Mako 5.0

Mako 5.0 Release Notes

New feature work

These are the principal new or improved features of Mako 5.0

MAKO-846 Integrate Jaws 4 into Mako to replace the Jaws 3.x components

Mako incorporates the Jaws RIP to interpret PDF (and now PostScript) and to render content to a bitmap from the Mako DOM (Document Object Model). Updating to the latest version of Jaws improves accuracy and performance.

MAKO-1227 Jaws 4 integration: PostScript Input

Mako 5.0 supports PostScript as an input format directly, adding a new input class, IPSInput. This eliminates the need to first convert to PDF with a separate component (Jaws PDFLib).

MAKO-1357 Transition DOM APIs to exceptions for error handling instead of a simple bool pass/fail

This work simplifies a huge number of Mako APIs, and provides more detail when an exception is triggered. It also resulted in classes, enumerations, typedefs and more being renamed for consistency.

Note: These are breaking changes, meaning existing code will have to be modified. Additional information is provided later in this document.

MAKO-1465 Update development tools

Mako 5.0 is built with more recent compiler versions on all platforms. For example, on Windows, Visual Studio 2019 is now supported (and recommended). Mac and iOS builds use the latest SDKs (macOS 10.15, iOS 13.2).

MAKO-1469 EDS (Error Diffusion Screens)

A new class, `IJawsRenderer::CEDSHalftone` is added, enabling images to be rendered using a halftone representing an error diffusion screen. Allows the production of results containing a variable number of gray levels per channel using a range of error diffusion screens.

This class may be used for both monochrome and color rendering with `renderScreened()` and `renderScreenedToFrameBuffers()`. The existing `renderMonochrome()` and `renderMonochromeToFrameBuffer()` are still available, they simply call the new APIs.

Rather than simply offer typical EDS screens such as Floyd-Steinberg and Stucki, the class allows the user to specify parameters that describe the algorithm.

For example:

Floyd-Steinberg

```
// The halftone - this example is Floyd-Steinberg with drop sizes of 3,
// serpentine enabled and no perturbation
IJawsRenderer::CEDSHalftone halftone;
halftone.dropSizes = 3;
halftone.rows = 2;
halftone.columns = 3;
halftone.pixelColumn = 1;
halftone.denominator = 16;
halftone.weights[0] = 0;  halftone.weights[1] = 0;  halftone.weights[2] =
7;  halftone.weights[3] = 0;  halftone.weights[4] = 0;
halftone.weights[5] = 3;  halftone.weights[6] = 5;  halftone.weights[7] =
1;  halftone.weights[8] = 0;  halftone.weights[9] = 0;
halftone.weights[10] = 0; halftone.weights[11] = 0; halftone.weights[12] =
0; halftone.weights[13] = 0; halftone.weights[14] = 0;
halftone.weights[15] = 0; halftone.weights[16] = 0; halftone.weights[17] =
0; halftone.weights[18] = 0; halftone.weights[19] = 0;
halftone.useSerpentine = true;
halftone.perturbation = 0.0;
```

Example code

Mako Converter with custom transform to remove OPI information.

makoconverter.cpp

Many other code examples are included in the SDK, in the `makoapps` folder.

Full API documentation is published online at http://api.globalgraphics.com/mako/5_0_x

To facilitate creating these parameters, a Windows-based utility, Mako EDS Workbench, is available. Learn more and download an installer from the documentation **page.**

MAKO-1788   Provide a method to retain and edit property values attached to form and image XObjects

In PDF, OEMs can put extra key value pairs into PDF XObject dictionaries. This is often used to store in application-specific data, as well as for things like variable data tags.

Until now, Mako has not been preserving these; with this change it will.

Additionally, there are new member functions in `IDOMImageBrush` and `IDOMForm` to get and set the properties:

```
IDOMForm::getPdfPropertiesDictionary()
IDOMForm::setPdfPropertiesDictionary()
IDOMImageBrush::getPdfPropertiesDictionary()
IDOMImageBrush::setPdfPropertiesDictionary()
```

While meeting the requirement expressed in the title of this story, this feature has developed into a generalized method for adding/querying custom entries at the catalog and page levels.

