Technology
Solutions (UK) Ltd

APPLICATION NOTE:
TECHNOLOGY SOLUTIONS ASCII 2 APIs THEORY OF OPERATION

CONTENT

History

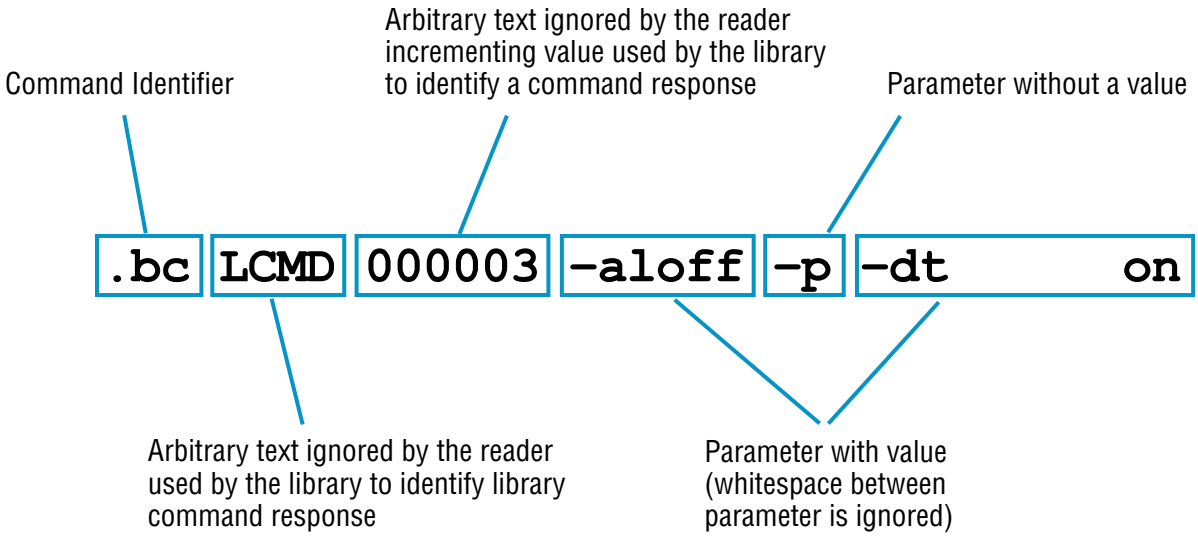
Version	Date	Modifications
1.0	22/05/2014	First Release

INTRODUCTION

Technology Solutions has implemented a common object model for Android, iOS, Windows Desktop, Windows Mobile and Windows Phone. This document is provided as an overview of using this object model to perform ASCII 2 commands with a Technology Solutions ASCII 2 device.

QUICK SUMMARY OF THE ASCII 2 PROTOCOL

COMMAND STRUCTURE



A command is a single ASCII line terminated with carriage return, line feed or both. It starts with a period '.' followed by two lower case characters to identify the command (e.g. '.iv' for inventory).

Each command supports a list of optional parameters. If the parameters are not sent to the reader in a particular command then the reader uses the last value sent for the parameter or the parameter's default if the parameter has never been sent. The value of the parameters are stored per command (i.e. the inventory output power is separate from the read transponder output power). A parameter on the command line starts with the minus sign '-' followed by one or more characters to identify the parameter and then the parameter value (if any).

".bc" Performs a barcode scan using the current parameters

".bc -x" Performs a barcode scan resetting the command to its defaults before performing the scan

".bc –al off –n" Performs a barcode command to set the alert action off with the 'take no action' parameter (update the parameter without executing the command).

Between the command identifier and the first parameter the reader will ignore any alpha numeric or white space. The Technology Solutions APIs use this fact to insert a fixed string “LCMD” to identify the source of the command as a library command and an incremented command index “000001” to uniquely identify the command. This is not used at all by the reader but echoed in the command started response line (see responses) which enables the library to identify the response for a specific command.

“.bc LCMD 000002 –al off –dt on”

RESPONSES

A response from a Technology Solutions ASCII 2 device consists of multiple lines. All but the last line in the response starts with a two capital letter response header followed by a colon ‘:’. The last line in the response is an empty line which marks the end of a response. Each line has a carriage return line feed pair to terminate the line. For each line the two capital letters before the colon identify what the value after the colon represents.

