brennenstuhl®

_ smart technology.

Premium-Web-Line V3

CGI specification

Version 3.3
03.05.2022

System requirements: 3.3.6 or later

brennenstuhl®

_ smart technology.

Table of Contents

CGI handling functions

{tc \l 2 "CGI handling functions"}{xe "CGI handling functions"}Classes

struct _sCgiReply

Typedefs

typedef enum _eCgiReplyType teCgiReplyType
typedef struct _sCgiReply tsCgiReply
typedef struct _sCgiReply * ptsCgiReply
typedef bool *(* tpfnCgi) (struct http_state *psHttpState)

Enumerations

enum _eCgiReplyType { ReplyNone, ReplyErr, ReplyFile, ReplyString, ReplyStringAlloc, ReplyHtmlString }

Functions

static bool cgiIsValidHexDigit (const char cDigit)
static bool cgiDecodeHexEscape (const char *pcEncoded, char *pcDecoded)
static bool cgiCheckDecimalParam (const char *pcValue, int32_t *pi32Value, int32_t i32Len)
bool cgiExecute (ptsHttpState psHttpState)
uint32_t cgiEncodeFormString (const char *pcDecoded, char *pcEncoded, uint32_t ui32Len)
uint32_t cgiDecodeFormString (const char *pcEncoded, char *pcDecoded, uint32_t ui32LenDecoded)
const char * cgiSelectParameter (const char *pcParams, uint32_t ui32Num)
const char * cgiGetParameterValue (const char *pcParam)
uint32_t cgiGetParameterStringLength (const char *pcValue)
const char * cgiFindParameter (const char *pcParmName, const char *pcParams)
int32_t cgiFindParameterLong (const char *pcParmName, const char *pcParams, bool *pbError)

Detailed Description

This module contains all CGI handling functions.

Typedef Documentation

{xe "ptsCgiReply:CGI handling functions"}{xe "CGI handling functions:ptsCgiReply"}ptsCgiReply

Typedef for pointer to a structure _sCgiReply.

{xe "teCgiReplyType:CGI handling functions"}{xe "CGI handling functions:teCgiReplyType"}teCgiReplyType

Typedef of structure _eCgiReplyType.

{xe "tpfnCgi:CGI handling functions"}{xe "CGI handling functions:tpfnCgi"}typedef bool *(* tpfnCgi) (struct http_state *psHttpState)

Function pointer to a cgi function.

{xe "tsCgiReply:CGI handling functions"}{xe "CGI handling functions:tsCgiReply"}tsCgiReply

Typedef of structure _sCgiReply.

Enumeration Type Documentation

{xe "_eCgiReplyType:CGI handling functions"}{xe "CGI handling functions:_eCgiReplyType"}enum _eCgiReplyType

Enumeration that defines the reply type of a cgi-function.

Enumerator:

{xe "ReplyNone:CGI handling functions"}{xe "CGI handling functions:ReplyNone"}ReplyNone	CGI-function will return nothing.
{xe "ReplyErr:CGI handling functions"}{xe "CGI handling functions:ReplyErr"}ReplyErr	CGI-function will return NULL and report an http status 400.
{xe "ReplyFile:CGI handling functions"}{xe "CGI handling functions:ReplyFile"}ReplyFile	CGI-function will return a filename.
{xe "ReplyString:CGI handling functions"}{xe "CGI handling functions:ReplyString"}ReplyString	CGI-function will return a string.
{xe "ReplyStringAlloc:CGI handling functions"}{xe "CGI handling functions:ReplyStringAlloc"}ReplyStringAlloc	CGI-function will return a string allocated from heap.
{xe "ReplyHtmlString:CGI handling functions"}{xe "CGI handling functions:ReplyHt	CGI-function will return a complete html body as string.

mlString"}ReplyHt mlString	

Function Documentation

{xe "cgiCheckDecimalParam:CGI handling functions"}{xe "CGI handling functions:cgiCheckDecimalParam"}static bool cgiCheckDecimalParam (const char * *pcValue*, int32_t * *pi32Value*, int32_t *i32Len*) `[static]`

Ensures that a string passed represents a valid decimal number and, if so, converts that number to a long.

Parameters

in	*pcValue*	points to a null terminated string which should contain an ASCII representation of a decimal number.
out	*pi32Value*	points to storage which will receive the number represented by pcValue assuming the string is a valid decimal number.
in	*i32Len*	contains the length of the pi32Value buffer.

This function determines whether or not a given string represents a valid decimal number and, if it does, converts the string into a decimal number which is returned to the caller. The String *pcValue* does not have to be terminated, as its length in *i32Len* is considered.

Returns

Returns **true** if the string is a valid representation of a decimal number or **false** if not.

{xe "cgiDecodeFormString:CGI handling functions"}{xe "CGI handling functions:cgiDecodeFormString"}uint32_t cgiDecodeFormString (const char * *pcEncoded*, char * *pcDecoded*, uint32_t *ui32LenDecoded*)

Decodes a string encoded as part of an HTTP URI.

Parameters

pcEncoded	is a pointer to a null terminated string encoded as per RFC1738, section 2.2.
pcDecoded	is a pointer to storage for the decoded, null terminated string.
ui32LenDecoded	is the number of bytes of storage pointed to by pcDecoded.

This function decodes a string which has been encoded using the method described in RFC1738, section 2.2 for URLs. If the decoded string is too long for the provided output buffer, the output will be truncated.

Returns

Returns the number of character written to the output buffer, not including the terminating NULL.

{xe "cgiDecodeHexEscape:CGI handling functions"}{xe "CGI handling functions:cgiDecodeHexEscape"}static bool cgiDecodeHexEscape (const char * *pcEncoded*, char * *pcDecoded*) `[static]`

Decodes a single xx escape sequence as an ASCII character.

Parameters

in	*pcEncoded*	points to the ``‘%’' character at the start of a three character escape sequence which represents a single ASCII character.
out	*pcDecoded*	points to a byte which will be written with the decoded character assuming the escape sequence is valid.

This function decodes a single escape sequence of the form ``‘xy’' where x and y represent hexadecimal digits. If each digit is a valid hex digit, the function writes the decoded character to the pcDecoded buffer and returns true, else it returns false.

Returns

Returns **true** on success or **false** if pcEncoded does not point to a valid escape sequence.

{xe "cgiEncodeFormString:CGI handling functions"}{xe "CGI handling functions:cgiEncodeFormString"}uint32_t cgiEncodeFormString (const char * *pcDecoded*, char * *pcEncoded*, uint32_t *ui32Len*)

Encodes a string for use within an HTML tag, escaping non alphanumeric characters.

Parameters

in	*pcDecoded*	is a pointer to a null terminated ASCII string.
out	*pcEncoded*	is a pointer to a storage for the encoded string.
in	*ui32Len*	is the number of bytes of storage pointed to by pcEncoded.

This function encodes a string, adding escapes in place of any special, non-alphanumeric characters. If the encoded string is too long for the provided output buffer, the output will be truncated.

Returns

Returns the number of characters written to the output buffer not including the terminating NULL.

{xe "cgiExecute:CGI handling functions"}{xe "CGI handling functions:cgiExecute"}bool cgiExecute (ptsHttpState *psHttpState*)

Executes a cgi function.

Parameters

in	*psHttpState*	is the instance variable of the HTTP connection.

This function executes a cgi function.

Returns

true if a cgi function has been executed, false if not.

{xe "cgiFindParameter:CGI handling functions"}{xe "CGI handling functions:cgiFindParameter"}const char * cgiFindParameter (const char * *pcParmName*, const char * *pcParams*)

Searches a list of CGI parameters for a specific parameter.

Parameters

in	*pcParmName*	is a pointer to a string containing the name of the parameter that is to be found.
in	*pcParams*	is a CGI parameter string.

This function searches an array of parameters to find the string passed in *pcParmName* . The parameter string has to be encoded in the following form:

`"<Parm1>=<Value1>&<Parm2>=<Value2>&...&<ParmN>=<ValueN>"`.

If the string is found, the pointer to that string within the *pcParams* array is returned, otherwise NULL is returned. As within forms it is allowed to have multiple check boxes with the same name, the same parameter may appear several times with different values. In this case, this function has to be called multiple times, until it returns NULL. On entry, *pcParams* is always the return value of the previous call.

Returns

Pointer to *pcParmName* within array *pcParam* or NULL if the string does not exist in the array.

{xe "cgiFindParameterLong:CGI handling functions"}{xe "CGI handling functions:cgiFindParameterLong"}int32_t cgiFindParameterLong (const char * *pcParmName*, const char * *pcParams*, bool * *pbError*)

Reads a CGI parameter as a decimal number.

Parameters

in	*pcParmName*	is a pointer to a string containing the name of the parameter that is to be found.
in	*pcParams*	is a CGI parameter string.
out	*pbError*	is a pointer that will be written to **true** if there is any error during the parameter parsing process (parameter not found, value is not a valid decimal number); otherwise it will keep its value on entry.

This function searches an array of parameters to find the string passed in *pcParmName* . The parameter string has to be encoded in the following form: `"<Parm1>=<Value1>&<Parm2>=<Value2>&...&<ParmN>=<ValueN>"` . If the string is found, the corresponding parameter value is read from *pcParams* array and checked to make sure that it is a valid decimal number. If so, the number is returned. If any error is detected, parameter *pbError* is written to **true** .

Note

pbError is NOT written if the parameter is successfully found and validated. This is to allow multiple parameters to be parsed without the caller needing to check return codes after each individual call.

Returns

Returns the value of the parameter or 0 if an error is detected (in which case **pbError* will be **true**).

{xe "cgiGetParameterStringLength:CGI handling functions"}{xe "CGI handling functions:cgiGetParameterStringLength"}uint32_t cgiGetParameterStringLength (const char * *pcValue*)

Retrieves the string length of a cgi parameter.

Parameters

in	*pcValue*	is a pointer to the string.

This function determines the string length of a cgi parameter. The syntax is "<Parm>=<Value>". The value will not be null-terminated, as it is a pointer within the whole parameter string. So it ends on "&" or a CRLF sequence. If pcValue points to the value-part of the string, the result will be the string length of the parameter value.

Returns

String length.

{xe "cgiGetParameterValue:CGI handling functions"}{xe "CGI handling functions:cgiGetParameterValue"}const char * cgiGetParameterValue (const char * *pcParam*)

Retrieves the value of a cgi parameter.

Parameters

in	*pcParam*	is the parameter/value string of a cgi request.

This function searches for the value part of a parameter/value pair. The syntax is "<Parm>=<Value>". The value will not be null-terminated, as it is a pointer within the whole parameter string. So it ends on "&" or a CRLF sequence.

Returns

Returns a pointer to the value of the parameter, or NULL if it does not exist.

{xe "cgiIsValidHexDigit:CGI handling functions"}{xe "CGI handling functions:cgiIsValidHexDigit"}static bool cgiIsValidHexDigit (const char *cDigit*)`[static]`

Determines whether a given character is a valid hexadecimal digit.

Parameters

in	*cDigit*	is the ASCII character code to be checked for validity.

This function checks the passed character to determime whether or not it is a valid hexadecimal digit.

Returns

Returns *true* if the passed character is a valid hex digit or *false* otherwise.

{xe "cgiSelectParameter:CGI handling functions"}{xe "CGI handling functions:cgiSelectParameter"}const char * cgiSelectParameter (const char * *pcParams*, uint32_t *ui32Num*)

Select a parameter/value pair out of a parameter string.

Parameters

pcParams	is the parameter/value string of a cgi request.
ui32Num	is the number of the parameter/value pair to be returned.

This function searches the parameter/value pair of number *ui32Num* .

Returns

pointer to start of parameter/value pair. NULL, if no value was found.

CGI functions

{tc \l 2 "CGI functions"}{xe "CGI functions"}Macros

#define xstr(a)   str(a)
#define str(a)   #a
#define CCHMAXCGIPOWERSTATE   21
#define CCHMAXCGITEMPERATURE   36
#define CCHMAXCGIDTSETTINGS   55
#define CCHMAXCGIRELAYSTATE   RELAY_COUNT*2
#define CCHMAXCGIGETSWITCHSET   60
#define CCHMAXCGIGETIMGDATA   32
#define CCHMAXCGIGETSYSSET   57
#define CCHMAXCGIPWRLOGSET   8
#define CCHMAXCGISIZEFS   8
#define CCHMAXCGIFREEFS   8
#define CCHMAXCGIDATAFS   16
#define CCHMAXCGIIPSET   67
#define CCHMAXCGISMTPSET   169
#define CCHMAXCGITEMPSET   26
#define CCHMAXCGIWDSET   80
#define CCHMAXCGIGETENETMODE   2
#define CCHMAXCGIHTTPSET   8
#define CCHMAXCGISNMPSET   102
#define CCHMAXCGISLOGSET   19
#define CCHMAXCGIUPNPSET   10
#define CCHMAXCGIFTPDSET   25
#define CCHMAXCGIACLSET   90
#define CCHMAXCGIGETTFTPSET   21
#define CCHMAXCGICHECKACC   128
#define CCHMAXTEMP   12
#define CCHMAXFILENAME   64
#define CCHCRLF   (sizeof(pcCrLf)-1)
#define CCHCRLFCRLF   (sizeof(pcCrLfCrLf)-1)
#define GEN_FS_ENTRY_1(fktName,   fktNextEntry,   cgiName,   flags)   const tsFsDataFile file_ ## fktName = {&fktNextEntry, (const unsigned char *)#cgiName, (const uint8_t *)fktName, FS_EXEC | flags}
#define GEN_FS_ENTRY_N(fktName,   fktNextEntry,   cgiName,   flags)   const tsFsDataFile file_ ## fktName = {&file_ ## fktNextEntry, (const unsigned char *)#cgiName, (const uint8_t *)fktName, FS_EXEC | flags}
#define CCHMAXTEMPBUFFER   19
#define CCHMAXMACBUFFER   18
#define NUM_PANELS   (sizeof(pcPanelId)/sizeof(char *))
#define file_cgiFktFirst   file_cgiFktGetJsonData
#define FS_CGI_ROOT   file_cgiFktFirst

Typedefs

typedef enum _eIdPanel teIdPanel

Enumerations

enum _eIdPanel { idHeader, idMain, idHistory, idScheduler, idState, idSetSwitch, idSysSet, idDateTime, idPower, idLAN, idEmail, idWatchdog, idStandby, idWebserver, idSnmp, idSyslog, idUPnP, idFtp, idAcl, idFilesystem, idSaveRestore, idFwUpd, idFsUpd, idUser, idLogoff }

