  - Every change is applied to the whole text, unless the option **Advanced** is enabled. In this case formatting strings can be pasted into the text to apply the changes only on segments of the Text Mark's value.
  - Special characters can also be inserted using the button **Typographic**.
- **Mirror** the contents of the SmartMark by enabling the Mirrored option.
- Specify custom text box dimensions or enable automatic sizing of the text box by enabling or disabling the **Automatic Size** option.

Note:

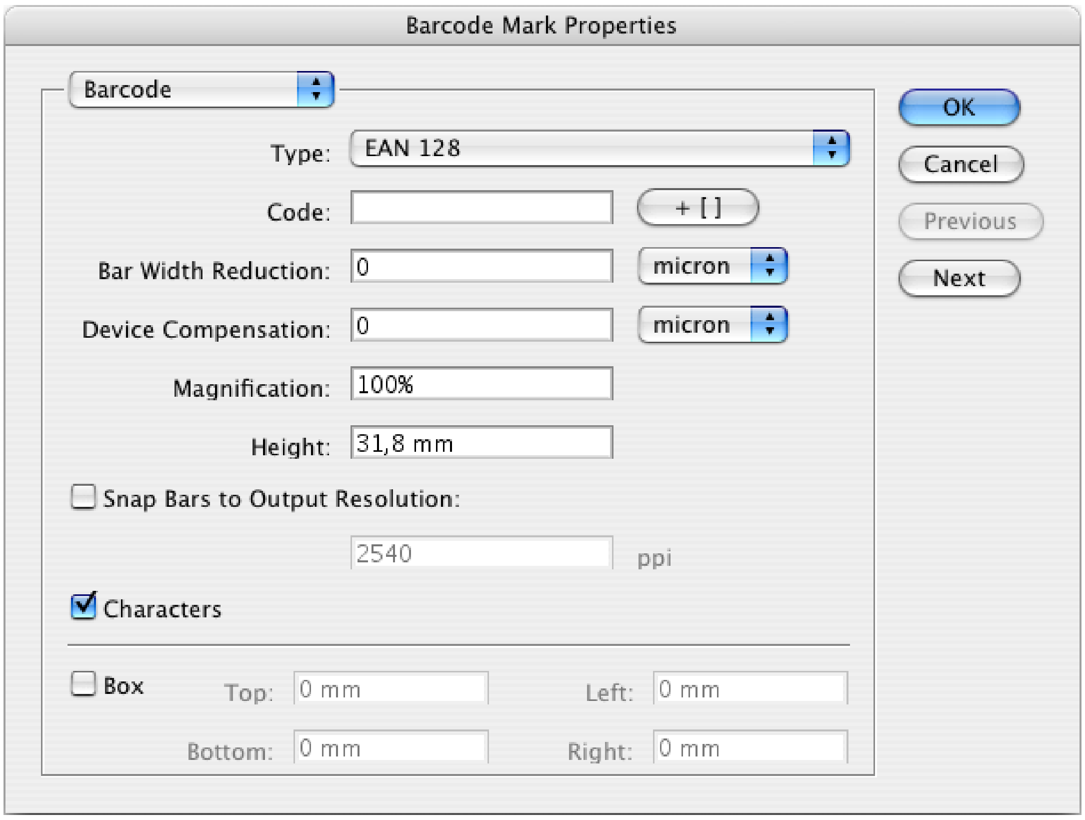
Only fonts installed on the BackStage server can be used with Text Mark. Font Manager is used for font importing.

Note:

The Inks and Position pages of the dialog are explained in separate chapters, as they are almost identical for all three SmartMark types.

Inserting Bar Code Marks

A Bar Code Mark is used to generate variable bar code elements. The first page of the Bar Code Mark Properties dialog allows the user to choose a type, define a value and some additional options for a variable bar code.



Barcode

Type: EAN 128

Code: + [ ]

Bar Width Reduction: 0 micron

Device Compensation: 0 micron

Magnification: 100%

Height: 31,8 mm

☐ Snap Bars to Output Resolution: 2540 ppi

☑ Characters

☐ Box Top: 0 mm Left: 0 mm

Bottom: 0 mm Right: 0 mm

The following types of bar codes are supported:

- 2 of 5
- CLF-8
- CODABAR
- CODE 128
- CODE 128 (long)
- CODE 39
- DATAMATRIX
- EAN 128
- EAN 13
- EAN 8
- HIBC 128
- HIBC 39
- ITF-14
- ITF-16
- Int. 2 of 5
- Laetus Pharma Code
- M+S 7
- MSI
- Marks & Spencer
- PZN
- Paraf Italy

- UPC-A
- UPC-E

Every code has its own set of attributes. You can:

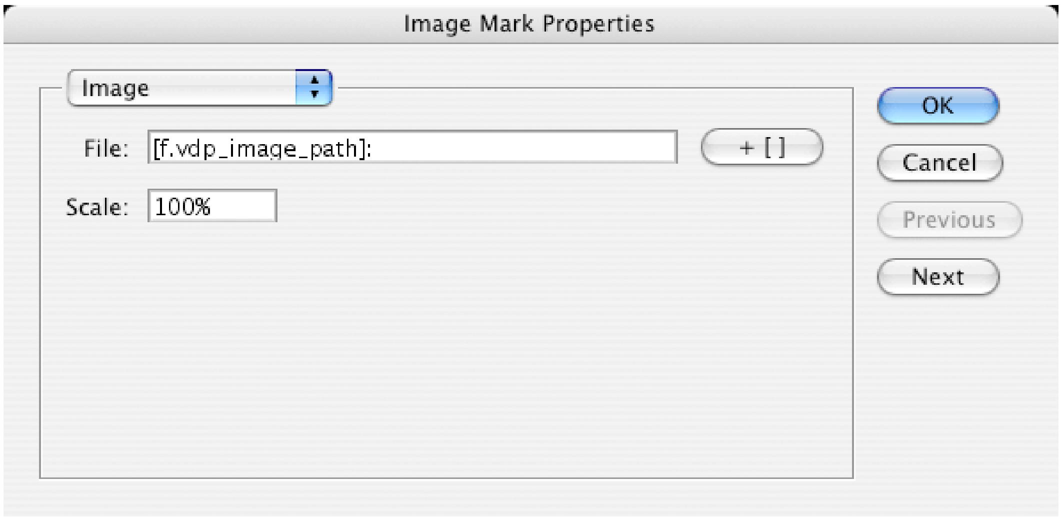
- Enter the bar code value in the **Code** field. Use the '+ [ ]' button to insert a File SmartName into the barcode value.
- Enable the generation of readable characters under the bar code element by enabling or disabling the **Characters** option.
- Create a masking white rectangle around the barcode to ensure legibility of the barcode by enabling the **Box** option.

Note:

For more information on all the specific barcode options, please consult the barX documentation.

Inserting an Image Mark

An Image Mark is used to insert variable bitmap or vector-based images into the design based on their file locations.



File: [f.vdp_image_path]:

Scale: 100%

The first page of the Image Mark Properties dialog contains the File field, which specifies the location of the image.

Use the '+ [ ]' button to insert a File SmartName into the File field’s value. The vdp_image_path File SmartName is already prefilled for you. This SmartName can be configured in the File SmartNames dialog and allows the user to specify a folder path, where all the variable images are located.

Example The full path to an image looks like this:

```
\\server\share\customer\job\images\image1.ct
```

Use `vdp_image_path` to specify the fixed part of the path:

```
\\server\share\customer\job\images\
```

(Local addresses are translated to network addresses automatically.)

Then, in the ImageMark settings, you can specify the full path as follows:

```
[f.vdp_image_path]\image[vdp_index].ct
```

Using `vdp_image_path` is recommended since it makes your job **more portable**. If the variable job is moved to another Jobfolder, it is sufficient to change the document's image path SmartName and the correct value for all the Image Marks is reestablished.

The **Scale** option provides the possibility of changing the size of the Image Mark, relatively to its original print size.

