



WIP AT Commands User Guide

WIPsoft 5.41



SIERRA
WIRELESS

WM_DEV_OAT_UGD_076
009
February 18, 2011

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where blasting is in progress, where explosive atmospheres may be present, near medical equipment, near life support equipment, or any equipment which may be susceptible to any form of radio interference. In such areas, the Sierra Wireless modem **MUST BE POWERED OFF**. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM® 3G.



This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group.

Copyright

© 2011 Sierra Wireless. All rights reserved.

Trademarks

AirCard® and Watcher® are registered trademarks of Sierra Wireless. Sierra Wireless™, AirPrime™, AirLink™, AirVantage™ and the Sierra Wireless logo are trademarks of Sierra Wireless.

wavecom®, , ®, inSIM®, WAVECOM®, WISMO®, Wireless Microprocessor®, Wireless CPU®, Open AT® are filed or registered trademarks of Sierra Wireless S.A. in France and/or in other countries.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh and Mac OS are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of the respective owners.

Contact Information

| | | |
|-------------|---|--|
| Sales Desk: | Phone: | 1-604-232-1488 |
| | Hours: | 8:00 AM to 5:00 PM Pacific Time |
| | E-mail: | sales@sierrawireless.com |
| Post: | Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4 | |
| Fax: | 1-604-231-1109 | |
| Web: | www.sierrawireless.com | |

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: www.sierrawireless.com

Document History

| Version | Date | Updates |
|---------|-------------------|---|
| 001 | March 14 2008 | Creation |
| 002 | May 30 2008 | Update for WIPsoft v5.01 |
| 003 | December 15, 2008 | Update for WIPsoft v5.11 |
| 004 | March 9, 2009 | Update for WIPsoft v5.12 |
| 005 | July 28, 2009 | <p>Updated definition of Error 850 and 852 definitions updated.</p> <p>Added Note: If the Ethernet bearer support is defined, the Ethernet driver is subscribed on executing (+WIPCFG = 1).</p> <p>AT_WIP_NET_OPT_AUTO_SWITCH definition updated.</p> <p><bid> parameter identifier updated to ETHER.</p> <p>Seven new rows added to Parameters table starting with Parameter number 23 through <netmask IP @*>.</p> <p>Caution regarding WIP_BOPT_IP_NETMASK and WIP_BOPT_IP_GW added.</p> <p>Example updated starting at AT+WIPBR=1,4 through the end of the table.</p> <p>Note that The Ethernet bearer can be started only in client mode added.</p> <p>Definition updated and note added to discuss that data can be transferred using two modes.</p> <p>Mode 5 added and AT+WIPFILE=4,1,5,"report.log" example enhanced to reflect new mode.</p> <p>New +WIPDATA error added.</p> |
| 006 | February 22, 2010 | Added opt num parameters WIP_BOPT_DEBUG_PKT and WIP_BOPT_RESTART to the +WIPBR command. |
| 007 | March 24, 2010 | Updated version number |
| 008 | July 20, 2010 | Clarified the definition of the WIP_BOPT_PPP_CHAP option for the +WIPBR command. |

| Version | Date | Updates |
|---------|-------------------|--|
| 009 | February 18, 2011 | <p>Corrected erroneous syntax for mode=1 in the Service Creation +WIPCREATE section.</p> <p>Added a new <opt num> 28 and its corresponding GPRS CME Errors to the Bearers Handling +WIPBR section.</p> <p>Added new DATA Offline function including:</p> <ul style="list-style-type: none"> • New DATA Offline session +WIPDATARW section. • New <opt num> 15 in the IP Stack Handling +WIPCFG section. • New <mode> 6, 7, 8, and 9 in the File Exchange +WIPFILE section. • New <mode> 3 and <error> 836 in the Socket Data exchange +WIPDATA section. • New <error> 853. • New FTP DATA Offline example. <p>Added new MMS function including:</p> <ul style="list-style-type: none"> • Added new MMS <mode> = 8 and details to the subsections of the following commands: <ul style="list-style-type: none"> • Service Creation +WIPCREATE • Closing a Service +WIPCLOSE • Service Option Handling +WIPOPT, including additional options that can be applied to MMS sessions • File Exchange +WIPFILE • MMS Example • MMS Errors 890 through 897 • Note regarding the WIP_MMS_DONE option. <p>Added new TCP Options function as <opt num> 16 and 17 in the IP Stack Handling +WIPCFG section.</p> |



Contents

| | |
|---|-----------|
| 1. INTRODUCTION | 10 |
| 1.1. Abbreviations | 10 |
| 1.2. Logos | 11 |
| 1.3. AT Commands Presentation Rules | 12 |
| 2. AT COMMAND SYNTAX | 13 |
| 2.1. Command Line | 13 |
| 2.2. Information Responses and Result Codes | 13 |
| 3. PRINCIPLES | 14 |
| 3.1. Sockets Identification | 15 |
| 3.1.1. Possible Protocols | 15 |
| 3.1.2. Number of Sockets | 15 |
| 3.1.3. Notes | 15 |
| 4. GENERAL CONFIGURATION | 16 |
| 4.1. IP Stack Handling +WIPCFG | 16 |
| 4.1.1. Description | 16 |
| 4.1.2. Syntax | 16 |
| 4.1.3. Parameters and Defined Values | 18 |
| 4.1.4. Parameter Storage | 21 |
| 4.1.5. Possible Errors | 21 |
| 4.1.6. Examples | 22 |
| 4.1.7. Notes | 24 |
| 4.2. Bearers Handling +WIPBR | 25 |
| 4.2.1. Description | 25 |
| 4.2.2. Syntax | 25 |
| 4.2.3. Parameters and Defined Values | 26 |
| 4.2.4. Parameter Storage | 29 |
| 4.2.5. Possible Errors | 30 |
| 4.2.5.1. General CME Errors | 30 |
| 4.2.5.2. GPRS CME Errors | 30 |
| 4.2.6. Examples | 31 |
| 4.2.7. Notes | 33 |
| 4.2.7.1. For Starting a Bearer | 33 |
| 5. IP PROTOCOL SERVICES | 35 |
| 5.1. Service Creation +WIPCREATE | 35 |
| 5.1.1. Description | 35 |
| 5.1.2. Syntax | 36 |
| 5.1.3. Parameters and Defined Values | 38 |
| 5.1.4. Parameter Storage | 40 |
| 5.1.5. Possible Errors | 40 |
| 5.1.6. Examples | 40 |
| 5.1.7. Notes | 42 |

| | | |
|-----------|--|-----------|
| 5.2. | Closing a Service +WIPCLOSE | 44 |
| 5.2.1. | Description | 44 |
| 5.2.2. | Syntax | 44 |
| 5.2.3. | Parameters and Defined Values | 44 |
| 5.2.4. | Parameter Storage | 45 |
| 5.2.5. | Possible Errors | 45 |
| 5.2.6. | Examples | 45 |
| 5.2.7. | Notes | 46 |
| 5.3. | Service Option Handling +WIPOPT | 47 |
| 5.3.1. | Description | 47 |
| 5.3.2. | Syntax | 47 |
| 5.3.3. | Parameters and Defined Values | 48 |
| 5.3.4. | Parameter Storage | 48 |
| 5.3.5. | Possible Errors | 49 |
| 5.3.6. | Examples | 49 |
| 5.3.7. | Notes | 51 |
| 5.3.7.1. | Options that can be applied to UDP, TCP Client, TCP Server Sockets | 51 |
| 5.3.7.2. | Options that can be applied to FTP Session | 52 |
| 5.3.7.3. | Options that can be applied to HTTP Session | 52 |
| 5.3.7.4. | Options that can be applied to SMTP Session | 53 |
| 5.3.7.5. | Options that can be applied to POP3 Session | 54 |
| 5.3.7.6. | Options that can be applied to MMS sessions | 55 |
| 6. | DATA EXCHANGE FOR PROTOCOL SERVICES..... | 58 |
| 6.1. | File Exchange +WIPFILE | 58 |
| 6.1.1. | Description | 58 |
| 6.1.1.1. | [ETX] Escaping Mechanism | 58 |
| 6.1.1.2. | [DLE] Escaping Mechanism | 59 |
| 6.1.2. | FTP/HTTP/SMTP Session in Continuous Mode | 60 |
| 6.1.3. | FTP Session in Continuous Transparent Mode | 61 |
| 6.1.4. | Syntax | 61 |
| 6.1.5. | Parameters and Defined Values | 63 |
| 6.1.6. | Parameter Storage | 65 |
| 6.1.7. | Possible Errors | 65 |
| 6.1.8. | Examples | 66 |
| 6.1.9. | Notes | 67 |
| 6.2. | Socket Data exchange +WIPDATA..... | 68 |
| 6.2.1. | Description | 68 |
| 6.2.2. | Continuous Mode | 68 |
| 6.2.2.1. | TCP Sockets in Continuous mode | 68 |
| 6.2.2.2. | UDP Sockets in Continuous mode | 68 |
| 6.2.2.3. | [ETX] Escaping Mechanism | 69 |
| 6.2.2.4. | [DLE] Escaping Mechanism | 71 |
| 6.2.3. | Continuous Transparent Mode | 72 |
| 6.2.3.1. | TCP Sockets in Continuous Transparent Mode | 72 |
| 6.2.3.2. | UDP Sockets in Continuous Transparent Mode | 72 |
| 6.2.4. | Leaving Continuous /Continuous Transparent Mode | 72 |
| 6.2.5. | Resetting TCP Sockets | 73 |
| 6.2.6. | Syntax | 73 |
| 6.2.7. | Parameters and Defined Values | 74 |
| 6.2.8. | Parameter Storage | 74 |

| | | |
|-----------|---|-----------|
| 6.2.9. | Possible Errors | 74 |
| 6.2.10. | Examples | 75 |
| 6.2.11. | Notes | 76 |
| 6.2.11.1. | Continuous Mode (Non Transparent) for a TCP Mapped Socket..... | 76 |
| 6.2.11.2. | Mapping/Unmapping of a Mapped UDP and TCP Socket..... | 77 |
| 6.2.11.3. | Time out Mechanism to know the state of the Peer TCP Socket | 78 |
| 6.2.11.4. | Packet Segmentation in TCP Socket | 78 |
| 6.2.11.5. | Packet Segmentation in UDP Socket..... | 79 |
| 6.3. | DATA Offline session +WIPDATARW..... | 80 |
| 6.3.1. | Restrictions..... | 80 |
| 6.3.2. | Syntax | 80 |
| 6.3.3. | Parameters and Defined Values | 81 |
| 6.3.4. | Parameter Storage | 81 |
| 6.3.5. | Possible Errors | 82 |
| 6.3.6. | Examples..... | 82 |
| 7. | PING SERVICES | 84 |
| 7.1. | PING command +WIPPING | 84 |
| 7.1.1. | Description | 84 |
| 7.1.2. | Syntax | 84 |
| 7.1.3. | Parameters and Defined Values | 85 |
| 7.1.4. | Parameter Storage | 85 |
| 7.1.5. | Possible Errors | 85 |
| 7.1.6. | Examples..... | 86 |
| 8. | WIPSOFT LIBRARY API..... | 87 |
| 8.1. | Required Header File | 87 |
| 8.2. | The wip_ATCmdSubscribe Function..... | 87 |
| 8.2.1. | Prototype | 87 |
| 8.2.2. | Parameters..... | 87 |
| 8.2.3. | Returned Values | 87 |
| 8.3. | The wip_ATCmdUnsubscribe Function..... | 88 |
| 8.3.1. | Prototype | 88 |
| 8.3.2. | Parameters..... | 88 |
| 8.3.3. | Returned Values | 88 |
| 9. | EXAMPLES OF APPLICATION..... | 89 |
| 9.1. | TCP Socket | 89 |
| 9.1.1. | TCP Server Socket | 89 |
| 9.1.1.1. | Using GPRS bearer | 89 |
| 9.1.1.2. | Using GSM bearer | 90 |
| 9.1.2. | TCP Client Socket..... | 91 |
| 9.1.2.1. | Using GPRS Bearer..... | 91 |
| 9.1.2.2. | Using GSM Bearer | 92 |
| 9.2. | UDP Socket | 93 |
| 9.3. | PING | 95 |
| 9.4. | FTP | 96 |
| 9.5. | FTP DATA Offline..... | 97 |
| 9.6. | HTTP | 98 |

| | | |
|------------|--|------------|
| 9.7. | SMTP..... | 99 |
| 9.8. | POP3 | 101 |
| 9.9. | MMS | 103 |
| 9.10. | Creating a TCP Server, spawning the maximum TCP Socket (for the configured Server) 106 | |
| 9.11. | Creating a Server and try to create a TCP Client/Server on a reserved index (reserved by the Server) will fail. | 107 |
| 9.12. | Create a TCP Client and try to create a TCP Server with indexes range containing TCP Client will fail. | 109 |
| 9.13. | Creating 8 UDP sockets, 8 TCP clients and 4 TCP servers. | 110 |
| 9.14. | Changing the MAX SOCK_NUM option value and try to create 8 UDP sockets, 8 TCP Client sockets and 4 TCP Server sockets. | 113 |
| 9.15. | Creating 8 UDP sockets, 8 TCP Clients, 4 TCP Servers and either one FTP/HTTP/SMTP/POP3..... | 115 |
| 9.16. | Subscribe/Unsubscribe WIPsoft AT commands using WIPsoft Library API | 119 |
| 9.17. | Creating TCP client and server sockets in the same Wireless CPU at the same time mapping or unmapping the UART to exchange the data between the sockets | 120 |
| 10. | ERROR CODES | 121 |
| 10.1. | General CME Error Codes | 121 |
| 10.2. | GPRS CME Error Codes | 123 |

>> 1. Introduction



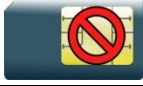




The aim of this document is to provide Sierra Wireless customers with a full description of the AT commands associated with the WIP feature.

1.1. Abbreviations

| Abbreviation | Definition |
|--------------|--|
| APN | Access Point Name |
| ASCII | American Standard Code for Information Interchange |
| AT | ATtention |
| BCC | Blind Carbon Copy |
| CC | Carbon Copy |
| CHAP | Challenge Handshake Authentication Protocol |
| CHV | Card Holder Verification |
| CID | Context IDentifier |
| CMUX | Converter Multiplexer |
| CPU | Central Processing Unit |
| DNS | Domain Name System |
| GGSN | Gateway GPRS Support Node |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile communication |
| HTTP | Hyper Text Transfer Protocol |
| IP | Internet Protocol |
| IPCP | Internet Protocol Control Protocol |
| M | Mandatory |
| MS | Mobile Station |
| N/A | Not Applicable |
| MSCHAP | MicroSoft Challenge Handshake Authentication |
| MSS | Maximum Segment Size |
| NU | Not Used |
| O | Optional |
| OS | Operating System |
| PAP | Password Authentication Protocol |
| PDP | Packet Data Protocol |
| PIN | Personal Identity Number |
| POP3 | Post Office Protocol |
| PPP | Point-to-Point Protocol |
| SIM | Subscriber Information Module |
| SMTP | Simple Mail Transfer Protocol |
| TCP | Transmission Control Protocol |
| TOS | Type Of Service |
| TTL | Time To Live |

| Abbreviation | Definition |
|--------------|---|
| UART | Universal Asynchronous Receiver Transmitter |
| UDP | User Data Protocol |
| URL | Uniform Resource Locator |
| WIP | Wavecom Internet Protocol |

1.2. Logos

| Logo | Definition |
|---|--|
|  | This picture indicates the +WIND indication from which the AT command is allowed. X values can be: 1, 3, 4, or 16. |
|  | This picture indicates that a SIM card must be inserted to support the AT command. |
|  | This picture indicates that an AT command is supported even if the SIM card is absent. |
|  | This picture indicates that the PIN 1 /CHV 1 code must be entered to support the AT command. |
|  | This picture indicates that an AT command is supported even if the PIN 1 /CHV 1 code is not entered. |
|  | This picture indicates that the PIN 2 /CHV 2 code must be entered to support the AT command. |
|  | This picture indicates that an AT command is supported even if the PIN 2/CHV 2 code is not entered. |

1.3. AT Commands Presentation Rules

The AT commands to be presented in the document are as follows:

- A "Description" section as Heading 3 provides general information on the AT command (or response) behavior.
- A "Syntax" section as Heading 3 describes the command and response syntaxes and all parameters description.
- A "Parameters and Defined Values" section as Heading 3 describes all parameters and values.
- A "Parameter Storage" as Heading 3 presents the command used to store the parameter value and/or the command used to restore the parameter default value.
- An "Examples" section as Heading 3 presents the real use of the described command.
- A "Note" section as Heading 3 can also be included indicating some remarks about the command use.

Figures are provided where necessary.

2. AT Command Syntax

This section describes the AT command format and the default value for their parameters.

2.1. Command Line

Commands always start by the standard prefix "AT+WIP" and end with the <CR> character. Optional parameters are shown in brackets [].

Example:

```
AT+WIPcmd=<Param1>[,<Param2>]
```

<Param2> is optional. When the AT+WIPcmd is executed without <Param2> the default value of <param2> is used.

2.2. Information Responses and Result Codes

Responses start and end with <CR><LF>, except for the ATV0 DCE response format and the ATQ1 (result code suppression) commands.

- If command syntax is incorrect, the "ERROR" string is returned.
- If command syntax is correct but transmitted with wrong parameters, the "+CME ERROR: <Err>" or "+CMS ERROR: <SmsErr>" strings is returned with adequate error codes if CMEE was previously set to 1. By default, CMEE is set to 0, and the error message is only "ERROR".
- If the command line has been executed successfully, an "OK" string is returned.

In some cases, such as "AT+CPIN?" or (unsolicited) incoming events, the product does not return the "OK" string as a response.

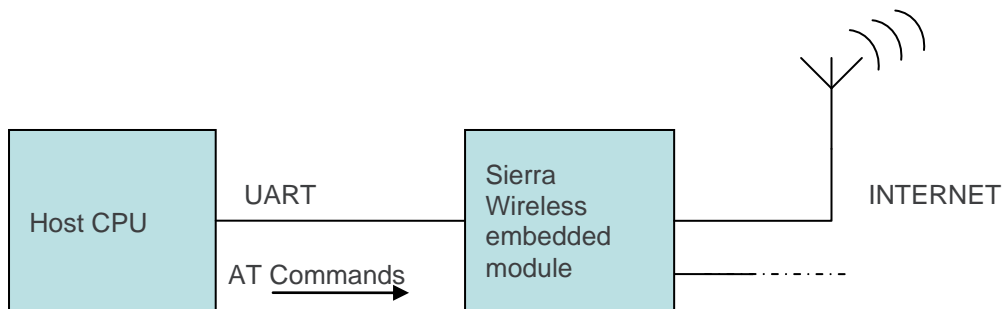
In the following examples <CR> and <CR><LF> are intentionally omitted.

>> 3. Principles

WIPsoft is an application that implements the TCP/IP protocols using custom AT commands. This application operates in co-operative mode and must be downloaded to the Sierra Wireless embedded module. The commands are sent from an external application and the corresponding responses are sent back from the module to the external application. The WIPsoft uses the APIs provided by WIPlib and provides custom AT command interface to the external application.

AT+WIP commands involve:

- a host computer, which issues AT+WIP commands
- Sierra Wireless intelligent embedded module
- the rest of the Internet / Intranet



Multiplexing: Several sockets can be operating at once. The +WIPDATA command allows to temporarily identify the UART in data mode with a given socket. The data written on UART is transferred through the socket. The data which arrives on the socket can be read from the UART.

In AT mode, the host receives an unsolicited event when the data arrives on the socket.

Multiple UARTs: There can be several UARTs simultaneously active at once, and different UARTs can map a different socket simultaneously. However, it is forbidden to map a single socket on several UARTs simultaneously.

3.1. Sockets Identification

Sockets are identified by a pair of numbers: the first one identifies the protocol; the second one identifies a given socket of this protocol.

3.1.1. Possible Protocols

The possible protocols are,

- 1 = UDP
- 2 = TCP in connect mode (Client)
- 3 = TCP in listen mode (Server)
- 4 = FTP
- 5 = HTTP
- 6 = SMTP
- 7 = POP3
- 8 = MMS

Two pairs with a different protocol number but the same index identify two distinct sockets.

Example: Both 1,7 and 2,7 are valid identifiers simultaneously; the former identifies a UDP socket and the later, a TCP connected socket.

3.1.2. Number of Sockets

The number of sockets per protocol is limited.

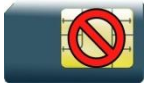
- UDP : 8 sockets
- TCP Clients : 8 sockets
- TCP Servers : 4 sockets

3.1.3. Notes

The creation of basic sockets (TCP/UDP) is not commercial but other features are locked by a commercial feature named "internet plug-in". The WIPsoft commands used for socket/session creation will return a "+CME ERROR: 839" error code if the feature is not enabled. To enable the features, you can refer to the Firmware AT Commands Interface Manual (especially the AT+WCFM command) and we recommend you to contact your Sierra Wireless distributor or sales point for further details.

4. General Configuration

4.1. IP Stack Handling +WIPCFG



4.1.1. Description

The +WIPCFG command is used for performing the following operations:

- start TCP/IP stack
- stop TCP/IP stack
- configuring TCP/IP stack
- displaying version information

4.1.2. Syntax

- if <mode>=0,1

Action Command

AT+WIPCFG=<mode>

OK

- if <mode>=2

Action Command

AT+WIPCFG=<mode>,<opt num>,<value>

OK

- if <mode>=3

Action Command

AT+WIPCFG=<mode>

WIPsoft vXX.YY.ZZ on Open AT OS vA.B

MMM-DDD-YYYY HH:MM:SS <WIPLib: version number> <WIPsoft: version number>

OK

- if <mode>=4

*Action Command***AT+WIPCFG=<mode>,<action>**

OK

*Read Command***AT+WIPCFG?**

+WIPCFG: <optnum>,<value>

[+WIPCFG: <optnum>,<value>[...]]

