# Contents

## 1 Description

## 2 Program Organization Units

### 2.1 FTP

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTP (FB)</td>
<td>3</td>
</tr>
<tr>
<td>FbFTP.DirCreate (METH)</td>
<td>3</td>
</tr>
<tr>
<td>FbFTP.DirRemove (METH)</td>
<td>5</td>
</tr>
<tr>
<td>FbFTP.FileAppend (METH)</td>
<td>7</td>
</tr>
<tr>
<td>FbFTP.FileAppendFromMem (METH)</td>
<td>8</td>
</tr>
<tr>
<td>FbFTP.FileRead (METH)</td>
<td>10</td>
</tr>
<tr>
<td>FbFTP.FileReadFromMem (METH)</td>
<td>11</td>
</tr>
<tr>
<td>FbFTP.FileRemove (METH)</td>
<td>13</td>
</tr>
<tr>
<td>FbFTP.FileRename (METH)</td>
<td>14</td>
</tr>
<tr>
<td>FbFTP.FileWrite (METH)</td>
<td>15</td>
</tr>
<tr>
<td>FbFTP.FileWriteFromMem (METH)</td>
<td>17</td>
</tr>
<tr>
<td>FbFTP.List (METH)</td>
<td>18</td>
</tr>
</tbody>
</table>

### 2.1.1 FbFTP.DirCreate (FB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirCreate (METH)</td>
<td>44</td>
</tr>
<tr>
<td>FbFTPES.DirRemove (METH)</td>
<td>47</td>
</tr>
<tr>
<td>FbFTPES.FileAppend (METH)</td>
<td>48</td>
</tr>
<tr>
<td>FbFTPES.FileAppendFromMem (METH)</td>
<td>50</td>
</tr>
<tr>
<td>FbFTPES.FileRead (METH)</td>
<td>51</td>
</tr>
<tr>
<td>FbFTPES.FileReadToMem (METH)</td>
<td>53</td>
</tr>
<tr>
<td>FbFTPES.FileRemove (METH)</td>
<td>55</td>
</tr>
<tr>
<td>FbFTPES.FileRename (METH)</td>
<td>57</td>
</tr>
<tr>
<td>FbFTPES.FileWrite (METH)</td>
<td>58</td>
</tr>
<tr>
<td>FbFTPES.FileWriteFromMem (METH)</td>
<td>60</td>
</tr>
<tr>
<td>FbFTP.List (METH)</td>
<td>61</td>
</tr>
</tbody>
</table>

### 2.1.2 FbFTPES.DirCreate (FB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirCreate (FB)</td>
<td>65</td>
</tr>
</tbody>
</table>

### 2.1.3 FbFTPES.DirRemove (FB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirRemove (FB)</td>
<td>67</td>
</tr>
</tbody>
</table>

### 2.2 FTPES

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirCreate (FB)</td>
<td>44</td>
</tr>
</tbody>
</table>

### 2.2.1 FbFTPES.DirCreate (FB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirCreate (METH)</td>
<td>47</td>
</tr>
<tr>
<td>FbFTPES.DirRemove (METH)</td>
<td>48</td>
</tr>
<tr>
<td>FbFTPES.FileAppend (METH)</td>
<td>50</td>
</tr>
<tr>
<td>FbFTPES.FileAppendFromMem (METH)</td>
<td>51</td>
</tr>
<tr>
<td>FbFTPES.FileRead (METH)</td>
<td>53</td>
</tr>
<tr>
<td>FbFTPES.FileReadToMem (METH)</td>
<td>55</td>
</tr>
<tr>
<td>FbFTPES.FileRemove (METH)</td>
<td>57</td>
</tr>
<tr>
<td>FbFTPES.FileRename (METH)</td>
<td>58</td>
</tr>
<tr>
<td>FbFTPES.FileWrite (METH)</td>
<td>60</td>
</tr>
<tr>
<td>FbFTPES.FileWriteFromMem (METH)</td>
<td>61</td>
</tr>
<tr>
<td>FbFTP.List (METH)</td>
<td>63</td>
</tr>
</tbody>
</table>

### 2.2.2 FbFTPES.DirRemove (FB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FbFTPES.DirRemove (FB)</td>
<td>65</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>3.1</td>
<td>Status (GVL)</td>
</tr>
<tr>
<td>4</td>
<td>Version History (GVL)</td>
</tr>
<tr>
<td>5</td>
<td>Library Reference</td>
</tr>
</tbody>
</table>
CHAPTER 1

Description

This document is automatically generated. Because of this, the chapter 30 Visualization is not shown in this document. If you are interested in getting to know more about visualization, we refer to the library manager of e!Cockpit.

Subject to Changes

WAGO Kontakttechnik GmbH Co. KG reserves the right to provide for any alterations or modifications. WAGO Kontakttechnik GmbH Co. KG owns all rights arising from the granting of patents or from the legal protection of utility patents. Third-party products are always mentioned without any reference to patent rights. Thus, the existence of such rights cannot be excluded.

Personnel Qualification

All tasks that are carried out with libraries made for the e!COCKPIT software must only be performed by qualified electrical specialists instructed in PLC programming according to IEC 61131-3.

All tasks that have an effect on the properties or the behavior of automation hardware or software products must only be performed by qualified employees with a thorough knowledge of handling the products concerned.

Intended Use of e!COCKPIT Libraries

Libraries created for the e!COCKPIT software are used to simplify the development of application projects in the IEC 61131-3 programming languages.

For automation tasks, WAGO offers programmable logic controllers in a wide variety of performance classes. In combination with a wide range of I/O modules, the controllers can process standard types of field signals. Controllers can be implemented centrally or in decentralized configurations. The controllers offer interfaces for the most commonly used fieldbuses for use in decentralized configurations. Fieldbus independent I/O modules are then linked via fieldbus couplers. WAGO controllers offer a runtime environment for user programs called e!RUNTIME. Software projects for implementation in e!RUNTIME environments can be created in e!COCKPIT. The programming environment in e!COCKPIT is based on the established CODESYS 3 industrial standard. Users with a previous knowledge of CODESYS 3 will thus find this environment largely familiar. The following programming languages of the IEC 61131-3 standard are available:

- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Sequential Function Chart (SFC)
- Continuous Function Chart (CFC)

The individual programming languages can also be combined as required during the development of the software. A portfolio of prepared libraries can be accessed for many frequently used functions in order to make software development more efficient. This document provides an overview of the WagoAppFTP that WAGO offers for e!COCKPIT.
This library provide Client-Services for FTP, FTPS, FTPES and SFTP

Further library information are summerized here:

Company  WAGO
Title  WagoAppFTP
Version  1.3.1.2
Categories  WAGO FunctionalView|Connectivity|Network; WAGO LayerView|App; WAGO Internal|Feature|Network|FTP; Application
Author  WAGO / u013972
Placeholder  WagoAppFTP

---

Based on WagoAppFTP.library, last modified 13.08.2019, 21:03:59.
The content of this file was automatically generated with None on 13.08.2019, 21:04:11
20 Program Organization Units

2.1 FTP

2.1.1 FbFTP (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xDone</td>
<td>BOOL</td>
<td></td>
<td>Process successfull performed.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Indicates an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransfered</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

Function

This function block provides FTP-Client-Services.
Graphical Illustration

![Function Block Diagram]

Function Description

The FbFTP is the base function block for FTP-Client-Services. The Fb provide functions like read and write files, rename or remove files and create or remove directories.

The inputs of the FbFTP handle the general information for the FTP connection and the outputs signal the process and the result of the operations. The operations are triggered by call the specific method for the operation. After a call the methods will return immediately and the FbFTP start the operation.

Note: This function block must called cyclic.

Example:

```plaintext
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;
  iState: INT;

  //List
  sListBuffer : STRING(255);
  //Write
  sTxBuffer : STRING(255) := 'Hello World!';
  //Read
  sRxBuffer : STRING(255);
END_VAR

FbFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode: FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
```

2.1. FTP
WagoAppFTP, Release 1.3.1.2

IF FbFTP.xDone = TRUE THEN
  iState := iState + 1;
ELSIF FbFTP.xError = TRUE THEN
  iState := 99;
END_IF

IF FbFTP.xBusy = FALSE THEN
  CASE iState OF
    0:
      //Makes a directory listing of the user's home directory.
      FbFTP.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer),
                 udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);
    1:
      //Creates the directory 'NewDir' and the file 'HelloWorld.txt'. Writes
      //the data from sTxBuffer into the new file.
      FbFTP.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt',
                            pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
    2:
      //Create a new directory. A Path with '/' at the begin starts in the
      //root directory
      FbFTP.DirCreate(sRemotePath := '//root/NewDir/NewDir2');
    3:
      //Makes a directory.
      FbFTP.List(sRemotePath := 'NewDir', pRxBuffer := ADR(sListBuffer),
                 udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '*.txt', xWithDetails := TRUE);
    4:
      //Rename and replace the file.
      FbFTP.FileRename(sOldFileName := '//root/NewDir/HelloWorld.txt',
                       sNewFileName := 'NewDir/NewDir2/Rename.txt');
    5:
      //Reads the data from the file.
      FbFTP.FileReadToMem(sRemotePath := '//root/NewDir/NewDir2/Rename.txt',
                          pRxBuffer := ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));
    6:
      //Removes the file.
      FbFTP.FileRemove(sRemotePath := 'NewDir/NewDir2/Rename.txt');
    7:
      //Removes a directory
      FbFTP.DirRemove(sRemotePath := '//root/NewDir/NewDir2');
    8:
      //Removes a directory
      FbFTP.DirRemove(sRemotePath := 'NewDir/');
    99:
      //Some Error Handling
  END_CASE
END_IF

FbFTP.DirCreate (METH)

Interface variables

2.1. FTP
**Function**

Create a directory

**Graphical Illustration**

![Graphical Interface of FbFTP.DirCreate](image)

**Fig. 2.2: Graphical Interface of FbFTP.DirCreate**

**Function description**

This method starts an operation to create a new directory specified by `sRemotePath`. The method create only the last directory in the path. If a directory in the path is missing or the directory, that should create, already exists an error occur.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

**Example:**

```fortran
VAR
    FbFTP : WagoAppFTP.FbFTP;
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransferred: UDINT;
    xDo : BOOL;
END_VAR

FbFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
)```

2.1. FTP
IF xDo = TRUE THEN
  //Create a new directory. A Path with '//' at the begin starts in the root_
  ←directory
  FbFTP.DirCreate(sRemotePath := '//'root/NewDir');
  xDo := FALSE;
END_IF

FbFTP.DirRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirRemove</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the directory that should be removed.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'</td>
</tr>
</tbody>
</table>

Remove a directory

Graphical Illustration

![Fig. 2.3: Graphical Interface of FbFTP.DirRemove](image)

Function description

This method starts an operation to remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exist, the directory cannot removed (in use, or not empty), or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

Example:

```plaintext
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
```
WagoAppFTP, Release 1.3.1.2

```
udiNBytesToTransfer: UDINT;
udiNBytesTransferred: UDINT;

xD0: BOOL;
END_VAR

FbFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Removes a directory. A Path with '/' at the begin starts in the root directory
  FbFTP.DirRemove(sRemotePath := '//root/NewDir');
  xDo := FALSE;
END_IF
```

**FbFTP.FileAppend (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppend</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should append to.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

**Function**

Append a local file to a remote file

**Graphical Illustration**

![Graphical Interface of FbFTP.FileAppend](image)

Fig. 2.4: Graphical Interface of FbFTP.FileAppend
Function description

This method starts an operation to append a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exist, it will be created.

For “`sLocalPath`” you can use the following prefixes, known form WagoAppFileDir:

- `HOME://`
- `CARD://`
- `TEMP://`
- `ROOT://`

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE</td>
</tr>
<tr>
<td>EINV AL</td>
<td>The input is invalid</td>
</tr>
<tr>
<td>ENOENT</td>
<td>The local file doesn’t exist</td>
</tr>
</tbody>
</table>

Example:

```plaintext
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;
  xDo : BOOL;
END_VAR

FbFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Append the local file to the remote file
  FbFTP.FileAppend(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/>HelloWorld.txt', xCreateDirs := TRUE);
  xDo := FALSE;
END_IF
```
FbFTP.FileAppendFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppendFromMem</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the file, the data form the transfer buffer should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Append a data from a buffer to a remote file

Graphical Illustration

![Graphical Interface of FbFTP.FileAppendFromMem](image)

Function description

This method starts an operation to append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

Example:

```var
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiINBytesToTransfer: UDINT;
  udiINBytesTransfered: UDINT;
```

2.1. FTP 10
xDo : BOOL;

//Append
sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTP(sServer:= '192.168.1.17',
uiPort:= 0,
sUser:= 'root',
sPassword:= 'wago',
xActiveMode:= FALSE,
sActivePort:= '',
tKeepAlive:= T#5S,
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Append the data from sTxBuffer into the remote file.
  FbFTP.FileAppendFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbFTP.FileRead (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRead</td>
<td>Wago.Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Function
Read a file

Graphical Illustration

Fig. 2.6: Graphical Interface of FbFTP.FileRead

Function description
This method starts an operation to read data from the remote file, that is specified by the input parameter `sRemotePath`, and stores it in the local file, that is specified by the input parameter `sLocalPath`.

For “sLocalPath” you can use the following prefixes, known from `WagoAppFileDir`:

- `HOME://`
- `CARD://`
- `TEMP://`
- `ROOT://`

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Result Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

Example:

```plaintext
VAR
    FbFTP : WagoAppFTP.FbFTP;
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
    xDo : BOOL;

    //Read
    sRxBuffer : STRING(255);
END_VAR

FbFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Reads the data from the file.
    FbFTP.FileRead(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/>HelloWorld.txt');
    xDo := FALSE;
END_IF
```
FbFTP.FileReadToMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileReadToMem</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '/'.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDIINT</td>
<td>Size of the receive buffer</td>
</tr>
</tbody>
</table>

Function

Read a file

Graphical Illustration

Fig. 2.7: Graphical Interface of FbFTP.FileReadToMem

Function description

This method starts an operation to reads the data form the remote file, that is specified by the input parameter sRemotePath, and stored it in the RxBuffer, that is specified by the input parameter pRxBuffer and udiRxBufferSize.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started. WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE WagoTypes.eResultCode.EINVAL The input is invalid

Example:

```
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDIINT;
  udiNBytesTransferred: UDIINT;
  xDo : BOOL;

  //Read
```

2.1. FTP
WagoAppFTP, Release 1.3.1.2

DECLARE
sRxBuffer : STRING(255);
END_VAR

FbFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Reads the data from the file.
  FbFTP.FileReadToMem(sRemotePath := 'NewDir/HelloWorld.txt', pRxBuffer :=
    ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));
  xDo := FALSE;
END_IF

FbFTP.FileRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRemove</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should remove. For a path</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>form the home directory, the path should start without slash. For a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Function

Remove a file

Graphical Illustration

![Graphical Interface of FbFTP.FileRemove](image)

Function description

This method starts an operation to remove the file, that is specified by sRemotePath. An error is returned if the file does not exists, the file cannot be removed, or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTP.

2.1. FTP
This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.
WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
WagoTypes.eResultCode.EINVAL The input is invalid.

Example:

```plaintext
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Removes a file. A Path with './' at the begin starts in the root directory
  FbFTP.FileRemove(sRemotePath := '//root/NewDir/HelloWorld.txt');
  xDo := FALSE;
END_IF
```

**FbFTP.FileRename (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRename</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server, that define the file that should renamed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Input</td>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td>Path on the remote server, that define the new file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
</tbody>
</table>

**Function**

Rename a file
Graphical Illustration

![Graphical Interface of FbFTP.FileRename]

**Function description**

This method starts an operation to rename the file, that is defined by the input parameter `sOldFileName`, to the name, that is defined by the input parameter `sNewFileName`.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Result Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
VAR
    FbFTP : WagoAppFTP.FbFTP;
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
    xDo : BOOL;
END_VAR

FbFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    // Rename and replace the file. A path with '/' at the begin starts in the root directory, a path without slash at the begin starts in the users home directory
    FbFTP.FileRename(sOldFileName := '//root/NewDir/HelloWorld.txt', sNewFileName := 'NewDir/Rename.txt');
```

2.1. FTP
FbFTP.FileWrite (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite</td>
<td>WagoTypes.eResultCode</td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should write to. For a path form the home directory, the path should start without slash. For an absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration

![Graphical Interface of FbFTP.FileWrite](image)

Function description

This method starts an operation to write the data from the local file, that is specified by the input parameter sLocalPath, to the remote file, that is specified by the input parameter sRemotePath.

If the remote file exit, it will be overwritten.

For “sLocalPath” you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

- WagoTypes.eResultCode.OK Success, operation has been started.
- WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVAL The input is invalid.
- WagoTypes.eResultCode.ENOENT The local file doesn’t exist.

Example:
VAR
    FbFTP : WagoAppFTP.FbFTP;
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
    xDo: BOOL;
END_VAR

FbFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    // Write the local file to the remote server
    FbFTP.FileWrite(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt', xCreateDirs := TRUE);
    xDo := FALSE;
END_IF

FbFTP.FileWriteFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite-FromMem</td>
<td>Wago-Types.eResult</td>
<td>Code</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the data from the transfer buffer should write to. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration
### Function description

This method starts an operation to write the data from the TxBuffer to a file on the remote host specified by `sRemotePath`. If the remote file exit, it will be overwritten.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while `xBusy` is `TRUE`.
- **WagoTypes.eResultCode.EINVAL** The input is invalid.

