



Telemotive Client Library 3.0.1

User's manual

Generated by Doxygen 1.8.0

Wed Dec 7 2016 13:08:33

# Inhaltsverzeichnis

<b>1</b>	<b>User's manual - Telemotive Client Library 3.0.1</b>	<b>1</b>
1.1	General	1
1.2	Functionality	1
1.3	Compiler/Linker	2
1.4	Thread safety	2
1.5	Demo project	2
<b>2</b>	<b>Deprecated List</b>	<b>15</b>
<b>3</b>	<b>Hierarchical Index</b>	<b>16</b>
3.1	Class Hierarchy	16
<b>4</b>	<b>Class Index</b>	<b>17</b>
4.1	Class List	17
<b>5</b>	<b>File Index</b>	<b>19</b>
5.1	File List	19
<b>6</b>	<b>Class Documentation</b>	<b>20</b>
6.1	BPNGDevice Struct Reference	20
6.2	BPNGError Struct Reference	21
6.3	BPNGLoggerDetector Class Reference	21
6.4	CANPseudoMessagesProperties Struct Reference	27
6.5	ClientProperties Struct Reference	28
6.6	CommonProperties Struct Reference	29
6.7	ConversionProperties Struct Reference	29
6.8	DataSpan Struct Reference	30
6.9	IBPNGClient Struct Reference	30
6.10	IBPNGClientListener Struct Reference	46
6.11	IChannel Struct Reference	51
6.12	IChannelList Struct Reference	52
6.13	IClientProperties Struct Reference	53
6.14	IConversionSet Struct Reference	57
6.15	IFalseMeasureSignal Struct Reference	59
6.16	IFalseMeasureSignalList Struct Reference	60
6.17	IFormatInfo Struct Reference	60
6.18	IFormatList Struct Reference	61
6.19	IRdbEvent Struct Reference	61
6.20	IRdbEventList Struct Reference	62
6.21	IRdbTraceBlock Struct Reference	62
6.22	IRdbTraceBlockList Struct Reference	62
6.23	ITesttoolsChannel Struct Reference	62
6.24	ITesttoolsChannellist Struct Reference	63

6.25	LogInData Struct Reference . . . . .	63
6.26	MemoryFillLevel Struct Reference . . . . .	64
6.27	MOSTPseudoMessagesProperties Struct Reference . . . . .	64
6.28	OnlineLoggerInfo Struct Reference . . . . .	65
6.29	RdbEvent2 Struct Reference . . . . .	66
6.30	RdbEventList Class Reference . . . . .	67
6.31	TSLCluster Class Reference . . . . .	68
<b>7</b>	<b>File Documentation</b>	<b>72</b>
7.1	BPNGDefines.h File Reference . . . . .	72
7.2	BPNGLoggerDetector.hh File Reference . . . . .	81
7.3	IBPNGClient.h File Reference . . . . .	82
7.4	IBPNGClientListener.h File Reference . . . . .	84
7.5	IClientProperties.h File Reference . . . . .	85
7.6	RdbDefines.h File Reference . . . . .	85
7.7	RdbEventList.hh File Reference . . . . .	86
	<b>Index</b>	<b>86</b>

# Kapitel 1

## User's manual - Telemotive Client Library 3.0.1

### 1.1 General

This is the documentation for the C++ Telemotive Client library which is compatible with all Microsoft compilers. The library's interface class [IBPNGClient](#) uses only base data type parameters like *int*, *long* and *char*, pointers to those types and pointers to complex proprietary data objects that are entirely defined within the library. To access the data of such objects the library comes with own interface definitions for all of those complex data types (like e.g. [IConversionSet](#), see [BPNGDefines.h](#)). All library functions are blocking functions. Status and progress information is processed via listener callbacks (see [IBPNGClientListener](#)). Errors are processed by the functions' return values (see section Error handling for more details).

### 1.2 Functionality

The Telemotive Client Library provides methods for base functionality like:

- downloading the logger's/TSL raw trace data as offline data sets
- converting trace data to nearly all common file formats
- reading and reconfiguring the data logger/TSL
- updating the logger's/TSL firmware
- creating bug reports

Besides that there are several more functions for deleting data, setting the logger's/TSL time and marker, scanning the network for available loggers/TSL, etc.

#### 1.2.1 Error handling and listener mechanism

All errors are processed by the functions' return values. If the return value states an error a call to `getLastError()` provides details about the error(s) occurred. Warnings are not intended to abort

a process. That's why they are reported via the function `IBPNGClientListener::onWarning()`. It's up to the user to handle them or not.

Progress and status information is also processed via listener callbacks. You have to derive your own class from `IBPNGClientListener` and implement all functions you need. Register an object of your listener class at the executing `IBPNGClient` with `IBPNGClient::addListener()`.

## 1.3 Compiler/Linker

The library is build with Microsoft Visual C++ and is linked to the C-Runtime Library with the Multi-threaded resp. Multi-threaded Debug compiler switch (/MT resp. /MTd). The user's project must have the same settings. Applications with mixed runtime library linkage may cause errors that are difficult to diagnose and to handle. The debug version of the library is named with a "\_d" suffix.

## 1.4 Thread safety

The library is thread safe when using different objects of `IBPNGClient` resp. the objects' pointers in different threads. It is NOT thread safe for one `IBPNGClient` instance in several threads!

## 1.5 Demo project

The "sample" directory contains a demo project for the Telemotive Client Library.

Exampe for lib unsage:

```
//*****
//
// main.cc
//
//*****

#include <sys/stat.h>
#include <cerrno>
#include <ctime>
#include <iostream>
#include <direct.h>
#include <fstream>
#include <sstream>
#include <map>

#include "windirent.h"

#include "BPNGDefines.h"

#include "IBPNGClientListener.h"
#include <IBPNGClient.h>

#include "BPNGLoggerDetector.hh"
#include "RdbEventList.hh"

#ifdef _MSC_VER
#include <sys/stat.h>
#else
#include <direct.h>
#define mkdir(a, b) _mkdir(a)
#endif

using namespace std;
```

```

/*****ONLINE*DOWNLOAD*****/
static void sampleFunctionDownload(BPNGDevice device);
static void sampleFunctionTSLDownload(TSLCluster tsl);
/*****ONLINE*CONVERSION*****/
static void sampleFunctionOnlineConversion(BPNGDevice device);
static void sampleFunctionTSLOnlineConversion(TSLCluster tsl);
/*****CONFIG*****/
static void sampleFunctionConfiguration(BPNGDevice device);
static void sampleFunctionTSLConfiguration(TSLCluster tsl);
/*****OFFLINE*CONVERSION*****/
static void sampleFunctionOfflineConversion();
static void sampleFunctionOfflineTSLConversion();
/*****HELPERS*****/
static string getLocalDateString();
static vector<string> readZipsFromDirectory(string dir);

int main()
{
    // Get list of all currently available blue PiraT 2 devices
    BPNGLoggerDetector detector;
    vector<BPNGDevice> devices = detector.getLoggerList(0);
    vector<TSLCluster> tsls = detector.getTSLs(devices);

    // select the device you want to work with
    BPNGDevice device;
    TSLCluster targetTsl;
    bool found = false;
    bool tslFound = false;

    //map for tsl's

    for (size_t i = 0; i < devices.size(); ++i)
    {
        if (devices[i].ipAddress == "192.168.0.233")
        {
            device = devices[i];
            found = true;
            break;
        }
    }

    for (size_t i = 0; i < tsls.size(); ++i)
    {
        TSLCluster tsl = tsls[i];
        cout << "Found TSL [" << tsl.getTSLName() << "] with devices\n";

        tsl.print();

        cout << "\n";

        /* insert target tsl name here*/
        if (tsl.getTSLName() == "emaNLSToN")
        {
            targetTsl = tsl;
            tslFound = true;
        }
    }

    /* activate one of the sample functions */
    if (found)
    {
        //sampleFunctionDownload(device);
        //sampleFunctionOnlineConversion(device);
        //sampleFunctionConfiguration(device);
    }

    if (tslFound)
    {
        //sampleFunctionTSLConfiguration(targetTsl);
        //sampleFunctionTSLDownload(targetTsl);
        //sampleFunctionTSLOnlineConversion(targetTsl);
    }
}

```

```

    /* offline samples*/
    //sampleFunctionOfflineConversion();
    //sampleFunctionOfflineTSLConversion();

    system("pause");
}

/*****
*****ONLINE*DOWNLOAD*****
*****/

// We want to download all traces since last startup to an offline data set
static void sampleFunctionDownload(BPNGDevice device)
{
    IBPNGClient *client = getBPNGClient();

    // connect logger
    BOOL ret = client->connectLogger(device.ipAddress.c_str());
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to connect logger. " << endl;
        cout << "BPNGErrorCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    ret = client->initOnline();
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to init online." << endl;
        cout << "BPNGErrorCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    IRdbEventList *list = client->getEventList();
    RdbEventList eventList(list);

    if (eventList.size() == 0)
    {
        cout << "Empty event list" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    uint64_t startupId = 0;
    uint64_t endId = -1; //max value for uint64 to include everything in the id range

    // search last startup
    for (int i = eventList.size() - 1; i >= 0; --i)
    {
        if (eventList[i].type == STARTUP)
        {
            startupId = eventList[i].uniqueID;
            break;
        }
    }

    DataSpan span;
    span.type = DST_IDSPAN;
    span.start = startupId;
    span.end = endId;

    // if you want to download several spans, put them in a vector
    vector<DataSpan> spanVec;
    spanVec.push_back(span);

    ret = mkdir("../testoutdir", 0x777);
    if (ret != 0 && errno != EEXIST)
    {
        cout << "Failed to create output directory" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    ret = client->downloadDataSpans(spanVec.size(), &spanVec[0], "../testoutdir\\

```

```

    BP2_Offline.zip", 0);
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to download data." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// disconnect
client->disconnectLogger();
// free memory
client->release();
}

// We want to download all traces since last startup to an offline data set
static void sampleFunctionTSLDownload(TSLCluster tsl)
{
    IBPNGClient *client = getTSLClient(tsl.getNumMembers(
        TSLCluster::DOWNLOAD));

    // connect logger
    BOOL ret = client->connectLogger(tsl.getConnectionString(
        TSLCluster::DOWNLOAD).c_str());
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to connect tsl. " << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    ret = client->initOnline();
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to init online." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    IRdbEventList *list = client->getEventList();
    RdbEventList eventList(list);

    if (eventList.size() == 0)
    {
        cout << "Empty event list" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    uint64_t startupId = 0;
    uint64_t endId = -1; //max value for uint64 to include everything in the id range

    // search last startup
    for (int i = eventList.size() - 1; i >= 0; --i)
    {
        if (eventList[i].type == STARTUP)
        {
            startupId = eventList[i].uniqueID;
            break;
        }
    }

    DataSpan span;
    span.type = DST_IDSPAN;
    span.start = startupId;
    span.end = endId;

    // if you want to download several spans, put them in a vector
    vector<DataSpan> spanVec;
    spanVec.push_back(span);

    ret = mkdir("../testoutdir", 0x777);

```



```

    if (ret != 0 && errno != EEXIST)
    {
        cout << "Failed to create output directory" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    ret = client->downloadDataSpans(spanVec.size(), &spanVec[0], "..\\testoutdir\\
        TSL_Offline.zip", 0);
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to download data." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    // disconnect
    client->disconnectLogger();
    // free memory
    client->release();
}

/*****ONLINE*CONVERSION*****/
// We want to convert all CAN traces from the logger
// around the last Marker to CANoe asc and BLF format.
static void sampleFunctionOnlineConversion(BPNGDevice device)
{
    IBPNGClient *client = getBPNGClient();

    // connect logger
    BOOL ret = client->connectLogger(device.ipAddress.c_str());
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to connect logger." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    ret = client->initOnline();
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to init online." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    IRdbEventList* list = client->getEventList();
    RdbEventList eventList(list);
    if (eventList.size() == 0)
    {
        cout << "Empty event list" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    uint64_t markerTimeStamp = 0;
    // search last marker
    for (int i = eventList.size() - 1; i >= 0; --i)
    {
        if (eventList[i].type == MARKER)
        {
            markerTimeStamp = eventList[i].timeStamp;
            break;
        }
    }

    if (markerTimeStamp == 0)
    {

```

```

        cout << "No marker found." << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    // Ensure the out directory exists
    ret = mkdir("../testoutdir", 0x777);
    if (ret != 0 && errno != EEXIST)
    {
        cout << "Failed to create output directory" << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    // Get a conversion set
    IConversionSet* conversionSet = client->createNewConversionSet();

    // The time span has to be 60s before and 60s after the marker
    uint64_t startTime = markerTimeStamp - 60 * 1000000; // in usec
    uint64_t endTime = markerTimeStamp + 60 * 1000000; // in usec

    // If you want to convert more than one span,
    // call this function several times
    conversionSet->addTimeSpan(startTime, endTime);

    // CAN #1 and CAN #2 are supposed to be written to one asc output file each.
    // CAN #3 and CAN #4 are supposed to be written together in another asc file.
    // All other CAN channels are supposed to be written together in one BLF file.
    const IChannelList* channels = client->getLoggerChannels();
    for (int i = 0; i < channels->getSize(); ++i)
    {
        ChannelType type = channels->getChannel(i)->getType();
        if (type != CH_CAN)
            continue;

        // Note: channel indices are zero based
        int index = channels->getChannel(i)->getIndex();
        if (index == 0 || index == 1)
        {
            // CAN #1 and #2 in separate files
            // -1 as fileId parameter creates a separate file for this channel
            conversionSet->addChannel(type, index, CANOE, -1);
        }
        else if (index == 2 || index == 3)
        {
            // CAN #3 and #4 in the same file.
            // fileId != -1 will write all channels with the same format and same
            // file Id to the same output file (if procurable in accordance with
            // the format specification.
            conversionSet->addChannel(type, index, CANOE, 10);
        }
        else
        {
            // All other CAN channels to one BLF file.
            conversionSet->addChannel(type, index, BLF, 20);
        }
    }

    ret = client->convertData(conversionSet, "../testoutdir");
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to convert data." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }

    // disconnect
    client->disconnectLogger();
    // free memory
    client->release();
}

static void sampleFunctionTSLOnlineConversion(TSLCluster tsl)
{

```

```

IBPNGClient *client = getTSLClient(tsl.getNumMembers(
    TSLCluster::CONVERSION));

// connect logger
BOOL ret = client->connectLogger(tsl.getConnectionString(
    TSLCluster::CONVERSION).c_str());
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to connect tsl." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->release();
    return;
}

ret = client->initOnline();
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to init online." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

IRdbEventList* list = client->getEventList();
RdbEventList eventList(list);
if (eventList.size() == 0)
{
    cout << "Empty event list" << endl;
    client->disconnectLogger();
    client->release();
    return;
}

uint64_t markerTimeStamp = 0;
// search last marker
for (int i = eventList.size() - 1; i >= 0; --i)
{
    if (eventList[i].type == MARKER)
    {
        markerTimeStamp = eventList[i].timestamp;
        break;
    }
}

if (markerTimeStamp == 0)
{
    cout << "No marker found." << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// Ensure the out directory exists
ret = mkdir("../testoutdir", 0x777);
if (ret != 0 && errno != EEXIST)
{
    cout << "Failed to create output directory" << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// Get a conversion set
IConversionSet* conversionSet = client->createNewConversionSet();

// The time span has to be 60s before and 60s after the marker
uint64_t startTime = markerTimeStamp - 60 * 1000000; // in usec
uint64_t endTime = markerTimeStamp + 60 * 1000000; // in usec

// If you want to convert more than one span,
// call this function several times
conversionSet->addTimeSpan(startTime, endTime);

// CAN #1 and CAN #2 are supposed to be written to one asc output file each.
// CAN #3 and CAN #4 are supposed to be written together in another asc file.
// All other CAN channels are supposed to be written together in one BLF file.

