Adobe Print Services User Guide

Version 1.0

Contents

Contents **2**
List of Tables **6**

1 Preface **8**

1.1 Audience and Purpose 8
1.2 File Paths 8
1.3 Notational Conventions 8

2 Installation and System Requirements **10**

2.1 Overview 10
2.2 Delivery Files 12
2.2.1 APS Versioning 12
2.3 Install the APS SDK on Windows 13
2.4 Install the APS SDK on Linux 13
2.5 Resource Archive Installation (Optional) 14
2.6 Platform-Specific Requirements for the APS Demo 14
2.6.1 Example JSON files 14
2.7 Building the APS Demo App 15
2.7.1 Building on Windows 16
2.7.1.1 Windows System Requirements 16
2.7.1.2 Building the Demo on Windows 16
2.7.2 Building on Linux 17
2.7.2.1 Linux System Requirements 17
2.7.2.2 Building the Demo on Linux 17
2.7.2.3 Building the Demo 17
2.8 APS Demo Command Line Parameters 18

3 Configuring Jobs with input.json **19**

3.1 Job Configuration File 19
3.2 Sample Input JSON Schema Structure 20
3.3 Input JSON Example 22
3.4 Limitation of an Imposition Job 22

4 Report.json file **23**

4.1 Sample Report JSON Structure 23
4.2 Example report.json File 24

5 APS Demo Application **26**

5.1 Demo code workflows 26
5.2 Getting the Library Version 26
5.3 Single Job Mode 26
5.4 Continuous Job Mode 27
5.5 APS Demo Functions 28
5.6 Identifying a Completed Job 29
5.7 Finding the Results 29

6 AI-Driven Optimizations 30

6.1 AI for Processing Efficiency 30
6.2 Manage AI and Manual Configuration 30
6.2.1 Automatic Application of Features Based on other Features 31
6.2.2 AI and Reporting in report.json 31
6.3 AI on, operations=NeverApply, some Features on 33
6.4 AI on, operations=NeverApply, some Features off 33
6.5 AI on, operations=NeverApply, some Feature Parameters Provided 33
6.6 AI on, operations=AlwaysApply, some Features on or off 34

7 Print Quality Enhancements 35

7.1 Image Resampling 35
7.1.1 Image Down-sampling 36
7.1.2 Image Up-sampling 38
7.1.2.1 Sensei algorithm 38
7.2 Image Stitching and Pattern Stitching 41
7.2.1 Image Stitching 41
7.2.1.1 Limitations 42
7.2.2 Pattern Stitching 43
7.3 Outlined Text, Barcode, and QR Code Detection 43
7.3.1 Input JSON Parameters 44
7.3.1.1 Doc-Level Detection Mode 44
7.3.1.2 Page-Level Detection Mode 44
7.3.1.3 Object-Level Detection Mode 45
7.3.2 Outlined Object Post-Processing Examples 46
7.3.3 Barcodes and QR Codes Support 48
7.3.3.1 Detectable Barcode and QR Codes 48
7.3.3.2 Unsupported Barcode and QR Codes 48
7.3.3.3 Limitations 49
7.4 Source Color Replacement 49
7.4.1 Shade Support 50
7.4.2 Input JSON Parameters 51
7.4.3 Spot Aliasing and Spot Alternate Modification 53
7.4.4 Live Mode 54
7.4.5 APIs 54
7.4.6 Limitations 55

8 Performance and Processing Optimizations 56

8.1 SMask and Mask Removal 56
8.2 XObject Reuse using VDP Optimization 57
8.3 Identical GState Reuse to Improve Caching 59
8.4 Form XObject Optimization 61
8.4.1 Empty Form XObject Removal 61
8.4.2 PDF Form BBox Reduction 61
8.5 Duplicate Font Dictionary Removal 62
8.6 Duplicate Resource Remover 64
8.7 Non-Visible Graphic Content Removal 66
8.8 Image Cropping 67
8.8.1 Limitations 69
8.9 Irrelevant Overprint Attribute Removal 69

8.10 AutoGR Pre-Analysis ... 71
8.11 GR Flattening ... 74
8.11.1 Usage and Workflow ... 75
8.11.2 Input JSON ... 75
8.12 Invalid Image Data Detection and Restoration ... 76
8.13 Page Resource Optimization ... 78
8.14 Zero Alpha Object Removal ... 79
8.15 Image Vectorization ... 80
8.15.1 Limitations ... 82

9 Object Inspector ... 83

9.1 Overview ... 83
9.2 Input JSON: ... 83
9.2.1 Parent JSON ... 83
9.2.2 Child JSON ... 84
9.3 Output JSON ... 85
9.3.1 Error Handling in different scenarios: ... 90
9.4 Object Inspector API ... 91

10 Post-Press Enhancements ... 92

10.1 Image Cutpath Generation ... 92
10.1.1 Input JSON ... 93
10.1.2 Limitations ... 94
10.2 Bin Packing ... 95
10.2.1 Input and Output Image Examples ... 95
10.2.2 Input JSON ... 96
10.2.3 Limitations ... 97

11 Error Detection and Analysis ... 99

11.1 Efficient Ripping with AllRGBorAllGray ... 99
11.2 Font Repair ... 100
11.2.1 TrueType Invalid maxp Table Problem ... 101
11.2.2 Type 3 Invalid Bounding Box Problem ... 102
11.3 Report Font Hinting Errors ... 103
11.4 PDF Validation ... 104
11.5 Page Tree Structure Reorganization ... 105
11.6 Running Font Hinting Errors and PDF Validation in Isolation ... 107
11.7 PDF Properties Report Feature ... 108
11.7.1 Top-level PDF Properties ... 109
11.7.2 Document-Level Attributes ... 110
11.7.3 Page Level Attributes ... 112
11.7.4 PDF Properties Example report.json ... 113