### Example:

```pascal
VAR
  FbFTP : WagoAppFTP.FbFTP;
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;

  //Append
  sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTP(  
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Write the data from sTxBuffer into the remote server.
  FbFTP.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
```
FbFTP.List (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>List</td>
<td>Wago-Types.eResultCode</td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBuffer-Size</td>
<td>UDINT</td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>sFileSpec</td>
<td>STRING(255)</td>
<td>Specify a specific file or type, e.g. ‘*.txt’</td>
</tr>
<tr>
<td></td>
<td>xWithDetails</td>
<td>BOOL</td>
<td>List more than only the names of the files or directories</td>
</tr>
</tbody>
</table>

Function
List files and directories

Graphical Illustration

![Graphical Interface of FbFTP.List](image)

Function description
This method starts an operation to performs a directory listing. The listing includ only those files that match the specification stored in “sFileSpec”. The input parameter pRxBuffer is pointing to the first element of the buffer and the buffersize is defined by the input parameter udiRxBufferSize.

If more details of the files and directories should list, the input parameter xWithDetails must set to true.

The status and result of the operation are displayed by the outputs of the FbFTP.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.
WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
WagoTypes.eResultCode.EINV AL The input is invalid.

Example:
VAR
   FbFTP : WagoAppFTP.FbFTP;
   xBusy: BOOL;
   xDone: BOOL;
   xError: BOOL;
   oStatus: WagoSysErrorBase.FbResult;
   udiNBytesToTransfer: UDINT;
   udiNBytesTransfered: UDINT;
   xDo: BOOL;

   //List
   sListBuffer: STRING(255);

END_VAR

FbFTP(
   sServer:= '192.168.1.17',
   uiPort:= 0,
   sUser:= 'root',
   sPassword:= 'wago',
   xActiveMode:= FALSE,
   sActivePort:= '',
   tKeepAlive:= T#5S,
   tTimeout:= T#5S,
   xBusy=> xBusy,
   xDone=> xDone,
   xError=> xError,
   oStatus=> oStatus,
   udiNBytesToTransfer=> udiNBytesToTransfer,
   udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
   //Makes a directory listing of the user’s home directory.
   FbFTP.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer), udiRxBufferSize := → SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);
   xDo := FALSE;

END_IF

2.1.2 FbFTP_DirCreate (FB)

Interface variables
**Function**

Create a directory

**Graphical Illustration**

![Graphical Interface of FbFTP_DirCreate](image)

**Function description**

This function block creates a new directory specified by `sRemotePath`. The Fb creates only the last directory in the path. If a directory in the path is missing or the directory that should create already exists, an error occurs.
Transition to TRUE on xTrigger triggers the process to create a new directory. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

**Example**

```plaintext
VAR
    FbFTP_DirCreate : WagoAppFTP.FbFTP_DirCreate;
    Trigger : BOOL := TRUE;
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
END_VAR
FbFTP_DirCreate(
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sRemotePath:= 'NewDir',
    xBusy=> xBusy,
    xError=> xError,
    oStatus=> oStatus);
```

### 2.1.3 FbFTP_DirRemove (FB)

**Interface variables**
<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server with directorys, that should be removed. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSy-sError-Base.FbResult</td>
<td>Execution state or error code</td>
<td></td>
</tr>
</tbody>
</table>

**Function**

Remove a directory

**Graphical Illustration**

![Graphical Interface of FbFTP.DirRemove](image)

**Fig. 2.14: Graphical Interface of FbFTP.DirRemove**

**Function description**

This functionblock remove remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exists, the directory cannot be removed (in use, or not empty), or if the user does not have the
appropriate privilege level.

Transition to TRUE on xTrigger triggers the process to remove a directory. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

```
VAR
  FbFTP_DirRemove : WagoAppFTP.FbFTP_DirRemove;
  Trigger : BOOL := TRUE;
  xBusy : BOOL;
  xError : BOOL;
  oStatus : WagoSysErrorBase.FbResult;
END_VAR

FbFTP_DirRemove(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= 'NewDir',
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus);
```

2.1.4 FbFTP_FileAppend (FB)

Interface variables
Scope | Name | Type | Initial | Comment
--- | --- | --- | --- | ---
Inout | xTrigger | BOOL | | Will be reset by the functionblok if it is terminated.
Input | sServer | STRING(255) | | Name or IP of the Server, e.g. '192.168.1.17'
 | uiPort | UINT | 0 | Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.
 | sUser | STRING(255) | | Authentication
 | sPassword | STRING(255) | | Authentication
 | xActiveMode | BOOL | FALSE | If xActiveMode is true then the FTP-Client try to connect in active mode.
 | sActivePort | STRING(255) | '0' | If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.
 | tKeepAlive | TIME | | Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.
 | tTimeout | TIME | | The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.
 | sLocalPath | STRING(255) | '0' | Path to the local file that should append to the remote file.
 | sRemotePath | STRING(255) | | Path on the remote server to the file, the local file should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.
 | xCreateDirs | BOOL | | Create all missing directories in the sRemotePath
Output | xBusy | BOOL | | True while not terminated.
 | xError | BOOL | | Signals presence of an error.
 | oStatus | WagoSysErrorBase.FbResult | | Execution state or error code.
 | udiNBytesTransferred | UDINT | | Number of Bytes, that are currently transferred.

**Function**

Append a local file to a remote file

**Graphical Illustration**

![Graphical Interface of FbFTP_FileAppend](image)

Fig. 2.15: Graphical Interface of FbFTP_FileAppend
Function description

This function block appends a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn't exist, it will be created.

Transition to `TRUE` on `xTrigger` triggers the process to append the local file to the remote file. The function block resets `xTrigger` to `FALSE` again after it has finished the process. If something went wrong during the process `xTrigger` resets to `FALSE` and `xError` is set to `TRUE`. The output `xBusy` indicates that the function block is still processing.

For “`sLocalPath`” you can use the following prefixes, known from WagoAppFileDir:

- `HOME://`
- `CARD://`
- `TEMP://`
- `ROOT://`

Example

```var
VAR
  FbFTP_FileAppend : WagoAppFTP.FbFTP_FileAppend;
  Trigger : BOOL := TRUE;
  xBusy : BOOL;
  xError : BOOL;
  oStatus : WagoSysErrorBase.FbResult;
  udiNBytesTransfered : UDIM;
END_VAR

FbFTP_FileAppend(
  xTrigger := Trigger,
  sServer := '192.168.1.17',
  uiPort := 0,
  sUser := 'root',
  sPassword := 'wago',
  xActiveMode := FALSE,
  sActivePort := '',
  tKeepAlive := T#5S,
  tTimeout := T#5S,
  sLocalPath := 'HOME://testbench/myfirstfile.txt',
  sRemotePath := 'Append.txt',
  xCreateDirs := FALSE,
  xBusy => xBusy,
  xError => xError,
  oStatus => oStatus,
  udiNBytesTransfered => udiNBytesTransfered);
```

2.1.5 FbFTP_FileAppendFromMem (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Input</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the data from the transfer buffer should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '/'.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Append a data from a buffer to a remote file

### Graphical Illustration

2.1. FTP
**Function description**

This function block append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exit, it will be created.

Transition to TRUE on xTrigger triggers the process to append the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

**Example**

```plaintext
VAR
    FbFTP_FileAppendFromMem : WagoAppFTP.FbFTP_FileAppendFromMem;
    Trigger : BOOL := TRUE;
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesTransfere: UDINT;
    sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTP_FileAppendFromMem(
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sRemotePath:= 'AppendFromMem.txt',
    pTxBuffer:= ADR(sTxBuffer),
    udiTxNBytes := LENGTH(sTxBuffer),
    xCreateDirs := FALSE,
    xBusy=> xBusy,

2.1. FTP
2.1.6 FbFTP_FileRead (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keep alive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For an absolute path the path should start with ‘/’</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

Function

Read a file

Graphical Illustration
Function description

This function block reads the data from the remote file, that is specified by the input parameter \texttt{sRemotePath}, and stores it in the local file, that is specified by the input parameter \texttt{sLocalPath}.

Transition to \texttt{TRUE} on \texttt{xTrigger} triggers the process to read the remote file to a local file. The function block resets \texttt{xTrigger} to \texttt{FALSE} again after it has finished the process. If something went wrong during the process \texttt{xTrigger} resets to \texttt{FALSE} and \texttt{xError} is set to \texttt{TRUE}. The output \texttt{xBusy} indicates that the function block is still processing.

For \texttt{"sLocalPath"} you can use the following prefixes, known from \texttt{WagoAppFileDir}:

- \texttt{HOME://}
- \texttt{CARD://}
- \texttt{TEMP://}
- \texttt{ROOT://}

Example

```plaintext
VAR
  FbFTP_FileRead : WagoAppFTP.FbFTP_FileRead;
  Trigger : BOOL := TRUE;
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;
END_VAR

FbFTP_FileRead(  
  xTrigger:= Trigger,  
  sServer:= '192.168.1.17',  
  uiPort:= 0,  
  sUser:= 'root',  
  sPassword:= 'wago',  
  xActiveMode:= FALSE,  
  sActivePort:= '',  
  tKeepAlive:= T#5S,  
  tTimeout:= T#5S,
```

Fig. 2.17: Graphical Interface of FbFTP_FileRead
2.1.7 FbFTP_FileReadToMem (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ’192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>’0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-‘ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td></td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSystemErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

Function

Read a file

Graphical Illustration
Function description

This functionblock reads the data form the remote file, that is specified by the input parameter \texttt{sRemotePath}, and stored it in the \texttt{RxBuffer}, that is specified by the input parameter \texttt{pRxBuffer} and \texttt{udiRxBufferSize}. If the remote file not exist or the parameter \texttt{pRxBuffer} and \texttt{udiRxBufferSize} are 0 then an error occur.

Transition to TRUE on \texttt{xTrigger} triggers the process to remove a remote file. The function block resets \texttt{xTrigger} to FALSE again after it has finished the process. If something went wrong during the process \texttt{xTrigger} resets to FALSE and \texttt{xError} is set to TRUE. The output \texttt{xBusy} indicates that the function block is still processing.

Example

```plaintext
VAR
  FbFTP_FileReadToMem : WagoAppFTP.FbFTP_FileReadToMem;
  Trigger : BOOL := TRUE;
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  sRxBuffer : STRING(255);
END_VAR

FbFTP_FileReadToMem(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive: T#5S,
  tTimeout:= T#5S,
  sRemotePath:= 'Some.txt',
  pRxBuffer:= ADR(sRxBuffer),
  udiRxBufferSize:= SIZEOF(sRxBuffer),
  xBusy-> xBusy,
)
```

Fig. 2.18: Graphical Interface of FbFTP_FileReadToMem

2.1. FTP
2.1.8 FbFTP_FileRemove (FB)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ’192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>’0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ’-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should remove. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ’/’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

**Function**

Remove a file

**Graphical Illustration**
Function description

This function block removes the file, that is specified by sRemotePath. An error is returned if the file does not exist, the file cannot be removed, or if the user does not have the appropriate privilege level.

Transition to TRUE on xTrigger triggers the process to remove a remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

```plaintext
VAR
    FbFTP_FileRemove : WagoAppFTP.FbFTP_FileRemove;
    Trigger : BOOL := TRUE;
    xBusy : BOOL;
    xError : BOOL;
    oStatus : WagoSysErrorBase.FbResult;
END_VAR

FbFTP_FileRemove{
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sRemotePath:= 'AppendFromMem.txt',
    xBusy=> xBusy,
    xError=> xError,
    oStatus=> oStatus);
```

2.1.9 FbFTP_FileRename (FB)

Interface variables
### Function

Rename a file

### Graphical Illustration

![Graphical Interface of FbFTP_FileRename](image)

**Fig. 2.20: Graphical Interface of FbFTP_FileRename**

---

2.1. FTP

---

36
Function description

This function block rename the file, that is defined by the input parameter sOldFileName, to the name, that is defined by the input parameter sNewFileName.

Transition to TRUE on xTrigger triggers the process to rename a remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example:

```
VAR
  FbFTP_FileRename : WagoAppFTP.FbFTP_FileRename;
  Trigger : BOOL := TRUE;
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTP_FileRename{
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sOldFileName:= 'Test.txt',
  sNewFileName:= 'Rename.txt',
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus);
```

2.1.10 FbFTP_FileWrite (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
</tbody>
</table>

#### Input

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.1'</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>Path on the remote server to the file, the local file should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

#### Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Write a file

### Graphical Illustration

![Function Block Diagram](image)

**Fig. 2.21: Graphical Interface of FbFTP_FileWrite**
**Function description**

This function block writes the data from the local file, that is specified by the input parameter \texttt{sLocalPath}, to the remote file, that is specified by the input parameter \texttt{sRemotePath}.

Transition to TRUE on \texttt{xTrigger} triggers the process to write the local file to the remote file. The function block resets \texttt{xTrigger} to FALSE again after it has finished the process. If something went wrong during the process \texttt{xTrigger} resets to FALSE and \texttt{xError} is set to TRUE. The output \texttt{xBusy} indicates that the function block is still processing.

For "\texttt{sLocalPath}" you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

**Example:**

```plaintext
VAR
  FbFTP_FileWrite : WagoAppFTP.FbFTP_FileWrite;
  Trigger : BOOL := TRUE;
  xBusy: BOOL;
  xError: BOOL;
  uStatus : WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;
END_VAR

FbFTP_FileWrite(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sLocalPath:= 'HOME://testbench/myfirstfile.txt',
  sRemotePath:= 'Write.txt',
  xCreateDirs:= FALSE,
  xBusy=> xBusy,
  xError=> xError,
  uStatus=> uStatus,
  udiNBytesTransfered=> udiNBytesTransfered);
```

### 2.1.11 FbFTP_FileWriteFromMem (FB)

**Interface variables**
### Scope

<table>
<thead>
<tr>
<th>Inout</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xTrigger</td>
<td>BOOL</td>
<td>True</td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the data form the transfer buffer should write to. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td>True</td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSyssErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Write a file

### Graphical Illustration

---

2.1. FTP
Function description

This function block writes the data from the TxBuffer to a file on the remote host specified by sRemotePath.

Transition to TRUE on xTrigger triggers the process to write the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process, xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example:

```
VAR
    FbFTP_FileWriteFromMem : WagoAppFTP.FbFTP_FileWriteFromMem;
    Trigger : BOOL := TRUE;
    xBusy : BOOL;
    xError : BOOL;
    oStatus : WagoSysErrorBase.FbResult;
    udiNBytesTransferred : UDINT;
    sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTP_FileWriteFromMem(
    xTrigger := Trigger,
    sServer := '192.168.1.17',
    uiPort := 0,
    sUser := 'root',
    sPassword := 'wago',
    xActiveMode := FALSE,
    sActivePort := '',
    tKeepAlive := T#5S,
    tTimeout := T#5S,
    sRemotePath := 'Write.txt',
    pTxBuffer := ADR(sTxBuffer),
    udiTxNBytes := LENGTH(sTxBuffer),
    xCreateDirs := FALSE,
    xBusy := xBusy,
    xError := xError,
)```

Fig. 2.22: Graphical Interface of FbFTP_FileWriteFromMem
2.1.12 FbFTP_List (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td>Input</td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td></td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>sFileSpec</td>
<td>STRING(255)</td>
<td></td>
<td>Specify a specific file or type. e.g. '*.txt'</td>
</tr>
<tr>
<td></td>
<td>xWithDetails</td>
<td>BOOL</td>
<td></td>
<td>List more than only the names of the files or directories</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

Function
List files and directories

Graphical Illustration
Function description

This function block performs a directory listing. The listing includes only those files that match the specification stored in `sFileSpec`. The input parameter `pRxBuffer` is pointing to the first element of the buffer and the buffer size is defined by the input parameter `udiRxBufferSize`.

If more details of the files and directories should be listed, the input parameter `xWithDetails` must be set to true.

Transition to `TRUE` on `xTrigger` triggers the process to make a directory listing. The function block resets `xTrigger` to `FALSE` again after it has finished the process. If something went wrong during the process, `xTrigger` resets to `FALSE` and `xError` is set to `TRUE`. The output `xBusy` indicates that the function block is still processing.

Example:

```plaintext
VAR
  FbFTP_List : WagoAppFTP.FbFTP_List;
  Trigger : BOOL := TRUE;
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransferred: UDINT;
  sRxBuffer : STRING(255);
END_VAR

FbFTP_List(
  xTrigger:=Trigger,
  sServer:='192.168.1.17',
  uipPort:=0,
  sUser:='root',
  sPassword:='wago',
  xActiveMode:=FALSE,
  sActivePort:='',
  tKeepAlive:=T#5S,
  tTimeout:=T#5S,
  sRemotePath:='')
```

Fig. 2.23: Graphical Interface of FbFTP_List
2.2 FTPES

2.2.1 FbFTPES (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>0</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s-Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xDone</td>
<td>BOOL</td>
<td></td>
<td>Process successfull performed.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Indicates an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sErrorBase.FbResu</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>udiNBytesTo-</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>udiNBytes-</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transferred</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function

This function block provides FTPES-Client-Services.

Graphical Illustration
### Function Description

The FbFTPES is the base function block for FTPES-Client-Services. The Fb provide functions like read and write files, rename or remove files and create or remove directories.

The inputs of the FbFTPES handle the general information for the FTPES connection and the outputs signal the process and the result of the operations. The operations are triggered by call the specific method for the operation. After a call the methods will return immediately and the FbFTPES start the operation.