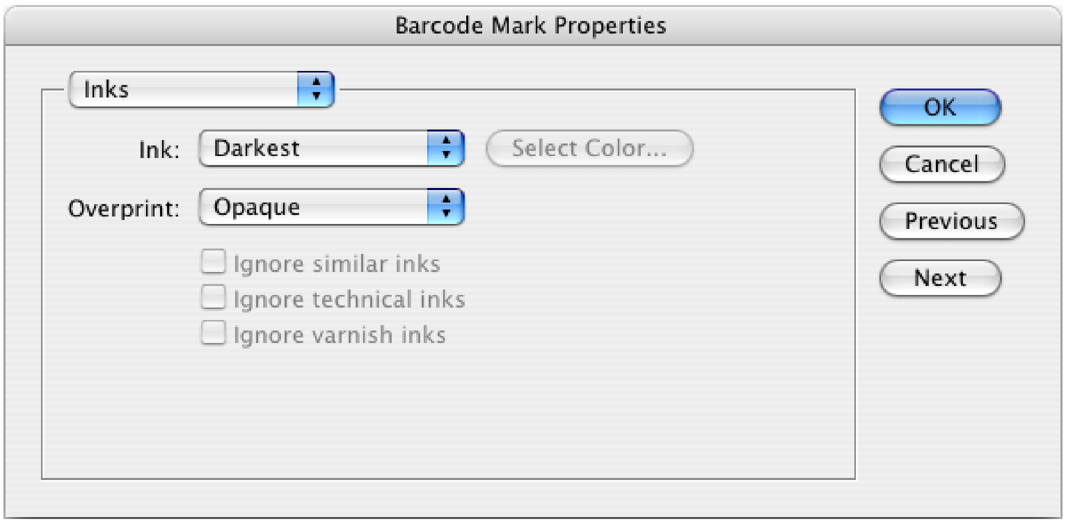
When variable images are used in the design, the files must be stored in a container so that the BackStage server can reach them. It is recommended to store all images in a **Jobfolder**. If the same set of images is used in multiple jobs it is possible to store them only once and reuse them from all jobs.

Image requirements:

- All images must be in **CT** or **Normalized PDF** format, and they must not use RGB or LAB color space. Other file types can be converted into the CT format using the **Import Image** task on the BackStage server.
- Clipping paths and alpha channels are not supported for bitmap file formats.
- Also, based on the option defined in the Inks page, all the images must be either multi-channel or single-channel, as mixing of different color-spaced images is not allowed.

Setting Inks options

The **Inks** page of the **SmartMark Properties** dialog allows you to define an ink or a color (combination of inks) which will be used for the specific SmartMark object. The options for Text Marks and Bar code Marks are identical; ink options for Image Marks are a little different.


Barcode Mark Properties

Inks

Ink: Darkest

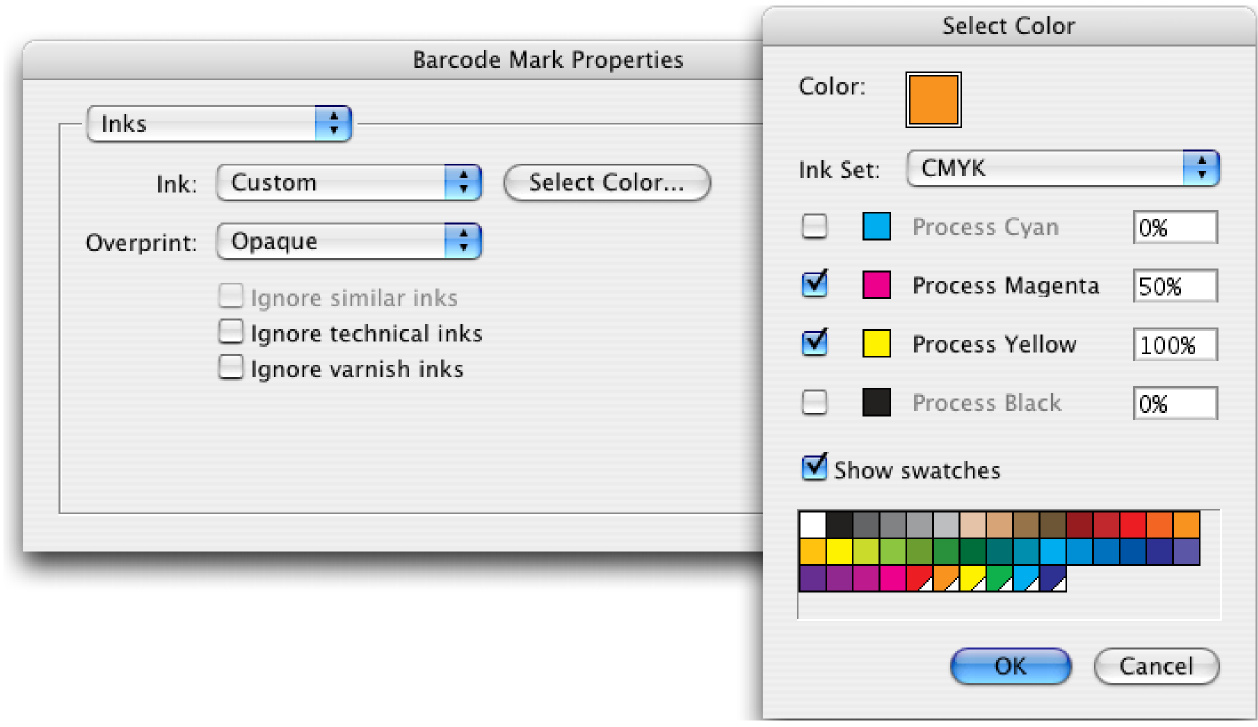
Overprint: Opaque

☐ Ignore similar inks
☐ Ignore technical inks
☐ Ignore varnish inks

The Inks page for Text Marks and Barcode Marks provides you the following options:

- **Ink** indicates the inks that are used for the current SmartMark.

- *Custom*: You can define a custom color. It is possible to choose either from all the inks used in the document (including spot colors), or from predefined ink sets such as CMYK, HP IndiChrome or HP IndiChrome Plus. A percentage of each ink can be chosen, and a preview of the resulting color is generated.



Inks
Ink: Custom
Overprint: Opaque
☐ Ignore similar inks
☐ Ignore technical inks
☐ Ignore varnish inks

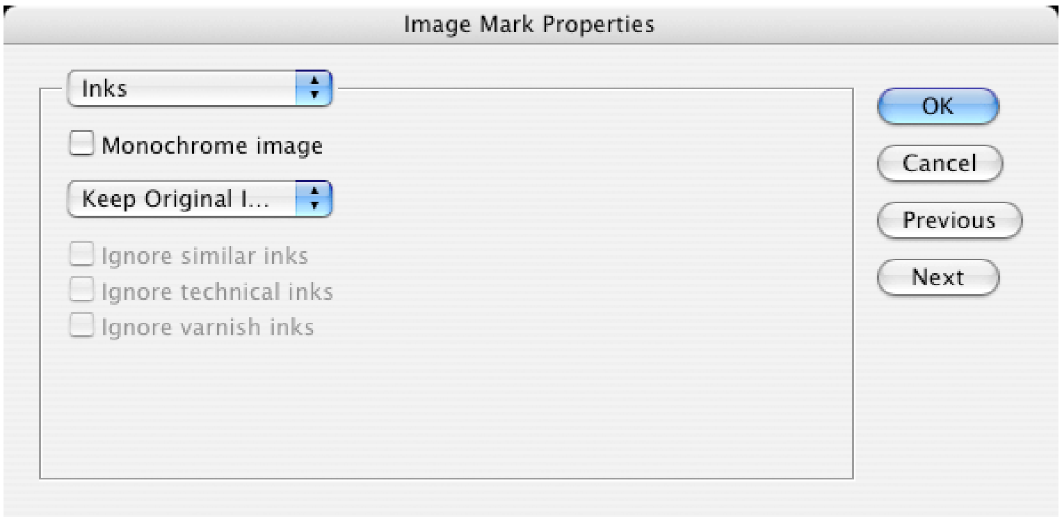
Color:
Ink Set: CMYK
☐ Process Cyan 0%
☑ Process Magenta 50%
☑ Process Yellow 100%
☐ Process Black 0%
☑ Show swatches

For CMYK colors, it is also possible to use swatches as a reference – an easy way to guarantee the required color. Moreover, if the swatch is made global, a link between the SmartMark and a swatch name can be established. In this case, if the swatch color is changed, the color of the SmartMark will be updated as well. The link can be destroyed by clicking on the broken chain icon once it has been established.

- *Darkest* sets the fill of the object to 100% of the darkest ink used in the document. Please note that the Black is not always the darkest ink. Certain Pantones might be darker.
- *Knockout*: the object fill is set to white so the background will be masked out.

- **Overprint** indicates the overprint mode that is being used for the current SmartMark.
  - *Opaque* will have the SmartMark object to knock out any object located underneath it.
  - *PostScript* will make the object behave according to the PostScript overprint definition.
- **Ignore Similar Inks**: an ink with different variants of screening or lineature is used only once. This feature is not important for digital printing.
- **Ignore Technical Inks**: inks of which the group type is set to 'Technical' are ignored for this SmartMark.
- **Ignore Varnish Inks**: inks of which the group type is set to 'Varnish' are ignored for this SmartMark.