Functions

bool cgiFktGetGraphDataTS (ptsHttpState psHttpState)
bool cgiFktGetGraphData (ptsHttpState psHttpState)
bool cgiFktSendTestMail (ptsHttpState psHttpState)
bool cgiFktGetMailState (ptsHttpState psHttpState)
bool cgiFktTempReset (ptsHttpState psHttpState)
bool cgiFktGetTftpSettings (ptsHttpState psHttpState)
bool cgiFktSwUpdate (ptsHttpState psHttpState)
bool cgiFktMarkForInst (ptsHttpState psHttpState)
bool cgiFktReboot (ptsHttpState psHttpState)
bool cgiFktDeleteImage (ptsHttpState psHttpState)
bool cgiFktQueryImageData (ptsHttpState psHttpState)
bool cgiFktGetTransferState (ptsHttpState psHttpState)
bool cgiFktGetPowerState (ptsHttpState psHttpState)
bool cgiFktGetRelayState (ptsHttpState psHttpState)
bool cgiFktGetAllRelays (ptsHttpState psHttpState)
bool cgiFktToggleRelay (ptsHttpState psHttpState)
bool cgiFktSetRelay (ptsHttpState psHttpState)
bool cgiFktGetSwitchSettings (ptsHttpState psHttpState)
bool cgiFktGetSysSettings (ptsHttpState psHttpState)
bool cgiFktGetPwrLogSettings (ptsHttpState psHttpState)
bool cgiFktGetIPSettings (ptsHttpState psHttpState)
bool cgiFktGetSmtpSettings (ptsHttpState psHttpState)
bool cgiFktGetTempSettings (ptsHttpState psHttpState)
bool cgiFktGetWdSettings (ptsHttpState psHttpState)
bool cgiFktGetEnetMode (ptsHttpState psHttpState)
bool cgiFktGetHttpSettings (ptsHttpState psHttpState)
bool cgiFktGetSnmpSettings (ptsHttpState psHttpState)
bool cgiFktGetSlogSettings (ptsHttpState psHttpState)
bool cgiFktGetUPnPSettings (ptsHttpState psHttpState)
bool cgiFktGetFtpdSettings (ptsHttpState psHttpState)
bool cgiFktGetAclFSettings (ptsHttpState psHttpState)
bool cgiFktConfiguration (ptsHttpState psHttpState)
bool cgiFktIsFlashFs (ptsHttpState psHttpState)
bool cgiFktIsTemperature (ptsHttpState psHttpState)
bool cgiFktTemperatureGet (ptsHttpState psHttpState)
bool cgiFktDateTimeGet (ptsHttpState psHttpState)
bool cgiFktGetSizeFS (ptsHttpState psHttpState)
bool cgiFktGetFreeFS (ptsHttpState psHttpState)
bool cgiFktGetDataFS (ptsHttpState psHttpState)
bool cgiFktSetScheduler (ptsHttpState psHttpState)
bool cgiFktSetIPData (ptsHttpState psHttpState)
bool cgiFktSetSysFunction (ptsHttpState psHttpState)
bool cgiFktSetDateTime (ptsHttpState psHttpState)
bool cgiFktSetSwitchSettings (ptsHttpState psHttpState)
bool cgiFktSetPwrLogSettings (ptsHttpState psHttpState)
bool cgiFktSetFactory (ptsHttpState psHttpState)
bool cgiFktRestore (ptsHttpState psHttpState)
bool cgiFktGetRestRes (ptsHttpState psHttpState)
bool cgiFktLoadFw (ptsHttpState psHttpState)
bool cgiFktLoadFs (ptsHttpState psHttpState)
bool cgiFktSetSyslogSettings (ptsHttpState psHttpState)
bool cgiFktSetIPACLSettings (ptsHttpState psHttpState)
bool cgiFktSetWatchdog (ptsHttpState psHttpState)
bool cgiFktSetSnmpSettings (ptsHttpState psHttpState)
bool cgiFktSetUPnPSettings (ptsHttpState psHttpState)

bool cgiFktSetWebSrvSettings (ptsHttpState psHttpState)
bool cgiFktSetSmtpSettings (ptsHttpState psHttpState)
bool cgiFktSetTempSettings (ptsHttpState psHttpState)
bool cgiFktSetFtpdSettings (ptsHttpState psHttpState)
bool cgiFktSetFileSystem (ptsHttpState psHttpState)
bool cgiFktSetEnetMode (ptsHttpState psHttpState)
bool cgiFktSetTFTPServer (ptsHttpState psHttpState)
bool cgiFktSetUser (ptsHttpState psHttpState)
bool cgiFktGetAccessRights (ptsHttpState psHttpState)
bool cgiFktLogoff (ptsHttpState psHttpState)
bool cgiFktGetJsonData (ptsHttpState psHttpState)
static bool cgiFktFlashCheckAddress (uint32_t ui32Address, uint32_t ui32Range)
static bool cgiFktFlashWrite (uint32_t ui32Address, int32_t i32DataLen, char *pcData)
static void cgiFktCalcReplyLen (tsCgiReply *psCgiReply)
void cgiFktAbortLoad (ptsHttpState psHttpState)
static uint32_t cgiFktAddJsonPart (char **ppcBuffer, uint32_t *pui32LenBuffer, const char *pcFormat,...)
static int32_t cgiFktGetPanelId (const char *pcValPanel)
static void cgiFktAddJsonLevel (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t ui32Level)
static void cgiFktLevelOpen (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level)
static void cgiFktLevelClose (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, bool bIsLast)
static void cgiFktDomainOpen (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcDomain)
static void cgiFktDomainClose (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, bool bIsLast)
static void cgiFktArrayOpen (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcArray)
static void cgiFktArrayClose (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, bool bIsLast)
static void cgiFktAddJsonBool (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcVariable, bool bValue, bool bIsLast)
static void cgiFktAddJsonString (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcVariable, const char *pcValue, bool bIsLast)
static void cgiFktAddJsonInt (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcVariable, int32_t i32Value, bool bIsLast)
static void cgiFktAddJsonUInt (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcVariable, uint32_t ui32Value, bool bIsLast)
static void cgiFktAddJsonFloat (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcVariable, uint32_t ui32Value, bool bIsLast)
static void cgiFktAddJsonStringValue (char **ppcBuffer, uint32_t *pui32LenBuffer, uint32_t *pui32Level, const char *pcValue, bool bIsLast)
void cgiFktHeaderJsonData (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, teUserType eUserType, bool bIsLast)
void cgiFktJsonMain (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonHistory (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonScheduler (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonState (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonSetSwitch (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonSysSet (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonDateTime (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonPower (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonLAN (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonEmail (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonWatchdog (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonStandby (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)

void cgiFktJsonWebserver (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonSnmp (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonSyslog (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonUPnP (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonFtpd (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonAcl (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonFSys (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonUser (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonFwUpdate (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
void cgiFktJsonFsUpdate (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, bool bIsLast)
static void cgiFktJsonCreate (char **ppcBuffer, uint32_t *pui32Len, uint32_t *pui32Level, int32_t i32PanelId, teUserType eUserType, bool bIsLast)

Variables

const char g_pcErrorReply [] = "/error.ssi"
bool g_bIsFileReply = true
const char pcIndex [] = "/index.htm"
const char pcScheduler [] = "/timer.ssi"
const char pcSetUser [] = "/user.ssi"
const char pcSetStd [] = "/set_std.htz"
const char pcSetLAN [] = "/set_lan.htz"
const char pcSetExp [] = "/set_exp.ssi"
const char pcUpdate [] = "/update.ssi"
const char pcOn [] = "on"
const char pcOff [] = "off"
const char pcTrue [] = "1"
const char pcFalse [] = "0"
const char pcNoTemp [] = "-|-|-"
const char pcConfiguration [] = xstr(RELAY_COUNT)"|"xstr(POWM_COUNT)
const char *const ppcStateReply [3] = {"0", "1", "2"}
const char *const pcCtlUser [] = { "uname0", "uname1", "uname2" }
const char *const pcCtlPwd1 [] = { "upwd0", "upwd1", "upwd2" }
const char *const pcCtlPwd2 [] = { "upwd20", "upwd21", "upwd22" }
const char *const pcCtlType [] = { "utype0", "utype1", "utype2" }
const char pcCrLf [] = "\r\n"
const char pcCrLfCrLf [] = "\r\n\r\n"
bool g_bRestoreSuccess = true
void * g_pConnForLoad = NULL
static const char pcIdAclIp [] = "acl?"
static const char *const pcPanelId []
const char g_pcErrorReply []
bool g_bIsFileReply
const tsFsDataFile file_cgiFktFirst
const char pcOn []
const char pcOff []

Detailed Description

This module contains all CGI functions.

Note

parameter string in pHTTPState->pcCgiParm,
parameter length in pHTTPState->ui16LenCgiParm.

Macro Definition Documentation

{xe "CCHCRLF:CGI functions"}{xe "CGI functions:CCHCRLF"}#define CCHCRLF (sizeof(pcCrLf)-1)

Stringlength of pcCrLf, saving a call to strlen().

{xe "CCHCRLFCRLF:CGI functions"}{xe "CGI functions:CCHCRLFCRLF"}#define CCHCRLFCRLF (sizeof(pcCrLfCrLf)-1)

Stringlength of pcCrLfCrLf, saving a call to strlen().

{xe "CCHMAXCGIACLSET:CGI functions"}{xe "CGI functions:CCHMAXCGIACLSET"}#define CCHMAXCGIACLSET 90

Max. string length for /cgi/getAclSet incl. '\0'.

{xe "CCHMAXCGICHECKACC:CGI functions"}{xe "CGI functions:CCHMAXCGICHECKACC"}#define CCHMAXCGICHECKACC 128

Max. string length for /cgi/checkAcc incl. '\0'.

{xe "CCHMAXCGIDATAFS:CGI functions"}{xe "CGI functions:CCHMAXCGIDATAFS"}#define CCHMAXCGIDATAFS 16

Max. string length for /cgi/getDataFS incl. '\0'.

{xe "CCHMAXCGIDTSETTINGS:CGI functions"}{xe "CGI functions:CCHMAXCGIDTSETTINGS"}#define CCHMAXCGIDTSETTINGS 55

Max. string length for /cgi/getDateTime incl. '\0'.

{xe "CCHMAXCGIFREEFS:CGI functions"}{xe "CGI functions:CCHMAXCGIFREEFS"}#define CCHMAXCGIFREEFS 8

Max. string length for /cgi/getFreeFS incl. '\0'.

{xe "CCHMAXCGIFTPDSET:CGI functions"}{xe "CGI functions:CCHMAXCGIFTPDSET"}#define CCHMAXCGIFTPDSET 25

Max. string length for /cgi/getFtpdSet incl. '\0'.

{xe "CCHMAXCGIGETENETMODE:CGI functions"}{xe "CGI functions:CCHMAXCGIGETENETMODE"}#define CCHMAXCGIGETENETMODE 2

Max. string length for /cgi/getEnetMode incl. '\0'.

{xe "CCHMAXCGIGETIMGDATA:CGI functions"}{xe "CGI functions:CCHMAXCGIGETIMGDATA"}#define CCHMAXCGIGETIMGDATA 32

Max. string length for /cgi/getImgData incl. '\0'.

{xe "CCHMAXCGIGETSWITCHSET:CGI functions"}{xe "CGI functions:CCHMAXCGIGETSWITCHSET"}#define CCHMAXCGIGETSWITCHSET 60

Max. string length for /cgi/getSwitchSet incl. '\0'.

{xe "CCHMAXCGIGETSYSSET:CGI functions"}{xe "CGI functions:CCHMAXCGIGETSYSSET"}#define CCHMAXCGIGETSYSSET 57

Max. string length for /cgi/getSysSet incl. '\0'.

{xe "CCHMAXCGIGETTFTPSET:CGI functions"}{xe "CGI functions:CCHMAXCGIGETTFTPSET"}#define CCHMAXCGIGETTFTPSET 21

Max. string length for /cgi/getTftpSrv incl. '\0'.

{xe "CCHMAXCGIHTTPSET:CGI functions"}{xe "CGI functions:CCHMAXCGIHTTPSET"}#define CCHMAXCGIHTTPSET 8

Max. string length for /cgi/getHttpSet incl. '\0'.

{xe "CCHMAXCGIIPSET:CGI functions"}{xe "CGI functions:CCHMAXCGIIPSET"}#define CCHMAXCGIIPSET 67

Max. string length for /cgi/getIpSet incl. '\0'.

{xe "CCHMAXCGIPOWERSTATE:CGI functions"}{xe "CGI functions:CCHMAXCGIPOWERSTATE"}#define CCHMAXCGIPOWERSTATE 21

Max. string length for /cgi/getPower incl. '\0'.

{xe "CCHMAXCGIPWRLOGSET:CGI functions"}{xe "CGI functions:CCHMAXCGIPWRLOGSET"}#define CCHMAXCGIPWRLOGSET 8

Max. string length for /cgi/getPwLogSet incl. '\0'.

{xe "CCHMAXCGIRELAYSTATE:CGI functions"}{xe "CGI functions:CCHMAXCGIRELAYSTATE"}#define CCHMAXCGIRELAYSTATE RELAY_COUNT*2

Max. string length for /cgi/getRelays incl. '\0'.

{xe "CCHMAXCGISIZEFS:CGI functions"}{xe "CGI functions:CCHMAXCGISIZEFS"}#define CCHMAXCGISIZEFS 8

Max. string length for /cgi/getSizeFS incl. '\0'.

{xe "CCHMAXCGISLOGSET:CGI functions"}{xe "CGI functions:CCHMAXCGISLOGSET"}#define CCHMAXCGISLOGSET 19

Max. string length for /cgi/getSlogSet incl. '\0'.

{xe "CCHMAXCGISMTPSET:CGI functions"}{xe "CGI functions:CCHMAXCGISMTPSET"}#define CCHMAXCGISMTPSET 169

Max. string length for /cgi/getEmailSet incl. '\0'.

{xe "CCHMAXCGISNMPSET:CGI functions"}{xe "CGI functions:CCHMAXCGISNMPSET"}#define CCHMAXCGISNMPSET 102

Max. string length for /cgi/getSnmpSet incl. '\0'.

{xe "CCHMAXCGITEMPERATURE:CGI functions"}{xe "CGI functions:CCHMAXCGITEMPERATURE"}#define CCHMAXCGITEMPERATURE 36

Max. string length for /cgi/getTemp incl. '\0'.

{xe "CCHMAXCGITEMPSET:CGI functions"}{xe "CGI functions:CCHMAXCGITEMPSET"}#define CCHMAXCGITEMPSET 26

Max. string length for /cgi/getTempSet incl. '\0'.

