[MS-VUVP]:

VT-UTF8 and VT100+ Protocols

Revision Summary

Date	Revision History	Revision Class	Comments
5/11/2007	0.1	New	Version 0.1 release
8/10/2007	0.1.1	Editorial	Changed language and formatting in the technical content.
9/28/2007	0.1.2	Editorial	Changed language and formatting in the technical content.
10/23/2007	0.1.3	Editorial	Changed language and formatting in the technical content.
11/30/2007	0.1.4	Editorial	Changed language and formatting in the technical content.
1/25/2008	1.0	Editorial	Changed language and formatting in the technical content.
3/14/2008	2.0	Major	Updated and revised the technical content.
5/16/2008	2.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	2.0.2	Editorial	Changed language and formatting in the technical content.
7/25/2008	2.0.3	Editorial	Changed language and formatting in the technical content.
8/29/2008	2.0.4	Editorial	Changed language and formatting in the technical content.
10/24/2008	2.0.5	Editorial	Changed language and formatting in the technical content.
12/5/2008	3.0	Major	Updated and revised the technical content.
1/16/2009	3.0.1	Editorial	Changed language and formatting in the technical content.
2/27/2009	3.0.2	Editorial	Changed language and formatting in the technical content.
4/10/2009	3.0.3	Editorial	Changed language and formatting in the technical content.
5/22/2009	4.0	Major	Updated and revised the technical content.
7/2/2009	4.0.1	Editorial	Changed language and formatting in the technical content.
8/14/2009	4.0.2	Editorial	Changed language and formatting in the technical content.
9/25/2009	4.1	Minor	Clarified the meaning of the technical content.
11/6/2009	4.1.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	4.1.2	Editorial	Changed language and formatting in the technical content.
1/29/2010	4.1.3	Editorial	Changed language and formatting in the technical content.
3/12/2010	4.1.4	Editorial	Changed language and formatting in the technical content.
4/23/2010	4.1.5	Editorial	Changed language and formatting in the technical content.
6/4/2010	5.0	Major	Updated and revised the technical content.
7/16/2010	5.0	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	5.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	5.0	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
11/19/2010	5.0	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	5.1	Minor	Clarified the meaning of the technical content.
9/23/2011	5.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	6.0	Major	Updated and revised the technical content.
3/30/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	6.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	7.0	Major	Updated and revised the technical content.
11/14/2013	7.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	8.0	Major	Significantly changed the technical content.
10/16/2015	9.0	Major	Significantly changed the technical content.
7/14/2016	10.0	Major	Significantly changed the technical content.
6/1/2017	10.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The VT-UTF8 and VT100+ Protocols are used for point-to-point serial communication for **terminal** control and headless server configuration.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

American National Standards Institute (ANSI) character set: A character set defined by a code page approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [ISO/IEC-8859-1]. In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-**Unicode** or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on **Unicode**, so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

ASCII: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

console host: A server process that sends and receives data from a hosted text-based/character-mode application client.

management console: A remote computer that is used to interact with a local computer via a **terminal emulator**. A **management console** is often in a geographically different location than the local computer. A single **management console** can be used to interact with one or more local computers.

terminal: A text-based console. **Terminals** can be local or remote. A local **terminal** on a PC is typically an 80 × 25 text-format cell-based output that is displayed on a monitor.

terminal emulator: Software that runs a remote **terminal** on a **management console**. The **terminal emulator** uses a specified terminal type that must be agreed upon in advance via the local console and the remote **terminal**.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

uninterruptible power supply (UPS): A device that provides a backup short-term power source for occasions when utility power is lost. A **UPS** can be an intelligent device with which **management consoles** interact.

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

VT100: A terminal type, as defined by [VT100]. [VT100] provides the definition for an English language, 80 × 25 text console.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[VT100] Digital Equipment Corporation, "VT100 Series Technical Manual", September 1980, http://vt100.net/docs/vt100-tm/ek-vt100-tm-002.pdf

1.2.2 Informative References

[ACPI] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation, "Advanced Configuration and Power Interface Specification", October 2006, http://acpi.info/DOWNLOADS/ACPIspec30b.pdf

[MSDN-ANSI] Microsoft Corporation, "Unicode and Character Sets", http://msdn.microsoft.com/en-us/library/dd374083.aspx

[MSDN-ConsoleRef] Microsoft Corporation, "Console Reference", https://msdn.microsoft.com/en-us/library/windows/desktop/ms682087(v=vs.85).aspx

[XTermControl] Moy, E., Gildea S., and Dickey T., "XTerm Control Sequences", http://invisible-island.net/xterm/ctlseqs/ctlseqs.html

1.3 Overview

The VT-UTF8 and VT100+ protocols are used for point-to-point serial client/server communication.