OK

*Test Command***AT+WIPCFG=?**

OK

4.1.3. Parameters and Defined Values

| | |
|-------------------------|--|
| <mode>: | requested operation |
| 0 | stop TCP/IP stack |
| 1 | start TCP/IP stack Note: If the Ethernet bearer support is defined, the Ethernet driver is subscribed on executing (+WIPCFG = 1) |
| 2 | configure TCP/IP stack |
| 3 | display TCP/IP application version |
| 4 | TCP/IP stack configuration management |
| <opt num>: | configuration option identifier |
| 0 | <p>WIP_NET_OPT_IP_TTL – Default TTL of outgoing data grams</p> <p>This option is a limit on the period of time or number of iterations or transmissions that a unit of data can experience before it should be discarded. The time to live (TTL) is an 8-bit field in the Internet Protocol (IP) header. It is the 9th octet of 20. The default value of this parameter is 64. Its value can be considered as an upper bound on the time that an IP datagram can exist in an internet system. The TTL field is set by the sender of the datagram, and reduced by every host on the route to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then the datagram is discarded. This is used to avoid a situation in which an undelivered datagram keeps circulating in the network.</p> <p>range: 0-255 (default value: 64)</p> |
| 1 | <p>WIP_NET_OPT_IP_TOS – Default TOS of outgoing parameters</p> <p>The IP protocol provides a facility for the Internet layer to know about the various tradeoffs that should be made for a particular packet. This is required because paths through the Internet vary widely in terms of the quality of service provided. This facility is defined as the "Type of Service" facility, abbreviated as the "TOS facility".</p> <p>The TOS facility is one of the features of the Type of Service octet in the IP datagram header. The Type of Service octet consists of following three fields:</p> <pre> 0 1 2 3 4 5 6 7 +---+---+---+---+---+---+---+---+ PRECEDENCE TOS MBZ +---+---+---+---+---+---+---+ </pre> <p>The first field is "PRECEDENCE". It is intended to denote the importance or priority of the datagram.</p> <p>The second field is "TOS" which denotes how the network should maintain the tradeoffs between throughput, delay, reliability, and cost.</p> <p>The last field is "MBZ" (Must Be Zero), is currently unused and is set to 0. The TOS field can have the following values:</p> <p>1000 -- minimize delay 0100 -- maximize throughput</p> |

| | |
|----|---|
| | <p>0010 -- maximize reliability</p> <p>0001 -- minimize monetary cost</p> <p>0000 -- normal service</p> <p>For more information on this field please refer to RFC1349.</p> <p>range: 0-255 (default value: 0)</p> |
| 2 | <p>WIP_NET_OPT_IP_FRAG_TIMEO - Time to live in seconds of incomplete fragments</p> <p>When a datagram's size is larger than the MTU (Maximum Transmission Unit) of the network, then the datagram is divided into smaller fragments. These divided fragments are sent separately. The "WIP_NET_OPT_IP_FRAG_TIMEO" option specifies the Time to live for these fragments.</p> <p>range: 1-65535 (default value: 60)</p> |
| 3 | <p>WIP_NET_OPT_TCP_MAXINITWIN – Number of segments of initial TCP window</p> <p>This option is used to specify the number of segments in the initial TCP window.</p> <p>A TCP window specifies the amount of outstanding (unacknowledged by the recipient) data a sender can send on a particular connection before it gets an acknowledgment back from the receiver. The primary reason for the window is congestion control.</p> <p>range: 0-65535 (default value: 0)</p> |
| 4 | <p>WIP_NET_OPT_TCP_MIN_MSS - Default MSS of off-link connections</p> <p>This option is used by the WIPLib Plug-In internally. This parameter specifies the maximum size of TCP segment which would be sent. By default, the value of this parameter is set to 536. Hence WIPLib Plug-In would not send any TCP segment having a length greater than 536 bytes without header.</p> <p>range: 536-1460 (default value: 536)</p> |
| 5 | <p>WIP_NET_OPT_DEBUG_PORT</p> <p>This option is used to specify the port on which the debug traces are to be sent.</p> <p>range: 0-3 (default value: 0)</p> |
| 12 | <p>AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE - Used for TCP sockets to configure the packet segmentation on IP network side</p> <p>This option is used to specify the maximum time to wait between two successive data chunks received from the mapped UART/serial port (please see +WIPDATA AT command). It allows the application to buffer a certain amount of data before writing on IP network side.</p> <p>Each unit in the range represents 100 msec. For example, value 10 for this option will give a wait time of 1sec (10*100msec).</p> <p>Default value for AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE option is 0. This value means that no specific process is done to avoid TCP packets segmentation: data are written onto IP network without any delay after the reception of data from the mapped UART/serial port (please see +WIPDATA AT command). In this case some TCP packets sent on the IP network may be smaller than TCP_MIN_MSS value.</p> <p>Setting e.g. a 10 value for this option will make the application to wait at least 1 second or twice the TCP_MIN_MSS value to be reached before sending data on IP network. In this case, TCP packets size sent on the IP network should be equal to at least TCP_MIN_MSS (Default value = 536 bytes).</p> <p>range: 0- 100 (default value: 0)</p> |
| 13 | <p>AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT : Used to configure whether a "+++" escape sequence should be sent as data to the peer. By default, this option is set to 0 which means that the "+++"sequence is sent to the peer as data. If set to 1, "+++"sequence is not sent as data to the peer.</p> <p>range: 0 -1(default value:0)</p> |

| | | |
|---|----|---|
| | 14 | AT_WIP_NET_OPT_AUTO_SWITCH - Used for TCP socket, to switch back automatically to AT command mode when the TCP connection is closed by peer entity 0: Does not switch automatically to AT mode 1: Switches automatically to AT mode range: 0-1 (default value:0) |
| | 15 | Set DATA offline (+WIPDATARW) RX and TX buffer size in bytes range : 1 – 32000 (default value 2048) |
| | 16 | WIP_NET_OPT_TCP_REXMT_MAX – Maximum timeout of TCP packets; the maximum time between TCP retransmissions range: 1 - 64 (default value:64 seconds) |
| | 17 | WIP_NET_OPT_TCP_REXMT_MAXCNT - Max number of TCP packet retransmissions range: 1 - 12 (default value:12) |
| <action>: | | requested operation on TCP/IP stack parameter management |
| | 0 | configuration storage (when existing) is freed |
| | 1 | stores the configuration parameters |
| <value>: | | value range for different configuration options |
| <XX.YY.ZZ >: | | WIPsoft release version |
| <A.B>: | | Open AT [®] OS release version |
| <MM-DD-YYYY>: | | date of built of WIPsoft application |
| <HH:MM:SS>: | | time of built of WIPsoft application |
| <WIPlib: version number>: | | WIPlib version |
| <WIPsoft: version number>: | | internally identifying WIPsoft version |

Note: (WIP_NET_OPT_SOCKET_MAX + 1) sockets are reserved when UDP sockets are created (and not for TCP sockets); one socket buffer is added to support/afford DNS accesses

Note: For <opt num> numbers 6 through 11, the AT+WIPS command must be issued. For complete details regarding AT+WIPS, please refer the Firmware AT Commands Interface Manual.

4.1.4. Parameter Storage

Only one IP stack configuration set can be saved into the FLASH memory.

- “AT+WIPCFG=4,1” is used to store the TCP/IP stack configuration parameters into the FLASH memory
- “AT+WIPCFG=4,0” is used to free the TCP/IP stack configuration storage

Executing “AT+WIPCFG=1” will apply default parameters when existing. Still it is possible to change option values at run time using “AT+WIPCFG=2,<optnum>,<optvalue>”.

4.1.5. Possible Errors

The possible error message is displayed only if “AT+CMEE=1” is activated else “ERROR” is displayed.

| “+CMEE” AT error code | Description |
|-----------------------|---|
| 800 | invalid option |
| 801 | invalid option value |
| 802 | not enough memory left |
| 820 | error writing configuration in FLASH memory |
| 821 | error freeing configuration in FLASH memory |
| 844 | stack already started |
| 850 | initialization failed |
| 852 | IP stack not initialized |

4.1.6. Examples

| Command | Responses |
|---|---|
| AT+WIPCFG=1 <i>Note: Start IP Stack</i> | OK |
| AT+WIPCFG? | +WIPCFG: 0,64 +WIPCFG: 1,0 +WIPCFG: 2,60 +WIPCFG: 3,0 +WIPCFG: 4,536 +WIPCFG: 5,0 +WIPCFG: 6,8 +WIPCFG: 7,32 +WIPCFG: 8,0 +WIPCFG: 9,0 +WIPCFG: 10,4 +WIPCFG: 11,4 +WIPCFG: 12,10 +WIPCFG: 13,0 +WIPCFG: 14,0 +WIPCFG: 15,2048 +WIPCFG: 16,64 +WIPCFG: 17,12 OK |
| AT+WIPCFG=2,0,10 <i>Note: Configure TTL of IP Stack</i> | OK |
| AT+WIPCFG? | +WIPCFG: 0,10 +WIPCFG: 1,0 +WIPCFG: 2,60 +WIPCFG: 3,0 +WIPCFG: 4,536 +WIPCFG: 5,0 +WIPCFG: 6,8 +WIPCFG: 7,32 +WIPCFG: 8,0 +WIPCFG: 9,0 +WIPCFG: 10,4 +WIPCFG: 11,4 +WIPCFG: 12,10 +WIPCFG: 13,0 +WIPCFG: 14,0 +WIPCFG: 15,2048 +WIPCFG: 16,64 +WIPCFG: 17,12 OK |

| Command | Responses |
|---|---|
| AT+WIPCFG=3 | WIPsoft v202 on Open AT OS v312 Mar 26 2007 11:45:46 WIPLib:v2a07 WIPsoft:v1a12 |
| <i>Note: Display software version</i> | OK |
| AT+WIPCFG=0 <i>Note: Stop the TCP/IP Stack</i> | OK |
| AT+WIPCFG=4 , 1 <i>Note: Store IP configuration parameters into FLASH</i> | OK |
| AT+WIPCFG=4 , 0 <i>Note: Free IP configuration parameters stored in FLASH</i> | OK |

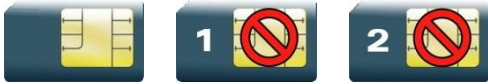
4.1.7. Notes

It is recommended to change the default settings of the WIP stack using +WIPCFG only when it is required. Changing the parameter values especially the max number of sockets and the max TCP buffer size with the high values lead to over consumption of the stack memory which causes the WIPsoft to crash. Hence, care must be taken when the default settings of the stack is changed using +WIPCFG command.

Following option values set by +WIPCFG command are taken into consideration at the run time. The below option values except for AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE and AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT will be taken into consideration at next start up only if these are saved in the flash before stopping the stack.

- WIP_NET_OPT_IP_TTL
- WIP_NET_OPT_IP_TOS
- WIP_NET_OPT_IP_FRAG_TIMEO
- WIP_NET_OPT_TCP_MAXINITWIN
- WIP_NET_OPT_TCP_MIN_MSS
- WIP_NET_OPT_DEBUG_PORT
- AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE
- AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT
- AT_WIP_NET_OPT_AUTO_SWITCH

4.2. Bearers Handling +WIPBR



4.2.1. Description

The +WIPBR command can be used to

- select the bearer
- start/close the bearer
- configure different bearer options such as access point name

4.2.2. Syntax

- if <cmdtype>=0,1 or 5

Action Command

```
AT+WIPBR=<cmdtype>,<bid>
```

OK

- if <cmdtype>=2

Action Command

```
AT+WIPBR=<cmdtype>,<bid>,<opt num>,<value>
```

OK

- if <cmdtype>=3

Action Command

```
AT+WIPBR=<cmdtype>,<bid>,<opt num>
```

```
+WIPBR: <bid>,<opt num>,<value>
```

OK

- if <cmdtype>=4

Action Command

```
AT+WIPBR=<cmdtype>,<bid>,<mode>[,<login>,<password>,[<caller  
identity>]]
```

OK

- if <cmdtype>=6

Action Command
AT+WIPBR=<cmdtype>,<bid>,<mode>
 OK

Read Command
AT+WIPBR?
 <bid>,<state>
 [<bid>,<state>[...]]
 OK

Test Command
AT+WIPBR=?
 OK

- if <mode>=1

Unsolicited response
 +WIPBR: <bid>,<status>,<local IP @>,<remote IP @>,<DNS1 @>,<DNS2 @>

4.2.3. Parameters and Defined Values

| | | |
|--------------------------|---|---------------------------------------|
| <cmd type>: | | type of command |
| | 0 | close bearer |
| | 1 | open bearer |
| | 2 | set value of different bearer options |
| | 3 | get value of different bearer options |
| | 4 | start bearer |
| | 5 | stop bearer |
| | 6 | bearer configuration management |
| <bid>: | | bearer Identifier |
| | 1 | UART1 |
| | 2 | UART2 |

| | | |
|-------------------------|--------|--|
| | 3 | N/A |
| | 4 | ETHER |
| | 5 | GSM |
| | 6 | GPRS |
| | 11..14 | CMUX port over UART1 |
| | 21..24 | CMUX port over UART2 |
| <opt num>: | | bearer option identifier |
| | 0 | WIP_BOPT_LOGIN – username (string) max: 64 characters |
| | 1 | WIP_BOPT_PASSWORD – password (string) max: 64 characters |
| | 2 | WIP_BOPT_DIAL_PHONENB – phone number (string) max: 32 characters |
| | 5 | WIP_BOPT_DIAL_RINGCOUNT - Number of rings to wait before sending the WIP_BEV_DIAL_CALL event range: 0-65535 |
| | 6 | WIP_BOPT_DIAL_MSNULLMODEM - Enable MS-Windows null-modem protocol ("CLIENT"/"SERVER" handshake) range: 0-1 |
| | 7 | WIP_BOPT_PPP_PAP - Allow PAP authentication range: 0-1 |
| | 8 | WIP_BOPT_PPP_CHAP - Allow CHAP authentication for the connection between the PC and the Wireless CPU (UART bearer) range: 0-1 |
| | 9 | WIP_BOPT_PPP_MSCHAP1 - Allow MSCHAPv1 authentication range: 0-1 |
| | 10 | WIP_BOPT_PPP_MSCHAP2 - Allow MSCHAPv2 authentication range: 0-1 |
| | 11 | WIP_BOPT_GPRS_APN - Address of GGSN (string) max: 96 characters |
| | 12 | WIP_BOPT_GPRS_CID - Cid of the PDP context range: 1-4 |
| | 13 | WIP_BOPT_GPRS_HEADERCOMP - Enable PDP header compression range: 0-1 |
| | 14 | WIP_BOPT_GPRS_DATACOMP - Enable PDP data compression range: 0-1 |
| | 15 | WIP_BOPT_IP_ADDR - Local IP address (IP/string) |

| | | |
|-------------------------------|----|--|
| | 16 | WIP_BOPT_IP_DST_ADDR - Destination IP address (IP/string) |
| | 17 | WIP_BOPT_IP_DNS1 - Address of primary DNS server (IP/string) |
| | 18 | WIP_BOPT_IP_DNS2 - Address of secondary DNS server (IP/string) |
| | 19 | WIP_BOPT_IP_SETDNS - Configure DNS resolver when connection is established range: 0-1 |
| | 20 | WIP_BOPT_IP_SETGW - Set interface as default gateway when connection is established range: 0-1 |
| | 21 | WIP_BOPT_GPRS_TIMEOUT - Define a time limit to connect GPRS bearer. For example, value 300 for this option sets a wait time of 30s (300*100ms). Note: If timer expires before GPRS bearer connects, error 847 is returned. range: 300-1200 (default: 1200). |
| | 22 | WIP_BOPT_DEBUG_PKT - Enable the debug traces of NET level 10 range: 0-1 |
| | 23 | WIP_BOPT_IP_DHCP - Enables auto-configuration of IP address and Netmask with DHCP range:0-1 |
| | 24 | WIP_BOPT_IP_MAC - Reads the MAC address, functioning as a read only option |
| | 25 | WIP_BOPT_IP_NETMASK - Sets the Network mask |
| | 26 | WIP_BOPT_IP_GW - Sets address of default gateway |
| | 27 | WIP_BOPT_RESTART - Automatically restart server after connection is terminated |
| | 28 | WIP_BOPT_GPRS_ERROR_REPORTING – report GPRS CME errors instead of WIPsoft generic error range:0-1 (default=0 for WIPsoft generic error) |
| <mac IP @*>: | | MAC address of Ethernet driver |
| <gateway IP @*>: | | default gateway address |
| <netmask IP @*>: | | network mask address |
| <value>: | | range of value for different bearer options |
| <mode>: | | mode of operation |
| | 0 | client |
| | 1 | server |
| <state>: | | current state of the bearer |
| | 0 | stopped |
| | 1 | started |

| | |
|---------------------------------|--|
| <status>: | result of the connection process |
| 0 | successful |
| any other value | to be matched to error code value (e.g. "814" means PPP authentication failure) |
| <local IP @*>: | local IP address |
| <remote IP @*>: | remote IP address. (first node in internet) |
| <DNS1 IP @*>: | Domain Name Server address |
| <DNS2 IP @*>: | Domain Name Server address |
| <login>: | PPP login |
| <passwd>: | PPP password |
| <caller identity>: | optional ASCII string (type ascii*). If not specified, then target will accept all DATA calls (independently of caller identification). If specified, then target will only accept calls from <caller identity>(which is the GSM data call number of the GSM client). |

* IP @ are displayed in alpha numeric dot format. e.g. 192.168.0.1...When no IP address is known, "0.0.0.0" is displayed.

Caution: *The options WIP_BOPT_IP_ADDR, WIP_BOPT_IP_DST_ADDR, WIP_BOPT_IP_DNS1 and WIP_BOPT_IP_DNS2 can be read after the bearer connection is established successfully. If an attempt is made to read the options value before the bearer connection is established successfully, incorrect IP address will be received.*

Caution: *The options WIP_BOPT_IP_NETMASK and WIP_BOPT_IP_GW can be read after the bearer connection is established successfully. If an attempt is made to read the options value before the bearer connection is established successfully, incorrect IP address will be received.*

Also the option WIP_BOPT_IP_MAC can be read after the bearer connection is open successfully. If an attempt is made to read the options value before the bearer connection is open, incorrect IP address will be received.

If the Ethernet bearer supported is defined, the MAC address is read from the Ethernet driver on opening the bearer(i.e., +WIPBR=4,1).

4.2.4. Parameter Storage

Several bearer configuration set can be saved.

Calling twice AT+WIPBR=6,<bid>,1 with the same <bid> will store the last configuration set.

- "AT+WIPBR=6,<bid>,1" is used to store the bearer configuration parameters set associated with the bearer <bid> into the FLASH memory.
- "AT+WIPBR=6,<bid>,0" is used to free the bearer configuration parameters set associated with the bearer <bid>.

Executing "AT+WIPBR=1,<bid>" will open bearer <bid> with default parameters of the bearer when existing.