**Note:** This function block must called cyclic.

### Example:

Example without host and peer verification.

```plaintext
VAR
    FbFTPES : WagoAppFTP.FbFTPES;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
    iState : INT;
    //List
    sListBuffer : STRING(255);
    //Write
    sTxBuffer : STRING(255) := 'Hello World!';
    //Read
    sRxBuffer : STRING(255);
END_VAR
```
WagoAppFTP, Release 1.3.1.2

FBFTPES
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy:= xBusy,
  xDone:= xDone,
  xError:= xError,
  oStatus:= oStatus,
  udiNBytesToTransfer:= udiNBytesToTransfer,
  udiNBytesTransferred:= udiNBytesTransferred);

IF FBFTPES.xDone = TRUE THEN
  iState := iState + 1;
ELSIF FBFTPES.xError = TRUE THEN
  iState := 99;
END_IF

IF FBFTPES.xBusy = FALSE THEN
  CASE iState OF
    0:
      // Makes a directory listing of the user’s home directory.
      FBFTPES.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer),
        udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);
      1:
      // Creates the directory 'NewDir' and the file 'HelloWorld.txt'. Writes
      // the data from sTxBuffer into the new file.
      FBFTPES.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt',
        pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
      2:
      // Create a new directory. A Path with '///' at the begin starts in the
      // root directory.
      FBFTPES.DirCreate(sRemotePath := '//root/NewDir/NewDir2');
      3:
      // Makes a directory.
      FBFTPES.List(sRemotePath := 'NewDir', pRxBuffer := ADR(sListBuffer),
        udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '*.txt', xWithDetails := TRUE);
      4:
      // Rename and replace the file.
      FBFTPES.FileRename(sOldFileName := '//root/NewDir/NewDir2/Rename.txt',
        sNewFileName := 'NewDir/NewDir2/Rename.txt');
      5:
      // Reads the data from the file.
      FBFTPES.FileReadToMem(sRemotePath := '//root/NewDir/NewDir2/Rename.txt',
        pRxBuffer := ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));
      6:
      // Removes the file.
      FBFTPES.FileRemove(sRemotePath := 'NewDir/NewDir2/Rename.txt');
      7:
      // Removes a directory
      FBFTPES.DirRemove(sRemotePath := '//root/NewDir/NewDir2');
      8:
      // Removes a directory
      FBFTPES.DirRemove(sRemotePath := 'NewDir/');
FbFTPES.DirCreate (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirCreate</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server with the directory, that should create. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Function*
Create a directory

Graphical Illustration

![Graphical Interface of FbFTPES.DirCreate](image)

Fig. 2.25: Graphical Interface of FbFTPES.DirCreate

Function description
This method starts an operation to create a new directory specified by sRemotePath. The method create only the last directory in the path. If a directory in the path is missing or the directory, that should create, already exists an error occur.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

Example:
Example without host and peer verification.

```vbnet
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= ''),
```
WagoApp FTP, Release 1.3.1.2

sClientCert_KeyPasswd := '');
xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDo := BOOL;

END_VAR

FbFTPES(
sServer:= '192.168.1.17',
uiPort:= 0,
sUser:= 'root',
sPassword:= 'wago',
xActiveMode:= FALSE,
xActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Create a new directory. A Path with '//' at the begin starts in the root directory
    FbFTPES.DirCreate(sRemotePath := '//root/NewDir');
    xDo := FALSE;
END_IF

FbFTPES.DirRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirRemove</td>
<td>Wago-Types.eResultCode</td>
<td>Path on the remote server to the directory that should be removed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '/'.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the directory that should be removed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '/'.</td>
</tr>
</tbody>
</table>

Remove a directory

Graphical Illustration

---

Fig. 2.26: Graphical Interface of FbFTPES.DirRemove
Function description

This method starts an operation to remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exists, the directory cannot removed (in use, or not empty), or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

Example:

Example without host and peer verification.

```plaintext
VAR
    FbFTPES : WagoAppFTP.FbFTPES;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');

    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;

    xDo : BOOL;
END_VAR

FBFTPES(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Removes a directory. A Path with '// ' at the begin starts in the root directory
    FbFTPES.DirRemove(sRemotePath := '//root/NewDir');
```

2.2. FTPES
xDo := FALSE;
END_IF

FbFTPES.FileAppend (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppend</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should append to.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Append a local file to a remote file

Graphical Illustration

![Graphical Interface of FbFTPES.FileAppend](image)

Function description

This method starts an operation to append a local file specified by sLocalPath to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

For “sLocalPath” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

- WagoTypes.eResultCode.OK Success, operation has been started.
- WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVAL The input is invalid.
- WagoTypes.eResultCode.ENOENT The local file doesn’t exist.

Example:

Example without host and peer verification.
VAR

FbFTPES : WagoAppFTP.FbFTPES;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xFdo : BOOL;
END_VAR

FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  // Append the local file to the remote file
  FbFTPES.FileAppend(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt', xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbFTPES.FileAppendFromMem (METH)

Interface variables
### Function

Append a data from a buffer to a remote file

#### Graphical Illustration

![Graphical Interface of FbFTPES.FileAppendFromMem](image)

Fig. 2.28: Graphical Interface of FbFTPES.FileAppendFromMem

### Function description

This method starts an operation to append data from the TxBuffer to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exist, it will be created.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

- `WagoTypes.eResultCode.OK` Success, operation has been started.
- `WagoTypes.eResultCode.EBUSY` The Fb is already in process. No new operation can start while `xBusy` is `TRUE`.
- `WagoTypes.eResultCode.EINV AL` The input is invalid.

### Example:

Example without host and peer verification.

```plaintext
VAR
    FbFTPES : WagoAppFTP.FbFTPES;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= ''
    );
```

2.2. FTPES
WagoAppFTP, Release 1.3.1.2

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDo : BOOL;

//Append
sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTPES(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Append the data from sTxBuffer into the remote file.
    FbFTPES.FileAppendFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
    xDo := FALSE;
END_IF

FbFTPES.FileRead (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRead</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
</tbody>
</table>
|       | sRemotePath | STRING(255)  | Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For an absolute path the path should start with ‘/’.

Function

Read a file

Graphical Illustration

2.2. FTPES
Function description

This method starts an operation to read the data from the remote file, that is specified by the input parameter sRemotePath, and stored it in the local file, that is specified by the input parameter sLocalPath.

For “sLocalPath” you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

- WagoTypes.eResultCode.OK Success, operation has been started.
- WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVAL The input is invalid.

Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  "sRemotePath": 'C:\Data\File.txt',
  "sLocalPath": 'C:\Local\File.txt'
)
```

2.2. FTPES
WagoAppFTP, Release 1.3.1.2

xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
    //Reads the data from the file.
    FbFTPES.FileRead(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt');
    xDo := FALSE;
END_IF

FbFTPES.FileReadToMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileReadToMem</td>
<td>Wago-Types.eResultCode</td>
<td>Code</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '/'.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td>Size of the receive buffer</td>
</tr>
</tbody>
</table>

Function

Read a file

Graphical Illustration

Function description

This method starts an operation to reads the data form the remote file, that is specified by the input parameter sRemotePath, and stored it in the RxBuffer, that is specified by the input parameter pRxBuffer and udiRxBufferSize.

The status and result of the operation are displayed by the outputs of the FbFTPES.

2.2. FTPES 55
This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while **xBusy** is TRUE.
- **WagoTypes.eResultCode.EINV** The input is invalid.

### Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;

  xDo : BOOL;

//Read
  sRxBuffer : STRING(255);
END_VAR

FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Reads the data from the file.
  FbFTPES.FileReadToMem(sRemotePath := 'NewDir/HelloWorld.txt', pRxBuffer := ADR(sRxBuffer), udiRxBufferSize := sizeof(sRxBuffer));
  xDo := FALSE;
END_IF
```

2.2. FTPES
WagoAppFTP, Release 1.3.1.2

FbFTPES.FileRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRemove</td>
<td>Wago-Types.eResultCode</td>
<td>Return</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should remove. For a path</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>form the home directory, the path should start without slash. For a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Function

Remove a file

Graphical Illustration

![Fig. 2.31: Graphical Interface of FbFTPES.FileRemove](image)

Function description

This method starts an operation to remove the file, that is specified by sRemotePath. An error is returned if the file does not exists, the file cannot be removed, or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.
WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
WagoTypes.eResultCode.EINVAL The input is invalid.

Example:

Example without host and peer verification.

```var
FbFTPES : WagoAppFTP.FbFTPES;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;
```

2.2. FTPES 57
FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Removes a file. A Path with '//' at the begin starts in the root directory
  FbFTPES.FileRemove(sRemotePath := '//root/NewDir/HelloWorld.txt');
  xDo := FALSE;
END_IF

FbFTPES.FileRename (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRename</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td>Path on the remote server, that define the file that should renamed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>sNewFileName</td>
<td>STRING(255)</td>
<td>Path on the remote server, that define the new file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
</tbody>
</table>

Function

Rename a file

Graphical Illustration

Fig. 2.32: Graphical Interface of FbFTPES.FileRename
Function description

This method starts an operation to rename the file, that is defined by the input parameter \textit{sOldFileName}, to the name, that is defined by the input parameter \textit{sNewFileName}.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

\begin{itemize}
  \item \texttt{WagoTypes.eResultCode.OK} Success, operation has been started.
  \item \texttt{WagoTypes.eResultCode.EBUSY} The Fb is already in process. No new operation can start while \texttt{xBusy} is \texttt{TRUE}.
  \item \texttt{WagoTypes.eResultCode.EINVAL} The input is invalid.
\end{itemize}

Example:

Example without host and peer verification.

\begin{verbatim}
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTPES{
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Rename and replace the file. A path with '/' at the begin starts in the root directory, a path without slash at the begin starts in the user's home directory
  FbFTPES.FileRename(sOldFileName := '//root/NewDir/HelloWorld.txt',
                   sNewFileName := 'NewDir/Rename.txt');
\end{verbatim}
FbFTPES.FileWrite (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration

![Graphical Interface of FbFTPES.FileWrite](image)

Function description

This method starts an operation to write the data form the local file, that is specified by the input parameter sLocalPath, to the remote file, that is specified by the input parameter sRemotePath.

If the remote file exit, it will be overwritten.

For “sLocalPath” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.
WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
WagoTypes.eResultCode.EINVAL The input is invalid.
WagoTypes.eResultCode.ENOENT The local file doesn’t exist.

Example:

Example without host and peer verification.

2.2. FTPES 60
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Write the local file to the remote server
  FbFTPES.FileWrite(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt', xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbFTPES.FileWriteFromMem (METH)

Interface variables

2.2. FTPES
### Function

Write a file

### Graphical Illustration

![Graphical Interface of FbFTPES.FileWriteFromMem](image)

**Method**

**FbFTPES.FileWriteFromMem**

- **sRemotePath**: STRING
- **pTxBuffer**: POINTER TO BYTE
- **udiTxNBytes**: UDINT
- **xCCreateDirs**: BOOL

**Function description**

This method starts an operation to write the data from the TxBuffer to a file on the remote host specified by **sRemotePath**. If the remote file exit, it will be overwritten.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

**Example:**

Example without host and peer verification.

```vbnet
VAR
FbFTPES : WagoAppFTP.FbFTPES;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
```

### Example:

Example without host and peer verification.
xFbFTPES{
  sServer:= '192.168.1.17',
  uPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiBytesToTransfer:= udiBytesTransfered,
  udiBytesTransfered:= udiBytesTransfered);

IF xDo = TRUE THEN
  //Write the data from sTxBuffer into the remote server.
  FbFTPES.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbFTPES.List (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>List</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>sFileSpec</td>
<td>STRING(255)</td>
<td>Specify a specific file or type. e.g. ‘*.txt’</td>
</tr>
<tr>
<td></td>
<td>xWithDetails</td>
<td>BOOL</td>
<td>List more than only the names of the files or directories</td>
</tr>
</tbody>
</table>

Function

List files and directories

Graphical Illustration
Function description

This method starts an operation to perform a directory listing. The listing includes only those files that match the specification stored in `sFileSpec`. The input parameter `pRxBuffer` is pointing to the first element of the buffer, and the buffer size is defined by the input parameter `udiRxBufferSize`.

If more details of the files and directories should be listed, the input parameter `xWithDetails` must be set to true.

The status and result of the operation are displayed by the outputs of the FbFTPES.

This method has the following return values:

- `WagoTypes.eResultCode.OK`: Success, operation has been started.
- `WagoTypes.eResultCode.EBUSY`: The Fb is already in process. No new operation can start while `xBusy` is TRUE.
- `WagoTypes.eResultCode.EINV`: The input is invalid.

Example:

Example without host and peer verification.

```pascal
VAR
  FbFTPES : WagoAppFTP.FbFTPES;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;

  xDo : BOOL;
END_VAR

FbFTPES(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '')
```
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Makes a directory listing of the user’s home directory.
  FbFTPES.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer), udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);
  xDo := FALSE;
END_IF

2.2.2 FbFTPES_DirCreate (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server with the directory, that should create. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-sError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>
**Function**

Create a directory

**Graphical Illustration**

![Function Block Diagram](image)

**Function description**

This function block creates a new directory specified by `sRemotePath`. The Fb creates only the last directory in the path. If a directory in the path is missing or the directory that should create already exists, an error occurs.

Transition to TRUE on `xTrigger` triggers the process to create a new directory. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process, `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbFTPES_DirCreate : WagoAppFTP.FbFTPES_DirCreate;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPES_DirCreate(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'admin',
  sPassword:= 'password',
  xActiveMode:= TRUE,
  sActivePort:= '21',
  typSSL_Options:= typSSL_Options,
  tKeepAlive: TIME = 3000,
  tTimeout: TIME = 30000,
  sRemotePath:= '/new/directory');
```

**2.2. FTPES**
sUser:= 'root',
sPassword:= 'wago',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sRemotePath:= 'NewDir',
xBusy=> xBusy,
xError=> xError,
oStatus=> oStatus);

### 2.2.3 FbFTPES_DirRemove (FB)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType- Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server with directories, that should be removed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’ .</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys- sError- Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

*Function*

Remove a directory

**Graphical Illustration**
Fig. 2.37: Graphical Interface of FbFTPES_DirRemove

Function description

This function block remove remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exist, the directory cannot be removed (in use, or not empty), or if the user does not have the appropriate privilege level.

Transition to TRUE on xTrigger triggers the process to remove a directory. The function block resets xTrigger to FALSE after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
    FbFTPES_DirRemove : WagoAppFTP.FbFTPES_DirRemove;
    Trigger : BOOL := TRUE;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPES_DirRemove(
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
```
WagoAppFTP, Release 1.3.1.2

2.2.4 FbFTPES_FileAppend (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the local file should append to. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sErrorBase.FbResult</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

Function

Append a local file to a remote file

Graphical Illustration
**Function description**

This function block append a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exist, it will be created.

Transition to TRUE on `xTrigger` triggers the process to append the local file to the remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For “sLocalPath” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbFTPES_FileAppend : WagoAppFTP.FbFTPES_FileAppend;

  Trigger : BOOL := TRUE;

  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransferred: UDINT;
```

**Fig. 2.38: Graphical Interface of FbFTPES_FileAppend**
### 2.2.5 FbFTPES_FileAppendFromMem (FB)

**Interface variables**
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
</tbody>
</table>

#### Input

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the data form the transfer buffer should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
</tbody>
</table>

#### Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Append a data from a buffer to a remote file

### Graphical Illustration

2.2. FTPES 72
**Function description**

This function block append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

Transition to TRUE on xTrigger triggers the process to append the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbFTPES_FileAppendFromMem : WagoAppFTP.FbFTPES_FileAppendFromMem;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;

  sTxBuffer : STRING(255) := 'Hello World';
END_VAR

FbFTPES_FileAppendFromMem(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  sRemotePath := 'my/path/to/file.txt',
  typSSL_Options := typSSL_Options,
  xActiveMode := TRUE)
```

Fig. 2.39: Graphical Interface of FbFTPES_FileAppendFromMem

---

WagoAppFTP, Release 1.3.1.2
uiPort:= 0,
sUser:= 'root',
sPassword:= 'wago',
xAactiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sRemotePath:= 'AppendFromMem.txt',
pTxBuffer:= ADR(sTxBuffer),
udINBytes := LENGTH(sTxBuffer),
xCreateDirs := FALSE,
xBusy-> xBusy,
xError-> xError,
oStatus-> oStatus,
udiNBytesTransferred-> udiNBytesTransferred);

2.2.6 FbFTPES_FileRead (FB)

Interface variables
### Input

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client tries to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType- s- Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

### Output

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys- sError- Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTo- Transfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytes- Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

**Function**

Read a file

**Graphical Illustration**

2.2. FTPES
Function description

This function block reads the data form the remote file, that is specified by the input parameter \texttt{sRemotePath}, and stored it in the local file, that is specified by the input parameter \texttt{sLocalPath}.

Transition to \texttt{TRUE} on \texttt{xTrigger} triggers the process to read the remote file to a local file. The function block resets \texttt{xTrigger} to \texttt{FALSE} again after it has finished the process. If something went wrong during the process \texttt{xTrigger} resets to \texttt{FALSE} and \texttt{xError} is set to \texttt{TRUE}. The output \texttt{xBusy} indicates that the function block is still processing.