12 Creating a Custom App ... 121

12.1 APS SDK API ... 121
12.2 Obtain SDK Version Information ... 121
12.3 Submit a Job for Processing ... 121
12.4 Linking with the APS SDK ... 122
12.5 Files Required at Runtime ... 122
12.6 Windows-Specific Visual Studio Runtime Requirements ... 122

13 Frequently Asked Questions (FAQ).. 123

Index 124

List of Tables

Table 1 Notational conventions . . . 8
Table 2.1 APS Features . . . 11
Table 2.2 Delivery files for each Release . . . 13
Table 2.3 Windows Installation SDK Delivery Files . . . 13
Table 2.4 Linux Installation SDK Delivery Files . . . 14
Table 2.5 Sample input.json files . . . 15
Table 2.6 System Requirements: Windows (64-bit) . . . 16
Table 2.1 System Requirements: Linux Ubuntu 22 64-bit . . . 17
Table 2.2 APS Demo Application: Command Line Parameters . . . 18
Table 3.1 input.json JSON Schema Keys . . . 20
Table 4.1 General schema for report.json . . . 23
Table 4.2 None entry of the summary dictionary of report.json . . . 23
Table 4.3 GeneralInfo entry of the summary dictionary of report.json . . . 24
Table 4.4 JobStatus entry of the summary.GeneralInfo dictionary of report.json . . . 24
Table 4.5 Build Info entry of the summary.GeneralInfo.JobStatus dictionary of report.json . . . 24
Table 5.1 APS Demo: Supported Functions . . . 28
Table 6.1 AI integration: unsupported Features . . . 30
Table 7.1 Down-sampling Feature: Configuration Parameters . . . 36
Table 7.2 Up-sampling: Configuration Parameters . . . 39
Table 7.3 Image Stitching: Configuration parameters . . . 42
Table 7.4 OutlinedTextDetection: JSON input parameters . . . 44
Table 7.5 One Dimensional (Linear) Codes . . . 48
Table 7.6 Two Dimensional Codes . . . 48
Table 7.7 One Dimensional (Linear) Codes . . . 48
Table 7.8 Two Dimensional Codes . . . 49
Table 7.9 Color Replacement: JSON Configuration Parameters . . . 51
Table 8 Spot Aliasing: JSON Configuration Parameters . . . 53
Table 8.1 VDP optimization: Configuration Parameters . . . 58
Table 8.2 PDF Form BBox Reduction: Configuration Parameters . . . 62
Table 8.3 DropObjectsOutsideClip: Configuration Parameters . . . 67
Table 8.4 Image Cropping: Configuration Parameters . . . 68
Table 8.5 AutoGR Pre-Aanalysis: Required page dictionary key . . . 71
Table 8.6 input.json: AutoGR Configuration Parameters . . . 72
Table 8.7 AutoGR Flattening: Global Rasterization Parameters . . . 75
Table 8.8 AutoGR Flattening: Resolving Transparency Parameters . . . 76

Table 8.9 PageRsrcFix: Configuration Parameters 79
Table 8.10 ImageVectorize: Configuration Parameters 81
Table 9.1 Object Inspector: Parent JSON Configuration Parameters 84
Table 9.2 Object Inspector: Child JSON Configuration Parameters 84
Table 9.3 Object Inspector: Output JSON Fields 85
Table 9.4 Object Inspector: Output JSON Fields for Transparency Groups 86
Table 9.5 Object Inspector: Output JSON Fields for ObjectList 86
Table 9.6 Object Inspector: Output JSON Fields for FunctionList 89
Table 10.1 Image Cutpath Generation Feature: Configuration Parameters 93
Table 10.2 BinPacking Feature: Configuration Parameters 96
Table 11.1 FontRepair: Configuration Parameters 100
Table 11.2 Page Tree Repair: Configuration Parameters 107
Table 11.3 PDF Properties Reporting: General Properties (Required) 110
Table 11.4 PDF Properties Reporting: Document Properties (Required) 111
Table 11.5 PDF Properties Reporting: Page-Level Properties (Required) 112
Table 13.1 Frequently Asked Questions 123

1 Preface

1.1 Audience and Purpose

This guide is intended for OEM partners and consulting engineers who assist OEM partners. It describes the process of integrating APS into your products. It provides information about all the necessary APIs exposed in the APS shared library that are required for implementing a configuration appropriate for your products. The functionality is also implemented in the APS Demo (also known as the "TachyonDemo") provided with the download and described here.

The reader should be familiar with PDF and rendering.

1.2 File Paths

This guide uses the term *`directory`* instead of *`folder`* when talking about the organization of files on the system. The directory where APS files are installed is called `<Your install root>`. Paths to specific APS SDK files and directories are relative this directory. For example, the path `EXE_Tachyon/` is shorthand notation for `<your install root>/EXE_Tachyon/` which indicates that this is a subdirectory of your Adobe Print Services installation directory.

1.3 Notational Conventions

This guide uses typefaces to identify the code elements. The following table describes this document's typeface usage.

Table 1 Notational conventions

Typeface	How used	Example
None	File names	input.json
None. CamelCasing	PDF Cos objects are identified by their camel-cased name.	BBox
`Monospaced`	Code samples and object in any language; names of functions, types, fields, and constants defined in the C or C++ languages; names of classes or methods defined in the C++ language; file names and paths. Standalone object names not in text are not formatted. For example, a parameter name by itself in a table cell.	`pdf_output_url` specifies the full file path to output PDF file
`Monospaced`	Directory paths	`<your install root>/ EXE_Tachyon/Release/`

Table 1 Notational conventions

Typeface	How used	Example
`Monospaced Italic`	Variable items for which you must substitute a value, such as a parameter value or user-defined file name	The default value for `XYZparameter` is *`true`*.