4.2.5. Possible Errors

4.2.5.1. General CME Errors

The possible error message is displayed only if “AT+CMEE=1” is activated else “ERROR” is displayed.

| “+CMEE” AT error code | Description |
|-----------------------|--|
| 800 | invalid option |
| 801 | invalid option value |
| 802 | not enough memory left |
| 803 | operation not allowed in the current WIP stack state |
| 804 | device already open |
| 807 | bearer connection failure : line busy |
| 808 | bearer connection failure : no answer |
| 815 | bearer connection failure : PPP authentication failed |
| 816 | bearer connection failure : PPP IPCP negotiation failed |
| 820 | error writing configuration in FLASH memory |
| 821 | error freeing configuration in FLASH memory |
| 847 | bearer connection failure: WIP_BOPT_GPRS_TIMEOUT time limit expired before GPRS bearer connected |
| 848 | impossible to connect to the bearer |
| 849 | connection to the bearer has succeeded but a problem has occurred during the data flow establishment |

4.2.5.2. GPRS CME Errors

GPRS CME errors are listed in the table below.

| Error code | Meaning | Resulting from the following commands |
|------------|---|---------------------------------------|
| 103 | Incorrect MS identity.(#3) | +CGATT |
| 132 | service option not supported (#32) | +CGACT +CGDATA ATD*99 |
| 133 | requested service option not subscribed (#33) | +CGACT +CGDATA ATD*99 |
| 134 | service option temporarily out of order (#26, #34, #38) | +CGACT +CGDATA ATD*99 |
| 148 | unspecified GPRS error | All GPRS commands |
| 149 | PDP authentication failure (#29) | +CGACT +CGDATA ATD*99 |
| 150 | invalid mobile class | +CGCLASS +CGATT |

4.2.6. Examples

| Command | Responses |
|--|---|
| AT+WIPBR? | 1,0 6,1 OK <i>Note: Bearer UART1 is open but not started bearer GPRS is open and started</i> |
| AT+WIPBR? | OK <i>Note: No bearer has been opened yet</i> |
| AT+WIPBR=1,6 <i>Note: Open GPRS bearer</i> | OK |
| AT+WIPBR=2,6,11,"APN name" <i>Note: Set APN of GPRS bearer</i> | OK |
| AT+WIPBR=3,6,11 <i>Note: Get APN of GPRS bearer</i> | +WIPBR: 6,11,"APN name" OK |
| AT+WIPBR=2,6,21,600 <i>Note: set GPRS connection timeout value to 60s</i> | OK |
| AT+WIPBR=4,6,0 <i>Note: Start GPRS bearer</i> | OK |
| AT+WIPBR=5,6 <i>Note: Stop GPRS bearer</i> | OK |
| AT+WIPBR=0,6 <i>Note: Close GPRS bearer</i> | OK |
| AT+WIPBR=1,5 <i>Note: Open GSM bearer</i> | OK |
| AT+WIPBR=2,5,0,"login" <i>Note: Set the login for GSM bearer</i> | OK |
| AT+WIPBR=2,5,1,"password" <i>Note: Set the password for GSM bearer</i> | OK |
| AT+WIPBR=2,5,2,"phonenumber" <i>Note: Set the phone number for GSM bearer</i> | OK |
| AT+WIPBR=2,5,15,"1.1.1.1" <i>Note: Set the local IP address for GSM bearer</i> | OK |
| AT+WIPBR=2,5,16,"2.2.2.2" <i>Note: Set the destination IP address for GSM bearer</i> | OK |

| Command | Responses |
|---|---|
| AT+WIPBR=3,5,15 | +WIPBR: 5,15,"0.0.0.0" |
| | OK |
| <i>Note: Read the local IP address for GSM bearer</i> | <i>Note: Local IP address is not set as GSM bearer is still not connected</i> |
| AT+WIPBR=3,5,16 | +WIPBR: 5,16,"0.0.0.0" |
| | OK |
| <i>Note: Read the destination IP address for GSM bearer</i> | <i>Note: Destination IP address is not set as GSM bearer is still not connected</i> |
| AT+WIPBR=4,5,0 | OK |
| <i>Note: Start the GSM bearer as a client</i> | |
| AT+WIPBR=3,5,15 | +WIPBR: 5,15,"1.1.1.1" |
| <i>Note: Read the local IP for GSM bearer</i> | OK |
| AT+WIPBR=3,5,16 | +WIPBR: 5,16,"2.2.2.2" |
| <i>Note: Read the destination IP for GSM bearer</i> | OK |
| AT+WIPBR=5,5 | OK |
| <i>Note: Stop the GSM bearer</i> | |
| AT+WIPBR=0,5 | OK |
| <i>Note: Close the GSM bearer</i> | |
| AT+WIPBR=1,4 | OK |
| <i>Note: Opens the Ethernet bearer.</i> | |
| AT+WIPBR=4,4,0 | OK |
| <i>Note: Starts the Ethernet bearer in client mode.</i> | |
| AT+WIPBR=5,4 | OK |
| <i>Note: Stops the Ethernet bearer.</i> | |
| AT+WIPBR=0,4 | OK |
| <i>Note: Closes the Ethernet bearer.</i> | |
| AT+WIPBR=2,4,23,"1" | OK |
| <i>Note: Sets the DHCP to TRUE. Default: TRUE.</i> | |
| AT+WIPBR=3,4,24 | +WIPBR: 4,24,"1.1.1.1" |
| <i>Note: Reads the MAC address.</i> | OK |
| AT+WIPBR =2,4,25," <getway IP @*>" | OK |
| <i>Note: Sets the Default gateway address.</i> | |
| AT+WIPBR=2,4,26," <netmask IP @*>" | OK |
| <i>Note: Sets the Network mask address.</i> | |

| Command | Responses |
|---|--------------------------------------|
| AT+WIPBR=3,4,23 <i>Note: Reads the DHCP value.</i> | +WIPBR: 4,23,1 OK |
| AT+WIPBR=3,4,25 <i>Note: Reads the Default gateway address.</i> | +WIPBR: 4,25,"10.66.67.193" OK |
| AT+WIPBR=3,4,26 <i>Note: Reads the Network mask address.</i> | +WIPBR: 4,26,"255.255.255.192" OK |

4.2.7. Notes

4.2.7.1. For Starting a Bearer

The mandatory parameters to start a bearer in

- server mode: <cmdtype>, <bid>, <mode>, <login> and <password>
- client mode: <cmdtype>, <bid> and <mode>

Depending on the mode and the bearer type, additional parameters are required or forbidden:

| Bid | Mode | Other Parameters |
|-----------------|------|--|
| 1,3,11,14,21,24 | 0 | None |
| 1,3,11,14,21,24 | 1 | <PPP login>, <PPP password> |
| 5 | 0 | None |
| 5 | 1 | <login>,<password>[,<caller identity>] |
| 6 | 0 | None |

Starting bearer as a server requires additional parameters as mentioned in the above table.

- For PPP server, only parameters <login> and <password> are required. They will be compared with remote PPP client login and password.
- For GSM server, <login> and <password> will be used for PPP over GSM establishment (same behavior as described for PPP server).

The <caller identity> is an optional ASCII string (type ASCII*). If not specified, then target will accept all DATA calls (independently of caller identification). If specified, then target will only accept calls from <caller identity> (which is the GSM data call number of the GSM client).

Opening bearer only consists in associating the IP protocol stack with the specified bearer. The corresponding bearer setup has to be done through the adequate already existing AT commands (please refer to +WMFM commands for UART1 and UART2, +CMUX command for CMUX virtual ports and GSM/GPRS AT commands).

Several bearers can be opened at the same time but only one bearer can be started at a time.

If both DNS1 and DNS2 are displayed as "0.0.0.0" in the unsolicited message when bearer is opened in server mode, it means that connecting to a remote IP host through an URL will fail.

The options WIP_BOPT_DIAL_REDIALCOUNT and WIP_BOPT_DIAL_REDIALDELAY will not be implemented through AT commands. Nevertheless, for future compatibility reason, Opt num 3 and 4 are kept as reserved.

For GSM bearer, the options WIP_BOPT_IP_ADDR and WIP_BOPT_IP_DST_ADDR will display valid addresses only when the bearer is started and connected, else it will display an address "0.0.0.0".

The Ethernet bearer can be started only in client mode.

>> 5. IP Protocol Services

5.1. Service Creation +WIPCREATE

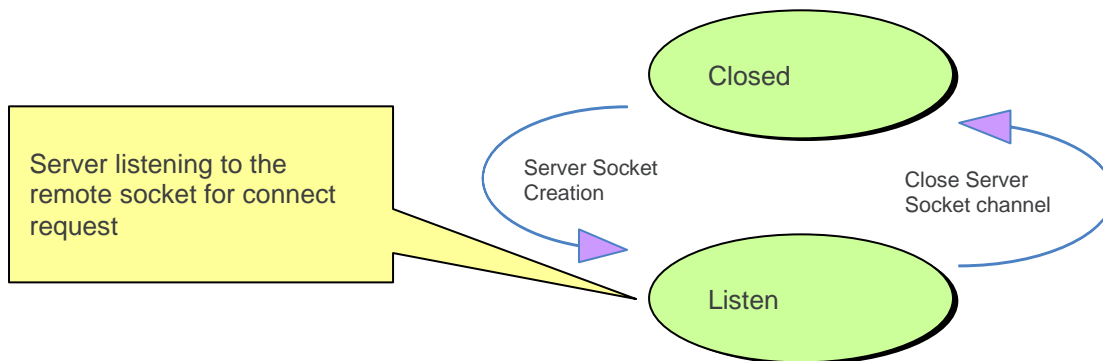


5.1.1. Description

The +WIPCREATE command is used to create UDP, TCP client and TCP server sockets associated with the specified index and FTP/HTTP/SMTP/POP3 service. Only one FTP/HTTP/SMTP/POP3/MMS session at a time is available.

If a local port is specified while creating a socket, the created socket will be assigned to this port; if not, a port will be assigned dynamically by WIP application. If peer IP and peer port is specified, the created socket will be connected to the specified IP and port.

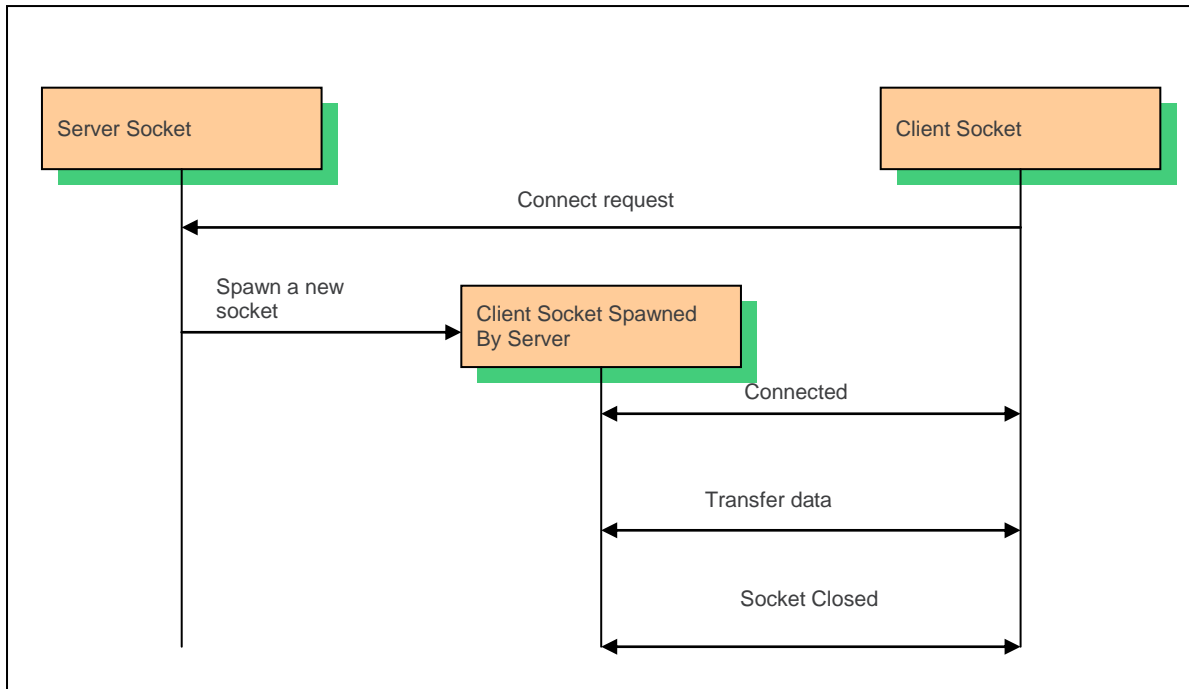
TCP server cannot be used to transfer data. To transfer data, it creates a local TCP client socket. This process of creating local socket is referred as “spawning”. When a server socket is created using, socket passively listens on a specified port for incoming connections. The below mentioned diagram shows different states managed for TCP server.



On reception of a connection request from a remote client socket, a server socket does the following,

- spawns a new socket (client) to connect to the remote socket
- data transfer is done between the spawned socket and the remote socket
- server socket remains in the listening mode and is ready to accept the request from other clients

Below mentioned diagram shows connection establishment procedure.



5.1.2. Syntax

- if <mode>=1

```

Action Command
AT+WIPCREATE=<mode>,<communication index>,<local port> [,<peer IP>,<peer port>]
OK
    
```

- if <mode>=2

```

Action Command
AT+WIPCREATE=<mode>,<communication index>,<peer IP>,<peer port>
OK
    
```

- if <mode>=3

Action Command

AT+WIPCREATE=<mode>,<server index>,<local port>,<from idx>,<to idx>

OK

- if <mode>=4

Action Command

AT+WIPCREATE=<mode>,<index>,<server>[,<peer_port>],<username>,<password>[,<account>]

OK

- if <mode>=5

Action Command

AT+WIPCREATE=<mode>,<index>,[<server>[,<peer port>]][, <username>,<password>][,<header list>[...]]

OK

- if <mode>=6 or 7

Action Command

AT+WIPCREATE=<mode>,<index>,<server>[,<peer port>][, <username>,<password>]

OK

- if <mode>=8

Action Command

AT+WIPCREATE=<mode>,<index>,<server>,<peer port>,<url>

OK

Read Command

AT+WIPCREATE?

NONE

Test Command
AT+WIPCREATE=?
 OK

- if <mode>=1 or 2

Unsolicited response
 +WIPREADY: <mode>,<communication index>

- if <mode>=3

Unsolicited response
 +WIPACCEPT: <server index>,<communication idx>

- if <mode>=5, 6 or 7

Unsolicited response
 +WIPREADY: <mode>,<index>

5.1.3. Parameters and Defined Values

| | |
|----------------------------|---|
| <mode>: | specifies type of socket |
| | 1 UDP |
| | 2 TCP Client |
| | 3 TCP server |
| | 4 FTP |
| | 5 HTTP Client |
| | 6 SMTP Client |
| | 7 POP3 Client |
| | 8 MMS Client |
| <index>: | TCP/UDP/FTP/HTTP/SMTP/POP3/MMS session identifier |
| <local port>: | local TCP/UDP port |
| <peer IP>: | peer IP address; a string between quotes indicating an address either in numeric form (e.g. "85.12.133.10") or as a DNS entry (e.g. "www.sierrawireless.com") |

| | |
|-------------------------------------|--|
| <peer port>: | peer port or the server port For TCP/UDP, this parameter is the port of the peer socket. For FTP,HTTP,SMTP, POP3, and MMS, this parameter is the server port range: 1-65535 (default value for FTP: 21 default value for HTTP: 80 default value for SMTP: 25 default value for POP3: 110 default value for MMS: 8080) |
| <from idx>: | minimum index for spawned TCP sockets range: 1-8 |
| <server index>: | TCP server socket identifier range: 1-4 |
| <to idx>: | maximum index for spawned TCP sockets range: 1-8 |
| <communication index>: | indexes reserved for spawned sockets It cannot be used by other sockets even if the spawned sockets are not created yet. range: 1-8 |
| <server>: | server address or proxy address This parameter is the server address for FTP, SMTP and POP3 protocol and for HTTP it is proxy server address. It can either be a 32 bit number in dotted-decimal notation ("xxx.xxx.xxx.xxx") or an alpha numeric string format for hostname. |
| <user name>: | username for the authentication in string format Authentication is disabled when this parameter is not specified for HTTP, SMTP and POP3. |
| <password>: | password for the authentication in string format Authentication is disabled when this parameter is not specified for HTTP, SMTP and POP3. |
| <account>: | account information of the user in string format This is required by some FTP server during authentication phases. |
| <header list>: | HTTP header message (name-value pair) The first string in the message header field is the name of the header and the second string is the value of the header. |
| <url>: | URL of the MMS server This is an alphanumeric string format for hostname starting with "http://". |
| <...> | additional HTTP message header fields more pairs(name, value) of HTTP message header field can be added |

5.1.4. Parameter Storage

None

5.1.5. Possible Errors

| "+CMEE" AT error code | Description |
|-----------------------|---|
| 3 | operation not allowed |
| 800 | invalid option |
| 803 | operation not allowed in the current WIP stack state |
| 830 | bad index |
| 832 | bad port number |
| 834 | not implemented |
| 836 | memory allocation error |
| 837 | bad protocol |
| 839 | error during channel creation |
| 840 | UDP/TCP socket or FTP/HTTP/SMTP/POP3 session is already active |
| 842 | destination host unreachable (whether host unreachable, Network unreachable, response timeout) |
| 845 | attempt is made to reserve/create a client socket which is already reserved/opened by TCP server/client |
| 851 | incorrect number of parameters submitted |
| 860 | protocol undefined or internal error |
| 861 | user name rejected by server |
| 862 | password rejected by server |
| 865 | authentication error |
| 866 | server not ready error |

5.1.6. Examples

| Command | Responses |
|--|--|
| AT+WIPCREATE=1,1,80 | OK |
| <i>Note: Create the UDP socket on local port 80 with communication index = 1 ⇔ embedded module acts as an UDP server awaiting for incoming datagram on local port 80</i> | <i>Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for usage</i> |
| AT+WIPCREATE=1,1,"www.sierrawireless.com",80 | OK |

| Command | Responses |
|--|--|
| <i>Note: Create the UDP socket on arbitrary free local port with peer IP and peer port 80 with communication index = 1 ⇔ embedded module acts as a UDP client that can send datagram towards the remote entity</i> | <i>Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for usage</i> |
| AT+WIPCREATE=1,1,80,"www.sierrawireless.com",80 | OK |
| <i>Note: Create the UDP socket on local port 80 with peer IP and peer port 80 with communication index = 1 ⇔ embedded module acts as a UDP client and an UDP server : it can send datagram towards the remote entity and receiving datagram on the specified local port.</i> | <i>Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for usage</i> |
| AT+WIPCREATE=3,1,80,5,8 | OK |
| <i>Note: Create the TCP server on port 80 with server index=1 ⇔ embedded module acts as a TCP server : it will from now on spawn TCP client socket from communication index 5 to 8</i> | <i>Note: An unsolicited event +WIPACCEPT: 1,5 will be received once the TCP server is ready for usage</i> |
| AT+WIPCREATE=2,1,"IP ADDR",80 | OK |
| <i>Note: Create the TCP client on port 80 with index=1 ⇔ embedded module acts as a TCP client : it can from now on communicate with the remote specified entity through communication index 1</i> | <i>Note: An unsolicited event +WIPREADY: 2,1 will be received once the TCP client is ready for usage</i> |
| AT+WIPCREATE=4,1,"ftp.wavecom.com","admin","123456" <i>Note: Create a FTP session ⇔ towards the remote specified FTP server. Communication index to be used then is 1</i> | OK |
| AT+WIPCREATE=5,1,"proxyaddress", ,"user name","password","User-Agent", "WIP-HTTP-Client/1.0" | OK +WIPREADY: 5, 1 |
| | <i>Note: HTTP session with proxy and 1 message header field</i> <i>Use default 80 proxy port number</i> <i>1 message header field:</i> <i>Message header field name is "User-Agent"</i> <i>Message header field value is "WIP-HTTP-Client/1.0"</i> |
| AT+WIPCREATE=5,1,"proxyaddress", ,"user name","password","User-Agent", "WIP-HTTP-Client/1.0","Accept-Encoding", "gzip","Accept-Language", "en-US" | OK +WIPREADY: 5, 1 |

| Command | Responses |
|--|--|
| | <p><i>Note: HTTP session with proxy and 3 message header fields</i></p> <p><i>Use default 80 proxy port number</i></p> <p><i>3 message header fields:</i></p> <p><i>Message header field name is "User-Agent" and header field value is "WIP-HTTP-Client/1.0"</i></p> <p><i>Message header field name is "Accept-Encoding" and header field value is "gzip"</i></p> <p><i>Message header field name is "Accept-Language" and header field value is "en-US"</i></p> |
| AT+WIPCREATE=5,1,"proxyaddress", ,"user","pass" | <p>OK</p> <p>+WIPREADY: 5, 1</p> |
| | <i>Note: Authentication connection on default proxy server port 80</i> |
| AT+WIPCREATE=6,1,"smtp.mail.yaho o.fr","587","user","pass" | <p>OK</p> <p>+WIPREADY: 6, 1</p> |
| | <i>Note: Connect to SMTP server port 587 with given username and password</i> |
| AT+WIPCREATE=7,1,"192.168.1.4", 110","user","pass" | <p>OK</p> <p>+WIPREADY: 7, 1</p> |
| | <i>Note: Connect to POP3 server port 110 with given username and password</i> |
| AT+WIPCREATE=7,1, "pop.mail.server.com" | <p>OK</p> <p>+WIPREADY: 7, 1</p> |
| | <i>Note: Connect to the default port 110 of POP3 server, with no authentication required</i> |
| AT+WIPCREATE=8,1, "192.168.10.200",8080,"http://mm s.orange.fr" | <p>OK</p> <p>+WIPREADY: 8, 1</p> |
| | <i>Note: Connect to the MMS server, with no authentication required</i> |

5.1.7. Notes

The maximum number of sockets can be set to 23 so that WIPsoft can handle in the same time either one FTP session (in passive mode)/HTTP/SMTP/POP3, 8 UDP sockets, 8 TCP client sockets and 4 TCP servers.

Starting a TCP server requires to specify the maximum number of communication sockets that can be spawned. This can be done using <from idx> and <to idx> parameters. Note that the value set for <to idx> should be equal or more than <from idx>.

The maximum communication socket that can be created using WIPsoft is 8. Hence, the range for <communication index> and <from idx>, <to idx> is 1-8. Note that the spawned communication socket and the TCP client socket share the same communication index.

It is not possible to create a client socket with AT+WIPCREATE=2, x, y, z when x is already reserved by a server with AT+WIPCREATE=3,<server idx>, <local port>,a,b where $a \leq x \leq b$. Similarly, it is not possible to reserve a range with AT+WIPCREATE=3, <server idx>, <local port>, a, b if one of the TCP client socket indexes between a and b is already reserved, be it by a client or a server range

The <from idx> and <to idx> are reserved for the server socket till the server socket and the spawned sockets are closed explicitly. So when trying to create a new TCP server socket, the <from idx> and <to idx> should be different from what was used earlier. A parameter used as <from_idx> can't be used as <to_idx> anymore for other TCP server socket creation until spawned sockets with specified <from_idx> and <to_idx> are closed along with the TCP server socket explicitly and vice versa.

When no more communication index is available in the TCP server's range (or no more resources to accept new incoming connections), any peer trying to connect to the server will receive an accept () immediately followed by a shutdown () ("peer close").

It is possible to have a TCP client and TCP server sockets running at the same time in the same Wireless CPU. In this scenario, when the connection is established between the TCP server and TCP client sockets, it is necessary to unmap the mapped socket on one index in order to send/receive data on socket which is created on another index. It is possible to use CMUX logical ports and can have an interface connection (like UART connection) for each socket for e.g. TCP client socket on one logical port and TCP server socket on another. In this case, it is not necessary to map or unmap the UART connections to send or receive the data from the socket.