For \texttt{"sLocalPath"} you can use the following prefixes, known from \texttt{WagoAppFileDir}:

- \texttt{HOME://}
- \texttt{CARD://}
- \texttt{TEMP://}
- \texttt{ROOT://}

Example

Example without host and peer verification.

```
VAR
    FbFTPES_FileRead : WagoAppFTP.FbFTPES_FileRead;
    Trigger : BOOL := TRUE;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
```

Fig. 2.40: Graphical Interface of FbFTPES_FileRead
2.2.7 FbFTPES_FileReadToMem (FB)

Interface variables
## Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType- Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path shoulc start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td></td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSys- sError- Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytesTo- Transfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td>udiNBytes- Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

## Function

Read a file

## Graphical Illustration
Function description

This function block reads the data from the remote file, that is specified by the input parameter `sRemotePath`, and stored it in the `RxBuffer`, that is specified by the input parameter `pRxBuffer` and `udiRxBufferSize`. If the remote file not exist or the parameter `pRxBuffer` and `udiRxBufferSize` are 0 then an error occur.

Transition to TRUE on `xTrigger` triggers the process to remove a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
FbFTPES_FileReadToMem : WagoAppFTP.FbFTPES_FileReadToMem;

Trigger : BOOL := TRUE;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

sRxBuffer : STRING(255);
END_VAR

FbFTPES_FileReadToMem(
  xTrigger:= Trigger,
  ...
2.2.8 FbFTPES_FileRemove (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>Input</td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should remove. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-sError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>
Function
Remove a file

Graphical Illustration

![Graphical Interface of FbFTPES_FileRemove](image)

**FunctionBlock**

**FbFTPES_FileRemove**

```
FunctionBlock

FbFTPES_FileRemove

xTrigger   BOOL   xBusy
sServer    STRING BOOL xError
uiPort     UINT    FbResult oStatus
sUser      STRING
sPassword  STRING
xActiveMode BOOL
sActivePort STRING
typSSL_Options typSSL_Options
iKeepAlive  TIME
iTimeout   TIME
sRemotePath STRING
```

Fig. 2.42: Graphical Interface of FbFTPES_FileRemove

**Function description**

This function block removes the file, that is specified by `sRemotePath`. An error is returned if the file does not exist, the file cannot be removed, or if the user does not have the appropriate privilege level.

Transition to TRUE on `xTrigger` triggers the process to remove a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example**

Example without host and peer verification.

```plaintext
VAR
FbFTPES_FileRemove : WagoAppFTP.FbFTPES_FileRemove;

Trigger : BOOL := TRUE;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '','',
  sCA_Path:= '','',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '','',
  sClientCert_Key:= '','',
  sClientCert_KeyPasswd:= ' ');

xBusy: BOOL;
xErro: BOOL;
oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPES_FileRemove(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
```
2.2.9 FbFTPES_FileRename (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client tries to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server, that define the file that should be renamed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td></td>
<td>sNewFileName</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server, that define the new file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysError BASE.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

Function
Rename a file

Graphical Illustration

![Graphical Interface of FbFTPES_FileRename](image)

Function description

This function block rename the file, that is defined by the input parameter `sOldFileName`, to the name, that is defined by the input parameter `sNewFileName`.

Transition to TRUE on `xTrigger` triggers the process to rename a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

Example:

Example without host and peer verification.

```
VAR
  FbFTPES_FileRename : WagoAppFTP.FbFTPES_FileRename;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '', sCA_Path:= '', xVerifyPeer:= FALSE, xVerifyHost:= FALSE, sClientCert:= '', sClientCert_Key:= '', sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

```
sPassword := 'wago',
xActiveMode := FALSE,
sActivePort := '1',
typSSL_Options := typSSL_Options,
tKeepAlive := T#5S,
tTimeout := T#5S,
sOldFileName := 'Append.txt',
sNewFileName := 'Rename.txt',
xBusy => xBusy, 
xCError => xError, 
oStatus => oStatus);

2.2.10 FbFTPES_FileWrite (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>Input</td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client tries to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘:’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the local file should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>
Function
Write a file

Graphical Illustration

Fig. 2.44: Graphical Interface of FbFTPES_FileWrite

Function description
This function block write the data form the local file, that is specified by the input parameter sLocalPath, to the remote file, that is specified by the input parameter sRemotePath.

Transition to TRUE on xTrigger triggers the process to write the local file to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

For “sLocalPath” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

Example:
Example without host and peer verification.

```plaintext
VAR
FbFTPES_FileWrite : WagoAppFTP.FbFTPES_FileWrite;
Trigger : BOOL := TRUE;
typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
sCA_Path:= '',
xVerifyPeer:= FALSE,
xVerifyHost:= FALSE,
sClientCert:= FALSE,
sClientCert_Key:= '',
sClientCert_KeyPasswd:= '');
```

2.2. FTPES
xBusy: BOOL;
xErro: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesTransfered: UDINT;
END_VAR

FbFTPES_FileWrite(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sLocalPath:= 'HOME://testbench/myfirstfile.txt',
  sRemotePath:= 'Write.txt',
  xCreateDirs:= FALSE,
  xBusy=> xBusy,
 xErro=> xErro,
oStatus=> oStatus,
udiNBytesTransfered=> udiNBytesTransfered);

2.2.11 FbFTPES_FileWriteFromMem (FB)

Interface variables
## Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>’0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType-&lt;br&gt;Curl.typSSL_Options</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0s the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the data form the transfer buffer should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath.</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Write a file

### Graphical Illustration

2.2. FTPES 87
**Function description**

This function block writes the data from the TxBuffer to a file on the remote host specified by sRemotePath.

Transition to TRUE on xTrigger triggers the process to write the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

**Example:**

Example without host and peer verification.

```plaintext
VAR
    FbFTPES_FileWriteFromMem : WagoAppFTP.FbFTPES_FileWriteFromMem;
    Trigger : BOOL := TRUE;

    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');

    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransferred: UDINT;

    sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTPES_FileWriteFromMem(
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sActivePort: STRING,
    sRemotePath: STRING,
    iuProcessID: UINT,
    typSSL_Options: typSSL_Options,
    iuTimeout: TIME,
    sActiveMode: BOOL,
    iuTLSFlags: UINT,
    sCreateDirs: STRING,
    sPassword: STRING,
    sUsername: STRING,
    xBusy: BOOL,
    xError: BOOL,
    oStatus: FbResult,
    udiNBytesToTransfer: UDINT,
    udiNBytesTransferred: UDINT)
```
2.2.12 FbFTPES_List (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
</tbody>
</table>

### Input

- **sServer** | STRING(255) | Name or IP of the Server, e.g. ‘192.168.1.17’
- **uiPort**  | UINT         | 0       | Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.
- **sUser**   | STRING(255) |         | Authentication                                                          |
- **sPassword** | STRING(255) |         | Authentication                                                          |
- **xActive-Mode** | BOOL | FALSE  | If xActiveMode is true then the FTP-Client try to connect in active mode. |
- **sActivePort** | STRING(255) | ’0’     | If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ’-‘ to specify a port range. If the port specified is 0, the operating system will pick a free port. |
- **typSSL_Options** | WagoTypeCurl.typSSL_Options | Authentication |
- **tKeepAlive** | TIME | Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled |
- **tTimeout** | TIME | TIME#10s0ms | The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated |
- **sRemotePath** | STRING(255) | | Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’. |
- **pRxBuffer** | POINTER TO BYTE | Address of the receive buffer |
- **udiRxBufferSize** | UDINT | | Size of the receive buffer |
- **sFileSpec** | STRING(255) | | Specify a specific file or type. e.g. ‘*.txt’ |
- **xWithDetails** | BOOL | | List more than only the names of the files or directories |

### Output

- **xBusy** | BOOL | True while not terminated. |
- **xError** | BOOL | Signals presence of an error. |
- **oStatus** | WagoSysErrorBase.FbResult | Execution state or error code |
- **udiNBytesTransferred** | UDINT | Number of Bytes, that are currently transferred |

### Function

List files and directories

### Graphical Illustration

2.2. FTPES 90
Function description

This function block performs a directory listing. The listing includes only those files that match the specification stored in \textit{sFileSpec}. The input parameter \textit{pRxBuffer} is pointing to the first element of the buffer and the buffersize is defined by the input parameter \textit{udiRxBufferSize}.

If more details of the files and directories should be listed, the input parameter \textit{xWithDetails} must be set to true.

Transition to \textit{TRUE} on \textit{xTrigger} triggers the process to make a directory listing. The function block resets \textit{xTrigger} to \textit{FALSE} again after it has finished the process. If something went wrong during the process, \textit{xTrigger} resets to \textit{FALSE} and \textit{xError} is set to \textit{TRUE}. The output \textit{xBusy} indicates that the function block is still processing.

Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPES_List : WagoAppFTP.FbFTPES_List;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;

  sRxBuffer : STRING(255);
END_VAR
```

2.2. FTPES
FbFTPES_List(
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sRemotePath:= '',
    pRxBuffer:= ADR(sRxBuffer),
    udiRxBufferSize:= SIZEOF(sRxBuffer),
    sFileSpec:= '',
    xWithDetails:= FALSE,
    xBusy=> xBusy,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesTransfered=> udiNBytesTransfered);

2.3 FTPS

2.3.1 FbFTPS (FB)

Interface variables
### WagoAppFTP, Release 1.3.1.2

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>Input</td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-Json::Curl.typSSL_Options</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xDone</td>
<td>BOOL</td>
<td></td>
<td>Process successfully performed.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Indicates an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSy-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiN-BytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

This function block provides FTPS-Client-Services.

### Graphical Illustration

![Graphical Interface of FbFTPS](image)

Fig. 2.47: Graphical Interface of FbFTPS

---

2.3. FTPS
**Function Description**

The FbFTPS is the base function block for FTPS-Client-Services. The Fb provide functions like read and write files, rename or remove files and create or remove directories.

The inputs of the FbFTPS handle the general information for the FTPS connection and the outputs signal the process and the result of the operations. The operations are triggered by call the specific method for the operation. After a call the methods will return immediately and the FbFTPS start the operation.

**Note:** This function block must called cyclic.

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (  
    sCA_Cert:= '',  
    sCA_Path:= '',  
    xVerifyPeer:= FALSE,  
    xVerifyHost:= FALSE,  
    sClientCert := '',  
    sClientCert_Key := '',  
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;  
  xDone: BOOL;  
  xError: BOOL;  
  oStatus: WagoSysErrorBase.FbResult;  
  udiNBytesToTransfer: UDINT;  
  udiNBytesTransferred: UDINT;  
  iState : INT;  
  //List  
  sListBuffer : STRING(255);  
  //Write  
  sTxBuffer : STRING(255) := 'Hello World!';  
  //Read  
  sRxBuffer : STRING(255);
END_VAR

FbFTPS(  
  sServer:= '172.29.233.1',  
  uiPort:= 0,  
  sUser:= 'user',  
  sPassword:= 'password',  
  xActiveMode:= FALSE,  
  sActivePort:= '',  
  typSSL_Options := typSSL_Options,  
  tKeepAlive:= T#5S,  
  tTimeout:= T#5S,  
  xBusy=> xBusy,  
  xDone=> xDone,  
  xError=> xError,  
  oStatus=> oStatus,  
  udiNBytesToTransfer=> udiNBytesToTransfer,  
  udiNBytesTransferred=> udiNBytesTransferred);

IF FbFTPS.xDone = TRUE THEN  
  iState := iState + 1;
```

2.3. FTPS
ELSIF FbFTPS.xError = TRUE THEN  
   iState := 99;  
END_IF

IF FbFTPS.xBusy = FALSE THEN  
CASE iState OF  
  0: // Makes a directory listing of the user's home directory.  
     FbFTPS.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer),  
                 udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);  
  1: // Creates the directory 'NewDir' and the file 'HelloWorld.txt'. Writes  
     the data from sTxBuffer into the new file.  
     FbFTPS.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt', sTxBuffer := pTxBuffer,  
                           udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);  
  2: // Create a new directory. A Path with '//' at the begin starts in the  
     root directory  
     FbFTPS.DirCreate(sRemotePath := 'NewDir/NewDir2');  
  3: // Makes a directory.  
     FbFTPS.List(sRemotePath := 'NewDir', pRxBuffer := ADR(sListBuffer),  
                 udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '*.txt', xWithDetails :=  
                 TRUE);  
  4: // Rename and replace the file.  
     FbFTPS.FileRename(sOldFileName := 'NewDir/HelloWorld.txt',  
                       sNewFileName := 'NewDir/NewDir2/Rename.txt');  
  5: // Reads the data from the file.  
     FbFTPS.FileReadToMem(sRemotePath := 'NewDir/NewDir2/Rename.txt',  
                          pRxBuffer := ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));  
  6: // Removes the file.  
     FbFTPS.FileRemove(sRemotePath := 'NewDir/NewDir2/Rename.txt');  
  7: // Removes a directory  
     FbFTPS.DirRemove(sRemotePath := 'NewDir/NewDir2');  
  8: // Removes a directory  
     FbFTPS.DirRemove(sRemotePath := 'NewDir/');  
  99: // Some Error Handling  
END_CASE
END_IF

FbFTPS.DirCreate (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DirCreate</td>
<td>Wago-Types.eResultCode</td>
<td>Code</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server with the directory, that should create. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
</tbody>
</table>

Function*
Create a directory

Graphical Illustration

Fig. 2.48: Graphical Interface of FbFTPS.DirCreate

Function description

This method starts an operation to create a new directory specified by sRemotePath. The method create only the last directory in the path. If a directory in the path is missing or the directory, that should create, already exists an error occur.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

- WagoTypes.eResultCode.OK Success, operation has been started.
- WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINV The input is invalid

Example:

Example without host and peer verification.

```pascal
VAR
    FbFTPS : WagoAppFTP.FbFTPS;
    typSSL_Options : WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd:= '');
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransferred: UDINT;
    xDo : BOOL;
END_VAR

FbFTPS(
    sServer:= '172.29.233.1',
    uiPort:= 0,
    sUser:= 'user',
    sPassword:= 'password',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
)```

2.3. FTPS
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
Error=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Create a new directory. A Path with '//' at the begin starts in the root directory
    FbFTPS.DirCreate(sRemotePath := '//root/NewDir');
    xDo := FALSE;
END_IF

FbFTPS.DirRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirRemove</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the directory that should be removed. For a path from the home directory, the path should start without slash. For an absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the directory that should be removed. For a path from the home directory, the path should start without slash. For an absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Remove a directory

Graphical Illustration

Fig. 2.49: Graphical Interface of FbFTPS.DirRemove

Function description

This method starts an operation to remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exist, the directory cannot removed (in use, or not empty), or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK Success, operation has been started. WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE WagoTypes.eResultCode.EINVAL The input is invalid</th>
</tr>
</thead>
</table>

Example:

Example without host and peer verification.
**VAR**

FbFTPS : WagoAppFTP.FbFTPS;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDone : BOOL;
END_VAR

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Removes a directory. A Path with '// ' at the begin starts in the root-
directory
  FbFTPS.DirRemove(sRemotePath := '//root/NewDir');
  xDo := FALSE;
END_IF

**FbFTPS.FileAppend (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppend</td>
<td>WagoTypes.eResultCode</td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should append to. For a path form the home directory, the path shiould start without slash. For a absolute path the path should start with '/'.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Input</td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
Function

Append a local file to a remote file

Graphical Illustration

![Graphical Interface of FbFTPS.FileAppend](image)

Fig. 2.50: Graphical Interface of FbFTPS.FileAppend

Function description

This method starts an operation to append a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exit, it will be created.

For `'sLocalPath` you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while `xBusy` is TRUE.
- **WagoTypes.eResultCode.EINVAL** The input is invalid.
- **WagoTypes.eResultCode.ENOENT** The local file doesn’t exist.

Example:

Example without host and peer verification.

```pascal
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (  
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;  
  xDone: BOOL;  
  xError: BOOL;  
  oStatus: WagoSysErrorBase.FbResult;  
  udiNBytesToTransfer: UDINT;  
  udiNBytesTransferred: UDINT;  
  xDo : BOOL;
```

2.3. FTPS
END_VAR

FbFTPS(
    sServer:= '172.29.233.1',
    uiPort:= 0,
    sUser:= 'user',
    sPassword:= 'password',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Append the local file to the remote file
    FbFTPS.FileAppend(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt', xCreateDirs := TRUE);
    xDo := FALSE;
END_IF

FbFTPS.FileAppendFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppendFromMem</td>
<td>WagoTypes.eResultCode</td>
<td>Path on the remote server to the file, the data form the transfer buffer should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Append a data from a buffer to a remote file

Graphical Illustration
Fig. 2.51: Graphical Interface of FbFTPS.FileAppendFromMem

Function description

This method starts an operation to append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.  
WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE.  
WagoTypes.eResultCode.EINV VAL The input is invalid

Example:

Example without host and peer verification.

```pascal
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;

  xDo : BOOL;

  //Append
  sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
)
WagoAppFTP, Release 1.3.1.2

```plaintext
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Append the data from sTxBuffer into the remote file.
  FbFTPS.FileAppendFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer :=
ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbFTPS.FileRead (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRead</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
</tbody>
</table>

Function
Read a file

Graphical Illustration

Fig. 2.52: Graphical Interface of FbFTPS.FileRead

Function description
This method starts an operation to reads the data form the remote file, that is specified by the input parameter sRemotePath, and stored it in the local file, that is specified by the input parameter sLocalPath.

For “sLocalPath” you can use the following prefixes, known form Wago.AppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

2.3. FTPS
The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while $xBusy$ is TRUE.
- **WagoTypes.eResultCode.EINV** The input is invalid.

Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Reads the data from the file.
  FbFTPS.FileRead(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/>HelloWorld.txt');
  xDo := FALSE;
END_IF
```

2.3. FTPS
FbFTPS.FileReadToMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileReadToMem</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, that should read. For a path</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>form the home directory, the path should start without slash. For a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td>Size of the receive buffer</td>
</tr>
</tbody>
</table>

Function

Read a file

Graphical Illustration

![Graphical Interface of FbFTPS.FileReadToMem](image)

Fig. 2.53: Graphical Interface of FbFTPS.FileReadToMem

Function description

This method starts an operation to reads the data from the remote file, that is specified by the input parameter sRemotePath, and stored it in the RxBuffer, that is specified by the input parameter pRxBuffer and udiRxBufferSize.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

- WagoTypes.eResultCode.OK: Success, operation has been started.
- WagoTypes.eResultCode.EBUSY: The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVVAL: The input is invalid.

