



## 1. Introduction

The Test Step function “Communication over *network socket*” allows sending data from ComGage to a network server. The sent data is configured in several text files that may contain wildcards, which ComGage replaces with the proper value while the test step function is executed.

When the function is called, the user can select one of the configured destination servers from a list and enter a comment which may be added to the sent data using its specific wildcard ( see below ).

### Important notes:

- This test step function will be executed in a test order only !
- This test step function requires a License for Module 73 to operate.
- ComGage V4.12 is required to use this test step function.

## 2. Configuration

The function needs to be added to a Test Step of a ComGage Test Scheme. The setup button of this function opens the following dialog :

This dialog allows to configure the following parameters :

- **TCP client / TCP server**  
Allows to configure the SFct065 acting as TCP client ( connect to 1 of 5 TCP servers ) or as TCP server ( allows connection of TCP clients to configured port in measuring mode ).
- **Destination 1..5**  
Allows to configure *Description*, *Hostname / IP Address* and *TCP port* for up to 5 different destinations if configured as TCP client. The button “Test” allows testing the communication with the remote destination using all the settings configured below. The received answer will be shown in an info dialog.  
If configured as TCP server, only the server port can be configured.
- **HTTP Header**  
File name of text file containing the HTTP header ( e.g. a HTTP post message ).<sup>1,2,3,4</sup>

<sup>1</sup> Wildcards in this file are replaced by the proper value while the test step function is executed.

<sup>2</sup> The button [...] opens a dialog which allows to select a txt file.

<sup>3</sup> The [Edit] button opens the configured file for editing with the default system editor.

<sup>4</sup> The length of the file name – including the path - is limited to 200 characters.



- **Content Header**  
File name of text file with the message content header.<sup>1,2,3,4</sup>
- **Content**  
File name of text file with the message content.<sup>1,2,3,4</sup>
- **Content Footer**  
File name of text file with the message content footer.<sup>1,2,3,4</sup>  
**Note :**  
You do not need to define all 4 files. If you do not fill in all filenames, then only the files with valid filenames are sent.
- **Answer search string**  
If configured, this string will be searched inside the received answer.
- **Retries**  
Number of retries [1..255] before the function returns an error.
- **Answer timeout**  
Interval in seconds [1..255] within the system expects an answer, otherwise the systems assumes a failed attempt.
- **Result Register**  
Contains the result of the function execution.
  - (0) – function executed successfully.
  - (1) – Socket error. The function could not establish a connection to the selected destination.
  - (2) – Timeout occurred while waiting for an answer from network server.
  - (3) – The received answer does not contain the search string.
  - (4) – No clients are connected ( only on TCP server ).
- **Show error messages**  
When enabled, error messages are shown when the function is executed.
- **Debug messages**  
When enabled, the function generates debug output of its last call into the file SFct065\_debug.log in the configured ComGage directory for test orders.

### 3. Function Execution dialog

If the function is configured as TCP client and if the function is executed, a dialog opens which allows the user to select one of the destinations. Also the user can enter a comment. By clicking OK, the configured information will be sent to the selected network server :

When the user clicks on **X** no data will be sent.

### 4. Wildcards

The following table contains the list of available wildcards :

Wildcard	Function
<\$HL>	Length of Content Header + Content + Content Footer in Bytes
<\$HC>	Comment ( is be entered by the user before sending the data )
Wildcard	Function
<\$ON>	Test order number
<\$ON\$-B>	Test order number ( blank characters are removed )
<\$ON\$L15>	Test order number ( length=15 bytes / missing bytes=blank )
<\$AR>	Article number
<\$AR\$-B>	Article number ( blank characters are removed )
<\$AR\$L15>	Article number ( length=15 bytes / missing bytes=blank )
<\$AN>	Article name
<\$AN\$-B>	Article name ( blank characters are removed )
<\$AN\$L15>	Article name ( length=15 bytes / missing bytes=blank )
<\$DY>	Output Date : Year ( 4 digits )
<\$DM>	Output Date: Month ( 2 digits )
<\$DD>	Output Date: Day ( 2 digits )
<\$TH>	Output Date: Hour ( 2 digits )
<\$TM>	Output Date: Minute ( 2 digits )
<\$TS>	Output Date: Second ( 2 digits )

