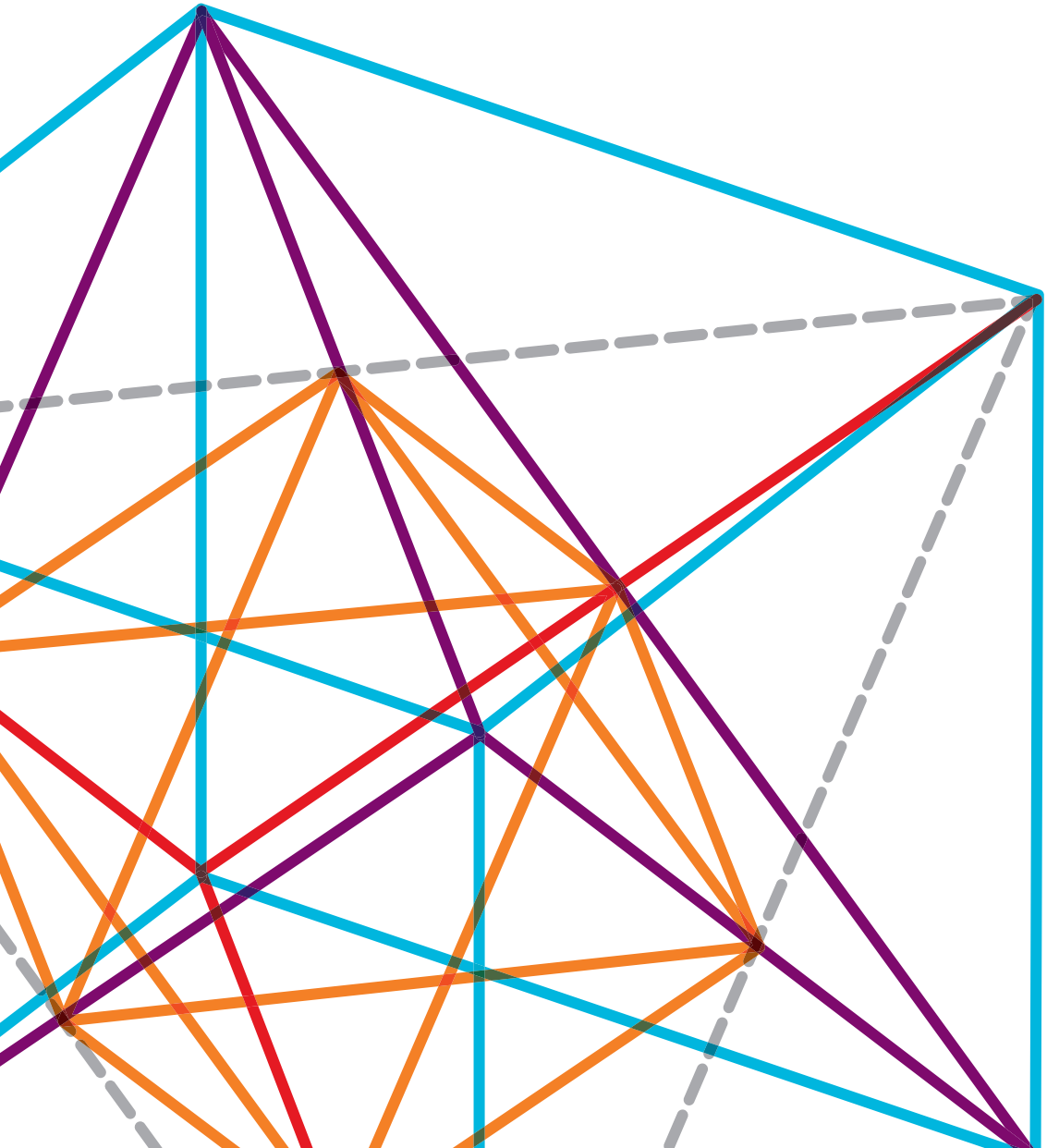
Active Near Field Communication
AppNote 2456935.590150
October 5, 2014

ZEBRA TECHNOLOGIES

SEE MORE. DO MORE.