Additionally, there is an API for reading raw PDF objects from a PDF document, which is added to allow developers to reach any object they may wish to pull into a property.

A new interface, `IPDFObjectStore`, gives access to Mako's internal `CObjectStore`

Its APIs allow users to:

- resolve objects
- add objects
- mark objects as dirty (for incremental update purposes)
- create new indirect references (`IPDFReference`)
- generate far references (`IPDFFarReference`) that allow the objects to be referred to from inside image and form properties

`IPage` and `IDocument` acquire new APIs to:

- obtain the object store attached to the page or document
- search the page or document for a given far reference (also locating the `IPDFObjectStore w`here that object was present)

Example: A user wants to add a stream to attach to a form (maybe the contents of the document that was the source material). The steps they would take are:

- Create the `IPDFStream` object for their stream
- Store it in one of the object stores (probably the page but putting it in the document store is fine too) and obtain an indirect reference.
- Create a far reference from that and store that reference in the pdf properties dictionary attached to the form.

The user can now put arbitrary indirect objects however they would like, and via the far reference mechanism can refer to the same objects in multiple places.

Conversely if they see a far reference inside a PDF form properties dictionary that they have loaded, they can ask the `IPage` or the `IDocument` to locate the object and return it to them (or they can look at the object stores themselves).

A new simple example (see `custompdfentries.cpp`), included in the SDK, demonstrates how they can do both things. The sample code also demonstrates how custom entries can be added to a page dictionary (the root object of the page store) or to the catalog (the root object of the document store).

MAKO-1470   Expanded bindings

Mako development in **C#** is introduced in Mako 5.0, implemented using SWIG, an interface compiler that connects programs written in C and C++ with scripting languages.

The Mako build for C# development will not be included in the Mako 5.0 release but is available as a preview version upon request. A NuGet package is available for Visual Studio on Windows. Support for other platforms, such as macOS and Linux, will follow. Getting started documentation can be found **here**.

MAKO-1736   Support 16-bit rendering

Jaws supports rendering to 16 bit-per-pixel images, but previously not exposed in the Mako IJawsRenderer class. For Mako 5.0, an additional parameter controls bit depth (8 or 16).

Support issues fixed

MAKO-1558 (Mako Support Request #10237) Exception thrown from IPage::getContent

MAKO-1745 (Mako Support Request #10268) PlanetPDF Press PDF cannot be processed

MAKO-1926 (Mako Support Request #10311) Exception = Bad arguments passed to an API function

MAKO-1927 (Mako Support Request #10312) Exception = PCL5 Processing Error

MAKO-1929 (Mako Support Request #10310) Incorrect rendering of RGB colors

MAKO-1932 (Mako Support Request #10306) pcl to pdf issue

MAKO-1895 (Mako Support Request #10299) RGB and CMYK give different output

MAKO-1889 (Mako Support Request #10300) PDF causing "Internal RIP error: Rangecheck error"

MAKO-1623 (Mako Support Request #10241) Recovering from an error

MAKO-1986 (Mako Support Request #10320) Mako v4.8.0 - PCL5 - PJL Issue

Other changes

MAKO-1711 Add BigTIFF support to Mako and clean up Mako TIFF encoding

Mako now provides an option for OEMs to encode TIFF as bigtiff (see IDOMTIFFImage::encode()). Huge images can now be safely written.

MAKO-1831 Add abort to IPDFInput::scanPdfForInks

This process can now be safely aborted with IAbort::signalAbort()

MAKO-1859 Provide a method of querying and modifying a PDF MediaBox

New APIs, IPage::getMediaBox() and IPage::setMediaBox(), enable a developer to query or change a PDF document’s MediaBox. Unlike most Mako APIs which expect Mako units (1/96"), these APIs expect PDF units (1/72").