Example:

Example without host and peer verification.

```pascal
VAR
    FbFTPS : WagoAppFTP.FbFTPS;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
    );
```

2.3. FTPS
sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xDone: BOOL;
xErro r: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransferred: UDINT;

xDo := BOOL;

//Read
sRxBuffer := STRING(255);

END_VAR

FbFTPS(
    sServer:= '172.29.233.1',
    uiPort:= 0,
    sUser:= 'user',
    sPassword:= 'password',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
    //Reads the data from the file.
    FbFTPS.FileReadToMem(sRemotePath := 'NewDir/HelloWorld.txt', pRxBuffer := →ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));

    xDo := FALSE;
END_IF

**FbFTPS.FileRemove (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRemove</td>
<td>Wago-Types.eResultCode</td>
<td>Path on the remote server to the file, that should remove. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'</td>
</tr>
</tbody>
</table>

**Function**

Remove a file

**Graphical Illustration**
WagoAppFTP, Release 1.3.1.2

Fig. 2.54: Graphical Interface of FbFTPS.FileRemove

Function description

This method starts an operation to remove the file, that is specified by sRemotePath. An error is returned if the file does not exist, the file cannot be removed, or if the user does not have the appropriate privilege level.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

Example:

Example without host and peer verification.

```
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (  
    sCA_Cert:= '',  
    sCA_Path:= '',  
    xVerifyPeer:= FALSE,  
    xVerifyHost:= FALSE,  
    sClientCert:= '',  
    sClientCert_Key:= '',  
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;

  xDo : BOOL;
END_VAR

FbFTPS(  
  sServer:= '172.29.233.1',  
  uiPort:= 0,  
  sUser:= 'user',  
  sPassword:= 'password',  
  xActiveMode:= FALSE,  
  sActivePort:= '',  
  typSSL_Options := typSSL_Options,  
  tKeepAlive:= T#5S,  
  tTimeout:= T#5S,  
  xBusy=> xBusy,  
  xDone=> xDone,  
  xError=> xError,  
  oStatus=> oStatus,  
```
IF xDo = TRUE THEN
//Removes a file. A Path with '//' at the begin starts in the root directory
FbFTPS.FileRemove(sRemotePath := '//root/NewDir/HelloWorld.txt');
xDo := FALSE;
END_IF

FbFTPS.FileRename (METH)

Function variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRename</td>
<td>Wago-Types.eResultCode</td>
<td>Path on the remote server, that define the file that should be renamed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'</td>
</tr>
<tr>
<td>Input</td>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td>Path on the remote server, that define the file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'</td>
</tr>
<tr>
<td>Input</td>
<td>sNewFileName</td>
<td>STRING(255)</td>
<td>Path on the remote server, that define the file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'</td>
</tr>
</tbody>
</table>

Function

Rename a file

Graphical Illustration

Fig. 2.55: Graphical Interface of FbFTPS.FileRename

Function description

This method starts an operation to rename the file, that is defined by the input parameter sOldFileName, to the name, that is defined by the input parameter sNewFileName.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started. WagoTypes.eResultCode.EBUSY The Fb is already in process. No new operation can start while xBusy is TRUE WagoTypes.eResultCode.EINVAL The input is invalid

Example:

Example without host and peer verification.
VAR

FbFTPS : WagoAppFTP.FbFTPS;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert := '',
  sCA_Path := '',
  xVerifyPeer := FALSE,
  xVerifyHost := FALSE,
  sClientCert := '',
  sClientCert_Key := '',
  sClientCert_KeyPasswd := '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDo : BOOL;
END_VAR

FDFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  // Rename and replace the file. A path with '/' at the begin starts in the 
  → root directory, a path without slash at the begin starts in the users home 
  → directory
  FbFTPS.FileRename(sOldFileName := '//root/NewDir/HelloWorld.txt', sNewFileName := 'NewDir/Rename.txt');

  xDo := FALSE;
END_IF

FbFTPS.FileWrite (METH)

Interface variables
### Function

Write a file

### Graphical Illustration

**Method**

```
FbFTPS.FileWrite

sLocalPath STRING eResultCode
sRemotePath STRING
xCreatedirs BOOL
```

*Fig. 2.56: Graphical Interface of FbFTPS.FileWrite*

### Function description

This method starts an operation to write the data from the local file, that is specified by the input parameter `sLocalPath`, to the remote file, that is specified by the input parameter `sRemotePath`.

If the remote file exits, it will be overwritten.

For "sLocalPath" you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is <code>TRUE</code>.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.ENOENT</td>
<td>The local file doesn’t exist.</td>
</tr>
</tbody>
</table>

### Example:

Example without host and peer verification.

```
VAR
FbFTPS : WagoAppFTP.FbFTPS;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '', sCA_Path:= '','', xVerifyPeer:= FALSE, xVerifyHost:= FALSE, sClientCert:= '')
```

### Table

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite</td>
<td>Wago-Types.eResultCode</td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the local file should write to. For a path from the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>
sClientCert_Key:= '',
sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDone: BOOL;

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesTransfered,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Write the local file to the remote server
  FbFTPS.FileWrite(sLocalPath := 'HOME://Some.txt', sRemotePath := 'NewDir/HelloWorld.txt', xCreateDirs := TRUE);

  xDo := FALSE;
END_IF

FbFTPS.FileWriteFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWriteFromMem</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Path on the remote server to the file, the data form the transfer buffer should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration
Method

**FbFTPS.FileWriteFromMem**

- **sRemotePath** STRING
- **pTxBuffer** Pointer
- **udiTNBytes** UDINT
- **xCreatdirs** BOOL
- **eResultCode**

Fig. 2.57: Graphical Interface of FbFTPS.FileWriteFromMem

**Function description**

This method starts an operation to write the data from the TxBuffer to a file on the remote host specified by **sRemotePath**. If the remote file exit, it will be overwritten.

The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

<table>
<thead>
<tr>
<th>StatusCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

**Example:**

Example without host and peer verification.

```pascal
VAR
FbFTPS : WagoAppFTP.FbFTPS;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
  sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPassword:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xDo : BOOL;
END_VAR

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
```
xFbFTPS.FileWriteFromMem(sRemotePath := 'NewDir/HelloWorld.txt', pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);

xFbFTPS.List (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>List</td>
<td>Wago-Types.eresultCode</td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td>Specify a specific file or type. e.g. '*.txt'</td>
</tr>
<tr>
<td></td>
<td>sFileSpec</td>
<td>STRING(255)</td>
<td>List more than only the names of the files or directories</td>
</tr>
<tr>
<td></td>
<td>xWithDetails</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Function

List files and directories

Graphical Illustration

Fig. 2.58: Graphical Interface of FbFTPS.List

Function description

This method starts an operation to performs a directory listing. The listing includ only those files that match the specification stored in "sFileSpec". The input parameter pRxBuffer is pointing to the first element of the buffer and the buffer size is defined by the input parameter udiRxBufferSize.

If more details of the files and direcotries should list, the input parameter xWithDetails must set to true.
The status and result of the operation are displayed by the outputs of the FbFTPS.

This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while `xBusy` is TRUE.
- **WagoTypes.eResultCode.EINVAL** The input is invalid.

Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPS : WagoAppFTP.FbFTPS;
  typSSL_Options : WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= ''
  );
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo : BOOL;
END_VAR

FbFTPS(
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Makes a directory listing of the user's home directory.
  FbFTPS.List(sRemotePath := '', pRxBuffer := ADR(sListBuffer), udiRxBufferSize := SIZEOF(sListBuffer), sFileSpec := '', xWithDetails := FALSE);
  xDo := FALSE;
END_IF
```
## 2.3.2 FbFTPS_DirCreate (FB)

### Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>Input</td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType- Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server with the directory, that should create. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysError- Base.FbResult</td>
<td>Execution state or error code</td>
<td></td>
</tr>
</tbody>
</table>

*Function*

Create a directory

*Graphical Illustration*
Function description

This function block creates a new directory specified by `sRemotePath`. The Fb create only the last directory in the path. If a directory in the path is missing or the directory that should create, already exists an error occurs.

Transition to TRUE on `xTrigger` triggers the process to create a new directory. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process, `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
    FbFTPS_DirCreate : WagoAppFTP.FbFTPS_DirCreate;
    typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
        sCA_Cert:= '',
        sCA_Path:= '',
        xVerifyPeer:= FALSE,
        xVerifyHost:= FALSE,
        sClientCert:= '',
        sClientCert_Key:= '',
        sClientCert_KeyPasswd: = '');
    Trigger : BOOL := TRUE;
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPS_DirCreate(
    xTrigger:= Trigger,
    sServer:= '172.29.233.1',
    uiPort:= 0,
    sUser:= 'user',
    sPassword:= 'password',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout: TIME,
    sRemotePath := 'newDirectoryPath';
```

2.3. FTPS
Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentification</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentification</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoTypes-Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server with directories, that should be removed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSysError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

Function*
Remove a directory

Graphical Illustration

2.3. FTPS
### FunctionBlock

FbFTPS_DirRemove

<table>
<thead>
<tr>
<th>xTrigger</th>
<th>BOOL</th>
<th>xBusy</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>sServer</td>
<td>STRING</td>
<td>tKeepAlive</td>
<td>STRING</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>FbResult</td>
<td>oStatus</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.60: Graphical Interface of FbFTPS_DirRemove

### Function description

This function block remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exist, the directory cannot be removed (in use, or not empty), or if the user does not have the appropriate privilege level.

Transition to TRUE on xTrigger triggers the process to remove a directory. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

### Example

Example without host and peer verification.

```plaintext
VAR
FbFTPS_DirRemove : WagoAppFTP.FbFTPS_DirRemove;
Trigger : BOOL := TRUE;
typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
sCA_Path:= '',
xVerifyPeer:= FALSE,
xVerifyHost:= FALSE,
sClientCert:= '',
sClientCert_Key:= '',
sClientCert_KeyPasswd:= '');
xBusy: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPS_DirRemove{
xTrigger:= Trigger,
sServer:= '172.29.233.1',
uiPort:= 0,
sUser:= 'user',
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
}
```

---

2.3. FTPS
### 2.3.4 FbFTPS_FileAppend (FB)

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-</td>
<td></td>
<td>WagoType-SSL_Options</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the local file should append to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
<tr>
<td>Input</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>Output</td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error-Base,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Function**

Append a local file to a remote file

**Graphical Illustration**
**Function description**

This function block append a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exist, it will be created.

Transition to TRUE on `xTrigger` triggers the process to append the local file to the remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For `"sLocalPath"` you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbFTPS_FileAppend : WagoAppFTP.FbFTPS_FileAppend;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;
```

**Fig. 2.61: Graphical Interface of FbFTPS_FileAppend**
END_VAR

FbFTPS_FileAppend(
  xTrigger:= Trigger,
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sPassword:= 'password',
  xActiveMode:= FALSE,
  sActivePort:= '',
  typSSL_Options := typSSL_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sLocalPath:= 'HOME://testbench/myfirstfile.txt',
  sRemotePath:= 'Append.txt',
  xCreateDirs:= FALSE,
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesTransfered=> udiNBytesTransfered);

2.3.5 FbFTPS_FileAppendFromMem (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. <code>192.168.1.17</code></td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a <code>-</code> to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the data form the transfer buffer should append to. For a path form the home directory, the path should start without slash. For an absolute path the path should start with <code>//</code>.</td>
</tr>
<tr>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath.</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Append a data from a buffer to a remote file

### Graphical Illustration

2.3. FTPS
Function description

This function block append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

Transition to TRUE on xTrigger triggers the process to append the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
   FbFTPS_FileAppendFromMem : WagoAppFTP.FbFTPS_FileAppendFromMem;
   Trigger : BOOL := TRUE;
   typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
      sCA_Cert:= '',
      sCA_Path:= '',
      xVerifyPeer:= FALSE,
      xVerifyHost:= FALSE,
      sClientCert:= '',
      sClientCert_Key:= '',
      sClientCert_KeyPasswd:= '');

   xBusy: BOOL;
   xError: BOOL;
   oStatus: WagoSysErrorBase.FbResult;
   udiNBytesTransferred: UDINT;

   sTxBuffer : STRING(255) := 'Hello World';
END_VAR

FbFTPS_FileAppendFromMem(
   xTrigger:= Trigger,
   sServer:= '172.29.233.1',
```

---

2.3. FTPS
2.3.6 FbFTPS_FileRead (FB)

Interface variables

```c
uiPort:= 0,
sUser:= 'user',
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sRemotePath:= 'AppendFromMem.txt',
pTxBuffer:= ADR(sTxBuffer),
udiTxNBytes := LENGTH(sTxBuffer),
xCreateDirs := FALSE,  
xBusy=> xBusy,  
xError=> xError,
oStatus-> oStatus,  
udiBytesTransferred=> udiBytesTransferred);
```
## Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
</tbody>
</table>

## Function

Read a file

## Graphical Illustration

2.3. FTPS
**Function description**

This function block reads the data from the remote file, that is specified by the input parameter `sRemotePath`, and stores it in the local file, that is specified by the input parameter `sLocalPath`.

Transition to TRUE on `xTrigger` triggers the process to read the remote file to a local file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For `"sLocalPath"` you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbFTPS_FileRead : WagoAppFTP.FbFTPS_FileRead;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '', sCA_Path:= '', xVerifyPeer:= FALSE, xVerifyHost:= FALSE, sClientCert:= '', sClientCert_Key:= '', sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
```

---

2.3. FTPS 125
END_VAR

FbFTPS_FileRead(
    xTrigger:= Trigger,
    sServer:= '172.29.233.1',
    uiPort:= 0,
    sUser:= 'user',
    sPassword:= 'password',
    xActiveMode:= FALSE,
    sActivePort:= '',
    typSSL_Options := typSSL_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sLocalPath:= 'Read.txt',
    sRemotePath:= 'AppendFromMem.txt',
    xBusy=> xBusy,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

2.3.7 FbFTPS_FileReadToMem (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inout</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should read. For a path form the home directory, the path should start without slash. For a absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td>udiRxBufferSize</td>
<td>UDINT</td>
<td></td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td></td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td>udiNBytesTransfered</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Read a file

### Graphical Illustration

2.3. FTPS
Function description

This function block reads the data form the remote file, that is specified by the input parameter `sRemotePath`, and stored it in the `RxBuffer`, that is specified by the input parameter `pRxBuffer` and `udiRxBufferSize`. If the remote file not exist or the parameter `pRxBuffer` and `udiRxBufferSize` are 0 then an error occur.

Transition to TRUE on `xTrigger` triggers the process to remove a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
FbFTPS_FileReadToMem : WagoAppFTP.FbFTPS_FileReadToMem;

Trigger : BOOL := TRUE;

typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
  sCA_Path:= '',
  xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= '');

xBusy: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

sRxBuffer : STRING(255);
END_VAR

FbFTPS_FileReadToMem(
  xTrigger:= Trigger,
```
WagoAppFTP, Release 1.3.1.2

sServer:= '172.29.233.1',
uiPort:= 0,
sUser:= 'user',
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sRemotePath:= 'AppendFromMem.txt',
pRxBuffer:= ADR(sRxBuffer),
udiRxBufferSize:= SIZEOF(sRxBuffer),
xBusy=> xBusy,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransfered=> udiNBytesTransfered);

2.3.8 FbFTPS_FileRemove (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, that should remove. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

2.3. FTPS
Function
Remove a file

Graphical Illustration

FunctionBlock

Fig. 2.65: Graphical Interface of FbFTPS_FileRemove

Function description
This function block remove the file, that is specified by sRemotePath. An error is returned if the file does not exist, the file cannot be removed, or if the user does not have the appropriate privilege level.

Transition to TRUE on xTrigger triggers the process to remove a remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example
Example without host and peer verification.

```
VAR
  FbFTPS_FileRemove : WagoAppFTP.FbFTPS_FileRemove;

  Trigger : BOOL := TRUE;

  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');

  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPS_FileRemove( xTrigger:= Trigger,
  sServer:= '172.29.233.1',
  uiPort:= 0,
```

2.3. FTPS
sUser:= 'user',
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sRemotePath: = 'Append.txt',
xBusy=> xBusy,
xError=> xError,
oStatus=> oStatus);

2.3.9 FbFTPS_FileRename (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>‘0’</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType-s-Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server, that define the file that should renamed. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td></td>
<td>sNewFileName</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server, that define the new file name. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

Function
Rename a file

**Graphical Illustration**

![Graphical Interface of FbFTPS_FileRename](image)

**Fig. 2.66: Graphical Interface of FbFTPS_FileRename**

**Function description**

This function block rename the file, that is defined by the input parameter `sOldFileName`, to the name, that is defined by the input parameter `sNewFileName`.