{xe "CCHMAXCGIUPNPSET:CGI functions"}{xe "CGI functions:CCHMAXCGIUPNPSET"}#define CCHMAXCGIUPNPSET 10

Max. string length for /cgi/getUpnpSet incl. '\0'.

{xe "CCHMAXCGIWDSET:CGI functions"}{xe "CGI functions:CCHMAXCGIWDSET"}#define CCHMAXCGIWDSET 80

Max. string length for /cgi/getWdSet incl. '\0'.

{xe "CCHMAXFILENAME:CGI functions"}{xe "CGI functions:CCHMAXFILENAME"}#define CCHMAXFILENAME 64

Max. string length of a filename incl. '\0'.

{xe "CCHMAXMACBUFFER:CGI functions"}{xe "CGI functions:CCHMAXMACBUFFER"}#define CCHMAXMACBUFFER 18

Max. MAC string length in cgiFktJsonSysSet incl. '\0'.

{xe "CCHMAXTEMP:CGI functions"}{xe "CGI functions:CCHMAXTEMP"}#define CCHMAXTEMP 12

Max. string length of temperature incl. '\0'.

{xe "CCHMAXTEMPBUFFER:CGI functions"}{xe "CGI functions:CCHMAXTEMPBUFFER"}#define CCHMAXTEMPBUFFER 19

Max. temp string length in cgiFktGetJsonData incl. '\0'.

{xe "file_cgiFktFirst:CGI functions"}{xe "CGI functions:file_cgiFktFirst"}#define file_cgiFktFirst file_cgiFktGetJsonData

First entry of directory for cgi functions. See FS_CGI_ROOT.

{xe "FS_CGI_ROOT:CGI functions"}{xe "CGI functions:FS_CGI_ROOT"}#define FS_CGI_ROOT file_cgiFktFirst

Root of the linked list of directory entries of the flash file system. *FS_CGI_ROOT* includes the cgi-functions, whereas *FS_ROOT* (defined in io_fsdata_def.h) is the root of all data files only.

{xe "GEN_FS_ENTRY_1:CGI functions"}{xe "CGI functions:GEN_FS_ENTRY_1"}#define GEN_FS_ENTRY_1(fktName, fktNextEntry, cgiName, flags) const tsFsDataFile file_ ## fktName = {&fktNextEntry, (const unsigned char *)#cgiName, (const uint8_t *)fktName, FS_EXEC | flags}

Macro to define the first entry of the chained list of cgi handlers.

{xe "GEN_FS_ENTRY_N:CGI functions"}{xe "CGI functions:GEN_FS_ENTRY_N"}#define GEN_FS_ENTRY_N(fktName, fktNextEntry, cgiName, flags) const tsFsDataFile file_ ## fktName = {&file_ ## fktNextEntry, (const unsigned char *)#cgiName, (const uint8_t *)fktName, FS_EXEC | flags}

Macro to define an entry of the chained list of cgi handlers.

{xe "NUM_PANELS:CGI functions"}{xe "CGI functions:NUM_PANELS"}#define NUM_PANELS (sizeof(pcPanelId)/sizeof(char *))

Number of panels in pcPanelId.

{xe "str:CGI functions"}{xe "CGI functions:str"}#define str(a)   #a

Stringify argument.

{xe "xstr:CGI functions"}{xe "CGI functions:xstr"}#define xstr(a)   str(a)

Evaluate argument of macro for stringification.

Typedef Documentation

{xe "teIdPanel:CGI functions"}{xe "CGI functions:teIdPanel"}teIdPanel

Typedef of enum _eIdPanel.

Enumeration Type Documentation

{xe "_eIdPanel:CGI functions"}{xe "CGI functions:_eIdPanel"}enum _eIdPanel

Enumeration for panel ids.

Enumerator:

{xe "idHeader:CGI functions"}{xe "CGI functions:idHeader"}idHeader	Panel id of the webpage-header (cgiFktHeaderJsonData).
{xe "idMain:CGI functions"}{xe "CGI functions:idMain"}idMain	Panel id for the main panel (cgiFktJsonMain).
{xe "idHistory:CGI functions"}{xe "CGI functions:idHistory"}idHistory	Panel id for the history panel (cgiFktJsonHistory).
{xe "idScheduler:CGI functions"}{xe "CGI functions:idScheduler"}idScheduler	Panel id for the scheduler panel (cgiFktJsonScheduler).
{xe "idState:CGI functions"}{xe "CGI functions:idState"}idState	Panel id for the state panel (cgiFktJsonState).
{xe "idSetSwitch:CGI functions"}{xe "CGI functions:idSetSwitch"}idSetSwitch	Panel id for the switch panel (cgiFktJsonSetSwitch).

{xe "idSysSet:CGI functions"}{xe "CGI functions:idSysSet"}idSysSet	Panel id for the system settings panel (cgiFktJsonSysSet).
{xe "idDateTime:CGI functions"}{xe "CGI functions:idDateTime"}idDateTime	Panel id for the date/time panel (cgiFktJsonDateTime).
{xe "idPower:CGI functions"}{xe "CGI functions:idPower"}idPower	Panel id for the power logging panel (cgiFktJsonPower).
{xe "idLAN:CGI functions"}{xe "CGI functions:idLAN"}idLAN	Panel id for the LAN settings panel (cgiFktJsonLAN).
{xe "idEmail:CGI functions"}{xe "CGI functions:idEmail"}idEmail	Panel id for the e-mail panel (cgiFktJsonEmail).
{xe "idWatchdog:CGI functions"}{xe "CGI functions:idWatchdog"}idWatchdog	Panel id for the watchdog panel (cgiFktJsonWatchdog).
{xe "idStandby:CGI functions"}{xe "CGI functions:idStandby"}idStandby	Panel id for the standby panel (cgiFktJsonStandby).
{xe "idWebserver:CGI functions"}{xe "CGI functions:idWebserver"}idWebserver	Panel id for the webserver settings panel (cgiFktJsonWebserver).
{xe "idSnmp:CGI functions"}{xe "CGI functions:idSnmp"}idSnmp	Panel id for the SNMP settings panel (cgiFktJsonSnmp).
{xe "idSyslog:CGI functions"}{xe "CGI functions:idSyslog"}idSyslog	Panel id for the syslog settings panel (cgiFktJsonSyslog).
{xe "idUPnP:CGI functions"}{xe	Panel id for the UPnP settings panel (cgiFktJsonUPnP.

"CGI functions:idUPnP"}idUPnP	
{xe "idFtp:CGI functions"}{xe "CGI functions:idFtp"}idFtp	Panel id for the ftp daemon settings panel (cgiFktJsonFtpd).
{xe "idAcl:CGI functions"}{xe "CGI functions:idAcl"}idAcl	Panel id for the ACL settings panel (cgiFktJsonAcl).
{xe "idFilesystem:CGI functions"}{xe "CGI functions:idFilesystem"}idFilesystem	Panel id for the file system panel (cgiFktJsonFSys).
{xe "idSaveRestore:CGI functions"}{xe "CGI functions:idSaveRestore"}idSaveRestore	Panel id for the save/restore settings panel (not used, no data).
{xe "idFwUpd:CGI functions"}{xe "CGI functions:idFwUpd"}idFwUpd	Panel id for the firmware update panel (cgiFktJsonFwUpdate).
{xe "idFsUpd:CGI functions"}{xe "CGI functions:idFsUpd"}idFsUpd	Panel id for the file system update panel (cgiFktJsonFsUpdate).
{xe "idUser:CGI functions"}{xe "CGI functions:idUser"}idUser	Panel id for the user settings panel (cgiFktJsonUser).
{xe "idLogoff:CGI functions"}{xe "CGI functions:idLogoff"}idLogoff	Panel id for the logoff panel (not used, no data).

Function Documentation

{xe "cgiFktAbortLoad:CGI functions"}{xe "CGI functions:cgiFktAbortLoad"}void cgiFktAbortLoad (ptsHttpState *psHttpState*)

Inform that the connection that transfers a file will be closed.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This function is called to inform the system, that a connection will be closed. The function checks, if currently a cgiFktLoadFs or cgiFktLoadFw is running, using the connection to be closed. In this case, there will be no additional call to that function and if an update is not yet completed it will be aborted. The used global variable g_pConnForLoad is used to ensure, that there will be no parallel updates to the file system started.

Returns

None.

{xe "cgiFktAddJsonBool:CGI functions"}{xe "CGI functions:cgiFktAddJsonBool"}static void cgiFktAddJsonBool (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcVariable*, bool *bValue*, bool *bIsLast*) `[static]`

Add a part to Json string: bool value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcVariable	is a string containing the variable name.
bValue	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: a bool value.

Returns

None.

{xe "cgiFktAddJsonFloat:CGI functions"}{xe "CGI functions:cgiFktAddJsonFloat"}static void cgiFktAddJsonFloat (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcVariable*, uint32_t *ui32Value*, bool *bIsLast*) `[static]`

Add a part to Json string: float value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcVariable	is a string containing the variable name.
ui32Value	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: a fix point float value.

Returns

None.

{xe "cgiFktAddJsonInt:CGI functions"}{xe "CGI functions:cgiFktAddJsonInt"}static void cgiFktAddJsonInt (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcVariable*, int32_t *i32Value*, bool *bIsLast*)`[static]`

Add a part to Json string: integer value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcVariable	is a string containing the variable name.
i32Value	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: an integer value.

Returns

None.

{xe "cgiFktAddJsonLevel:CGI functions"}{xe "CGI functions:cgiFktAddJsonLevel"}static void cgiFktAddJsonLevel (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t *ui32Level*)`[static]`

Add a part to Json string: indentation based on hierarchy level.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
ui32Level	is the JSON hierarchy.

This function creates a part of the Json string: the indentation based on the current hierarchy level (1 space per level).

Returns

None.

{xe "cgiFktAddJsonPart:CGI functions"}{xe "CGI functions:cgiFktAddJsonPart"}static uint32_t cgiFktAddJsonPart (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, const char * *pcFormat*, *...*)`[static]`

Add a part to Json string based on a format string.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pcFormat	is the format string to be used.

This function creates a part of the Json string based on a format string. The variable number of additional arguments has to fit to the format.

Returns

None.

{xe "cgiFktAddJsonString:CGI functions"}{xe "CGI functions:cgiFktAddJsonString"}static void cgiFktAddJsonString (char ** *ppcBuffer*,

uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcVariable*, const char * *pcValue*, bool *bIsLast*)`[static]`

Add a part to Json string: string value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcVariable	is a string containing the variable name.
pcValue	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: a string value.

Returns

None.

{xe "cgiFktAddJsonStringValue:CGI functions"}{xe "CGI functions:cgiFktAddJsonStringValue"}static void cgiFktAddJsonStringValue (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcValue*, bool *bIsLast*)`[static]`

Add a part to Json string: string value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcValue	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: a string value.

Returns

None.

{xe "cgiFktAddJsonUInt:CGI functions"}{xe "CGI functions:cgiFktAddJsonUInt"}static void cgiFktAddJsonUInt (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcVariable*, uint32_t *ui32Value*, bool *bIsLast*)`[static]`

Add a part to Json string: unsigned integer value.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcVariable	is a string containing the variable name.
ui32Value	is the value to be added.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: an unsigned integer value.

Returns

None.

{xe "cgiFktArrayClose:CGI functions"}{xe "CGI functions:cgiFktArrayClose"}static void cgiFktArrayClose (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, bool *bIsLast*) `[static]`

Add a part to Json string: closing bracket for array.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: the closing bracket for an array.

Returns

None.

{xe "cgiFktArrayOpen:CGI functions"}{xe "CGI functions:cgiFktArrayOpen"}static void cgiFktArrayOpen (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcArray*) `[static]`

Add a part to Json string: array.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcArray	is the name of the array.

This function creates a part of the Json string: the opening part of an array.

Returns

None.

{xe "cgiFktCalcReplyLen:CGI functions"}{xe "CGI functions:cgiFktCalcReplyLen"}static void cgiFktCalcReplyLen (tsCgiReply * *psCgiReply*) `[static]`

Calculate reply string length.

Parameters

psCgiReply	is a pointer to the reply structure contained in the instance variable of the HTTP connection (psHttpState).

This function calculates the length of the reply string of a cgi function. If the pointer to the string is NULL, 0 is returned. Otherwise strlen() is used to determine the string length.

Returns

None.

{xe "cgiFktConfiguration:CGI functions"}{xe "CGI functions:cgiFktConfiguration"}bool cgiFktConfiguration (ptsHttpState *psHttpState*)

CGI function `/cgi/configuration` : Get number of relays and power measurement devices.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get number of relays and power measurement devices. Request-type: GET.

Syntax:

```
/cgi/configuration
```

CGI-parameters: None.

Reply: `<relays>|<powmdev>]`.

with:

`relays` as the number of relays,
`powmdev` as the number of power measurement devices.

Returns

always true.

{xe "cgiFktDateTimeGet:CGI functions"}{xe "CGI functions:cgiFktDateTimeGet"}bool cgiFktDateTimeGet (ptsHttpState *psHttpState*)

CGI function `/cgi/getDateTime` : Request date/time settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Retrieve the current date/time settings of the system. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonDateTime

Syntax:

```
/cgi/getDateTime
```

CGI-parameters: None.

Reply:

`<TZone>|<isNTP>|<ntpServer>|<hh>|<mm>|<DD>|<MM>|<YYYY>|<valid>`,

with:

`TZone` is the time zone,
`isNTP:` bool [0|1],
`ntpServer` is the ntp servers url,
`hh, mm` is the time,
`DD, MM, YYYY` is the date,
`valid` is a boolean value, '1' if time is valid.

Returns

always true.

{xe "cgiFktDeleteImage:CGI functions"}{xe "CGI functions:cgiFktDeleteImage"}bool cgiFktDeleteImage (ptsHttpState *psHttpState*)

CGI function `/cgi/delInst` : deletes a downloaded firmware image from serial data flash.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Deletes a downloaded firmware image from serial data flash. Request-type: GET.

Syntax:

`/cgi/delInst`

CGI-parameters: None.

Reply:

There is no reply value.

Returns

always true.

{xe "cgiFktDomainClose:CGI functions"}{xe "CGI functions:cgiFktDomainClose"}static void cgiFktDomainClose (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, bool *bIsLast*) `[static]`

Add a part to Json string: closing bracket of a domain.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: the closing bracket of a domain.

Returns

None.

