RFID Tag Command Generator

Version 1.0

User's Guide

December 2017 **www.lexmark.com**

Contents

Overview

Use the utility to generate Radio Frequency Identification (RFID) tag commands and insert them into your documents for printing on Lexmark RFID printers.

The utility is used to do the following:

- Insert a tag command into a PDF document.
- Generate a tag command for PCL® data stream or Lexmark™ Forms Composer.
- Generate a parameter file to automate the PDF or PCL insertion process.

RFID tags are microchips that are attached to an object and communicated with wirelessly using RFID technology. Users can print and program RFID media containing RFID tags using a Lexmark printer configured with the RFID option.

The RFID option is a tray with an RFID encoder that converts your Lexmark printers into an RFID printer. It supports RFID tags conforming to the Electronic Product Code (EPC) Gen2 (ISO 18000-6C) for ultra high frequency (UHF) RFID tags. You can program tags from 16 to 496 bits of EPC memory in 16-bit increments and from 2 to 64 bytes of user memory.

Embedding the tag commands within the document structure controls the programming of an RFID tag. Each tag command represents a single tag, and provides information for the printer to program the tag. Tag commands are generated by the RFID Tag Command Generator, or are predefined values from previously released applications provided by Lexmark or by your RFID system integrator.

This document provides instructions on how to run and use the utility.

Deployment readiness checklist

Make sure that:

- ☐ The utility is available.
  Note: To download the utility, go to http://support.lexmark.com.

- ☐ Version 1.8 or later of either of the following Java environments is available on your computer:
  - Java Runtime Environment (JRE)
  - Java Development Environment (JDE)

Running the utility

1. From your computer, unzip RFIDTagGen.zip.

2. From the RFIDTagGen/bin directory, run the utility.
   - For Windows® operating systems, use the RFIDTagGen batch file.
   - For Mac OS operating systems or Unix operating systems, use the RFIDTagGen executable file.

Using the utility

The RFID Tag Command Generator controls the following tag command attributes:

- The memory bank to use
- The data written on to the tag
- The position of the tag on the page
- The read-and-write power levels to use when programming the tag

Insert the RFID data using any of the following methods:

- JSON parameters file
- JSON parameters on the command line
- Command-line options
- Predefined tag commands

You can also use the RFID Tag Command Generator to translate command-line parameters into a JSON file.

Generating tag commands for PDF

Automate the insertion of tag commands into a PDF. The RFID tag Command Generator controls the following page or document attributes for PDF:

- Two-sided printing
- The page number for the tag
- Color or black-and-white printing
- The source tray for the page containing the tag

You can run the utility as a standalone Java command-line operation. It can also be called multiple times on the same PDF. Parameter sets are added or modified based on the supplied parameters and the contents of the PDF.

Using JSON parameters

The RFID Tag Command Generator inserts the RFID tags using a JSON array in a file to specify the parameters. The RFID data is either included in the JSON file or is added on the command line using the `-d` option.

We recommend using this method when the tag parameters and the page attributes do not change from page to page or from document to document. For more information, see “JSON page attributes and tag parameters” on page 11.

Example 1: Modify input.pdf and write to output.pdf specifying parameters in parameters.json and data on the command line

```
RFIDTagGen -i input.pdf –o output.pdf –jf parameters.json –d
RFID_Tagdata
```

Where:

- The parameters are in the JSON file.
- The data is specified on the command line.

Example 2: Overwrite input.pdf specifying both parameters and data in parameters.json

```
RFIDTagGen –w –i input.pdf –jf parameters.json
```

Where:

- The parameters and data are in the JSON file.
- Input.pdf is overwritten with the changes.

Example 3: Specify parameters on the command line using JSON format

```
RFIDTagGen –w –i input.pdf –jc '{"Page":1,
"RfidTag":{"Bank":"EPC"}}' –d RFID_Tagdata
```

Where:

- The parameters are in the JSON file.
- The parameters and data are specified on the command line.
- Multiple pages are included as a JSON array.