```

```

// On TSL we have to add offset and mainboardnumber for channel identification.
const IChannelList* channels = client->getLoggerChannels();
for (int i = 0; i < channels->getSize(); ++i)
{
    ChannelType type = channels->getChannel(i)->getType();
    if (type != CH_CAN)
        continue;

    // Note: channel indices are zero based
    int index = channels->getChannel(i)->getIndex();
    if (index == 0 || index == 1)
    {
        // CAN #1 and #2 in separate files
        // -1 as fileId parameter creates a separate file for this channel
        conversionSet->addChannel(type, index, CANOE, -1, channels->
getChannel(i)->getOffset(), channels->getChannel(i)->
getMainboardNumber());
    }
    else if (index == 2 || index == 3)
    {
        // CAN #3 and #4 in the same file.
        // fileId != -1 will write all channels with the same format and same
        // file Id to the same output file (if procurable in accordance with
        // the format specification.
        conversionSet->addChannel(type, index, CANOE, 10, channels->
getChannel(i)->getOffset(), channels->getChannel(i)->
getMainboardNumber());
    }
    else
    {
        // All other CAN channels to one BLF file.
        conversionSet->addChannel(type, index, BLF, 20, channels->
getChannel(i)->getOffset(), channels->getChannel(i)->
getMainboardNumber());
    }
}

ret = client->convertData(conversionSet, "..\\testoutdir");
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to convert data." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// disconnect
client->disconnectLogger();
// free memory
client->release();
}

/*****CONFIG*****/

// This function shows how to:
// - download the configuration
// - reconfigure the logger device
// - set the default config
static void sampleFunctionConfiguration(BPNGDevice device)
{
    IBPNGClient *client = getBPNGClient();

    BOOL ret = client->connectLogger(device.ipAddress.c_str());
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to connect logger." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    // save current config
    string targetPath = "..\\testoutdir\\bpng_[" + device.mainboardNr + "].[" +
getLocalDateString() + "].zip";
    ret = client->getConfig(targetPath.c_str());
    if (ret == 0)

```

```

{
    BPNSError err = client->getLastError();
    cout << "Failed to download configuration." << endl;
    cout << "BPNSErrorCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// here you could change the downloaded configuration
// by extracting it and modifying the xml files
// see documentation of IBPNGClient::getConfig()
// or IBPNGClient::reconfigLogger().
// the new config archive needs a date in its filename in this form: YYYY-MM-DD_HH-MM-SS

// We use the same config that we downloaded
string newConfigPath = targetPath;
ret = client->reconfigLogger(newConfigPath.c_str());
if (ret == 0)
{
    BPNSError err = client->getLastError();
    cout << "Failed to reconfigure the logger." << endl;
    cout << "BPNSErrorCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// Setting the default config
ret = client->setDefaultConfig();
if (ret == 0)
{
    BPNSError err = client->getLastError();
    cout << "Failed to set default config to the logger." << endl;
    cout << "BPNSErrorCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// disconnect
client->disconnectLogger();
// free memory
client->release();
}

// This function shows how to:
// - download the configuration
// - reconfigure the logger device
// - set the default config
static void sampleFunctionTSLConfiguration(TSLCluster tsl)
{
    //get the tsl client instance
    IBPNGClient *client = getTSLClient(tsl.getNumMembers(
        TSLCluster::CONFIG));

    BOOL ret = client->connectLogger(tsl.getConnectionString(
        TSLCluster::CONFIG).c_str());
    if (ret == 0)
    {
        BPNSError err = client->getLastError();
        cout << "Failed to connect TSL." << endl;
        cout << "BPNSErrorCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    // save current config
    string targetPath = "..\\testoutdir\\tsl_[" + tsl.getTSLName() + "][" + getLocalDateString()
        + "].";
    ret = client->getConfig(targetPath.c_str());
    if (ret == 0)
    {
        BPNSError err = client->getLastError();
        cout << "Failed to download configurations." << endl;
        cout << "BPNSErrorCode: " << err.code << ", " << err.msg << endl;
        client->disconnectLogger();
        client->release();
        return;
    }
}

```

```

// here you could change the downloaded configuration
// by extracting it and modifying the xml files
// see documentation of IBPNGClient::getConfig()
// or IBPNGClient::reconfigLogger().
// the new config archive needs a date in its filename in this form: YYYY-MM-DD_HH-MM-SS

// We use the same config that we downloaded
// for tsl we have to create one string which includes all config-ip pairs
vector<string> configZips = readZipsFromDirectory(targetPath);

// prepare string <configPath:1>|<ipAddress:1>;<configPath:2>|<ipAddress:2>;...
string newConfigPath = "";

for (vector<BPNGDevice>::iterator iter = tsl.begin(); iter != tsl.end(); ++iter)
{
    BPNGDevice device = *iter;
    for (size_t i = 0; i = configZips.size(); ++i)
    {
        string configZip = configZips[i];
        if (configZip.find(device.mainboardNr) != string::npos)
        {
            newConfigPath += targetPath + "\\\" + configZip + "|" + device.
            ipAddress + ";";
            break;
        }
    }
}

ret = client->reconfigLogger(newConfigPath.c_str());
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to reconfigure the logger." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

ret = client->setDefaultConfig();
if (ret == 0)
{
    BPNGError err = client->getLastError();
    cout << "Failed to set default config to the logger." << endl;
    cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
    client->disconnectLogger();
    client->release();
    return;
}

// disconnect
client->disconnectLogger();
// free memory
client->release();
}

/*****OFFLINE*CONVERSION*****/

// We want to convert all CAN traces from an Offline data set
// around the last Marker to CANoe asc and BLF format.
static void sampleFunctionOfflineConversion()
{
    IBPNGClient *client = getBPNGClient();

    // We use the sample Offline Data Set that was downloaded
    // with sampleFunctionDownload().
    // Its up to you to ensure an existing file.
    BOOL ret = client->initOffline("../testoutdir\BP2_Offline.zip");
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to init offline." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }
}

IRdbEventList* list = client->getEventList();

```

```

RdbEventList eventList(list);
if (eventList.size() == 0)
{
    cout << "Empty event list" << endl;
    client->release();
    return;
}

uint64_t markerTimeStamp = 0;
// search last marker
for (int i = eventList.size() - 1; i >= 0; --i)
{
    if (eventList[i].type == MARKER)
    {
        markerTimeStamp = eventList[i].timeStamp;
        break;
    }
}

if (markerTimeStamp == 0)
{
    cout << "No marker found." << endl;
    client->release();
    return;
}

// Ensure the out directory exists
ret = mkdir("../testoutdir", 0x777);
if (ret != 0 && errno != EEXIST)
{
    cout << "Failed to create output directory" << endl;
    client->release();
    return;
}

// Get a conversion set
IConversionSet* conversionSet = client->createNewConversionSet();

// The time span has to be 60s before and 60s after the marker
uint64_t startTime = markerTimeStamp - 60 * 1000000; // in usec
uint64_t endTime = markerTimeStamp + 60 * 1000000; // in usec

// If you want to convert more than one span,
// call this function several times
conversionSet->addTimeSpan(startTime, endTime);

// CAN #1 and CAN #2 are supposed to be written to one asc output file each.
// CAN #3 and CAN #4 are supposed to be written together in another asc file.
// All other CAN channels are supposed to be written together in one BLF file.
const IChannelList* channels = client->getLoggerChannels();
for (int i = 0; i < channels->getSize(); ++i)
{
    ChannelType type = channels->getChannel(i)->getType();
    if (type != CH_CAN)
        continue;

    // Note: channel indices are zero based
    int index = channels->getChannel(i)->getIndex();
    if (index == 0 || index == 1)
    {
        // CAN #1 and #2 in separate files
        // -1 as fileId parameter creates a separate file for this channel
        conversionSet->addChannel(type, index, CANOE, -1);
    }
    else if (index == 2 || index == 3)
    {
        // CAN #3 and #4 in the same file.
        // fileId != -1 will write all channels with the same format and same
        // file Id to the same output file (if procurable in accordance with
        // the format specification.
        conversionSet->addChannel(type, index, CANOE, 10);
    }
    else
    {
        // All other CAN channels to one BLF file.
        conversionSet->addChannel(type, index, BLF, 20);
    }
}

ret = client->convertData(conversionSet, "../testoutdir");

```

```

    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to convert data." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    // free memory
    client->release();
}

// We want to convert all A/I, D/I traces from an Offline TSL data set
// to XTMT
static void sampleFunctionOfflineTSLConversion()
{
    // We use the sample Offline Data Set that was downloaded
    // with sampleFunctionTSLDownload().
    // Its up to you to ensure an existing file.
    // informations about number of tsl members can be retrieved by client lib function
    getNumTSLMemberFromOfflineDataSet
    string offlineDataSetPath = "..\\testoutdir\\TSL_Offline.zip";
    int numMembers = 0;
    getNumTSLMemberFromOfflineDataSet(offlineDataSetPath.c_str(), &
        numMembers);

    // create a tsl client: parameter is number of tsl member
    IBPNGClient *client = getTSLClient(numMembers);

    // Its up to you to ensure an existing file.
    BOOL ret = client->initOffline(offlineDataSetPath.c_str());
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to init offline." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }

    // Ensure the out directory exists
    ret = _mkdir("..\\testoutdir");
    if (ret != 0 && errno != EEXIST)
    {
        cout << "Failed to create output directory" << endl;
        client->release();
        return;
    }

    // Get a conversion set
    IConversionSet* conversionSet = client->createNewConversionSet();

    //select all spans
    conversionSet->addTimeSpan(0, 0xFFFFFFFFFFFFFFFFUL);

    // All A/I, D/I channels are supposed to be written in separated XTMT files.
    const IChannelList* channels = client->getLoggerChannels();
    for (int i = 0; i < channels->getSize(); ++i)
    {
        ChannelType type = channels->getChannel(i)->getType();
        if (type == CH_DIGITAL_IN || type == CH_ANALOG_IN)
        {
            // -1 as fileId parameter creates a separate file for this channel
            // for tsl the offset and mainboardnumber fields are needed
            conversionSet->addChannel(type, channels->getChannel(i)->
                getIndex(), XTMT, -1, channels->getChannel(i)->getOffset(), channels->
                getChannel(i)->getMainboardNumber());
        }
    }

    ret = client->convertData(conversionSet, "..\\testoutdir");
    if (ret == 0)
    {
        BPNGError err = client->getLastError();
        cout << "Failed to convert data." << endl;
        cout << "BPNGErrCode: " << err.code << ", " << err.msg << endl;
        client->release();
        return;
    }
}

```



```
// free memory
client->release();
}

/*****HELPERS*****/
static string getLocalDateString()
{
    time_t timeObj;
    time(&timeObj);
    tm *pTime = localtime(&timeObj);
    char buffer[100];
    int hour = pTime->tm_isdst ? pTime->tm_hour + 1 : pTime->tm_hour;
    sprintf(buffer, "%d-%02d-%02d-%02d-%02d", pTime->tm_year + 1900, pTime->tm_mon + 1, pTime->tm_mday,
        hour, pTime->tm_min, pTime->tm_sec);

    return buffer;
}

static vector<string> readZipsFromDirectory(string dir)
{
    vector<string> output;
    DIR* directory = NULL;
    directory = opendir(dir.c_str());
    struct dirent* entry = readdir(directory);

    string file;
    while (entry != NULL)
    {
        file = entry->d_name;

        if (file.find(".zip") != string::npos)
        {
            output.push_back(file);
        }

        entry = readdir(directory);
    }

    closedir(directory);
    return output;
}
```

## Kapitel 2

# Deprecated List

**Class CANPseudoMessagesProperties**

use *IClientProperties* interface instead

**Class ClientProperties**

use *IClientProperties* interface instead

**Class CommonProperties**

use *IClientProperties* interface instead

**Class ConversionProperties**

use *IClientProperties* interface instead

**Member DEV\_BP2**

For blue PiraT 2 devices use type *DEV\_BP2\_V1X*, for new blue PiraT 2 5E devices use *DEV\_BP2\_V2X*

**Class MOSTPseudoMessagesProperties**

use *IClientProperties* interface instead

## Kapitel 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BPNGDevice . . . . .	20
BPNGError . . . . .	21
CANPseudoMessagesProperties . . . . .	27
ClientProperties . . . . .	28
CommonProperties . . . . .	29
ConversionProperties . . . . .	29
DataSpan . . . . .	30
IBPNGClient . . . . .	30
IBPNGClientListener . . . . .	46
BPNGLoggerDetector . . . . .	21
IChannel . . . . .	51
IChannelList . . . . .	52
IClientProperties . . . . .	53
IConversionSet . . . . .	57
IFalseMeasureSignal . . . . .	59
IFalseMeasureSignalList . . . . .	60
IFormatInfo . . . . .	60
IFormatList . . . . .	61
IRdbEvent . . . . .	61
IRdbEventList . . . . .	62
IRdbTraceBlock . . . . .	62
IRdbTraceBlockList . . . . .	62
ITesttoolsChannel . . . . .	62
ITesttoolsChannelList . . . . .	63
LogInData . . . . .	63
MemoryFillLevel . . . . .	64
MOSTPseudoMessagesProperties . . . . .	64
OnlineLoggerInfo . . . . .	65
RdbEvent2 . . . . .	66
TSLCluster . . . . .	68
vector	
RdbEventList . . . . .	67