The ink options for Image Marks are identical to those of Text Mark and Barcode Mark only when the Monochrome Image option is turned on. In this case, the ink defined in the image file is ignored and the ink specified in the SmartMark Properties dialog is used. However, multichannel images are also supported. The Monochrome Image option must be turned off and the user is presented with the following options:



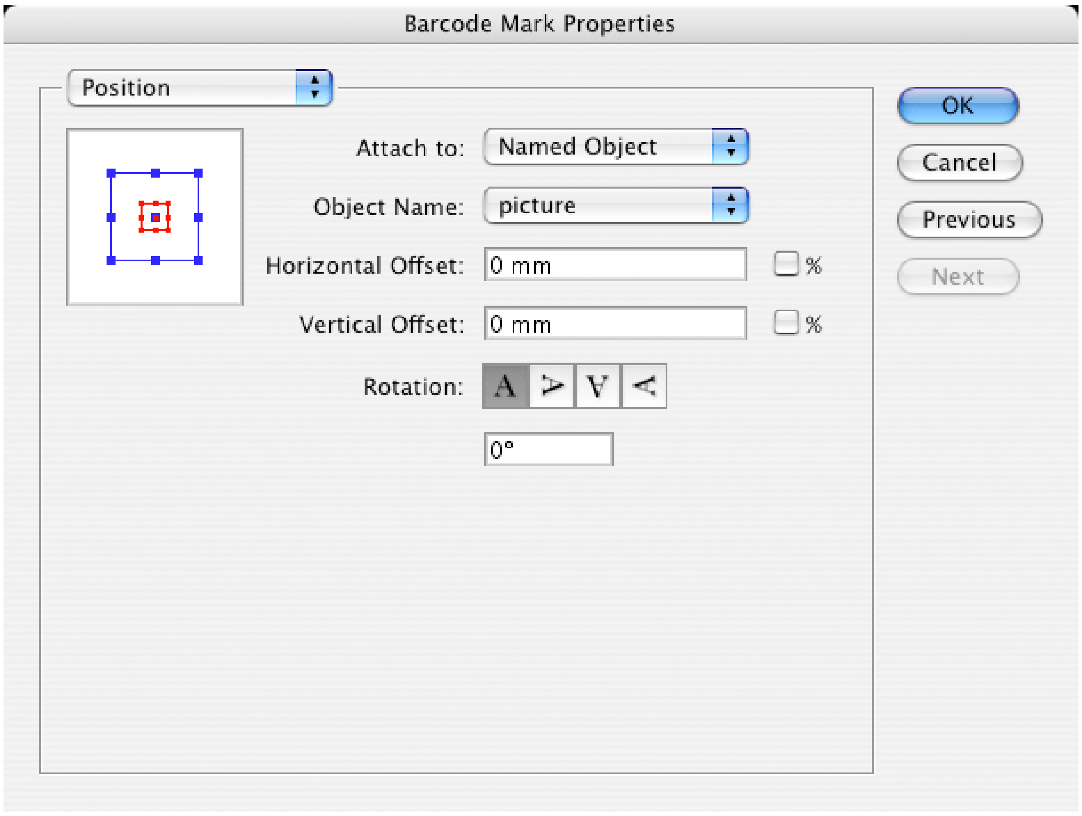
- **Keep Original Inks**: the Image Mark will use inks defined in its document, so the appearance will not be changed.
- **Map Inks**: ink no. 1 from the one-up will be mapped onto ink no. 1 from the image file, etc. Note that the appearance of the Image Mark may change dramatically. 'Ignore Ink' conditions can be set for this option being enabled.
- **Keep Ink Names**: inks will appear only if they are present in the one-up document.

As mentioned before, when you are not using predefined ink sets such as CMYK or HP IndiChrome, only the inks which are part of the job can be assigned to a SmartMark. To work around this limitation you can define an object with the requested ink outside of the Media Box of the document. The ink will then become part of the job and can be assigned to any SmartMark object.

If a SmartMark contains a spot ink that will later be converted into a color space of the Indigo press it is not advised to use PostScript overprint. Opaque should be used instead. Separation into inks of the press combined with PostScript overprint might cause the SmartMark to disappear. This rule also applies in case CMS is involved.

Setting Position options

Position is always specified relative to an object. When the size or position of the referenced object is changed, the position of the attached SmartMark will be updated accordingly.



Attach to: Named Object
Object Name: picture
Horizontal Offset: 0 mm
Vertical Offset: 0 mm
Rotation: 0°

The page contains the following options:

- **Position diagram**: here the relative position of the SmartMark to the reference object can be set. Click the diagram to change the position.
- **Attach to**: defines the reference object for determining the SmartMark's position.
    - *Trim Box*: the SmartMark is referenced to the Trim Box of the document.
    - *Media Box*: the SmartMark is referenced to the Media Box of the document.
    - *Named Object*: the SmartMark is positioned relatively to a Named Object. A Named Object can be another SmartMark with a custom name, or a graphic element or a group of graphic elements contained in the design with an assigned name. The name can be assigned to an object using the SmartMarks palette options. It should not be longer than 100 characters.
- **Object name**: here a specific Named Object for Attach to / Named Object option is chosen.
- **Horizontal and Vertical Offset**: the offset of the SmartMark from the referenced object.
- **Rotation** indicates the rotation of the SmartMark object.

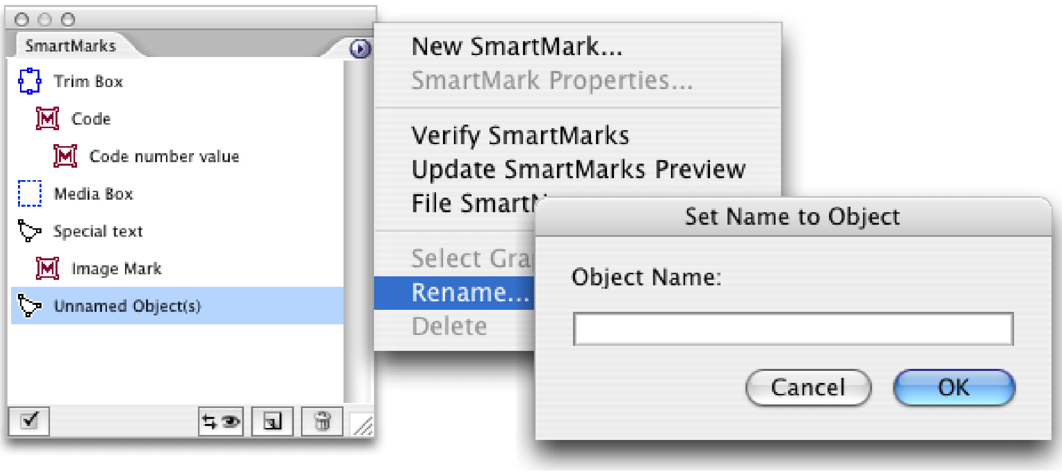
Working with Named Objects

When determining the position of SmartMarks, you must attach the marks either to the Trim Box, the Media Box, or to a Named Object.

To create a Named Object:

1. Select the desired graphic object using the Illustrator's selection tool. An Unnamed Objects(s) item appears in the SmartMarks palette.

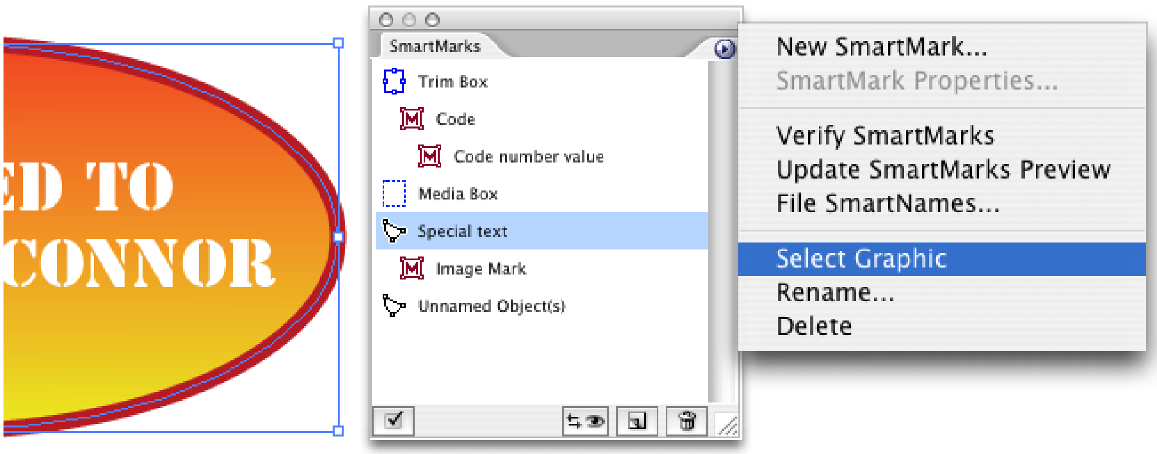
2. Rename the unnamed object by clicking the Rename... command in the palette's fly-out menu. This will assign a new name to the object. From this moment, SmartMarks can be attached to the newly created Named Object.



