[MS-VUVP]:
VT-UTF8 and VT100+ Protocols

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
05/11/2007	0.1		MCPP Milestone 4 Initial Availability
08/10/2007	0.1.1	Editorial	Revised and edited the technical content.
09/28/2007	0.1.2	Editorial	Revised and edited the technical content.
10/23/2007	0.1.3	Editorial	Revised and edited the technical content.
11/30/2007	0.1.4	Editorial	Revised and edited the technical content.
01/25/2008	1.0	Editorial	Revised and edited the technical content.
03/14/2008	2.0	Major	Updated and revised the technical content.
05/16/2008	2.0.1	Editorial	Revised and edited the technical content.
06/20/2008	2.0.2	Editorial	Revised and edited the technical content.
07/25/2008	2.0.3	Editorial	Revised and edited the technical content.
08/29/2008	2.0.4	Editorial	Revised and edited the technical content.
10/24/2008	2.0.5	Editorial	Revised and edited the technical content.
12/05/2008	3.0	Major	Updated and revised the technical content.
01/16/2009	3.0.1	Editorial	Revised and edited the technical content.
02/27/2009	3.0.2	Editorial	Revised and edited the technical content.
04/10/2009	3.0.3	Editorial	Revised and edited the technical content.
05/22/2009	4.0	Major	Updated and revised the technical content.
07/02/2009	4.0.1	Editorial	Revised and edited the technical content.
08/14/2009	4.0.2	Editorial	Revised and edited the technical content.
09/25/2009	4.1	Minor	Updated the technical content.
11/06/2009	4.1.1	Editorial	Revised and edited the technical content.
12/18/2009	4.1.2	Editorial	Revised and edited the technical content.
01/29/2010	4.1.3	Editorial	Revised and edited the technical content.
03/12/2010	4.1.4	Editorial	Revised and edited the technical content.
04/23/2010	4.1.5	Editorial	Revised and edited the technical content.
06/04/2010	5.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
07/16/2010	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	5.1	Minor	Clarified the meaning of the technical content.
09/23/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	6.0	Major	Significantly changed the technical content.
03/30/2012	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	7.0	Major	Significantly changed the technical content.

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 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

ASCII
Unicode
UTF-8

The following terms are specific to this document:

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** may 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**.

terminal type: The specification of how certain byte sequences should be interpreted as data is sent to and from the **terminal**. For example, a **terminal type** might define how the foreground and background display colors are set.

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

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 described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[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

If you have any trouble finding [VT100], please check here.

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

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

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 may 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.

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 only apply to text-mode serial connections.

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+

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 an 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 one 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

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.

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

Color value	Description
44	Background blue
45	Background magenta
46	Background cyan
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.

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

Character or key	Character sequence
ALT modifier	<ESC><Ctrl>a
CONTROL modifier	<ESC><Ctrl>c
Reserved	<ESC>#
Reserved	<ESC>A
Reserved	<ESC>B
Reserved	<ESC>C
Reserved	<ESC>D
Reserved	<ESC>&
Reserved	<ESC>*
Reserved	<ESC>.
Reserved	<ESC>R
Reserved	<ESC>r

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

When the server receives an escape character, it MUST enter an escape state for two seconds as it waits for additional characters. For more information, see section 3.1.2.

3.1.2 Timers

When an escape sequence is signaled, the server MUST receive the escaped characters within two seconds. For example, the sequence "<ESC>(" invokes the service processor. The "(" character must be received by the server within two seconds of when "<ESC>" is received.

3.1.3 Initialization

The server requires no initialization.

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
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.

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

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

4.2 VT100+ Example

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.

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

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.

7 Change Tracking

This section identifies changes that were made to the [MS-VUVP] protocol document between the January 2013 and August 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix A: Product Behavior	Modified this section to include references to Windows Server 2012 R2 operating system.	Y	Content updated.

8 Index

T

U

V