Transition to TRUE on `xTrigger` triggers the process to rename a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbFTPS_FileRename : WagoAppFTP.FbFTPS_FileRename;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (sCA_Cert:= '',
  sCA_Path:= '', xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= ' ');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbFTPS_FileRename(
  xTrigger:= Trigger,
  sServer:= '172.29.233.1',
  uiPort:= 0,
  sUser:= 'user',
  sOldFileName:= 'oldfile.txt',
  sNewFileName:= 'newfile.txt',
  typSSL_Options := (sCA_Cert:= '',
  sCA_Path:= '', xVerifyPeer:= FALSE,
  xVerifyHost:= FALSE,
  sClientCert:= '',
  sClientCert_Key:= '',
  sClientCert_KeyPasswd:= ' ');
)
```

2.3. FTPS
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sOldFileName:= 'Append.txt',
sNewFileName:= 'Rename.txt',
xBusy:= xBusy,
xEr1or:= xError,
oStatus:= oStatus);

2.3.10 FbFTPS_FileWrite (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoType- s- Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the file, the local file should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘/’.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSy- sError- Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytes- Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>
Function
Write a file

Graphical Illustration

![FunctionBlock Diagram]

**Fig. 2.67: Graphical Interface of FbFTPS_FileWrite**

**Function description**

This function block writes the data from the local file, that is specified by the input parameter `sLocalPath`, to the remote file, that is specified by the input parameter `sRemotePath`.

Transition to TRUE on `xTrigger` triggers the process to write the local file to the remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For “`sLocalPath`” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbFTPS_FileWrite : WagoAppFTP.FbFTPS_FileWrite;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
```

2.3. FTPS  134
xBusy: BOOL;
addError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesTransfered: UDINT;

END_VAR

FbFTPS_FileWrite{
xTrigger:= Trigger,
sServer:= '172.29.233.1',
uiPort:= 0,
sUser:= 'user',
sPassword:= 'password',
xActiveMode:= FALSE,
sActivePort:= '',
typSSL_Options := typSSL_Options,
tKeepAlive:= T#5S,
tTimeout:= T#5S,
sLocalPath:= 'HOME://testbench/myfirstfile.txt',
sRemotePath:= 'Write.txt',
xCreateDirs:= FALSE,
xBusy=> xBusy,
addError=> xError,
oStatus=> oStatus,
udiNBytesTransfered=> udiNBytesTransfered);

2.3.11 FbFTPS_FileWriteFromMem (FB)

Interface variables
<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td></td>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a ‘-’ to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td></td>
<td>typSSL_Options</td>
<td>WagoTypes</td>
<td></td>
<td>Path on the remote server to the file, the data form the transfer buffer should write to. For a path form the home directory, the path should start without slash. For a absolute path the path should start with ‘//’.</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath.</td>
</tr>
</tbody>
</table>

| Output | xBusy        | BOOL        |         | True while not terminated.                                                |
|        | xError       | BOOL        |         | Signals presence of an error.                                             |
|        | oStatus      | WagoSysErrorBase.FbResult |         | Execution state or error code                                             |
|        | udiNBytesTransfered | UDINT |         | Number of Bytes, that are currently transferred                           |

**Function**

Write a file

**Graphical Illustration**

2.3. FTPS 136
**Function description**

This function block writes the data from the TxBuffer to a file on the remote host specified by `sRemotePath`. Transition to `TRUE` on `xTrigger` triggers the process to write the transfer buffer to the remote file. The function block resets `xTrigger` to `FALSE` again after it has finished the process. If something went wrong during the process, `xTrigger` resets to `FALSE` and `xError` is set to `TRUE`. The output `xBusy` indicates that the function block is still processing.

**Example:**
Example without host and peer verification.

```plaintext
VAR
  FbFTPS_FileWriteFromMem : WagoAppFTP.FbFTPS_FileWriteFromMem;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransferred: UDINT;
 FONT
  sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbFTPS_FileWriteFromMem(
  xTrigger:= Trigger,
  sServer:= '172.29.233.1',
  uiPort:= 0,
  typSSL_Options
);```

Fig. 2.68: Graphical Interface of FbFTPS_FileWriteFromMem
2.3.12 FbFTPS_List (FB)

Interface variables
## Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td>xActiveMode</td>
<td>BOOL</td>
<td>FALSE</td>
<td>If xActiveMode is true then the FTP-Client try to connect in active mode.</td>
</tr>
<tr>
<td>sActivePort</td>
<td>STRING(255)</td>
<td>'0'</td>
<td>If the active mode is enabled, this tells the remote server to connect to the specified port, optionally followed by a '-' to specify a port range. If the port specified is 0, the operating system will pick a free port.</td>
</tr>
<tr>
<td>typSSL_Options</td>
<td>WagoType-Curl.typSSL_Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending Keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>Path on the remote server to the directory, the should list. For a path form the home directory, the path should start without slash. For an absolute path the path should start with '//'.</td>
</tr>
<tr>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td>udiRxBuffer-Size</td>
<td>UDINT</td>
<td></td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td>sFileSpec</td>
<td>STRING(255)</td>
<td></td>
<td>Specify a specific file or type. e.g. ‘*.txt’</td>
</tr>
<tr>
<td>xWithDetails</td>
<td>BOOL</td>
<td></td>
<td>List more than only the names of the files or directories</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

List files and directories

### Graphical Illustration

2.3. FTPS
Function description

This function block performs a directory listing. The listing includes only those files that match the specification stored in sFileSpec. The input parameter pRxBuffer is pointing to the first element of the buffer and the buffer size is defined by the input parameter udiRxBufferSize.

If more details of the files and directories should be listed, the input parameter xWithDetails must be set to true.

Transition to TRUE on xTrigger triggers the process to make a directory listing. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example:

Example without host and peer verification.

```plaintext
VAR
  FbFTPS_List : WagoAppFTP.FbFTPS_List;
  Trigger : BOOL := TRUE;
  typSSL_Options: WagoAppFTP.WagoTypesCurl.typSSL_Options := (
    sCA_Cert:= '',
    sCA_Path:= '',
    xVerifyPeer:= FALSE,
    xVerifyHost:= FALSE,
    sClientCert:= '',
    sClientCert_Key:= '',
    sClientCert_KeyPasswd:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransferred: UDINT;
  sRxBuffer : STRING(255);
END_VAR
```

2.3. FTPS
2.4 SFTP

2.4.1 FbSFTP (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. '192.168.1.17'</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>typSSH_Options</td>
<td>WagoType-s-Curl.typSSH_Options</td>
<td>Options for the authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xDone</td>
<td>BOOL</td>
<td></td>
<td>Process successfully performed.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Indicates an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-sError-Base.FbResult</td>
<td>Execution state or error code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>udiNBytesToTransfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytesTransferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes that are currently transferred</td>
</tr>
</tbody>
</table>

Function
This function block provides SFTP-Client-Services.

**Graphical Illustration**

![Graphical Interface of FbSFTP](image)

**Function Description**

The FbSFTP is the base function block for SFTP-Client-Services. The Fb provide functions like read and write files, rename or remove files and create or remove directories.

The inputs of the FbSFTP handle the general information for the SFTP connection and the outputs signal the process and the result of the operations. The operations are triggered by call the specific method for the operation. After a call the methods will return immediately and the SFTP start the operation.

**Note:** This function block must called cyclic.

**Example:**

Example with authentication with password

```plaintext
VAR
FbSFTP : WagoAppFTP.FbSFTP;

typSSH_Options: WagoTypesCurl.typSSH_Options := (  
xAuthPassword:= TRUE,  //Authentication with password is enabled
xAuthKey:= FALSE,
sPrivateKey:= '',
sPassphrase:= '',
xAuthHost:= FALSE,
sKnownHosts:= '');

xBusy: BOOL;
xDone: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

iState : INT;

//List
sListBuffer : STRING(255);
//Write
sTxBuffer : STRING(255) := 'Hello World!';
//Read
sRxBuffer : STRING(255);
```

2.4. SFTP
WagoAppFTP, Release 1.3.1.2

END_VAR

FbSFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiBytesTransfered=> udiBytesTransfered);

IF FbSFTP.xDone = TRUE THEN
    iState := iState + 1;
ELSIF FbSFTP.xError = TRUE THEN
    iState := 99;
END_IF

IF FbSFTP.xBusy = FALSE THEN
    CASE iState OF
        0:
            //Makes a directory listing of the user's home directory.
            FbSFTP.List(sRemotePath := '/root', pRxBuffer := ADR(sListBuffer),
            udiRxBufferSize := SIZEOF(sListBuffer), xWithDetails := FALSE);
        1:
            //Creates the directory 'NewDir' and the file 'HelloWorld.txt'. Writes
            //the data from sTxBuffer into the new file.
            FbSFTP.FileWriteFromMem(sRemotePath := '/root/NewDir/HelloWorld.txt',
            pTxBuffer := ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
        2:
            //Create a new directory.
            FbSFTP.DirCreate(sRemotePath := '/root/NewDir/NewDir2');
        3:
            //Makes a directory.
            FbSFTP.List(sRemotePath := '/root/NewDir', pRxBuffer := ADR(sListBuffer),
            udiRxBufferSize := SIZEOF(sListBuffer), xWithDetails := TRUE);
        4:
            //Rename and replace the file.
            FbSFTP.FileRename(sOldFileName := '/root/NewDir/HelloWorld.txt',
            sNewFileName := '/root/NewDir/NewDir2/Rename.txt');
        5:
            //Reads the data from the file.
            FbSFTP.FileReadToMem(sRemotePath := '/root/NewDir/NewDir2/Rename.txt',
            pRxBuffer := ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));
        6:
            //Removes the file.
            FbSFTP.FileRemove(sRemotePath := '/root/NewDir/NewDir2/Rename.txt');
        7:
            //Removes a directory
            FbSFTP.DirRemove(sRemotePath := '/root/NewDir/NewDir2');
        8:
            //Removes a directory
            FbSFTP.DirRemove(sRemotePath := '/root/NewDir/');
        99:
            //Some Error Handling
    END_CASE
**FbSFTP.DirCreate (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirCreate</td>
<td>WagoTypes.eResultCode</td>
<td>The absolute path on the remote server with the directory, that should create.</td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server with the directory, that should create.</td>
</tr>
</tbody>
</table>

*Function*

Create a directory

**Graphical Illustration**

![Graphical Interface of FbSFTP.DirCreate](image.png)

**Function description**

This method starts an operation to create a new directory specified by `sRemotePath`. The method create only the last directory in the path. If a directory in the path is missing or the directory, that should create, already exists an error occur.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE WagoTypes.eResultCode.EINVAL</td>
</tr>
</tbody>
</table>

**Example:**

Example with authentication with password

```FORTRAN
VAR
    FbSFTP : WagoAppFTP.FbSFTP;
    typSSH_Options: WagoTypesCurl.typSSH_Options := (
        xAuthPassword:= TRUE,  //Authentication with password is enabled
        xAuthKey:= FALSE,  
        sPrivateKey:= '',
        sPassphrase:= '',
        xAuthHost:= FALSE, 
        sKnownHosts:= '');
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
```

---

2.4. SFTP
WagoAppFTP, Release 1.3.1.2

```plaintext
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

xD0 := BOOL;

END_VAR

FbSFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Create a new directory.
    FbSFTP.DirCreate(sRemotePath := '/root/NewDir');
    xDo := FALSE;
END_IF

FbSFTP.DirRemove (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>DirRemove</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the directory that should be removed.</td>
</tr>
</tbody>
</table>

Remove a directory

Graphical Illustration

Fig. 2.72: Graphical Interface of FbSFTP.DirRemove

Function description

This method starts an operation to remove the directory, that is specified by sRemotePath. An error is returned if the directory does not exist, the directory cannot be removed (in use, or not empty), or if the user does not have the appropriate privilege level.

2.4. SFTP 145
The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:  

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

Example:
Example with authentication with password

```plaintext
VAR
  FbSFTP : WagoAppFTP.FbSFTP;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE, //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo := BOOL;
END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Removes a directory
  FbSFTP.DirRemove(sRemotePath := '/root/NewDir');
  xDo := FALSE;
END_IF

FbSFTP.FileAppend (METH)

Interface variables

2.4. SFTP
Function

Append a local file to a remote file

Graphical Illustration

![Graphical Illustration](image.png)

**Function description**

This method starts an operation to append a local file specified by `sLocalPath` to the end of a file on the remote host specified by `sRemotePath`. If the remote file doesn’t exist, it will be created.

For “`sLocalPath`” you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>WagoTypes.eResultCode.OK</th>
<th>Success, operation has been started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.ENOENT</td>
<td>The local file doesn’t exist</td>
</tr>
</tbody>
</table>

**Example:**

Example with authentication with password

```plaintext
VAR
   FbSFTP : WagoAppFTP.FbSFTP;
   typSSH_Options: WagoTypesCurl.typSSH_Options := (
      xAuthPassword:= TRUE, //Authentication with password is enabled
      xAuthKey:= FALSE,
      sPrivateKey:= '',
      sPassphrase:= '',
      xAuthHost:= FALSE,
      sKnownHosts:= '');
   xBusy: BOOL;
```

2.4. SFTP
xFbSFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone => xDone,
    xError => xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered => udiNBytesTransfered);

IF xDo = TRUE THEN
    // Append the local file to the remote file
    FbSFTP.FileAppend(sLocalPath := 'HOME://Some.txt', sRemotePath := '/root/HelloWorld.txt', xCreateDirs := TRUE);
    xDo := FALSE;
END_IF

FbSFTP.FileAppendFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileAppendFromMem</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the file, the data form the transfer buffer should append to.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Append a data from a buffer to a remote file

Graphical Illustration
Function description

This method starts an operation to append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

- WagoTypes.eResultCode.OK: Success, operation has been started.
- WagoTypes.eResultCode.EBUSY: The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVAL: The input is invalid.

Example:

Example with authentication with password

```pascal
VAR
  FbSFTP : WagoAppFTP.FbSFTP;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE, //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo := BOOL;

  //Append
  sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
```
xF done => xDone,
xF error => xError,
oStatus => oStatus,
udiNBytesToTransfer => udiNBytesToTransfer,
udiNBytesTransfered => udiNBytesTransfered);

IF xDo = TRUE THEN
  //Append the data from sTxBuffer into the remote file.
  FbSFTP.FileAppendFromMem(sRemotePath := '/root/HelloWorld.txt', pTxBuffer :=
ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);
  xDo := FALSE;
END_IF

FbSFTP.FileRead (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRead</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file, in that the remote file will be written.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the file, that should read.</td>
</tr>
</tbody>
</table>

Function
Read a file

Graphical Illustration

![Method diagram](image)

Fig. 2.75: Graphical Interface of FbSFTP.FileRead

Function description
This method starts an operation to reads the data form the remote file, that is specified by the input parameter sRemotePath, and stored it in the local file, that is specified by the input parameter sLocalPath.

For "sLocalPath" you can use the following prefixes, known form WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

WagoTypes.eResultCode.OK Success, operation has been started.
WagoTypes.eResultCode.EBUSY
The Fb is already in process. No new operation can start while xBusy is TRUE Wag-oTypes.eResultCode.EINVAL The input is invalid

Example:

Example with authentication with password

VAR

  FbSFTP : WagoAppFTP.FbSFTP;

  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivatekey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;

  xDo := BOOL;

END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Reads the data from the file.
  FbSFTP.FileRead(sLocalPath := 'HOME://Some.txt', sRemotePath := '/root/
  HelloWorld.txt');

  xDo := FALSE;
END_IF

FbSFTP.FileReadToMem (METH)

Interface variables
### Function

Read a file

### Graphical Illustration

![Graphical Interface of FbSFTP.FileReadToMem](image)

Function description

This method starts an operation to read the data from the remote file, that is specified by the input parameter `sRemotePath`, and stored it in the RxBuffer, that is specified by the input parameter `pRxBuffer` and `udiRxBufferSize`.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINV</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

Example:

Example with authentication with password

```plaintext
VAR
    FbSFTP : WagoAppFTP.FbSFTP;
    typSSH_Options: WagoTypesCurl.typSSH_Options := {
        xAuthPassword:= TRUE,  //Authentication with password is enabled
        xAuthKey:= FALSE,
        sPrivateKey:= '',
        sPassphrase:= '',
        xAuthHost:= FALSE,
        sKnownHosts:= '');</n
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
```

---

2.4. SFTP

152
WagoAppFTP, Release 1.3.1.2

```plaintext
udiNBytesTransfered: UDINT;

xDo := BOOL;

//Read
sRxBuffer : STRING(255);
END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Reads the data from the file.
  FbSFTP.FileReadToMem(sRemotePath := '/root/HelloWorld.txt', pRxBuffer :=
  ADR(sRxBuffer), udiRxBufferSize := SIZEOF(sRxBuffer));
  xDo := FALSE;
END_IF
```

**FbSFTP.FileRemove (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileRemove</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the file, that should remove.</td>
</tr>
</tbody>
</table>

**Function**

Remove a file

**Graphical Illustration**

![Graphical Interface of FbSFTP.FileRemove](image)

Fig. 2.77: Graphical Interface of FbSFTP.FileRemove

**Function description**

This method starts an operation to remove the file, that is specified by sRemotePath. An error is returned if the file does not exists, the file cannot be removed, or if the user does not have the appropriate privilege level.

2.4. SFTP
The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Result Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while xBusy is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid.</td>
</tr>
</tbody>
</table>

Example:

Example with authentication with password

```plaintext
VAR
    FbSFTP : WagoAppFTP.FbSFTP;
    typSSH_Options: WagoTypesCurl.typSSH_Options := (
        xAuthPassword:= TRUE, //Authentication with password is enabled
        xAuthKey:= FALSE,
        sPrivateKey:= '',
        sPassphrase:= '',
        xAuthHost:= FALSE,
        sKnownHosts:= '');
    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;
    xDo := BOOL;
END_VAR

FbSFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Removes the file.
    FbSFTP.FileRemove(sRemotePath := '/root/HelloWorld.txt');
    xDo := FALSE;
END_IF

FbSFTP.FileRename (METH)

Interface variables
### Function

Rename a file

### Graphical Illustration

![Method](image)

Fig. 2.78: Graphical Interface of FbSFTP.FileRename

### Function description

This method starts an operation to rename the file, that is defined by the input parameter `sOldFileName`, to the name, that is defined by the input parameter `sNewFileName`.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>Success, operation has been started</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>The Fb is already in process. No new operation can start while <code>xBusy</code> is TRUE.</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>The input is invalid</td>
</tr>
</tbody>
</table>

### Example:

Example with authentication with password

```plaintext
VAR
    FbSFTP : WagoAppFTP.FbSFTP;
    typSSH_Options: WagoTypesCurl.typSSH_Options := (xAuthPassword:= TRUE, //Authentication with password is enabled
        xAuthKey:= FALSE,
        sPrivateKey:= '',
        sPassphrase:= '',
        xAuthHost:= FALSE,
        sKnownHosts:= '');

    xBusy: BOOL;
    xDone: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
    udiNBytesToTransfer: UDINT;
    udiNBytesTransfered: UDINT;

    xDo := BOOL;
END_VAR
```
FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransferred=> udiNBytesTransferred);

IF xDo = TRUE THEN
  //Rename and replace the file.
  FbSFTP.FileRename(sOldFileName := '/root/HelloWorld.txt', sNewFileName := '/root/Rename.txt');
  xDo := FALSE;
END_IF

FbSFTP.FileWrite (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the file, the local file should write to.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration

```
METHOD
FbSFTP.FileWrite
  sLocalPath STRING eResultCode FileWrite
  sRemotePath STRING
  xCreateDirs BOOL
```

Fig. 2.79: Graphical Interface of FbSFTP.FileWrite

Function description

This method starts an operation to write the data form the local file, that is specified by the input parameter sLocalPath, to the remote file, that is specified by the input parameter sRemotePath. If the remote file exit, it will be overwritten.