## Kapitel 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">BPNGDevice</a> . . . . .	20
<a href="#">BPNGError</a> Error struct with error code and optional error message . . . . .	21
<a href="#">BPNGLoggerDetector</a> . . . . .	21
<a href="#">CANPseudoMessagesProperties</a> CAN pseudo message properties, . . . . .	27
<a href="#">ClientProperties</a> <a href="#">ClientProperties</a> , . . . . .	28
<a href="#">CommonProperties</a> Common properties, . . . . .	29
<a href="#">ConversionProperties</a> Conversion Properties, . . . . .	29
<a href="#">DataSpan</a> . . . . .	30
<a href="#">IBPNGClient</a> Interface class for the Telemotive Client Library . . . . .	30
<a href="#">IBPNGClientListener</a> . . . . .	46
<a href="#">IChannel</a> Channel interface . . . . .	51
<a href="#">IChannelList</a> Channel list interface . . . . .	52
<a href="#">IClientProperties</a> The <a href="#">IClientProperties</a> interface replaces the deprecated <a href="#">ClientProperties</a> struct	53
<a href="#">IConversionSet</a> A conversion set stores all conversion relevant settings . . . . .	57
<a href="#">IFalseMeasureSignal</a> False measure signal interface . . . . .	59
<a href="#">IFalseMeasureSignalList</a> False measure signal list interface . . . . .	60
<a href="#">IFormatInfo</a> FormatInfo interface . . . . .	60
<a href="#">IFormatList</a> Format list interface . . . . .	61
<a href="#">IRdbEvent</a> . . . . .	61

<a href="#">IRdbEventList</a> . . . . .	62
<a href="#">IRdbTraceBlock</a> . . . . .	62
<a href="#">IRdbTraceBlockList</a> . . . . .	62
<a href="#">ITesttoolsChannel</a>	
Channel interface . . . . .	62
<a href="#">ITesttoolsChannelList</a>	
TesttoolsChannel list interface . . . . .	63
<a href="#">LoginData</a>	
Structure for login . . . . .	63
<a href="#">MemoryFillLevel</a>	
Stores memory fill level of a device . . . . .	64
<a href="#">MOSTPseudoMessagesProperties</a>	
MOST pseudo properties, . . . . .	64
<a href="#">OnlineLoggerInfo</a>	
Struct with information about a logger found in LAN . . . . .	65
<a href="#">RdbEvent2</a>	
Implementation class for a wrapper of <a href="#">IRdbEvent</a> using STL classes . . . . .	66
<a href="#">RdbEventList</a>	
Implementation class for a wrapper of <a href="#">IRdbEventList</a> using STL classes . . . . .	67
<a href="#">TSLCluster</a> . . . . .	68

## Kapitel 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">BPNGDefines.h</a>	Defines for Telemotive Client Library . . . . .	72
<a href="#">BPNGLoggerDetector.hh</a>	Logger Detector Sample . . . . .	81
<a href="#">IBPNGClient.h</a>	Interface class for the BPNGClient DLL . . . . .	82
<a href="#">IBPNGClientListener.h</a>	Interface class for the BPNGClient listener . . . . .	84
<a href="#">IClientProperties.h</a>	Interface for client properties . . . . .	85
<a href="#">RdbDefines.h</a>	Public interfaces for Telemotive Reference Database access . . . . .	85
<a href="#">RdbEventList.hh</a>	<a href="#">IRdbEvent</a> wrapper . . . . .	86

## Kapitel 6

# Class Documentation

### 6.1 BPNGDevice Struct Reference

```
#include <BPNGLoggerDetector.hh>
```

#### Public Member Functions

- **BPNGDevice** (const [OnlineLoggerInfo](#) \*logger)

#### Public Attributes

- std::string [ipAddress](#)  
*the device ip*
- std::string [name](#)  
*current name of device*
- std::string [mainboardNr](#)  
*mainboard number of device*
- bool [connected](#)  
*user connected state*
- std::string [connectedUser](#)  
*the name of the connected user*
- unsigned [status](#)  
*the device state*
- int [tslNetworkId](#)  
*the tsl network id, all devices with same id belong to one TSL, -1 means no TSL*
- std::string [tslName](#)  
*the tsl network name*
- [BPNGDeviceType](#) [deviceType](#)  
*the device product type*

#### 6.1.1 Detailed Description

This struct matches the [OnlineLoggerInfo](#) but uses STL strings.

### 6.1.2 Member Data Documentation

#### 6.1.2.1 unsigned BPNGDevice::status

the device state

See Also

[BPNGLoggerStatus](#)

The documentation for this struct was generated from the following file:

- [BPNGLoggerDetector.hh](#)

## 6.2 BPNGError Struct Reference

Error struct with error code and optional error message.

```
#include <BPNGDefines.h>
```

### Public Attributes

- [BPNGErrCode](#) *code*  
*error code*
- const char \* [msg](#)  
*error message*

### 6.2.1 Detailed Description

Error struct with error code and optional error message.

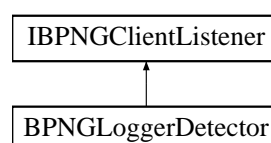
The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.3 BPNGLoggerDetector Class Reference

```
#include <BPNGLoggerDetector.hh>
```

Inheritance diagram for BPNGLoggerDetector:





## Public Member Functions

- [BPNGLoggerDetector](#) ()
- `std::vector< BPNGDevice > getLoggerList` (unsigned searchTimeOut)
- `std::vector< TSLCluster > getTSLs` (`std::vector< BPNGDevice > loggersInNetwork`)
- virtual void WINAPI [onBPNGDeviceDetected](#) ([OnlineLoggerInfo](#) \*info)  
*Called to notify a detected logger in network.*
- virtual void WINAPI [onBPNGDeviceDisappeared](#) ([OnlineLoggerInfo](#) \*info)  
*Called to notify a disappeared logger.*
- virtual void WINAPI [onBPNGDeviceStateChange](#) ([OnlineLoggerInfo](#) \*info)  
*Called to notify a logger's state change.*
- virtual int WINAPI [onProgressDataDownload](#) (int percentCompleted)  
*Called to indicate the current progress of a file transfer.*
- virtual int WINAPI [onProgressConversion](#) (int percentCompleted, const char \*status)  
*Called to indicate the current progress of file conversion.*
- virtual void WINAPI [onStatusMessage](#) (const char \*statusMsg)  
*Called to send additional information of the current process to the calling app.*
- virtual void WINAPI [onWarning](#) ([BPNGWarningCode](#) warningCode, const char \*warnMsg)  
*Called to inform about a warning.*
- virtual int WINAPI [onTargetPathTooLong](#) (char \*newTarget, int maxSize)  
*Called on a too long target directory.*
- virtual int WINAPI [getOverwritingPermission](#) (const char \*filePath)  
*Called on existing output trace files.*
- virtual const char \*WINAPI [onLogInDataRequired](#) (const char \*ipAddress)  
*Called on accessing password protected functions.*
- virtual void WINAPI [onInvalidPwConfigFound](#) (const char \*ipAddress)  
*Called if invalid pw file found on device.*
- virtual void WINAPI [onLogInDataFailed](#) ()
- virtual void WINAPI [onResetLogInDataFailed](#) ()
- virtual void WINAPI [onFuncAccessDenied](#) ()
- virtual int WINAPI [onCriticalDiskSpace](#) (uint64\_t freeSpace, uint64\_t neededSpace, const char \*drive)
- virtual void WINAPI [onFirmwareUpdateProgress](#) (int percentage, int stepId, int subStepId, const char \*desc)  
*Called on firmware update progress.*
- virtual void WINAPI [onFirmwareUpdateError](#) (int errorId)
- virtual int WINAPI [onGetLogReportProgress](#) (int percentage, const char \*desc)
- virtual int WINAPI [onCriticalDiskSpace](#) (uint64\_t freeSpace, uint64\_t neededSpace, const char \*drive, const char \*msg)  
*Called in case of not enough free disk space.*
- virtual void WINAPI [onDownloadStart](#) (int64\_t totalAmountOfBytes)  
*Notifies the listeners before the download starts about the total amount of bytes to be downloaded.*
- virtual void WINAPI [onConversionStart](#) (int64\_t totalAmountOfBytes)  
*Notifies the listeners before the conversion starts about the total amount of bytes to be converted.*

### 6.3.1 Detailed Description

A simple minimal implementation for a logger detection. Uses an [IBPNGClient](#) instance and the method [IBPNGClient::scanNetworkForLogger\(\)](#). Sets itself as [IBPNGClientListener](#) to the [IBPNGClient](#) instance ([IBPNGClient::addListener\(IBPNGClientListener\\* listener\)](#)). The callbacks [onBPNGDeviceDetected\(OnlineLoggerInfo \\*info\)](#), [onBPNGDeviceDisappeared\(OnlineLoggerInfo \\*info\)](#) and [onBPNGDeviceStateChange\(OnlineLoggerInfo \\*info\)](#) informs the class about the device states in network.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 BPNGLoggerDetector::BPNGLoggerDetector ( ) [inline]

Constructor

### 6.3.3 Member Function Documentation

#### 6.3.3.1 std::vector<BPNGDevice> BPNGLoggerDetector::getLoggerList ( unsigned searchTimeout )

Returns a vector of detected [BPNGDevice](#) in local networks.

##### Parameters

<i>searchTime-Out</i>	the search timeout in seconds
-----------------------	-------------------------------

##### Returns

vector of [BPNGDevice](#)

#### 6.3.3.2 virtual int WINAPI BPNGLoggerDetector::getOverwritingPermission ( const char \* filePath ) [inline], [virtual]

Called on existing output trace files.

When an output trace file already exists this function is called. The listener has the possibility to return one of following values: -1: no, don't overwrite file -2: no, overwrite neither this nor any following file 1: yes, overwrite file 2: yes, overwrite this and all following files 0: cancel conversion

Implements [IBPNGClientListener](#).

#### 6.3.3.3 std::vector<TSLCluster> BPNGLoggerDetector::getTSLs ( std::vector< BPNGDevice > loggersInNetwork )

Checks a vector of [BPNGDevice](#) for TSL chains. Combines the devices with same tslNetworkId (except -1) to [TSLCluster](#).

##### Parameters

<i>loggersIn-Network</i>	the <a href="#">BPNGDevice</a> in network found by <a href="#">getLoggerList(unsigned searchTimeout)</a>
--------------------------	--

**Returns**

vector of [TSLCluster](#)

**6.3.3.4** virtual void WINAPI BPNGLoggerDetector::onBPNGDeviceDetected ( [OnlineLoggerInfo](#) \* *info* )  
[virtual]

Called to notify a detected logger in network.

All char\* of the passed [OnlineLoggerInfo](#)\* are only valid for the time of the function call. Please ensure to copy the string values.

Implements [IBPNGClientListener](#).

**6.3.3.5** virtual void WINAPI BPNGLoggerDetector::onBPNGDeviceDisappeared ( [OnlineLoggerInfo](#) \* *info* ) [virtual]

Called to notify a disappeared logger.

All char\* of the passed [OnlineLoggerInfo](#)\* are only valid for the time of the function call. Please ensure to copy the string values.

Implements [IBPNGClientListener](#).

**6.3.3.6** virtual void WINAPI BPNGLoggerDetector::onBPNGDeviceStateChange ( [OnlineLoggerInfo](#) \* *info* ) [virtual]

Called to notify a logger's state change.

All char\* of the passed [OnlineLoggerInfo](#)\* are only valid for the time of the function call. Please ensure to copy the string values.

Implements [IBPNGClientListener](#).

**6.3.3.7** virtual void WINAPI BPNGLoggerDetector::onConversionStart ( int64\_t *totalAmountOfBytes* )  
[inline], [virtual]

Notifies the listeners before the conversion starts about the total amount of bytes to be converted.

**Parameters**

<i>totalAmount-OfBytes</i>	Total data size to be converted
----------------------------	---------------------------------

Implements [IBPNGClientListener](#).

**6.3.3.8** virtual int WINAPI BPNGLoggerDetector::onCriticalDiskSpace ( uint64\_t *freeSpace*, uint64\_t *neededSpace*, const char \* *drive*, const char \* *msg* ) [inline], [virtual]

Called in case of not enough free disk space.

This notifies the listener about not enough free disk space for data download or conversion.

The user can continue or abort the process. Returning 0 will abort the process. In some cases continuing without providing more disk space will call this function immediately again.

#### Parameters

<i>freeSpace</i>	Amount of free space
<i>neededSpace</i>	Amount of needed space
<i>drive</i>	Name of the drive where to store data
<i>msg</i>	Additional message to display

#### Returns

return 0 when process should be aborted, 1 to ignore

Implements [IBPNGClientListener](#).

**6.3.3.9** virtual void WINAPI BPNGLoggerDetector::onDownloadStart ( int64\_t *totalAmountOfBytes* )  
[inline], [virtual]

Notifies the listeners before the download starts about the total amount of bytes to be downloaded.

#### Parameters

<i>totalAmount-OfBytes</i>	Total data size to be downloaded
----------------------------	----------------------------------

Implements [IBPNGClientListener](#).

**6.3.3.10** virtual int WINAPI BPNGLoggerDetector::onGetLogReportProgress ( int *percentage*, const char \* *desc* ) [inline], [virtual]

Called on creation of log report

#### Returns

return value 0 indicates an abort request from the implementing class

Implements [IBPNGClientListener](#).

**6.3.3.11** virtual void WINAPI BPNGLoggerDetector::onInvalidPwConfigFound ( const char \* *ipAddress* )  
[inline], [virtual]

Called if invalid pw file found on device.

An error may occurred on transferring the passwordconfiguration to the device, as a result the passwordconfiguration is invalid and needs to be reset to default. Inform the user.

Implements [IBPNGClientListener](#).

**6.3.3.12** `virtual const char* WINAPI BPNGLoggerDetector::onLoginDataRequired ( const char * ipAddress ) [inline], [virtual]`

Called on accessing password protected functions.

When password protected functions are called this listener function queries for login parameters that must be returned from the implementing class.

#### Parameters

<i>ipAddress</i>	IP address of the password protected device
------------------	---

Implements [IBPNGClientListener](#).

**6.3.3.13** `virtual int WINAPI BPNGLoggerDetector::onProgressConversion ( int percentCompleted, const char * status ) [inline], [virtual]`

Called to indicate the current progress of file conversion.