The +WIPCREATE command causes the connection and authentication to the FTP server. If several file uploads and retrievals are required to/from the same server, a single connection with +WIPCREATE is needed. Then, each file operation will be done (one +WIPFILE command per operation), and the FTP connection will be released with +WIPCLOSE.

SIM card is required only if FTP session is established through GSM or GPRS. An FTP session upon an UART will work without a SIM card.

5.2. Closing a Service +WIPCLOSE



5.2.1. Description

The +WIPCLOSE command is used to close a socket or FTP/HTTP/SMTP/POP3/MMS session. When one serial port (UART or CMUX DLCI) is used to map a socket for read/write operations, [ETX] character can also be used to close the socket.

An unsolicited event is generated, when socket or FTP/HTTP/SMTP/POP3/MMS session is closed.

5.2.2. Syntax

Action command
AT+WIPCLOSE=<protocol>,<idx>
 OK

Read Command
AT+WIPCLOSE?
 NONE

Test Command
AT+WIPCLOSE=?
 OK

Unsolicited response
 +WIPPEERCLOSE: <protocol>,<idx>

5.2.3. Parameters and Defined Values

| | |
|--------------------------|---------------|
| <protocol>: | protocol type |
| 1 | UDP |
| 2 | TCP client |
| 3 | TCP server |
| 4 | FTP |

| | | |
|---------------------|--|------|
| | 5 | HTTP |
| | 6 | SMTP |
| | 7 | POP3 |
| | 8 | MMS |
| <idx>: | socket identifier or FTP/HTTP/SMTP/POP3 session identifier This parameter is the index of the socket or FTP/HTTP/SMTP/POP3 session created with +WIPCREATE command. | |

5.2.4. Parameter Storage

None

5.2.5. Possible Errors

| "+CMEE" AT error code | Description |
|------------------------------|--|
| 802 | not enough memory |
| 803 | operation not allowed in the current WIP stack state |
| 830 | bad index |
| 831 | bad state |
| 834 | not implemented |
| 837 | bad protocol |

5.2.6. Examples

| Command | Responses |
|--|---|
| AT+WIPCLOSE=1,1 | OK |
| <i>Note: Close UDP socket with communication index 1</i> | |
| AT+WIPCLOSE=2,1 | OK |
| <i>Note: Close TCP client with communication index 1</i> | |
| AT+WIPCLOSE=3,1 | OK |
| <i>Note: Close TCP server with communication index 1</i> | |
| AT+WIPCLOSE=4,1 | OK |
| <i>Note: Close FTP session with index 1</i> | <i>Note: An unsolicited event +WIPPEERCLOSE: 4,1 is received once the FTP session is closed</i> |
| AT+WIPCLOSE=5,1 | OK |
| <i>Note: Close HTTP session with index 1</i> | |

| Command | Responses |
|--|-----------|
| AT+WIPCLOSE=6,1 | OK |
| <i>Note: Close SMTP session with index 1</i> | |
| AT+WIPCLOSE=7,1 | OK |
| <i>Note: Close POP3 session with index 1</i> | |
| AT+WIPCLOSE=8,1 | OK |
| <i>Note: Close MMS session with index 1</i> | |

5.2.7. Notes

After issuing +WIPCLOSE command, no more data can be sent and received over the socket/session. In case of FTP protocol, the closure of FTP session is indicated by +WIPEERCLOSE unsolicited response when +WIPCLOSE command is used for closing the session.

In case of TCP and UDP sockets, response "OK" is returned when the +WIPCLOSE command is executed irrespective of whether the socket is active or not. But in case of FTP/HTTP/SMTP/POP3 session, "OK" response is returned if +WIPCLOSE command is executed when the session is active else "+CME ERROR: 831" error code is returned.

5.3. Service Option Handling +WIPOPT



5.3.1. Description

The +WIPOPT command is used to read and/or to configure different parameters on sockets and FTP/HTTP/SMTP/POP3/MMS service.

5.3.2. Syntax

- if <action>=1

Action Command

```
AT+WIPOPT=<protocol>,<idx>,<action>,<optnum>
```

OK

- if <action>=2 and <protocol> does not equal 8

Action Command

```
AT+WIPOPT=<protocol>,<idx>,<action>,<optnum>,<optval>
```

OK

- if <action>=2 and <protocol>=8

Action Command

```
AT+WIPOPT=<protocol>,<idx>,<action>,<optnum>,<optval>  
[,<optval2>,[<optval3>],[<optval4>]]
```

OK

Read Command

```
AT+WIPOPT?
```

NONE

Test Command

```
AT+WIPOPT=?
```

OK

- if <action>=1

Unsolicited response

```
+WIPOPT: <protocol>, <optnum>, <optval>
```

if <action>=1 and <protocol>=5 and <optnum>=54

Unsolicited response

```
+WIPOPT: 5, 54, <message header field name>, <message header field value>, [...]
```

5.3.3. Parameters and Defined Values

| | |
|--------------------------|---|
| <protocol>: | protocol type |
| 1 | UDP |
| 2 | TCP client |
| 3 | TCP server |
| 4 | FTP |
| 5 | HTTP |
| 6 | SMTP |
| 7 | POP3 |
| 8 | MMS |
| <idx>: | socket or FTP/HTTP/SMTP/POP3/MMS session identifier |
| <action>: | requested operation |
| 1 | read the value of an option |
| 2 | write the value of an option |
| <optnum>: | option that can be read/written |
| <optval>: | value of an option |
| <optval2>: | value of an extra option (optional) |
| <optval3>: | value of an extra option (optional) |
| <optval4>: | value of an extra option (optional) |

5.3.4. Parameter Storage

None

5.3.5. Possible Errors

| “+CMEE” AT error code | Description |
|------------------------------|--|
| 800 | invalid option |
| 801 | invalid option value |
| 803 | operation not allowed in the current WIP stack state |
| 830 | bad index |
| 834 | not implemented |
| 835 | option not supported |
| 837 | bad protocol |
| 850 | invalid channel option or parameter value (for example, HTTP user name too long) |
| 860 | protocol undefined or internal error |
| 863 | protocol delete error |
| 864 | protocol list error |

5.3.6. Examples

| Command | Responses |
|--|-----------------------|
| AT+WIPOPT=2,1,2,8,20 <i>Note: Set TTL for TCP client</i> | OK |
| AT+WIPOPT=2,1,1,8 <i>Note: Get TTL for TCP client</i> | +WIPOPT: 2,8,20 OK |
| AT+WIPOPT=3,1,2,9,10 <i>Note: Set TOS for TCP server</i> | OK |
| AT+WIPOPT=3,1,1,9 <i>Note: Get TOS for TCP server</i> | +WIPOPT: 3,9,10 OK |
| AT+WIPOPT=1,1,1,1 <i>Note: Get peer port for UDP</i> | +WIPOPT: 1,1,80 OK |
| AT+WIPOPT=4,1,2,40,1 <i>Note: Set data representation type for FTP</i> | OK |
| AT+WIPOPT=4,1,1,40 <i>Note: Get data representation type for FTP</i> | +WIPOPT: 4,1,1 OK |
| AT+WIPOPT=5,1,2,52,0 <i>Note: Set HTTP version to 1.0</i> | OK |
| AT+WIPOPT=5,1,2,53,6 <i>Note: Set maxredirect to 6</i> | OK |

| Command | Responses |
|--|---|
| AT+WIPOPT=5,1,1,52 | +WIPOPT: 5,52,0 OK |
| <i>Note: Get HTTP version</i> | |
| AT+WIPOPT=6,1,2,61,"senderaddresses@mail.com" | OK |
| <i>Note: Set the sender address</i> | |
| AT+WIPOPT=6,1,2,67,0 | OK |
| <i>Note: The application will format the mail header and send it during the data sending phase</i> | |
| AT+WIPOPT=6,1,1,61 | +WIPOPT: 6,61,"senderaddress@mail.com" OK |
| <i>Note: Get the sender address</i> | |
| AT+WIPOPT=6,1,1,60 | +WIPOPT:6,60,220,"220 innosoft.com SMTP service ready" OK |
| <i>Note: Get last protocol error / status</i> | |
| AT+WIPOPT=6,1,1,66 | +WIPOPT: 6,66,"My mail subject" OK |
| <i>Note: Get the set mail subject</i> | |
| AT+WIPOPT=7,1,1,72 | +WIPOPT: 7,72,243000 OK |
| <i>Note: Get total mail size</i> | |
| AT+WIPOPT=7,1,1,73 | +WIPOPT: 7,73,"1,1024" +WIPOPT: 7,73,"2,5237" +WIPOPT: 7,73,"3,128" +WIPOPT: 7,73,"4,36400" +WIPOPT: 7,73,"5,356" OK |
| <i>Note: Get mail listing</i> | |
| AT+WIPOPT=7,1,2,74,10 | +WIPOPT: 7,74,10 OK |

| Command | Responses |
|---|---|
| <i>Note: Delete mail ID 10</i> | |
| AT+WIPOPT=8,1,2,82,"Mr Smith <mr.smith@example.com" | +WIPOPT: 8,75, "Mr Smith <mr.smith@example.com" OK |
| <i>Note: Add an email address to the To-field of an MMS.</i> | |

5.3.7. Notes

It is possible to change and retrieve option value using +WIPOPT command only when the socket/session (given by <idx>) is active else it returns error.

5.3.7.1. Options that can be applied to UDP, TCP Client, TCP Server Sockets

| opt num | Value format | Option Type | Description | UDP | TCP client | TCP server |
|---------|--------------|-----------------------|---|-----|------------|------------|
| 0 | 0-65535 | WIP_COPT_PORT | Port of the socket | R | R | R |
| 1 | 0-65535 | WIP_COPT_PEER_PORT | Port of the peer socket | R | R | - |
| 2 | string | WIP_COPT_PEER_STRADDR | Address of the peer socket | R | R | - |
| 3 | 0-1 | WIP_COPT_BOUND | Specifies whether the socket is bounded2 to a peer socket or not default: 1 | R | - | - |
| 4 | 1-5839 | WIP_COPT_SND_LOWAT | Minimum amount of available space that must be available in the emission buffer before triggering a WIP_CEV_WRITE event default: 1024 | - | RW | RW |
| 6 | 0-65535 | WIP_COPT_NREAD | Number of bytes that can currently be read on that socket default: 0 | R | R | - |

| opt num | Value format | Option Type | Description | UDP | TCP client | TCP server |
|---------|--------------|------------------|---|-----|------------|------------|
| 7 | 0-1 | WIP_COPT_NODELAY | When set to TRUE, TCP packets are sent immediately, even if the buffer is not full enough. When set to FALSE, the packets will be sent either, a) by combining several small packets into a bigger packet b) when the data is ready to send and the stack is idle. default: 0 | - | RW | RW |
| 8 | 0-255 | WIP_COPT_TTL | Time-to-leave for packets default: 64 | RW | RW | RW |
| 9 | 0-255 | WIP_COPT_TOS | Type of service default: 0 | RW | RW | RW |

2 The option WIP_COPT_BOUND is used to check whether an UDP socket is bound to any other UDP socket or not. When the UDP socket is created without specifying the IP address of the peer, then the option WIP_COPT_BOUND will be read as FALSE. This is because there is no destination IP address to communicate with. If the UDP socket is created by specifying the peer IP address, the option WIP_COPT_BOUND will be read as TRUE. This is because the peer IP address will be resolved by the DNS and the socket is said to be bounded to the peer socket. Hence this option will be read as TRUE.

5.3.7.2. Options that can be applied to FTP Session

| opt num | Value format | Value type | Description |
|---------|--------------|------------|--|
| 40 | 0-1 | boolean | data representation type. 0: ASCII 1: binary default: 0 |
| 41 | 0-1 | boolean | FTP mode. 0: active 1: passive default: 1 |

5.3.7.3. Options that can be applied to HTTP Session

| opt num | Value format | Value type | Option type | Description | Type |
|---------|--------------|------------|----------------------|---|------|
| 50 | | u32 | WIP_COPT_RCV_BUFSIZE | set the size of the TCP socket receive buffer default: 0 | RW |

| opt num | Value format | Value type | Option type | Description | Type |
|---------|--------------|--------------|---|---|------|
| 51 | | u32 | WIP_COPT_SND_BUFSIZE | set the size of the TCP socket send buffer. default: 0 | RW |
| 52 | 0-1 | u8 | WIP_COPT_HTTP_VERSION 0: HTTP 1.0 1: HTTP 1.1 | define the HTTP version to be used by the session default: 1 | RW |
| 53 | | u32 | WIP_COPT_HTTP_MAXREDIRECT | set the maximum number of allowed redirects a zero value disables automatic redirects default: 8 | W |
| 54 | | <ascii list> | WIP_COPT_HTTP_HEADER | return the HTTP message header field (or a list of message header fields) from the last WIPFILE call default: depends on the HTTP server | R |

Caution: Option 54(WIP_COPT_HTTP_HEADER) is not implemented and hence attempt to read this option will result in +CME ERROR: 834.

5.3.7.4. Options that can be applied to SMTP Session

| opt num | Value format | Value type | Option type | Description | Type |
|---------|--------------|------------|-----------------------------|---|------|
| 60 | digit/string | u32/ascii | WIP_COPT_SMTP_STATUS_CODE | get last protocol error code and associated error string default: NULL string | R |
| 61 | string | ascii | WIP_COPT_SMTP_SENDER | set the sender address default: NULL string | RW |
| 62 | string | ascii | WIP_COPT_SMTP_SENDERNAME | set the sender name default: NULL string | RW |
| 63 | string | ascii | WIP_COPT_SMTP_REC | set the recipients list default: NULL string | RW |
| 64 | string | ascii | WIP_COPT_SMTP_CC_REC | set the CC recipients list default: NULL string | RW |
| 65 | string | ascii | WIP_COPT_SMTP_BCC_REC | set the BCC recipients list default: NULL string | RW |
| 66 | string | ascii | WIP_COPT_SMTP_SUBJ | set the mail subject default: NULL string | RW |
| 67 | digit | u32 | WIP_COPT_SMTP_FORMAT_HEADER | decide if the SMTP library will format the mail header or if the application is in charge of formatting it 0: Application formats mail header 1: SMTP lib formats mail header default: 1 | RW |

Caution: When option `WIP_COPT_SMTP_FORMAT_HEADER` is set to 0, application can format the mail header to attach documents (see RFC 2822 for Standard for the Format of ARPA Internet Text Messages for formatting details). Note that `+WIPFILE` command is used to send both mail header and body.

Caution: When option `WIP_COPT_SMTP_STATUS_CODE` is used to retrieve the error code and the associated error string for the SMTP session creation, it will not return any error code and error string if no error occurred during that particular SMTP session creation. For example, After the SMTP session is created successfully, an attempt to retrieve the error code and the associated error string, using the option `WIP_COPT_SMTP_STATUS_CODE`, will result in an error code "0" and the error string corresponding to the successful case. Create a SMTP session for the second time which will result in the "+CME ERROR: 840" error code because the session is already active. Now an attempt to retrieve the error code along with the associated error string, using the option `WIP_COPT_SMTP_STATUS_CODE`, will result in error code "0" and the associated error string because the first SMTP session was successful.

5.3.7.5. Options that can be applied to POP3 Session

| opt num | Value format | Value type | Option type | Description | Type |
|---------|--------------|------------|---------------------------|--|------|
| 70 | digit/string | u32/ascii | WIP_COPT_POP3_STATUS_CODE | get last protocol error code and associated error string | R |
| 71 | | u32 | WIP_COPT_POP3_NB_MAILS | get total number of mails default: depends on the mails available in the mail box | R |
| 72 | | u32 | WIP_COPT_POP3_MAILSIZE | get total mail size default: depends on the mails available in the mail box | R |
| 73 | digit/string | ascii | not a POP3 wip option | get mail listing The return value is a list of strings containing mail ID and mail size information. default: depends on the mails available in the mail box | R |
| 74 | | u32 | not a POP3 wip option | delete the mail ID The mail ID corresponds to the mail ID returned by the mail listing option. default: depends on the mails available in the mail box | W |

Caution: When option `WIP_COPT_POP3_STATUS_CODE` is used to retrieve the error code and the associated error string for the POP3 session creation, it will not return any error code and error string if no error occurred during that particular POP3 session creation.

For example, after the SMTP session is created successfully, an attempt to retrieve the error code and the associated error string, using the option `WIP_COPT_POP3_STATUS_CODE`, will result in an error code "0" and the error string corresponding to the successful case. Create a POP3 session for the second time which will result in the "+CME ERROR: 840" error code because the session is already active. Now an attempt to retrieve the error code along with the associated error string, using the option `WIP_COPT_POP3_STATUS_CODE`, will result in error code "0" and the associated error string because the first POP3 session was successful

5.3.7.6. Options that can be applied to MMS sessions

| opt num | Value format | Value type | Option type | Description | Type |
|---------|---------------------------|------------|---------------------------|---|------|
| 80 | u32 | u32 | WIP_MMS_DATE | Set the value of the date and time of the MMS in the following format : <i>Month/Day/Year Hour:Min:Sec</i> | RW |
| 81 | string | ascii | WIP_MMS_TO_PHONE* | Adds a telephone number to the TO field in the MMS | RW |
| 82 | string | ascii | WIP_MMS_TO_EMAIL* | Adds an email address to the TO field in the MMS | RW |
| 83 | string | ascii | WIP_MMS_CC_PHONE* | Adds a telephone number to the CC field in the MMS | RW |
| 84 | string | ascii | WIP_MMS_CC_EMAIL* | Adds an email address to the CC field in the MMS | RW |
| 85 | string | ascii | WIP_MMS_BCC_PHONE* | Adds a telephone number to the BCC field in the MMS | RW |
| 86 | string | ascii | WIP_MMS_BCC_EMAIL* | Adds an email address to the BCC field in the MMS | RW |
| 87 | string | ascii | WIP_MMS_SUBJECT | Set the value of the Subject field in the MMS | RW |
| 88 | See Table | u32 | WIP_MMS_CLASS | Set the class of the MMS | RW |
| 89 | See Table | u32 | WIP_MMS_PRIORITY | Set the priority of the MMS | RW |
| 90 | u32 | u32 | WIP_MMS_SENDER_VISIBILITY | Set the sender visibility of the MMS show=0 default hide=1 | RW |
| 91 | string | ascii | WIP_MMS_FROM | Set the sender of the MMS | RW |
| 92 | u32 | u32 | WIP_MMS_MULTIPART_TYPE | Set the value of the MMS Multipart Type via <optval> as u32. Mixed=0 default. Related=1. In the case of "Related" the presentation file type is sent in <optval2> as a string and the start file identification is sent in <optval3> as a string. | RW |
| 93 | See Table | u32 | WIP_MMS_ADD_FILE | About to attach a file of type specified in <optval> as u32, please see Table 5.3.7.6.1 for possible values. The size in Bytes as u32 is sent in <optval2>. The file name is sent as a string in <optval3> Content-id is sent in <optval4>. At least one of either file name or content-id must be set! | W |

| opt num | Value format | Value type | Option type | Description | Type |
|---------|--------------|------------|-------------------------|--|------|
| 94 | string | ascii | WIP_MMS_ADD_FILE_ANY | About to attach a file of type specified in <optval> as a string using the format of mime-type, Example "image/xyz". The size in Bytes is sent in <optval2>. The file name is sent as a string in <optval3> Content-id is sent in <optval4>. At least one of either file name or content-id must be set. | W |
| 95 | u32 | u32 | WIP_MMS_HTTP_DATA_ENCOD | Set the HTTP data transfer encoding no encoding=0 chunked data transfer encoding=1 default. | RW |
| 96 | | - | WIP_MMS_DONE | Sent as the last command to signal that the MMS sending is considered done. This is to catch deadlocks, when for instance a user misses to send a last file. <i>NOTE : Once the MMS is sent, the +WIPPEERCLOSE: 8,* is received to indicate that MMS session is closed. Therefore, there is no need to issue the AT+WIPCLOSE command to close it).</i> | |

* See the [Phone/Mail Option Notes](#) subsections for additional information about this option type.

5.3.7.6.1. Values of <optval> WIP_MMS_CLASS

These are the different types of message class.

| optval | Option Type |
|--------|------------------|
| 0 | PERSONAL default |
| 1 | INFORMATIONAL |
| 2 | ADVERTISEMENT |
| 3 | AUTO |

5.3.7.6.2. Values of <optval> WIP_MMS_PRIORITY

These are the different types of priority.

| optval | Option Type |
|--------|----------------|
| 0 | LOW |
| 1 | NORMAL default |
| 2 | HIGH |

5.3.7.6.3. Values of <optval> WIP_MMS_ADDFILE

These are the file types that correspond to the values predefined in the MMS header specification. For other types of file the MIME type has to be explicitly set as a string.

| optval | Option Type | Description |
|--------|-------------|--|
| 0 | UTF8 | A text file of type UTF8 |
| 1 | UTF16 | A text file of type UTF16 |
| 2 | UCS2 | A text file of type USC2 |
| 3 | US_ASCII | A text file of type (US) ASCII |
| 4 | JPEG | An image file of type JPEG |
| 5 | GIF | An image file of type GIF |
| 6 | TIFF | An image file of type TIFF |
| 7 | PNG | An image file of type PNG |
| 8 | WBMP | An image file of type WBMP |
| 9 | SMIL | A multimedia presentation of type SMIL |

5.3.7.6.4. Phone/Mail Option Notes

5.3.7.6.4.1. Recipients

The total number of recipient (To+Cc+Bcc) must be less than or equal to 12, and for each recipient list (To or Cc or Bcc), the string length must be less than 250 characters, including "/TYPE=PLMN" in case of phone recipient type.