Notes:

- To enclose the JSON object, use your command-line shell convention for quoting parameters that contain quotation marks, spaces, and special characters.
- If you are copying this example, then change the single and double quotation marks to ASCII characters (0x27 and 0x22 respectively). Font selection in rich-text documents may cause the code to fail.

Using command-line options

For more information on the options and their values, see "Command-line syntax" on page 10.

Example 1: Overwrite input.pdf and specify X and Y position using -xpos and -ypos options

```
RFIDTagGen –w –i input.pdf –xpos 254 –ypos 108 –b EPC –d
RFID_Tagdata
```

Where:

- The parameters and data are specified using command-line options.
- Input.pdf is overwritten with the changes.

Using predefined tag commands

We recommend using this method only for previously released applications.

To use a predefined tag command directly, use the `–tc` option.

Example 1: Overwrite input.pdf using a predefined tag command

```
RFIDTagGen –w –i input.pdf –tc A00005700206960RFID_Tagdata
```

Where:

- The predefined tag command is specified in the command line.
- Input.pdf is overwritten with the changes.

Generating tag commands for PCL data stream or Lexmark Forms Composer

To generate the tag command for Lexmark Forms Composer and PCL documents, provide the required parameters in JSON or as a command-line option, and then use the `-tc` option. The utility processes the parameters and returns a formatted tag command or tag command portion.

Example 1: Generate a tag command that includes X and Y positions without including the data portion

Note: This example is commonly used for Forms Composer usage.

```
RFIDTagGen -tc –xpos 254 –ypos 208 –b EPC -l 96
```

Where:

- The return tag command is `A0800FE006C005700206960`.
- The parameters are specified on the command line.
- The tag command string is printed to the standard output.

Example 2: Generate a tag command from a JSON parameter file including data

Note: This example is commonly used for PCL usage.

```
RFIDTagGen -tc -jf parameters.json -d RFID_Tagdata
```

Where:

- The return tag command is `A0800FE006C005700206960RFID_Tagdata`.
- The parameters are specified in JSON file.
- The tag command string is printed to the standard output.

Auxiliary functions

Converting data to hexadecimal or ASCII

Make sure that the data is compatible with the handheld reader display capabilities. If the data is originally in ASCII and the handheld reader shows the data only in hexadecimal, then it becomes difficult to read and validate.

To convert a hexadecimal string to its ASCII equivalent, use the `-H2A` option.
To convert an ASCII string to its hexadecimal equivalent, use the `-A2H` option.

Example 1: Convert a hexadecimal string into its ASCII equivalent

```
RFIDTagGen -H2A 524649445F54616764617461
```

Where:

- The return value is `RFID_Tagdata`.
- The input string is the hexadecimal value corresponding to ASCII data.
- The output string may include non-printable characters.

Example 2: Convert an ASCII string into its hexadecimal equivalent

```
RFIDTagGen -A2H RFID_Tagdata
```

Where:

- The return value is `524649445F54616764617461`.
- The input string is the ASCII value corresponding to hexadecimal data.
- The output string may contain characters from `0` to `9` and from A to F.

References

Command-line syntax

RFIDTagGen options and their descriptions

Option	Description
`-?`	A help message is generated.
`-h`	
`-a <address>`	The address where to write the user memory. The default value is `0`.
`-A2H <ASCII>`	ASCII data is converted to its hexadecimal equivalent.
`-H2A <hexadecimal>`	Hexadecimal data is converted to its ASCII equivalent.
`-b <bank>`	The memory bank to use where `<bank>` can be any of the following values: • EPC—Standard 96-bit EPC • XPC—Extended EPC • USER—User memory The default value is EPC.
`-d <data>`	The tag data is inserted into the file. The default value is in ASCII format.
`-gen_json`	A JSON file is generated from the command-line parameters.
`-gen_tc`	A tag command is generated from provided parameters.
`-d <data> –hex`	The data provided with the -d option or the JSON file in hexadecimal format.
`-i <file name>`	The input file. The default value is `stdin`.
`-o <file name>`	The output file. The default value is `stdout`.
`-jc <JSON>`	The JSON on command line.
`-jf <file name>`	The JSON in parameter file.
`-l <length>`	The length of the data in bits for EPC or in bytes for user memory. The default value is the calculated length of the provided data.