{xe "cgiFktDomainOpen:CGI functions"}{xe "CGI functions:cgiFktDomainOpen"}static void cgiFktDomainOpen (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, const char * *pcDomain*) `[static]`

Add a part to Json string: domain.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
pcDomain	is the name of the domain.

This function creates a part of the Json string: the opening part of a domain.

Returns

None.

{xe "cgiFktFlashCheckAddress:CGI functions"}{xe "CGI functions:cgiFktFlashCheckAddress"}static bool cgiFktFlashCheckAddress (uint32_t *ui32Address*, uint32_t *ui32Range*) `[static]`

Checks, if an address range is behind the application image and within the internal flash.

Parameters

ui32Address	is the start address to be checked.
ui32Range	is the address range to be checked.

To avoid erroneous overwriting of the application area, this function checks, if an address range is within the free area of the internal. flash memory.

Returns

true, if the address range is within the free area of the flash.

{xe "cgiFktFlashWrite:CGI functions"}{xe "CGI functions:cgiFktFlashWrite"}static bool cgiFktFlashWrite (uint32_t *ui32Address*, int32_t *i32DataLen*, char * *pcData*) `[static]`

Writes a buffer into the internal flash memory.

Parameters

ui32Address	is the start address to be checked.
i32DataLen	is the size of the buffer.
pcData	is the start address of the source buffer.

This function writes a buffer into the internal flash memory. This function ensures a correct write operation even if the start address or buffer size is not a multiple of 4.

Returns

true, if the write operation was successful.

{xe "cgiFktGetAccessRights:CGI functions"}{xe "CGI functions:cgiFktGetAccessRights"}bool cgiFktGetAccessRights (ptsHttpState *psHttpState*)

CGI function `/cgi/checkAcc` : Return access rights to files of current user.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return the access rights to a list of files of the current user. Request-type: GET.

Syntax:

`/cgi/checkAcc?files=<file1>,<file2>,...,<fileN>`

CGI-parameters:

`files=<file1>` ,<file2>,...,<fileN> is the list of N files.

Reply: `<a1>|<a2>|...|<aN>|` .

with:

`a1` as the access right to file1: 1 is granted, 0 is denied.
`aN` as the access right to fileN: 1 is granted, 0 is denied.

Returns

always true.

{xe "cgiFktGetAclFSettings:CGI functions"}{xe "CGI functions:cgiFktGetAclFSettings"}bool cgiFktGetAclFSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getAclSet` : Get ACL filter settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get ACL filter settings. Request-type: GET.

[Deprecated](#):

Replacement see [cgiFktJsonAcl](#)

Syntax:

`/cgi/getAclSet`

CGI-parameters: None.

Reply:

`<isACL>|<aclF1>|<aclF2>|<aclF3>|<aclF4>|<aclF5>|<aclF6>|<aclF7>|<aclF8>.`

with:

`isACL` being '1', if the the ACL filter is activated,
`aclF1` ... `aclF8` being the ip address of the device being allowed for access.

Returns

always true.

{xe "cgiFktGetAllRelays:CGI functions"}{xe "CGI functions:cgiFktGetAllRelays"}bool cgiFktGetAllRelays (ptsHttpState *psHttpState*)

CGI function `"/cgi/getRelays"` : Return state of all relays.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return state of a relay. Request-type: GET.

[Deprecated](#):

Replacement see [cgiFktJsonMain](#)

Syntax:

`/cgi/getRelays`

CGI-parameters: None.

Reply:

`[['0'|'1']'|']*` for the on/off state of the relays.

Returns

always true.

{xe "cgiFktGetDataFS:CGI functions"}{xe "CGI functions:cgiFktGetDataFS"}bool cgiFktGetDataFS (ptsHttpState *psHttpState*)

CGI function `/cgi/getDataFS` : Return information about the file system in the external flash memory.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return power state. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFSys

Syntax:

`/cgi/getDataFS`

CGI-parameters: None.

Reply: `<SizeFS>|<FreeFS>` ,

with:

`SizeFS` as the size of the file system,
`FreeFS` as the size of the free memory in the file system.

Returns

always true.

{xe "cgiFktGetEnetMode:CGI functions"}{xe "CGI functions:cgiFktGetEnetMode"}bool cgiFktGetEnetMode (ptsHttpState *psHttpState*)

CGI function `/cgi/getEnetMode` : Get standby settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get standby settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonStandby

Syntax:

`/cgi/getEnetMode`

CGI-parameters: None.

Reply: `<stbyMode>` .

with:

`stbyMode` being the standby mode, with

- *0* : uses 100Base-TX or 10Base-T, whichever is the fastest supported by the partner device,
- *1* : uses 10Base-T,
- *2* : will start with 100Base-TX and switch back to 10Base-T when in sleep mode.

Returns

always true.

{xe "cgiFktGetFreeFS:CGI functions"}{xe "CGI functions:cgiFktGetFreeFS"}bool cgiFktGetFreeFS (ptsHttpState *psHttpState*)

CGI function `/cgi/getFreeFS` : Get free available space of the file system in the external flash memory.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Retrieve the free memory in the file system. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFSys

Syntax:

`/cgi/getFreeFS`

CGI-parameters: None.

Reply: `<FreeFS>` ,

with:

`FreeFS` as the free flash memory in Bytes.

Returns

always true.

{xe "cgiFktGetFtpdSettings:CGI functions"}{xe "CGI functions:cgiFktGetFtpdSettings"}bool cgiFktGetFtpdSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getFtpdSet` : Get ftp daemon settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get ftp daemon settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFtpd

Syntax:

`/cgi/getFtpdSet`

CGI-parameters: None.

Reply: `<isFtp>|<isFtpActive>|<ftpUser>|<ftpPasswd>|<ftpWrite>`
.

with:

`isFtp` being '1', if the ftp daemon is present,
`isFtpActive` being '1', if the ftp daemon is active,
`ftpUser` being the ftp username,
`ftpPasswd` being the ftp password,
`ftpWrite` being '0', if ftp is in readonly mode, '1' for read/write.

Returns

always true.

{xe "cgiFktGetGraphData:CGI functions"}{xe "CGI functions:cgiFktGetGraphData"}bool cgiFktGetGraphData (ptsHttpState *psHttpState*)

CGI function `/cgi/graph` : Return the logged power values for graphical presentation.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return the logged power values for graphical presentation by page index. Request-type: GET.

Syntax:

`/cgi/graph?id=<num>&type=[0|1]`

CGI-parameters:

`id=<num>` is an integer value containing the 0-based index of the measurement page (each packet has 508 values).
`type=` [0|1] is '0' for 10s values and '1' for 24h values.

Reply: A binary field of data containing the requested measurement values.

Returns

always true.

{xe "cgiFktGetGraphDataTS:CGI functions"}{xe "CGI functions:cgiFktGetGraphDataTS"}bool cgiFktGetGraphDataTS (ptsHttpState *psHttpState*)

CGI function `/cgi/graphTs` : Return the logged power values for graphical presentation.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return the logged power values for graphical presentation by time stamp. Request-type: GET.

Syntax:

`/cgi/graphTs?id=<timestamp>&type=[0|1]`

CGI-parameters:

`id=<timestamp>` is the time stamp of the requested measurement page (each packet has 508 values).
`type=` [0|1] is '0' for 10s values and '1' for 24h values.

Reply: A binary field of data containing the requested measurement values.

Returns

always true.

{xe "cgiFktGetHttpSettings:CGI functions"}{xe "CGI functions:cgiFktGetHttpSettings"}bool cgiFktGetHttpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getHttpSet` : Get web server settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get web server settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonWebserver

Syntax:

`/cgi/getHttpSet`

CGI-parameters: None.

Reply: `<toState>|<toVal>.`

with:

`toState` being the state of timeout setting ('0' = infinite, '1' = active).

`toVal` as the value for the inactivity timeout in seconds . This is the time, after which a user has to login again after inactivity.

Returns

always true.

{xe "cgiFktGetIPSettings:CGI functions"}{xe "CGI functions:cgiFktGetIPSettings"}bool cgiFktGetIPSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getIpSet` : Get ip settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get ip settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonLAN

Syntax:

```
/cgi/getIpSet
```

CGI-parameters: None.

Reply:

```
<devName>|<ipMode>|<ipAddr>|<netMask>|<gwAddr>|<dnsServer>.
```

with:

`devName` as the device name,
`ipMode` as the ip mode ('0': static ip address; '1': use dhcp; '2': use autoip),
`ipAddr` as the ip address,
`netMask` as the network mask,
`gwAddr` as the gateway address,
`dnsServer` as the ip address of an optional dns server.

Returns

always true.

{xe "cgiFktGetJsonData:CGI functions"}{xe "CGI functions:cgiFktGetJsonData"}bool cgiFktGetJsonData (ptsHttpState *psHttpState*)

CGI function `/cgi/getJsonData` : Get data of a specific panel in Json format.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Retrieves the data for a specific panel in Json format. Request type: GET.

Syntax:

```
/cgi/getJsonData?panel=<panel>
```

CGI-parameters:

`panel` is one of bsMain, bsState, bsSetSwitch, bsSysSet, bsDateTime, bsPower, bsLAN, bsEmail, bsSaveRestore, bsWatchdog, bsStandby, bsWebserver, bsFilesystem, bsSnmp, bsSyslog, bsUPnP, bsFtp, bsAcl, bsFwUpd, bsFsUpd, bsUser, bsLogoff.

Reply:

A set of variables in Json format is returned.

Returns

always true.

{xe "cgiFktGetMailState:CGI functions"}{xe "CGI functions:cgiFktGetMailState"}bool cgiFktGetMailState (ptsHttpState *psHttpState*)

CGI function `/cgi/getMailSt` : requests the current state of the e-mail sending process.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Requests the current state of the e-mail sending process. Request-type: GET.

Syntax:

`/cgi/getMailSt`

CGI-parameters: None.

Reply: `['0'|'1'|'2'],`

with:

`'0'` : idle,
`'1'` : ok,
`'2'` : nok.

Returns

always true.

{xe "cgiFktGetPanelId:CGI functions"}{xe "CGI functions:cgiFktGetPanelId"}static int32_t cgiFktGetPanelId (const char * *pcValPanel*)`[static]`

Get panel id.

Parameters

pcValPanel	is the name of the panel.

This function converts a panel name into its id.

Returns

panel id.

{xe "cgiFktGetPowerState:CGI functions"}{xe "CGI functions:cgiFktGetPowerState"}bool cgiFktGetPowerState (ptsHttpState *psHttpState*)

CGI function `/cgi/getPower` : Return power state.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return power state. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonMain

Syntax:

`/cgi/getPower?Pow=<index>`

CGI-parameters:

`Pow=<index>` is the 0-based index of the power measurement unit (i.e. 0 for relay 0).

Reply: `<power_r>|<power_a>|<energy>`.

with:

`power_r` as relative power value.
`power_a` as absolute power value.
`energy` as energy value.

Returns

always true.

{xe "cgiFktGetPwrLogSettings:CGI functions"}{xe "CGI functions:cgiFktGetPwrLogSettings"}bool cgiFktGetPwrLogSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getPwLogSet` : Get power logging settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get the power logging settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonPower

Syntax:

`/cgi/getPwLogSet`

CGI-parameters: None.

Reply: `<pwLog10s>|<pwLogMail10s>|<pwLog24h>|<pwLogMail24h>`.

with:

`pwLog10s` is '1', if 10s logging is enabled.
`pwLogMail10s` is '1', if 10s values are sent by email.
`pwLog24h` is '1', if 24h logging is enabled.
`pwLogMail24h` is '1', if 24h values are sent by email.

Returns

always true.

{xe "cgiFktGetRelayState:CGI functions"}{xe "CGI functions:cgiFktGetRelayState"}bool cgiFktGetRelayState (ptsHttpState *psHttpState*)

CGI function "`/cgi/relaySt`" : Return state of relay.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return state of a relay. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonMain

Syntax:

`/cgi/relaySt?Rel=<index>`

CGI-parameters:

`Rel=<index>` is the 0-based index of the relay.

Reply:

`["on"|"off"]` for the on/off relay state.

Returns

always true.

{xe "cgiFktGetRestRes:CGI functions"}{xe "CGI functions:cgiFktGetRestRes"}bool cgiFktGetRestRes (ptsHttpState *psHttpState*)

CGI function `/cgi/getRestRes` : Return success state of last restore operation (`/cgi/restore`).

Parameters

psHttpState	is the instance variable of the HTTP connection.

Return the success of the last restore operation. Request-type: GET.

Syntax:

`/cgi/getRestRes`

CGI-parameters: None.

Reply: `["0"|"1"]`

0 if the settings file was invalid, 1 if it was valid.

Returns

always true.

{xe "cgiFktGetSizeFS:CGI functions"}{xe "CGI functions:cgiFktGetSizeFS"}bool cgiFktGetSizeFS (ptsHttpState *psHttpState*)

CGI function `/cgi/getSizeFS` : Get the size of the file system in the external flash memory.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Retrieve the size of the external flash memory. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFSys

Syntax:

`/cgi/getSizeFS`

CGI-parameters: None.

Reply: `<SizeFS>` ,

with:

`SizeFS` as the size of the file system in Bytes.

Returns

always true.

{xe "cgiFktGetSlogSettings:CGI functions"}{xe "CGI functions:cgiFktGetSlogSettings"}bool cgiFktGetSlogSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getSlogSet` : Get syslog settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get syslog settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonSyslog

Syntax:

`/cgi/getSlogSet`

CGI-parameters: None.

Reply: `<isSlog>|<slogIP>|<slogPort>.`

with:

`isSlog` being '1', if syslog is activated,
`slogIP` being the syslog destination ip address,
`slogPort` being the syslog port.

Returns

always true.

{xe "cgiFktGetSmtpSettings:CGI functions"}{xe "CGI functions:cgiFktGetSmtpSettings"}bool cgiFktGetSmtpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getEmailSet` : Get smtp settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get smtp settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonEmail

Syntax:

`/cgi/getEmailSet`

CGI-parameters: None.

Reply:

`<isSMTP>|<smtpServer>|<smtpPort>|<smtpAccnt>|<smtpPwd>|<smtpFrom>|<smtpTo>.`

with:

`isSMTP` being '1', if smtp is active, '0' if otherwise,
`smtpServer` as the name of the smtp server,
`smtpPort` as the smtp port,
`smtpAccnt` as the smtp account name,
`smtpPwd` as the smtp password,
`smtpFrom` as the smtp *from* entry,
`smtpTo` as the smtp *to* entry.

Returns

always true.