Measuring values :

Wildcard	Function
<\$C1..C128\$RB>	Marks the beginning of a measuring value output block for outputting the complete table of measuring values of one characteristic. The content of the output block will be repeated for each saved measuring value of the selected characteristic. When additional characteristics within this block are configured for output, they will use the same index than the characteristic.
<\$C1..C128\$RE>	End of a measuring value output block.
<\$C1..C128\$RI>	Index ( = number of the measuring value ) of measuring value within the output block, starting from 1.

Wildcard	Function
<\$C1..128\$NA>	Characteristic 1..128 : Name
<\$C1..128\$NA\$-B>	Characteristic 1..128 : Name ( blank characters are removed )
<\$C1..128\$NA\$L15>	Characteristic 1..128 : Name ( length=15 bytes / missing bytes=blank )
<\$C1..128\$UN>	Characteristic 1..128 : Unit
<\$C1..128\$UN\$-B>	Characteristic 1..128 : Unit ( blank characters are removed )
<\$C1..128\$UN\$L15>	Characteristic 1..128 : Unit ( length=15 bytes / missing bytes=blank )
<\$C1..128\$NS>	Characteristic 1..128 : Nominal size
<\$C1..128\$NS\$L12>	Characteristic 1..128 : N.S. ( length=12 bytes / missing bytes="0" )
<\$C1..128\$US>	Characteristic 1..128 : Upper specification limit ( relative to nom. size )
<\$C1..128\$US\$L12>	Characteristic 1..128 : USL ( length=12 bytes / missing bytes="0" )
<\$C1..128\$UC>	Characteristic 1..128 : Upper controlling limit ( relative to nom. size )
<\$C1..128\$UC\$L12>	Characteristic 1..128 : UCL ( length=12 bytes / missing bytes="0" )
<\$C1..128\$LC>	Characteristic 1..128 : Lower controlling limit ( relative to nom. size )
<\$C1..128\$LC\$L12>	Characteristic 1..128 : LCL ( length=12 bytes / missing bytes="0" )
<\$C1..128\$LS>	Characteristic 1..128 : Lower specification limit ( relative to nom. size )
<\$C1..128\$LS\$L12>	Characteristic 1..128 : LSL ( length=12 bytes / missing bytes="0" )
<\$C1..128\$UT>	Characteristic 1..128 : Upper tolerance limit ( \$NS + \$US )
<\$C1..128\$UT\$L12>	Characteristic 1..128 : UT ( length=12 bytes / missing bytes="0" )
<\$C1..128\$LT>	Characteristic 1..128 : Lower tolerance limit ( \$NS + \$LS )
<\$C1..128\$LT\$L12>	Characteristic 1..128 : LT ( length=12 bytes / missing bytes="0" )
<\$C1..128\$M1>	Characteristic 1..128 : Master Value 1
<\$C1..128\$M1\$L12>	Characteristic 1..128 : Master 1 ( length=12 bytes / missing bytes="0" )
<\$C1..128\$M2>	Characteristic 1..128 : Master Value 2
<\$C1..128\$M2\$L12>	Characteristic 1..128 : Master 2 ( length=12 bytes / missing bytes="0" )
<\$C1..128\$CO\$L15>	Characteristic 1..128 : Cal.Offset ( length=15 bytes / missing bytes="0" )
<\$C1..128\$CF\$L15>	Characteristic 1..128 : Cal.Factor ( length=15 bytes / missing bytes="0" )
<\$C1..128\$NO>	Characteristic 1..128 : Note
<\$C1..128\$NO\$-B>	Characteristic 1..128 : Note ( blank characters are removed )
<\$C1..128\$NO\$L15>	Characteristic 1..128 : Note ( length=15 bytes / missing bytes=blank )

