RICOH

RICOH Transforms Suite

PDF2PDFUA Tool User Guide

Version 1.5

Using the PDF2PDFUA Tool 1

Ensuring PDF/UA and WCAG compliance 2

Java Application Programming Interface 3

For information not in this manual, refer to the Help System in your product.

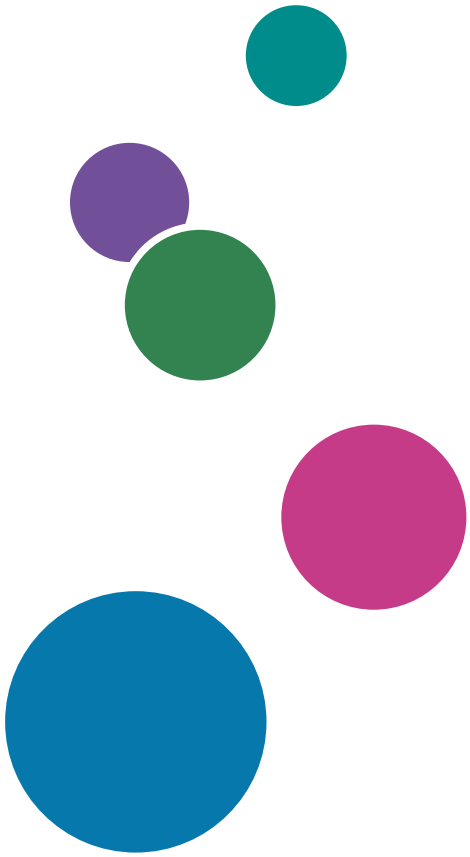


TABLE OF CONTENTS

Introduction

Important ... 2
Cautions regarding this guide ... 2
Publications for this Product ... 2
Symbols ... 2
Abbreviations ... 3
Trademarks ... 3

1 Using the PDF2PDFUA Tool

Using the PDF2PDFUA Tool through the AFP2PDF Plus Transform ... 5
Using the PDF2PDFUA Tool from command line ... 5
Syntax ... 5
Parameters ... 6
Configuring the `PDF2PDFUA.cfg` options file ... 7
Features supported by the PDF2PDFUA Transform ... 8

2 Ensuring PDF/UA and WCAG compliance

Input file requirements for PDF/UA compliance ... 9
Validating the input PDF file and the output PDF UA file ... 9
Input file requirements for WCAG 2.0 (Level AA) compliance ... 9
Input file requirements for WCAG 2.0 (Level AAA) compliance ... 10
Input file requirements for WCAG 2.1 (Level AA) compliance ... 10
Validating PDF/UA or WCAG compliance ... 10

3 Java Application Programming Interface

Classes ... 11
Examples ... 15

Introduction

Important

To the maximum extent permitted by applicable laws, in no event will the manufacturer be liable for any damages whatsoever arising out of failures of this product, losses of documents or data, or the use or non-use of this product and operation manuals provided with it.

Make sure that you always copy or have backups of important documents or data. Documents or data might be erased due to your operational errors or malfunctions of the software. Also, you are responsible for taking protective measures against computer viruses, worms, and other harmful software.

In no event will the manufacturer be responsible for any documents created by you using this product or any results from the data executed by you.

Cautions regarding this guide

- Some illustrations or explanations in this guide could differ from your product due to improvement or change in the product.
- The contents of this document are subject to change without notice.
- No part of this document may be duplicated, replicated, reproduced in any form, modified, or quoted without prior consent of the supplier.

Publications for this Product

This publication provides an overview of the tool, and information about the installation and the use of the PDF2PDFUA tool.

Instruction manuals

For more information about the PDF2PDFUA tool, see these documents:

- *PDF2PDFUA Setup Guide*
- *PDF2PDFUA User's Guide*

Symbols

The following symbols are used in this manual to help you to identify content quickly.



- This symbol indicates points to pay attention to when using the product. Be sure to read these explanations.

Note

- This symbol indicates helpful supplementary information that is not essential to completing a task.