For “sLocalPath” you can use the following prefixes, known form WagoAppFileDir:
The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

- **WagoTypes.eResultCode.OK** Success, operation has been started.
- **WagoTypes.eResultCode.EBUSY** The Fb is already in process. No new operation can start while `xBusy` is `TRUE`.
- **WagoTypes.eResultCode.EINV AL** The input is invalid.
- **WagoTypes.eResultCode.ENOENT** The local file doesn’t exist.

Example:

Example with authentication with password

```pascal
VAR
  FbSFTP : WagoAppFTP.FbSFTP;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE, //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');

  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;

  xDo := BOOL;
END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  xBusy=> xBusy,
  xDone=> xDone,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Write the local file to the remote server
  FbSFTP.FileWrite(sLocalPath := 'HOME://Some.txt', sRemotePath := '/root/HelloWorld.txt', xCreateDirs := TRUE);
  xDo := FALSE;
END_IF
```

2.4. SFTP
FbSFTP.FileWriteFromMem (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>FileWrite-FromMem</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the file, the data form the transfer buffer should write to.</td>
</tr>
<tr>
<td></td>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
</tbody>
</table>

Function

Write a file

Graphical Illustration

![Graphical Interface of FbSFTP.FileWriteFromMem](image)

Function description

This method starts an operation to write the data from the TxBuffer to a file on the remote host specified by sRemotePath. If the remote file exist, it will be overwritten.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

- WagoTypes.eResultCode.OK: Success, operation has been started.
- WagoTypes.eResultCode.EBUSY: The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINV: The input is invalid.

Example:

Example with authentication with password

```plaintext
VAR
  FbSFTP : WagoAppFTP.FbSFTP;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
```

2.4. SFTP
xBusy: BOOL;
xDone: BOOL;
xDone: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesToTransfer: UDINT;
udiNBytesTransfered: UDINT;

//Write
sTxBuffer : STRING(255) := 'Hello World!';
END_VAR

FbSFTP(
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    xBusy=> xBusy,
    xDone=> xDone,
    xError=> xError,
    oStatus=> oStatus,
    udiNBytesToTransfer=> udiNBytesToTransfer,
    udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
    //Creates the directory 'NewDir' and the file 'HelloWorld.txt'. Writes the
    //data from sTxBuffer into the new file.
    FbSFTP/FileWriteFromMem(sRemotePath := '/root/HelloWorld.txt', pTxBuffer :=
ADR(sTxBuffer), udiTxNBytes := LENGTH(sTxBuffer), xCreateDirs := TRUE);

    xDo := FALSE;
END_IF

FbSFTP.List (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>List</td>
<td>Wago-Types.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td>The absolute path on the remote server to the directory, the should list.</td>
</tr>
<tr>
<td></td>
<td>pRxBuffer</td>
<td>POINTER TO BYTE</td>
<td>Address of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>udiRxBuffer-Size</td>
<td>UDINT</td>
<td>Size of the receive buffer</td>
</tr>
<tr>
<td></td>
<td>xWithDetails</td>
<td>BOOL</td>
<td>List more than only the names of the files or directories</td>
</tr>
</tbody>
</table>

Function

List files and directories

Graphical Illustration
Function description

This method starts an operation to perform a directory listing. The input parameter pRxBuffer is pointing to the first element of the buffer and the buffersize is defined by the input parameter udiRxBufferSize.

If more details of the files and directories should list, the input parameter xWithDetails must set to true.

The status and result of the operation are displayed by the outputs of the FbSFTP.

This method has the following return values:

- WagoTypes.eResultCode.OK: Success, operation has been started.
- WagoTypes.eResultCode.EBUSY: The Fb is already in process. No new operation can start while xBusy is TRUE.
- WagoTypes.eResultCode.EINVAL: The input is invalid.

Example:

Example with authentication with password

```plaintext
VAR
  FbSFTP : WagoAppFTP.FbSFTP;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE, //Authentication with password is enabled
    xAuthKey:= FALSE,  
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xDone: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  xDo := BOOL;

  //List
  sListBuffer : STRING(255);
END_VAR

FbSFTP(
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
```
WagoAppFTP, Release 1.3.1.2

```plaintext
tTimeout:= T#5S,
xBusy=> xBusy,
xDone=> xDone,
xError=> xError,
oStatus=> oStatus,
udiNBytesToTransfer=> udiNBytesToTransfer,
udiNBytesTransfered=> udiNBytesTransfered);

IF xDo = TRUE THEN
  //Makes a directory listing of the user’s home directory.
  FbSFTP.List(sRemotePath := '/root', pRxBuffer := ADR(sListBuffer),
  udiRxBufferSize := SIZEOF(sListBuffer), xWithDetails := FALSE);
END_IF
```

2.4.2 FbSFTP_DirCreate (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>Input</td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>typSSH_Options</td>
<td>WagoType-ss-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server with the directory, that should create.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-sError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

*Function*
Create a directory

*Graphical Illustration*
Fig. 2.82: Graphical Interface of FbSFTP_DirCreate

**Function description**

This function block creates a new directory specified by `sRemotePath`. The Fb creates only the last directory in the path. If a directory in the path is missing or the directory that should create, already exists, an error occurs.

Transition to **TRUE** on `xTrigger` triggers the process to create a new directory. The function block resets `xTrigger` to **FALSE** again after it has finished the process. If something went wrong during the process, `xTrigger` resets to **FALSE** and `xError` is set to **TRUE**. The output `xBusy` indicates that the function block is still processing.

**Example**

Example without host and peer verification.

```plaintext
VAR
    FbSFTP_DirCreate : WagoAppFTP.FbSFTP_DirCreate;
    Trigger : BOOL := TRUE;
    typSSH_Options: WagoTypesCurl.typSSH_Options := (
        xAuthPassword:= TRUE, "Authentication with password is enabled"
        xAuthKey:= FALSE,
        sPrivateKey:= '',
        sPassphrase:= '',
        xAuthHost:= FALSE,
        sKnownHosts:= '');
    xBusy: BOOL;
    xError: BOOL;
    oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbSFTP_DirCreate{ 
    xTrigger:= Trigger,
    sServer:= '192.168.1.17',
    uiPort:= 0,
    sUser:= 'root',
    sPassword:= 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive:= T#5S,
    tTimeout:= T#5S,
    sRemotePath:= '/root/NewDir',
    xBusy=> xBusy,
    xError=> xError,
    oStatus=> oStatus);
```
2.4.3 FbSFTP_DirRemove (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>typSSH_Options</td>
<td>WagoType-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If $tKeepAlive = T#0S$ the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If $tTimeout = 0$, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server with directorys, that should be removed.</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
</tbody>
</table>

Function*

Remove a directory

Graphical Illustration

![FunctionBlock](image)

Fig. 2.83: Graphical Interface of FbSFTP_DirRemove

Function description

This functionblock remove remove the directory, that is specified by $sRemotePath$. An error is returned if the directory does not exists, the directory cannot be removed (in use, or not empty), or if the user does not have the appropriate privilege level.
Transition to TRUE on xTrigger triggers the process to remove a directory. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

Example without host and peer verification.

```
VAR
  FbSFTP_DirRemove : WagoAppFTP.FbSFTP_DirRemove;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbSFTP_DirRemove(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= '/root/NewDir',
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus);
```

2.4.4 FbSFTP_FileAppend (FB)

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblok if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>typSSH_Options</td>
<td>WagoType-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>TIME#10s0ms The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should append to the remote file.</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server to the file, the local file should append to.</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Append a local file to a remote file

**Graphical Illustration**

![Graphical Interface of FbSFTP_FileAppend](image)

Fig. 2.84: Graphical Interface of FbSFTP_FileAppend

### Function description

2.4. SFTP
This function block appends a local file specified by \texttt{sLocalPath} to the end of a file on the remote host specified by \texttt{sRemotePath}. If the remote file doesn’t exist, it will be created.

Transition to TRUE on \texttt{xTrigger} triggers the process to append the local file to the remote file. The function block resets \texttt{xTrigger} to FALSE again after it has finished the process. If something went wrong during the process, \texttt{xTrigger} resets to FALSE and \texttt{xError} is set to TRUE. The output \texttt{xBusy} indicates that the function block is still processing.

For “\texttt{sLocalPath}” you can use the following prefixes, known from \texttt{WagoAppFileDir}:

- \texttt{HOME://}
- \texttt{CARD://}
- \texttt{TEMP://}
- \texttt{ROOT://}

Example

Example without host and peer verification.

```plaintext
VAR
FbSFTP_FileAppend : WagoAppFTP.FbSFTP_FileAppend;

Trigger : BOOL := TRUE;

typSSH_Options: WagoTypesCurl.typSSH_Options := (
  xAuthPassword:= TRUE, //Authentication with password is enabled
  xAuthKey:= FALSE,
  sPrivateKey:= '',
  sPassphrase:= '',
  xAuthHost:= FALSE,
  sKnownHosts:= '');

xBusy: BOOL;
xError: BOOL;
oStatus: WagoSysErrorBase.FbResult;
udiNBytesTransfered: UDINT;
END_VAR

FbSFTP_FileAppend(\n  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options:= typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sLocalPath:= 'HOME://Some.txt',
  sRemotePath:= '/root/HelloWorld.txt',
  xCreateDirs:= FALSE,
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesTransfered=> udiNBytesTransfered);
```

2.4.5 \texttt{FbSFTP\_FileAppendFromMem (FB)}

Interface variables
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>typSSH_Options</td>
<td>WagoType-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td></td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server to the file, the data form the transfer buffer should append to.</td>
</tr>
<tr>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSysErrorBase.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Append a data from a buffer to a remote file

### Graphical Illustration

![Graphical Interface of FbSFTP_FileAppendFromMem](image)

**Fig. 2.85:** Graphical Interface of FbSFTP_FileAppendFromMem
Function description

This function block append data from the TxBuffer to the end of a file on the remote host specified by sRemotePath. If the remote file doesn’t exist, it will be created.

Transition to TRUE on xTrigger triggers the process to append the transfer buffer to the remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

Example

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_FileAppendFromMem : WagoAppFTP.FbSFTP_FileAppendFromMem;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;
  sTxBuffer : STRING(255);
END_VAR

FbSFTP_FileAppendFromMem(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= '/root/AppendFromMem.txt',
  pTxBuffer:= ADR(sTxBuffer),
  udiTxNBytes:= LENGTH(sTxBuffer),
  xCreateDirs:= FALSE,
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesTransfered=> udiNBytesTransfered);
```

2.4.6 FbSFTP_FileRead (FB)

Interface variables
<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>typSSH_Options</td>
<td>WagoType-s-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file, in which the remote file will be written.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server to the file, that should read.</td>
</tr>
<tr>
<td></td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>Output</td>
<td>oStatus</td>
<td>WagoSys-Error-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiN-BytesTo-Transfer</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes to transfer.</td>
</tr>
<tr>
<td></td>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of.Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

**Function**

Read a file

**Graphical Illustration**

![Graphical Interface of FbSFTP_FileRead](image)

**Function description**

2.4. SFTP
This function block reads the data from the remote file, that is specified by the input parameter `sRemotePath`, and stores it in the local file, that is specified by the input parameter `sLocalPath`.

Transition to TRUE on `xTrigger` triggers the process to read the remote file to a local file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For "sLocalPath" you can use the following prefixes, known from WagoAppFileDir:

- HOME://
- CARD://
- TEMP://
- ROOT://

Example

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_FileRead : WagoAppFTP.FbSFTP_FileRead;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (xAuthPassword:= TRUE, //Authentication with password is enabled
  xAuthKey:= FALSE,
  sPrivateKey:= '',
  sPassphrase:= '',
  xAuthHost:= FALSE,
  sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
END_VAR

FbSFTP_FileRead(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sLocalPath:= 'HOME://Some.txt',
  sRemotePath:= '/root/HelloWorld.txt',
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);
```

### 2.4.7 FbSFTP_FileReadToMem (FB)

**Interface variables**
### Scope

#### Inout
- **xTrigger**: BOOL
  - Will be reset by the function block if it is terminated.

#### Input
- **sServer**: STRING(255)
  - Name or IP of the Server, e.g. ‘192.168.1.17’
- **uiPort**: UINT
  - 0
  - Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.
- **sUser**: STRING(255)
  - Authentication
- **sPassword**: STRING(255)
  - Authentication
- **typSSH_Options**: WagoType-Curl.typSSH_Options
  - Options for the authentication
- **tKeepAlive**: TIME
  - Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled
- **tTimeout**: TIME
  - The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated
- **sRemotePath**: STRING(255)
  - Path on the remote server to the file, that should read.
- **pRxBuffer**: POINTER TO BYTE
  - Address of the receive buffer
- **udiRxBufferSize**: UDINT
  - Size of the receive buffer

#### Output
- **xBusy**: BOOL
  - True while not terminated.
- **xError**: BOOL
  - Signals presence of an error.
- **oStatus**: WagoSysErrorBase.FbResult
  - Execution state or error code
- **udiBytesToTransfer**: UDINT
  - Number of Bytes to transfer.
- **udiBytesTransferred**: UDINT
  - Number of Bytes, that are currently transferred

### Function

**Read a file**

### Graphical Illustration

![Fig. 2.87: Graphical Interface of FbsFTP_FileReadToMem](image)

**FunctionBlock**

```plaintext
FbsFTP_FileReadToMem
```

- **xTrigger**: BOOL
- **sServer**: STRING
- **uiPort**: UINT
- **sUser**: STRING
- **sPassword**: STRING
- **typSSH_Options**: typSSH_Options
- **tKeepAlive**: TIME
- **tTimeout**: TIME
- **sRemotePath**: STRING
- **pRxBuffer**: Pointer
- **udiRxBufferSize**: UDINT
- **xBusy**: BOOL
- **xError**: BOOL
- **oStatus**: FbResult
- **udiBytesToTransfer**: UDINT
- **udiBytesTransferred**: UDINT

2.4. SFTP
**Function description**

This function block reads the data form the remote file, that is specified by the input parameter `sRemotePath`, and stored it in the `RxBuffer`, that is specified by the input parameter `pRxBuffer` and `udiRxBufferSize`. If the remote file not exist or the parameter `pRxBuffer` and `udiRxBufferSize` are 0 then an error occur.

Transition to TRUE on `xTrigger` triggers the process to remove a remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example**

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_FileReadToMem : WagoAppFTP.FbSFTP_FileReadToMem;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');

  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesToTransfer: UDINT;
  udiNBytesTransfered: UDINT;
  sRxBuffer : STRING(255);
END_VAR

FbSFTP_FileReadToMem(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= '/root/HelloWorld.txt',
  pRxBuffer:= ADR(sRxBuffer),
  udiRxBufferSize:= SIZEOF(sRxBuffer),
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesToTransfer=> udiNBytesToTransfer,
  udiNBytesTransfered=> udiNBytesTransfered);
```

**2.4.8 FbSFTP_FileRemove (FB)**

**Interface variables**
Function
Remove a file

Graphical Illustration

![Graphical Interface of FbSFTP_FileRemove](image)

Fig. 2.88: Graphical Interface of FbSFTP_FileRemove

Function description

This function block remove the file, that is specified by `sRemotePath`. An error is returned if the file does not exist, the file cannot be removed, or if the user does not have the appropriate privilege level.

Transition to `TRUE` on `xTrigger` triggers the process to remove a remote file. The function block resets `xTrigger` to `FALSE` again after it has finished the process. If something went wrong during the process `xTrigger` resets to `FALSE` and `xError` is set to `TRUE`. The output `xBusy` indicates that the function block is still processing.

Example

## 2.4. SFTP
Example without host and peer verification.