Typically, the client is a **terminal emulator** and acts as a **management console**; the server is a platform component that can be a basic input/output (BIOS), **uninterruptible power supply (UPS)** processor, service processor, or software driver. For example, the protocols allow server power management to be invoked from a serial console.

Alternatively, the server is a terminal emulator and acts as a **console host** for an application client. The application can be running locally on the same machine as the emulator or remotely over any form of network connection. The client application emits a sequence of characters that are transported to the terminal emulator server and presented to the user on the screen. The VT100+ protocol allows

graphical signaling information to be interleaved with character data within the sequence of characters as it travels between the client and server.

1.3.1 VT-UTF8

The VT-UTF8 protocol uses **UTF-8** encoding to allow **Unicode** characters to be used without conflicting with the original **VT100** protocol commands. Using Unicode characters, for example, allows non-English output on a client display.

1.3.2 VT100+

The VT100+ protocol extends the original **VT100 terminal** specification ([VT100]) to support the use of color in a client display terminal, to define character sequences for function keys on the U.S. standard keyboard (101 keys), and to make provisions for additional graphic characters.

1.4 Relationship to Other Protocols

This protocol extends the **VT100** protocol, as specified in [VT100].

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

The VT-UTF8 and VT100+ protocols can apply to text-mode serial connections to physical hardware devices in emergency scenarios such as power outages.

A text-mode serial connection can alternatively be the connection between a client application and a **console host** window process. In this case, "serial" refers to the practice of signaling messages on a single stream.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

The following sections specify how the VT-UTF8 and VT100+ protocols are transported and message syntax.

2.1 Transport

The VT-UTF8 and VT100+ protocols are transmitted over a serial port (COM port) connection.

2.2 Message Syntax

2.2.1 VT-UTF8 and VT100+ for Serial/UPS

The VT-UTF8 and VT100+ client console command request or server response consists of a single field that contains the "<ESC>" character followed by one or more characters. The entire sequence MUST be sent within 2 seconds of the initial <ESC>, as specified in sections 3.2.2 and 3.2.6.

Command_Sequence: The character sequence containing the entire client request.

Character sequence	Description
<ESC>R<ESC>r<ESC>R	Reset. If the server is a BIOS with control of the serial port and reset is supported, the system MUST be reset within 5 seconds. If the server is a **UPS**, an application-specific integrated circuit (ASIC), a service processor, or a software driver, and has control of the serial port, the server MUST be reset within 1 second.
<ESC>(	Invoke the server ASIC or service processor. After detecting this command sequence, the server ASIC or service processor MUST take control of the server serial port for console input/output (I/O). The server ASIC or service processor MUST return an Acknowledge Sequence within 1 second.
<ESC>)	Invoke the UPS processor. After detecting this command sequence, the server UPS processor MUST take control of the server serial port for console I/O. The server UPS processor MUST return an Acknowledge Sequence within 1 second.
<ESC>*	Acknowledge sequence. This response MUST be returned by the server UPS, ASIC, or service processor before any other server response, and within 1 second after it is invoked.
<ESC>Q	Exit without displaying the user interface. The server UPS, ASIC, or service processor MUST immediately release control of the server serial port, without interaction with the client.
<ESC>^	Wake up. This requests that the server ASIC or service processor turn on the server within 1 second or wake the server from sleep state S1–S4 (for more information on sleep states, see [ACPI]). If the server is already turned on, server operation MUST NOT be disturbed. The server ASIC or service processor MUST return an Acknowledge Sequence within 1 second.

2.2.2 VT100+ Character Extensions for Serial/UPS

The VT100+ character extensions conform to ANSI conventions for setting client display foreground and background colors. The **VT100** standard, approved by the American National Standards Institute, defines meanings to coded sequences of characters passed from computer to terminal, as specified in [VT100]. The VT100+ extensions use the same general format of coded sequences of characters, but assign additional meanings for sequences that were not defined in the VT100 standard. The VT100+

character and key extensions also support selected keyboard keys and graphics characters that are not part of the original VT100 **terminal** specification. Function keys on a U.S. standard keyboard (101 keys) are not equivalent to similarly named keys on a VT100 terminal keyboard.