This function notifies the listener about the conversion progress of the raw Telemotive trace data. If the *percentCompleted* value has changed, but the *status* is still the same, the application passes an empty string as status to the function.

#### Parameters

<i>percent-Completed</i>	Percent of the entire conversion process (from 0...100%), -1 indicates the same value as from last function call
<i>status</i>	Status of the conversion process (e.g. "Converting trace data. Block 5 of 32")

#### Returns

return value 0 indicates an abort request from the implementing class

Implements [IBPNGClientListener](#).

**6.3.3.14** `virtual int WINAPI BPNGLoggerDetector::onProgressDataDownload ( int percentCompleted ) [inline], [virtual]`

Called to indicate the current progress of a file transfer.

This function notifies the listener about the download progress of the raw Telemotive trace data.

#### Parameters

<i>percent-Completed</i>	Percentage of the entire download process (from 0...100%). A negative value can be passed if only the abort request is checked. A negative value of -1 indicates a broken ftp connection.
--------------------------	---

#### Returns

return value 0 indicates an abort request from the implementing class

Implements [IBPNGClientListener](#).

**6.3.3.15** virtual void WINAPI BPNGLoggerDetector::onStatusMessage ( const char \* *statusMsg* )  
[inline], [virtual]

Called to send additional information of the current process to the calling app.

This function transmit message strings to the listener class. Those messages are only for information purpose. The receiver doesn't have to react on it but can display it on the screen.

Implements [IBPNGClientListener](#).

**6.3.3.16** virtual int WINAPI BPNGLoggerDetector::onTargetPathTooLong ( char \* *newTarget*, int *maxSize* )  
[inline], [virtual]

Called on a too long target directory.

Called when the resulting file name of the converted files or the files of an offline data set is longer than the maximum allowed size of the file system (Windows 260). The lib user has to pass a new (shorter) base target directory to the passed char array with strcpy. The memory of the array is already allocated by the library and it's size is maxSize. When a new directory was set the value 1 must be returned. Returning another value than 1 will abort the current process with an error result.

Implements [IBPNGClientListener](#).

**6.3.3.17** virtual void WINAPI BPNGLoggerDetector::onWarning ( BPNGWarningCode *warningCode*,  
const char \* *warnMsg* ) [inline], [virtual]

Called to inform about a warning.

This function transmit a warning message to the listener class. Warnings have a WARNING\_CODE and a warning message. Warnings do not interrupt the current process but should be noticed from the user to possibly initiate further provisions.

Implements [IBPNGClientListener](#).

The documentation for this class was generated from the following file:

- [BPNGLoggerDetector.hh](#)

## 6.4 CANPseudoMessagesProperties Struct Reference

CAN pseudo message properties,.

```
#include <BPNGDefines.h>
```

### Public Attributes

- uint8\_t **canTimeStampMessage**
- uint32\_t **channelTS**
- uint32\_t **dlcTS**
- uint32\_t **canIdTS**
- uint32\_t **hourBitPos**

- uint32\_t **minuteBitPos**
- uint32\_t **secondBitPos**
- uint32\_t **dayBitPos**
- uint32\_t **monthBitPos**
- uint32\_t **yearBitPos**
- uint8\_t **canTriggerMessage**
- uint32\_t **channelTrigger**
- uint32\_t **dlcTrigger**
- uint32\_t **canIdTrigger**
- uint32\_t **triggerNumBitPos**
- uint32\_t **reserved1**
- uint32\_t **reserved2**
- uint32\_t **reserved3**
- uint32\_t **reserved4**

#### 6.4.1 Detailed Description

CAN pseudo message properties,.

**Deprecated** use [IClientProperties](#) interface instead

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.5 ClientProperties Struct Reference

[ClientProperties](#),.

```
#include <BPNGDefines.h>
```

### Public Attributes

- [CommonProperties](#) **common**
- [CANPseudoMessagesProperties](#) **canPseudoMessages**
- [MOSTPseudoMessagesProperties](#) **mostPseudoMessages**
- [ConversionProperties](#) **conversionProp**

#### 6.5.1 Detailed Description

[ClientProperties](#),.

**Deprecated** use [IClientProperties](#) interface instead

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.6 CommonProperties Struct Reference

Common properties,.

```
#include <BPNGDefines.h>
```

### Public Attributes

- char \* **nameOfTester**
- uint32\_t **maxOutputSizeMB**
- uint8\_t **separatedTimeFormat**
- uint8\_t **separatedTimeFormatInOfflineSet**
- uint32\_t **maxOutputSizeMBSortedDownload**
- char \* **alternativeLoggerName**
- uint8\_t **useAlternativeLoggerName**
- uint8\_t **useSubdirectories**
- uint8\_t **midnightSplitting**
- uint8\_t **fileTimeSpansLikeSelection**
- uint8\_t **markerNumbersInFileNames**
- uint8\_t **subfolderWithLoggerName**
- uint32\_t **reserved1**
- uint32\_t **reserved2**
- uint32\_t **reserved3**
- uint32\_t **reserved4**

### 6.6.1 Detailed Description

Common properties,.

**Deprecated** use [\*IClientProperties\*](#) interface instead

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.7 ConversionProperties Struct Reference

Conversion Properties,.

```
#include <BPNGDefines.h>
```

### Public Attributes

- uint8\_t **useSatelliteTimeForGPSFormats**
- char \* **isochronalMost150Channels**



### 6.7.1 Detailed Description

Conversion Properties,.

**Deprecated** use [IClientProperties](#) interface instead

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.8 DataSpan Struct Reference

### Public Attributes

- `uint8_t` [type](#)  
*set type to 0 for a id based range, set type to 1 for a time based range*
- `uint64_t` [start](#)  
*start time/id of data span*
- `uint64_t` [end](#)  
*end time/id of data span*

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.9 IBPNGClient Struct Reference

Interface class for the Telemotive Client Library.

```
#include <IBPNGClient.h>
```

### Public Member Functions

- virtual void WINAPI [scanNetworkForLogger](#) ()=0  
*Scan network for logger.*
- virtual BOOL WINAPI [connectLogger](#) (const char \*ipAddress)=0  
*Connect to logger with passed IP address.*
- virtual void WINAPI [disconnectLogger](#) ()=0  
*Disconnect the currently connected logger.*
- virtual BOOL WINAPI [isConnected](#) ()=0  
*Check the logger connection, returns 1 for valid connection and 0 for no or broken connection.*
- virtual BOOL WINAPI [initOnline](#) ()=0  
*Initialisation of download and online conversion process.*
- virtual BOOL WINAPI [initOffline](#) (const char \*path)=0  
*Initialisation of offline conversion process.*

- virtual int WINAPI [downloadDataSpans](#) (uint16\_t numSpans, [DataSpan](#) \*dataSpans, const char \*target, BOOL doSorting)=0  
*Download trace data.*
- virtual [IConversionSet](#) \*WINAPI [createNewConversionSet](#) ()=0  
*Returns the pointer to a new conversion set. Deprecated, use static function [createNewConversionSet](#) instead.*
- virtual int WINAPI [convertData](#) ([IConversionSet](#) \*conversionSet, const char \*target)=0  
*Convert all data specified by conversionSet.*
- virtual BOOL WINAPI [getConfig](#) (const char \*path)=0  
*Download the current logger configuration to the passed path.*
- virtual BOOL WINAPI [reconfigLogger](#) (const char \*configZip)=0  
*Reconfig logger with the zipped new configuration.*
- virtual BOOL WINAPI [setDefaultConfig](#) ()=0  
*Reconfig logger/TSL with the default configuration.*
- virtual [IRdbEventList](#) \*WINAPI [getEventList](#) ()=0  
*Get list of all events from the RDB.*
- virtual [IRdbTraceBlockList](#) \*WINAPI [getTraceBlockList](#) ()=0  
*Get list of all trace blocks from the RDB.*
- virtual BOOL WINAPI [synchronizeRdb](#) ()=0  
*Synchronizes the RDB.*
- virtual const char \*WINAPI [getInstanceName](#) ()=0  
*Return the instance name passed to the [getBPNGClient\(\)](#) function.*
- virtual int WINAPI [getInstancelId](#) ()=0  
*Returns the instance ID that is unique for all [IBPNGClient](#) instances in one process.*
- virtual const char \*WINAPI [getReferenceDataBasePath](#) ()=0  
*Get path to the reference data base.*
- virtual const char \*WINAPI [getConfigPath](#) ()=0  
*Get path to the config directory (after calling one of the init functions)*
- virtual const char \*WINAPI [getDeviceName](#) ()=0  
*Get name of device.*
- virtual const [IChannelList](#) \*WINAPI [getLoggerChannels](#) ()=0  
*Returns pointer to a channel list interface.*
- virtual const [ITesttoolsChannelList](#) \*WINAPI [getLoggerTesttoolsChannels](#) ()=0
- virtual const char \*WINAPI [getVersions](#) ()=0  
*Get the firmware and hardware version string. On TSL the result will be splitted by "<versionstring>|<ip>".*
- virtual const char \*WINAPI [getTMTVersion](#) ()=0  
*Get the current tmt version string.*
- virtual BOOL WINAPI [updateFirmware](#) (const char \*fwPath, BOOL force)=0  
*Update firmware.*
- virtual BOOL WINAPI [isUserAuthenticated](#) (PwdPrivilegesFuncId actionName)=0
- virtual BOOL WINAPI [updateLicenses](#) (const char \*licenseFilePath)=0  
*Update licenses.*
- virtual const char \*WINAPI [getLicenses](#) ()=0  
*Returns the license file's content as string.*
- virtual BOOL WINAPI [removeAllLicenses](#) ()=0

- Removes the current license file from the logger.*
- virtual int WINAPI [deleteData](#) (uint16\_t numSpans, [DataSpan](#) \*dataSpans)=0  
*Delete trace data.*
- virtual int WINAPI [deleteAllData](#) ()=0  
*Delete all trace data on the logger.*
- virtual BOOL WINAPI [setInfoEvent](#) (const char \*msg)=0  
*Set an info event with the passed string on the connected logger.*
- virtual BOOL WINAPI [setMarker](#) ()=0  
*Set a marker on the connected logger. Returns 0 on error.*
- virtual int WINAPI [getCurrentLoggerTime](#) ()=0  
*Returns the current loggertime in seconds since 01.01.1970 UTC.*
- virtual int WINAPI [setTime](#) (int time)=0  
*Set logger time and date to the passed UTC time stamp.*
- virtual void WINAPI [keepLoggerAlive](#) (const char \*ip)=0  
*Call this to keep logger alive.*
- virtual void WINAPI [stopKeepLoggerAlive](#) (const char \*ip)=0  
*Called to stop sending keep alive pings to the logger specified via the passed ip.*
- virtual [IFormatList](#) \*WINAPI [getAvailableFormats](#) ()=0  
*Return pointer to a format list interface. Returns null in case of error.*
- virtual void WINAPI [flashDeviceLED](#) ()=0  
*Let the connected device blink its front LEDs for identification.*
- virtual int WINAPI [createCCPXCPSeqFile](#) (const char \*xsdDir, const char \*xmlDir, bool forceFlag)=0  
*Parse CCPXCPMeasurement.xml and CCPXCPCConfiguration.xml and create CCPXCPSquence.xml.*
- virtual int WINAPI [createCCPXCPDbcFiles](#) (const char \*dbcDir, const char \*xsdDir, const char \*xmlDir)=0  
*Parse CCPXCPMeasurement.xml and CCPXCPCConfiguration.xml and create a Vector DBC file for each device.*
- virtual const [IFalseMeasureSignalList](#) \*WINAPI [getFalseMeasureSignals](#) ()=0  
*Return pointer to a false measure signal list interface.*
- virtual void WINAPI [addListener](#) ([IBPNGClientListener](#) \*listener)=0  
*Add a listener.*
- virtual void WINAPI [removeListener](#) ([IBPNGClientListener](#) \*listener)=0  
*Remove a listener.*
- virtual [BPNGError](#) WINAPI [getLastError](#) ()=0  
*Get last error code.*
- virtual int WINAPI [getNumConversionErrors](#) ()=0  
*Returns the number of errors occurred during the last conversion process.*
- virtual [BPNGError](#) WINAPI [getConversionError](#) (int index)=0  
*Returns the conversion error at index.*
- virtual int WINAPI [getNumDownloadErrors](#) ()=0  
*Returns the number of errors occurred during the last download process.*
- virtual [BPNGError](#) WINAPI [getDownloadError](#) (int index)=0  
*Returns the download error at index.*
- virtual const char \*WINAPI [getLibVersion](#) ()=0

- Returns the current client library version.*
- virtual const char \*WINAPI **getFWVersion** ()=0  
*Returns the current fw version.*
- virtual void WINAPI **release** ()=0  
*Free memory of this [IBPNGClient](#) instance.*
- virtual [ClientProperties](#) WINAPI **getProperties** ()=0  
*Returns the current properties, deprecated use [getClientProperties\(\)](#) instead.*
- virtual void WINAPI **setProperties** ([ClientProperties](#) val)=0  
*Set client properties, deprecated use [setClientProperties\(IClientProperties\\*\)](#) instead.*
- virtual [IClientProperties](#) \*WINAPI **getClientProperties** ()=0
- virtual void WINAPI **setClientProperties** ([IClientProperties](#) \*properties)=0
- virtual void WINAPI **saveProperties** (const char \*pathToIniFile)=0  
*Save properties to ini file.*
- virtual void WINAPI **loadProperties** (const char \*pathToIniFile)=0  
*Load properties from ini file.*
- virtual void WINAPI **clearDBCFileAssignments** ()=0  
*Remove all DBC file assignments.*
- virtual void WINAPI **assignDBCFile** (int channelIndexCAN, const char \*dbcFilePath)=0  
*Assign a DBC file to a CAN channel. Multiple files for one CAN channel are allowed, but double used message IDs will ignored.*
- virtual int WINAPI **resetMarkerCounter** ()=0  
*Reset marker counter.*
- virtual int WINAPI **setPwdFile** (const char \*path)=0
- virtual const char \*WINAPI **getPwdFile** ()=0
- virtual BOOL WINAPI **isPasswordProtectionSupported** ()=0
- virtual int WINAPI **downloadBugReport** (const char \*targetPath, [BPNGBugreportMode](#) mode, uint64\_t startTime, uint64\_t endTime)=0  
*Download bug report.*
- virtual int WINAPI **restartDevice** (BOOL waitForRestart)=0  
*restarts the device or TSL*
- virtual int WINAPI **shutdownDevice** ()=0  
*shut down the device or TSL*
- virtual BOOL WINAPI **getMemoryFillLevel** ([MemoryFillLevel](#) \*fillLevel)=0  
*get memory fill level of device*
- virtual BOOL WINAPI **convertFileNameTimeStampsToLocalTime** (const char \*pathToOfflineDataSet)=0  
*Converts all time stamps in an offline data set's file names to local time.*
- virtual BOOL WINAPI **filterSignals** (const char \*pathToFilterSettings, const char \*targetPath)=0  
*Signal filtering.*
- virtual BOOL WINAPI **filterSignalsFromOfflineData** (const char \*pathToOfflineDataSet, const char \*pathToFilterSettings, const char \*targetPath)=0  
*Signal filtering.*

### 6.9.1 Detailed Description

Interface class for the Telemotive Client Library.