INTRODUCTION

Near Field Communication (NFC) is a set of standards for short-range radio communication for data exchange. There are two types of NFC technology in use: Passive NFC devices and Active NFC devices. Passive NFC devices are tags with simple electronics that are powered by the NFC reader, which can be read from and written to as a small storage device but cannot perform any other tasks.

An Active NFC device can communicate information freely back and forth, exchanging data without limitations. Active NFC devices can also write to and read from Passive NFC devices. At least one Active NFC device is required for NFC to operate with another Active or Passive NFC device.

While all ZQ500™ series printers have a Passive tag (Print Touch) containing manufacturing information on the side of the unit, some models of ZQ500 series printers include a printer operating system controlled emulated Passive NFC functionality. The ZQ500 series implementation includes functionality as an emulated Passive NFC device, in which Active NFC devices can read from and write to the printer as if it were a Passive (non-powered) NFC device.

NFC SETGETDO COMMANDS

The Active NFC feature can be controlled using the following SetGetDo (SGD) commands.

device.feature.nfc

Purpose: Indicates if an Active NFC reader is present on the printer
Range: “not available”, “not present”, or “present”
Default: N/A
Example: `! U1 getvar "device.feature.nfc"`

NOTE: The command must be followed by a carriage return/line feed or a space character. This command will return "`not present`" if the feature is unavailable, or "`present`" if it is available. Printers that do not support Active NFC will respond with "`not available`" .

nfc.revision

Purpose: Provides information about the revision of the NFC device
Range: returns the revision number of the NFC device
Default: N/A
Example: `! U1 getvar "nfc.revision"`

NOTE: The command must be followed by a carriage return/line feed or a space character. This command causes the printer to return information about the revision of the Active NFC reader installed. If no reader is present, this will be blank. This is a `getvar` only command.

nfc.model

Purpose: Provides information about the model of the NFC device
Range: returns the model of the Active NFC device
Default: N/A
Example: `! U1 getvar "nfc.model"`

NOTE: The command must be followed by a carriage return/line feed or a space character.

nfc.status

Purpose: Causes the printer to return the current status of the Active NFC reader
Range: idle, in progress, timed out, successful
Default: idle
Example: `! U1 getvar "nfc.status"`

NOTE: The command must be followed by a carriage return/line feed or a space character. This command reports on the current status of the reader or status of last operation completed. If the reader is inactive or hasn't been used, this command returns "idle". If an operation is pending, the command returns "in progress". If an operation has been completed successfully, it returns "successful". If the operation has timed out, "timed out" will be returned.

nfc.timeout

Purpose: Set the timeout for the current read or write operation (in seconds)
Range: 0 to 3600, setting to 0 disables the timeout feature
Default: 10
Example: `! U1 setvar "nfc.timeout" "20"`

NOTE: The command must be followed by a carriage return/line feed or a space character.

nfc.read_content.text

Purpose: Returns content stored in the last Active NFC read event
Range: max of 2048 characters
Default: ""
Example: `! U1 setvar "nfc.read_content.text" "value"`

NOTE: The command must be followed by a carriage return/line feed or a space character.

nfc.write_content.text

Purpose: Sets the content to be transmitted by the Active NFC system
Range: max of 2048 characters
Default: ""
Example: `! U1 setvar "nfc.write_content.text" "value"`

NOTE: The command must be followed by a carriage return/line feed or a space character. This command is how the data is read or the data to write is set. These values must be sent in base-64 encoded binary. There is a character limit imposed on this field, based on the type of tag. Only the text field of the tag can be read from or written to at this time. After reading or writing is complete, these commands can both be reset to empty if sensitive information is being read or written. The read or write content must be set before the mode is configured.

nfc.mode

Purpose: Set the Active NFC reader mode
Range: off, read, write
Default: off
Example: `! U1 setvar "nfc.mode" "value"`

NOTE: The command must be followed by a carriage return/line feed or a space character. This commnd starts or stops a read or write operation. At power on, this is set to "off". It returns to "off" after a read or write operation completes, fails or times out. To begin an operation, set this value to the desired "read" or "write".

EXAMPLES

Reading from an active or passive tag from another device

To read from a passive or active tag on a ZQ500 series printer, issue the following series of commands. This example has a time out of 20 seconds.