In this example the size of a page is increased by 25mm on all sides:

#define MM2PDFUNITS(value) value / 25.4 * 72.0

```
// Get the page
IPagePtr page = document->getPage(pageIndex);

// New API
FBox mediaBox = page->getMediaBox();

// Modify size
mediaBox.left -= MM2PDFUNITS(25);
mediaBox.top += MM2PDFUNITS(25);
mediaBox.right += MM2PDFUNITS(25);
mediaBox.bottom -= MM2PDFUNITS(25);
page->setMediaBox(mediaBox);
```

MAKO-1374 Add customtransform header to distribution

customtransform.h (see this **page** for more details) is now included in the standard distribution, with improved comments to guide developers using an IDE such as Visual Studio or XCode.

MAKO-1635 Make several previously internal-only transforms public

This opens access to private headers, prompted by a customer's need to make use of the font processor. This change includes:

- `IPatternConverterTransform`
- `IFontProcessorTransform` and `IFontProcessorDeferredTransform`
- `IBlendSimplifierTransform`
- `ISoftMaskConverterTransform`

As they are included in the public interface, they are documented at https://api.globalgraphics.com/mako/5_0_x

MAKO-1713 Make the glyphs clusters API public

- Header file: `edl/iglyphsclusters.h`
- Defines the `IGlyphsClusters class`
- `IDOMGlyphs` now offers `getClusters()` / `setClusters()` methods

As they are included in the public interface, they are documented at https://api.globalgraphics.com/mako/5_0_x

MAKO-716 Improve the "Web" preset to support byte serving

Presets that can be used with `IPDFOutput::setPreset` (or used on the makoconverter command line) are documented **here**:

The web preset has been improved to support byte serving, whereby the PDF cross-reference table is moved to the beginning of the file to allow web browsers random access to the PDF content. For large files this can significantly improve the responsiveness of web browsing

Breaking changes

We took the opportunity in the Mako 5.0 project to make changes aimed at making the APIs more consistent in their naming, behavior or return types. Developers new to Mako will be unaware of these changes, but existing code written for Mako 4.x may require refactoring to work with Mako 5.0.

Exceptions versus Boolean return values

The most extensive change to Mako classes is the use of exceptions to signal a problem, rather than a Boolean return value. Exceptions can provide more detail via an error code than the simple pass/fail of a Boolean.

In most cases, this means that their signature changes too. Instead of passing a pointer to an object that is updated by the call, the method returns the new object. For example, to obtain the colorspace from an `IDOMColor` object in Mako 4.x you would write:

```
IDOMColorSpacePtr inkColorSpace;
inkColor->getColorSpace(inkColorSpace);
```

Whereas in Mako 5.0, you write:

```
DOMColorSpacePtr inkColorSpace = inkColor->getColorSpace();
```

This change often results in simpler code.

Code written for Mako 4.x that tests the Boolean return value to check that the call was successful can be surrounded by a try-catch block. The examples provided with the SDK demonstrate this approach.

This change affects all the most commonly used classes:

`IDOMNode`	`IDOMBrush`
`IDOMColor`	`IDOMPathGeometry`
`IDOMGroup`	`IDOMFont`
`IDOMImage` and `IImage`	`IDOMGlyphs`
`IDOMResource`	`IDOMFunction`

IDOMPage	IDOMPath
IDOMShape	IEDLTempStore

MAKO-1771   Make CEDLVector usable with the C++11 for loop iterator syntax

Mako 5.0 now supports the loop iterator. The following “before and after” code snippets also highlights an example of where a Mako enumerator (`IDOMFigureCollectionEnum`) has been replaced by the more convenient vector, `CDOMPathFigureVect`.