2 Installation and System Requirements

2.1 Overview

Adobe Print Services 1.0 is a software development kit (SDK) that leverages the Adobe PDF Library for analyzing and optimizing print jobs. When coupled with Adobe PDF Print Engine (APPE), APS delivers an unparalleled printing experience and is an essential tool for improving print quality and speeding up processes in commercial printing applications. Key benefits include:

- **Print quality**: Image sampling, image and pattern stitching, smask removal, and more.
- **Performance optimization**: VDP optimization, object reuse, unneeded object removal, page tree reorganization, and other features increase performance, thereby reducing time-to-print.
- **Repair invalid PDFs**: Repair fonts and other items so that the PDF conforms to PDF ISO standards.
- **Artificial intelligence**: AI and machine learning integration leverage advanced algorithms so that you can design scalable print workflows with minimal intervention. APS supports a "mix and match" approach so that you can use AI with manual configuration to balance APS processing efficiency with print quality across a broad spectrum of input jobs.
- **In-depth analysis**: Analyzes PDFs in detail and reports actionable findings to help you optimize your job ticket; for example, it identifies which pages which are gray only.
- **Post Press Enhancements**: Print post-press optimization features like Image Cutpath Generation and Bin Packing that optimize material usage and production efficiency through intelligent automation.

The SDK includes a dynamic shared library as well as a demo application that helps you evaluate the offering in actual printing environments. Several examples are provided with the demo app to get you started. Input configuration and output files are in a simple JSON format. APS analyzes and optimizes PDFs based on user-specified configuration parameters, makes any needed repairs, and then provides a detailed report itemizing the issues and what feature was applied to fix the problem.

Table 2.1 lists the print quality, performance optimization, and error checking features you can enable for PDF analysis prior to sending for printing. You can apply these features individually or in tandem with other features. You can also reduce manual configuration with the APS artificial intelligence (AI) feature to leverage it’s machine learning algorithms for predicting when a feature should be evaluated and applied to each processed PDF.

Note that some of these features have dependencies on other features, and the rules are as follows:

- When any of Duplicate Resource Remover, Duplicate Font Dictionary Removal, PDF Validation, or Identical GState Reuse to Improve Caching are enabled, then Page Resource Optimization is also automatically enabled.
- Page Tree Structure Reorganization is always enabled unless explicitly disabled at the feature level. That is, when `features.PageTreeFix.operation` is *`NeverApply`*. This behavior differs from other features in that it doesn't obtain its default from `features.operation` when not explicitly specified at the feature level.

Table 2.1 APS Features

Feature	Feature Label	Benefit	Predictable by AI?
Image Resampling	ImageSampler	Print quality	Yes
Image Stitching and Pattern Stitching	ImageStitching	Print quality	Yes
Outlined Text, Barcode, and QR Code Detection	OutlinedTextDetection	Print quality	No
Source Color Replacement	SourceColorReplacement	Print quality	No
SMask and Mask Removal	SMaskRemoval	Performance optimization	Yes
XObject Reuse using VDP Optimization	VDPOptimization	Performance optimization	Yes
Identical GState Reuse to Improve Caching	FormXobjectGStateOptimization	Performance optimization	Yes
Empty Form XObject Removal	FormXobjectOptimization	Performance optimization	Yes
PDF Form BBox Reduction	FormXobjectOptimization	Performance optimization	Yes
Duplicate Font Dictionary Removal	FontMerge	Performance optimization	Yes
Duplicate Resource Remover	DuplicateResourceRemover	Performance optimization	Yes
Non-Visible Graphic Content Removal	DropObjectsOutsideClip	Performance optimization	Yes
Image Cropping	ImageCropper	Performance optimization	Yes
Irrelevant Overprint Attribute Removal	IrrelevantOP	Performance optimization	Yes
AutoGR Pre-Analysis	GRHint	Performance optimization	No
GR Flattening	Flattener	Performance optimization	No
Invalid Image Data Detection and Restoration	ImageValidation	Performance optimization	Yes
Page Resource Optimization	PageRsrcFix	Performance optimization	Yes
Zero Alpha Object Removal	ZeroAlphaRemoval	Performance optimization	Yes
Image Vectorization	ImageVectorize	Performance optimization	No
Object Inspector	ObjectInspector	PDF inspection	No
Image Cutpath Generation	ImageCutPath	Post-press enhancement	No
Bin Packing	BinPacking	Post-press enhancement	No
Efficient Ripping with AllRGBorAllGray	AllRGBorAllGray	Error detection/repair	Yes
Font Repair	FontRepair	Error detection/repair	Yes
Report Font Hinting Errors	ReportHintingErrors	Error detection/repair	Yes
PDF Validation	PDFValidation	Error detection/repair	Yes
Page Tree Structure Reorganization	PageTreeFix	Error detection/repair	Yes
PDF Properties Report Feature	PDFProperties	Error detection/repair	No