5.3.7.6.4.2. Command and Response

When getting WIP_MMS_TO_PHONE/WIP_MMS_TO_MAIL or WIP_MMS_CC_PHONE/WIP_MMS_CC_MAIL or WIP_MMS_BCC_PHONE/WIP_MMS_BCC_MAIL string is received in +WIPOPT, the response will include the complete TO or CC or BCC recipient list, preceded by TYPE/PLMN for phone recipient type. For example:

If you enter

AT+WIPOPT=8,1,2,81,"0683517984" (set WIP_MMS_TO_PHONE option)

then

AT+WIPOPT=8,1,2,82,ovc@sierrawireless.com (set WIP_MMS_TO_MAIL option)

After entering AT+WIPOPT=8,1,1,81 (get WIP_MMS_TO_PHONE option) or AT+WIPOPT=8,1,1,82 (get WIP_MMS_TO_MAIL option), the response will be:

+WIPOPT: 8,82,"0683517984/TYPE=PLMN;ovc@sierrawireless.com"



6. Data Exchange for Protocol Services

The section deals with the data exchange for the services over TCP/IP. All the commands required for the data exchange through different services are mentioned in succeeding sections.

6.1. File Exchange +WIPFILE



6.1.1. Description

The +WIPFILE command defines the “file system” services that send a block of data through standard TCP/IP protocols. This command is used for file transfer/reception.

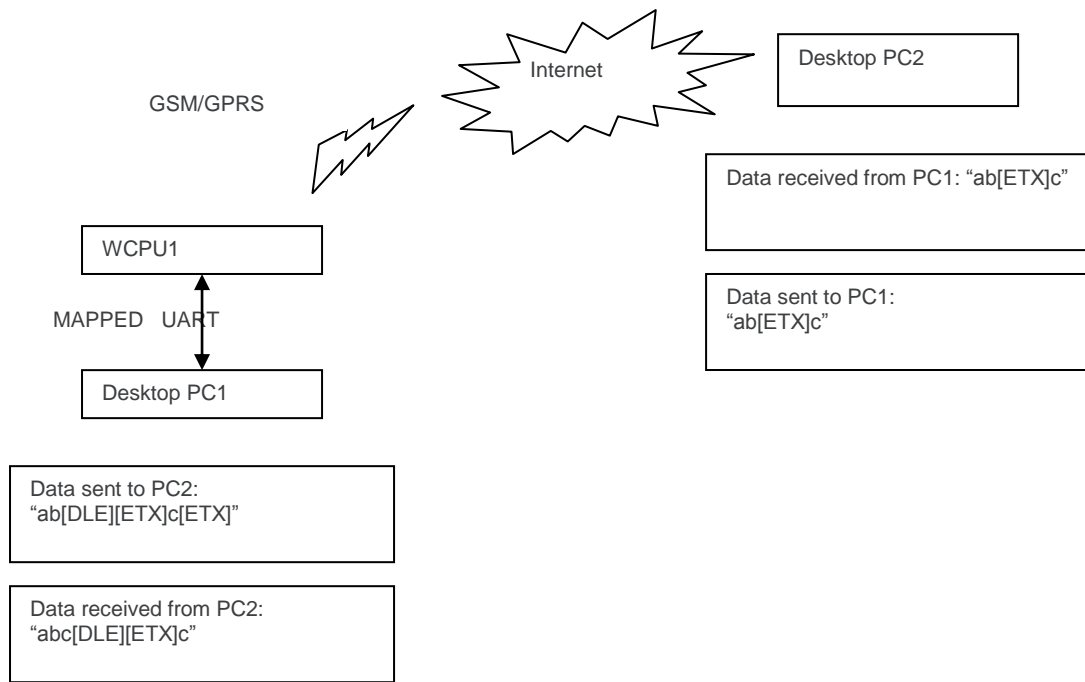
The data can be transferred using two modes: continuous mode and continuous transparent mode. The FTP/HTTP/SMTP/MMS protocols support continuous mode of operation. But, continuous transparent mode is supported only by FTP protocol.

By default, all these protocols transfer data using continuous mode. However, data transfer using FTP protocol can be configured using <dle_mode> parameter.

Note: There is no <dle_mode> parameter specified in the +WIPFILE command to configure mode of operation for HTTP/SMTP protocol.

6.1.1.1. [ETX] Escaping Mechanism

In case an [ETX] character needs to be transmitted as data, it should be preceded by [DLE] character. A single [ETX] character marks the end of transmission. Similarly, [ETX] characters received from the internet are sent to the host through the serial port preceded by a [DLE] character.



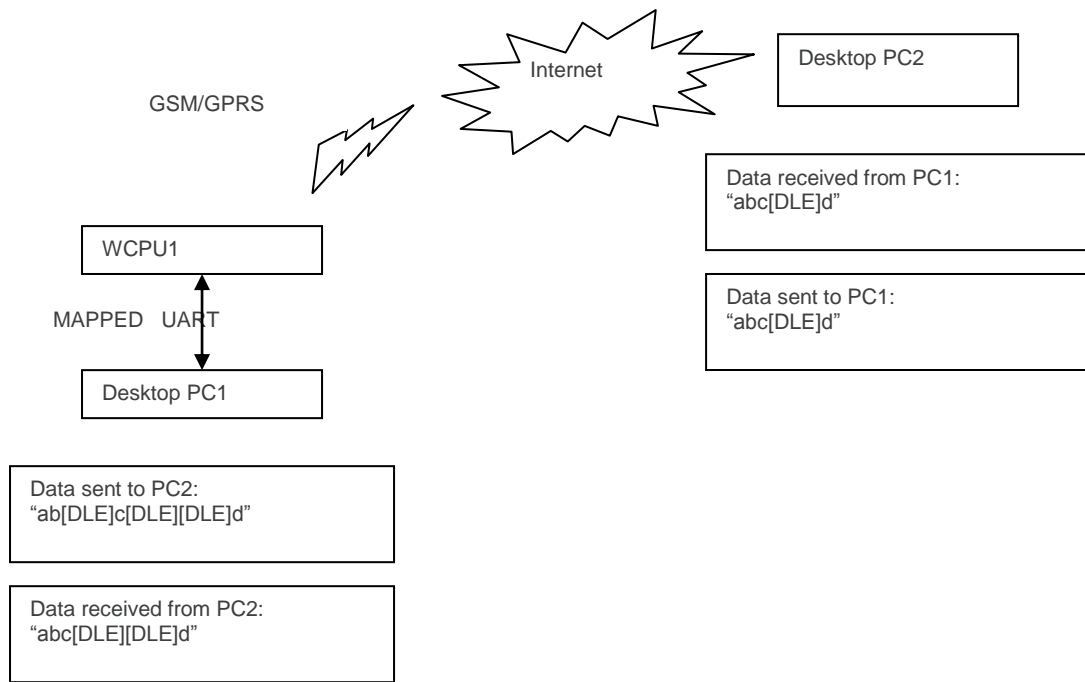
The above schematic explains how [ETX] characters which have a special meaning in WIPsoft are handled on Sierra Wireless embedded module.

On transmitting side, when [ETX] characters are escaped by a DLE (use case: Desktop PC1 sends data to the embedded module. Data contains an [ETX] character escaped by a [DLE] character ([DLE] [ETX] sequence), then the [ETX] character is transmitted as data.

On the receiving side, when [ETX] character is received as data (use case: The PC2 sends data to the embedded module. Data contains an [ETX] character), then the [ETX] character will be preceded by a [DLE] character when it is sent to host through the serial port.

6.1.1.2. [DLE] Escaping Mechanism

In case a [DLE] character needs to be transmitted as data, it should be preceded by another [DLE] character. A single [DLE] character, not preceded by a [DLE] character will not be transmitted. Similarly, [DLE] characters received are sent to the host through the serial port preceded by a [DLE] character.



The above schematic explains how [DLE] characters which have a special meaning in WIPsoft are handled on Sierra Wireless embedded module.

On the transmitting side, when [DLE] characters are escaped by another [DLE] character (use case: Desktop PC1 sends data to the embedded module. Data contains a non escaped [DLE] character, and another escaped [DLE] character ([DLE][DLE] sequence), then the [DLE] character is transmitted as data. A single [DLE] character is ignored and not transmitted.

On the receiving side, when [DLE] character is received as data (use case: The PC2 sends data to the embedded module. Data contains an [DLE] character), then the [DLE] character will be preceded by another [DLE] character when it is sent to host through the serial port.

6.1.2. FTP/HTTP/SMTP Session in Continuous Mode

In continuous mode, an [ETX] character is considered as an end of data. In case an [ETX]/[DLE] character needs to be transmitted as data, it should be preceded by [DLE] character. Similarly, [ETX]/[DLE] characters received by the TCP/IP stack from the internet are sent to the host through the serial port preceded by a [DLE] character.

The mapped UART can be switched back to AT mode either by:

1. sending ETX character
2. sending +++ sequence with 1 second guard time before and after the sequence
3. controlling the DTR signal using AT&D command

When the UART leaves data mode, the currently unsend data are transferred.

6.1.3. FTP Session in Continuous Transparent Mode

In this mode, [DLE]/[ETX] characters are considered as normal data and not as special characters. In case [ETX]/[DLE] character is received, it will not be preceded by a [DLE] character before sending it to the mapped UART.

The mapped UART can be switched back to AT mode either by,

1. sending +++ sequence with 1 second guard time before and after the sequence
2. controlling the DTR signal using AT&D command

When the UART leaves data mode, the currently unsent data are transferred.

6.1.4. Syntax

- if <protocol>=4

Action Command

AT+WIPFILE=<protocol>,<index>,<mode>,<filename>[,<dle_mode>]

CONNECT

...

OK

- if <protocol>=5

Action Command

AT+WIPFILE=<protocol>,<index>,<mode>,<filename>[,<username>,<password>][,<headers list>[...]]

CONNECT

...

OK

- if <protocol>=6

Action Command

AT+WIPFILE=<protocol>,<index>,<mode>

CONNECT

...

OK

- if <protocol>=7

*Action Command***AT+WIPFILE=<protocol>,<index>,<mode>,<filename>**

CONNECT

...

OK

- if <protocol>=8

*Action Command***AT+WIPFILE=<protocol>,<index>,<mode>,[<filename>],[<content-id>][<lastindicator>]**

CONNECT

...

OK

- if <protocol>=5

*Unsolicited response***+WIPFILE: 5,<index>,<mode>,<http status code>,<http status reason>***Read command***AT+WIPFILE?**

OK

*Test Command***AT+WIPFILE=?**

OK

6.1.5. Parameters and Defined Values

| | | |
|--------------------------|---|--|
| <protocol>: | protocol type | |
| | 4 | FTP |
| | 5 | HTTP |
| | 6 | SMTP |
| | 7 | POP3 |
| | 8 | MMS |
| <idx>: | channel identifier | |
| <mode>: | file transfer mode | |
| | 1 | <p>This command switches the UART to data mode and prints the content of the file on UART. The end of the file is marked by [ETX] character and UART switches back to AT mode.</p> <p>This mode is used for downloading file from the FTP server if <protocol>=4.</p> <p>This mode is used for downloading data of the specified URL using HTTP GET method if <protocol>=5.</p> <p>This mode is used for retrieving mail without deleting it from the POP3 server if <protocol>=7.</p> <p>This mode is not supported by SMTP protocol.</p> |
| | 2 | <p>This command switches the UART to data mode and accepts a stream of data terminated by [ETX] character.</p> <p>This mode is used for uploading file to the FTP server if <protocol>=4.</p> <p>This mode is used for uploading data to the specified URL using HTTP PUT method if <protocol>=5.</p> <p>This mode is used for sending mail to the SMTP server if <protocol>=6.</p> <p>This mode is not supported by POP3 protocol.</p> |
| | 3 | <p>This mode is used for deleting the specified URL using HTTP DELETE method if <protocol>=5.</p> <p>This mode is used for retrieving mail and deletion after retrieval from the POP3 server if <protocol>=7.</p> <p>This mode is not supported by FTP and SMTP protocol.</p> |
| | 4 | <p>This command switches the UART in data mode and accepts a stream of data terminated by [ETX] character.</p> <p>This mode is used for uploading data to the HTTP server using HTTP POST method if <protocol>=5.</p> <p>This mode is not supported by FTP, SMTP and POP3 protocol.</p> |
| 5 | <p>This command switches the UART to data mode and accepts a stream of data terminated by [ETX] character.</p> <p>This mode is used for uploading file using FTP APPEND method server if <protocol>=4.</p> <p>This mode is not supported by other protocols</p> | |

| | |
|------------------------------------|---|
| | <p>6 This mode is used to upload DATA in Data Offline mode⁽¹⁾ (by using +WIPDATARW command). This mode is used for uploading files to the FTP server if <protocol>=4. This mode is used for sending mail to the SMTP server if <protocol>=6. This mode is not supported by other protocols.</p> <p>7 This mode is used to download and display DATA in Data Offline mode⁽¹⁾ (by using +WIPDATARW command). This mode is used for downloading a file from the FTP server if <protocol>=4. This mode is used for downloading data of the specified URL using HTTP GET method if <protocol>=5. This mode is used for retrieving mail without deleting it from the POP3 server if <protocol>=7. This mode is not supported by other protocols.</p> <p>8 This mode is used for retrieving mail and deleting after retrieval from the POP3 server if <protocol>=7 in Data Offline mode⁽¹⁾ (by using +WIPDATARW command). This mode is not supported by other protocols.</p> <p>9 This mode is used for uploading a file using FTP APPEND method server if <protocol>=4 in Data Offline mode⁽¹⁾ (by using +WIPDATARW command). This mode is not supported by other protocols</p> |
| <filename>: | <p>file name</p> <p>if <protocol>=4: specify the name of the file to upload or download</p> <p>The maximum file length is limited to 128 characters. The actual filename, including path name has to be used.</p> <p>if <protocol>=5: URL of the HTTP request</p> <p>if <protocol>=7: mail id in string format</p> <p>if <protocol>=8: the identifier matching the identifier specified in WIPOPT. Please note that the order of the files sent using WIPFILE must match the order of the files specified using WIPOPT.</p> |
| <dle_mode>: | <p>Mode to configure continuous/continuous transparent mode</p> <p>This option specifies whether the file should be uploaded/downloaded using continuous or continuous transparent mode using FTP protocol. By default the mode will be set to 0 i.e., continuous mode. If this value is set to 1, data will be transferred using continuous transparent mode.</p> <p>Range: 0–1 (default value: 0)</p> |
| <user name>: | <p>user name in string format</p> |
| <password>: | <p>Password in string format</p> |
| <header list>: | <p>HTTP header message (name-value pair)</p> <p>The first string in the message header field is the name of the header and the second string is the value of the header.</p> |
| <...> | <p>additional HTTP message header fields</p> <p>more pairs(name, value) of HTTP message header field can be added</p> |
| <http status code>: | <p>HTTP 3 digit status code of the response</p> |
| <http status reason>: | <p>HTTP status reason of the response in string format</p> |
| <content-id> | <p>MMS Content-id header</p> |

| | |
|------------------------------|---|
| <lastindicator> | Indicates that the file is the last of the files to send. |
|------------------------------|---|

¹: See the [DATA Offline session +WIPDATARW](#) section for more information.

6.1.6. Parameter Storage

None

6.1.7. Possible Errors

| “+CMEE” AT error code | Description |
|------------------------------|--|
| 800 | invalid option |
| 803 | operation not allowed in the current WIP stack state |
| 830 | bad index |
| 831 | bad state |
| 834 | not implemented |
| 836 | memory allocation error |
| 837 | bad protocol |
| 839 | error during channel creation |
| 846 | internal error: FCM subscription failure |
| 860 | protocol undefined or internal error |
| 867 | POP3 email retrieving error |
| 868 | POP3 email size error |
| 880 | SMTP sender email address rejected by server |
| 881 | SMTP recipient email address rejected by server |
| 882 | SMTP CC recipient email address rejected by server |
| 883 | SMTP BCC recipient email address rejected by server |
| 884 | SMTP email body send request rejected by server |
| 890 | Service denied |
| 891 | Message format corrupt |
| 892 | Address unresolved |
| 893 | Message not found |
| 894 | Network problem |
| 895 | Content not accepted |
| 896 | Unsupported message |
| 897 | Unspecified error |

6.1.8. Examples

| Command | Responses |
|--|---|
| AT+WIPFILE=4,1,1,"data.bin" | CONNECT <data received terminated by [ETX] character> |
| <i>Note: Download file in continuous mode</i> | OK |
| AT+WIPFILE=4,1,2,"report.log" | CONNECT <data terminated by [ETX] character> |
| <i>Note: Upload file in continuous mode</i> | OK |
| AT+WIPFILE=4,1,5,"report.log" | CONNECT <data terminated by [ETX] character> |
| <i>Note: Upload file in continuous mode; data will be added at the end of file</i> | OK |
| AT+WIPFILE=4,1,1,"data.bin",1 | CONNECT <data> +++ |
| | OK |
| <i>Note: Download file in continuous transparent mode</i> | <i>Note: +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPFILE=4,1,2,"report.log",1 | CONNECT <data> +++ |
| | OK |
| <i>Note: Upload file in continuous transparent mode</i> | <i>Note: +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPFILE=4,1,1,"data.bin",0 | CONNECT <data received terminated by [ETX] character> |
| <i>Note: Download file in continuous mode</i> | OK |
| AT+WIPFILE=4,1,2,"report.log",0 | CONNECT <data terminated by [ETX] character> |
| <i>Note: Upload file in continuous mode</i> | OK |
| AT+WIPFILE=5,1,2,"urlForPut" | CONNECT <data terminated by [ETX] character> OK +WIPFILE:5,1,2,<http status code>,<http status reason> |

| Command | Responses |
|--|--|
| <i>Note: Send a HTTP PUT request to URL</i> | |
| AT+WIPFILE=5,1,3,"urlForDelete" | CONNECT <data received terminated by [ETX] character> OK +WIPFILE:5,1,3,<http status code>,<http status reason> |
| <i>Note: Send a HTTP DELETE request to URL</i> | |
| AT+WIPFILE=5,1,4,"urlForPost" | CONNECT <data received terminated by [ETX] character> OK +WIPFILE:5,1,4,<http status code>,<http status reason> |
| <i>Note: Send a HTTP POST request to URL</i> | |
| AT+WIPFILE=6,1,2 | CONNECT <data sent terminated by [ETX] character> OK |
| <i>Note: Send data mail content</i> | |
| AT+WIPFILE=7,1,1,"15" | CONNECT <data received terminated by [ETX] character > OK |
| <i>Note: Retrieve data from the given ID</i> | <i>Note: Retrieve mail ID 15 Mail is not deleted after retrieval</i> |
| AT+WIPFILE=7,1,3,"1" | CONNECT <data received terminated by [ETX] character > OK |
| <i>Note: Retrieve data from the given ID</i> | <i>Note: Retrieve mail ID 1 and delete it after retrieval</i> |

6.1.9. Notes

The [ETX] character is considered as an end of data. Hence, in case [ETX] character needs to be transmitted, it should be preceded by [DLE] character.

For MMS, when sending file data through AT+WIPFILE command, if data size is greater than the one specified via WIP_MMS_ADD_FILE or WIP_MMS_ADD_FILE_ANY options, the data will be truncated to said option's size, and module will leave the data mode. When going back to AT mode, +WIPFILE: proto,index,size will be received with the size equal to the size of the data that will be included in the MMS file.

6.2. Socket Data exchange +WIPDATA



6.2.1. Description

The +WIPDATA command is used to read/write from/to a socket. On successful execution of the command, the UART switches to data mode. The UART can be switched back to AT mode by sending “+++” with 1 second guard time before and after the sequence. If data is not read using +WIPDATA command, further data will be delayed.

An unsolicited event is received when there is a data to read on socket.

Data can be sent on the sockets using two modes

- continuous mode
- continuous transparent mode

Note: When using the UDP protocol, consider that you cannot send more than the WIP_COPT_RCV_BUFSIZE data receiving buffer size. Based on this, be sure that no data will be lost, as it depends on the bearer "speed".

6.2.2. Continuous Mode

6.2.2.1. TCP Sockets in Continuous mode

In continuous mode, an [ETX] character is considered as an end of data. When an [ETX] character is sent on the mapped UART, the TCP socket is shutdown and the peer side is informed of this shutdown with the indication “[CR][LF]SHUTDOWN[CR][LF]” on the mapped UART.

In case an [ETX]/[DLE] character needs to be transmitted as data, it should be preceded by [DLE] character. Similarly, [ETX]/[DLE] characters received by the TCP/IP stack from the internet are sent to the host through the serial port preceded by a [DLE] character.

To close sockets, switch the UART to AT command mode and use +WIPCLOSE command.

6.2.2.2. UDP Sockets in Continuous mode

UDP is a connectionless protocol and hence there is no way to detect or cause a shutdown. However, an [ETX] character is used to mark the boundaries of datagrams.