The first line in a response is always command started ‘CS:’. The value of this line is a copy of the command line sent to the reader that it is responding to. The last non empty line is either ‘OK:’ or ‘ER: nnn’ indicating whether the command completed successfully or not, ‘nnn’ will be a defined error code number. If a command ends with ‘ER: nnn’ then typically this will be preceded with an ‘ME:’ message line with a human readable version of the error code. Other lines can appear in the response depending on the command being issued.

Shown below is a successful inventory command response for two transponders which includes index ‘IX:’, EPC ‘EP:’, RSSI ‘RI’, PC ‘PC:’ and checksum ‘CR:’ for each transponder.

```
CS: .iv LCMD 000001 -con -ixon -eon -ron
IX: 0001
EP: 41524E4C3030303030310000
RI: -56
CR: 21AB
PC: 3000
IX: 0002
EP: 330DE29525C0210005F5F8F2
RI: -50
CR: 44D2
PC: 3000
OK:
{empty line}
```

EVENTS

The reader can be configured to send events that are not part of a command response. Each event is sent as a single line with a two capital letter identifier followed by a colon ':' then the event value. For example asynchronous switch notifications can be enabled using the switch action command '.sa –aon'. Switch events are sent each time the switch state changes.

The examples below show the event sent when asynchronous notifications sent when the switch changes to the double pressed state, off and single pressed state respectively.

```
SW: double
SW: off
SW: single
```

TRIGGER ACTIONS

The Technology Solutions ASCII 2 devices support a trigger with a single and double press. Even where the physical trigger is not present there are commands to perform a virtual press for a period of seconds. When a single or double press is actioned a command is executed sequentially multiple times until the press is released. The action for each type, single or double, can be set to off / read / write / inventory / barcode or a custom command line specified by a separate switch configuration command. By default the single press action is inventory transponders and the double press action is to scan a barcode.

API CLASSES OVERVIEW

The main class is a commander that is responsible for setting up and tearing down the connection to Technology Solutions ASCII 2 device.

The commander uses a serial transport. The serial transport has a write line method to write a command to the connected device and a read line method to read a line of a response from the reader. The serial transport also notifies the commander when new data is available to read. This abstract transport can be implemented as a serial port, a Bluetooth stream, a network socket or an External Accessory stream based on the mechanism used to communicate with the Technology Solutions ASCII 2 device.

The commander has a responder chain for receiving from the Technology Solutions ASCII 2 device. The responder chain holds a number of ASCII command responders that look for specific responses returned from the reader. For each ASCII line received the responders are asked in order to process the line until one responder indicates it has processed the line. Once a responder has processed the line then no further responders in the chain see the line. A specialized responder is the synchronous responder, this is used when a command is executed synchronously (see later). The synchronous responder delegates the processing of the line to the currently executing synchronous command.

MAPPING COMMANDS TO THE ASCII 2 PROTOCOL

The ASCII 2 protocol document details all the commands supported by the readers that implement a particular protocol version.