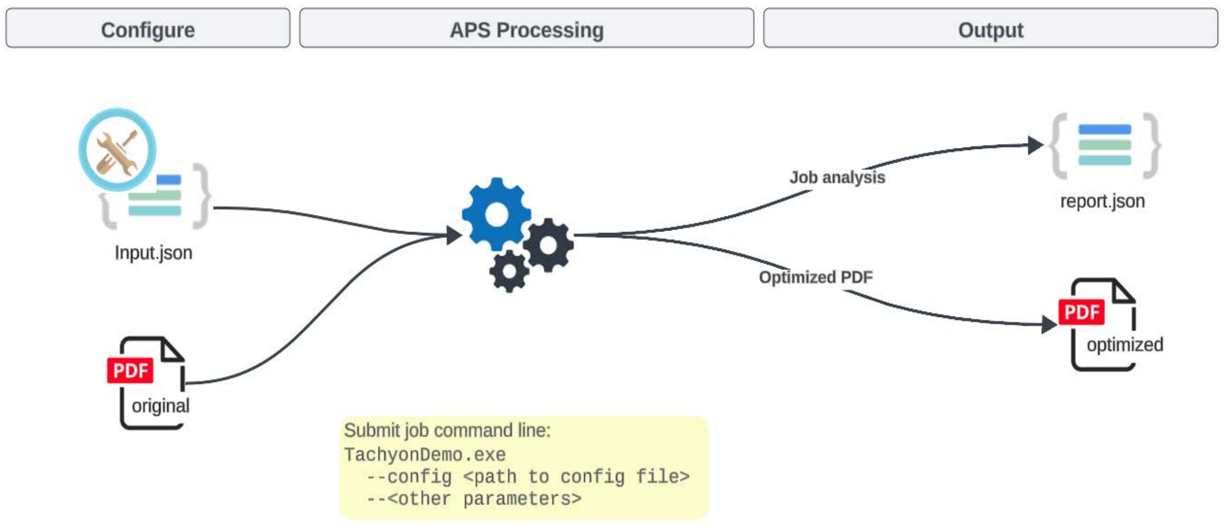
Once deployed, optimizing print jobs via APS is easy and involves few steps:

1. Configure job processing in input.json. You'll specify the absolute input file path, optimized output file path, and report.json file path as well as the features to apply.

2. Run the job; for example, from the command line. You'll specify the location of the input.json config file as well as a location to place that file when the job completes.

3. Job output includes the optimized PDF as well as a report.json file which describes what features were applied in what location in the PDF.

Figure 1 Basic job workflow



```
Submit job command line:
TachyonDemo.exe
  --config <path to config file>
  --<other parameters>
```

2.2 Delivery Files

Deliverable names identify the Adobe Print Services version number as well as the names of platform-specific archive files indicating the relevant platform and type of file/archive.

2.2.1 APS Versioning

APS versions SDK downloads with a four-part version number in the form of v.v.v.v: major version.minor version.build type.build ID-platform ID-SDK.filetype:

```
AdobePrintServices-v.v.v.v-<platform Win|Linux>-SDK.<filetype zip|tgz>.
```

For example, the SDK for Windows major version 1, minor version 0, build type 4000, and build ID 0 would be:

```
AdobePrintServices-1.0.4000.0-Win64.zip
```

Other resources use a similar, but truncated, versioning methodology:

- Release notes and documentation use a 3 part version; for example, AdobePrintServices-1.0.4000
- Resource archives only uses the major version number. The trailing number indicates the update number of the archive. For example, AdobePrintServices-1.x-Resources-1.

Table 2.2 Delivery files for each Release

Delivery file name	Contains
AdobePrintServices-v.v.v.v-<platform>-SDK.<filetype>	Platform-specific APS SDK along with demo code
AdobePrintServices-*v.v.v*-Documentation-SDK.pdf	Platform-independent PDF Guide describing the APS SDK
AdobePrintServices-1.x-Resource-v.N.<filetype>	Platform-independent APS SDK example input.json files, code samples, performance data, and so on.

2.3 Install the APS SDK on Windows

To install APS on Windows:

1. Download the APS package.
2. Extract the archives to your local drive.

    Tip: This document refers to the extracted directory such as "AdobePrintServices-1.0.4000.0-Win64" as `<your install root>`.

3. Navigate to the APS installation directory: `<your install root>`.
4. Verify the presence of the items in Table 2.3.

Table 2.3 Windows Installation SDK Delivery Files

Directory Name	Description
Demo	Contains the APS demo build files.
EXE_Tachyon	Contains the product working directory for executing Adobe Print Services which includes the pre-built demo executable and the files it needs to run.
Shared	Contains the API interface files, font resources, libraries and color profiles.
BuildTachyon.sln	The Visual Studio solution file for rebuilding the APS demo application on Windows.
cmake_Tachyon	Contains utilities used by APS CMake build system

2.4 Install the APS SDK on Linux

To install APS on Linux:

1. Download the APS package.
2. Extract the archive file using the `tar` command with the options *`zxvf`* and the path to your download directory.

```
tar -xvzf AdobePrintServices-vvvv-Linux-SDK.tgz
```

3. Verify the presence of the items in Table 2.4.

Table 2.4 Linux Installation SDK Delivery Files

Directory Name	Description
Demo	Contains the APS demo build files.
EXE_Tachyon	Contains the product working directory for executing Adobe Print Services which includes the pre-built demo executable and the files it needs to run.
Shared	Contains the API interface files, font resources, libraries and color profiles.
cmake_Tachyon	Contains utilities used by APS CMake build system

2.5 Resource Archive Installation (Optional)

The APS resource archive provides both sample input.json files as well as a set of files to help you evaluate the effectiveness of APS. It includes both sample code and example input.json configuration files.

To install:

1. Download and extract the resources archive into the already existing SDK directory (the location is not required, but it is recommended).
2. Confirm the following is present:
   - `AdobePrintServices-1.x-Resources-1/ExampleInputJsons`

2.6 Platform-Specific Requirements for the APS Demo

The APS demo delivered with the APS SDK is called "TachyonDemo". The demo source code and related files reside in `<Your install root>/Demo/`.

2.6.1 Example JSON files

`ExampleInputJsons/` contains examples of input.json files to meet various product goals. You are not limited by the combinations shown in these examples and should leverage any combination of features which are appropriate for your product. You must modify the paths mentioned in these files before you try them yourself.