[IBPNGClient](#) is the interface class of the blue PiraT Client library. To get access to a single blue PiraT 2, blue PiraT 2 5E, blue PiraT mini or blue PiraT Remote data logger you need a pointer to an implementing instance of the [IBPNGClient](#) interface. Use [getBPNGClient\(\)](#) to get such a pointer. This will create an implementing instance on the heap. To avoid conflict between different runtime libraries it is obligatory to release this object with its [IBPNGClient::release\(\)](#) function when not needed any more. Don't call the delete operator directly on this pointer.

To get access to a device chain combined via Telemotive System Link (TSL) you also need a pointer to an implementing instance of [IBPNGClient](#) interface. Use [getTSLClient\(int numTSLMember\)](#) to get such a pointer. It must also be released with [IBPNGClient::release\(\)](#).

### 6.9.2 Member Function Documentation

**6.9.2.1** `virtual void WINAPI IBPNGClient::assignDBCFile ( int channelIndexCAN, const char * dbcFilePath )` [pure virtual]

Assign a DBC file to a CAN channel. Multiple files for one CAN channel are allowed, but double used message IDs will ignored.

#### Parameters

<i>channelIndexCAN</i>	Zero based CAN channel index
<i>dbcFilePath</i>	Absolute path to the dbc file

**6.9.2.2** `virtual BOOL WINAPI IBPNGClient::connectLogger ( const char * ipAddress )` [pure virtual]

Connect to logger with passed IP address.

While the logger is connected, it won't go to standby mode until the last [IBPNGClient](#) instance is disconnected. If connect fails the function will return 0. On success the return value is 1. In case of failure further information can be retrieved with [getLastError\(\)](#).

**On TSL instance you have to pass all ip addresses of devices in TSL. The ip's have to be separated by semicolons ';'. For example: "192.168.0.233;192.168.0.10;192.168.0.1". The number of ip's have to match the number on [getTSLClient\(int numTSLMember\)](#) creation, else the connect function will return with a failure.**

#### Parameters

<i>ipAddress</i>	IP address of the logger that should be connected
------------------	---

#### Returns

0 on failure, 1 on success

**6.9.2.3** `virtual int WINAPI IBPNGClient::convertData ( IConversionSet * conversionSet, const char * target )` [pure virtual]

Convert all data specified by *conversionSet*.

Before data from a logger or an offline data set can be converted, [IBPNGClient::initOnline\(\)](#) resp. [IBPNGClient::initOffline\(\)](#) must have been called before.

The data specified by *conversionSet* is converted to the passed *target* directory.

Function will return 0 on failure, 1 on success and -1 on user abort. In case of failure further information can be retrieved with [getLastError\(\)](#).

If [getLastError\(\)](#) returns `BPNG_CONVERSION_ERRORS` several errors occurred. Use [getNumConversionErrors\(\)](#) and [getConversionError\(int index\)](#) for detailed information.

#### Parameters

<i>conversionSet</i>	conversion settings, see <a href="#">IConversionSet</a>
<i>target</i>	target directory for the converted trace files. Depending on the passed <a href="#">Client-Properties</a> the files may be stored in sub folders named by date

#### Returns

0 on failure, 1 on success and -1 on user abort.

**6.9.2.4** `virtual int WINAPI IBPNGClient::deleteAllData ( )` [pure virtual]

Delete all trace data on the logger.

In case of failure further information can be retrieved with [getLastError\(\)](#).

#### Returns

0 on failure, 1 on success and -1 on user abort.

**6.9.2.5** `virtual int WINAPI IBPNGClient::deleteData ( uint16_t numSpans, DataSpan * dataSpans )` [pure virtual]

Delete trace data.

Pass the size and the pointer to an array of [DataSpan](#). Each span specifies either a time span or an index span from the reference data base's entry IDs (`DataBaseEntryId`). If you want to create spans with those IDs you have to call the [initOnline\(\)](#) function first to get the current RDB file.

Function will return 0 on failure, 1 on success and -1 on user abort. In case of failure further information can be retrieved with [getLastError\(\)](#).

#### Parameters

<i>numSpans</i>	Size of the passed <a href="#">DataSpan</a> array in second parameter
<i>dataSpans</i>	Array of <a href="#">DataSpan</a> , specifying the time or ID spans that should be deleted

**Returns**

0 on failure, 1 on success and -1 on user abort.

**6.9.2.6** virtual int WINAPI IBPNGClient::downloadBugReport ( const char \* *targetPath*,  
BPNGBugreportMode *mode*, uint64\_t *startTime*, uint64\_t *endTime* ) [pure virtual]

Download bug report.

The downloaded bug report is a ZIP archive with several log data and system files for error analyzing purposes.

**Parameters**

<i>targetPath</i>	Path inclusive file name under that the bug report will be stored.
<i>mode</i>	that specifies what kind of data should be included in the report,

**See Also**

[BPNGBugreportMode](#)

**Parameters**

<i>startTime</i>	Start time for the time span of trace data that should be included (usec since 01.01.1970 UTC). Only for mode BR_FULL_ALL_TRACES and BR_FULL_TIMESPAN_TRACES
<i>endTime</i>	End time for the time span of trace data that should be included (usec since 01.01.1970 UTC). Only for mode BR_FULL_ALL_TRACES and BR_FULL_TIMESPAN_TRACES

**Returns**

0 on failure, 1 on success and -1 on user abort.

**6.9.2.7** virtual int WINAPI IBPNGClient::downloadDataSpans ( uint16\_t *numSpans*, DataSpan \*  
*dataSpans*, const char \* *target*, BOOL *doSorting* ) [pure virtual]

Download trace data.

Pass the size and the pointer to an array of [DataSpan](#). Each span specifies either a time span or an index span from the reference data base's entry IDs (DataBaseEntryId). [IBPNGClient::init-Online\(\)](#) must have been called before.

Function will return 0 on failure, 1 on success and -1 on user abort. In case of failure further information can be retrieved with [getLastError\(\)](#).

If [getLastError\(\)](#) returns BPNG\_DOWNLOAD\_ERRORS several errors occurred. Use [getNumDownloadErrors\(\)](#) and [getDownloadError\(int index\)](#) for detailed information.

## Parameters

<i>numSpans</i>	Size of the passed <a href="#">DataSpan</a> array in second parameter
<i>dataSpans</i>	Array of <a href="#">DataSpan</a> , specifying the time or ID spans that should be downloaded
<i>target</i>	Path to the target directory or ZIP file. A passed directory must be empty or not existing. A passed ZIP path must not exist.
<i>doSorting</i>	Specifies whether the traces from different logger-internal sources should be sorted to one output stream or not.

## Returns

0 on failure, 1 on success and -1 on user abort.

**6.9.2.8** virtual BOOL WINAPI IBPNGClient::filterSignals ( const char \* *pathToFilterSettings*, const char \* *targetPath* ) [pure virtual]

Signal filtering.

This function parses all data of the offline data set that was previously set via initOffline and filters signals according to the complex filter settings created with the Telemotive System Client.

## Parameters

<i>pathToFilter-Setting</i>	path to the ZIP file including the complex filter settings created with Telemotive System Client
<i>targetPath</i>	path to the target directory where the filtered data should be written to

**6.9.2.9** virtual BOOL WINAPI IBPNGClient::filterSignalsFromOfflineData ( const char \* *pathToOfflineDataSet*, const char \* *pathToFilterSettings*, const char \* *targetPath* ) [pure virtual]

Signal filtering.

This function parses all data of an offline data set and filters signals according to the complex filter settings created with the Telemotive System Client.

## Parameters

<i>pathToOffline-DataSet</i>	path to the offline data set
<i>pathToFilter-Setting</i>	path to the ZIP file including the complex filter settings created with Telemotive System Client
<i>targetPath</i>	path to the target directory where the filtered data should be written to

**6.9.2.10** virtual void WINAPI IBPNGClient::flashDeviceLED ( ) [pure virtual]

Let the connected device blink its front LEDs for identification.

You can use this function to identify you device if you can't identify it over the Name or IP address given from the [IBPNGClientListener::onBPNGDeviceDetected](#) callback function. On TSL all device LEDs will flash.



6.9.2.11 `virtual IFormatList* WINAPI IBPNGClient::getAvailableFormats ( )` [pure virtual]

Return pointer to a format list interface. Returns null in case of error.

All formats returned by this function are available for data conversion.

See Also

[IFormatList](#), [IFormatInfo](#)

6.9.2.12 `virtual IClientProperties* WINAPI IBPNGClient::getClientProperties ( )` [pure virtual]

See Also

[IClientProperties](#), [setClientProperties\(\)](#)

6.9.2.13 `virtual BOOL WINAPI IBPNGClient::getConfig ( const char * path )` [pure virtual]

Download the current logger configuration to the passed path.

If you download the current configuration from the data logger you get a zip Archive that contains all relevant XML and XSD files to modify the configuration in a valid way and reconfigure the device with the [reconfigLogger\(\)](#) function.

**On TSL instance you have to pass a base path. All participants logger configurations will be saved as zip in that directory including a TSLConfig.txt file with additional informations.**

Please note: It is up to you to ensure a valid configuration if you want to modify it with your own tools. You should only modify the xml and not the xsd files. "DeviceConfiguration.xml" and "FirmwareConfiguration.xml" should also not be modified. They specify all xml files that are mandatory to reconfigure the data logger. You can validate the xml files with the supplied xsd files and a XML library of your choice. One possibility would be the XERCES library, see <http://xerces.apache.org/xerces-c/>

#### Parameters

<i>path</i>	The path inclusive file name where to store the downloaded configuration ZIP file or on TSL the basepath for config zips
-------------	--

6.9.2.14 `virtual BPNGError WINAPI IBPNGClient::getConversionError ( int index )` [pure virtual]

Returns the conversion error at *index*.

After getting the number of conversion errors with [getNumConversionErrors\(\)](#) you can get all single errors with this function.

6.9.2.15 `virtual const char* WINAPI IBPNGClient::getDeviceName ( )` [pure virtual]

Get name of device.

After calling one of the init functions [IBPNGClient::initOnline\(\)](#) or [IBPNGClient::initOffline\(\)](#) this function returns the currently configured device name. On TSL the device names will be separated by ','

#### Returns

The device name

#### See Also

[initOnline\(\)](#), [initOffline\(\)](#)

**6.9.2.16** `virtual BPNGError WINAPI IBPNGClient::getDownloadError ( int index ) [pure virtual]`

Returns the download error at *index*.

After getting the number of download errors with [getNumDownloadErrors\(\)](#) you can get all single errors with this function.

**6.9.2.17** `virtual IRdbEventList* WINAPI IBPNGClient::getEventList ( ) [pure virtual]`

Get list of all events from the RDB.

If [initOnline\(\)](#) was called before, the events of the logger's/TSL RDB is returned. If [initOffline\(\)](#) was called before, the events of the RDB included in the offline data set is returned.

#### Returns

Pointer to a [IRdbEventList](#)

**6.9.2.18** `virtual const IFalseMeasureSignalList* WINAPI IBPNGClient::getFalseMeasureSignals ( ) [pure virtual]`

Return pointer to a false measure signal list interface.

After calling the [IBPNGClient::createCCPXCPCDbcFiles\(\)](#) this function returns a pointer to a list with all measure signals which were ignored at DBC file generation.

#### See Also

[IFalseMeasureSignal](#)

**6.9.2.19** `virtual BPNGError WINAPI IBPNGClient::getLastError ( ) [pure virtual]`

Get last error code.

If any called BPNGClient function returns a value that indicates an error you can retrieve further information about that error with this function.

**Returns**

The error description with error code and optional string value.

**See Also**

[BPNGError](#)

**6.9.2.20 virtual const IChannelList\* WINAPI IBPNGClient::getLoggerChannels ( ) [pure virtual]**

Returns pointer to a channel list interface.

After calling one of the init functions [IBPNGClient::initOnline\(\)](#) or [IBPNGClient::initOffline\(\)](#) this function returns a pointer to the logger's/TSL resp. offline data set's channel list.

In case of error null is returned and further information can be retrieved with [getLastError\(\)](#).

**See Also**

[IChannelList](#)

**6.9.2.21 virtual BOOL WINAPI IBPNGClient::getMemoryFillLevel ( MemoryFillLevel \* fillLevel ) [pure virtual]**

get memory fill level of device

On TSL ensure the fillLevel structure has reserved enough space for all members

**Parameters**

<i>fillLevel</i>	structure description in bpngdefines
------------------	--------------------------------------

**Returns**

0 on failure, 1 on success

**6.9.2.22 virtual int WINAPI IBPNGClient::getNumConversionErrors ( ) [pure virtual]**

Returns the number of errors occurred during the last conversion process.

If [convertData\(\)](#) fails, [getLastError\(\)](#) can return different kinds of errors. There are types of errors that won't interrupt the conversion process but will be gathered during conversion and notified at the end. In that case the error code returned by [getLastError\(\)](#) will be `BPNG_CONVERSION_ERRORS` and you can get the number of errors with this function.

**See Also**

[getConversionError\(\)](#)

**6.9.2.23** virtual int WINAPI IBPNGClient::getNumDownloadErrors ( ) [pure virtual]

Returns the number of errors occurred during the last download process.

If [downloadDataSpans\(\)](#) fails, [getLastError\(\)](#) can return different kinds of errors. There are types of errors that won't interrupt the download process but will be gathered during download and notified at the end. In that case the error code returned by [getLastError\(\)](#) will be BPNG\_DOWNLOAD\_ERRORS and you can get the number of errors with this function.

See Also

[getDownloadError\(\)](#)

**6.9.2.24** virtual ClientProperties WINAPI IBPNGClient::getProperties ( ) [pure virtual]

Returns the current properties, deprecated use [getClientProperties\(\)](#) instead.