{xe "cgiFktGetSnmpSettings:CGI functions"}{xe "CGI functions:cgiFktGetSnmpSettings"}bool cgiFktGetSnmpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getSnmpSet` : Get snmp settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get snmp settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonSnmp

Syntax:

`/cgi/getSnmpSet`

CGI-parameters: None.

Reply:

`<snmpOn>|<snmpTrap>|<snmpTrapIP>|<snmpContact>|<snmpName>|<snmpLocation>|<snmpCommunity>|<snmpWrCommunity>.`

with:

`snmpOn` as being '1', if snmp is activated,
`snmpTrap` as being '2', if snmp traps are enabled,
`snmpTrapIP` being the trap destination ip address,
`snmpContact` being the snmp contact string,
`snmpName` being the snmp name string,
`snmpLocation` being the snmp location string,
`snmpCommunity` being the snmp community string.
`snmpWrCommunity` being the snmp write community string.

Returns

always true.

{xe "cgiFktGetSwitchSettings:CGI functions"}{xe "CGI functions:cgiFktGetSwitchSettings"}bool cgiFktGetSwitchSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getSwitchSet` : Get switching settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get the switching settings: setting after power on, on and off delay. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonSetSwitch

Syntax:

`/cgi/getSwitchSet`

CGI-parameters: None.

Reply:

`<action_0>|<onDelay_0>|<offDelay_0>|<action_1>|<onDelay_1>|<offDelay_1>.`

with:

`action_<index>` is the action after power on, with initial state:

- '0' : off,
- '1' : on,
- '2' : last mode.

`onDelay_<index>` is the on delay in seconds after power on.

`offDelay_<index>` is the time delay to automatically switch on after a switching off.

With `index` being the index of the relay.

Returns

always true.

{xe "cgiFktGetSysSettings:CGI functions"}{xe "CGI functions:cgiFktGetSysSettings"}bool cgiFktGetSysSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getSysSet` : Get system settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get the system settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonSysSet

Syntax:

`/cgi/getSysSet`

CGI-parameters: None.

Reply:

`<ms>|<msT>|<msP>|<stateMS>|<thresMSOn>|<thresMSOff>|<stateWOL>|<delayWOL>|<macWOL>|<delayMSOn>|<delayMSOff>.`

with:

`ms` is '1', if master switches off, if power is a defined period of time below a threshold, '0' if otherwise.

`msT` is the number of minutes the power has to remain below a defined power threshold before the master switches off. The value has to be between 1 and 9 minutes.

`msP` is the power that has to be undershot to switch off the master. This value has to be between 5 and 99.

`stateMS` is '1', if master/slave-function is on, '0' if otherwise.

`thresMSOn` is the on-threshold for the slave in 100mW. The value has to be above 20 (2W) and larger than thresMSOff.

`thresMSOff` is the off-threshold for the slave in 100mW. The value has to be above 10 (1W).

`stateWOL` is '1', if a WOL package will be sent after master is on, '0' if otherwise.

`delayWOL` is the delay for WOL in ms.

`macWOL` is the MAC address to be used in the WOL package in the form `aa.aa.aa.aa.aa.aa` .

`delayMSOn` is the delay in seconds before the slave is switched on.

`delayMSOff` is the delay in seconds before the slave is switched off.

Returns

always true.

{xe "cgiFktGetTempSettings:CGI functions"}{xe "CGI functions:cgiFktGetTempSettings"}bool cgiFktGetTempSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getTempSet` : Get temperature measurement settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get temperature measurement settings. Request-type: GET.

Syntax:

`/cgi/getTempSet`

CGI-parameters: None.

Reply:
`<isTemp>|<isTActive>|<tempIntf>|<isMail>|<thresMail>|<tempActive>|<tempAction>|<thresRel0>|<thresRel1>`.

with:

`isTemp` being '1', if temperature measurement hardware is present,
`isTActive` being '1', if temperature measurement is activated,
`tempIntf` as the temperature sensor type: '0': LM75B(I2C); '1': DS18B20(1-wire),
`isMail` being '1', if an email has to be sent on over temperature,
`thresMail` as the threshold temperature for email sending,
`tempActive` as a bitfield for relays; '1': switch relay based on temperature,
`tempAction` as a bitfield for relays; '0': switch off, '1': switch on,
`thresRel0` as the threshold temperature for relay 0,
`thresRel1` as the threshold temperature for relay 1.

Returns

always true.

{xe "cgiFktGetTftpSettings:CGI functions"}{xe "CGI functions:cgiFktGetTftpSettings"}bool cgiFktGetTftpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getTftpSrv` : returns the name of the TFTP server.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Returns the name of the TFTP server for software download as a single string. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFwUpdate

Syntax:

`/cgi/getTftpSrv`

CGI-parameters: None.

Reply: `<servername>`.

Returns

always true.

{xe "cgiFktGetTransferState:CGI functions"}{xe "CGI functions:cgiFktGetTransferState"}bool cgiFktGetTransferState (ptsHttpState *psHttpState*)

CGI function `/cgi/isTransfer` : queries the state of a firmware transfer.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Queries the state of the transfer of a firmware file. Request-type: GET.

Syntax:

`/cgi/isTransfer`

CGI-parameters: None.

Reply: `["0"|"1"]`

0, if there is no active transfer, 1 if there is an active transfer.

Returns

always true.

{xe "cgiFktGetUPnPSettings:CGI functions"}{xe "CGI functions:cgiFktGetUPnPSettings"}bool cgiFktGetUPnPSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getUpnpSet` : Get UPnP settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get UPnP settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonUPnP

Syntax:

`/cgi/getUpnpSet`

CGI-parameters: None.

Reply: `<isUPnP>|<upnpIntvl>.`

with:

`isUPnP` being '1', if UPnP is activated,
`upnpIntvl` being the interval for UPnP advertisements in seconds.

Returns

always true.

{xe "cgiFktGetWdSettings:CGI functions"}{xe "CGI functions:cgiFktGetWdSettings"}bool cgiFktGetWdSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/getWdSet` : Get watchdog settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Get watchdog settings. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonWatchdog

Syntax:

`/cgi/getWdSet`

CGI-parameters: None.

Reply:

`<wdActive[index]>|<wdProto[index]>|<wdIP[index]>|<wdPort[index]>|<wdIntvl[index]>|<wdRetry[index]>|<wdAction[index]>|<wdWait[index]>|<wdDelay[index]>.`

with:

`wdActive[index]` being '1', if watchdog is active,
`wdProto[index]` being '0' for icmp protocol, '1' for tcp protocol,
`wdIP[index]` as the ip address to be supervised,
`wdPort[index]` as the port to be supervised,
`wdIntvl[index]` as the interval in seconds between pings,
`wdRetry[index]` as the number of retries,
`wdAction[index]` as the watchdog action, being '0' if the relay has to be switched off, '1' if it has to be switch off and after a delay (wdDelay) switched on again,
`wdWait[index]` being '0', if watchdog has to be started immediately, '1' if it will wait for the first response before starting,
`wdDelay[index]` as the delay for a reset in seconds.

With `index` being the index of the relay.

Returns

always true.

{xe "cgiFktHeaderJsonData:CGI functions"}{xe "CGI functions:cgiFktHeaderJsonData"}void cgiFktHeaderJsonData (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, teUserType *eUserType*, bool *bIsLast*)

Create Json string for the page header.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
eUserType	is the currently active user type.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the page header (idHeader). Json syntax:

```
"header":
{
 "devId": string,                        (Name of device)
 "dateTime": "DD.MM.YYYY - hh:mm",       (Date/time of device)
 "userType": 1..3                        (1: teUserType.UserGuest, 2:
teUserType.UserStandard, 3: teUserType.UserAdmin)
}
```

Returns

None.

{xe "cgiFktIsFlashFs:CGI functions"}{xe "CGI functions:cgiFktIsFlashFs"}bool cgiFktIsFlashFs (ptsHttpState *psHttpState*)

CGI function `/cgi/isFAT` : Check, if flash file system is present.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Check, if the hardware is prepared for flash file system. Request-type: GET.

Syntax:

```
/cgi/isFlashFs
```

CGI-parameters: None.

Reply: `['0'|'1'],`

a boolean value, which is '1', if flash file system is present.

Returns

always true.

{xe "cgiFktIsTemperature:CGI functions"}{xe "CGI functions:cgiFktIsTemperature"}bool cgiFktIsTemperature (ptsHttpState *psHttpState*)

CGI function `/cgi/isTempHw` : Check, if temperature measurement is present.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Check, if the hardware is prepared for temperature measurement. Request-type:GET.

Syntax:

```
/cgi/isTempHw
```

CGI-parameters: None.

Reply: `['0'|'1'],`

a boolean value, which is '1', if temperature measurement is working.

Returns

always true.

{xe "cgiFktJsonAcl:CGI functions"}{xe "CGI functions:cgiFktJsonAcl"}void cgiFktJsonAcl (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the ACL panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the ACL panel (idAcl). Json syntax:

```
"bsAcl":
```

```
{
 "isAcl": [true|false],                 (ACL list activated)
 "acl1": [true|false],                  (ACL entry: ip address)
 ...                                    (More entries if available)
}
```

Returns

None.

{xe "cgiFktJsonCreate:CGI functions"}{xe "CGI functions:cgiFktJsonCreate"}static void cgiFktJsonCreate (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, int32_t *i32PanelId*, teUserType *eUserType*, bool *bIsLast*) `[static]`

Create Json string for a requested panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
i32PanelId	is the id of the requested panel.
eUserType	is the currently active user type.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the requested panel. On entry, pui32Len contains the buffer length, on exit the remaining bytes in the buffer. The function stops, if the buffer is full. A termination of the string is not guaranteed. To just calculate the needed buffer size, this function can be called with a NULL buffer.

Returns

None.

{xe "cgiFktJsonDateTime:CGI functions"}{xe "CGI functions:cgiFktJsonDateTime"}void cgiFktJsonDateTime (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the date time/settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the date/time settings panel (idDateTime). Json syntax:

```
"bsDateTime":
{
 "tzone": string,                       (Time zone string)
 "isNTP": [true|false],                 (NTP active)
 "NTPserver": string,                   (NTP server URL)
 "tHH": unsigned integer,               (manual date/time: hours)
 "tMM": unsigned integer,               (manual date/time: minutes)
 "dDD": unsigned integer,               (manual date/time: day)
 "dMM": unsigned integer,               (manual date/time: month)
 "dYY": unsigned integer                (manual date/time: year)
}
```

Returns

None.

{xe "cgiFktJsonEmail:CGI functions"}{xe "CGI functions:cgiFktJsonEmail"}void cgiFktJsonEmail (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the email settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the email settings panel (idEmail). Json syntax:

```
"bsEmail":
{
 "smtp": [true|false],                    (E-mail activated)
 "server": string,                        (SMTP server name)
 "port": unsigned integer,                (SMTP mail port)
 "account": string,                       (SMTP account name)
 "password": string,                      (SMTP account password)
 "eFrom": string,                         ("From" address)
 "eTo": string                            ("To" address)
}
```

Returns

None.

{xe "cgiFktJsonFsUpdate:CGI functions"}{xe "CGI functions:cgiFktJsonFsUpdate"}void cgiFktJsonFsUpdate (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for file system update panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the file system update panel (idFsUpd). Json syntax:

```
"bsFsUpd":
{
 "verFs": unsigned integer                (Major: 8 bit, Minor: 8 bit, Build: 16 bit)
}
```

Returns

None.

{xe "cgiFktJsonFSys:CGI functions"}{xe "CGI functions:cgiFktJsonFSys"}void cgiFktJsonFSys (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the file system panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the file system panel (idFilesystem). Json syntax:

```
"bsFilesystem":
{
 "fsSize": unsigned integer,           (Size of memory in Bytes)
 "fsFree": unsigned integer            (Size of available memory in Bytes)
}
```

Returns

None.

{xe "cgiFktJsonFtpd:CGI functions"}{xe "CGI functions:cgiFktJsonFtpd"}void cgiFktJsonFtpd (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the ftpd panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the ftpd panel (idFtp). Json syntax:

```
"bsFtp":
{
 "isFtpd": [true|false],               (FTP daemon activated)
 "ftpdUser": string,                   (FTP user name)
 "ftpdPwd": string,                    (FTP user password)
 "isWrite": [true|false]               (Write access activated)
}
```

Returns

None.

{xe "cgiFktJsonFwUpdate:CGI functions"}{xe "CGI functions:cgiFktJsonFwUpdate"}void cgiFktJsonFwUpdate (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for firmware update panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the firmware update panel (idFwUpd). Json syntax:

```
"bsFwUpd":
{
 "tftpServer": string,                 (TFTP Update server)
 "verSW": string,                      (Current software version)
 "verImg": unsigned integer,           (Software version of new image; Major: 8 bit,
Minor: 8 bit, Build: 16 bit)
 "imgState": unsigned integer,         (0: teTargetImageState.tiNone, 1:
teTargetImageState.tiArmed, 2: teTargetImageState.tiCopied, 3:
teTargetImageState.tiInvalid)
 "imgSize": unsigned integer           (Size of image in Bytes)
}
```

Returns

None.

{xe "cgiFktJsonHistory:CGI functions"}{xe "CGI functions:cgiFktJsonHistory"}void cgiFktJsonHistory (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the history panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the history panel (idHistory). Json syntax:

```
"bsHistory":
{
 "histTypes":
 [
  "10s",                               (History type with 10s resolution supported)
  "24h"                                (History type with 24h resolution supported)
 ]
}
```

Returns

None.

{xe "cgiFktJsonLAN:CGI functions"}{xe "CGI functions:cgiFktJsonLAN"}void cgiFktJsonLAN (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the LAN settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the LAN settings panel (idLAN). Json syntax:

```
"bsLAN":
{
 "DevName": string,                     (Device name)
 "hostip": string,                      (Host ip address)
 "netmask": string,                     (Netmask)
 "gateway": string,                     (Gateway ip address)
 "dns": string,                         (Optional DNS server)
 "isDHCP": [true|false]                 (DHCP enabled)
}
```

Returns

None.

{xe "cgiFktJsonMain:CGI functions"}{xe "CGI functions:cgiFktJsonMain"}void cgiFktJsonMain (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the main panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the main panel (idMain). Json syntax:

```
"bsMain":
{
 "relay0": [true|false],                (True: relay 0 is on)
 "relay1": [true|false],                (True: relay 1 is on)
 "energy": float,                       (Energy in Ws with a resolution of 100mWs)
 "pow_rel": integer,                    (Relative power 0...100%)
 "pow_abs": float                       (Power in W with a resolution of 100mW)
}
```

Returns

None.