Renaming an existing object is done in exactly the same way. First, select the desired object in the SmartMarks palette, and then rename it. The use of in-place editing of the object names in the SmartMarks palette is also possible.

Named objects can be either static objects contained in a design, or SmartMarks.

For SmartMarks, you can create even longer dependency chains. This is particularly useful in case the position of a SmartMark is dependent on a position of another SmartMark, but since SmartMarks are variable, the exact position cannot be resolved at a design time. If you want to **move the whole tree of such dependencies**, or you are just dealing with a too complex document, you can take advantage of the feature **Select Graphic**, which selects the graphic objects in the Illustrator document that corresponds to the selected SmartMarks palette item.



To **delete** a Named Object, use the Delete command on the fly-out menu. However, this functionality behaves differently for static Named Objects and for SmartMarks:

- Choosing Delete for **static objects** deletes just the **name of the object**; the graphics data of the object are preserved.
- Choosing Delete for **SmartMarks** deletes the **complete SmartMark**, along with its graphic data. Any SmartMarks attached to the object being deleted are re-attached to the Trim Box by default.

3.3.5 Viewing your SmartMarks with the Preview task

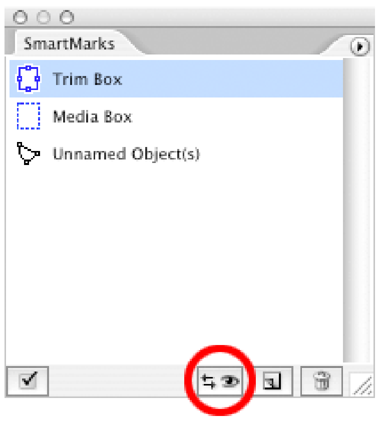
A SmartMark object can be moved with Illustrator tools. Each position change will automatically update the position offsets in the SmartMark Properties dialog box. However, for any other change to SmartMarks (such as change to text formatting, new ink assignment or a new rotation angle) the preview needs to be regenerated.

Also, for the initial creation of SmartMark objects the intervention of the BackStage server is required. The Update SmartMarks Preview task is used for this purpose.

Please note that even though the Variable Data plug-in is able to perform position offsets updating without the need to run the preview task, for the expansion the positions generated by the preview task are always taken into account. This is why it is recommended to **always run the preview task at the end of the design process**, to make sure that the final output will be in sync with the result displayed in Adobe Illustrator.

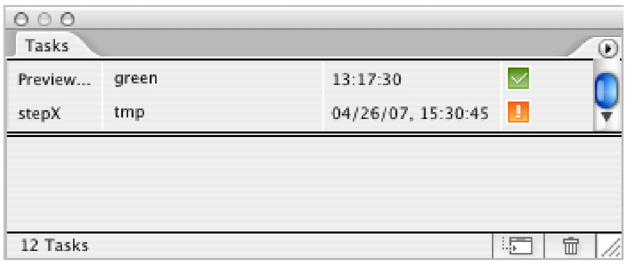
Running the Update SmartMarks Preview task

As already mentioned, SmartMarks are not generated by the Adobe Illustrator itself, but by the BackStage server. To regenerate the SmartMarks, you have to run the Update SmartMarks Preview task.



Running the task will:

- Send the one-up to the BackStage server.
- Save and close the document to prevent the user from modifying the design while the preview is being generated.
- Display the Tasks palette, showing the progress of the task. Once the task has finished, the document with regenerated SmartMarks can be re-opened by double-clicking the task‘s row in the Task palette.

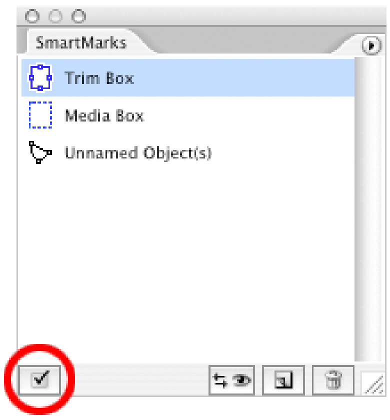


Preview...	green	13:17:30	✓
stepX	tmp	04/26/07, 15:30:45	!

SmartMarks consistency

You should not modify the contents of the Variable_Data layer directly, because by doing so the consistency of the SmartMark objects may be impaired. This is also why the **Verify SmartMarks** feature of the Variable Data plug-in is run just before launching the actual preview task. This feature checks whether the state of SmartMarks is consistent and shows a warning dialog if any inconsistencies are detected.

You can also check the consistency of SmartMarks manually without the need to run the preview task, by clicking the Verify SmartMarks menu item. It can be used to synchronize the contents of the SmartMarks palette with the actual state of the document.



3.3.6 Reviewing our work so far

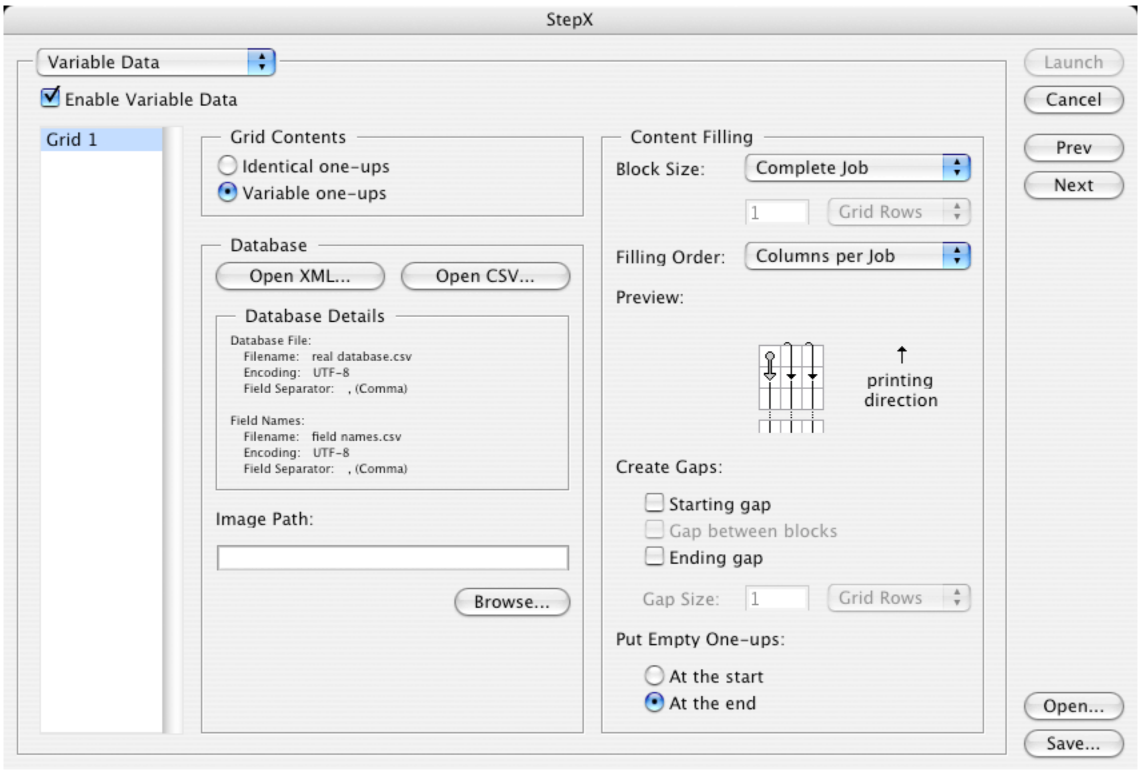
Up to this point in the workflow, we have:

- prepared a database;
- used the database to define File SmartNames in the Variable Data plug-in;
- created a SmartMark of the preferred type and desired attributes; and
- visualized the SmartMark by running the 'Update SmartMarks Preview' task.

To be able to define the positions of SmartMarks more in accordance with their mutual relationship, the concept of creating Named Objects is introduced.

3.4 Step and Repeat and launch Server tasks

Once the variable design is created using the Variable Data plug-in, a repetition must be made. A repetition (Step and Repeat) is performed by the stepX plug-in.