See Also

[ClientProperties](#), [setProperties\(\)](#)

**6.9.2.25** virtual const char\* WINAPI IBPNGClient::getReferenceDataBasePath ( ) [pure virtual]

Get path to the reference data base.

After calling one of the init functions [IBPNGClient::initOnline\(\)](#) or [IBPNGClient::initOffline\(\)](#) this function returns the path to the current Reference Data Base of the logger resp. the offline data set. For online processes, the RDB is downloaded from the logger to a tmp directory. For offline processes from a ZIP archive, the RDB is extracted to a tmp directory. For offline processes from a directory this function just returns the path to the RDB inside this directory.

Returns

Path to the downloaded or extracted RDB file

See Also

[initOnline\(\)](#), [initOffline\(\)](#)

**6.9.2.26** virtual IRdbTraceBlockList\* WINAPI IBPNGClient::getTraceBlockList ( ) [pure virtual]

Get list of all trace blocks from the RDB.

If [initOnline\(\)](#) was called before, the trace blocks of the logger's/TSL RDB is returned. If [initOffline\(\)](#) was called before, the trace blocks of the RDB included in the offline data set is returned.

Returns

Pointer to a [IRdbEventList](#)

**6.9.2.27** virtual BOOL WINAPI IBPNGClient::initOffline ( const char \* *path* ) [pure virtual]

Initialisation of offline conversion process.

For trace conversion from an offline data set this function must be called first.

Within this function the reference data base is read. Please note that reading a large RDB may take some time, especially in debug mode.

Function will return 0 on failure and 1 on success. In case of failure further information can be retrieved with [getLastError\(\)](#).

**If you have an TSL offline data set ensure you created the IBPNGClientLib instance by calling [getTSLClient\(int numTSLMember\)](#). The numTSLMember can be retrieved by calling [getNumTSLMemberFromOfflineDataSet\(const char\\* offlinePath, int\\* numMember\)](#). The T-SL offline data set can be identified by the abstract\_TSL.txt in the base path.**

**Returns**

0 on failure, 1 on success

**6.9.2.28** virtual BOOL WINAPI IBPNGClient::initOnline ( ) [pure virtual]

Initialisation of download and online conversion process.

For trace download and conversion directly from the device this function must be called after the logger is connected.

Within this function the reference data base is downloaded and read. Please note that reading a large RDB may take some time, especially in debug mode.

Function will return 0 on failure and 1 on success. In case of failure further information can be retrieved with [getLastError\(\)](#).

**Returns**

0 on failure, 1 on success

**6.9.2.29** virtual void WINAPI IBPNGClient::keepLoggerAlive ( const char \* *ip* ) [pure virtual]

Call this to keep logger alive.

The blue PiraT 2 data logger can be configured to go to standby after a specified timeout without any bus traffic on the connected interfaces. If you want to have access to a device without bus traffic, and you don't want to connect to it with [connectLogger\(\)](#) you have to keep it alive by calling this function. This will start a separate thread that sends periodically ping messages to the passed IP address. Receiving these ping messages, the firmware will not shutdown the system.

**Parameters**

<i>ip</i>	The IP address of the logger that should be kept alive
-----------	--

See Also

[stopKeepLoggerAlive\(\)](#)

**6.9.2.30** virtual BOOL WINAPI IBPNGClient::reconfigLogger ( const char \* *configZip* ) [pure virtual]

Reconfig logger with the zipped new configuration.

Reconfigures the logger with the passed configuration. The ZIP archive can be either one that was downloaded with the [getConfig\(\)](#) method, stored by the client software or a modified one. If you want to create your own configuration ZIP archive the structure of this file must be the same as of those mentioned above (xml files inside an "etc" directory). The abstract.txt file and all \*.xsd files are optional. The filename must include the current date in followed form: [YYYY-MM-DD\_-HH-MM-SS] -> Y=year, M=month, D=day, H=hour, M=minute, S=second

**On TSL instance you have to pass all configurations in one string using this format:**

<configpath:1>|<ipAddress:1>;<configPath:2>|<ipAddress:2>;... for example ..\testoutdir\tsl-[emaNLSToN]\_[2016-05-02\_18-03-13]\BP2Img\_MBNR\_1014179.zip|10.64.76.202;..\testoutdir\tsl-[emaNLSToN]\_[2016-05-02\_18-03-13]\BP2Img\_MBNR\_1026651.zip|10.64.76.149;

Please note: It is up to you to ensure a valid configuration if you want to modify it with your own tools. You should only modify the xml and not the xsd files. "DeviceConfiguration.xml" and "FirmwareConfiguration.xml" should also not be modified. They specify all xml files that are mandatory to reconfigure the data logger. You can validate the xml files with the supplied xsd files and a XML library of your choice. One possibility would be the XERCES library, see <http://xerces.apache.org/xerces-c/>

#### Parameters

<i>configZip</i>	Path to the zip file that contains the configuration.
------------------	---

#### Returns

0 on failure, 1 on success

**6.9.2.31** virtual void WINAPI IBPNGClient::release ( ) [pure virtual]

Free memory of this [IBPNGClient](#) instance.

With the call of [getBPNGClient\(\)](#) a new instance is created on the heap. The user is responsible to free its memory if it isn't needed any more. This function calls the delete operator on itself.

Important note: Any further function call on the [IBPNGClient](#) instance after [release\(\)](#) was called will cause a memory access violation and will crash the application!

**6.9.2.32** virtual BOOL WINAPI IBPNGClient::removeAllLicenses ( ) [pure virtual]

Removes the current license file from the logger.

Removes the current license file from the logger.

#### Returns

true on success, false on failure

**6.9.2.33** virtual int WINAPI IBPNGClient::restartDevice ( BOOL *waitForRestart* ) [pure virtual]

restarts the device or TSL

#### Parameters

<i>waitForRestart</i>	if 1 communication waits for the restart
-----------------------	--

#### Returns

0 on failure, 1 on success, -1 on false fw version

**6.9.2.34** virtual void WINAPI IBPNGClient::scanNetworkForLogger ( ) [pure virtual]

Scan network for logger.

This function sends one broadcast UDP messages via all network adapters and notifies the calling application about responding devices with the listener functions onBPNGDeviceDetected(), onBPNGDeviceDisappeared() and onBPNGDeviceStateChange() (see [IBPNGClientListener.h](#)). For each broadcast message sent, the function waits for 100ms for responding devices

The first function call notifies about all found devices. All following calls on the same [IBPNGClient](#) instance will only notify about changes to the previous call.

**6.9.2.35** virtual void WINAPI IBPNGClient::setClientProperties ( IClientProperties \* *properties* ) [pure virtual]

#### Parameters

<i>Pointer</i>	to <a href="#">IClientProperties</a> which can be retrieved from the static function <a href="#">createNewClientProperties()</a> or from <a href="#">IBPNGClient::getClientProperties()</a>
----------------	---

#### See Also

[IClientProperties](#), [getClientProperties\(\)](#), [createNewClientProperties](#)

**6.9.2.36** virtual BOOL WINAPI IBPNGClient::setDefaultConfig ( ) [pure virtual]

Reconfig logger/TSL with the default configuration.

An invalid configuration will set the logger/TSL in error state. To fix this one possibility is to set the logger's default configuration. On TSL every logger will be reset to default configuration.

#### Returns

0 on failure, 1 on success

**6.9.2.37** virtual BOOL WINAPI IBPNGClient::setInfoEvent ( const char \* *msg* ) [pure virtual]

Set an info event with the passed string on the connected logger.

You can set an info event to the RDB. This event will be from type INFO and the passed message is written to the event's comment column

**Returns**

Returns 0 on failure, 1 on success

**6.9.2.38** `virtual BOOL WINAPI IBPNGClient::setMarker ( ) [pure virtual]`

Set a marker on the connected logger. Returns 0 on error.

You can set a marker to the RDB. The set event will be from type MARKER. On TSL the marker will be broadcasted internally.

**Returns**

Returns 0 on failure, 1 on success

**6.9.2.39** `virtual void WINAPI IBPNGClient::setProperties ( ClientProperties val ) [pure virtual]`

Set client properties, deprecated use [setClientProperties\(IClientProperties\\*\)](#) instead.

**See Also**

[ClientProperties](#), [getProperties\(\)](#)

**6.9.2.40** `virtual int WINAPI IBPNGClient::setTime ( int time ) [pure virtual]`

Set logger time and date to the passed UTC time stamp.

The parameter time must be in seconds since 01.01.1970 UTC. On TSL the new time will be applied on every device.

**Returns**

-1 on clientLib busy, 0 on failure, 1 on success

**6.9.2.41** `virtual int WINAPI IBPNGClient::shutdownDevice ( ) [pure virtual]`

shut down the device or TSL

**Returns**

0 on failure, 1 on success, -1 on false fw version

**6.9.2.42** `virtual BOOL WINAPI IBPNGClient::synchronizeRdb ( ) [pure virtual]`

Synchronizes the RDB.

After calling [initOnline\(\)](#) once you can use this function to synchronize the RDB that [getEventList\(\)](#) and [getTraceBlockList\(\)](#) will return the updated lists.



6.9.2.43 virtual BOOL WINAPI IBPNGClient::updateFirmware ( const char \* *fwPath*, BOOL *force* ) [pure virtual]

Update firmware.

This function updates the logger's firmware. An internal version check is done. If the second parameter *force* is 0 only firmware components with an older version than the component's version inside the firmware packet will be updated.

**On TSL instance you have to call a updateFirmware for every device to handle the update in multiple threads. The fwPath argument contains the firmwarepacket-path and the ip separately by '|' <firmwarepacket-path>|<ip> for example C:\bPMini\_SW02-03-01.dat|10.64.76.202**

#### Parameters

<i>fwPath</i>	Path to the firmware packet file that should be installed.
<i>force</i>	Flag whether to update the components independently from the components' versions

#### Returns

0 on failure, 1 on success

6.9.2.44 virtual BOOL WINAPI IBPNGClient::updateLicenses ( const char \* *licenseFilePath* ) [pure virtual]

Update licenses.

Overwrites the current license file with the new one.

**On TSL instance you have to call a updateLicenses for every device The licenseFilePath argument contains the target ip and the license-file-path separately by '|' <ip>|<license-FilePath> for example 10.64.76.202|C:\proj\Lizenz\_bluePiraT2\_MB\_1014179.tml**

#### Parameters

<i>licenseFilePath</i>	Path to the new license file
------------------------	------------------------------

#### Returns

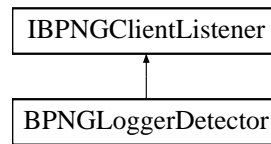
0 on failure, 1 on success

The documentation for this struct was generated from the following file:

- [IBPNGClient.h](#)

## 6.10 IBPNGClientListener Struct Reference

Inheritance diagram for IBPNGClientListener:



## Public Member Functions

- virtual void WINAPI [onBPNGDeviceDetected](#) ([OnlineLoggerInfo](#) \*info)=0  
*Called to notify a detected logger in network.*
- virtual void WINAPI [onBPNGDeviceDisappeared](#) ([OnlineLoggerInfo](#) \*info)=0  
*Called to notify a disappeared logger.*
- virtual void WINAPI [onBPNGDeviceStateChange](#) ([OnlineLoggerInfo](#) \*info)=0  
*Called to notify a logger's state change.*
- virtual int WINAPI [onProgressDataDownload](#) (int percentCompleted)=0  
*Called to indicate the current progress of a file transfer.*
- virtual int WINAPI [onProgressConversion](#) (int percentCompleted, const char \*status)=0  
*Called to indicate the current progress of file conversion.*
- virtual void WINAPI [onStatusMessage](#) (const char \*statusMsg)=0  
*Called to send additional information of the current process to the calling app.*
- virtual void WINAPI [onWarning](#) ([BPNGWarningCode](#) warningCode, const char \*warnMsg)=0  
*Called to inform about a warning.*
- virtual int WINAPI [onTargetPathTooLong](#) (char \*newTarget, int maxSize)=0  
*Called on a too long target directory.*
- virtual int WINAPI [getOverwritingPermission](#) (const char \*filePath)=0  
*Called on existing output trace files.*
- virtual const char \*WINAPI [onLogInDataRequired](#) (const char \*ipAddress)=0  
*Called on accessing password protected functions.*
- virtual void WINAPI [onInvalidPwConfigFound](#) (const char \*ipAddress)=0  
*Called if invalid pw file found on device.*
- virtual void WINAPI [onLogInDataFailed](#) ()=0
- virtual void WINAPI [onResetLogInDataFailed](#) ()=0
- virtual void WINAPI [onFuncAccessDenied](#) ()=0
- virtual int WINAPI [onCriticalDiskSpace](#) (uint64\_t freeSpace, uint64\_t neededSpace, const char \*drive, const char \*msg)=0  
*Called in case of not enough free disk space.*
- virtual void WINAPI [onFirmwareUpdateProgress](#) (int percentage, int stepId, int subStepId, const char \*desc)=0  
*Called on firmware update progress.*
- virtual void WINAPI [onFirmwareUpdateError](#) (int errorId)=0
- virtual int WINAPI [onGetLogReportProgress](#) (int percentage, const char \*desc)=0
- virtual void WINAPI [onDownloadStart](#) (int64\_t totalAmountOfBytes)=0  
*Notifies the listeners before the download starts about the total amount of bytes to be downloaded.*
- virtual void WINAPI [onConversionStart](#) (int64\_t totalAmountOfBytes)=0  
*Notifies the listeners before the conversion starts about the total amount of bytes to be converted.*

### 6.10.1 Member Function Documentation

**6.10.1.1** `virtual int WINAPI IBPNGClientListener::getOverwritingPermission ( const char * filePath )`  
[pure virtual]

Called on existing output trace files.

When an output trace file already exists this function is called. The listener has the possibility to return one of following values: -1: no, don't overwrite file -2: no, overwrite neither this nor any following file 1: yes, overwrite file 2: yes, overwrite this and all following files 0: cancel conversion

Implemented in [BPNGLoggerDetector](#).

**6.10.1.2** `virtual void WINAPI IBPNGClientListener::onBPNGDeviceDetected ( OnlineLoggerInfo * info )`  
[pure virtual]

Called to notify a detected logger in network.

All char\* of the passed OnlineLoggerInfo\* are only valid for the time of the function call. Please ensure to copy the string values.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.3** `virtual void WINAPI IBPNGClientListener::onBPNGDeviceDisappeared ( OnlineLoggerInfo * info )` [pure virtual]

Called to notify a disappeared logger.