All data written on an UDP socket is collected till an [ETX] character is encountered or the maximum size of the datagram¹ is reached and will be sent as a single datagram. Similarly when reading data,

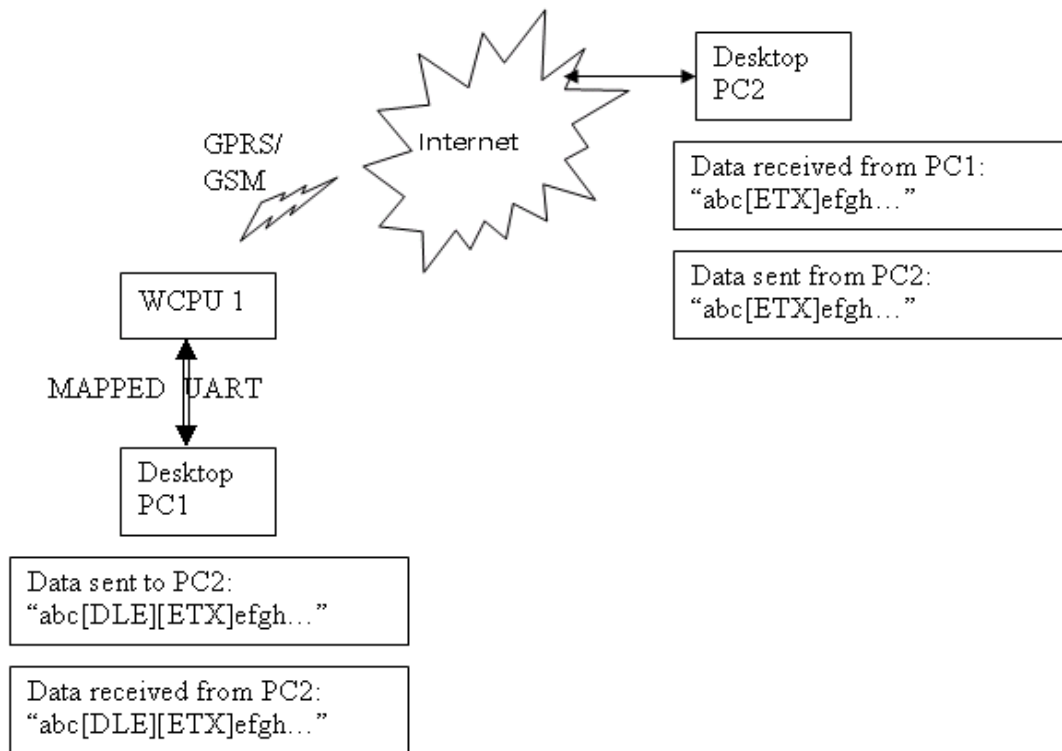
¹ Maximum size of an UDP datagram has been fixed to 5904 Bytes. This limit is an arbitrary one. Nevertheless, note that smaller the datagram is the surer it will reach the aimed destination. Note that UDP is not a reliable transport layer.

all data will be read till an [ETX] character is encountered which indicates the end of the datagram. Note that, in this mode, packet segmentation feature is not supported.

In case an [ETX]/[DLE] character needs to be transmitted, it should be preceded by [DLE] character similar to TCP socket.

When the UART leaves DATA mode, either because of “+++” escape sequence or because of an AT+WIPDATA=1, index, 0 on another UART, the currently unsent data are sent as a single datagram.

6.2.2.3. [ETX] Escaping Mechanism



The above schematic explains how [ETX] characters – which have a special meaning in WIPsoft – are handled on Sierra Wireless embedded module.

On transmitting side, when [ETX] are not escaped (use case: Desktop PC1 sends data towards embedded module. Data contain a non escaped [ETX] (⇔ no [DLE][ETX] sequence), then [ETX] is not transmitted but an action is done on embedded module regarding the concerned socket:

- UDP socket: a non escaped [ETX] marks the boundary of the current datagram to be sent. Datagram is immediately sent and the [ETX] is not sent towards the desktop PC2.
- TCP socket: a non escaped [ETX] causes a TCP shutdown operation on the transmitting direction: peer is informed that embedded module will not send any more data on that socket. Usually, peer will shutdown the other way (downlink) and this will result in a “peer close event” on the socket.

On receiving side, when [ETX] are not escaped (use case: embedded module sends data towards Desktop PC1. Data contain a non escaped [ETX] (⇔ no [DLE][ETX] sequence), then [ETX] means that a special “IP” event occurred on embedded module regarding the concerned socket:

- UDP socket: a non escaped [ETX] signals the boundary of the current received datagram.

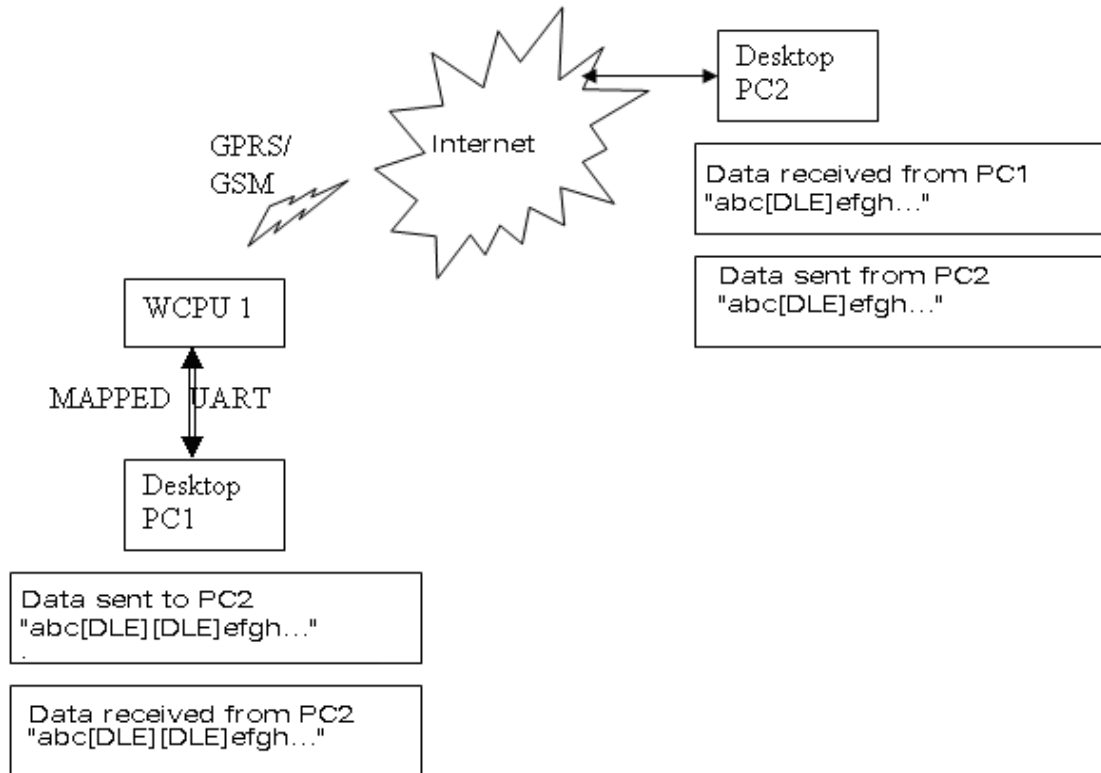
- TCP socket: a non escaped [ETX] signal that the peer TCP connected TCP unit shutdown the downlink way. Desktop PC1 should then close the uplink socket to totally terminate the TCP "session".

| Protocol | Mapped UART | IP Network (active socket) |
|----------|--------------------------------------|---|
| UDP | Data containing [DLE][ETX] sequence. | Data containing [ETX]. |
| UDP | [ETX] alone. | Mark the boundary of the UDP Datagram received/to be transmitted. |
| TCP | Data containing [DLE][ETX] sequence. | Data containing [ETX]. |
| TCP | [ETX] alone. | Causes/signals a shutdown operation on TCP socket. |

Note: The behavior is symmetrical: apply both on transmitting/receiving side of mapped UART.

6.2.2.4. [DLE] Escaping Mechanism

A [DLE] character will be sent as data only when it is preceded by another [DLE] character. A single [DLE] character which is not preceded by a [DLE] character will not be transmitted.



The above schematic explains how [DLE] characters – which have a special meaning in WIPsoft – are handled on Sierra Wireless embedded module.

On transmitting side, when [DLE] is not escaped (use case: Desktop PC1 sends data towards embedded module. Data contains a non escaped [DLE] (⇔ no [DLE][DLE] sequence), then [DLE] is not transmitted.

On transmitting side, when [DLE] is escaped (use case: Desktop PC1 sends data towards embedded module. Data contain an escaped [DLE] (⇔ [DLE][DLE] sequence) then [DLE] data is transmitted.

On the receiving side (use case: when Desktop PC2 sends data towards embedded module. Data contains a no escaped [DLE]) the data sent from the embedded module to Desktop PC1 will contain an escaped [DLE] preceding the [DLE] character (Desktop PC1 receives [DLE][DLE] character from embedded module).

The scenario is same for both TCP and UDP sockets.

| Protocol | Mapped UART | IP Network (active socket) |
|----------|--------------------------------------|----------------------------|
| UDP | Data containing [DLE][DLE] sequence. | Data containing [DLE]. |

| Protocol | Mapped UART | IP Network (active socket) |
|----------|--------------------------------------|----------------------------|
| UDP | [DLE] alone. | A single [DLE] is ignored. |
| TCP | Data containing [DLE][DLE] sequence. | Data containing [DLE]. |
| TCP | [DLE] alone. | A single [DLE] is ignored. |

6.2.3. Continuous Transparent Mode

6.2.3.1. TCP Sockets in Continuous Transparent Mode

In this mode there is no special meaning associated for [DLE]/[ETX] characters. They are considered as normal data and all the data will be transmitted on the mapped UART.

6.2.3.2. UDP Sockets in Continuous Transparent Mode

In this mode there is no special meaning associated for [DLE]/[ETX] characters. They are considered as normal data and all the data will be transmitted on the mapped UART. In case [ETX]/[DLE] character is received, it will not be preceded by a [DLE] character before sending it to the mapped UART.

6.2.4. Leaving Continuous /Continuous Transparent Mode

The UART can be switched back to AT mode

- by sending “+++” with 1 second guard time before and after the sequence
- by sending an AT+WIPDATA=<proto.,<index>,0 on another UART in AT mode

When the UART leaves data mode either because of “+++” escape sequence or because of an unmapping done on another UART, the currently unsent data are sent as a single datagram.

6.2.5. Resetting TCP Sockets

A TCP socket is reset when the connection is aborted due to an error on the socket. When the socket is reset, an [ETX] character is sent on the mapped UART to indicate the end of communication. The mapped UART switches to AT mode and "+CME ERROR: 843" is displayed on the UART.

6.2.6. Syntax

Action Command

AT+WIPDATA=<protocol>,<idx>,<mode>[,<send size>,<wait time>]

CONNECT

Note: Once the +WIPDATA indication has been received, on peer closed, +WPPEERCLOSE indication won't be received unless AT+WIPDATA has been sent.

Read Command

AT+WIPDATA?

NONE

Test Command

AT+WIPDATA=?

OK

- if <protocol>=1

Unsolicited response

+WIPDATA: <protocol>,<idx>,<datagram size>,<peer IP>,<peer port>

Caution: Using +WIP AT commands, when receiving several UDP datagrams on an IP bearer, +WIPDATA indication is sent once for the first received datagram. Next indication (for next remaining UDP datagram to read) is sent once the first datagram have been read (using +WIPDATA command).

- if <protocol>=2

Unsolicited response

+WIPDATA: <protocol>,<idx>,<number of readable bytes>

Caution: The value returned by <number of readable bytes> indicates that there is some TCP data ready to be read but number of bytes returned might not be reliable. Moreover, using +WIPAT commands, when receiving several TCP packets on an IP bearer, +WIPDATA indication is sent once for the first received packet. The next indication (for the next remaining TCP packet to read) is sent after the first packet have been read (using +WIPDATA command).

6.2.7. Parameters and Defined Values

| | | |
|---------------------------|---|--|
| <protocol>: | socket type | |
| | 1 | UDP |
| | 2 | TCP client |
| <idx>: | socket identifier | |
| <mode>: | mode of operation | |
| | 0 | unmap: switch the UART (mapped to continuous mode) to AT mode. |
| | 1 | continuous: switch the UART to data mode. |
| | 2 | continuous transparent: switch the UART to data mode. In this mode, [DLE]/[ETX] characters are considered as normal data and not special characters. |
| 3 | Data Offline: Activate Data Data Offline mode ⁽¹⁾ on the specified socket. In this mode, [DLE]/[ETX] characters are considered as normal data and not special characters. | |
| <send size>: | data packet size: This parameter specifies the size of the data packet that needs to be sent to the peer. This parameter is supported only for UDP continuous transparent mode. range: 8-1460 (default value: 1020) | |
| <wait time>: | timeout for configuring the packet segmentation on IP network side: This parameter specifies the timeout after which the buffered data will be sent to the peer irrespective of size of the data packet. This parameter is supported only for UDP continuous transparent mode. range: 1-100 (default value: 2) | |

¹: See the [DATA Offline session +WIPDATARW](#) section for more information.

6.2.8. Parameter Storage

None

6.2.9. Possible Errors

| "+CMEE" AT error code | Description |
|-----------------------|--------------------------|
| 831 | bad state |
| 836 | memory allocation error |
| 837 | bad protocol |
| 843 | connection reset by peer |

6.2.10. Examples

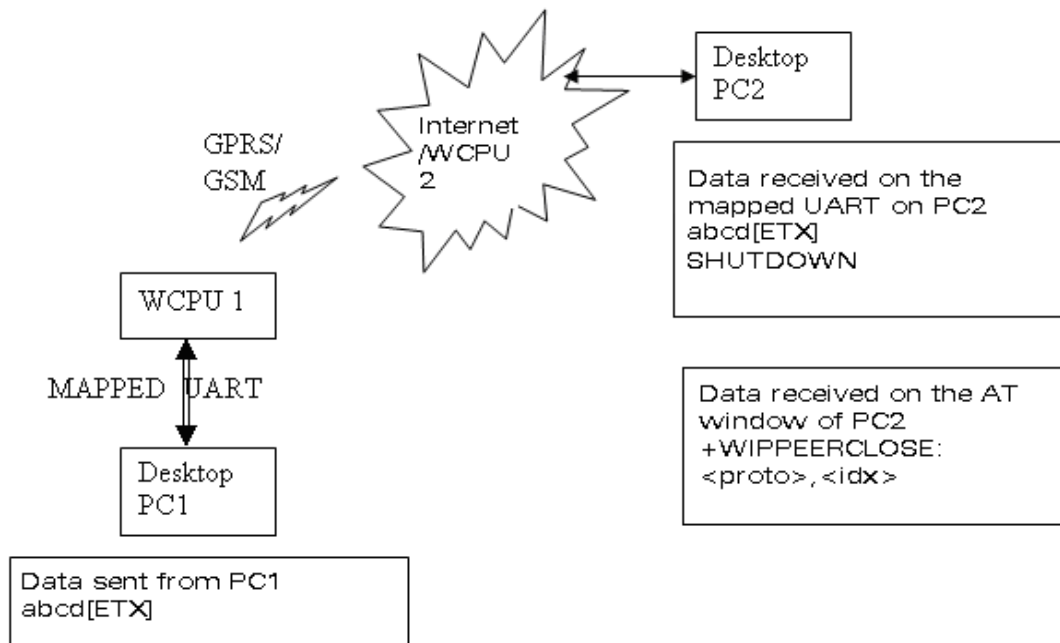
| Command | Responses |
|--|--|
| AT+WIPDATA=2,5,1 | CONNECT <read/write data> +++ OK |
| <i>Note; TCP Client with index 5 can send/read data in continuous mode</i> | <i>Note; +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=2,5,1,10,5 | CONNECT <read/write data> +++ OK |
| <i>Note; TCP Client with index 5 can send/read data in continuous mode</i> | <i>Note; +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=1,5,1 | CONNECT <read/write data> +++ OK |
| <i>Note; UDP with index 5 can send/read data in continuous mode</i> | <i>Note; +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=1,5,1 | CONNECT <read/write data> <ETX> OK |
| <i>Note; UDP with index 5 can send/read data in continuous mode</i> | <i>Note; [ETX] character indicates end of data</i> |
| AT+WIPDATA=1,5,2 | CONNECT <read/write data> +++ OK |
| <i>Note; UDP with index 5 can send/read data in continuous transparent mode with default value set for <send size> and <wait time></i> | <i>Note; +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=1,5,2,20,2 | CONNECT <read/write data> +++ OK |
| <i>Note; UDP with index 5 can send/read data in continuous transparent mode with <send size> set to 20 and <wait time> set to 2</i> | <i>Note; +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=2,5,1,20,10 | CONNECT <read/write data> +++ OK |

| Command | Responses |
|---|--|
| <i>Note: TCP with index 5 can send/read data in continuous mode with <send size> set to 20 and <wait time> set to 10</i> | <i>Note: +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=2,5,2,10,5 | CONNECT <read/write data> +++ OK |
| <i>Note: TCP with index 5 can send/read data in continuous transparent mode with <send size> set to 10 and <wait time> set to 5</i> | <i>Note: +++ sequence causes the UART to switch to AT mode</i> |
| AT+WIPDATA=2,5,2 | CONNECT <read/write data> +++ OK |
| <i>Note: TCP with index 5 can send/read data in continuous transparent mode</i> | <i>Note: +++ sequence causes the UART to switch to AT mode</i> |

6.2.11. Notes

6.2.11.1. Continuous Mode (Non Transparent) for a TCP Mapped Socket

If the [ETX] character is sent from the peer, it is considered as an end of data transfer. After sending an [ETX] character, the socket will be shutdown and the peer will be informed of this shutdown by a “[CR][LF]SHUTDOWN[CR][LF]” indication on its mapped UART and the UART does not switch to AT mode. This indicates that no more data can be sent from the host socket, but it can receive data. The below schematic shows the shutdown procedure for a TCP socket:



In the above schematic, a TCP socket is connected. On the transmitting side, data and [ETX] is sent (use case: Desktop PC1 is a embedded module which sends data to PC2 which is either a PC or a embedded module), the data is received on PC2 and [ETX] character shutdowns the socket on the transmitting side and displays a message “[CR][LF]SHUTDOWN[CR][LF]” on the mapped UART of PC2.

When PC2 is switched back to AT mode, “+WIPPEERCLOSE: <protocol>,<idx>” indication is received indicating that no more data can be sent by PC1 but can read data sent from PC2.

There are different indications received for shutdown and reset for a TCP socket. When a TCP socket is reset, [ETX] character is sent on the mapped UART to indicate the end of communication. The mapped UART switches to AT mode and “+CME ERROR: 843” is displayed on the UART. The reset and shutdown can therefore be distinguished by the indications received on the UART.

6.2.11.2. Mapping/Unmapping of a Mapped UDP and TCP Socket

When a TCP socket is unmapped and still active, it is possible to map it again in another mode which is different from the previous one without closing the TCP socket.

The UART switches back to AT mode due to “+++”with 1 second guard time before and after the sequence or by sending an AT+WIPDATA=<proto>,<index>,0 on another UART in AT mode. This applies to both UDP and TCP protocols.

When +++ is issued, embedded module switches from DATA mode to AT mode. If ATO command is used to switch the embedded module back to DATA mode,

- +CME ERROR:3 will be received when GPRS bearer is used
- no response is received when GSM bearer is used

To switch the embedded module back to DATA mode, AT+WIPDATA=x,x,x should be used instead of ATO. After executing AT+WIPDATA=x,x,x command, “CONNECT” will be received to indicate that the embedded module is switched back to DATA mode.

Note that un-mapping socket using +WIPDATA command with <send size> and <wait time> specified results in “ERROR”.

6.2.11.3. Time out Mechanism to know the state of the Peer TCP Socket

In a TCP server-client connection between two remote devices if the peer socket is closed down abruptly (e.g. powered off) the peer TCP socket does not get any indication message. This is a normal behavior. The TCP protocol uses a timeout mechanism to check the state of the TCP sockets in a TCP socket connection. According to this mechanism, to know the state of the peer TCP socket the data needs to be sent and wait for the acknowledgement within a specified time period. If the acknowledgement is not received within the specified time out period then the data is retransmitted. But if the time out occurs before receiving acknowledgement then it implies that the peer TCP socket is closed.

TCP Timeout Period = function (R, N)

Where,

R = Round trip time. This is the time for a TCP packet to go to the remote TCP socket and the time to receive the acknowledgement by the transmitter TCP socket. The typical round trip time is 1 seconds for GPRS.

N = Number of retransmission allowed before the time out happens.

Hence, the typical timeout period is 10 minutes depending on the network and also the peer TCP socket localization.

In WIPsoft, to know the state of the peer socket, data needs to be sent. If acknowledgement is not received within the timeout period then "+CME ERROR: 842" is returned. This indicates that the peer socket is closed.

Please note that the retransmission of the data to the peer TCP socket within the timeout period is managed by the WIPlib Plug-In.

6.2.11.4. Packet Segmentation in TCP Socket

The parameters used for packet segmentation can be configured using +WIPDATA or +WIPCFG command. In case if it is not configured using +WIPDATA command, then the values already set for option WIP_NET_OPT_TCP_MIN_MSS and AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE will be used.

Note that if an attempt is made to set data packet size more than twice the value of WIP_NET_OPT_TCP_MIN_MSS using +WIPDATA command results in "+CME ERROR: 847".

The data sent to a mapped TCP socket through UART will be buffered before sending it to the peer. This buffered data will be sent to the peer when:

- total amount of buffered data is twice or more than the preferred segmentation size. The preferred segmentation size is configurable through the "AT+WIPCFG = 2, 4, <size>" (WIP_NET_OPT_TCP_MIN_MSS) or +WIPDATA command.
- internal timer expires. The timeout period is configurable through the "AT+WIPCFG = 2,12,<time>" (AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE) or +WIPDATA command.
- socket is unmapped, shut down or closed

In some scenarios, there might be a segmentation of data packets because of timer expiration, network problems etc. Thus a single packet of data may be received in more than one packet at the peer

6.2.11.5. Packet Segmentation in UDP Socket

This feature for UDP is supported only in case of continuous transparent mode. If the +WIPDATA command is executed in continuous mode to use this feature, "ERROR" will be returned. The parameters used for packet segmentation can be configured using +WIPDATA command. In case if it is not configured using +WIPDATA command, default value of these parameters will be used.