NOTE: The commands must be followed by a carriage return/line feed or a space character.

```
! U1 SETVAR "nfc.timeout" "20"
! U1 SETVAR "nfc.mode" "read"
```

Next, enter a loop where you wait for the following command to return. Initially this command will return "`in progress`". At this time, the printer is ready to read. Bump the device or tag to be read now. The SGD will change to "timeout," "successful," or an error message once the operation completes.

```
! U1 GETVAR "nfc.status"
```

Assuming the operation succeeds, issue the following getvar to get the result.

```
! U1 GETVAR "nfc.read_content.text"
```

The value this field returns will be in base-64 encoded binary and must be decoded by the software to get the data.

Writing to an active or passive tag on another device

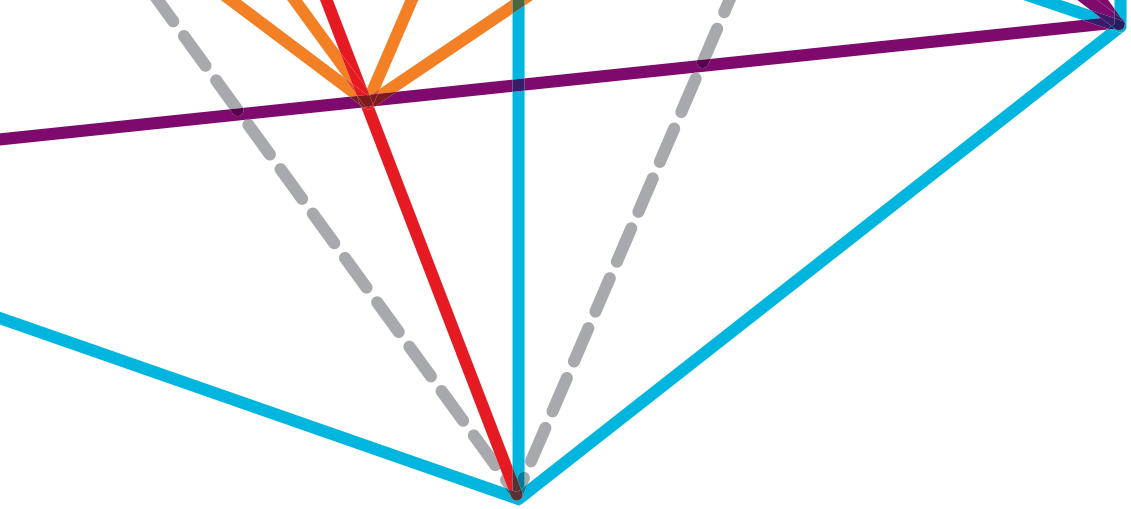
To write to a passive or active tag on a ZQ500 series printer, issue the following series of commands. This example has a time out of 20 seconds, and sets up the text “Hello World!” to be written. This text must be base-64 encoded.

```
! U1 SETVAR "nfc.timeout" "20"
! U1 SETVAR "nfc.write_content.text" "SGVsbG8gV29ybGQh"
! U1 SETVAR "nfc.mode" "write"
```

Next, enter a loop where you wait for the following command to return. At this time, the printer is ready to write. Bump the device or tag to be written to now. Initially this SGD will return “in progress”, but it will change to either “timeout,” “successful,” or an error message once the operation completes.

```
! U1 GETVAR "nfc.status"
```

If the command returns "`successful`," the write was successful.



Corporate Headquarters
+1 800 423 0442
inquiry4@zebra.com

Asia-Pacific Headquarters
+65 6858 0722
apacchannelmarketing@zebra.com

EMEA Headquarters
+44 (0)1628 556000
mseurope@zebra.com

Latin America Headquarters
+1 847 955 2283
inquiry4@zebra.com

Other Locations / USA: California, Georgia, Illinois, Rhode Island, Texas, Wisconsin **Europe:** France, Germany, Italy, the Netherlands, Poland, Spain, Sweden, Turkey, United Kingdom **Asia Pacific:** Australia, China, Hong Kong, India, Indonesia, Japan, Malaysia, Philippines, Singapore, South Korea, Taiwan, Thailand, Vietnam **Latin America:** Argentina, Brazil, Colombia, Florida (LA Headquarters in USA), Mexico **Africa/Middle East:** Dubai, South Africa

(12/14)