Table 2.5 Sample input.json files

Example File	Notes
SampleJson1_ PDFValidation_ReportHintingErrors.json	Demonstrates PDF validation and error reporting. All features are disabled except PDFValidation and ReportHintingErrors.
SampleJson2_ PDFProperties.json	Demonstrates how to analyze a PDF to obtain crucial details you'll want to leverage when creating your job ticket. It disables all features except PDFProperties, and demonstrates how to ask for specific items to be reported.
SampleJson3_ ImgSampler_ImgStitch_IrrelevantOP_.json	Demonstrates three features by disabling all features except ImageSampler, ImageStitching and IrrelevantOP. This will merge adjacent images into a single image and then upsample or downsample as appropriate using the best bicubic sampling algorithm for that image. It also removes irrelevant overprint to avoid unnecessary transparency processing.
SampleJson4_ MultipleFeatures.json	Demonstrates AllRGBorAllGray, SMaskRemoval, ZeroAlphaRemoval, VDPOptimization, and DuplicateResourceRemover. All other features are disabled.
SampleJson5_ MultipleFeatures.json	Demonstrates FontRepair, GRHint, FormXobjectGStateOptimization, VDPOptimization DuplicateResourceRemover. All other features are disabled. Note that deviceConfiguration has been added to support the requirements of GRHint.
SampleJson6_ PredictApplicableFeatures.json	Demonstrates the AI-based PredictApplicableFeatures feature which uses machine learning to specify what features should be applied to an input PDF. All other features are disabled.
SampleJson7_OutlineTextDetection.json	Demonstrate OutlinedTextDetection with OutlinedTextDetectionMode=3 (Insert hints specific to each PDF vector object if it is an outlined text, barcode, or QR code). Only OutlinedTextDetection feature is enabled.
SampleJson8_ImageVectorization.json	Demonstrates Image Vectorization feature that identifies suitable raster images and converts it to vector graphics. All other features are disabled
SampleJson9_ObjectInspector_SrcClrRepla cement.json	Demonstrate Source Color Replacement and then checking the object details with Object inspector. All other features are disabled.
SampleJson10_CutPathGeneration.json	Demonstrates CutPathGeneration feature to generate cut paths around the artwork outer boundary. All other features are disabled.
SampleJson11_BinPacking.json	Demonstartes BinPacking feature that repeats artwork on provided sheet size. All other features are disabled.
SampleJson12_GRFlattening.json	Demonstartes GRFlattening that provides modular flattening service and support for AutoGR-based rasterization workflows, since the top most priority is given to the AutoGR, so the file process through this json would take Auto GR route since AutoGR is enable.

2.7 Building the APS Demo App

After extracting the APS files, you should rebuild the release version of the APS demo executable. Doing so introduces you to the build process on your platform and verifies that all the necessary source and development files are accessible in the expected locations. Building a release version of the TachyonDemo executable creates files in <your install root>/EXE_Tachyon/Release/ which is the product working directory. The newly built files replace the existing files in this directory.

Building the release version of the APS demo executable creates these files in `<your install root>/EXE_Tachyon/Release/` product working directory. The newly built files replace the existing files in this directory.

2.7.1 Building on Windows

2.7.1.1 Windows System Requirements

APS on Windows is developed with Visual Studio 2022. During APS deployment, you must ensure that the compatible runtime libraries are installed in your customers' runtime environment. For example, if your product has an installer, you could distribute the VS<version>-compatible C/C++ runtime libraries to your customers as a part of your product's installer. Alternatively, you could instruct your customers to install the runtime libraries on every machine on which they run the APS demo app.

Table 2.6 System Requirements: Windows (64-bit)

Component	Version
Microsoft Visual Studio Enterprise 2022	17.12.2 + 35521.163
Microsoft .NET Framework	4.8.09032
Component: Windows 11 SDK	Version: 10.0.26100.0

1. If the runtime libraries are not installed on a particular machine, download the Visual C++ Redistributable for Visual Studio 2022 from Microsoft. For additional information about deployment of Visual C++ applications, see <https://docs.microsoft.com/en-us/visualstudio/>.

Note: An executable product built with Visual Studio cannot be launched unless compatible C/C++ runtime libraries (DLLs) are installed in the appropriate directory in the runtime environment. These libraries must be *installed*; it is not sufficient simply to copy the files.

2.7.1.2 Building the Demo on Windows

To build the APS demo:

- In the unzipped package directory, run:

```
cmake -A "x64" -G "Visual Studio 17 2022" ./Demo/cmake/TachyonDemo -B ./Demo/
build_cmake/TachyonDemo -DCMAKE_SYSTEM_VERSION=10.0.26100.0
```

- Convert absolute paths to relative paths in vcxproj files:

```
python cmake_Tachyon/Utilities/convertPathAbsToRelative.py
```

- Open the solution file <your install root>/BuildTachyon.sln in Microsoft Visual Studio.
- In Visual Studio, choose the **release** solution configuration.
- From Visual Studio's top menu, select **Build > Rebuild Solution**.

  This builds the APS demo executable, linking in the necessary libraries. APS demo source code is compiled and linked against shared dynamic link libraries.
- When Visual Studio finishes building, navigate to the product working directory <your install root>/EXE_Tachyon/Release/. Verify that the files in this directory were just created or copied.
- Verify the application runs:

```
cd <your install root>/EXE_Tachyon/Release

TachyonDemo.exe --version
```

Note: The Windows SDK Release build is digitally signed. If you recompile the SDK demo code with the Smart App Control enabled (Windows 11 and later), you must re-sign the executable for the binaries to work correctly.

2.7.2 Building on Linux

2.7.2.1 Linux System Requirements

Before you build the TachyonDemo on a Linux platform, install prerequisites.