2.2.2.1 Client Display Terminal Color Extensions

The following sections list the character sequences and color values for the VT100+ extensions.

2.2.2.1.1 Character Sequences

The following table lists the character sequences for the VT100+ extensions for **uninterruptible power supply (UPS)**.

Character sequence	Description
<ESC>[%1m	Sets video mode and color, where %1 is the color value.
<ESC>[%1;%2;%3m	Sets multiple color values, where %1, %2, and %3 are the color values. Color values MUST NOT overlap.

2.2.2.1.2 Color Values

The following table lists the color values for the VT100+ extensions.

Color value	Description
1	Video bold mode
5	Video blinking mode
30	Foreground black
31	Foreground red
32	Foreground green
33	Foreground yellow
34	Foreground blue
35	Foreground magenta
36	Foreground cyan
37	Foreground white
40	Background black
41	Background red
42	Background green
43	Background yellow
44	Background blue
45	Background magenta
46	Background cyan

Color value	Description
47	Background white

2.2.2.2 Character and Key Extensions

The following table lists the character sequences that correspond to the VT100+ character and key extensions for **uninterruptible power supply (UPS)**.

Note If a modifier sequence (SHIFT modifier, ALT modifier, or CONTROL modifier) is not followed by a character sequence within 2 seconds, the modifier sequence is disregarded.

Character or key	Character sequence
HOME key	<ESC>h
END key	<ESC>k
INSERT key	<ESC>+
DELETE key	<ESC>-
PAGE UP key	<ESC>?
PAGE DOWN key	<ESC>/
F1 key	<ESC>1
F2 key	<ESC>2
F3 key	<ESC>3
F4 key	<ESC>4
F5 key	<ESC>5
F6 key	<ESC>6
F7 key	<ESC>7
F8 key	<ESC>8
F9 key	<ESC>9
F10 key	<ESC>0
F11 key	<ESC>!
F12 key	<ESC>@
SHIFT modifier	<ESC><Ctrl>s
ALT modifier	<ESC><Ctrl>a
CONTROL modifier	<ESC><Ctrl>c
Reserved	<ESC>#
Reserved	<ESC>A
Reserved	<ESC>B

Character or key	Character sequence
Reserved	<ESC>C
Reserved	<ESC>D
Reserved	<ESC>&
Reserved	<ESC>*
Reserved	<ESC>.
Reserved	<ESC>R
Reserved	<ESC>r

2.2.3 VT100+ Character Extensions for Console Host

The VT100+ character extensions for **console host** conform to **ANSI** conventions for setting client display foreground and background colors. The extensions use the same general format of coded sequences of characters, but assign additional meanings to align with Xterm control sequences, as described in [XTermControl]. This provides interoperability with terminal emulators on Linux and Mac OS computers.<1>

2.2.3.1 Client Display Terminal Color Extensions

The following sections list the character sequences and color values for the VT100+ extensions.

2.2.3.1.1 Character Sequences

The following table lists the character sequences for the VT100+ extensions for **console host**.

Character sequence	Description
<ESC>[m	Sets default video mode and color. Equivalent to <ESC>[0m.
<ESC>[%1m	Sets video mode and color, where %1 is the color value.
<ESC>[%1;%2;…;%16m	Sets multiple color values, where %1, %2, and %16 are the color values. Up to 16 values can be used separated by semicolons. Additional values beyond 16 are discarded.

2.2.3.1.2 Color Values

The following table lists the color values for the VT100+ extensions for **console host**.

Color value	Description
0	Video default mode—clears flags and restores colors to default (when the session began)
1	Video bold/intense mode—implementation-specific color/font differentiation
4	Video underline mode
7	Video reverse mode—swaps foreground and background colors

Color value	Description
24	Unset video underline mode
27	Unset video reverse mode
30	Foreground black
31	Foreground red
32	Foreground green
33	Foreground yellow
34	Foreground blue
35	Foreground magenta
36	Foreground cyan
37	Foreground white
39	Foreground default
40	Background black
41	Background red
42	Background green
43	Background yellow
44	Background blue
45	Background magenta
46	Background cyan
47	Background white
49	Background default
90	Foreground black bold/intense
91	Foreground red bold/intense
92	Foreground green bold/intense
93	Foreground yellow bold/intense
94	Foreground blue bold/intense
95	Foreground magenta bold/intense
96	Foreground cyan bold/intense
97	Foreground white bold/intense
100	Background black bold/intense
101	Background red bold/intense
102	Background green bold/intense
103	Background yellow bold/intense

Color value	Description
104	Background blue bold/intense
105	Background magenta bold/intense
106	Background cyan bold/intense
107	Background white bold/intense

2.2.3.2 Character and Key Extensions

The following table lists the character sequences that correspond to the VT100+ character and key extensions for **console host**.