# ComGage – Test step function SFct065

## “Communication over network socket”



Last measuring value from file :

Wildcard	Function
<\$C1..128\$MV\$5>	Characteristic 1..128 : Measuring value \$5 : Number of decimal places No entry = 6 decimal places \$0 = no decimal places \$1 ... 5 = number of decimal places according to numeral
<\$C1..128\$MV\$5\$L12>	Characteristic 1..128 : Value ( length=12 bytes / missing bytes="0" )
<\$C1..128\$RS\$x\$y\$z>	Characteristic 1..128 : Tolerance-result of last measuring value x : Export-Text on value within controlling limits or within tolerance limits on deactivated controlling limits. y : Export-Text on value outside controlling limits but within tolerance limits z : Export-Text on value outside tolerance limits
<\$C1..128\$DY>	Characteristic 1..128 : Date of measuring value : Year ( 4 digits )
<\$C1..128\$DM>	Characteristic 1..128 : Date of measuring value : Month ( 2 digits )
<\$C1..128\$DD>	Characteristic 1..128 : Date of measuring value : Day ( 2 digits )
<\$C1..128\$TH>	Characteristic 1..128 : Time of measuring value : Hour ( 2 digits )
<\$C1..128\$TM>	Characteristic 1..128 : Time of measuring value : Minute ( 2 digits )
<\$C1..128\$TS>	Characteristic 1..128 : Time of measuring value : Second ( 2 digits )
<\$C1..128\$R1..30>	Characteristic 1..128 : Reference information dataset ( Machine, Batch, ... )
<\$C1..128\$R1..30\$-B>	Characteristic 1..128 : Reference information ( blank characters are removed )
<\$C1..128\$R1..30\$L15>	Characteristic 1..128 : Reference information ( length=15 bytes / missing bytes=blank )
<p><b>!!! The numbers R1...30 of the reference information datasets you find in menu “Options / Reference information” !!!</b></p> <p><b>!!! The output of Reference Information in a measuring value output block does actually not work !!!</b></p>	

ASCII-codes :

Wildcard	Function
<00> ... <ff>	ASCII-code ( as HEX-number ) of one character



### Example 1 : Send a SOAP message to web service [www.ptsv2.com](http://www.ptsv2.com)

Configure the SFct065 as shown in chapter 2, using your SOAP server's IP address + port :

#### Contents of file HTTP header.txt ( ANSI text file )

```
POST /post.php?dir=example2 HTTP/1.0
User-Agent: NuSOAP/4.22
Host: www.ptsv2.com
Content-Type: text/xml;charset=UTF-8
Content-length: <$HL>
```

#### Contents of file Content header.txt ( ANSI text file )

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
```

#### Contents of file Content.txt ( ANSI text file )

```
<ns6803:getProd xmlns:ns6803="http://tempuri.org">
  <category xsi:type="xsd:string">books</category>
</ns6803:getProd>
```

#### Contents of file Content footer.txt ( ANSI text file )

```
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Now you can check your message at [www.ptsv2.com](http://www.ptsv2.com).

To send a CSV list with all saved measurement values of characteristic C1, you may use the following content template :

```
<ns4159:getProd xmlns:ns4159="http://tempuri.org">
  <category xsi:type="xsd:string">
    Kommentar: <$HC>
    index;value;offset;factor;Year;Month;Day;Hour;Minute;Second;Machine
    <$C1$RB><$C1$RI>;<$C1$MV>;<$C1$CO>;<$C1$CF>;<$C1$DY>;<$C1$DM>;<$C1$DD>;<$C1$
    TH>;<$C1$TM>;<$C1$TS>;<$C1$R5>
    <$C1$RE></category>
  </ns4159:getProd>
```

### Example 2 : Get file /index.html from web server localhost

Use the following file content for HTTP header file, leave all other files empty

```
GET /index.html HTTP/1.1
Host: localhost
```

**Note:** You must end the file with 2 carriage returns. (See [rfc7230](#) for details ).