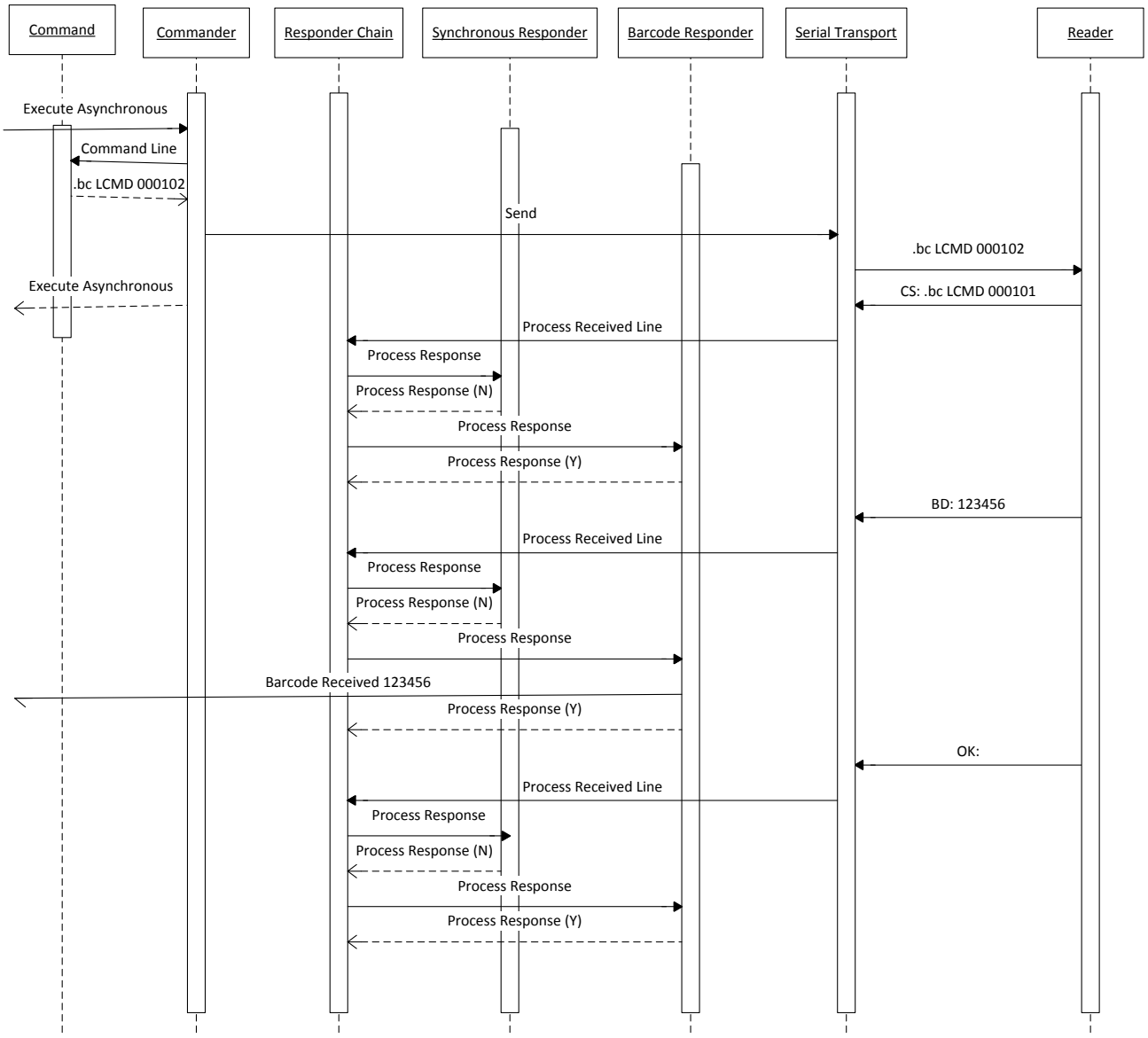
For each ASCII 2 protocol command there is a corresponding command class. Each command class provides properties for each optional parameter defined in the command. These properties default to null, nil, nothing so that they won't be sent to the reader when the command is executed unless set to a value. In this way the optional parameters are only sent to the reader when they are assigned a value.

Depending on the API each command class either 'has a' or implements its own responder. This responder can be added to the responder chain where a command is used to capture an asynchronous response or is used when the command is executed synchronously. When a command class's responder captures a response it provides properties to read the values received. Command classes may also raise an event, notification or provide a delegate for when a particular response is received. For example the barcode command class signals as a barcode is scanned and the commands that return one or more transponders signal as each transponder is received.

ASYNCHRONOUS AND SYNCHRONOUS COMMANDS

Commands are executed by passing a command class instance to the execute method of the commander. There is a concept of executing the command asynchronously or synchronously.