The data sent to a mapped UDP socket through UART will be buffered before sending it to the peer. This buffered data will be sent to the peer when:

- the buffered data size is equal to segmentation size. Note that if the buffered data is greater than segmentation size, then the data will be written to the channel in chunks of segmentation size.
- the timer expires
- socket is unmapped or closed

In some scenarios, there might be a segmentation of data packets because of timer expiration, network problems etc. Thus a single packet of data may be received in more than one packet at the peer.

6.3. DATA Offline session +WIPDATARW

This command is used to upload or download data on UART without switching to DATA mode (CONNECT/OK Online mode) for +WIPDATA and +WIPFILE commands.

DATA offline session starts when AT+WIPDATA command is used with mode (3) or when AT+WIPFILE command is used with mode (6, 7, 8 or 9). Otherwise +WIPDATARW command can't be use and returns +CME ERROR 831 error message.

In this mode,[DLE]/[ETX] characters are considered as normal data and not special characters. Data read or written are in hexadecimal dump format.

6.3.1. Restrictions

- DATA offline feature is not recommended to send or received quickly high data volume. In this case, the nominal online mode shall be used.
- To improve AT command exchange and behavior, the use of USB com port is recommended. Otherwise, UART com port buffer threshold shall be decrease with +WHCNF command (see General AT Command User Guide).
- Be careful for TCP socket (+WIPDATA), a maximum of 5840 bytes can be send. Over this size data are lost but no error is returned.
- +WIDATARW command should not be used to fill TX sender buffer with more than 5840 bytes (4*1460) for TCP socket.

6.3.2. Syntax

Action abort : <command>= 0

AT+WIPDATARW=<command>,<idx>

+WIPDATARW: <state>,<idx>

OK

Action write : <command>= 1, 2 or 3

AT+WIPDATARW=<command>,<idx>,"<DATA DUMP>"

OK

Unsolicited message <state> = 0 or 1

+WIPDATARW: <state>,<idx>

Unsolicited message <state> = 3

+WIPDATARW: <state>,<idx>,<size>


```

Unsolicited message <state> = 2
+WIPDATARW: <state>,<idx>,<nb_block>,<num_block>,"<DATA DUMP>"
+WIPDATARW: <state>,<idx>,<nb_block>,<num_block>,"<DATA DUMP>"
    
```

```

Read Command
AT+WIPDATARW?
OK
    
```

```

Test Command
AT+WIPDATARW=?
OK
    
```

6.3.3. Parameters and Defined Values

| | |
|--------------------------|--|
| <command>: | |
| 0 | Close WIPDATARW session. |
| 1 | WRITE and Send: Add DATA in TX buffer then send all data stored in the buffer. |
| 2 | WRITE EXT : Add DATA in TX buffer (but buffer is not sent) |
| 3 | WRITE, Send and close: Add DATA in TX buffer, send all data stored in the buffer, then close the current DATA offline session. This command is only supported for +WIPDATA session. |
| <idx> | channel identifier |
| <state>: | |
| 0 | WIPDATARW session Closed for the channel <idx> specified |
| 1 | WIPDATARW session Opened for the channel <idx> specified. Ready to send or received data |
| 2 | Data reading |
| 3 | Data sent |
| <DATA DUMP> | DATA in hexadecimal format. |

To send more than 200 bytes data, use +WIPDATARW with <command> = 2 to add data to buffer and use for the last block

Data must be written in TX buffer with <command> = 2 by 200 byte block and the last block must be written with <command> =1 to write it and send the TX buffer.”

Only 200 data bytes can be read or written at a time. DATA received are displayed by 200 byte block if received data buffer to display is bigger than 200 bytes.

6.3.4. Parameter Storage

None

6.3.5. Possible Errors

| "+CMEE" AT error code | Description |
|-----------------------|--------------------------|
| 800 | Invalid option |
| 801 | Invalid option value |
| 830 | Bad index |
| 831 | Bad state |
| 836 | Memory allocation error |
| 853 | Data offline buffer full |

6.3.6. Examples

| Command | Responses |
|---|---|
| AT+WIPDATA=1,1,3 <i>(UDP socket have been previously created)</i> | +WIPDATARW:1,1 OK |
| <i>Note : exchange data on socket index 1 with AT Command</i> | <i>Note WIPDATARW session ready on channel idx 1.</i> |
| AT+WIPDATARW=1,1,"30313233343536373839" | +WIPDATARW:3,1,10 OK |
| <i>Note : send 10 data byte on channel idx 1</i> | <i>Note : 10 data bytes had been sent</i> |
| | +WIPDATA: 1,1,10,"192.168.1.2",1357 +WIPDATARW:2,1,1,1,"3132333435363738" OK |
| | <i>Note : Data dump received on channel idx 1</i> |
| AT+WIPDATARW=2,1,"30 [...] 39" | OK |
| <i>Note : Write 200 Data bytes in TX Buffer</i> | <i>Note : 200 data bytes are stored in TX Buffer</i> |
| AT+WIPDATARW=2,1,"30 [...] 39" | OK |
| <i>Note : Write 200 Data bytes in TX Buffer</i> | <i>Note : 200 data bytes are stored in TX Buffer</i> |
| AT+WIPDATARW=1,1,"3039" | +WIPDATARW:3,1,402 OK |
| <i>Note : Write 2 Data bytes in TX Buffer and send buffer.</i> | <i>Note : 402 data bytes have been sent</i> |
| AT+WIPDATARW=0,1 | +WIPDATARW:0,1 OK |

| | |
|--|---|
| <i>Note : Close WIPDATARW session on channel idx 1</i> | |
| AT+WIPDATARW=1,1,"30313233343536373839" | +CME ERROR: 831 |
| <i>Note : try to send 10 bytes on channel index 1</i> | <i>Note : WIPDATARW session is not open in channel idx 1</i> |
| | |
| AT+WIPFILE=4,2,6,"./filename.txt" | +WIPDATARW:1,2 OK |
| <i>Note : Start data offline session for uploading file "filename.txt"</i> | |
| AT+WIPDATARW=1,2,"30313233343536373839" | +WIPDATARW:4,2,10 +WIPDATARW:0,2 OK |
| <i>Note : Upload data</i> | <i>Note : 10 bytes uploaded WIPDATARW session closed on channel idx 2</i> |
| AT+WIPFILE=4,2,7,"./filename.txt" | +WIPDATARW:3,2,2,1,"31 [...]" +WIPDATARW:3,2,2,2,"31 [...]" +WIPDATARW:0,2 OK |
| <i>Note : Download and display "filename.txt" file</i> | <i>Note : More than 200 data downloaded WIPDATARW session closed on channel idx 2</i> |

7. Ping Services

7.1. PING command +WIPPING



7.1.1. Description

The +WIPPING command is used to configure different PING parameters and to send PING requests. An unsolicited response is displayed each time a “PING” echo event is received or a timeout expires.

7.1.2. Syntax

Action Command

```
AT+WIPPING=<host>, [<repeat>,<interval>, [<timeout>, [<nwrite>, [<ttl>]]]]
```

OK

Read Command

```
AT+WIPPING?
```

OK

Test Command

```
AT+WIPPING=?
```

OK

Unsolicited response

```
+WIPPING:<timeout_expired>,<packet_idx>,<response_time>
```

7.1.3. Parameters and Defined Values

| | |
|---------------------------------|--|
| <host>: | host name or IP address string |
| <repeat>: | number of packets to send range: 1-65535 (default value:1) |
| <interval>: | number of milliseconds between packets range: 1-65535 (default value:2000) |
| <timeout>: | number of milliseconds before a packet is considered lost range: 1-65535 (default value:2000) |
| <tTL>: | IP packet Time To Live. Default value is set by WIP_NET_OPT_IP_TTL +WIPCFG option range : 0-255 |
| <nwrite>: | size of packets range : 1-1500 (default value:64) |
| <timeout_expired>: | PING result 0: PING response received before <timeout> 1: <timeout> expired before the response was received |
| <packet_idx>: | packet index in the sequence |
| <response_time>: | PING response time in millisecond |

7.1.4. Parameter Storage

None

7.1.5. Possible Errors

| “+CMEE” AT error code | Description |
|------------------------------|-----------------------|
| 800 | invalid option |
| 801 | invalid option value |
| 819 | error on ping channel |

7.1.6. Examples

| Command | Responses |
|---|---|
| AT+WIPPING="www.sierrawireless.com" | OK +WIPPING: 1,0,0 |
| <i>Note: Ping "www.sierrawireless.com"</i> | <i>Note: Ping "www.sierrawireless.com failed : timeout expired"</i> |
| AT+WIPPING="192.168.0.1" | OK +WIPPING: 0,0,224 |
| <i>Note: Ping "192.168.0.1"</i> | <i>Note: Ping "192.168.0.1 succeeded. Ping response received in 224 ms"</i> |
| AT+WIPPING="192.168.0.1",2,2000,1000 | OK +WIPPING: 0,0,880 +WIPPING: 1,1,xxxx |
| <i>Note: Send 2 successive ping requests to "192.168.0.1". Each Ping is every 2000 ms, timeout is set to 1000 ms (if ping responses time is more than 1000 ms then timeout expires)</i> | <i>Note: Ping "192.168.0.1 succeeded. First Ping response received in 880 ms. Second one was not received before specified timeout (1000 ms) ⇔ timeout expired"</i> |

8. WIPsoft Library API

The WIPsoft Application provides a comprehensive and flexible environment to use the IP feature using AT commands. The WIPsoft Application is an application and it uses the WIPlib Plug-In as the TCP/IP protocol stack. Hence when the WIPsoft application executed no other application can be executed in the embedded module. WIPsoft API allow customer application to subscribe for AT+WIP commands

Customer application can subscribe to AT+WIP commands using WIPsoft library API. This feature allows customer application to use ADL services with WIPsoft services. Note that concurrent access to IP stack from WIPsoft library and WIP library results in unpredictable events and behavior. Hence it is recommended to us either WIPsoft library API or WIP library at a time but not both at the same time.

The FCM flow, through which the WIP AT commands are executed, is subscribed by the WIPsoft library to transfer data between the embedded module and the external device. Hence, if the WIPsoft library is subscribed from the application, same FCM flow should not be subscribed from the same application.

8.1. Required Header File

The header file for the WIP AT command interface is wip_atcmd.h.

8.2. The wip_ATCmdSubscribe Function

The wip_ATCmdSubscribe function subscribes to +WIPCFG, +WIPBR, +WIPPING, +WIPCREATE, +WIPDATA, +WIPFILE, +WIPOPT AT commands provided by WIPsoft.

8.2.1. Prototype

```
s32 wip_ATCmdSubscribe ( void );
```

8.2.2. Parameters

None

8.2.3. Returned Values

The function returns

- 0 on success
- negative error code on failure as described below:

| Error Code | Description |
|------------|--|
| -1 | subscription for WIP AT commands fails |
| -2 | WIP AT commands already subscribed |

8.3. The wip_ATCmdUnsubscribe Function

The wip_ATCmdUnsubscribe function unsubscribes to +WIPCFG, +WIPBR, +WIPPING, +WIPCREATE, +WIPDATA, +WIPFILE, +WIPOPT AT commands provided by WIPsoft.

8.3.1. Prototype

```
s32 wip_ATCmdUnsubscribe ( void );
```

8.3.2. Parameters

None

8.3.3. Returned Values

The function returns

- 0 on success
- negative error code on failure as described below:

| Error Code | Description |
|------------|---|
| -3 | WIP AT commands already unsubscribed |
| -4 | un-subscription for WIP AT commands fails |



9. Examples of Application

9.1. TCP Socket

9.1.1. TCP Server Socket

9.1.1.1. Using GPRS bearer

```
AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name (<login>)
OK
AT+WIPBR=2,6,1,"passwd" //set password (<password>)
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=3,1,80,5,8 //create the server on port 80, idx = 1. The server
//is listening for connection request on port
//80. Spawned sockets will be given the index 5,
//6, 7 and 8. It will accept connection request
//until it has no more socket left.
+WIPACCEPT: 1,5 //unsolicited: the server accepted a connection
//resulting TCP client on idx 5.
AT+WIPDATA=2,5,1 //exchange data on socket index 5
CONNECT
... //read, write
+++ //switch to AT mode
OK
AT+WIPCLOSE=2,5 //close the TCP client socket index 5
OK
```

9.1.1.2. Using GSM bearer

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,5 //open GSM bearer
OK
AT+WIPBR=2,5,2,"Phone number" //set phone number for GSM bearer
OK
AT+WIPBR=2,5,0,"user name" //set user name
OK
AT+WIPBR=2,5,1,"passwd" //set password
OK
AT+WIPBR=4,5,0 //start GSM bearer
OK
AT+WIPCREATE=3,1,80,5,8 //create the server on port 80, idx = 1. The server
//is listening for connection request on port
//80.Spawned sockets will be given the index 5,
//6, 7 and 8. It will accept connection request
//until it has no more socket left.
+WIPACCEPT: 1,5 //unsolicited: the server accepted a connection
//resulting TCP client on idx 5
AT+WIPDATA=2,5,1 //exchange data on socket idx 5
CONNECT
... //read, write
+++ //switch to AT mode
OK
AT+WIPCLOSE=2,5 //close the TCP client socket index 5
OK

```

9.1.2. TCP Client Socket

9.1.2.1. Using GPRS Bearer

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=2,1,"ip addr",80 //create a TCP client towards peer IP device @ "ip
//addr", port 80.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 2,1 //unsolicited: the TCP client socket is connected
//to the peer
AT+WIPDATA=2,1,1 //exchange data on socket idx 1:
CONNECT
... //read, write
+++ //switch to AT mode
OK
AT+WIPCLOSE=2,1 //close the TCP client socket index 1
OK

```

9.1.2.2. Using GSM Bearer

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,5 //open GSM bearer
OK
AT+WIPBR=2,5,2,"Phone number" //set phone number for GSM bearer
OK
AT+WIPBR=2,5,0,"user name" //set user name
OK
AT+WIPBR=2,5,1,"passwd" //set password
OK
AT+WIPBR=4,5,0 //start GSM bearer
OK
AT+WIPCREATE=2,1,"ip addr",80 //create a TCP client towards peer IP device @ "ip
//addr", port 80
OK //all parameters and IP stack behavior are OK
+WIPREADY: 2,1 //unsolicited: the TCP client socket is connected to
//the peer
AT+WIPDATA=2,1,1 //exchange data on socket idx 1
CONNECT
... //read, write
+++ //switch to AT mode
OK
AT+WIPCLOSE=2,1 //close the TCP client socket index 1
OK

```

9.2. UDP Socket

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=1,1,80,"www.sierrawireless.com",80 //create a UDP client towards peer IP device @
//"www.sierrawireless.com", port 80
OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,1 //unsolicited: the UDP client socket is "pseudo"
//connected to the peer (no //real connection is
UDP)

AT+WIPDATA=1,1,1 //exchange data on socket idx 1:
CONNECT
... //read, write
+++ //switch to AT mode
OK
AT+WIPCLOSE=1,1 //close the UDP socket index 1
OK
AT+WIPCREATE=1,1,1234 //start a UDP server and listen for datagram on port
//1234
OK //all parameters and IP stack //behavior are OK

+WIPREADY: 1,1 //unsolicited: the UDP client socket is "pseudo"
//connected to the peer (no real connection is UDP)

+WIPDATA: //one datagram is ready to be read : it was sent
1,1,25,"192.168.0.2",2397 from //192.168.0.2 on port //2397 and is composed
of 25 //bytes
AT+WIPDATA=1,1,1

```

```
CONNECT
abcdedghijklmnopqrstuvwxyz [ETX] //here 25 bytes + the [ETX] character (marking the
//bound of the datagram) have been read.

+++ or AT+WIPDATA=1,1,0 //type on this UART "+++" escape sequence or un
//map the UART on other control port (USB UART)

OK //here UART is back to AT command mode. If some
//other remote IP devices sent some one or more
//datagrams while reading for the first one, then a
//new datagram indication is received

+WIPDATA: //one datagram is ready to be read : it was sent
1,1,50,"192.168.0.4",58 from //192.168.0.4 on port 58 and is composed of
50 //bytes

AT+WIPDATA=1,1,1

CONNECT
abcdedghijklmnopqrstuvwxyzabcd //here 25 bytes + the [ETX] character (marking the
ghijklmnopqrstuvwxyz [ETX] //bound of the datagram) have been read.
```

9.3. PING

```
AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPPING="192.168.0.1" //start PING session
OK
+WIPPING:0,0,224
```

9.4. FTP

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=4,1,"FTP server",21,"username","passwd" //create FTP session
OK
AT+WIPFILE=4,1,2,"./filename.txt" //upload file "filename.txt"
CONNECT
<data>
[ETX]
OK
AT+WIPFILE=4,1,1,"./filename.txt" //download file "filename.txt"
CONNECT
<data>
[ETX]
OK

```


9.5. FTP DATA Offline

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=4,1,"FTP server",21,"username","passwd" //create FTP session
OK
AT+WIPFILE=4,1,6,"./filename.txt" //upload file "filename.txt"
+WIPDATARW: 1,1
OK
AT+WIPDATARW=1,1,"30313233343536373839" //upload data file
+WIPDATARW: 3,1,10 // 10 data bytes sent
+WIPDATARW: 0,1 // WIPDATARW session closes on channel idx 1
OK
AT+WIPFILE=4,1,7,"./filename.txt" //download and dump "filename.txt" file
+WIPDATARW: 2,1,2,1,"3132333435 [...]"
+WIPDATARW: 2,1,2,2,"3132333435 [...]"
+WIPDATARW: 0,1
OK

```

9.6. HTTP

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=5,1,"www.siteaddress //connect to remote HTTP proxy server port 81
.com",81,"username","password","h //with authentication and some header fields
header name"," header value"
OK
+WIPREADY: 5,1 //connection and authentication are successful
AT+WIPOPT=5,1,1,51 //get size of the TCP send buffer size
+WIPOPT:5,51,<sender buffer size>
OK //get option successful
AT+WIPOPT=5,1,2,53,6 //set maximum number of redirects
OK
AT+WIPFILE=5,1,1,"urlForGet","use //HTTP GET method
rname","password","Accept","text/
html","Transfer-
codings","compress"
CONNECT
<user starts getting the mail
with the UART in data mode and
ends with an [ETX] >
OK
+WIPFILE: 5,1,1,255,"Found" //unsolicited string on the HTTP status code
//and reason

```

9.7. SMTP

```
AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=6,1,"192.168.1.2",25 //connect to remote SMTP server
,"user","password"
OK
+WIPREADY: 6,1 //connection and authentication are successful
```

```
AT+WIPOPT=6,1,2,61,"sender@mail.com" //set sender mail address
om"
OK
AT+WIPOPT=6,1,2,62,"sender name" //set sender name
OK
AT+WIPOPT=6,1,2,63," //set receiver mail address
rec01@mail.com, rec02@mail.com"
OK
AT+WIPOPT=6,1,2,64,"ccrec01@mail. //set CC receiver mail address
com, ccrec02@mail.com"
OK
AT+WIPOPT=6,1,2,65,"bccrec01@mail //set BCC mail address
.com, bccrec02@mail.com"
OK
AT+WIPOPT=6,1,2,66,"mail subject" //set mail subject
OK
AT+WIPFILE=6,1,2 //send mail
CONNECT
<user starts sending mail with
the UART in data mode and ends
with an [ETX] character >
OK
```

9.8. POP3

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=7,1,"192.168.1.2",11 //connect to remote POP3 server
0,"user","password"
OK
//connection and authentication are successful
+WIPREADY: 7,1
AT+WIPOPT=7,1,1,71 //get total number of mails
+WIPOPT: 7,71,10
OK
AT+WIPOPT=7,1,1,72 //get total mail size
+WIPOPT: 7,72,124000
OK
AT+WIPFILE=7,1,1,"5" //retrieve mail id 5
CONNECT
<user starts getting the mail
with the UART in data mode and
ends with an [ETX] >
OK

```

```
AT+WIPFILE=7,1,3,"1" //retrieve mail id 1 and delete it from the server  
//after retrieving  
CONNECT  
  
<user starts getting the mail  
with the UART in data mode and  
ends with an [ETX] >  
  
OK
```

9.9. MMS

Example of sending an MMS with multiple recipients and multiple files with the same extensions.

Please note that files are not buffered, but sent directly to the MMS Server.