All char\* of the passed OnlineLoggerInfo\* are only valid for the time of the function call. Please ensure to copy the string values.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.4** `virtual void WINAPI IBPNGClientListener::onBPNGDeviceStateChange ( OnlineLoggerInfo * info )` [pure virtual]

Called to notify a logger's state change.

All char\* of the passed OnlineLoggerInfo\* are only valid for the time of the function call. Please ensure to copy the string values.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.5** `virtual void WINAPI IBPNGClientListener::onConversionStart ( int64_t totalAmountOfBytes )`  
[pure virtual]

Notifies the listeners before the conversion starts about the total amount of bytes to be converted.

#### Parameters

<i>totalAmountOfBytes</i>	Total data size to be converted
---------------------------	---------------------------------

Implemented in [BPNGLoggerDetector](#).

**6.10.1.6** `virtual int WINAPI IBPNGClientListener::onCriticalDiskSpace ( uint64_t freeSpace, uint64_t neededSpace, const char * drive, const char * msg )` [pure virtual]

Called in case of not enough free disk space.

This notifies the listener about not enough free disk space for data download or conversion. The user can continue or abort the process. Returning 0 will abort the process. In some cases continuing without providing more disk space will call this function immediately again.

#### Parameters

<i>freeSpace</i>	Amount of free space
<i>neededSpace</i>	Amount of needed space
<i>drive</i>	Name of the drive where to store data
<i>msg</i>	Additional message to display

#### Returns

return 0 when process should be aborted, 1 to ignore

Implemented in [BPNGLoggerDetector](#).

**6.10.1.7** `virtual void WINAPI IBPNGClientListener::onDownloadStart ( int64_t totalAmountOfBytes )` [pure virtual]

Notifies the listeners before the download starts about the total amount of bytes to be downloaded.

#### Parameters

<i>totalAmount-OfBytes</i>	Total data size to be downloaded
----------------------------	----------------------------------

Implemented in [BPNGLoggerDetector](#).

**6.10.1.8** `virtual int WINAPI IBPNGClientListener::onGetLogReportProgress ( int percentage, const char * desc )` [pure virtual]

Called on creation of log report

#### Returns

return value 0 indicates an abort request from the implementing class

Implemented in [BPNGLoggerDetector](#).

**6.10.1.9** `virtual void WINAPI IBPNGClientListener::onInvalidPwConfigFound ( const char * ipAddress )` [pure virtual]

Called if invalid pw file found on device.

An error may occurred on transferring the passwordconfiguration to the device, as a result the passwordconfiguration is invalid and needs to be reset to default. Inform the user.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.10** `virtual const char* WINAPI IBPNGClientListener::onLoginDataRequired ( const char * ipAddress ) [pure virtual]`

Called on accessing password protected functions.

When password protected functions are called this listener function queries for login parameters that must be returned from the implementing class.

#### Parameters

<i>ipAddress</i>	IP address of the password protected device
------------------	---

Implemented in [BPNGLoggerDetector](#).

**6.10.1.11** `virtual int WINAPI IBPNGClientListener::onProgressConversion ( int percentCompleted, const char * status ) [pure virtual]`

Called to indicate the current progress of file conversion.

This function notifies the listener about the conversion progress of the raw Telemotive trace data. If the *percentCompleted* value has changed, but the *status* is still the same, the application passes an empty string as status to the function.

#### Parameters

<i>percent-Completed</i>	Percent of the entire conversion process (from 0...100%), -1 indicates the same value as from last function call
<i>status</i>	Status of the conversion process (e.g. "Converting trace data. Block 5 of 32")

#### Returns

return value 0 indicates an abort request from the implementing class

Implemented in [BPNGLoggerDetector](#).

**6.10.1.12** `virtual int WINAPI IBPNGClientListener::onProgressDataDownload ( int percentCompleted ) [pure virtual]`

Called to indicate the current progress of a file transfer.

This function notifies the listener about the download progress of the raw Telemotive trace data.

#### Parameters

<i>percent-Completed</i>	Percentage of the entire download process (from 0...100%). A negative value can be passed if only the abort request is checked. A negative value of -1 indicates a broken ftp connection.
--------------------------	---

### Returns

return value 0 indicates an abort request from the implementing class

Implemented in [BPNGLoggerDetector](#).

**6.10.1.13** `virtual void WINAPI IBPNGClientListener::onStatusMessage ( const char * statusMsg )` [pure virtual]

Called to send additional information of the current process to the calling app.

This function transmit message strings to the listener class. Those messages are only for information purpose. The receiver doesn't have to react on it but can display it on the screen.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.14** `virtual int WINAPI IBPNGClientListener::onTargetPathTooLong ( char * newTarget, int maxSize )` [pure virtual]

Called on a too long target directory.

Called when the resulting file name of the converted files or the files of an offline data set is longer than the maximum allowed size of the file system (Windows 260). The lib user has to pass a new (shorter) base target directory to the passed char array with strcpy. The memory of the array is already allocated by the library and it's size is maxSize. When a new directory was set the value 1 must be returned. Returning another value than 1 will abort the current process with an error result.

Implemented in [BPNGLoggerDetector](#).

**6.10.1.15** `virtual void WINAPI IBPNGClientListener::onWarning ( BPNGWarningCode warningCode, const char * warnMsg )` [pure virtual]

Called to inform about a warning.

This function transmit a warning message to the listener class. Warnings have a WARNING\_CODE and a warning message. Warnings do not interrupt the current process but should be noticed from the user to possibly initiate further provisions.

Implemented in [BPNGLoggerDetector](#).

The documentation for this struct was generated from the following file:

- [IBPNGClientListener.h](#)

## 6.11 IChannel Struct Reference

Channel interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual [ChannelType](#) [getType](#) () const =0  
*Returns the ChannelType.*
- virtual uint8\_t [getIndex](#) () const =0  
*Returns the channel's index.*
- virtual const char \* [getName](#) () const =0  
*Returns the channel's name.*
- virtual uint32\_t [getMainboardNumber](#) () const =0  
*Returns the channel's package id.*
- virtual uint32\_t [getOffset](#) () const =0  
*Returns the channel's offset.*
- virtual BOOL [isMappingActive](#) () const =0  
*Returns whether the channel is mapped.*
- virtual uint8\_t [getMappedChannelIndex](#) () const =0  
*Returns the channel's mapped channel index.*

#### 6.11.1 Detailed Description

Channel interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.12 IChannelList Struct Reference

Channel list interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual int [getSize](#) () const =0  
*Returns the number of channels.*
- virtual const [IChannel](#) \* [getChannel](#) (int index) const =0  
*Returns the [IChannel](#) at index.*

#### 6.12.1 Detailed Description

Channel list interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.13 IClientProperties Struct Reference

The [IClientProperties](#) interface replaces the deprecated [ClientProperties](#) struct.

```
#include <IClientProperties.h>
```

### Public Member Functions

- virtual void WINAPI [setCommonProperties](#) (const char \*nameOfTester, int maxOutputSizeMB, BOOL separatedTimeFormat, BOOL separatedTimeFormatInOfflineSet, const char \*alternativeLoggerName, BOOL useAlternativeLoggerName, BOOL useSubDirectories, BOOL midnightSplitting, BOOL fileTimeSpansLikeSelection, BOOL markerNumberInFileNames, BOOL subfolderWithLoggerName, int maxOfflineZipSizeMB, int maxOutputSizeMBSortedDownload)=0  
*Set Common properties.*
- virtual const char \*WINAPI [getNameOfTester](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual int WINAPI [getMaxOutputSize](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isSeparatedTimeFormat](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isSeparatedTimeFormatInOfflineSet](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual const char \*WINAPI [getAlternativeLoggerName](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isAlternativeLoggerNameActive](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isConvertedFilesInSubDirsActive](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isMidnightSplittingActive](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isFileTimeSpansLikeSelection](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isMarkerNumbersInFileNames](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual BOOL WINAPI [isSubfolderWithLoggerName](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual int WINAPI [getMaxOfflineZipSize](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual int WINAPI [getMaxOutputSizeSortedDownload](#) ()=0  
*see parameter description of [setCommonProperties\(\)](#)*
- virtual void WINAPI [setCANPseudoMsgTimeStampProperties](#) (BOOL writeTimeStampMsg, uint32\_t channelIndex, uint32\_t dlc, uint32\_t canID, uint32\_t hourBitPos, uint32\_t minBitPos, uint32\_t secBitPos, uint32\_t dayBitPos, uint32\_t monthBitPos, uint32\_t yearBitPos)=0  
*Set CAN pseudo properties for writing time stamp messages.*
- virtual BOOL WINAPI [isCANPseudoMsgTimeStampActive](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*



- virtual uint32\_t WINAPI [getCANPseudoMsgChannelIndexTimeStamp](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgDlcTimeStamp](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgCanIDTimeStamp](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgHourBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgMinBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgSecBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgDayBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgMonthBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgYearBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTimeStampProperties\(\)](#)*
- virtual void WINAPI [setCANPseudoMsgTriggerProperties](#) (BOOL writeTriggerMessage, uint32\_t channelIndex, uint32\_t dlc, uint32\_t canID, uint32\_t triggerNumBitPos)=0  
*Set CAN pseudo properties for writing trigger messages.*
- virtual BOOL WINAPI [isCANPseudoMsgTriggerActive](#) ()=0  
*see parameter description of [setCANPseudoMsgTriggerProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgChannelIndexTrigger](#) ()=0  
*see parameter description of [setCANPseudoMsgTriggerProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgDlcTrigger](#) ()=0  
*see parameter description of [setCANPseudoMsgTriggerProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgCanIDTrigger](#) ()=0  
*see parameter description of [setCANPseudoMsgTriggerProperties\(\)](#)*
- virtual uint32\_t WINAPI [getCANPseudoMsgTriggerNumBitPos](#) ()=0  
*see parameter description of [setCANPseudoMsgTriggerProperties\(\)](#)*
- virtual void WINAPI [setMOSTPseudoMsgProperties](#) (BOOL active, uint32\_t src, uint32\_t target, uint32\_t fktBlockID, uint32\_t fktID)=0  
*Set MOST pseudo properties.*
- virtual BOOL WINAPI [isMOSTPseudoMsgActive](#) ()=0  
*see parameter description of [setMOSTPseudoMsgProperties\(\)](#)*
- virtual uint32\_t WINAPI [getMOSTPseudoMsgSourceAddr](#) ()=0  
*see parameter description of [setMOSTPseudoMsgProperties\(\)](#)*
- virtual uint32\_t WINAPI [getMOSTPseudoMsgTargetAddr](#) ()=0  
*see parameter description of [setMOSTPseudoMsgProperties\(\)](#)*
- virtual uint32\_t WINAPI [getMOSTPseudoMsgFktBlockID](#) ()=0  
*see parameter description of [setMOSTPseudoMsgProperties\(\)](#)*
- virtual uint32\_t WINAPI [getMOSTPseudoMsgFktID](#) ()=0  
*see parameter description of [setMOSTPseudoMsgProperties\(\)](#)*
- virtual void WINAPI [useSatelliteTimeForGPSFormats](#) (BOOL flag)=0  
*Set whether to use the satellite time stamp in GPS formats instead of the logger time stamp.*
- virtual BOOL WINAPI [isSatelliteTimeForGPSFormats](#) ()=0

- Returns whether to use the satellite time stamp in GPS formats instead of the logger time stamp.*
- virtual void WINAPI [setIsochronousMost150Channels](#) (const char \*channels)=0  
*Set the channelLabels of the isochronous channels as comma separated string.*
- virtual const char \*WINAPI [getIsochronousMost150Channels](#) ()=0  
*Returns the channelLabels of the isochronous channels as comma separated string.*
- virtual void WINAPI [setAnalogToCANpseudoActive](#) (BOOL flag)=0  
*Set whether to activate the analogue data to CAN pseudo message feature.*
- virtual void WINAPI [addAnalogPortSettings](#) (uint16\_t analogPort, BOOL isActive, uint32\_t canChannel, uint32\_t canID, const char \*dbcPath)=0  
*Set analog port settings.*
- virtual void WINAPI [clearAnalogPortSettings](#) ()=0  
*Clears all port settings set with the [addAnalogPortSettings\(\)](#) function.*

### 6.13.1 Detailed Description

The [IClientProperties](#) interface replaces the deprecated [ClientProperties](#) struct.

Call [IBPNGClient::getClientProperties\(\)](#) to get a pointer to an instance of this interface class.

### 6.13.2 Member Function Documentation

- 6.13.2.1 virtual void WINAPI [IClientProperties::addAnalogPortSettings](#) ( uint16\_t *analogPort*, BOOL *isActive*, uint32\_t *canChannel*, uint32\_t *canID*, const char \* *dbcPath* ) [pure virtual]

Set analog port settings.

#### Parameters

<i>analogPort</i>	Analogue port index
<i>isActive</i>	Specifies whether the data of this port should be written to CAN pseudo messages
<i>canChannel</i>	Specifies the CAN channel that should be used for the pseudo messages
<i>canID</i>	Specifies the CAN ID that should be used for the pseudo messages
<i>dbcPath</i>	The path to the DBC file that specifies the signal of the CAN ID's message that should carry the value

- 6.13.2.2 virtual void WINAPI [IClientProperties::setCANpseudoMsgTimeStampProperties](#) ( BOOL *writeTimeStampMsg*, uint32\_t *channelIndex*, uint32\_t *dlc*, uint32\_t *canID*, uint32\_t *hourBitPos*, uint32\_t *minBitPos*, uint32\_t *secBitPos*, uint32\_t *dayBitPos*, uint32\_t *monthBitPos*, uint32\_t *yearBitPos* ) [pure virtual]

Set CAN pseudo properties for writing time stamp messages.

#### Parameters

<i>writeTimeStampMsg</i>	Active flag for writing periodical CAN pseudo messages with absolute time stamps
<i>channelIndex</i>	CAN channel for the time stamp pseudo messages
<i>dlc</i>	DLC for the time stamp pseudo messages

<i>canID</i>	CAN ID for the time stamp pseudo messages
<i>hourBitPos</i>	Bit position for the hour (0..23, 5 bit length) value in the CAN data bytes
<i>minBitPos</i>	Bit position for the minute (0..59, 6 bit length) value in the CAN data bytes
<i>secBitPos</i>	Bit position for the second (0..59, 6 bit length) value in the CAN data bytes
<i>dayBitPos</i>	Bit position for the day (1..31, 5 bit length) value in the CAN data bytes
<i>monthBitPos</i>	Bit position for the month (1..12, 4 bit length) value in the CAN data bytes
<i>yearBitPos</i>	Bit position for the year (8 bit length) value in the CAN data bytes

**6.13.2.3** `virtual void WINAPI IClientProperties::setCANPseudoMsgTriggerProperties ( BOOL writeTriggerMessage, uint32_t channelIndex, uint32_t dlc, uint32_t canID, uint32_t triggerNumBitPos )` [pure virtual]

Set CAN pseudo properties for writing trigger messages.