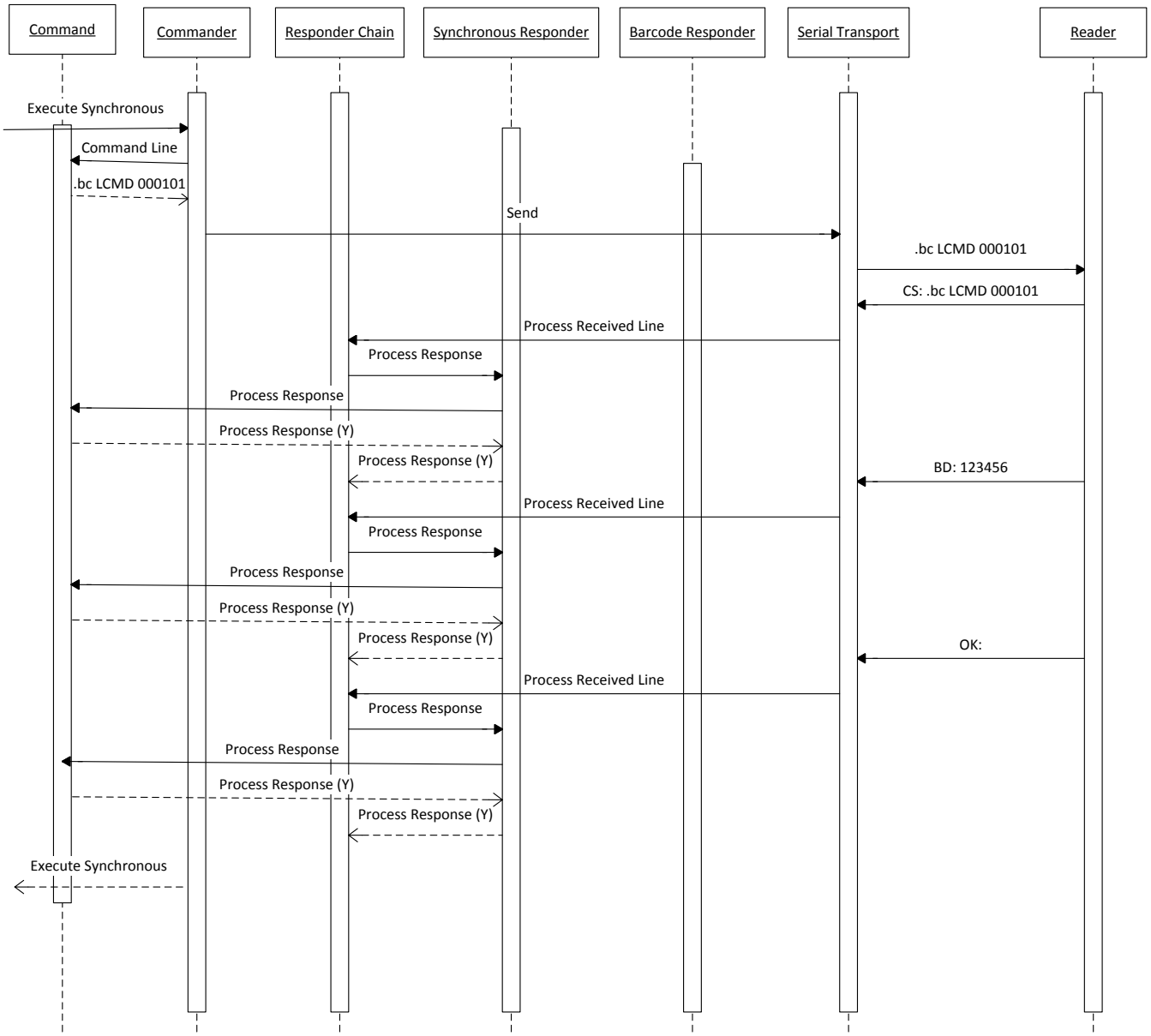
Asynchronous Execution



When a command is executed asynchronously the command instance passed to the execute method is only used to generate the command line to send to the reader based on the command parameters that have been set. Once the command has been sent the execute method returns. The responder chain will capture the response from the reader when it arrives.

In the diagram above the commander has a responder chain that consists of a synchronous responder followed by a barcode responder. Recall that the responders in the chain are asked in turn to process the received line. As there is no command executing synchronously the synchronous responder returns false for each response line. The barcode responder recognises the command started of a barcode response and therefore returns true for each line. As the barcode responder captures the response no further responders are checked.

Synchronous Execution



When a command is executed synchronously the command instance passed to the execute method is used to generate the command line and this is sent to the reader as before. The execute method then blocks for up to the max synchronous wait time to capture the response. To execute a command synchronously there must be a synchronous responder in the responder chain (typically an exception is thrown if a command is executed synchronously and there is no synchronous responder). The synchronous responder delegates its processing of each received line to the currently executing command. In this way the responder of the command passed to execute synchronously is updated with the response from the reader. Once the response is fully received or the max synchronous timeout is reached the execute method returns. When execute returns the command will contain the response from the reader (assuming it was received successfully).

When the command line is generated for a command class instance this is passed to the command's responder. In this way when a command is executed synchronously the only response that the command will capture is one which starts with a 'CS:' header that matches the LCMD and command index for the command sent. In this way a response that is returned from a trigger press will not be captured incorrectly as a response to the command sent to the reader.