Table 2.1 System Requirements: Linux Ubuntu 22 64-bit

Component	Version
Clang Compiler	18.1.8
GLIBC	2.35

2.7.2.2 Building the Demo on Linux

To install and setup the Clang compiler, run the following:

```
curl -fsSL https://apt.llvm.org/llvm-snapshot.gpg.key | gpg --dearmor > /etc/
apt/trusted.gpg.d/llvm18.gpg


chmod 0644 /etc/apt/trusted.gpg.d/llvm18.gpg
echo "deb http://apt.llvm.org/$(lsb_release -cs)/ llvm-toolchain-$(lsb_release -
cs)-18 main" > /etc/apt/sources.list.d/llvm18.list

apt-get update

apt-get install -y build-essential libtiff5-dev clang-18 llvm-18 libllvm18 llvm-
18-runtime libclang1-18 clang-format-18 lldb-18 lld-18 libc++-18-dev libc++abi-
18-dev libomp-18-dev

snap install cmake –classic
```

Note: The `PredictApplicableFeatures` AI feature has a dependency on the libomp utility.

2.7.2.3 Building the Demo

To build the APS demo:

1. Navigate to `<Your install root>`. You should see `cmake_Tachyon`.
2. Set up the compiler paths:

```
export CC=/usr/lib/llvm-18/bin/clang
export CXX=/usr/lib/llvm-18/bin/clang++
```

3. Run the following:

```
cmake_Tachyon/Utilities/clean_script.sh CMAKE_BUILD_TYPE=Release

cmake -S ./Demo/cmake/TachyonDemo -B ./Demo/build_cmake/TachyonDemo -
DCMAKE_BUILD_TYPE=Release
```

4. Convert absolute paths to relative paths:

```
python3 cmake_Tachyon/Utilities/convertPathAbsToRelative.py
```

5. Build the app:

```
cmake --build ./Demo/build_cmake/TachyonDemo -j 10
```

6. When the build concludes, view the product working directory at <your install root>/EXE_Tachyon/Release/. Check the file dates to verify that the files in this directory were just created or copied.

7. Verify the application runs by executing:

```
./TachyonDemo  --version
```

2.8 APS Demo Command Line Parameters

The demo app supports the parameters listed in Table 2.2. See Chapter 5, "APS Demo Application" for a detailed description of the workflow and how the APS demo parses these parameters.

Table 2.2 APS Demo Application: Command Line Parameters

Parameters	Description
--version	Displays the version number of APS library and exits.
--help	Displays the help message and exits.
--watchdir	Specifies a directory the APS demo monitors for input.json job configuration files. This parameter is valid only in continuous job mode. See "Continuous Job Mode" on page 27. **Default**: current working directory
--watchdircompleted	Specifies a directory of all job configuration files processed by the APS demo. This parameter is valid only for continuous or multi-job modes when --config is unspecified. See "Continuous Job Mode" on page 27. **Default**: current working directory
--config	Specifies the full path of job input.json configuration file. URLs are not supported. On Windows, for example: `TachyonDemo.exe --config ./directory/input.json` This parameter is valid only for single job mode. (See "Single Job Mode" on page 26). The input.json contains the path to the input PDF file along with the settings/features/optimizations to apply to the input PDF.

3 Configuring Jobs with input.json

input.json contains a job's configuration options, and the schema provided here describes all of the options supported in this release. You can configure global settings, individual feature settings, whether or not to enable AI evaulation and repair, or any mix of these as needed.

The input.json is configured and submitted to the TachyonDemo in single job mode or continuous job mode. For details about using the file in these job modes, see "Continuous Job Mode" on page 27.

> **Tip:** There are several example input.json configuration files provided in the resource archive delivered with this documentation guide. See `<your install root>/Resources/ExampleInputJson`.

3.1 Job Configuration File

An input.json configuration file is in JSON format. The configuration file (or job submission file) is configured with the job options APS should use when processing a PDF; that is, data configured in `input.json` file controls the capabilities of APS special features.

The configuration file contains a list of feature and feature properties that define APS operations for each, individual job. The feature catalog is built with a `features` field. This field is a dictionary type resource that lists the features and how they should be applied.

The top-level parent `features` resource has the following properties:

- <Operation control> `operation` (string).
- <Individual feature name>. The name of 1-N specific features appearing as children of the feature resource. For example: `FontRepair` (object)
  - <Operation control> `operation(string)` This operation key is optional. If missing at the feature level, it inherits the global `operation` value. It controls whether to apply this feature to input PDF (values may be *`AlwaysApply`*, *`NeverApply`*, *`AskBeforeApplying`*)
  - Optional setting parameters to control this individual feature; for example, `TrueType`.

Note: JSON objects in input.json are case-sensitive, including feature names, parameters, and other fields.

As shown below, the `FontRepair` feature has two properties to control font repair operation in input PDF (`TrueType,operation`).

Example 3.1: input.json: FontRepair

```
"FontRepair":{
     "operation":"AlwaysApply",
     "Type3":true,
     "TrueType":true
}
```

3.2 Sample Input JSON Schema Structure

The input.json file has a clear specification in the following JSON schema. The JSON schema properties specify every property type, their options, and a description (Table 3.1).

Table 3.1 input.json JSON Schema Keys

Key	Required	Type	Description
TachyonVersion	Yes	integer	The version number of the file format. The version number will increase as the file format is revised. `"TachyonVersion": 1` is the current version. An increase in the version value indicates that more fields have likely been added to the configuration file format/schema. If the specified version does not match the current version of the deployed product, an error is displayed and the job does not process further. For example, if `TachyonVersion` is specified as 2 when version 1 is deployed, then the following appears: Error: I:\ExampleInputFiles\PrintMyPDF.json version 2 != 1
pdf_url	Yes	string	The absolute file path of PDF file to optimize. It can be an absolute path or file:///<path to the input PDF>. For example: "I:\ExamplePdfFiles\PrintMyPDF.pdf"
pdf_output_url	No	string	The absolute path or file:///<path> to the optimized PDF file. If the path is invalid, the default file path is the file path of pdf_url ('.pdf' is replaced with '.opt.pdf').
report_url	No	string	The absolute path or file:///<path> of report.json. If the path is invalid, the default file path is the file path of `pdf_url` ('.pdf ' is replaced with '.report.json'). For example: "report_url" : "I:/ReportFiles/ReportMyPDF.json",
tmpdir	No	string	The absolute path of the directory which will be utilized for temporary storage during processing. These files are deleted at the end of job. **Default**: the current working directory
config	No	dict	Specifies a job's general configuration parameters needed by APS itself or shared by several features.
config.timeout	No	integer	Specifies the number of seconds after which the job cancels if job processing has not completed. For example: "config":{"timeout":100 }

features	Yes	dict	Specifies the configuration parameters of all features. It contains the default configuration parameters to control the job operations for all features. For example:
			`"features":{`
			`    "operation":"NeverApply",`
			`}`