Abbreviations

AES

Advanced Encryption Standard

AFP

Advanced Function Presentation

PDF

Portable Document Format

UA

Universal Accessibility

WCAG

Web Content Accessibility Guideline

Trademarks

Adobe, the Adobe logo, PostScript, and the PostScript logo, PDF, and the PDF logo are either registered trademarks or trademarks of Adobe in the United States and/or other countries.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Sentinel is a registered trademark of Thales DIS CPL USA, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

1. Using the PDF2PDFUA Tool

- **Using the PDF2PDFUA Tool through the AFP2PDF Plus Transform**
- **Using the PDF2PDFUA Tool from command line**
- **Syntax**
- **Parameters**
- **Configuring the `PDF2PDFUA.cfg` options file**
- **Features supported by the PDF2PDFUA Transform**

Using the PDF2PDFUA Tool through the AFP2PDF Plus Transform

To convert AFP files into PDF/UA documents, you must add this option parameter in the AFP2PDF Plus configuration file:

PDFUA_CONTROL_FILE=*`file`*

Specifies the fully qualified name of the PDF2PDFUA tool control file. This file is an XML that contains information on the input file content. The PDF2PDFUA Tool uses the document to mark the content as the PDF/UA standard requires.

PDFUA_CACHE_SIZE=*`number`*

Specifies the number of most used PDFUA control files stored in cache. The number should be greater than 0.

Note

- For an encrypted or a linearized PDF/UA output file set the AFP2PDF Plus option file entries, when running through the AFP2PDF Plus transform:

  For encryption:

  ```
  Default_Encryption_Permissions
  Default_owner_password
  Default_user_passowrd
  ```

  For creating linearized documents:

  ```
  Default_linearization
  ```

  For more information on how to set the options, see the chapter *`AFP2PDF Plus Transform Options File`* from the *`AFP2PDF Plus User's Guide`*.

Limitation:

- When creating PDF/UA files, AES 256-bit encryption is not supported.

Using the PDF2PDFUA Tool from command line

When you run the PDF2PDFUA Tool from command line, the input file can be a PDF document and a control file converted by the PDF2PDFUA Tool into a PDF/UA compliant file.

Syntax

Depending on the operating system, the syntax for the PDF2PDFUA transform is:

pdf2pdfua.*nnn* [**-o** *`output file name`*] [**-i** *`control filename`*] [**-b** [*`number_of_threads`*] *`input directory output directory`*] [**-d** *`output for dumping text`*] [**-linearize** *`true/false`*] [**-encrypt**

owner password user password encryption flags] [**-decrypt** *password*] [**-key** *password*] [**-trace** *regular expression*] [*inputfile*]

where *.nnn* represents the extension **.bat** for Windows and **.sh** for Linux and AIX.

1

Parameters

-b

This parameter is used to run the PDF2PDFUA transform in batch mode. The transform converts all the PDF files from a directory using a single control file that is specified through the **-i** parameter to PDF/UA files.

To specify the number of threads, add the number right after the **-b** flag. Default thread number is 1.

Note

- In batch mode, the *input directory* and *output directory* parameters are mandatory. The *input directory* parameter replaces the *input file* parameter.

-linearize

This parameter is optional.

The implicit value is not specified.

- If the value is true, the output is linearized.
- If the value is false, the output is not linearized, even if the input is linearized.
- If the value is not specified, the output is linearized only if the input is linearized.

-encrypt

This parameter is optional.

By default, the user has no permission. The following flags are used to grant permissions:

`-allowAnnotations`

`-allowAssemble`

`-allowCopy`

`-allowFill`

`-allowModify`

`-allowPrintDegraded`

`-allowPrintHigh`

`-allowPrint`, `-allowPrintDegraded`, or `-allowPrintHigh`.

-decrypt *password*

This parameter removes the original encryption.

-key *password*

This parameter opens locked documents without removing the original encryption. The permissions are preserved.

-trace *regular expression*

This parameter is used to debug control files. All tracing elements are active by default in command line mode.

-validate

This parameter enables the validator. It can be used without a control file when you pre-validate the input PDF. If a control file is specified, it also checks the output PDF.

Configuring the PDF2PDFUA.cfg options file

Additional parameters to control settings for the PDF2PDFUA transform are specified in an options file. The name is PDF2PDFUA.cfg. The PDF2PDFUA.cfg must be located in the root directory of the PDF2PDFUA tool.