Mako 4.8

```
uint32 getPathPointCount(const IDOMPathGeometryPtr& geometry)
{
    IDOMFigureCollectionEnumPtr figures = geometry-
>getFigureCollectionEnum();
    uint32 pointCount = 1;
    IDOMPathFigurePtr figure;
    while (figures->getNext(&figure))
    {
        IDOMSegmentCollectionEnumPtr segments = figure-
>getSegmentCollectionEnum();
        IDOMPathSegmentPtr segment;
        while (segments->getNext(&segment))
        {
            pointCount += edlobj2IDOMArcSegment(segment) ? 1 : 0;
            pointCount += edlobj2IDOMPolyLineSegment(segment) ?
                edlobj2IDOMPolyLineSegment(segment)->getPointsCount() : 0;
            pointCount += edlobj2IDOMPolyBezierSegment(segment) ?
                edlobj2IDOMPolyBezierSegment(segment)->getPointsCount() :
0;
            pointCount += edlobj2IDOMPolyQuadraticBezierSegment(segment) ?
                edlobj2IDOMPolyQuadraticBezierSegment(segment)-
>getPointsCount() : 0;
        }
    }
    return pointCount;
}
```

Compare to Mako 5.0:

Compare to Mako 5.0

```
uint32 getPathPointCount(const IDOMPathGeometryPtr& geometry)
{
    CDOMPathFigureVect figures = geometry->getFigures();
    uint32 pointCount = 1;
    for (const IDOMPathFigurePtr figure : figures)
    {
        CDOMPathSegmentVect segments = figure->getSegments();
        for (IDOMPathSegmentPtr segment : segments)
        {
            pointCount += edlobj2IDOMArcSegment(segment) ? 1 : 0;
            pointCount += edlobj2IDOMPolyLineSegment(segment) ?
                edlobj2IDOMPolyLineSegment(segment)->getPointsCount() : 0;
            pointCount += edlobj2IDOMPolyBezierSegment(segment) ?
                edlobj2IDOMPolyBezierSegment(segment)->getPointsCount() :
0;
            pointCount += edlobj2IDOMPolyQuadraticBezierSegment(segment) ?
                edlobj2IDOMPolyQuadraticBezierSegment(segment)-
>getPointsCount() : 0;
        }
    }
    return pointCount;
}
```

Known issues

PDF standards compliance

During testing it was found that PDF/X and PDF/A output was failing compliance testing due to the presence of OPI comments. As this issue can be solved with some additional code prior to writing an assembly to a new PDF/X or PDF/A document, it was decided to defer the fix to a planned maintenance release, Mako 5.0.1.

The additional code takes the form of a custom transform. A version of the standard sample, makoconverter, that includes the additional code is attached to this page for you to download and use.

Note: In Mako 5.0, the custom transform is no longer a private header and is available as a regular `#include.` The inline documentation has been improved to make it easier to implement. The attached example code performs a useful function but also serves as an excellent example of custom transform implementation.

Over-italicization of emulated fonts on Ubuntu

During testing it was found that some test jobs with italic text that required an emulated font because the original font was unavailable over-italicized the glyphs, which is to say increased the skew angle excessively.

Other Linux platforms did not exhibit the problem. The workaround is to install the **Microsoft TrueType core fonts** package.

Tooling updates

MAKO-1708   Update Windows build for VS2019 (compiler version 142)

MAKO-1827   Update Mac and iOS builds to use the latest SDKs (macOS 10.15, iOS 13.2)

Both part of an initiative while developing Mako 5.0 to update tooling across platforms

MAKO-1746   Drop 32-bit support for macOS builds

As the last 32-bit Mac came out in 2011 we have dropped 32-bit support. The distribution retains its ‘UB’ label (universal binary) for now.

MAKO-1822   Deprecate 32-bit Windows builds in Mako 5.0

32-bit libraries have been dropped from the principal Windows distribution (Visual Studio 2019). They remain available in the VS2015 distribution but will be dropped altogether in a future release.

Distribution

Mako 5.0 is built for the following platforms:

- Android
- iOS
- Linux (for Debian-based distributions, eg Ubuntu, Mint)
- Linux (Centos)
- Linux (for Debian Jessie)
- Linux (for MUSL distributions, eg Alpine Linux)
- macOS
- Windows (static and non-static libs, VS 2019 (V142) x64 only)
- Windows (static and non-static, VS2015 (V140), x86 and x64)
- Windows SWIG (Preview release of C# libraries)
- Windows UWP (Store apps, Windows 10 IoT)