Detailed information about the files is needed for the headers before and must be set for each file using WIPOPT before sending the file via WIPFILE.

```
AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"orange.fr" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=8,1, //create the connection to the MMS server.
"192.168.10.200",8080,
"http://mms.orange.fr"
OK
AT+WIPOPT=8, 1, 2, 82, "Mr,Smith //add email address to the TO field.
<smith@example.com>"
OK
```

```

AT+WIPOPT=8,1,2,83,"+33623456789" //add telephone number to the CC field.
OK
AT+WIPOPT=8,1,2,83,"0654321987" //add telephone number to the CC field.
OK
AT+WIPOPT=8,1,2,86, "X <x@y.com>" // add email address to the BCC field.
OK
AT+WIPOPT=8,1,2,92,1, // set the multipart type to Related and specify
"application/smil", "<001>" // that the first file that should be read is a type
OK // SMIL with the content-id "<001>"
AT+WIPOPT=8,1,2,93,9,100, // add a SMIL multimedia presentation file of
"1.smil", "<001>" // size 100 Bytes with filename "1.smil" and
OK // content-id "<001>"
AT+WIPOPT=8,1,2,93,2,222,"2.txt", // add a text file of type USC2 of size 222 bytes
OK // with the filename "2.txt" but no content-id.
AT+WIPOPT=8,1,2,93,3,304, ,<003>" // add a text file of type ASCII of size 304 Bytes
OK // with no filename but content-id "<003>".
AT+WIPOPT=8,1,2,93,4,1024,"4.jpeg" // add a JPEG picture of size 1024 Bytes with
, "<004>" // the filename "4.jpeg" and content-id "<004>".
OK
AT+WIPOPT=8,1,2,93,5,2048,"5.gif", // add a GIF picture of size 2048 Bytes with
"<005>" // the filename "5.gif" and content-id "<005>".
OK
AT+WIPOPT=8,1,2,94,"audio/xyz", // add a file of a content type specified in the
128,"6.xyz", "<XYZ>" // string of size 128 Bytes with the filename
OK // "6.xyz" and content-id "<XYZ>".

// NOW SEND THE CONTENT OF THE FILES IN
// THE SAME ORDER!

AT+WIPFILE=8,1,2,"1.smil", "<001>" // send the SMIL file previously specified by
OK // WIPOPT.

CONNECT

<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >

OK

```



```

AT+WIPFILE=8,1,2,"2.txt", // send the TXT file previously specified by
                               WIPOPT.
CONNECT
<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >
OK
AT+WIPFILE=8,1,2,,"<003>" // send the TXT file previously specified by
                               WIPOPT.
CONNECT
<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >
OK
AT+WIPFILE=8,1,2,"4.jpeg", "<004>" // send the JPEG previously specified by
                                       WIPOPT.
CONNECT
<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >
OK
AT+WIPFILE=8,1,2,"5.gif", "<005>" // send the GIF previously specified by
                                       WIPOPT.
CONNECT
<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >
OK
AT+WIPFILE=8,1,2,"6.xyz", "<XYZ>" // send the XYZ file previously specified by
                                       WIPOPT.
CONNECT
<user starts sending the file with
the UART in data mode and ends
with an [ETX] character >
OK
AT+WIPOPT=8,1,2,96 // send WIP_MMS_DONE to signal that the
OK // users has sent the last file. This is to avoid
// deadlock errors where the user missed to send
// a file.
+WIPACCEPT: 1,5 //unsolicited: the server accepted a connection
// and the MMS was sent successfully.

AT+WIPCLOSE=8,1 // Close the connection to the server.
OK

```

9.10. Creating a TCP Server, spawning the maximum TCP Socket (for the configured Server)

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=3,1,80,5,6 //create the server on port 80, idx = 1. The
//server is listening for connection request on
//port 80.Spawned sockets will be given the
//index 5 or 6. It will accept connection request
//until it has no more socket left.
+WIPACCEPT: 1,5 //unsolicited: the server accepted a connection
//resulting TCP client on idx 5.
+WIPACCEPT: 1,6 //unsolicited: the server accepted a connection
//resulting TCP client on idx 6.
AT+WIPCLOSE=2,5 //close the spawned TCP client socket index 5.
OK //now if the peer device try to connect to the
//server it shall receive an accept () immediately
//followed by an shutdown() (connection reset
//by peer)

```

9.11. Creating a Server and try to create a TCP Client/Server on a reserved index (reserved by the Server) will fail.

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=3,2,80,1,2 //create the server on port 80, idx=2. The server
//is listening for connection request on port 80.
//Spawned sockets will be given the index 1 or
//2.It will accept connection request until has
//nor more socket left.
OK
AT+WIPCREATE=2,3,"198.168.0.1",80 //create a TCP client towards peer IP device @
//"198.168.0.1", port 80,
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 2,3 //unsolicited: the TCP client socket is connected
//to the peer.
+WIPACCEPT: 2,1 //unsolicited: the server index accepted a
//connection; resulting TCP client on idx 1
AT+WIPDATA=2,3,1 //exchange data on socket index 3
CONNECT
AT+WIPDATA=2,1,1 //exchange data on socket index 1
CONNECT
[ETX] //send unescaped ETX character
+WIPPEERCLOSE: 2,3 //unsolicited: peer socket is closed
AT+WIPCLOSE=3,1 //close TCP server socket index 1
OK

```

```
AT+WIPCREATE=3,2,81,2,3
```

```
//create the server on port 81, idx=2 and  
from_idx=2 and to_idx=3
```

```
+CME ERROR:845
```

```
//TCP client socket with idx 2 was reserved by  
//the previous server socket and it was not  
//closed explicitly. Hence error is returned.
```

9.12. Create a TCP Client and try to create a TCP Server with indexes range containing TCP Client will fail.

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=2,1,"198.168.0.1",80 //create a TCP client towards peer IP device @
//"198.168.0.1", port 80
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 2,1 //unsolicited: the TCP client socket is connected
//to the peer.
AT+WIPCREATE=3,2,80,1,2 //create the server on port 80, idx=2. Range
//requested contains the already used index
//"1" and hence error is returned.
+CME ERROR: 845

```

9.13. Creating 8 UDP sockets, 8 TCP clients and 4 TCP servers.

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=1,1,55,"192.168.0.1" //create a UDP client towards peer IP device @
,75 //"192.168.0.1", port 75.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,1 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,2,56,"192.168.0.1" //create a UDP client towards peer IP device @
,76 //"192.168.0.1", port 76.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,2 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,3,57,"192.168.0.1" //create a UDP client towards peer IP device @
,77 //"192.168.0.1", port 77.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,3 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,4,58,"192.168.0.1" //create a UDP client towards peer IP device @
,78 //"192.168.0.1", port 78.

```

```

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,4 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,5,59,"192.168.0.1" //create a UDP client towards peer IP device @
,79 // "192.168.0.1", port 79.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,5 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,6,60,"192.168.0.1" //create a UDP client towards peer IP device @
,80 // "192.168.0.1", port 80.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,6 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,7,61,"192.168.0.1" //create a UDP client towards peer IP device @
,81 // "192.168.0.1", port 81

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,7 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,8,62,"192.168.0.1" //create a UDP client towards peer IP device @
,82 // "192.168.0.1", port 82.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,8 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,9,63,"192.168.0.1"
,83

+CME ERROR: 830 //8 UDP sockets have been created and hence
//9th attempt fails

AT+WIPCREATE=3,1,80,1,1 //create one server on port 80, idx = 1. One
//TCP client socket is reserved on index 1

OK

AT+WIPCREATE=3,2,81,2,2 //create one server on port 81, idx = 2. One
//TCP client socket is reserved on index 2

OK

AT+WIPCREATE=3,3,82,3,3 //create one server on port 82, idx = 3. One
//TCP client socket is reserved on index 3

OK

AT+WIPCREATE=3,4,83,4,4 //create one server on port 83, idx = 4. One
//TCP client socket is reserved on index 4

OK

AT+WIPCREATE=3,5,84,5,5 //4 TCP servers have been created and hence
//creation of 5th TCP server socket fails

+CME ERROR: 830

```

```

AT+WIPCREATE=2,1,"192.168.0.1",80 //create a TCP client socket towards peer IP
+CME ERROR: 845 //device @ "192.168.0.1", port 80. Index 1 is
//reserved by server index and hence error is
//returned.
//4 reserved TCP client sockets have been
//spawned by their TCP server.

+WIPACCEPT: 1,1 //unsolicited: the server index 1 accepted a
//connection; resulting TCP client on idx 1

+WIPACCEPT: 2,2 //unsolicited: the server index 2 accepted a
//connection; resulting TCP client on idx 2

+WIPACCEPT: 3,3 //unsolicited: the server index 3 accepted a
//connection; resulting TCP client on idx 3

+WIPACCEPT: 4,4 //unsolicited: the server index 4 accepted a
//connection; resulting TCP client on idx 4

AT+WIPCREATE=2,5,"192.168.0.1",80 //create a TCP client towards peer IP device @
OK // "192.168.0.1", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,5 //unsolicited: the TCP client socket is connected
//to the peer.

AT+WIPCREATE=2,6,"192.168.0.1",80 //create a TCP client towards peer IP device @
OK // "192.168.0.1", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,6 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,7,"192.168.0.1",80 //create a TCP client towards peer IP device @
OK // "192.168.0.1", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,7 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,8,"192.168.0.1",80 //create a TCP client towards peer IP device @
OK // "192.168.0.1", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,8 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,8,"192.168.0.1",80 //create a TCP client towards peer IP device @
+CME ERROR: 840 // "192.168.0.1", port 80. Index 8 is already
//used and corresponds to an active socket.

AT+WIPCREATE=2,9,"192.168.0.1",80 //create a TCP client towards a peer IP device @
+CME ERROR: 830 // "192.168.0.1", port 80. Index 9 is forbidden.

```


9.14. Changing the MAX_SOCKET_NUM option value and try to create 8 UDP sockets, 8 TCP Client sockets and 4 TCP Server sockets.

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPCFG=2,6,3 //MAX_SOCKET_NUM has been changed to 3
OK
AT+WIPCFG=4,1 //save the changed configuration to flash
OK
AT+WIPCFG=0 //close the IP stack
OK
AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=1,1,55,"192.168.0.1" //create a UDP client towards peer IP device @
,75 //"192.168.0.1", port 75.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,1 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,2,56,"192.168.0.1" //create a UDP client towards peer IP device @
,76 //"192.168.0.1", port 76.

```

| | |
|--|--|
| OK | <i>//all parameters and IP stack behavior are OK.</i> |
| +WIPREADY: 1,2 | <i>//unsolicited: the UDP client socket is "pseudo //connected to the peer (no real connection is // UDP)</i> |
| AT+WIPCREATE=1,3,57,"192.168.0.1" ,77 | <i>//create a UDP client towards peer IP device @ //"192.168.0.1", port 77.</i> |
| OK | <i>//all parameters and IP stack behavior are OK.</i> |
| +WIPREADY: 1,3 | <i>//unsolicited: the UDP client socket is "pseudo //connected to the peer (no real connection is // UDP)</i> |
| AT+WIPCREATE=1,4,58,"192.168.0.1" ,78 | <i>//create a UDP client towards peer IP device @ //"192.168.0.1", port 78.</i> |
| +CME ERROR: 838 | <i>//maximum 3 sockets can be created as the //MAX SOCK_NUM value has been changed to //3. Hence an attempt to create a fourth socket //returns error.</i> |

9.15. Creating 8 UDP sockets, 8 TCP Clients, 4 TCP Servers and either one FTP/HTTP/SMTP/POP3

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=1,1,55,"192.168.0.1" //create a UDP client towards peer IP device @
,75 // "192.168.0.1", port 75.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,1 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,2,56,"192.168.0.1" //create a UDP client towards peer IP device @
,76 // "192.168.0.1", port 76.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,2 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,3,57,"192.168.0.1" //create a UDP client towards peer IP device @
,77 // "192.168.0.1", port 77.
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 1,3 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)
AT+WIPCREATE=1,4,58,"192.168.0.1" //create a UDP client towards peer IP device @
,78 // "192.168.0.1", port 78.

```

```

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,4 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,5,59,"192.168.0.1" //create a UDP client towards peer IP device @
,79 // "192.168.0.1", port 79.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,5 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,6,60,"192.168.0.1" //create a UDP client towards peer IP device @
,80 // "192.168.0.1", port 80.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,6 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,7,61,"192.168.0.1" //create a UDP client towards peer IP device @
,81 // "192.168.0.1", port 81

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,7 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,8,62,"192.168.0.1" //create a UDP client towards peer IP device @
,82 // "192.168.0.1", port 82.

OK //all parameters and IP stack behavior are OK

+WIPREADY: 1,8 //unsolicited: the UDP client socket is "pseudo
//connected to the peer (no real connection is
// UDP)

AT+WIPCREATE=1,9,63,"192.168.0.1"
,83

+CME ERROR: 830 //8 UDP sockets have been created and hence
//9th attempt fails

AT+WIPCREATE=3,1,83,1,1 //create one server on port 83, idx = 1. One
//TCP client socket is reserved on index 1

OK

AT+WIPCREATE=3,2,84,2,2 //create one server on port 84, idx = 2. One
//TCP client socket is reserved on index 2

OK

AT+WIPCREATE=3,3,85,3,3 //create one server on port 85, idx = 3. One
//TCP client socket is reserved on index 3

OK

AT+WIPCREATE=3,4,86,4,4 //create one server on port 86, idx = 4. One
//TCP client socket is reserved on index 4

OK

AT+WIPCREATE=3,5,84,5,5 //4 TCP servers have been created and hence
//creation of 5th TCP server socket fails

+CME ERROR: 830

```

```

AT+WIPCREATE=2,1,"192.168.0.1",83 //4 TCP server have been created and each of
+CME ERROR: 845 //them reserved 1 TCP client socket and hence
//5th attempt of creating TCP server fails

//4 reserved TCP client sockets have been
//spawned by their TCP server.

+WIPACCEPT: 1,1 //unsolicited: the server index 1 accepted a
//connection; resulting TCP client on idx 1

+WIPACCEPT: 2,2 //unsolicited: the server index 2 accepted a
//connection; resulting TCP client on idx 2

+WIPACCEPT: 3,3 //unsolicited: the server index 3 accepted a
//connection; resulting TCP client on idx 3

+WIPACCEPT: 4,4 //unsolicited: the server index 4 accepted a
//connection; resulting TCP client on idx 4

AT+WIPCREATE=2,5,"192.168.0.2",80 //create a TCP client towards peer IP device @
OK //"192.168.0.2", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,5 //unsolicited: the TCP client socket is connected
//to the peer.

AT+WIPCREATE=2,6,"192.168.0.2",80 //create a TCP client towards peer IP device @
OK //"192.168.0.2", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,6 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,7,"192.168.0.2",80 //create a TCP client towards peer IP device @
OK //"192.168.0.2", port 80
//all parameters and IP stack behavior are OK

+WIPREADY: 2,7 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,8,"192.168.0.2",80 //create a TCP client towards peer IP device @
OK //"192.168.0.2", port 80.
//all parameters and IP stack behavior are OK

+WIPREADY: 2,8 //unsolicited: the TCP client socket is connected
//to the peer

AT+WIPCREATE=2,8,"192.168.0.2",80 //create a TCP client towards peer IP device @
+CME ERROR: 840 //"192.168.0.2", port 80. Index 8 is already
//used and corresponds to an active socket.

AT+WIPCREATE=2,9,"192.168.0.2",80 //create a TCP client towards a peer IP device @
+CME ERROR: 830 //"192.168.0.2", port 80. Index 9 is forbidden.

AT+WIPCREATE=4,1,"ftp //create FTP session using default port 21
server",,"user name","password"

OK //FTP session is created successfully.

AT+WIPCREATE=7,1,"POP3
server",,"user name","mail id"

+CME ERROR: 840 //attempt of creating a OP3 session returns an
//error as already 1 FTP session is active.

AT+WIPCLOSE=4,1 //close FTP session

```

```
OK
+WIPPEERCLOSE: 4,1 //unsolicited: FTP session is closed
//successfully
AT+WIPCREATE=7,1,"POP3 //create POP3 session using default port 110
server",,"user name","mail id"
OK //all parameters and IP stack behaviors are OK.
+WIPREADY: 7,1 //unsolicited: the POP3 session is created
//successfully
```

9.16. Subscribe/Unsubscribe WIPsoft AT commands using WIPsoft Library API

```
#include "adl_global.h"    // Global includes
#include "wip_atcmd.h"     // WIP AT command services
#if __OAT_API_VERSION__ >= 400
const u16 wm_apmCustomStackSize = 4096;
#else
u32 wm_apmCustomStack[1024];
const u16 wm_apmCustomStackSize = sizeof(wm_apmCustomStack);
#endif

void adl_main ( adl_InitType_e InitType )
{
    TRACE (( 1, "Embedded Application : Main" ));
    /* subscribe to the +WIP AT commands set service */
    if ( wip_ATCmdSubscribe() == 0) {
        /* The customer can write here its own application based on other
           plug -ins or its specific application target. */
        wip_ATCmdUnsubscribe();
    }
    else
    {
        /* Error while subscribing to WIPsoft library */
    }
}
```

9.17. Creating TCP client and server sockets in the same Wireless CPU at the same time mapping or unmapping the UART to exchange the data between the sockets

```

AT+WIPCFG=1 //start IP stack
OK
AT+WIPBR=1,6 //open GPRS bearer
OK
AT+WIPBR=2,6,11,"APN name" //set APN name of GPRS bearer
OK
AT+WIPBR=2,6,0,"user name" //set user name
OK
AT+WIPBR=2,6,1,"passwd" //set password
OK
AT+WIPBR=4,6,0 //start GPRS bearer
OK
AT+WIPCREATE=3,2,80,1,2 //create the server on port 80, idx=2. The server
//is listening for connection request on port 80.
//Spawned sockets will be given the index 1 or
//2.It will accept connection request until has
//nor more socket left.
AT+WIPCREATE=2,3,"198.168.0.1",80 //create a TCP client towards peer IP device @
//"198.168.0.1", port 80,
OK //all parameters and IP stack behavior are OK.
+WIPREADY: 2,3 //unsolicited: the TCP client socket is connected
//to the peer.
+WIPACCEPT: 2,1 //unsolicited: the server index accepted a
//connection; resulting TCP client on idx 1
AT+WIPDATA=2,3,1 //exchange data on socket index 3
CONNECT
abc+++ //data sent to socket index 1 and switched to
AT mode by giving +++
OK
AT+WIPDATA=2,1,1 //exchange data on socket index 1
CONNECT
abc+++ //data received from socket index 3
OK

```


>> 10. Error Codes

10.1. General CME Error Codes

| “+CMEE” AT error code | Description |
|------------------------------|---|
| 800 | invalid option |
| 801 | invalid option value |
| 802 | not enough memory |
| 803 | operation not allowed in the current WIP stack state |
| 804 | device already open |
| 805 | network interface not available |
| 806 | operation not allowed on the considered bearer |
| 807 | bearer connection failure : line busy |
| 808 | bearer connection failure : no answer |
| 809 | bearer connection failure : no carrier |
| 810 | bearer connection failure : no sim card present |
| 811 | bearer connection failure : sim not ready (no pin code entered, ...) |
| 812 | bearer connection failure : GPRS network failure |
| 813 | bearer connection failure : PPP LCP negotiation failed |
| 814 | bearer connection failure : PPP authentication failed |
| 815 | bearer connection failure : PPP IPCP negotiation failed |
| 816 | bearer connection failure : PPP peer terminates session |
| 817 | bearer connection failure : PPP peer does not answer to echo request |
| 818 | incoming call refused |
| 819 | error on Ping channel |
| 820 | error writing configuration in FLASH memory |
| 821 | error reading configuration in FLASH memory |
| 822-829 | reserved for future use |
| 830 | bad index |
| 831 | bad state |
| 832 | bad port number |
| 833 | bad port state |
| 834 | not implemented |
| 835 | option not supported |
| 836 | memory allocation error |
| 837 | bad protocol |
| 838 | no more free socket |
| 839 | error during channel creation |
| 840 | UDP/TCP socket or FTP/HTTP/SMTP/POP3 session is already active |
| 841 | peer closed, or error in the FTP connection |
| 842 | destination host unreachable (whether host unreachable, Network unreachable, response timeout) |

| “+CMEE” AT error code | Description |
|-----------------------|---|
| 843 | connection reset by peer |
| 844 | stack already started |
| 845 | attempt is made to reserve/create a client socket which is already reserved/opened by TCP server/client |
| 846 | internal error: FCM subscription failure |
| 847 | bearer connection failure: WIP_BOPT_GPRS_TIMEOUT time limit expired before GPRS bearer connected |
| 848 | impossible to connect to the bearer |
| 849 | connection to the bearer has succeeded but a problem has occurred during the data flow establishment |
| 850 | invalid channel option or parameter value (for example, HTTP user name too long) |
| 851 | specified parameters to the command is more or less than the maximum number of mandatory parameters |
| 852 | IP stack not initialized |
| 853 | Data offline buffer filled |
| 854-859 | reserved for future use |
| 860 | protocol undefined or internal error |
| 861 | username rejected by server |
| 862 | password rejected by server |
| 863 | delete error |
| 864 | list error |
| 865 | authentication error |
| 866 | server not ready error |
| 867 | POP3 email retrieving error |
| 868 | POP3 email size error |
| 869-879 | reserved for future use |
| 880 | SMTP sender email address rejected by server |
| 881 | SMTP recipient email address rejected by server |
| 882 | SMTP CC recipient email address rejected by server |
| 883 | SMTP BCC recipient email address rejected by server |
| 884 | SMTP email body send request rejected by server |
| 890 | Service denied |
| 891 | Message format corrupt |
| 892 | Address unresolved |
| 893 | Message not found |
| 894 | Network problem |
| 895 | Content not accepted |
| 896 | Unsupported message |
| 897 | Unspecified error |

10.2. GPRS CME Error Codes

| <error> | Meaning | Resulting from the following commands |
|----------------------|---|--|
| 103 | Incorrect MS identity.(#3) | +CGATT |
| 132 | Service option not supported (#32) | +CGACT +CGDATA ATD*99 |
| 133 | Requested service option not subscribed (#33) | +CGACT +CGDATA ATD*99 |
| 134 | Service option temporarily out of order (#26, #34, #38) | +CGACT +CGDATA ATD*99 |
| 148 | Unspecified GPRS error | All GPRS commands |
| 149 | PDP authentication failure (#29) | +CGACT +CGDATA ATD*99 |
| 150 | Invalid mobile class | +CGCLASS +CGATT |

 **Index**

+WIPBR, 25
+WIPCFG, 16
+WIPCLOSE, 44
+WIPCREATE, 35
+WIPDATA, 68
+WIPFILE, 58
+WIPOPT, 47
+WIPPING, 84
FTP, 96
HTTP, 98
PING, 95
POP3, 101
SMTP, 99
TCP Socket, 89
UDP Socket, 93
wip_ATCmdSubscribe, 87
wip_ATCmdUnsubscribe, 88



SIERRA
WIRELESS