StepX

Variable Data

[✓] Enable Variable Data

Grid 1

Grid Contents
( ) Identical one-ups
(●) Variable one-ups

Database

Database Details
Database File:
Filename: real database.csv
Encoding: UTF-8
Field Separator: , (Comma)

Field Names:
Filename: field names.csv
Encoding: UTF-8
Field Separator: , (Comma)

Image Path:

Content Filling
Block Size: Complete Job
Filling Order: Columns per Job

Create Gaps:
[ ] Starting gap
[ ] Gap between blocks
[ ] Ending gap
Gap Size: 1 Grid Rows

Put Empty One-ups:
( ) At the start
(●) At the end

The variable data-specific options are added to the stepX user interface as a new dialog page and offer the possibility to specify the expansion parameters. If the Enable Variable Data option is turned off, stepX will work exactly the same way as it would without the Variable Data Printing license. However please note, that the Alternate option is not supported if the 'Enable Variable Data' option is turned on.

Note:

For general information on the stepX functionality and all of its parameters, please consult the stepX documentation. A detailed explanation of all the variable data-specific options follows.

3.4.1 Grids

For variable data, multiple grids are supported.

The VDP parameters must be specified for each grid individually.To specify the parameters for a grid, select the grid from the Grids list.

The options for each grid are stored individually, so a change of parameters for one grid will never influence the output of another grid on a frame.

3.4.2 Grid Contents

The expansion task can handle two types of grid contents.

- '**Identical One-ups**' does not include any variable elements. It will fill the empty space on a substrate that is unused by variable data.
- '**Variable One-ups**' carries the design with variable data elements. The main difference for these two types of contents is the absence of a database specification and ordering pattern possibilities for the 'Identical One-ups' option.

Following enumeration gives you an overview of the differences:

Identical one-ups:

- Total number of one-ups for repetition
- Block size (optional)
- Gap size
- Gap at the beginning of the job (optional)
- Gap at the end of the job (optional)

Variable one-ups

- XML database – file URL
- CSV database – one or two file URLs + other options (delimiter character, …)
- Image Path URL – when Image Marks are used (optional)
- Block size: job, sheet or custom
- Gap at the beginning of the job (optional)
- Gap between blocks (optional)
- Gap at the end of the job (optional)
- Gap size
- Ordering
- Empty one-ups

3.4.3 Database

This parameter defines the database which will be used for expansion. The database file can be different from the one used for loading File SmartNames.The only requirement is that the identical column names are used. The database can be either in XML or a CSV file format and must use the UTF-8 encoding.

XML Database

This type of database is a file in the eXtensible Markup Language format. Clicking 'Open from XML...' opens a file browser, which allows you to select the XML file. Please note that the XML file must be stored within a container, in order to be visible to the BackStage server which performs the expansion task.

CSV Database

A CSV is a Comma Separated Value file. The 'Open from CSV...' button opens a dialog box, where the operator has to specify a URL to the database file, its Encoding and Field Separator options.If field names are not stored in the first line of the database file, you can specify a separate file with the required information by using the Field Names part of the dialog box.

CSV file format is the preferred format, since the expansion performance for larger databases is higher. An overview of all the settings will be displayed in the Variable Data page of the stepX dialog.

Image Path

The location specified here is used to fill in the `vdp_image_path`' SmartName, which might be necessary if the job uses Image Marks. It must contain a valid path to the images folder.

Please note, that the value specified in the File SmartNames dialog is used for designing / preview purposes only and therefore is not used by the expansion task. Therefore it needs to be specified once again here as an expansion task parameter.

3.4.4 Content Filling

This part of the dialog box is used to specify how the database values will flow through the variable job. Please note, that according to Variable Data Printing standards, all the jobs are **assumed to be portrait-oriented**.

Block Size

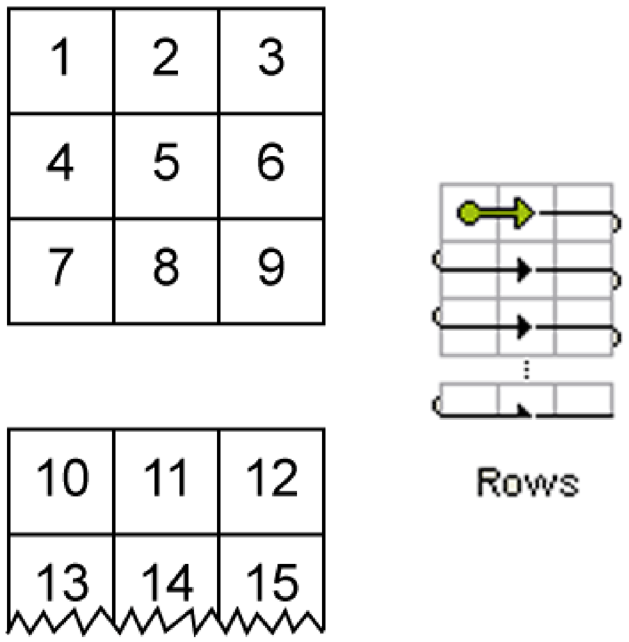
Following options are available:

- Complete Job: This option defines that database will flow through the job without any interruptions.
- One Frame: The size of a frame is equal to size of the document defined in stepX.
- Custom: The size of a block can be user-defined. Units that can be used are grid rows, frames, meters or feet. For units such as meters or feet, rounding to grid rows is applied.

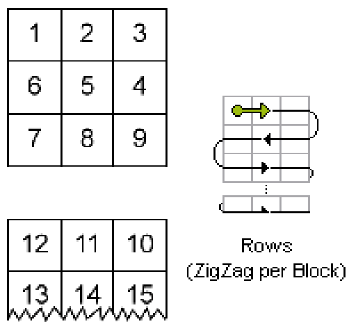
Filling Order

Following options are available:

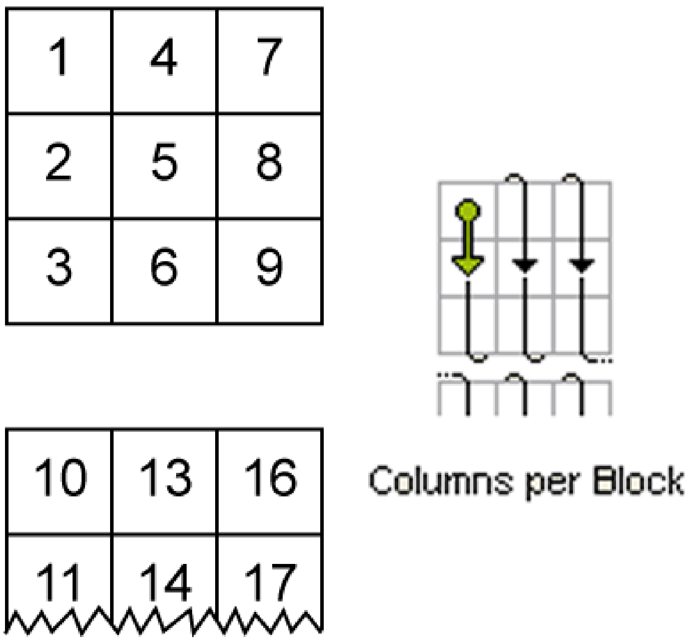
- **Rows:** The numbering is restarted when the end of a row is reached and continues from the left cell of the next row. The starting position is the top left cell of each block.



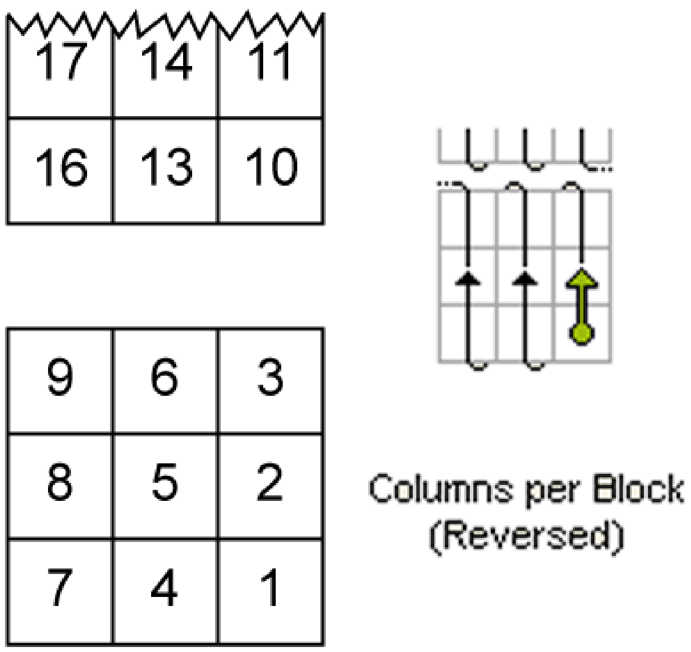
- **Rows (ZigZag):** The numbering pattern alternates direction for each row and is restarted once the beginning of a row is encountered. The starting position is the top left cell of each block.