RFIDTagGen options and their descriptions

Option	Description
`-page <page>`	The page number for insertion. The default value is 1.
`-rp <value>`	The read power ranging from 0 to 30dBm. The default value is 25.
`-w`	Overwrites the input.file with the changes.
`-wp <value>`	The write power ranging from 0 to 30dBm. The default value is 25.
`-source <value>`	The source tray. The default value is `Use printer default`.
`-tc <tag command>`	A predefined tag command is inserted into a PDF document.
`-xpos <value>`	The X position ranging from 0 to 355 millimeters (mm). The default value is 25.
`-ypos <value>`	The Y position ranging from 0 to 216mm. The default value is 108.
`-v`	The version information is printed.

JSON page attributes and tag parameters

Specify page attributes and tag parameters as a JSON array. The array contains one JSON object for each page that specifies the attributes for the page. An RFID tag is represented by an RFIDTag attribute in the page object. The RFIDTag object value is a JSON object that contains all the parameters for the tag.

Example 1: JSON for single pages with one tag

```
{ “PageNum”:1,
  “Source”:3,
  “RfidTag”:
   {
   “RfidData”:“RFID_Tagdata”,
   “Bank”:“EPC”,
   “X”:25,
   “Y”:108
   }
}
```

Where:

- The parameters are specified in the JSON file.
- The data in the JSON file is overwritten with the data specified on the command line.
- Spacing and line breaks are added for readability.

Example 2: Specify as a JSON array containing one JSON object (optional)

```
[
{ “PageNum”:1,
  “Source”:3,
  “RfidTag”:
  {
  “RfidData”:“RFID_Tagdata”,
  “Bank”:“EPC”,
  “X”:25,
  “Y”:108
  }
}
]
```

Example 3: JSON for multiple pages with one tag for every page

```
[
{ “PageNum”:1,
  “Source”:3,
  “RfidTag”:
  {
  “RfidData”:“RFID_Tagdata”,
  “Bank”:“EPC”,
  “X”:25,
  “Y”:108
  }
},
{ “PageNum”:2,
  “Source”:3,
  “RfidTag”:
  {
  “RfidData”:“MoreRfidData”,
  “Bank”:“EPC”,
  “X”:25,
  “Y”:108
  }
}
]
```

Where:

- The parameters are specified in the JSON file.
- The data in the JSON file is overwritten with the data specified on the command line.
- Spacing and line breaks are added for readability.

Supported page attributes

Attribute	Description	Type	Values
`PageNum`	The page number	String	Any number up to *N*, Where: *N* is the last page of the document.
`Source`	The source tray	Number	Any tray value such as `Tray 1` or `Tray 2`.
`Duplex`	The two-sided printing mode	String	• `Simplex` • `Long Edge` • `Short Edge`
`PrintMode`	Color or black-and-white print	String	• `Black and White` • `Color`
`RFIDTag`	The RFID tag parameters	JSON object	For more information, see "RFIDTag Objects and their parameters" on page 13.

Supported parameters of the RFIDTag object within the page object

When using a JSON parameter file, you can include page attributes and tag parameters for all the pages and RFID tags in one JSON file, and call `RFIDTagGen` one time for every document.