Character or key	Character sequence
Show Cursor	<ESC>[?h
Hide Cursor	<ESC>[?l (lowercase L)

3 Protocol Details

3.1 Server Details

This section applies to both the **console host** server and **uninterruptible power supply (UPS)** server implementations.<2>

3.1.1 Abstract Data Model

When the **uninterruptible power supply (UPS)** server receives an escape character, it MUST enter an escape state for 2 seconds as it waits for additional characters.

When the **console host** server receives an escape character, it MUST wait indefinitely for additional characters.

For more information, see section 3.1.2.

3.1.2 Timers

When an escape sequence is signaled to an **uninterruptible power supply (UPS)** server, the server MUST receive the escaped characters within 2 seconds. For example, the sequence "<ESC>(" invokes the service processor. The "(" character MUST be received by the server within 2 seconds of when "<ESC>" is received.

When an escape sequence is signaled to a **console host** server, the server MUST wait indefinitely for the next character before invoking the service processor.

3.1.3 Initialization

The **uninterruptible power supply (UPS)** server requires no initialization.

The **console host** server requires initialization by the client application or by the user. Client applications can initialize the server through the SetConsoleMode function (see [MSDN-ConsoleRef]). Users or system administrators can set initialization to occur by default by setting the registry key at HKCU\VirtualTerminalLevel for each user account to a nonzero value.

3.1.4 Higher-Layer Triggered Events

The server has no higher-layer triggered events.

3.1.5 Message Processing Events and Sequencing Rules

The following sections specify the behavior of this protocol when receiving correct requests. Incorrect requests MUST be ignored.

3.1.5.1 Sending VT-UTF8 and VT100+ Requests

The original **VT100** protocol, as specified in [VT100], uses the **ASCII** character set. The **UTF-8** algorithm MUST map a **Unicode** character into a string of 8-bit bytes. The number of 8-bit bytes depends on the bit width of the Unicode character, as shown in the following table.

Bit width	UTF8 encoding
0 - 7	0xxxxxxx

Bit width	UTF8 encoding
8 - 11	110xxxxx 10xxxxxx
12 - 16	1110xxxx 10xxxxxx 10xxxxxx

3.1.5.2 Receiving VT-UTF8 and VT100+ Requests

When a series of bytes is received by the server, it MUST be decoded into the appropriate 16-bit **Unicode** character. The leading byte MAY be 0x00000000.<3>

The decoded 16-bit Unicode character is then presented in the server representation, as specified in [VT100] table A-11.

If an escape sequence is received, the server processes all the characters in the escape sequence as a single action that is described by the escape sequence, instead of processing each literal character in the sequence.

3.1.5.3 Receiving Character and Key Extensions

When a series of bytes is received by the server, it MUST be decoded into the appropriate 16-bit **Unicode** character. The leading byte MAY be 0x00000000.

The decoded 16-bit Unicode character is then presented in the server representation according to the tables in [VT100] table A-11.

If an escape sequence is received, the server processes all the characters in the escape sequence as a single action that is described by the escape sequence, instead of processing each literal character in the sequence.

3.1.6 Timer Events

If the server does not receive the escaped characters within 2 seconds of sequence initiation, the entire sequence is discarded.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

When the client receives an escape character, it MUST enter an escape state for 2 seconds as it waits for additional characters. For more information, see section 3.2.2.

3.2.2 Timers

When an escape sequence is signaled, the client MUST receive the escaped characters within 2 seconds.

For example, the sequence "<ESC>(" invokes the service processor. The "(" character MUST be received by the server within 2 seconds of when "<ESC>" is received.

3.2.3 Initialization

The client requires no initialization.

3.2.4 Higher-Layer Triggered Events

The client has no higher-layer triggered events.

3.2.5 Message Processing Events and Sequencing Rules

The following sections specify this protocol's behavior when receiving correct requests. Incorrect requests MUST be ignored.