The PDF2PDFUA.cfg option parameters are:

AlwaysWriteProducer=TRUE | FALSE

Indicates that the output PDF file must contain information about the PDF2PDFUA transform version and the control file version within the PDF Producer field.

The default value is **False**.

When the value is **False**, the parameter has the same function as it is specified in the control file.

When the value is **True** , the parameter includes the producer meta data regardless of the control file specification. The meta data is useful for development. This parameter is no longer needed because newer versions of PDF2PDFUA include the meta data in non-standard ways. Developers can extract the meta data but a PDF reader will simply ignore it.

DefaultLogSize= integer

The default value is **10**. It represents the maximum size of the default log, in megabytes. This parameter has no effect if **useDefaultLog** is false.

errProgramVersion = true|false

Specifies how the transform handles control files that are designed for newer versions of PDF2PDFUA.

When the value is **true**, it throws an exception, preventing the transform from running incompatible control files.

When the values is **false**, it tries to run incompatible control files generating a warning on the print handler (default STDOUT) and in the logs. This might lead to incorrect output PDFs.

The default value is **true** and it is strongly recommended to keep this option on **true**.

LegacyLinePosition=TRUE | FALSE

The default value is **False**.

When the value is **False**, the position is given by the word with the smallest Y coordinate.

When the value is **True**, the position is given by the word with the biggest Y coordinate. If the line contains subscripts, the getLinePosition operator returns the Y of the subscript.

LegacyNextLine= TRUE | FALSE

The default value is **True**. It affects the behavior of **getNextLine** and **getNextLineInRegion** except for nextLine or nextLineInRegion. These operators always use the new behavior.

Within the legacy behavior, it searches for the first line with y(line) > y(cursor). It does not work correctly if the cursor is set with **getLinePosition**.

Within the new behavior: it searches for the first line with y(line) ≥ y(cursor). It fixes certain issues but introduces another issue:

```
if y(line) = y(cursor)
```

For example:

```
<Cursor> <Y> <Expr> getLinePosition(getNextLine) </Expr> </Y> </Cursor>
<Expr> $line1 = getNextLine </Expr>
<Expr> $line2 = getCurrentLine </Expr>
```

Here, line1 and line2 are the same.

UseDefaultLog= TRUE | FALSE

The default value is **True**.

When the value is **True**, it creates a default log file if none was passed from the API. It is useful in standalone mode.

When the value is **False**, the logging is enabled only if configured from the API.

Features supported by the PDF2PDFUA Transform

With the PDF2PDFUA Plus transform you can batch convert several input files that have a common generation template. You can create a control file for any PDF file, including invoices, account summaries, or credit card statements. You have the possibility of creating one control file for each invoice type.

PDF2PDFUA Plus provides enhanced support for handling variable data. PDF2PDFUA Plus can handle variable data at fixed positions such as names, address boxes, but also tables with a variable number of rows or tables that span on multiple pages. The PDF2PDFUA transform allows you to mark only the real content and artifact the irrelevant content.

Note

Contact your Ricoh representative to receive more information regarding the procedure of creating a control file.

2. Ensuring PDF/UA and WCAG compliance

- **Input file requirements for PDF/UA compliance**
- **Validating the input PDF file and the output PDF UA file**
- **Input file requirements for WCAG 2.0 (Level AA) compliance**
- **Input file requirements for WCAG 2.0 (Level AAA) compliance**
- **Input file requirements for WCAG 2.1 (Level AA) compliance**
- **Validating PDF/UA or WCAG compliance**

Input file requirements for PDF/UA compliance

If the input file contains one of these problems, the file cannot be processed or the resulting file is not PDF/UA complaint.

- PDF file contains flickering content or PDF elements that lead to flickering (Matterhorn Protocol – Ch03-X).
- Information within the PDF file is conveyed using contrast, color, or format layout (MP Ch04).
- PDF file contains sound (MP Ch05).
- PDF file uses character codes without Unicode mapping (MP Ch10).
- PDF file contains images with paragraphs of text (MP C13-06).
- PDF file uses non-embedded fonts (MP C21).