Parameters	Description	Type	Values
`Bank`	The memory bank	String	• EPC • XPC • USER
`Address`	The user memory address	Number	`0–65535`
`ReadPower`	The read power in decibel-milliwatts (dBm)	Number	`0–30`
`WritePower`	The write power in decibel - milliwatts	Number	`0–30`
`X`	The X position in millimeter (mm)	Number	`0–355`
`Y`	The Y position in millimeter	Number	`0–216`
`RFIDdata`	The RFID data	String	N/A

More examples

Example 1: Overwrite a PDF using default values for all parameters

```
RFIDTagGen -w –i input.pdf -d RFID_TagData
```

Where:

- The default parameters are used.
- Input.pdf is overwritten with the changes.

Example 2: Write a PDF to a separate file using default values for all parameters except data

```
RFIDTagGen –i input.pdf -o output.pdf -d RFID_TagData
```

Where:

- The default parameters are used.
- The data is specified on the command line.
- The changes are written to output.pdf.

Example 3: Write a 4-byte string to address 0x20 user memory, with the tag positioned at X=25mm Y=108mm, and a write power setting of 22dBm on the third page of the document. Pick media from tray 3.

```
RFIDTagGen -i input.pdf -o output.pdf -d ABCD -b USER -l 4 -a 20 -wp
22 –xpos 25 -ypos 108 -page 3 -source 3
```

Where:

- The parameters and data are specified on the command line.
- The change is written to output.pdf.

Example 4: Write an RFID tag to both pages 2 and 3 of a document.

```
RFIDTagGen -w -i input.pdf -d RFID_TagData -b EPC -xpos 25 -ypos 108
-page 2 -source 3
RFIDTagGen -w -i input.pdf -d More_TagData -b EPC -xpos 25 -ypos 108
-page 3 -source 3
```

Where:

- The parameters and data are specified on the command line.
- The `RFIDTagGen` is called one time for each tag that is added.
- Input.pdf is overwritten with the changes.

Example 5: Overwrite a PDF using parameters and data specified in a JSON file.

Use any of the following codes:

- `RFIDTagGen -w -i input.pdf -jf parameters.json`
- `RFIDTagGen -i input.pdf -o output.pdf -jf parameters.json`
- `RFIDTagGen -w -i input.pdf -jf parameters.json -d RFID_Tagdata`

Where:

- The parameters are specified in the JSON file.
- The data in the JSON file is overridden with the data specified on the command line.
- Input.pdf is overwritten with the changes.

Notices

December 2017

The following paragraph does not apply to any country where such provisions are inconsistent with local law: LEXMARK INTERNATIONAL, INC., PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in later editions. Improvements or changes in the products or the programs described may be made at any time.

References in this publication to products, programs, or services do not imply that the manufacturer intends to make these available in all countries in which it operates. Any reference to a product, program, or service is not intended to state or imply that only that product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any existing intellectual property right may be used instead. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by the manufacturer, are the user's responsibility.

For Lexmark technical support, visit http://support.lexmark.com.

For information on supplies and downloads, visit www.lexmark.com.

GOVERNMENT END USERS

The Software Program and any related documentation are "Commercial Items," as that term is defined in 48 C.F.R. 2.101, "Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. 12.212 or 48 C.F.R. 227.7202, as applicable. Consistent with 48 C.F.R. 12.212 or 48 C.F.R. 227.7202-1 through 227.7207-4, as applicable, the Commercial Computer Software and Commercial Software Documentation are licensed to the U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein.

Trademarks

Lexmark and the Lexmark logo are trademarks or registered trademarks of Lexmark International, Inc. in the United States and/or other countries.

PCL® is a registered trademark of the Hewlett-Packard Company. PCL is Hewlett-Packard Company's designation of a set of printer commands (language) and functions included in its printer products. This printer is intended to be compatible with the PCL language. This means the

printer recognizes PCL commands used in various application programs, and that the printer emulates the functions corresponding to the commands.

Java is a registered trademark of Oracle and/or its affiliates.

Windows is either a registered trademark or trademark of the Microsoft group of companies in the United States and other countries.

Mac OS is a registered trademark of Apple Inc.

All other trademarks are the property of their respective owners.