Table 3.1 input.json JSON Schema Keys

Key	Required	Type	Description
operation	No	string	Specifies the default operation mode for all features which do not explicitly override this value. This key may also be used at the individual feature level to override the global, default value. Note that when enabled, `PredictApplicableFeatures` may override these settings. Possible values: ● *`NeverApply`* (default) ● *`AlwaysApply`* ● *`AskBeforeApplying`*
features.operation	No	string	Specifies the operation mode for specific features. This mode overrides operation values. For example, if the parent operation mode is *`NeverApply`* and the feature operation mode is *`AlwaysApply`*, the feature is applied. Possible values: ● *`NeverApply`* (default): Specifies that APS should not apply this individual feature operation to the input PDF. *`NeverApply`* is the default operation mode applied to all features. ● *`AlwaysApply:`* Specifies that APS should apply this individual feature operation to the input PDF. ● *`AskBeforeApplying`*: Specifies that APS should report in report.json on the PDF structure as well as places where a feature can be applied rather than always applying it.
PageRange	No	Range (Array of 2 numbers)	A global level page range input. A single page range can be specified using this parameter. This attribute specifies what range of pages to process, thereby allowing control of the number of processed pages. ● If second value in range exceeds total pages in document or is negative, it will be replaced with total number of pages in the document. ● The page indexing is zero based i.e. first page in the PDF is indexed as 0. The first value in range is inclusive, the second is exclusive. ● This parameter applies to page level features--not to document level features. ● A valid range is recommended to ensure desired results. "PageRange" : [a, b] a, b should be in range of [0, total number of PDF pages] where a < b. Format: `"PageRange" : [a, b]` `"PageRange": [0, -1]` results in processing all pages.
SubPageRange	No	array of integers or array of array of integers	A global level page ranges input which supports multi-page range(s) as well as individual page(s). It overides the `PageRange` parameter. ● If both the `PageRange` and `SubPageRange` are not specified, then all pages in the document are processed. ● This parameter applies to page level features--not to document level features. ● The page indexing is zero based i.e. first page in the PDF is indexed as 0. Both values in ranges are inclusive. ● A valid range is recommended to ensure desired results. For example, `"SubPageRange" : [ a, [b, c], d, [e, f],...]`, these values should be in range [0, total number of PDF pages - 1] Format: `"SubPageRange" : [ a, [b, c], d, [e, f],...]`
password	No	string	The PDF owner password, if any.

Table 3.1 input.json JSON Schema Keys

Key	Required	Type	Description
outfile_perm	No	string	File permission of the optimized PDF file; for example, "0644"
outdir	No	string	The absolute path of the directory for saving optimized PDF files.
log_url	No	strings	The absolute path to the directory for saving log files. If the path is invalid, then the logs print only to the console.

3.3 Input JSON Example

It may be helpful to leverage the following example input.json as you create your configuration. This example input.json defines the parameters, options, settings and preferences applied on a Windows operating system.

Example 3.2: input.json: Generic configuration file

```
{
  "TachyonVersion": 1,
  "tmpdir": "I:/TempDir/",
  "pdf_output_url": "I:/ExampleOutputPdfFile/PrintMyPdf.opt.pdf",
  "pdf_url": "I:/ExampleInputPdfFile/PrintMyPdf.pdf",
  "report_url": "I:/ExampleReportFile/report_PrintMyPdf.json",
  "features": {
    "operation": "NeverApply",
    "AllRGBorAllGray": {
      "operation": "AlwaysApply",
    },
    "FontRepair": {
      "operation": "AlwaysApply",
      "Type3": true,
      "TrueType": true
    }
  },
  "config": {
    "timeout": 100
  }
}
```

3.4 Limitation of an Imposition Job

APS does not support imposition jobs or JDF parsing. It works on standalone PDFs and is agnostic of the imposition job in which the input PDF might be run. This means that APS applies optimizations based solely on the input PDF’s properties, and APPE may produce wrong output when this PDF is part of an imposition job. For example, the `IrrelevantOP` feature removes overprint from objects on a PDF page residing at the bottom of the drawing stack. However, if the PDF is imposed on top of some other PDF in an imposition job, then the backdrop PDF would not have an overprinted appearance in the final raster.

4 Report.json file

After the sample input.json is configured and submitted to APS for processing, APS stores the generated report for each job submission in the path specified by `report_url`:

Example 4.1: Setting the report.json file path

```
"report_url" : "I:/ExampleReportFile/report_PrintMyPdf.json",
```

In the above example, the generated report, `report_PrintMyPdf.json` is stored at `I:/ExampleReportFile/` path.