  Note

  If used together with the AFP2PDF Plus transform, make sure that it is configured to embed the fonts.
- PDF file contains forms.
- PDF file contains action elements.

Validating the input PDF file and the output PDF UA file

The PDF2PDFUA validator generates warnings that can refer to specific indexes from the PDF associations Matterhorn Protocol 1.1. The pre-validator checks some of the conditions mentioned in the Matterhorn Protocol.

To find more information about the displayed errors, search for the error ID in the Marterhorn protocol document.

Contact your PDF producer to investigate and solve this PDF error to obtain a PDF/UA file compatible output.

Input file requirements for WCAG 2.0 (Level AA) compliance

To make sure that the input file is compliant with the accessibility standards of the Web Content Accessibility Guidelines (WCAG) 2.0 Level AA, check the following:

WCAG 1.4.3: Minimum contrast

The input file should contain text and images with a minimum contrast of 4.5:1.

WCAG 1.4.5: Images as text

The input file should deliver information through text, rather than images containing text.

Input file requirements for WCAG 2.0 (Level AAA) compliance

To make sure that the input file is compliant with the accessibility standards of the Web Content Accessibility Guidelines (WCAG) 2.0 Level AA, check the following:

WCAG 1.4.6 – Enhanced Contrast requirement

The visual presentation of text and images of text have a contrast ratio of at least 7:1.

WCAG 1.4.8 – Visual presentation

The input file should meet these requirements:

1. Width is no more than 80 characters or glyphs (40 if CJK).
2. Text is not justified (aligned to both the left and the right margins).
3. Line spacing (leading) is at least space-and-a-half within paragraphs, and paragraph spacing is at least 1.5 times larger than the line spacing.

WCAG 1.4.9 Images of Text (No Exception)

Images of text are only used for pure decoration or where a particular presentation of text is essential to the information that is being conveyed.

Input file requirements for WCAG 2.1 (Level AA) compliance

1.4.11 Non-text Contrast (Level AA)

The visual presentation of graphical objects have a contrast ratio of at least 3:1 against adjacent colors.

Validating PDF/UA or WCAG compliance

You can use these tools to make sure that your PDF file is PDF/UA or WCAG compliant:

PAC – PDF Accessibility Checker

A free-to-download tool that verifies if the PDF file is PDF/UA or WCAG compliant. If the verification fails you can view any issues that lead to the file not being compliant. You can get it online from the PDF/UA Foundation: https://pac.pdf-accessibility.org/en.

3. Java Application Programming Interface

- **Classes**

This chapter describes the Java API provided by RICOH PDF2PDFUA.

The code surrounded by angle brackets refers to an XML element in the control file. For example, `<If>` refers to conditionals in the control file.

Classes

These classes are part of the Java application programming interface (API) for RICOH PDF2PDFUA.

enum EncryptionAlgorithm

Package: **com.ricoh.pdf2pdfua.api**

Option that specifies the encryption algorithm of the output PDF.

Values:

none

The output is not encrypted.

preserve

Preserves the encryption from the input PDF.

Standard128

An RC4 128 bit key-size stream cipher.

AES128

Advanced encryption standard that uses a 128-bit key length.

AES256

Advanced encryption standard that uses a 256-bit key length.

interface IControlFile

Package: **com.ricoh.pdf2pdfua.api**.

Represents the syntax tree of the control file. The control file is parsed once and applied on multiple PDF files.

```
IControlFile refresh() throws PreprocessingException;
```

Used to scan the control file for changes.

Returns `this` if the control file is up to date.

Returns a new `IControlFile` if the underlying files are no longer in-sync.

interface Pdfua

Package: **com.ricoh.pdf2pdfua.api**

The Java service provider interface (SPI) for RICOH PDF2PDFUA.

```
IControlFile newControlFile(String path);
```

Parses the control file specified by path. A single control file instance can be reused across multiple threads.

```
PdfUaOpt newOpt();
```

Creates an instance of PdfUaOpt with default configurations.

```
void run(FileInputStream fis, File out, IControlFile cf, PdfUaOpt opt)
throws PDFUAException;
```