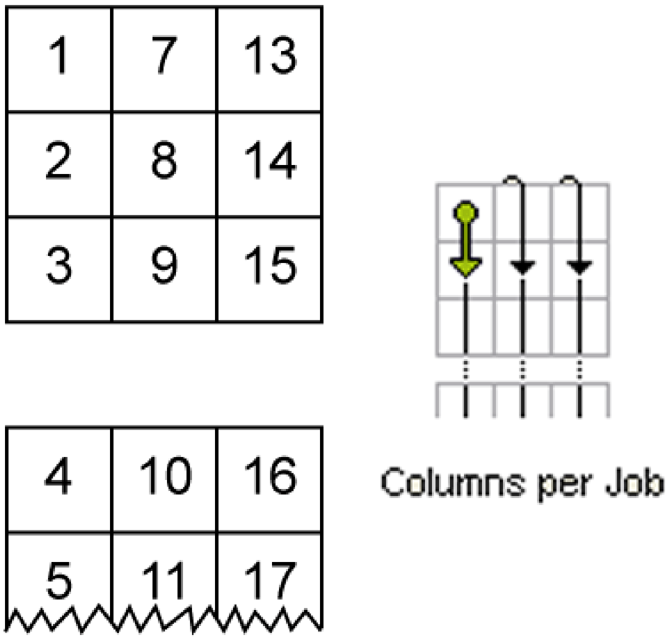
- **Columns per Block:** The numbering pattern is restarted when the end of a column in a block is reached and continues from the top cell of the next column. The starting position is the top left cell of each block.Note: If the Block Size is set to ‘Complete Job’, then this filling order is equal to 'Columns per Job'.



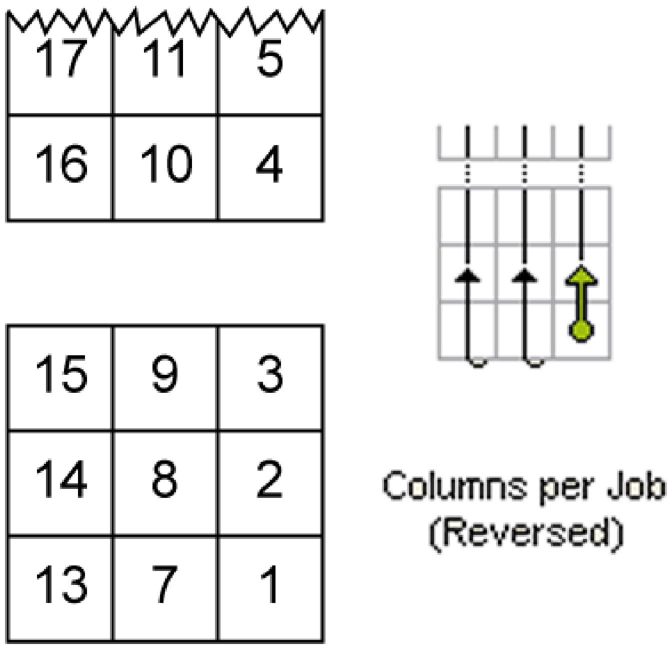
- **Columns per Block (Reversed):** The numbering is restarted when the end of a column in a block is reached and continues from the bottom cell of the next column. The starting position is the bottom right cell of each block. Note: If the Block Size is set to ‘Complete Job’, then this filling order is equal to 'Columns per Job (Reversed)'.



- **Columns per Job:** The numbering is restarted when the last cell of a column in the job is reached and continues from the top cell of the next column. The starting position is the top left cell of the job.



- **Columns per Job (Reversed):** The numbering is restarted when the last cell of a column in the job is reached and continues from the bottom cell of the next column. The starting position is the bottom right cell of the job.



Create Gaps

Some jobs need to be interrupted after a certain amount of labels. This amount can be defined using the Block Size parameter. Gaps can be inserted between these blocks as well as to the beginning and to the end of a job.

No one-ups are printed in the gap area, so the press operator can easily identify the gap as a white space. However, all the elements outside the one-up area, such as die cut marks, are printed.

Here you can specify where you want to the gaps to be generated. The following options are available:

- Starting gap This gap will be printed first (with respect to the printing direction), before any variable data are printed.
- Gap between blocksThese gaps will be generated in between Blocks.Note: When Block Size is set to 'Complete job', this toggle is useless.
- Ending gapThis gap will be printed last (with respect to the printing direction), after all the variable data have been printed.

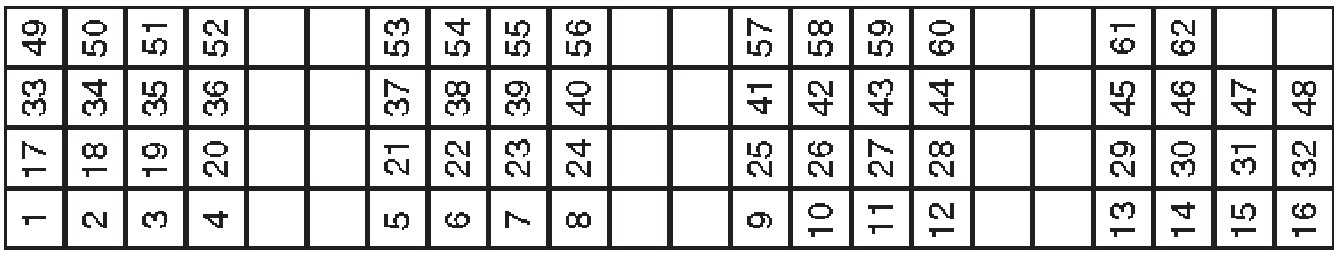
The 'Gap Size' option specifies the size of the gaps in grid rows, frames, meters or feet.

Put Empty one-ups

In general, the database size is not an exact multiple of the number of cells in a block. most cases, one of the blocks cannot be filled completely with database entries. It will be padded with empty one-ups. It will be padded with empty one ups.The toggles **At the start** or **At the end** specify where to put the empty one-ups from Filling Order point of view.

Examples

Printing direction



This job was created using the following settings: Block size: 4 grid rows; Filling order: Columns per Job; Gap between blocks: 2 grid rows; Empty One-ups: At the end

17	18	19	20			37	38	39	40			57	58	59	60				
13	14	15	16			33	34	35	36			53	54	55	56				
9	10	11	12			29	30	31	32			49	50	51	52				
5	6	7	8			25	26	27	28			45	46	47	48				
1	2	3	4			21	22	23	24			41	42	43	44			61	62

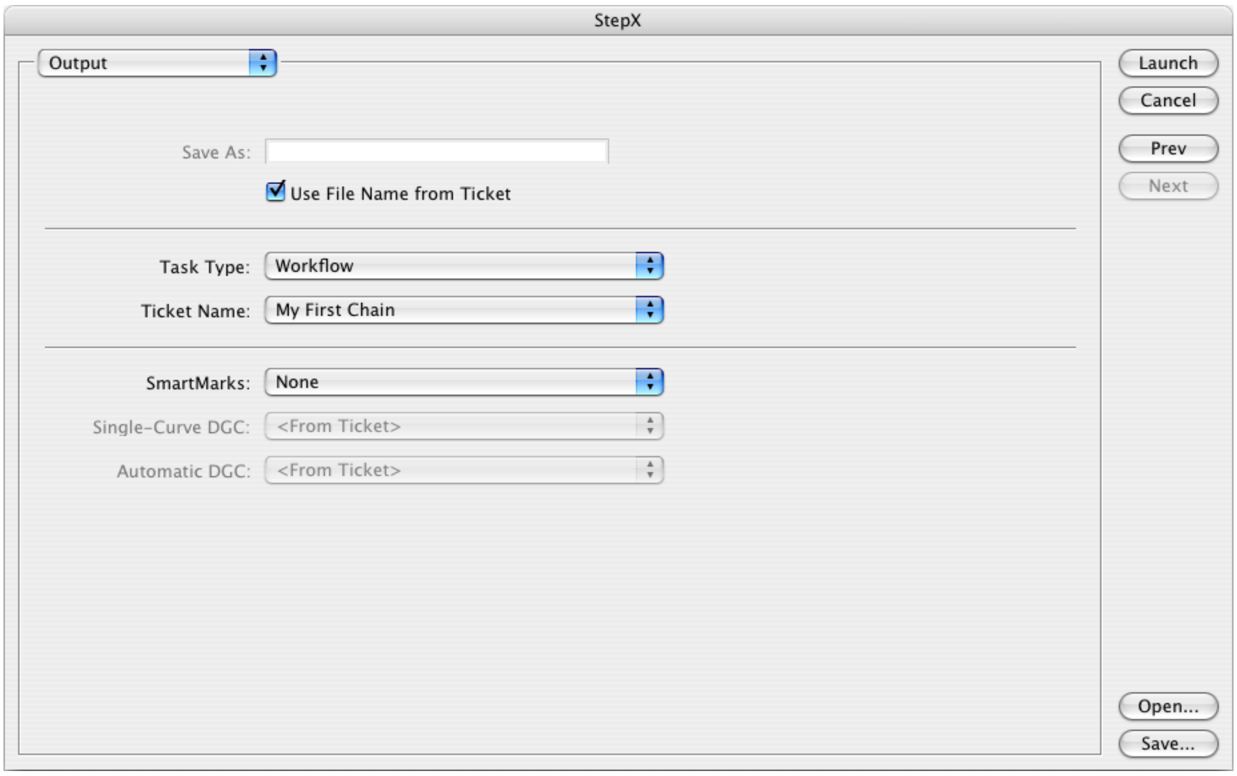
This job was created using the following settings: Block size: 4 grid rows; Filling order: Columns per Block; Gap between blocks: 2 grid rows; Empty One-ups: At the end

