Getting started with Place Content
It’s not rocket science...
This is rocket science...
LiveSlides web content

To view

Download the add-in.
liveslides.com/download

Start the presentation.
So how do I do this?

*(place content, not rockets…)*
Topics

• Using GitHub to get the template
• Template structure
• The place content lifecycle
• Example: building a proof page
• Proof page, the process plan
• Proof page, building the HTML template
Using GitHub
Some terminology

• **Git**
  A *free* and *open-source* version control system

• **Version control system**
  Magic system to save all versions of code, show you what the differences are, go back to previous versions, …

• **GitHub**
  The “git” system installed online, ready for use
“Our” GitHub

# Place Content Repository#

The "Place Content" fixup in c aliases pdfToolbox is designed to add additional content to existing PDF documents. It does so through the use of HTML templates, which are converted to PDF and added on top or in the back of the PDF document being processed.

Such templates can be started from scratch (they are just HTML after all), but it is often simpler to either start from an example or at least use a collection of support Javascripts / HTML snippets / CSS files.

This GitHub repository is used to maintain the templates and examples of said HTML templates. You can use it to always get the latest version of the scripts and examples, or you can submit your own changes for others to use. In the unlikely event there are any bugs in what we share here, you can use GitHub to submit your corrections.
Get the template

• Go to [github.com/fourpees/placecontent](https://github.com/fourpees/placecontent)
• Click “Clone or download”
• Select “Download ZIP”
• Unzip
• Open the resulting folder. The folder “_template” is what you want to start from
Template structure
The downloaded template folder
Files

- **Index.html**
  The main HTML file to be converted into PDF

- **README.md**
  A read me file with what I’m talking about here

- **Manifest.xml**
  Configuration file with additional things you want pdfToolbox to make available to your template
Folders

- **callas_tmp**
  A temporary folder, all files will be overwritten when you use this template in place content

- **callas_library**
  The library portion of the template. Stuff you use but don’t change

- **scripts & styles**
  Your Javascript and CSS files
The place content lifecycle
The place content fix up

**Name:**
03 - Fixed page size

**Comment:**

**Fixup category:**
- All
- Color spaces, spot colors, inks
- Pages
- Document info and Metadata
- Interactive elements and properties
- Document
- Page contents
- Layers

**Type of fixup:**
- Find

**Place content on page:**
Placed content based on the chosen folder on the page. The folder has to contain a file in a format, which is supported by callas pdfToolbox. This includes an index.html or index.svg file, that may reference CSS, JavaScript or similar files. Negative offset values may be used to place content left or below the chosen reference point.

- Place page number
- Place text
- Re-encode images using Indexed color spaces
- Remove .notdef glyphs

**Folder:**
03

**Horizontal offset:**
0

**Vertical offset:**
0

**Unit:**
pt

**Relative to:**
Lower left corner

**of:**
TrimBox

**Rotation (degrees):**
0

- **Create on layer**

- **Layername:**
Placed content

- **Place:**
On top

**Apply to:**
All
What does pdfToolbox do?
Seriously dude…
I know there’s more to it…

OK, OK, I’ll spell it out for you
1. Special requests?

• “Manifest.xml” read
  – Communication vehicle between your template and pdfToolbox
  – Ask for:
    • Ink (coverage) information
    • Hit information
2. Here’s your information

- “calsDocInfo.js” written
- A JSON (JavaScript Object Notation) file that contains:
  - Information about the document you’re processing
  - Information about the “Apply to” checks you can specify in the fixup
3. Where are you?

- Main file searched
- pdfToolbox uses either:
  - The file “index.html”
  - If that isn’t there, a “primary” file format file it finds in the template folder
4. Create decorations

• Result PDF created
• Either through:
  – Conversion from image, SVG, PostScript or office file to PDF
  – Conversion from the HTML template to PDF
5. Decorate the tree

• The decoration PDF is added on top of the original PDF
  – Repeated to match number of pages (if necessary)
An example:
a proofing page
How to convert this...
...in this?

<table>
<thead>
<tr>
<th>Job name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job ID</strong></td>
</tr>
<tr>
<td><strong>Format</strong></td>
</tr>
<tr>
<td><strong>Date received</strong></td>
</tr>
<tr>
<td><strong>Delivery date</strong></td>
</tr>
<tr>
<td><strong>Contact</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
</tbody>
</table>

Customer name: 23/10/2017
Proof page:
The process plan
Remember this slide?
Changing page size

• Place content cannot change the size of the document...

• Solutions?
  – Change mediabox to create white space
  – Use the imposition engine to place the incoming PDF on a standard sheet
What do you need?

• A background PDF file
• A short runlist file
• A short sheetconfig file
• The correct measurements for where you want to put the incoming PDF file
This...
A 2-line sheet configuration

NAME  Place on proof background - A4 – Landscape

<table>
<thead>
<tr>
<th>SLOT</th>
<th>preview</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>120mm</td>
<td>20mm</td>
</tr>
<tr>
<td></td>
<td>160mm</td>
<td>160mm</td>
</tr>
<tr>
<td></td>
<td>0mm</td>
<td>0mm</td>
</tr>
<tr>
<td></td>
<td>0mm</td>
<td>0mm</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
A 4-line runlist (*gasp*)

NAME Place on proof background - A4 – Landscape

NewSheet

PositionPage  FirstPage  Slot_1

Increment  FirstPage
Package it up

- Create the imposition files
- Test them in pdfToolbox Desktop
- Create an action using the Switchboard
- Create a new process plan
- Put the action into the process plan

- So far… no rocket science!
Proof page: The HTML template
Build in phases

1. Plain template (from GitHub)
2. Remove excess and set page size
3. Add layout containers
4. Add content
5. Style content
6. Update dynamic content
Thank you!
Questions?