This method runs the PDF2PDFUA conversion. It is thread-safe and reentrant.

```
byte[] run(InputStream inPdf, IControlFile cf, PdfUaOpt opt) throws PDFUAException;
```

This method runs the PDF2PDFUA conversion and returns a byte array containing the tagged PDF. It is thread-safe and reentrant.

class PDFUAException extends Exception

Package: **com.ricoh.pdf2pdfua.api**

Base class for PDF2PDFUA conversion errors.

class PdfUaLoader

Package: **com.ricoh.pdf2pdfua.api**

Trivial helper that loads the PDF2PDFUA service implementation. The service can be loaded directly with a ServiceLoader.

```
public static Pdfua load();
```

Loads the PDF2PDFUA service implementation.

abstract class PdfUaOpt

Package: **com.ricoh.pdf2pdfua.api**

Used to configure the behavior of PDF2PDFUA. It is a configuration container that has only getters and setters.

This is an abstract class, but it must never be extended by API users. To get an instance, use `Pdfua.newOpt()`.

A single instance can be used by multiple threads but should not be modified while PDF2PDFUA is running.

```
public boolean getAllowAnnotations();
public void setAllowAnnotations(boolean allowAnnotation);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowAssemble();
public void setAllowAssemble(boolean allowAssemble);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowCopy();
public void setAllowCopy(boolean allowCopy);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowFill();
public void setAllowFill(boolean allowFill);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowModify();
public void setAllowModify(boolean allowModify);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowPrint();
public void setAllowPrint(boolean allowPrint);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public boolean getAllowPrintDegraded();
public void setAllowPrintDegraded(boolean allowPrintDegraded);
```

Permission flag for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: false

```
public byte[] getDecryptionKey();
public void setDecryptionKey(byte[] decryptionKey);
```

This option is required if the input PDF is encrypted. Used to unlock the input without removing the encryption. The owner key is preferable.

Default value: null

```
public EncryptionAlgorithm getEncryption();
public void setEncryption(EncryptionAlgorithm encryption);
```

Specifies the encryption algorithm of the output.

Default value: EncryptionAlgorithm.preserve

```
public Boolean getLinearize();
public void setLinearize(Boolean linearize);
```

If linearize is true: the output PDF is linearized.

If linearize is false: the output PDF is non-linearized.

If linearize is null: the output is linearized only if the input PDF was linearized.

Default value: null

```
public Logger getLog();
public void setLog(Logger log);
```

Default value: specified in the configuration file.

```
public boolean getOnlyMergeXObjects();
public void setOnlyMergeXObjects(boolean onlyMergeXObjects);
```

This option is used for development and testing. It must not be used in a production environment.

Default value: false

```
public byte[] getOwnerPassword();
public void setOwnerPassword(byte[] ownerPassword);
```

The owner password for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: null

```
public PdfUaOutputHandler getStderrHandler();
public void setStderrHandler(PdfUaOutputHandler stderrHandler);
```

Used to handle the messages that are normally written to stderr. Default value: built-in handler as specified by the configuration file.

```
public PdfUaOutputHandler getStdoutHandler();
public void setStdoutHandler(PdfUaOutputHandler stdoutHandler);
```

Used to handle the messages that are normally written to stdout.

Default value: built-in handler as specified by the configuration file.

```
public PdfUaOutputHandler getTraceHandler();
public void setTraceHandler(PdfUaOutputHandler traceHandler);
```

Used only by the `<Trace>` command.

Default value: built-in handler as specified by the configuration file.

```
public boolean getUseLatestRevisions();
public void setUseLatestRevisions(boolean useLatestRevisions);
```

This option is used for development and testing. It must not be used in a production environment.

Default value: `false`

```
public byte[] getUserPassword();
public void setUserPassword(byte[] userPassword);
```

The user password for PDF encryption. Has no effect if the output is not encrypted by PDF2PDFUA.

Default value: null

interface PdfUaOutputHandler

Package: **com.ricoh.pdf2pdfua.api**

Handler for IO redirection.

```
void onMessage(Node src, int pageNo, String msg);
```

`src`: the XML node of the invoking control file rule. Null if the message cannot be traced to an XML node.

`pageNo`: the page number starts from 1. This value is zero if the message cannot be traced to a page.