{xe "cgiFktJsonPower:CGI functions"}{xe "CGI functions:cgiFktJsonPower"}void cgiFktJsonPower (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the power logging panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free

	space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the power logging panel (idPower). Json syntax:

```
"bsPower":
{
 "Plog10s": [true|false],              (Log 10s values)
 "Pmail10s": [true|false],             (Send e-mail for 10s values)
 "Plog24h": [true|false],              (Log 24h values)
 "Pmail24h": [true|false]              (Send e-mail for 24h values)
}
```

Returns

None.

{xe "cgiFktJsonScheduler:CGI functions"}{xe "CGI functions:cgiFktJsonScheduler"}void cgiFktJsonScheduler (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the scheduler settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the scheduler settings panel (idScheduler). Json syntax:

```
"bsScheduler":
{
 "schedMaxCount": 16,                  (Max. 16 schedulers supported)
 "schedRandMinutes": unsigned integer, (A randomize value of up to this value is added
to the switching point)
 "schedSeqOn": unsigned integer,       (Sequence "on" time in minutes)
 "schedSeqOff": unsigned integer,      (Sequence "off" time in minutes)
 "schedSeqDuration": unsigned integer, (Sequence length in minutes)
 "schedulers":
 [
  {
   "action": 0..1,                     (0: SCHEDACT_REL0, 1: SCHEDACT_REL1)
   "state": 0..2,                      (0: SCHEDSTATE_OFF, 1: SCHEDSTATE_ON, 2:
SCHEDSTATE_SEQ; Bit 0x80 (SCHEDSTATE_VAC) may be set)
   "day": unsigned integer,            (SCHEDDAY_MO, ..., SCHEDDAY_SU or
SCHEDDAY_DATE)
   "date": unsigned integer,           (see _sSchedTime.ui16Date)
   "hours": unsinged integer,          (see _sSchedTime)
   "minutes": unsigned integer         (see _sSchedTime)
  },
  ...
 ]
}
```

Returns

None.

{xe "cgiFktJsonSetSwitch:CGI functions"}{xe "CGI functions:cgiFktJsonSetSwitch"}void cgiFktJsonSetSwitch (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the switch settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the switch settings panel (idSetSwitch). Json syntax:

```
"bsSetSwitch":
{
 "chName0": string,                   (Name of relay 0)
 "chName1": string,                   (Name of relay 1)
 "bootSt0": 0..2,                     (Power-on action of relay 0: 0:
tePOnAction.POnOff, 1: tePOnAction.POnOn, 2: tePOnAction.POnLast)
 "bootSt1": 0..2,                     (Power-on action of relay 1: 0:
tePOnAction.POnOff, 1: tePOnAction.POnOn, 2: tePOnAction.POnLast)
 "delay0": unsigned integer,          (Power-on delay of relay 0 in seconds)
 "delay1": unsigned integer,          (Power-on delay of relay 1 in seconds)
 "reset0": unsigned integer,          (Re-activate delay of relay 0 in seconds)
 "reset1": unsigned integer           (Re-activate delay of relay 1 in seconds)
}
```

Returns

None.

{xe "cgiFktJsonSnmp:CGI functions"}{xe "CGI functions:cgiFktJsonSnmp"}void cgiFktJsonSnmp (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the snmp panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the snmp panel (idSnmp). Json syntax:

```
"bsSnmp":
{
 "isSnmp": [true|false],              (Activate SNMP)
 "isTraps": [true|false],             (Activate SNMP traps)
 "snmpTrapDest": string,              (Trap destination ip address)
 "snmpContact": string,               (Contact name)
 "snmpName": string,                  (Device name)
 "snmpLoc": string,                   (Location name)
 "snmpCommRead": string,              (Read community)
 "snmpCommWrite": string              (Write community)
}
```

Returns

None.

{xe "cgiFktJsonStandby:CGI functions"}{xe "CGI functions:cgiFktJsonStandby"}void cgiFktJsonStandby (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the standby panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the standby panel (idStandby). Json syntax:

```
"bsStandby":
{
 "standby": 0..2                       (0: teESleep.Ssleep, 1: teESleep.S100BT, 2:
teESleep.S10BT)
}
```

Returns

None.

{xe "cgiFktJsonState:CGI functions"}{xe "CGI functions:cgiFktJsonState"}void cgiFktJsonState (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the state panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the state panel (idState). Json syntax:

```
"bsState":
{
 "verSW": string,                      (Software version string)
 "verFS": unsigned integer,            (Major: 8 bit, Minor: 8 bit, Build: 16 bit)
 "verHW": unsigned integer,            (Major: 8 bit, Minor: 8 bit)
 "verBL": unsigned integer,            (Major: 8 bit, Minor: 8 bit, Build: 16 bit)
 "verImg": unsigned integer,           (Major: 8 bit, Minor: 8 bit, Build: 16 bit)
 "imgArmed": string,                   ("A": teTargetImageState.tiArmed, "I":
teTargetImageState.tiCopied, "-": teTargetImageState.tiNone)
 "features": unsigned integer,         (System features, see SYS_FEATURES)
 "sysRAM": unsigned integer,           (Size of RAM in Bytes)
 "sysFlash": unsigned integer,         (Size of flash memory in Bytes)
 "fsSize": unsigned integer,           (Size of file system)
 "fsFree": unsigned integer,           (Available space on file system)
 "ip": string,                         (Ip address of device)
 "mac": string,                        (MAC address of device)
 "visitors": unsigned integer,         (Number of visitors since system start)
 "reset": string,                          (Reset cause of system, one of
"SWR","WDR","BOR","POR","EXT","---")
 "tempCPU": string,                       (Current temperature of CPU in celsius degrees)
 "dateTime": "DD.MM.YYYY - HH:MM",        (Current date/time of system)
 "sync": unsigned integer,               (Time since sync of persistence data in seconds)
 "active": unsigned integer               (Time since startup in seconds)
}
```

Returns

None.

{xe "cgiFktJsonSyslog:CGI functions"}{xe "CGI functions:cgiFktJsonSyslog"}void cgiFktJsonSyslog (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the syslog panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the syslog panel (idSyslog). Json syntax:

```
"bsSyslog":
{
 "isSyslog": [true|false],                 (Activate syslog)
 "syslogIp": string,                       (Syslog server name or ip address)
 "syslogPort": unsigned integer            (Syslog port)
}
```

Returns

None.

{xe "cgiFktJsonSysSet:CGI functions"}{xe "CGI functions:cgiFktJsonSysSet"}void cgiFktJsonSysSet (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the system settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the system settings panel (idSysSet). Json syntax:

```
"bsSysSet":
{
 "BtnPS": [true|false],                    (Power dependent switch of master activated)
 "Toff": unsigned integer,                 (Power dependent switch of master : time in
seconds)
 "PSOff": unsigned integer,                (Power dependent switch of master : power in
Watt)
 "BtnMS": [true|false],                    (Slave follows master activated)
 "MSOn": unsigned integer,             (Slave follows master: "on"-value in Watt)
 "MSOff": unsigned integer,            (Slave follows master: "off"-value in Watt)
 "SlaveOn": unsigned integer,         (Slave follows master: "on"-delay in seconds)
 "SlaveOff": unsigned integer,        (Slave follows master: "off"-delay in seconds)
 "BtnWOL": [true|false],               (WOL activated)
 "WOLval": unsigned integer,           (WOL delay in seconds)
 "WOLMAC": "ff:ff:ff:ff:ff:ff"         (WOL target MAC)
}
```

Returns

None.

{xe "cgiFktJsonUPnP:CGI functions"}{xe "CGI functions:cgiFktJsonUPnP"}void cgiFktJsonUPnP (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the UPnP panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the UPnP panel (idUPnP). Json syntax:

```
"bsUPnP":
{
 "isUPnP": [true|false],               (UPnP activated)
 "upnpIntv": unsigned integer          (UPnP interval in seconds)
}
```

Returns

None.

{xe "cgiFktJsonUser:CGI functions"}{xe "CGI functions:cgiFktJsonUser"}void cgiFktJsonUser (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the user panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the user panel (idUser). Json syntax:

```
"bsUser":
{
 "maxUser": unsigned integer,          (Maximum amount of users)
 "user":
 [
  {
   "name": string,                     (Name of user)
   "passwd": string,                   (Password of user)
   "role": 0..4                       (0: teUserType.UserNone,1:
teUserType.UserGuest, 2: teUserType.UserStandard, 3: teUserType.UserAdmin, 4:
teUserType.UserEverybody)
  }
 ]
}
```

Returns

None.

{xe "cgiFktJsonWatchdog:CGI functions"}{xe "CGI functions:cgiFktJsonWatchdog"}void cgiFktJsonWatchdog (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the watchdog settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the watchdog settings panel (idWatchdog). Json syntax:

```
"bsWatchdog":
{
 "wdOn0": [true|false],               (Watchdog activated)
 "wdType0": 0..1,                     (0: teWDProt.WDP_ICMP, 1: teWDProt.WDP_TCP)
 "wdIp": string,                      (Host-IP)
 "wdPort0": unsigned integer,         (TCP-Port)
 "wdIntv0": unsigned integer,         (Ping interval in seconds)
 "wdRetry0": unsigned integer,        (Ping retry count)
 "wdWait0": [true|false],             (Wait for activity)
 "wdAct0": 0..1,                      (0: teWDAct.WDA_OFF, 1: teWDAct.WDA_RESET)
 "wdDelay0": unsigned integer,        (Reboot delay in seconds)
 ...                                  (Repeat for all other relays)
}
```

Returns

None.

{xe "cgiFktJsonWebserver:CGI functions"}{xe "CGI functions:cgiFktJsonWebserver"}void cgiFktJsonWebserver (char ** *ppcBuffer*, uint32_t * *pui32Len*, uint32_t * *pui32Level*, bool *bIsLast*)

Create Json string for the webserver settings panel.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32Len	is a pointer to the length of the buffer. On exit it will contain the remaining free space in the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates the Json string for the webserver settings panel (idWebserver). Json syntax:

```
"bsWebserver":
{
 "isActTimeout": [true|false],          (Activate timeout)
 "actTimeout": unsigned integer,        (Timeout in seconds)
 "httpRedir": [true|false]              (HTTP redirect to HTTPS)
}
```

Returns

None.

{xe "cgiFktLevelClose:CGI functions"}{xe "CGI functions:cgiFktLevelClose"}static void cgiFktLevelClose (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*, bool *bIsLast*) `[static]`

Add a part to Json string: closing bracket of a level.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.
bIsLast	is true, if this is the last entry on a hierarchy level.

This function creates a part of the Json string: the closing bracket of a level.

Returns

None.

{xe "cgiFktLevelOpen:CGI functions"}{xe "CGI functions:cgiFktLevelOpen"}static void cgiFktLevelOpen (char ** *ppcBuffer*, uint32_t * *pui32LenBuffer*, uint32_t * *pui32Level*) `[static]`

Add a part to Json string: level.

Parameters

ppcBuffer	is a pointer to the buffer to receive the resulting string.
pui32LenBuffer	is the length of the buffer.
pui32Level	is a pointer to a variable containing the JSON hierarchy.

This function creates a part of the Json string: the opening part of a level.

Returns

None.

{xe "cgiFktLoadFs:CGI functions"}{xe "CGI functions:cgiFktLoadFs"}bool cgiFktLoadFs (ptsHttpState *psHttpState*)

CGI function `/cgi/loadFs` : Upload file system image.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Upload a lmi flash file system image to the system. Request-type: POST.

Syntax:

```
/cgi/loadFs
```

CGI-parameters: None.

Reply: The html page `/index.htm` is returned.

Returns

False, if not all data have been receive. True, if the last package has been received and handled.

{xe "cgiFktLoadFw:CGI functions"}{xe "CGI functions:cgiFktLoadFw"}bool cgiFktLoadFw (ptsHttpState *psHttpState*)

CGI function `/cgi/loadFw` : Upload firmware image to the file system.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Upload a firmware image to the file system and mark it for installation. Request-type: POST.

Syntax:

`/cgi/loadFw`

CGI-parameters: None.

Reply: The html page `/index.htm` is returned.

Returns

True, if last block has been received, false otherwise.

{xe "cgiFktLogoff:CGI functions"}{xe "CGI functions:cgiFktLogoff"}bool cgiFktLogoff (ptsHttpState *psHttpState*)

CGI function `/cgi/logoff` : Logoff current user.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Logs off the current user. Request-type: GET.

Syntax:

`/cgi/logoff`

CGI-parameters: None.

Reply: None.

Returns

always true.

{xe "cgiFktMarkForInst:CGI functions"}{xe "CGI functions:cgiFktMarkForInst"}bool cgiFktMarkForInst (ptsHttpState *psHttpState*)

CGI function `/cgi/markInst` : Mark downloaded firmware image for installation.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Mark a downloaded firmware image for installation.

If the flash reserved area is used: The update process will transfer the image into the flash region for the application.

If the image is a file in the file-system: The software update image file will receive the SYSTEM attribute set.

Request-type: GET.

Syntax:

`/cgi/markInst`

CGI-parameters: None.

Reply: `["0"|"1"]`

0 if file has been armed, 1 if there is no valid file to be armed.

Returns

always true.

{xe "cgiFktQueryImageData:CGI functions"}{xe "CGI functions:cgiFktQueryImageData"}bool cgiFktQueryImageData (ptsHttpState *psHttpState*)

CGI function `/cgi/getImgData` : queries the image version and size of the image file in the serial data flash.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Queries the version, arm-state and size of the downloaded firmware image file in the serial data flash. Request-type: GET.

Deprecated:

Replacement see cgiFktJsonFwUpdate

Syntax:

`/cgi/getImgData`

CGI-parameters: None.

Reply: `<version>|["-"|"A"|"I"]|<size>`

with:

`version` as the image file software version string.

"-" if the image is not armed, "A" if the image is armed for installation, "I" if the image has been installed.

`size` as the image file size in bytes.

Returns

always true.

{xe "cgiFktReboot:CGI functions"}{xe "CGI functions:cgiFktReboot"}bool cgiFktReboot (ptsHttpState *psHttpState*)

CGI function `/cgi/reboot` : reboots the system.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Performs a system reset using SysCtlReset(). Request-type: GET.

Syntax:

`/cgi/reboot`

CGI-parameters: None.

Reply:

There is no reply value.

Returns

always true.

{xe "cgiFktRestore:CGI functions"}{xe "CGI functions:cgiFktRestore"}bool cgiFktRestore (ptsHttpState *psHttpState*)

CGI function `/cgi/restore` : Restore persistence data.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Restore device settings by setting the persistence data structure. Request-type: POST.