62	61			44	43	42	41			24	23	22	21			4	3	2	1
				48	47	46	45			28	27	26	25			8	7	6	5
				52	51	50	49			32	31	30	29			12	11	10	9
				56	55	54	53			36	35	34	33			16	15	14	13
				60	59	58	57			40	39	38	37			20	19	18	17

This job was created using the following settings: Block size: 4 grid rows; Filling order: Columns per Block (Reversed); Gap between blocks: 2 grid rows; Empty One-ups: At the beginning

3.4.5 Launching Server tasks

The last page of the stepX plug-in is used to launch server tasks, once all the repetitions and variable data parameters are specified correctly. The layout of the last page depends on whether the variable data are or are not used.



Save As:
☑ Use File Name from Ticket

Task Type: Workflow
Ticket Name: My First Chain

SmartMarks: None
Single-Curve DGC: <From Ticket>
Automatic DGC: <From Ticket>

For Variable Data Printing, the output options have been slightly modified:

Task Type has only two options: '*Expand Variable Data to PPML File (DeskPack only)*' and '*Workflow*'.The list of tasks that can follow will contain only the global tasks and workflows that start with the ticket 'Expand Variable Data to PPML File (DeskPack only)'. The tickets which were created according to the chapter “Prepare data” can be selected and run from here. The 'Expand Variable Data to PPML File (DeskPack only)' task is used only if you care about the expansion output but do not want to run the RIP task afterwards. In most cases, it is recommended to use a chained ticket with a 'RIP to WS4000 Series' task.

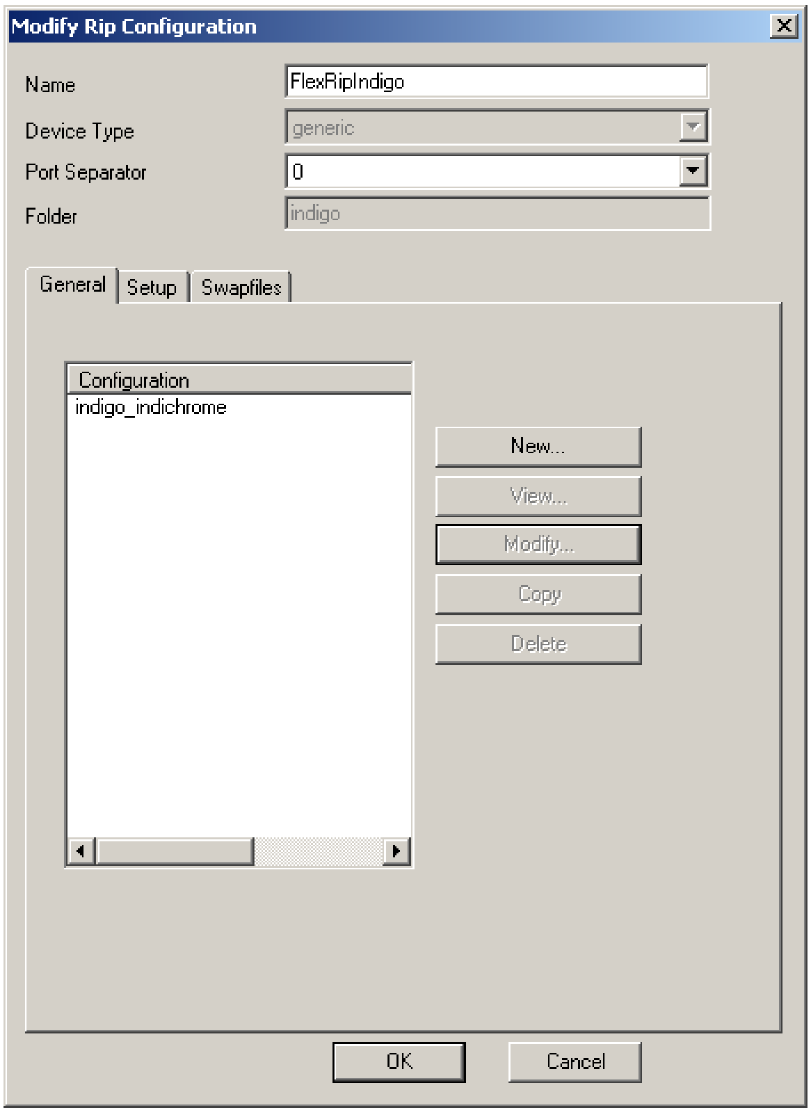
The **SmartMarks** option allows the user to specify a Grid Mark set (created using PackEdge or Plato), which results in generating appropriate die-cut marks in the design. You can contact us and request a specific SmartMark set, based on your needs.

Once the configuration and all the parameters contained in stepX are specified correctly, launching the appropriate workflow will result in expansion and ripping of the variable data, and placing the output JLT files into the hotfolder on the press.

4. Configuration

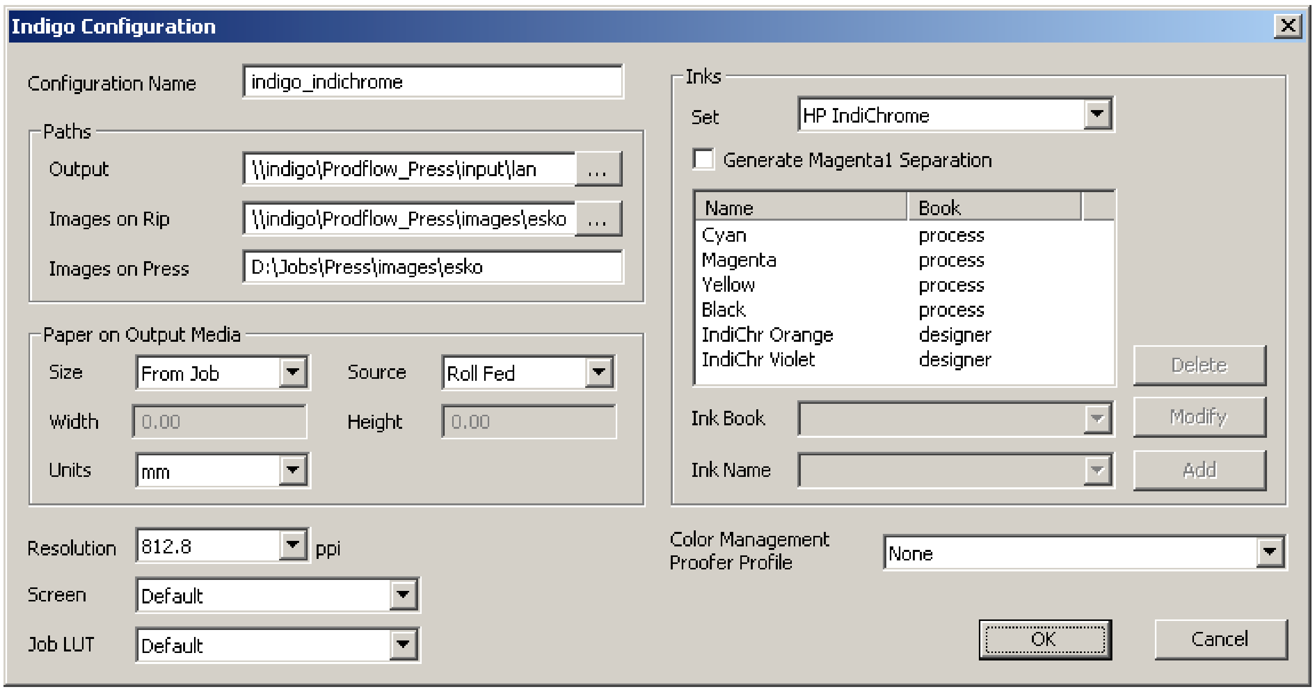
4.1 FlexRip/Indigo Configuration

To configure the RIP, a FlexRip/Indigo Dispatcher must exist. Usually a default one is created during the installation process. To modify its settings, use the FlexRip Configurator application.