```plaintext
VAR

   FbSFTP_FileRemove : WagoAppFTP.FbSFTP_FileRemove;

   Trigger : BOOL := TRUE;

   typSSH_Options: WagoTypesCurl.typSSH_Options := (
      xAuthPassword:= TRUE, //Authentication with password is enabled
      xAuthKey:= FALSE,
      sPrivateKey:= '',
      sPassphrase:= '',
      xAuthHost:= FALSE,
      sKnownHosts:= '');

   xBusy: BOOL;
   xError: BOOL;
   oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbSFTP_FileRemove(
   xTrigger:= Trigger,
   sServer:= '192.168.1.17',
   uiPort:= 0,
   sUser:= 'root',
   sPassword:= 'wago',
   typSSH_Options := typSSH_Options,
   tKeepAlive:= T#5S,
   tTimeout:= T#5S,
   sRemotePath:= '/root/HelloWorld.txt',
   xBusy=> xBusy,
   xError=> xError,
   oStatus=> oStatus);
```

2.4.9 FbSFTP_FileRename (FB)

Interface variables
## Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>typSSH_Options</td>
<td>WagoType-Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td>sOldFileName</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server, that define the file that should renamed.</td>
</tr>
<tr>
<td>sNewFileName</td>
<td>STRING(255)</td>
<td></td>
<td>The absolute path on the remote server, that define the new file name.</td>
</tr>
</tbody>
</table>

## Function

**Rename a file**

**Graphical Illustration**

![Graphical Interface of FbSFTP_FileRename](image)

**Function description**

This function block rename the file, that is defined by the input parameter sOldFileName, to the name, that is defined by the input parameter sNewFileName.

Transition to TRUE on xTrigger triggers the process to rename a remote file. The function block resets xTrigger to FALSE again after it has finished the process. If something went wrong during the process
xTrigger resets to FALSE and xError is set to TRUE. The output xBusy indicates that the function block is still processing.

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_FileRename : WagoAppFTP.FbSFTP_FileRename;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,   //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
END_VAR

FbSFTP_FileRename(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sOldFileName:= '/root/HelloWorld.txt',
  sNewFileName:= '/root/Rename.txt',
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus);
```

**2.4.10 FbSFTP_FileWrite (FB)**

**Interface variables**
### Function Block

**FbSFTP_FileWrite**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the function block if it is terminated.</td>
</tr>
<tr>
<td></td>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td></td>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td></td>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>typSSH_Options</td>
<td>WagoType-&lt;br&gt;Srsl.Curl.typSSH_Options</td>
<td></td>
<td>Options for the authentication</td>
</tr>
<tr>
<td></td>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled</td>
</tr>
<tr>
<td></td>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated</td>
</tr>
<tr>
<td></td>
<td>sLocalPath</td>
<td>STRING(255)</td>
<td></td>
<td>Path to the local file that should write to the remote file.</td>
</tr>
<tr>
<td></td>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>For a path form the home directory, the path should start without slash.</td>
</tr>
<tr>
<td></td>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Output</td>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td></td>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td></td>
<td>oStatus</td>
<td>WagoSys-&lt;br&gt;sError-&lt;br&gt;Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td></td>
<td>udiNBytes-&lt;br&gt;Transfered</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Write a file

### Graphical Illustration

![Graphical Interface of FbSFTP_FileWrite](image)

Fig. 2.90: Graphical Interface of FbSFTP_FileWrite

### Function description
This function block writes the data from the local file, that is specified by the input parameter `sLocalPath`, to the remote file, that is specified by the input parameter `sRemotePath`.

Transition to TRUE on `xTrigger` triggers the process to write the local file to the remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

For `"sLocalPath"` you can use the following prefixes, known from `WagoAppFileDir`:

- HOME://
- CARD://
- TEMP://
- ROOT://

Example:

Example without host and peer verification.

```plaintext
VAR
    FbSFTP_FileWrite : WagoAppFTP.FbSFTP_FileWrite;
    Trigger : BOOL := TRUE;
    typSSH_Options : WagoTypesCurl.typSSH_Options := (
        xAuthPassword := TRUE,  //Authentication with password is enabled
        xAuthKey := FALSE,
        sPrivateKey := '',
        sPassphrase := '',
        xAuthHost := FALSE,
        sKnownHosts := '');
    xBusy : BOOL;
    xError : BOOL;
    oStatus : WagoSysErrorBase.FbResult;
    udiNBytesTransfered : UDINT;
END_VAR
FbSFTP_FileWrite(
    xTrigger := Trigger,
    sServer := '192.168.1.17',
    uiPort := 0,
    sUser := 'root',
    sPassword := 'wago',
    typSSH_Options := typSSH_Options,
    tKeepAlive := T#5S,
    tTimeout := T#5S,
    sLocalPath := 'HOME://Some.txt',
    sRemotePath := '/root/HelloWorld.txt',
    xCreateDirs := FALSE,
    xBusy => xBusy,
    xError => xError,
    oStatus => oStatus,
    udiNBytesTransfered => udiNBytesTransfered);
```

### 2.4.11 FbSFTP_FileWriteFromMem (FB)

Interface variables

---

2.4. SFTP 178
### Scope

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xTrigger</td>
<td>BOOL</td>
<td></td>
<td>Will be reset by the functionblock if it is terminated.</td>
</tr>
<tr>
<td>sServer</td>
<td>STRING(255)</td>
<td></td>
<td>Name or IP of the Server, e.g. ‘192.168.1.17’</td>
</tr>
<tr>
<td>uiPort</td>
<td>UINT</td>
<td>0</td>
<td>Number to be the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING(255)</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td>typSSH_Options</td>
<td>WagoType-s-Curl.typSSH_Options</td>
<td>Options for the authentication</td>
<td></td>
</tr>
<tr>
<td>tKeepAlive</td>
<td>TIME</td>
<td></td>
<td>Sets the interval, in seconds, that the operating system will wait between sending keepalive probes. Not all operating systems support this option. If tKeepAlive = T#0S the KeepAlive is disabled.</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>TIME#10s0ms</td>
<td>The maximum time in seconds that you allow the libcurl transfer operation to take. If tTimeout = 0, the timeout is deactivated.</td>
</tr>
<tr>
<td>sRemotePath</td>
<td>STRING(255)</td>
<td></td>
<td>For a path form the home directory, the path should start without slash.</td>
</tr>
<tr>
<td>pTxBuffer</td>
<td>POINTER TO BYTE</td>
<td></td>
<td>Address of the transfer buffer</td>
</tr>
<tr>
<td>udiTxNBytes</td>
<td>UDINT</td>
<td></td>
<td>Length of the transfer buffer</td>
</tr>
<tr>
<td>xCreateDirs</td>
<td>BOOL</td>
<td></td>
<td>Create all missing directories in the sRemotePath</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xBusy</td>
<td>BOOL</td>
<td></td>
<td>True while not terminated.</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td></td>
<td>Signals presence of an error.</td>
</tr>
<tr>
<td>oStatus</td>
<td>WagoSy-sError-Base.FbResult</td>
<td></td>
<td>Execution state or error code</td>
</tr>
<tr>
<td>udiNBytes-Transferred</td>
<td>UDINT</td>
<td></td>
<td>Number of Bytes, that are currently transferred</td>
</tr>
</tbody>
</table>

### Function

Write a file

### Graphical Illustration

![Graphical Interface of FbSFTP_FileWriteFromMem](image)

---

---
**Function description**

This function block writes the data from the TxBuffer to a file on the remote host specified by `sRemotePath`.

Transition to TRUE on `xTrigger` triggers the process to write the transfer buffer to the remote file. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_FileWriteFromMem : WagoAppFTP.FbSFTP_FileWriteFromMem;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE,  //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;
  sTxBuffer : STRING(255);
END_VAR

FbSFTP_FileWriteFromMem(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= '/root/Write.txt',
  pTxBuffer:= ADR(sTxBuffer),
  udiTxNBytes := LENGTH(sTxBuffer),
  xCreateDirs := FALSE,
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesTransfered=> udiNBytesTransfered);
```

### 2.4.12 FbSFTP_List (FB)

**Interface variables**
### Function

List files and directories

### Graphical Illustration

![Graphical Interface of FbSFTP_List]

Fig. 2.92: Graphical Interface of FbSFTP_List
**Function description**

This function block performs a directory listing. The input parameter `pRxBuffer` is pointing to the first element of the buffer and the buffersize is defined by the input parameter `udiRxBufferSize`.

If more details of the files and directories should list, the input parameter `xWithDetails` must be set to true.

Transition to TRUE on `xTrigger` triggers the process to make a directory listing. The function block resets `xTrigger` to FALSE again after it has finished the process. If something went wrong during the process, `xTrigger` resets to FALSE and `xError` is set to TRUE. The output `xBusy` indicates that the function block is still processing.

**Example:**

Example without host and peer verification.

```plaintext
VAR
  FbSFTP_List : WagoAppFTP.FbSFTP_List;
  Trigger : BOOL := TRUE;
  typSSH_Options: WagoTypesCurl.typSSH_Options := (
    xAuthPassword:= TRUE, //Authentication with password is enabled
    xAuthKey:= FALSE,
    sPrivateKey:= '',
    sPassphrase:= '',
    xAuthHost:= FALSE,
    sKnownHosts:= '');
  xBusy: BOOL;
  xError: BOOL;
  oStatus: WagoSysErrorBase.FbResult;
  udiNBytesTransfered: UDINT;
  sRxBuffer : STRING(255);
END_VAR

FbSFTP_List(
  xTrigger:= Trigger,
  sServer:= '192.168.1.17',
  uiPort:= 0,
  sUser:= 'root',
  sPassword:= 'wago',
  typSSH_Options := typSSH_Options,
  tKeepAlive:= T#5S,
  tTimeout:= T#5S,
  sRemotePath:= '/root',
  pRxBuffer:= ADR(sRxBuffer),
  udiRxBufferSize:= SIZEOF(sRxBuffer),
  xWithDetails:= FALSE,
  xBusy=> xBusy,
  xError=> xError,
  oStatus=> oStatus,
  udiNBytesTransfered=> udiNBytesTransfered);
```
3.1 Status (GVL)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>ERROR</td>
<td>ARRAY [0..23] OF WagoTypesErrorBase.typResultItem</td>
</tr>
</tbody>
</table>

**Description:** Status Information

<table>
<thead>
<tr>
<th>Value</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WagoTypes.eResultCode.OK</td>
<td>WagoTypes.eSeverity.none</td>
<td>‘OK’</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.ENOENT</td>
<td>WagoTypes.eSeverity.error</td>
<td>‘No such file or directory.’</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EINVAL</td>
<td>WagoTypes.eSeverity.error</td>
<td>‘Invalid parameter(s)’</td>
</tr>
<tr>
<td>WagoTypes.eResultCode.EBUSY</td>
<td>WagoTypes.eSeverity.error</td>
<td>‘Functionblok is still in progress’</td>
</tr>
</tbody>
</table>
**VersionHistory (GVL)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>WagoSysVersion.ProjectInfo</td>
</tr>
</tbody>
</table>

**WagoAppFTP**

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.04.2019</td>
<td>1.3.1.2</td>
<td>WAGO / u013972</td>
<td>Bugfix for string termination</td>
</tr>
<tr>
<td>08.01.2019</td>
<td>1.3.1.0</td>
<td>WAGO / u015842</td>
<td>Properties: free placeholder added</td>
</tr>
<tr>
<td>24.01.2018</td>
<td>1.3.0.3</td>
<td>WAGO / u013972</td>
<td>Change in the documentation</td>
</tr>
<tr>
<td>30.11.2017</td>
<td>1.3.0.2</td>
<td>WAGO / u013972</td>
<td>Terminate the string after List()</td>
</tr>
<tr>
<td>07.12.2016</td>
<td>1.3.0.0</td>
<td>WAGO / u013972</td>
<td>Add Fbs for SFTP-Client-Services</td>
</tr>
<tr>
<td>21.11.2016</td>
<td>1.2.0.0</td>
<td>WAGO / u013972</td>
<td>Refactor the library and add Fbs for FTPS-Client-Services and FTPES-Client-Services</td>
</tr>
<tr>
<td>17.08.2016</td>
<td>1.1.1.0</td>
<td>WAGO / u013972</td>
<td>Check the existing of the local file for write operations</td>
</tr>
<tr>
<td>04.03.2016</td>
<td>1.1.0.0</td>
<td>WAGO / u013972</td>
<td>Replace WagoSysErrorBase with WagoSysErrorBase</td>
</tr>
<tr>
<td>30.10.2015</td>
<td>1.0.4.0</td>
<td>WAGO / u013972</td>
<td>Release Version</td>
</tr>
</tbody>
</table>
This is a dictionary of all referenced libraries and their name spaces.

**Analyzezation**

*Library Identification:*

<table>
<thead>
<tr>
<th>Placeholder: Anaylzation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Resolution: Anaylzation, 3.5.2.0 (System)</td>
</tr>
<tr>
<td>Namespace: Anaylzation</td>
</tr>
</tbody>
</table>

*Library Properties:*

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: True
- Optional: False

*Library Parameter:*

Parameter: TABLE_UPPER_BOUND = 15
Parameter: STRING_LENGTH_ADDRESS = 20
Parameter: STRING_LENGTH_EXP = 255
Parameter: STRING_LENGTH_COMMENT = 255
Parameter: TABLE_SHOW_VALID_ITEMS = FALSE
Parameter: STRING_LENGTH_OUTSTRING = 255

**IecSfc**

*Library Identification:*

<table>
<thead>
<tr>
<th>Placeholder: IecSfc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Resolution: IecSfc, 3.4.2.0 (System)</td>
</tr>
<tr>
<td>Namespace: IecSfc</td>
</tr>
</tbody>
</table>

*Library Properties:*

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: True
- Optional: False
Standard

Library Identification:
Placeholder: Standard
Default Resolution: Standard, * (System)
Namespace: Standard

Library Properties:

• LinkAllContent: False
• QualifiedOnly: False
• SystemLibrary: False
• Optional: False

WagoAppString

Library Identification:
Placeholder: WagoAppString
Default Resolution: WagoAppString, * (WAGO)
Namespace: WagoAppString

Library Properties:

• LinkAllContent: False
• QualifiedOnly: False
• SystemLibrary: False
• Optional: False

WagoSysCurl

Library Identification:
Placeholder: WagoSysCurl
Default Resolution: WagoSysCurl, * (WAGO)
Namespace: WagoSysCurl

Library Properties:

• LinkAllContent: False
• QualifiedOnly: False
• SystemLibrary: False
• Optional: False

WagoSysErrorBase

Library Identification:
Placeholder: WagoSysErrorBase
Default Resolution: WagoSysErrorBase, * (WAGO)
Namespace: WagoSysErrorBase

Library Properties:
• LinkAllContent: False
• QualifiedOnly: True
• SystemLibrary: False
• Optional: False

Library Parameter:
Parameter: RES_LOG_MAX_FILESIZE = 2000
Parameter: RES_LOG_MAX_FILES = 1
Parameter: RES_LOG_MAX_ENTRIES = 200
Parameter: RES_LOG_NAME = ‘WagoAppResultLogger’

WagoSysFileDir
Library Identification:
Placeholder: WagoSysFileDir
Default Resolution: WagoSysFileDir, * (WAGO)
Namespace: WagoSysFileDir

Library Properties:

• LinkAllContent: False
• QualifiedOnly: False
• SystemLibrary: False
• Optional: False

WagoSysPlainMem
Library Identification:
Placeholder: WagoSysPlainMem
Default Resolution: WagoSysPlainMem, * (WAGO)
Namespace: WagoSysPlainMem

Library Properties:

• LinkAllContent: False
• Optional: False
• QualifiedOnly: False
• SystemLibrary: False
• PublishSymbolsInContainer: True

WagoSysVersion
Library Identification:
Name: WagoSysVersion
Version: 1.0.0.0
Company: WAGO
Namespace: WagoSysVersion

Library Properties:
WagoTypesCommon

Library Identification:
Placeholder: WagoTypesCommon
Default Resolution: WagoTypesCommon, * (WAGO)
Namespace: WagoTypes

Library Properties:

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

WagoTypesCurl

Library Identification:
Placeholder: WagoTypesCurl
Default Resolution: WagoTypesCurl, * (WAGO)
Namespace: WagoTypesCurl

Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

WagoTypesErrorBase

Library Identification:
Placeholder: WagoTypesErrorBase
Default Resolution: WagoTypesErrorBase, * (WAGO)
Namespace: WagoTypesErrorBase

Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: True
- SystemLibrary: False
- PublishSymbolsInContainer: True

© WAGO Kontaktechnik GmbH & Co. KG, Germany 2018 – All rights reserved. For the avoidance of doubt, this copyright notice does not only apply to the information above but also and primarily to the described library itself. Please note that third-party products are always mentioned without reference to intellectual property rights,
including patents, utility models, designs and trademarks, accordingly the existence of such rights cannot be excluded. WAGO is a registered trademark of WAGO Verwaltungsgesellschaft mbH.