Syntax:

`/cgi/restore`

CGI-parameters: None.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSendTestMail:CGI functions"}{xe "CGI functions:cgiFktSendTestMail"}bool cgiFktSendTestMail (ptsHttpState *psHttpState*)

CGI function `/cgi/testMail` : Sends a test e-mail.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Sends a test-email with the current settings. Request-type: GET.

Syntax:

`/cgi/testMail`

CGI-parameters: None.

Reply: There is no reply value.

Returns

always true.

{xe "cgiFktSetDateTime:CGI functions"}{xe "CGI functions:cgiFktSetDateTime"}bool cgiFktSetDateTime (ptsHttpState *psHttpState*)

CGI function `/cgi/setDTSet` : set date/time of the device, including ntp settings. Return html page `/set_std.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the date and time settings of the device. Also the usage of the ntp protocol can be selected. Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setDTSet?tzone=<tz>&BtnTIME=[NTP|MAN]&NTPserver=<host>&
tHH=<hour>&tMM=<minute>&dDD=<day>&dMM=<month>&dYY=<year>&SUB
=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`tzone=<tz>` is the time zone string.
`BtnTime=` [NTP|MAN] selects between manual and ntp time setting.

Conditional parameters:

If `BtnTime` == NTP:

- `NTPserver=<host>` is the ntp server as ip-address or url.

If `BtnTime` = MAN:

- `tHH=<hour>` are the hours of the current time.
- `tMM=<minute>` are the minutes of the current time.
- `dDD=<day>` is the day of the current date.
- `dMM=<month>` is the month of the current date.
- `dYY=<year>` is the year of the current date.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetEnetMode:CGI functions"}{xe "CGI functions:cgiFktSetEnetMode"}bool cgiFktSetEnetMode (ptsHttpState *psHttpState*)

CGI function `/cgi/setEnetMode` : Set the ethernet PHY speed. Return html page `/set_exp.ssi`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the ethernet PHY speed. Return the `/set_exp.ssi` html page. Request-type: GET.

Syntax:

```
/cgi/setEnetMode?SB=[100BT|10BT|sleep]&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`SB=` [100BT|10BT|sleep] is the selected speed of the ethernet PHY:

- `100BT` stands for 10Base-T or 100Base-T,
- `10BT` stands for 10Base-T,
- `sleep` stands for 10Base-T or 100Base-T; 10Base-T in sleep-mode.

Reply: The html page `/set_exp.ssi` is returned.

Returns

always true.

{xe "cgiFktSetFactory:CGI functions"}{xe "CGI functions:cgiFktSetFactory"}bool cgiFktSetFactory (ptsHttpState *psHttpState*)

CGI function `/cgi/setFactory` : Reset device to factory settings.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Reset device to factory settings. Request-type: GET.

Syntax:

`/cgi/setFactory?Reset=Reset`

CGI-parameters:

`Reset=Reset` is the only parameter to indicate a request for the factory reset.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetFileSystem:CGI functions"}{xe "CGI functions:cgiFktSetFileSystem"}bool cgiFktSetFileSystem (ptsHttpState *psHttpState*)

CGI function `/cgi/formatFs` : Format the flash file-system. Return html page `/set_exp.ssi` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function formats the flash file system and returns the available size in the flash memory in kByte. Request-type: GET.

Syntax:

`/cgi/formatFs?Format=Fmt`

CGI-parameters:

`Format=Fmt` is the submit code to start the action.

Reply: `<SizeFS>|<FreeFS>` ,

with:

`SizeFS` as the size of the formatted file system,
`FreeFS` as the size of the free memory in the file system.

Returns

always true.

{xe "cgiFktSetFtpdSettings:CGI functions"}{xe "CGI functions:cgiFktSetFtpdSettings"}bool cgiFktSetFtpdSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setFtpdSet` : Set-up the ftp daemon. Return html page `/set_lan.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets-up the ftp daemon settings. Return the `/set_lan.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setFtpdSet?ftp=ftp&ftpU=<user>&ftpP=<pwd>&ftpW=ftpW&SUB
=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`ftp=ftp` is present, if the ftp daemon is activated.
`ftpU=<user>` represents the ftp user name.
`ftpP=<pwd>` represents the ftp user password.
`ftpW=ftpW` is present, if write access is granted.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetIPACLSettings:CGI functions"}{xe "CGI functions:cgiFktSetIPACLSettings"}bool cgiFktSetIPACLSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setIpaclSet` : Set-up the ip access control list (ACL). Return html page `/set_lan.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the ip access control list (ACL). Return the `/set_lan.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setIpaclSet&ACLF=ACLON&ACL<n>=<ip>&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`ACLF=ACLON` is present, if the ACL is activated.
`ACL<n>=<ip>`.

Where:

`<n>` is the index of the ACL entry,
`<ip>` is the ip address of an allowed client.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetIPData:CGI functions"}{xe "CGI functions:cgiFktSetIPData"}bool cgiFktSetIPData (ptsHttpState *psHttpState*)

CGI function `/cgi/setIP` : set the ip-settings of the device. Set the ip settings of the device.

Parameters

psHttpState	is the instance variable of the HTTP connection.

CGI function to set the ip settings of the device. Request-type: GET.

Syntax:

```
/cgi/setIP?DevName=<name>&GetIP=<type>&hostip=<h_ip>&netmask
=<mask>&gateway=<g_ip>&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`DevName=<name>` is the name of the device.
`GetIP=` [DHCP|Man] is the address resolution mode. **DHCP** activates the dhcp client, **Man** addresses a manual entry of data.
`hostip=<h_ip>` is the host ip-address of the device, transferred as a string.
`netmask=<mask>` is the network mask, transferred as a string.
`gateway=<g_ip>` is the gateway ip-address of the device, transferred as a string.
`dns=<d_ip>` is the name of the dns server, transferred as string.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetPwrLogSettings:CGI functions"}{xe "CGI functions:cgiFktSetPwrLogSettings"}bool cgiFktSetPwrLogSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setPwLogSet` : set-up power logging feature. Return html page `/set_std.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the behavior of power logging. Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setPwLogSet?Plog=[10s|24h|all]&Pmail=[10s|24h|all]&SUB=
Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`Plog=` [10s|24h|all] switches the power logging.

- `10s` sets logging to 10s-values only,
- `24h` sets logging to 24h-values only,
- `all` sets logging for all values.

`Pmail=` [10s|24h|all] switches email setting.

- `10s` enables email for 10s-values only,
- `24h` enables email for 24h-values only,
- `all` enables email for all values.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetRelay:CGI functions"}{xe "CGI functions:cgiFktSetRelay"}bool cgiFktSetRelay (ptsHttpState *psHttpState*)

CGI function `/cgi/setRelay` : Set relay.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Set a relay. Request-type: GET.

Syntax:

```
/cgi/cgiFktSetRelay?Rel=<index>&State=[0|1]
```

CGI-parameters:

`Rel=<index>` is the 0-based index of the relay.
`State=` [0|1] is the action (off/on).

Reply: There is no reply value.

Returns

always true.

{xe "cgiFktSetScheduler:CGI functions"}{xe "CGI functions:cgiFktSetScheduler"}bool cgiFktSetScheduler (ptsHttpState *psHttpState*)

CGI function `/cgi/setScheduler` : Set-up the schedulers.

Parameters

psHttpState	is the instance variable of the HTTP connection.

CGI function to set-up the schedulers. Request-type: POST.

Syntax:

```
/cgi/setScheduler?SUB=<action>&Rel<n>=<index>&Act<n>=<[0|1]>
&Mod<n>=<[0|1>&Mav=<rand>&MSqE=<seqOn>&MSqA=<seqOff>&MSqD=<d
elay>
```

CGI-parameters:

`SUB=Apply` is the submit code.
`Rel<n>=` [0|1] is the relay index.
`Act<n>=` [0|1] is the action (off/on).
`Mod<n>=` [0|1] is the scheduler mode (day/weekday).
`vcf<n>=<x>` switches on vacation function with x='V'.
`Mav=<rand>` is the randomize number of minutes for the vacation function (between 0 and rand minutes).
`MSqE=<seqOn>` is the number of minutes to switch sequence on.
`MSqA=<seqOff>` is the number of minutes to switch sequence off.
`MSqD=<delay>` is the number of minutes until sequence starts.

Conditional parameters:

If `Mod` == 0:

- `Ddd<n>=<day>` is the day of month.
- `Dmm<n>=<month>` is the month of year.
- `Dyy<n>=<year>` is the year.

If `Mod` == 1:

- `Day<n>=<weekday-bitfield>` is the bit field for the weekday, with 0=Monday, 1=Tuesday, 2=Wednesday, ...
- `Thh<n>=<hour>` is the hour.
- `Tmm<n>=<minute>` is the minute.

Where:

`<n>` is the timer number (0-based index).

Reply: The html page `/timer.ssi` is returned.

Returns

always true.

{xe "cgiFktSetSmtpSettings:CGI functions"}{xe "CGI functions:cgiFktSetSmtpSettings"}bool cgiFktSetSmtpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setEmailSet` : Set-up the e-mail client. Return html page `/set_std.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the parameters for the e-mail client. Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setEmailSet?smtp=smtp&srvr=<server>&port=<port>&accnt=<account>&pwd=<passwd>&eFrom=<from>&eTo=<to>&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`smtp=smtp` is present, if the e-mail client is activated.
`srvr=<server>` is the smtp e-mail server url.
`port=<port>` is the smtp e-mail server port.
`accnt=<account>` is the e-mail account.
`pwd=<passwd>` is the e-mail account password.
`eFrom=<from>` is the e-mail originator.
`eTo=<to>` is the e-mail recipient.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetSnmpSettings:CGI functions"}{xe "CGI functions:cgiFktSetSnmpSettings"}bool cgiFktSetSnmpSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setSnmpSet` : Set-up the snmp settings. Return html page `/set_lan.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets up the snmp agent settings. Return the `/set_lan.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setSnmpSet?StSNMP=SS1&StTrap=ST1&TDIP=<ip>&Cnt=<contact
>&Name=<name>&Loc=<position>&Comm=<community>&WrComm=<commun
ity>&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`StSNMP=SS1` is present, if SNMP is activated.
`StTrap=ST1` is present, if SNMP traps are activated.
`TDIP=<ip>` is the trap destination ip address.
`Cnt=<contact>` is the contact string.
`Name=<name>` is the name string.
`Loc=<position>` is the position string.
`Comm=<community>` is the community string.
`WrComm=<community>` is the write community string.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetSwitchSettings:CGI functions"}{xe "CGI functions:cgiFktSetSwitchSettings"}bool cgiFktSetSwitchSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setSwitchSet` : set behavior of the relays. Return html page `/set_std.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the behavior of the relays: name, delays, behavior after a system startup etc. Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setSwitchSet?Name<index>=<name>&Btn<index>=[Off|On|Last
]&Sw<index>s=<s1>&SwO<index>s=<s2>&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`Name<index>=<name>` is the name of the relay.
`Btn<index>=` [Off|On|Last] is the behavior after power on:

- `Off` leaves the relay in off mode after power on,
- `On` switches the relay on after power on,
- `Last` restores the state of the relay before power off.

`Sw<index>s=<s1>` is the delay time in seconds to switch the relay on after a power on.
`SwO<index>s=<s2>` is the period in seconds after which the relay is switched on again after switching it off.

Where `<index>` is the 0-based index of the relay.

Reply: The html page `/set_std.htz` is returned if g_bIsFileReply is true.

Returns

always true.

{xe "cgiFktSetSysFunction:CGI functions"}{xe "CGI functions:cgiFktSetSysFunction"}bool cgiFktSetSysFunction (ptsHttpState *psHttpState*)

CGI function `/cgi/setSysSet` : set system settings of the device. Return html page `/set_std.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the system settings for **Master/Slave** as well as **Wake-On-LAN** . Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setSysSet?BtnWOL=WOL&WOLval>=<time>&WOLmac=<mac>&BtnMS=
MS&MSOn=<threshold>&MSOff=<threshold>&SlaveOn=<time>&SlaveOf
f=<time>&Sub=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.

`BtnWOL=WOL` activates the **Wake-On-LAN** function.

`WOLval=<time>` is the delay time between switching relay 0 and sending the WOL package in seconds.

`WOLmac=<mac>` is the target MAC address of the Wake-On-LAN function. It has to follow the form xx.xx.xx.xx.xx.xx, with xx as a hexadecimal value.

`BtnMS=MS` activates the **Master/Slave** function.

`MSOn=<threshold>` is the threshold in Watt to switch the slave on.

`MSOff=<threshold>` is the threshold in Watt to switch the slave off.

`SlaveOn=<time>` is the delay time in seconds to switch the slave on.

`SlaveOff=<time>` is the delay time in seconds to switch the slave off.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetSyslogSettings:CGI functions"}{xe "CGI functions:cgiFktSetSyslogSettings"}bool cgiFktSetSyslogSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setSyslogSet` : Set-up the syslog service. Return html page `/set_lan.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the parameters for the syslog service. Return the `/set_lan.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setSyslogSet?SLState=SL1&SLip=<ip>&SLport=<port>&SUB=Ap
ply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`SLState=SL1` is present, if the syslog service is activated.
`SLip=<ip>` is the ip-address of the syslog server.
`SLport=<port>` is the port for the syslog service.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetTempSettings:CGI functions"}{xe "CGI functions:cgiFktSetTempSettings"}bool cgiFktSetTempSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setTempSet` : Set-up temperature measurement. Return html page `/set_std.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets-up temperature measurement: the interface type and actions on crossing temperature thresholds. Return the `/set_std.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setTempSet?Tmss=Tmss&Tif=Tif[0|1]&Tmail=Tmail&Tm=<temp>
&T1=T1&T1V=<t1>&T1S=T1S[0|1]&T2=T2&T2V=<t2>&T2S=T2S[0|1]&SUB
=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`SubmT=` [R|S] is the action:

- R: requests a reset of the temperature min/max values,
- S: requests to set new parameters.

`Tmss=Tmss` is present, if temperature measurement is switched on.
`Tif=Tif` [0|1] represents the interface type:

- Tif0 stands for the I²S interface with a LM75B sensor,
- Tif1 stands for the 1-wire interface with a DS18B20 sensor.

`Tmail=Tmail` is present, if e-mail sending is activated.
`Tm=<temp>` represents the temperature which results in sending an e-mail.
`T1=T1` is present, if relay 1 reacts on a temperature threshold.
`T1V=<t1>` represents the temperature on which relay 1 reacts.
`T1S=T1S` [0|1] is the action for relay 1:

- T1S0: the relay will be switched on,
- T1S1: the relay will be switched off.