`msg`: can contain multiple lines of text.

class PreprocessingException extends Exception

Package: **com.ricoh.pdf2pdfua.api**

Base class for errors that occur while the control file is preprocessed.

class UaStackTrace extends ArrayList<String>

Package: **com.ricoh.pdf2pdfua.api**

A stack trace that points to an XML element in the control file.

```
public static UaStackTrace unwind(final Node top);
```


Examples

This example shows how to parse a control file and how to apply it on multiple documents. All documents in a batch are supposed to be similar and compatible with the control file. For performance reasons, the control file is parsed a single time.

The software is expected to run on servers with multiple CPU cores. This example is multi-threaded because it is the standard use case.

```
package com.ricoh.pdf2pdfua.api.example;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ricoh.pdf2pdfua.api.IControlFile;
import com.ricoh.pdf2pdfua.api.PDFUAException;
import com.ricoh.pdf2pdfua.api.PdfUaLoader;
import com.ricoh.pdf2pdfua.api.PdfUaOpt;
import com.ricoh.pdf2pdfua.api.Pdfua;
import com.ricoh.pdf2pdfua.api.preprocessor.PreprocessingException;

public class Pdf2PdfUaApiExample {
    static class ThreadSafeIterator {
        Iterator<File> i;

        ThreadSafeIterator(List<File> files) {
            i = files.iterator();
        }

        synchronized File next() {
            if (i.hasNext())
                return i.next();
            return null;
        }
    }

    static void exit(String msg) {
```

```
        System.err.println(msg);
        System.exit(1);
    }

    static File changeExtension(File f) {
        return new File(f.getAbsolutePath().replace(".pdf", ".ua.pdf"));
    }

    static Thread[] createThreadArray() {
        return new Thread[Runtime.getRuntime().availableProcessors()];
    }

    static void convertBatch(String controlFilePath, List<File> pdfs)
            throws PreprocessingException, InterruptedException {
        final Pdfua pdfuaInstance = PdfUaLoader.load();
        final IControlFile cf = pdfuaInstance.newControlFile(controlFilePath);
        final PdfUaOpt opt = pdfuaInstance.newOpt();
        final ThreadSafeIterator it = new ThreadSafeIterator(pdfs);
        Thread[] threads = createThreadArray();
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread() {
                public void run() {
                    File pdfIn;
                    while ((pdfIn = it.next()) != null) {
                        try {
                            FileInputStream fis = new FileInputStream(pdfIn);
                            File out = changeExtension(pdfIn);
                            pdfuaInstance.run(fis, out, cf, opt);
                        } catch (FileNotFoundException | PDFUAException e) {
                            e.printStackTrace();
                        }
                    }
                }
            };
        }
        for (Thread t : threads)
            t.start();
        for (Thread t : threads)
            t.join();
        System.out.println("Batch complete!");
    }

    public static void main(String[] args)
            throws PreprocessingException, InterruptedException {
        if (args.length < 1)
            exit("No control file!");
        ArrayList<File> files = new ArrayList<>();
        for (int i = 1; i < args.length; i++)
            files.add(new File(args[i]));
        if (files.isEmpty())
            exit("No PDF file to convert!");
        convertBatch(args[0], files);
    }
}
```

This example illustrates how PDF2PDFUA can work without using files. This is useful when PDF2PDFUA is integrated in other systems such as web applications.

```
package com.ricoh.pdf2pdfua.api.example;

import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
import com.ricoh.pdf2pdfua.api.PdfUaLoader;
import com.ricoh.pdf2pdfua.api.Pdfua;

public class Pdf2PdfUaApiExampleBytesOutput {
    public static void main(String[] args) throws Exception {
        Pdfua ua = PdfUaLoader.load();
        byte[] outputBytes = ua.run(
            new FileInputStream("./untagged.pdf"),
            ua.newControlFile("./control-file.xml"),
            ua.newOpt()
        );
        var os = new FileOutputStream("./out.pdf");
        os.write(outputBytes);
        os.close();
    }
}
```

RICOH Transforms Suite PDF2PDFUA Tool User Guide

EN US EN GB EN AU 10/2026 G550-30700-00