3.2.5.1 Sending VT-UTF8 and VT100+ Requests

The original **VT100** protocol, as specified in [VT100], uses the **ASCII** character set. The **UTF-8** algorithm MUST map a **Unicode** character into a string of 8-bit bytes. The number of 8-bit bytes depends on the bit width of the Unicode character, as shown in the following table.

Bit width	UTF-8 encoding
0-7	0xxxxxxx
8-11	110xxxxx 10xxxxxx
12-16	1110xxxx 10xxxxxx 10xxxxxx

3.2.5.2 Receiving VT-UTF8 and VT100+ Requests

When a series of bytes is received by the client, it MUST be decoded into the appropriate 16-bit **Unicode** character. The leading byte MAY be 0x00000000.

The decoded 16-bit Unicode character is then presented in the client representation according to the tables as specified in [VT100] table A-11.

If an escape sequence is received, the client processes all the characters in the escape sequence as a single action that is described by the escape sequence, instead of processing each literal character in the sequence.

3.2.5.3 Receiving Client Display Terminal Color Extensions

When a series of bytes is received by the client, it MUST be decoded into the appropriate 16-bit **Unicode** character.

The leading byte MAY be 0x00000000. The decoded 16-bit Unicode character is then presented in the client representation according to the tables as specified in [VT100] table A-11.

If an escape sequence is received, the client processes all the characters in the escape sequence as a single action that is described by the escape sequence, instead of processing each literal character in the sequence.

3.2.5.4 Receiving Character and Key Extensions

When a series of bytes is received by the client, it MUST be decoded into the appropriate 16-bit **Unicode** character.

The leading byte MAY be 0x00000000. The decoded 16-bit Unicode character is then presented in the client representation according to the tables as specified in [VT100] table A-11.

If an escape sequence is received, the client processes all the characters in the escape sequence as a single action that is described by the escape sequence, instead of processing each literal character in the sequence.

3.2.6 Timer Events

If the client does not receive the escaped characters within 2 seconds of sequence initiation, the entire sequence is discarded.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 VT-UTF8 Example for Serial/UPS

A server wants to transmit the **Unicode** character stream that is represented by the following code point sequence.

```
<004D, 0430, 4E8C>
```

The VT-UTF8 encoding of the Unicode character stream would be

```
<4D D0 B0 E4 BA 8C>
```

where

- <4D> corresponds to 0x004D
- <D0 B0> corresponds to 0x0430
- <E4 BA 8C> corresponds to 0x4E8C

This stream can be transmitted to the client and then decoded by reconstructing the same Unicode character stream.

4.2 VT100+ Example for Serial/UPS

A user wishes to set the video mode to bold, the text foreground to black, and the background to green. The user sends the sequence

```
<ESC>[1,30,42m
```

as specified in section 2.2.2.1.1.

4.3 VT100+ Example for Console Host

The following sequence patterns can be found in section 2.2.3.1.1.

A user wishes to reset all color/font information in an area of text back to what it originally was when the session started. The user sends the sequence:

```
<ESC>[m
```

A user wishes to set a bright/bold green foreground color with a dark blue background color. The user has two options and sends either of the following sequences:

```
<ESC>[32;1;44m
-OR-
<ESC>[92;44m
```

Each item of the sequence will be applied in the order it is received. 32 will set the dark green foreground, 1 will turn it into a bright/bold foreground color, then 44 will set the dark blue background. Or 92 will set a bright green foreground in one step.

A user can specify multiple overlapping colors and they will be applied from beginning to end in the order received. The final applicable color in the sequence will be the resulting video mode. In the following example, a user sets blue foreground then magenta foreground:

```
<ESC>32;35m
```

The final result will be a magenta foreground mode.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Windows NT operating system
- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 2.2.3: The **console host terminal emulator** sequences listed in this section are not supported on the following versions of Windows:

- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows 10

- Windows Server 2016

<2> Section 3.1: The console host implementation applies only to Windows 10 and Windows Server 2016.

<3> Section 3.1.5.2: In the console host implementation, this service is provided by MultiByteToWideChar (see [MSDN-ANSI]) in respect to the current code page. The code page is loaded from the system on console host startup. It can be modified by the running application through GetConsoleOutputCP and SetConsoleOutputCP (see [MSDN-ConsoleRef]).

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

T

U

V