Modify Rip Configuration

Name	FlexRipIndigo
Device Type	generic
Port Separator	0
Folder	indigo

4.2 Dispatcher Configuration


Indigo Configuration

Configuration Name	indigo_indichrome

Paths

Output	\\indigo\Prodflow_Press\input\lan
Images on Rip	\\indigo\Prodflow_Press\images\esko
Images on Press	D:\Jobs\Press\images\esko

Paper on Output Media

Size	From Job	Source	Roll Fed
Width	0.00	Height	0.00
Units	mm		

Resolution	812.8 ppi
Screen	Default
Job LUT	Default

Inks

Set: HP IndiChrome

☐ Generate Magenta1 Separation

Name	Book
Cyan	process
Magenta	process
Yellow	process
Black	process
IndiChr Orange	designer
IndiChr Violet	designer

Ink Book:

Ink Name:

Color Management Proofer Profile: None

The **Paths** group contains the location of the Master JLT file and of all the variable JLT files (images). The output folder of the RIP is the hotfolder on the press. Please note, that all of the paths must use the UNC format, with the 'Images on Press' field being the only exception, since this path is used by the press to access the variable JLT files stored at its local drive. If your network drive is mapped to a drive letter, an automatic conversion to UNC format will be offered.

Typically paths look similar to this:

Output:

```
\\indigo\Prodflow_Press\input\lan
```

Images on Rip:

```
\\indigo\Prodflow_Press\images\esko
```

Images on Press:

```
D:\Jobs\Press\images\esko
```

4.3 Access to the Press PC

The BackStage server user BGSYSTEM must have write access to the hotfolder of the press PC. If not, the 'RIP to WS4000 Series' task will cause an error when writing the Master JLT to the hotfolder.

The configuration of access rights depends on the OS where the BackStage server runs:

- **Windows 2000 Server**: make sure that you have access to the Administrator account of the press PC. Create the user BGSYSTEM on the press PC with the same password as the one on the BackStage server system and grant this user access to the required folders.
- **Windows 2003 Server:**
  - Log in as user BGSYSTEM on the BackStage server.
  - Map the hotfolder of the Press:

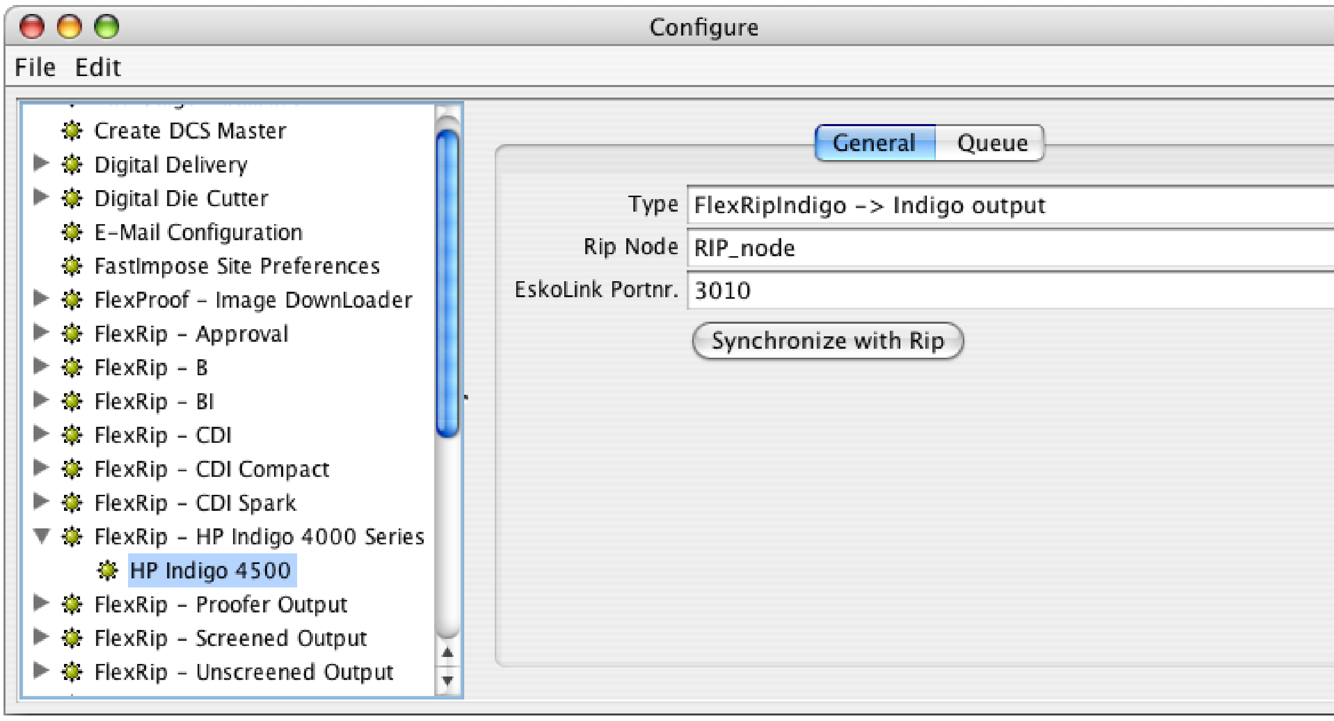
```
net use \\indigo\Prodflow_Press
/SAVECRED
user: . . .
password: . . .
```

`/SAVECRED` stores the credentials permanently. During subsequent logons the share on the press will be automatically accessible.

4.4 Dispatcher connection with the BackStage server

The BackStage server must be configured in order to be able to send data to the HP Indigo Dispatcher. A new proofer device that is linked to the HP Indigo Dispatcher must be created. This can be done using the Configure application (BackStage Pilot > Tools menu > Configure).

In the left pane of the Configure application, select 'FlexRip – HP Indigo 4000 Series' and click File > New to create a new device. In the right pane, enter the port number of the Indigo Dispatcher and click 'Synchronize with Rip'.



Configure

File Edit

- Create DCS Master
- ▶ Digital Delivery
- ▶ Digital Die Cutter
- E-Mail Configuration
- FastImpose Site Preferences
- ▶ FlexProof - Image DownLoader
- ▶ FlexRip - Approval
- ▶ FlexRip - B
- ▶ FlexRip - BI
- ▶ FlexRip - CDI
- ▶ FlexRip - CDI Compact
- ▶ FlexRip - CDI Spark
- ▼ FlexRip - HP Indigo 4000 Series
  - HP Indigo 4500
- ▶ FlexRip - Proofer Output
- ▶ FlexRip - Screened Output
- ▶ FlexRip - Unscreened Output

General | Queue

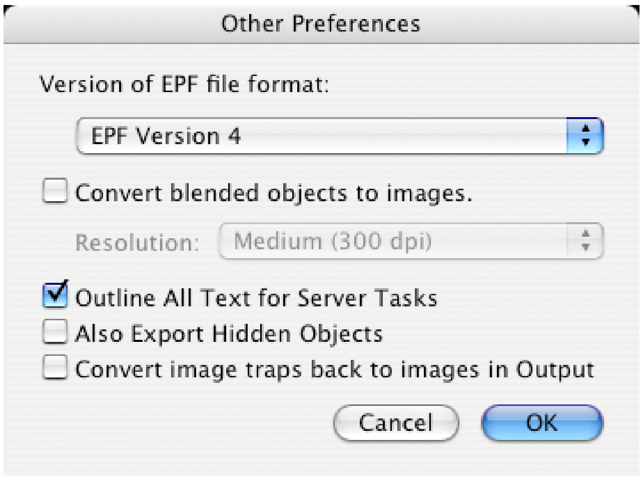
Type: FlexRipIndigo -> Indigo output

Rip Node: RIP_node

EskoLink Portnr.: 3010

4.5 DeskPack Configuration

Make sure that EPF Export in Illustrator is configured to use EPF version 4. This preference can be found in Illustrator CS2.On Windows: Edit > Preferences > Esko > Server > Other Preferences… On Mac: Illustrator > Preferences > Esko > Server > Other Preferences…



Other Preferences

Version of EPF file format: EPF Version 4

☐ Convert blended objects to images.

Resolution: Medium (300 dpi)

☑ Outline All Text for Server Tasks

☐ Also Export Hidden Objects

☐ Convert image traps back to images in Output