In the diagram above the commander has a responder chain that consists of a synchronous responder followed by a barcode responder. Recall that the responders in the chain are asked in turn to process the received line. As there is a command executing synchronously the synchronous responder passes the response lines to the executing command. As the CS line includes the command line the command last sent to the reader it recognises the response as for the command line that was sent. It processes the line and returns true for each line until the response is complete. The barcode responder does not see any response lines as the synchronous responder returns true for each line (as the synchronous command processed the line) therefore the responder chain does not pass the line on to any further responders.

RESPONDER CHAIN

In the diagrams above a simple responder chain is setup that has only a synchronous responder followed by a barcode responder. A typical responder chain would contain

- {Logger responder}
- Synchronous responder
- Inventory responder
- Barcode responder
- Switch responder

The logger responder is an optional developer aid. Being first in the responder chain it will see every line received from the reader but has been set up to never mark a line as processed. It simply logs each line to a debug output before allowing further responders in the chain to examine and possibly capture the response.

The synchronous responder relays the process response to the currently executing synchronous command. This enables a synchronous command to receive its own response. It is placed before the general responders in the responder chain so that it will capture its own specific response before a more general responder captures it. i.e. If you execute a inventory command synchronously it will be sent with LCMD and index (e.g. 000999) the synchronous responder and hence the executing command will capture the response that starts with command started with ‘CS: .iv LCMD 000999’ where as a general inventory responder will capture any inventory response ‘CS: .iv’.

The inventory responder will capture transponder responses that have been executed asynchronously included those sent via the trigger press. The trigger press by default just executes ‘.iv’ so the response can be differentiated from the library commands as the ‘LCMD’ is absent. If the inventory responder were to be placed before the synchronous responder it would capture the response as an inventory response and the synchronous command would never receive its response.

The barcode responder will capture asynchronous barcode responses that have been executed asynchronously either from a command or from a trigger press.

The switch responder will capture the switch change events sent if asynchronous switch notifications are enabled.

CHANGING THE PARAMETERS USED FOR A COMMAND

As the commands implement their own responders an inventory command instance is placed into the responder chain to receive transponder events. What the reader sends for each transponder can be modified with command parameters (RSSI, PC, Index, and Checksum) however the command in the responder is passively listening for responses from the reader. To modify what the reader outputs for each transponder an inventory command must be sent to the reader with the updated parameters using the execute method of the commander. These changes will remain in effect until commanded subsequently or the reader reboots. An auto-run file can be deployed to the micro SD card within a Technology Solutions ASCII 2 device to make a reader operate with non-default parameters as soon as a boot is complete.

READING PARAMETERS

Commands that support parameters have a read parameters optional parameter. When present in a command line ('-p') the reader will send a 'PR:' line in the response with the supported parameters and their current values. To read the parameters set the read parameters parameter to TriState.Yes this will output the '-p' to the command line.

To read the parameters without performing the command set the take no action optional parameter ('-n') to TriState. Yes this allows the parameters of a command to be updated without executing the command (useful for the auto-run file to set up the parameters for commands without performing them).

When the responder of a command class receives a 'PR:' line it uses a parse parameters method to update the parameters of the command with the values received from the reader.

The example below reads the parameters of a barcode command without performing a barcode scan:

```
.bc -p -n

CS: .bc -p -dton
PR: -al on -dt on -e on -n -p -t 9 -x
OK:
```

FURTHER INFORMATION

Please email *support@tsl.uk.com* if you have further questions.

ABOUT TSL

ABOUT

TSL designs and manufactures both standard and custom embedded, snap on and standalone peripherals for handheld computer terminals. Embedded technologies include:

- RFID - Low Frequency, High Frequency & UHF
- *Bluetooth*® wireless technology
- Contact Smartcard
- Fingerprint Biometrics
- 1D and 2D Barcode Scanning
- Magnetic Card Readers
- OCR-B and ePassport

Utilizing class leading Industrial design, TSL develops products from concept through to high volume manufacture for Blue Chip companies around the world. Using the above technologies TSL develops innovative products in a timely and cost effective manner for a broad range of handheld devices.

CONTACT

Address:	Technology Solutions (UK) Limited, Suite C, Loughborough Technology Centre, Epinal Way, Loughborough, Leicestershire, LE11 3GE. United Kingdom.
Telephone:	+44 (0)1509 238248
Fax:	+44 (0)1509 220020
Email:	enquiries@tsl.uk.com
Website:	www.tsl.uk.com

ISO 9001: 2008