4.1 Sample Report JSON Structure

The JSON schema that the generated output JSON must match or validate to is as follows:

Table 4.1 General schema for report.json

Key	Required	Type	Description
summary	Yes	dict	A dictionary of the summarized results for each feature. It may include an entry "None" for attributes which are generated but not specific to any feature. It will also contain a GeneralInfo dictionary describing the job status, time and APS build info.
summary.<feature name>	No	dict	A summary for each individual feature that was executed. In general, all feature summaries include the `num` and `msg` attributes. Refer to the feature specific documentation for other attributes that will appear here.
summary.<feature name>.num	No	integer	Specifies the number of feature specific issues identified or corrected in the processed PDF.
summary.<feature name>.msg	No	string	A human-readable (English) summary of the action taken by the feature.
summary.None	Yes	dict	See Table 4.2. Describes number of pages in the document.
summary.GeneralInfo	Yes	dict	See Table 4.3. A dictionary of general information, such as job status, job time and APS build info.
pdffile	Yes	string	Absolute path of the input PDF file.
outputfile	Yes	string	Absolute path where the optimized PDF was written.

Table 4.2 None entry of the summary dictionary of report.json

Key	Required	Type	Description
num	Yes	integer	Number of pages in the PDF
msg	Yes	string	The literal message: "Page(s) in PDF file."

Table 4.3 GeneralInfo entry of the summary dictionary of report.json

Key	Required	Type	Description
JobStatus	Yes	dict	See Table 4.4. dictionary of job status, job time and APS build information

Table 4.4 JobStatus entry of the summary.GeneralInfo dictionary of report.json

Key	Required	Type	Description
Status	Yes	string	Denotes the status of the job: "Success", Fail" or "Timeout"
Job Processing Time	Yes	float	Number of seconds taken to process the job.
Build Info	Yes	dict	See Table 4.5. Detailed version information for the APS SDK.

Table 4.5 Build Info entry of the summary.GeneralInfo.JobStatus dictionary of report.json

Key	Required	Type	Description
Product	Yes	string	Name of the product, such as "Adobe Print Services"
Version	Yes	string	Build version of the APS SDK, such as "1.0.4000.0"
Release Date	Yes	string	Release date of the APS SDK, such as "Tue Aug 13 16:24:49 2024"
Build Time	Yes	string	Build date of the APS SDK, such as "Tue Aug 13 15:53:51 2024"
Current Time	Yes	string	Date the report.json was generated, such as "Fri Aug 30 2024, 14:26:51"
Platform	Yes	string	OS platform, such as "windows-x64", "linux-x64", or "Unknown"

4.2 Example report.json File

The report output generated after processing the input PDF contains details about what features were applied, how many time it was applied, and so on.

Example 4.2: report.json: Generic Report

```
{
  "summary": {
    "PDFValidation": {
      "num": "1",
      "msg": "Issues in input PDF found",
      "info": [
        {
          "ErrorMsg": "NON-FATAL PDFL Exception: Invalid Type 3 font.",
          "Pages": "1"
        }
      ]
    },
    "PageTreeFix": {
      "num": "0",
      "msg": "Page Tree has been optimized"
    },
    "ReportHintingErrors": {
      "num": "0",
      "msg": "No Issue in fonts"
    },
    "None": {
        "num": "1",
        "msg": "Page(s) in PDF file."
      },
      "GeneralInfo": {
        "JobStatus": {
          "Status": "Success",
          "Job Processing Time": "0.0626871213",
          "Build Info": {
            "Product": "Adobe Print Services",
            "Version": "1.0.4000.0",
            "Release Date": "Tue Aug 13 16:24:49 2024",
            "Build Time": "Tue Aug 13 15:53:51 2024",
            "Current Time": "Fri Aug 30 2024, 14:26:51",
            "Platform": "windows-x64"
          }
        }
      }
    },
  "pdffile": "I:\/APS\/AdobePrintServices-1.x-Resources-1\/
AdobePrintServices_Samples\/APS1.0_EvaluationFiles_GenericSamples\/
FeaturesBenefit\/ValidationAndHinting\/PDF\/
ValidationAndHinting_Generic_Sample1.pdf",
  "outputfile": "I:\/APS\/ValidationAndHinting_Generic_Sample1.optimized.pdf"
}
```

Index

A
ACR ERROR
21 76
ACRSuppressibleErrorHandling 104
ADBE_GRHint 71
Adobe Print Services 35, 56
ApplyBestEffortForFont 104
arrayLenLimit 105
arrayLenThreshold 105
attributes 110
AutoGR analysis 71

B
Building an APS demo application 15
byteWidth 77

C
clip path 69
CosID 68

D
data stream 76
DocInfo 110
Document Level Attributes 110

E
Enhanced Print Services 35, 56

F
FeatureName 68
flate 78
FontInfo 111, 112
fontName 103
Form 70

G
General 111
glyphs 103

H
help 18
hint 71

I
IgnoreAll 104
IgnoreFontHintingErrors 104
Image Cropping 67
Image Upsampling 38
ImageCropper 68
ImageInfo 111
ImageResolution 111
ImageValidation 11, 76, 77
ImageXObject 77
Installation Directory 14
InvalidDHT error 78
invoke flattener 70

J
JPG2K library 78
JPX images 69
JSON Configuration Options 19, 23, 83, 92

M
MaxImageResolution 110
MinImageResolution 110
MinNumPages 79

O
OddOrEvenPages 110
overprint emulation 71

P
page tree repair 105
PageBoundaries 112
PageInfo 110
PageRange 110
PageReferences 111
PageRsrcFix 78
PagesReferences 111
PDEElementPath 68
PDEIndex 68
PDFProperties 108
PDFValidation 104
PDFVTInfo 111
planarization 71
Platform-Specific Requirements for Tachyon 14

R
raster image processor 71
Rasterization 71
ReferenceCount 111
ResultsLimit 110
ResX 111
ResY 111

S
scanlines 76
single Form 70
SpotColorInfo 113
SpotName 113
SubPageRange 110

T
Typefaces 8

V
version 18

X
XObject 76
XObject dictionary 76
XObjectInfo 111