`T2=T2` is present, if relay 2 reacts on a temperature threshold.
`T2V=<t2>` represents the temperature on which relay 2 reacts.
`T2S=T2S` [0|1] is the action for relay 2:

- T2S0: the relay will be switched on,

- T2S1: the relay will be switched off.

Reply: The html page `/set_std.htz` is returned.

Returns

always true.

{xe "cgiFktSetTFTPServer:CGI functions"}{xe "CGI functions:cgiFktSetTFTPServer"}bool cgiFktSetTFTPServer (ptsHttpState *psHttpState*)

CGI function `/cgi/swUpdateSrv` : initiate a software update. Return html page `/update.ssi`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function initiates a software update. Return the `/update.ssi` html page. Request-type: GET.

Syntax:

`/cgi/swUpdateSrv?TFTPserver=<server>&SUB=Apply`

CGI-parameters:

`SUB=Apply` is the submit code.
`TFTPserver=<server>` is the name of the tftp update-server.

Reply: The html page `/update.ssi` is returned.

Returns

always true.

{xe "cgiFktSetUPnPSettings:CGI functions"}{xe "CGI functions:cgiFktSetUPnPSettings"}bool cgiFktSetUPnPSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setUpnpSet` : Set-up the UPnP settings. Return html page `/set_lan.htz` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function configures the UPnP function. Return the `/set_lan.htz` html page. Request-type: GET.

Syntax:

`/cgi/setUpnpSet?upnp=upnp&intv=<intv>&SUB=Apply`

CGI-parameters:

`SUB=Apply` is the submit code.
`upnp=upnp` is present, if the UPnP function is active.
`intv=<intv>` is the UPnP advertisement interval in seconds.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetUser:CGI functions"}{xe "CGI functions:cgiFktSetUser"}bool cgiFktSetUser (ptsHttpState *psHttpState*)

CGI function `/cgi/setUser` : Set user name and password.

Parameters

psHttpState	is the instance variable of the HTTP connection.

CGI function to set user name and password. Request-type: GET.

Syntax:

```
/cgi/setUser?User1=<username>&Pwd11=<password>&Pwd12=<passwo
rd>&      User2=<username>&Pwd21=<password>&Pwd22=<password>&
User3=<username>&Pwd31=<password>&Pwd32=<password>&SUB="Appl
y"
```

CGI-parameters:

`User<num>=<username>` is the user name.
`Pwd<num>1=<password>` is the password.
`Pwd<num>2=<password>` is the repetition of the password
`SUB="Apply"` is the submit code.

Where `<num>` is the user number (1 = "guest", 2 = "standard", 3 = "admin").

Reply: The html page "/user.ssi" is returned.

Returns

always true.

{xe "cgiFktSetWatchdog:CGI functions"}{xe "CGI functions:cgiFktSetWatchdog"}bool cgiFktSetWatchdog (ptsHttpState *psHttpState*)

CGI function `/cgi/setWdSet` : Set-up the watchdog function. Return html page `/set_lan.htz`.

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets up the watchdog function to supervise external ip based devices. Return the `/set_exp.htz` html page. Request-type: GET.

Syntax:

```
/cgi/setWdSet?WD<n>state=WD&WD<n>Type=[WDICMP|WDTCP]&WD<n>ip
=<ip>&WD<n>p=<port>&WD<n>Intv=<intv>&WD<n>Retry=<retry>&
WD<n>Wait=WA&WD<n>Act=WD<n>Act[0|1]&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`WD<n>state=WD` is present, if the watchdog function is activated.
`WD<n>Type=` [WDICMP|WDTCP] stands for the watchdog type:

- `WDICMP` stands for an ICMP ping,
- `WDTCP` stands for a TCP ping.

`WD<n>ip=<ip>` is the supervised ip address.
`WD<n>p=<port>` is the port of the supervised device.

`WD<n>Intv=<intv>` is the ping interval.
`WD<n>Retry=<retry>` is the retry count.
`WD<n>Wait=WA:` if present, the watchdog function starts after a first ping has been answered. This avoids a permanent reset of devices with long boot-times.
`WD<n>Act=WD<n>Act` [0|1] is the watchdog action:

- `0` stands for the "switch off" action,
- `1` stands for the "reboot" action.

Reply: The html page `/set_lan.htz` is returned.

Returns

always true.

{xe "cgiFktSetWebSrvSettings:CGI functions"}{xe "CGI functions:cgiFktSetWebSrvSettings"}bool cgiFktSetWebSrvSettings (ptsHttpState *psHttpState*)

CGI function `/cgi/setWebsrv` : Set-up the web server. Return html page `/set_exp.ssi` .

Parameters

psHttpState	is the instance variable of the HTTP connection.

This cgi-function sets the parameters for the web server. Return the `/set_exp.ssi` html page. Request-type: GET.

Syntax:

```
/cgi/setWebsrv?Wact=Wact&IATime=<interval>&isHttpRedir=HttpR
edir&SUB=Apply
```

CGI-parameters:

`SUB=Apply` is the submit code.
`Wact=Wact` is present, if the web server has an inactivity timeout.
`IATime=<interval>` defines the inactivity timeout in seconds.
`isHttpRedir=HttpRedir` is present, if http requests will be redirected to https.

Reply: The html page `/set_exp.ssi` is returned.

Returns

always true.

{xe "cgiFktSwUpdate:CGI functions"}{xe "CGI functions:cgiFktSwUpdate"}bool cgiFktSwUpdate (ptsHttpState *psHttpState*)

CGI function `/cgi/downld` : download a firmware image.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Downloads a firmware image from tftp-server for software update, if available. Request-type: GET.

Syntax:

```
/cgi/downld
```

CGI-parameters: None.

Reply: There is no reply value.

Returns

always true.

{xe "cgiFktTemperatureGet:CGI functions"}{xe "CGI functions:cgiFktTemperatureGet"}bool cgiFktTemperatureGet (ptsHttpState *psHttpState*)

CGI function `/cgi/getTemp` : Request temperature values.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Retrieve the current temperatures of the system. Request-type:GET.

Syntax:

`/cgi/getTemp`

CGI-parameters: None.

Reply: `<TAct>|<TMin>|<TMax>]` ,

with:

`TAct` is the current temperature,
`TMin` is the minimum temperature measured,
`TMax` is the maximum temperature measured.

Returns

always true.

{xe "cgiFktTempReset:CGI functions"}{xe "CGI functions:cgiFktTempReset"}bool cgiFktTempReset (ptsHttpState *psHttpState*)

CGI function `/cgi/TempReset` : Resets the Min/Max values of temperature measurement.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Resets the Min/Max values of temperature measurement. Request-type: GET.

Syntax:

`/cgi/TempReset`

CGI-parameters: None.

Reply: There is no reply value.

Returns

always true.

{xe "cgiFktToggleRelay:CGI functions"}{xe "CGI functions:cgiFktToggleRelay"}bool cgiFktToggleRelay (ptsHttpState *psHttpState*)

CGI function `/cgi/toggleRelay` : Toggle relay.

Parameters

psHttpState	is the instance variable of the HTTP connection.

Toggle a relay. Request-type: GET.

Syntax:

```
/cgi/toggleRelay?Rel=<index>
```

CGI-parameters:

`Rel=<index>` is the 0-based index of the relay.

Reply: There is no reply value.

Returns

always true.

Variable Documentation

{xe "file_cgiFktFirst:CGI functions"}{xe "CGI functions:file_cgiFktFirst"}const tsFsDataFile file_cgiFktFirst `[extern]`

Root entry of cgi function for the file-system.

{xe "g_bIsFileReply:CGI functions"}{xe "CGI functions:g_bIsFileReply"}bool g_bIsFileReply = true

Compatibility mode for old web-pages using frames: This is the default. CGI set-functions reply complete html files. If value is set to false, only the http-error code is returned. Updates of the web-page has to be done with a query of the complete Json-structure.

{xe "g_bIsFileReply:CGI functions"}{xe "CGI functions:g_bIsFileReply"}bool g_bIsFileReply `[extern]`

Compatibility mode for old web-pages using frames: This is the default. CGI set-functions reply complete html files. If value is set to false, only the http-error code is returned. Updates of the web-page has to be done with a query of the complete Json-structure.

{xe "g_bRestoreSuccess:CGI functions"}{xe "CGI functions:g_bRestoreSuccess"}bool g_bRestoreSuccess = true

Controls the contents of the SSI-tag RestoreRet: success of /bin/restore.

{xe "g_pcErrorReply:CGI functions"}{xe "CGI functions:g_pcErrorReply"}const char g_pcErrorReply[] = "/error.ssi"

The file sent back to the browser in case when an error has been detected by one of the CGI handlers.

{xe "g_pcErrorReply:CGI functions"}{xe "CGI functions:g_pcErrorReply"}const char g_pcErrorReply[] `[extern]`

The file sent back to the browser in case when an error has been detected by one of the CGI handlers.

{xe "g_pConnForLoad:CGI functions"}{xe "CGI functions:g_pConnForLoad"}void* g_pConnForLoad = NULL

Pointer to ptsHttpState that has an active file system / system image update running (cgiFktLoadFs() or cgiFktLoadFw()).

{xe "pcConfiguration:CGI functions"}{xe "CGI functions:pcConfiguration"}const char pcConfiguration[] = xstr(RELAY_COUNT)"|"xstr(POWM_COUNT)

Configuration: r relays, p power measurement devices.

{xe "pcCrLf:CGI functions"}{xe "CGI functions:pcCrLf"}const char pcCrLf[] = "\r\n"

String for single line end.

{xe "pcCrLfCrLf:CGI functions"}{xe "CGI functions:pcCrLfCrLf"}const char pcCrLfCrLf[] = "\r\n\r\n"

String for double line end.

{xe "pcCtlPwd1:CGI functions"}{xe "CGI functions:pcCtlPwd1"}const char* const pcCtlPwd1[] = { "upwd0", "upwd1", "upwd2" }

Name of entry-field for password 1 on html page.

{xe "pcCtlPwd2:CGI functions"}{xe "CGI functions:pcCtlPwd2"}const char* const pcCtlPwd2[] = { "upwd20", "upwd21", "upwd22" }

Name of entry-field for password 2 on html page.

{xe "pcCtlType:CGI functions"}{xe "CGI functions:pcCtlType"}const char* const pcCtlType[] = { "utype0", "utype1", "utype2" }

Name of dropdown-field for user type on html page.

{xe "pcCtlUser:CGI functions"}{xe "CGI functions:pcCtlUser"}const char* const pcCtlUser[] = { "uname0", "uname1", "uname2" }

Name of entry-field for user on html page.

{xe "pcFalse:CGI functions"}{xe "CGI functions:pcFalse"}const char pcFalse[] = "0"

CGI return value for "false".

{xe "pcIdAclIp:CGI functions"}{xe "CGI functions:pcIdAclIp"}const char pcIdAclIp[] = "acl?" `[static]`

Id of entry field to receive an ip address for the ACL list. The question mark will be replace by the id index ('1' ... '8').

{xe "pcIndex:CGI functions"}{xe "CGI functions:pcIndex"}const char pcIndex[] = "/index.htm"

Reply html page for "/cgi/loadFs".

{xe "pcNoTemp:CGI functions"}{xe "CGI functions:pcNoTemp"}const char pcNoTemp[] = "-|-|-"

CGI return value, if temperature measurement is not present.

{xe "pcOff:CGI functions"}{xe "CGI functions:pcOff"}const char pcOff[] = "off"

String for "LED off".

{xe "pcOff:CGI functions"}{xe "CGI functions:pcOff"}const char pcOff[] `[extern]`

String for "LED off".

{xe "pcOn:CGI functions"}{xe "CGI functions:pcOn"}const char pcOn[] = "on"

String for "LED on".

{xe "pcOn:CGI functions"}{xe "CGI functions:pcOn"}const char pcOn[] `[extern]`

String for "LED on".

{xe "pcPanelId:CGI functions"}{xe "CGI functions:pcPanelId"}const char* const pcPanelId[][static]

```
Initial value:=
{
    "bsHeader",
    "bsMain",
    "bsHistory",
    "bsScheduler",
    "bsState",
    "bsSetSwitch",
    "bsSysSet",
    "bsDateTime",
    "bsPower",
    "bsLAN",
    "bsEmail",
    "bsWatchdog",
    "bsStandby",
    "bsWebserver",
    "bsSnmp",
    "bsSyslog",
    "bsUPnP",
    "bsFtp",
    "bsAcl",
    "bsFilesystem",
    "bsSaveRestore",
    "bsFwUpd",
    "bsFsUpd",
    "bsUser",
    "bsLogoff"
}
```

All available panels of web-site. Position within array corresponds to its id.

{xe "pcScheduler:CGI functions"}{xe "CGI functions:pcScheduler"}const char pcScheduler[] = "/timer.ssi"

Reply html page for "/cgi/setScheduler".

{xe "pcSetExp:CGI functions"}{xe "CGI functions:pcSetExp"}const char pcSetExp[] = "/set_exp.ssi"

Reply html page for "/cgi/setWSTimeout", "/cgi/formatFs", "/cgi/setEnetMode".

{xe "pcSetLAN:CGI functions"}{xe "CGI functions:pcSetLAN"}const char pcSetLAN[] = "/set_lan.htz"

Reply html page for "/cgi/setSyslogSet", "/cgi/setIpaclSet", "/cgi/setWdSet", "/cgi/setFtpdSet".

{xe "pcSetStd:CGI functions"}{xe "CGI functions:pcSetStd"}const char pcSetStd[] = "/set_std.htz"

Reply html page for "/cgi/setIP", "/cgi/setSysSet", "/cgi/setEmailSet", "/cgi/setTempSet", "/cgi/restore".

{xe "pcSetUser:CGI functions"}{xe "CGI functions:pcSetUser"}const char pcSetUser[] = "/user.ssi"

Reply html page for "/cgi/setUser".

{xe "pcTrue:CGI functions"}{xe "CGI functions:pcTrue"}const char pcTrue[] = "1"

CGI return value for "true".

{xe "pcUpdate:CGI functions"}{xe "CGI functions:pcUpdate"}const char pcUpdate[] = "/update.ssi"

Reply html page for "/cgi/swUpdateSrv".

{xe "ppcStateReply:CGI functions"}{xe "CGI functions:ppcStateReply"}const char* const ppcStateReply[3] = {"0", "1", "2"}

CGI return value for cgiFktGetMailState.