#### Parameters

<i>writeTriggerMessage</i>	Active flag for writing CAN pseudo messages with trigger information
<i>channelIndex</i>	CAN channel for the trigger pseudo messages
<i>dlc</i>	DLC for the trigger pseudo messages
<i>canID</i>	CAN ID for the trigger pseudo messages
<i>triggerNumBitPos</i>	Bit position for the trigger's index (16 bit length)

**6.13.2.4** `virtual void WINAPI IClientProperties::setCommonProperties ( const char * nameOfTester, int maxOutputSizeMB, BOOL separatedTimeFormat, BOOL separatedTimeFormatInOfflineSet, const char * alternativeLoggerName, BOOL useAlternativeLoggerName, BOOL useSubDirectories, BOOL midnightSplitting, BOOL fileTimeSpansLikeSelection, BOOL markerNumberInFileNames, BOOL subfolderWithLoggerName, int maxOfflineZipSizeMB, int maxOutputSizeMBSortedDownload )` [pure virtual]

Set Common properties.

#### Parameters

<i>nameOfTester</i>	Name of tester that is written to the converted file names
<i>maxOutputSizeMB</i>	Maximum file size for converted files. When this size is reached a new file is created.
<i>separatedTimeFormat</i>	Specifies the time format that should be used for converted files. Set 1 for long format (e.g. [2011-12-20]_10.15.48) or 0 for short format (e.g. 20111220_101548)
<i>separatedTimeFormatInOfflineSet</i>	Specifies the time format that should be used for offline conversion sets. Set 1 for long format (e.g. [2011-12-20]_10.15.48) or 0 for short format (e.g. 20111220_101548)
<i>alternativeLoggerName</i>	The logger device's name is included in the converted files' names. An alternative logger name can be used.
<i>useAlternativeLoggerName</i>	Set this field to 1 if the alternative logger name should be used in converted file names, 0 if not.

<i>useSub-Directories</i>	Set to 1 if converted files should be stored in subdirectories named by their start date, set 0 if they should not.
<i>midnight-Splitting</i>	Set to 1 if converted files should be splitted at 00:00:00 of each date, set to 0 if they should not.
<i>fileTime-SpansLike-Selection</i>	The file names of the converted files contain the time span of the included data. Setting this parameter to 1 will create time spans like they were specified in the <a href="#">IConversionSet</a> . Setting this to 0 will create time spans according to the effectively included data.
<i>marker-NumberInFile-Names</i>	Specifies whether the indices of the marker included in a converted file should be appended to its file name
<i>subfolder-WithLogger-Name</i>	Specifies whether the name of the subfolder the converted files are stored in should contain the logger name or not.
<i>maxOutput-SizeMB-Sorted-Download</i>	Maximum file size for sorted download trace files. When this size is reached a new file is created.

**6.13.2.5** virtual void WINAPI IClientProperties::setMOSTPseudoMsgProperties ( BOOL *active*, uint32\_t *src*, uint32\_t *target*, uint32\_t *fktBlockID*, uint32\_t *fktID* ) [pure virtual]

Set MOST pseudo properties.

#### Parameters

<i>active</i>	Active flag for writing MOST pseudo messages for trigger
<i>src</i>	Source address
<i>target</i>	Target address
<i>fktBlockID</i>	Function block ID
<i>fktID</i>	Function ID

The documentation for this struct was generated from the following file:

- [IClientProperties.h](#)

## 6.14 IConversionSet Struct Reference

A conversion set stores all conversion relevant settings.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual void [addChannel](#) (ChannelType channelType, uint8\_t channelIndex, const char \*formatId, int fileId=-1, int offset=0, int mbnr=-1, bool mappingActive=false, int mappedChannelId=-1)=0

*Adds a channel to the conversion set and assigns the target format to it.*

- virtual void [addTimeSpan](#) (uint64\_t startTime, uint64\_t endTime)=0  
*Adds a time span to the conversion set.*
- virtual void [addRdbldRange](#) (uint64\_t startId, uint64\_t endId)=0  
*Adds a ReferenceDB ID range to the conversion set.*
- virtual bool [loadFormats](#) (const char \*pathToIniFile)=0  
*Loads the format settings from an ini file.*
- virtual bool [saveFormats](#) (const char \*pathToIniFile)=0  
*Saves the format settings to an ini file.*

### 6.14.1 Detailed Description

A conversion set stores all conversion relevant settings.

To convert trace data a conversion set must be created. Several channels can be added to one conversion set. The trace data of that channels are converted to the assigned formats. The conversion set also includes the data spans that has to be converted.

### 6.14.2 Member Function Documentation

- 6.14.2.1 virtual void IConversionSet::addChannel ( **ChannelType** *channelType*, uint8\_t *channelIndex*, const char \* *formatId*, int *fileId* = -1, int *offset* = 0, int *mbnr* = -1, bool *mappingActive* = false, int *mappedChannelId* = -1 ) [pure virtual]

Adds a channel to the conversion set and assigns the target format to it.

Use the IBPNGClient::getLoggerChannel() function to get all existing channels.

Hint for offset, mappingActive and mappedChannelId: Use the configured values! Else the channel will not be found and the data not written. All information can be retrieved from [IChannel](#)

#### Parameters

<i>channelType</i>	must be one of the appropriate ChannelType enum.
<i>channelIndex</i>	zero-based channel index
<i>formatId</i>	must be one of the appropriate FormatId enum.
<i>fileId</i>	The data of all channels with same formatId and same fileId are written to the same output file. The default value -1 indicates always a separate file for each channel.
<i>offset</i>	Only needed for TSL, default 0. The offset to the original channel number. Can be read out from <a href="#">IChannel</a>
<i>mbnr</i>	Only needed for TSL, default -1. The mainboardnumber of the channels source device. Can be read out from <a href="#">IChannel</a>
<i>mapping-Active</i>	Only needed for Channelmapping, default false. If true the mappedChannelId will be used instead of original index. Can be read out from <a href="#">IChannel</a>
<i>mapped-ChannelId</i>	Only needed for Channelmapping, default -1. If mappingActive is true the mappedChannelId will be used instead of original index. Can be read out from <a href="#">IChannel</a>

**6.14.2.2** `virtual void IConversionSet::addRdbldRange ( uint64_t startId, uint64_t endId )` [pure virtual]

Adds a ReferenceDB ID range to the conversion set.

Passed parameter are IDs from the Reference Data Base (RDB). After calling on of the init functions [IBPNGClient::initOnline\(\)](#) or [IBPNGClient::initOffline\(\)](#) you can get the path to the RDB with [IBPNGClient::getReferenceDataBasePath\(\)](#).

The RDB includes all occurred events like startups, shutdowns, etc. but also all recorded trace files. Each RDB entry has a unique DataBaseEntryID. With this function you can easily select data between arbitrary RDB entries. For example you can convert all data between index X (which is e.g. a startup) and index Y (which is e.g. a shutdown). When the DataBaseEntryId of a trace file is passed, this trace block will be included by the conversion.

#### Parameters

<i>startId</i>	DataBaseEntryId that indicates the start of the data range to be converted
<i>endId</i>	DataBaseEntryId that indicates the end of the data range to be converted

**6.14.2.3** `virtual void IConversionSet::addTimeSpan ( uint64_t startTime, uint64_t endTime )` [pure virtual]

Adds a time span to the conversion set.

The data within the time span will be converted to the specified formats.

#### Parameters

<i>startTime</i>	must be in usec since 01.01.1970 (UTC)
<i>endTime</i>	must be in usec since 01.01.1970 (UTC)

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.15 IFalseMeasureSignal Struct Reference

False measure signal interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual uint8\_t [getDeviceId](#) () const =0  
*Returns the device Id.*
- virtual uint16\_t [getSignalNo](#) () const =0  
*Returns the signal number.*
- virtual [Reason](#) [getIgnoreReason](#) () const =0  
*Returns the ignore reason.*

### 6.15.1 Detailed Description

False measure signal interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.16 IFalseMeasureSignalList Struct Reference

False measure signal list interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual `size_t` [getSize](#) () const =0  
*Returns the number of signals.*
- virtual const [IFalseMeasureSignal](#) \* [getSignal](#) (size\_t index) const =0  
*Returns the [IFalseMeasureSignal](#) at index.*

### 6.16.1 Detailed Description

False measure signal list interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.17 IFormatInfo Struct Reference

FormatInfo interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual const char \* [getFormatId](#) () const =0  
*Returns the FormatId.*
- virtual const char \* [getName](#) (const char \*language) const =0  
*Returns the format's description name in the language passed as ISO 639-1 language code ("en", "de", etc.)*
- virtual `BOOL` [isMultipleChannelSupport](#) () const =0  
*Returns whether the format supports multiple channels in one output file.*
- virtual `BOOL` [isBinaryFormat](#) () const =0  
*Returns whether the format is binary.*
- virtual const char \* [getExtension](#) () const =0

- Returns the format's default extension.*
- virtual int [getNumSupportedChannelTypes](#) () const =0  
*Returns the number of supported channel types.*
- virtual [ChannelType](#) [getChannelType](#) (int index) const =0  
*Returns one supported ChannelType.*
- virtual const char \* [getRequiredLicense](#) () const =0  
*Returns the required license for the format, an empty string for free formats.*

### 6.17.1 Detailed Description

FormatInfo interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.18 IFormatList Struct Reference

Format list interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual int [getSize](#) () const =0  
*Returns the number of available formats.*
- virtual const [IFormatInfo](#) \* [getFormatInfo](#) (int index) const =0  
*Returns the IFormat at index.*

### 6.18.1 Detailed Description

Format list interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.19 IRdbEvent Struct Reference

The documentation for this struct was generated from the following file:

- [RdbDefines.h](#)



## 6.20 IRdbEventList Struct Reference

The documentation for this struct was generated from the following file:

- [RdbDefines.h](#)

## 6.21 IRdbTraceBlock Struct Reference

The documentation for this struct was generated from the following file:

- [RdbDefines.h](#)

## 6.22 IRdbTraceBlockList Struct Reference

The documentation for this struct was generated from the following file:

- [RdbDefines.h](#)

## 6.23 ITesttoolsChannel Struct Reference

Channel interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual [IChannel](#) \* [getIChannel](#) () const =0  
*Returns the [IChannel](#) of this [ITesttoolsChannel](#).*
- virtual BOOL [matchIChannel](#) (const [IChannel](#) \*iChannel) const =0  
*Returns whether the channel matches with the contained channel.*
- virtual uint32\_t [getContainerId](#) () const =0  
*Returns the channel's containerId.*
- virtual uint32\_t [getPseudoContainerId](#) () const =0  
*Returns the channel's associated containerId.*
- virtual const char \* [getPseudoChannelName](#) () const =0  
*Returns the channel's associated containerId name.*
- virtual uint16\_t [getBaseCanId](#) () const =0  
*Returns the channel's containerId.*
- virtual bool [isExtendedCanId](#) () const =0  
*Returns the channel's containerId is extended or not.*
- virtual const char \* [getHostIp](#) () const =0  
*Returns the ethernet host ip.*
- virtual const char \* [getDeviceIp](#) () const =0  
*Returns the ethernet device ip.*

- virtual unsigned int [getDevicePort](#) () const =0  
*Returns the ethernet device ip.*
- virtual int [getProtocol](#) () const =0  
*Returns protocol.*
- virtual int [getDebugLevel](#) () const =0  
*Returns debuglevel.*

### 6.23.1 Detailed Description

Channel interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.24 ITesttoolsChannelList Struct Reference

TesttoolsChannel list interface.

```
#include <BPNGDefines.h>
```

### Public Member Functions

- virtual int [getSize](#) () const =0  
*Returns the number of channels.*
- virtual const [ITesttoolsChannel](#) \* [getTesttoolsChannel](#) (int index) const =0  
*Returns the [ITesttoolsChannel](#) at index.*

### 6.24.1 Detailed Description

TesttoolsChannel list interface.

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.25 LoginData Struct Reference

structure for login

```
#include <BPNGDefines.h>
```

### Public Attributes

- const char \* **userName**
- const char \* **userPwd**

### 6.25.1 Detailed Description

structure for login

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.26 MemoryFillLevel Struct Reference

stores memory fill level of a device

```
#include <BPNGDefines.h>
```

### Public Attributes

- uint16\_t [ringBufferSize](#)  
*size of ringbuffer in GBytes*
- uint8\_t [percentageFill](#)  
*percentage filled*
- uint8\_t [percentageFillProtected](#)  
*percentage filled of protected areas*
- uint64\_t [mbnr](#)  
*mainboardnumber of device*

### 6.26.1 Detailed Description

stores memory fill level of a device

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.27 MOSTPseudoMessagesProperties Struct Reference

MOST pseudo properties,.

```
#include <BPNGDefines.h>
```

### Public Attributes

- uint8\_t [writeMessages](#)  
*Active flag for writing MOST pseudo messages for trigger.*
- uint32\_t [source](#)  
*Source address.*
- uint32\_t [target](#)

- *Target address.*
- uint32\_t [functionBlockID](#)
- *Function block ID.*
- uint32\_t [functionID](#)
- *Function ID.*
- uint32\_t **reserved1**
- uint32\_t **reserved2**
- uint32\_t **reserved3**
- uint32\_t **reserved4**

### 6.27.1 Detailed Description

MOST pseudo properties,.

**Deprecated** use [IClientProperties](#) interface instead

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.28 OnlineLoggerInfo Struct Reference

Struct with information about a logger found in LAN.

```
#include <BPNGDefines.h>
```

### Public Attributes

- const char \* [ip](#)  
*the logger's ip address*
- const char \* [name](#)  
*the logger's name*
- const char \* [mbnr](#)  
*mainboard number*
- const char \* [deviceSN](#)  
*device serial number, since FW 2.2.1*
- uint8\_t [occupied](#)  
*0 = not occupied, 1 = connected with client, 2 = occupied by temp config (via external media)*
- const char \* [currentUser](#)  
*user name of connected pc account*
- uint8\_t [loggerStatus](#)  
*current logger status,*
- uint8\_t [wlan](#)  
*Flag for connection type. 0 = ethernet, 1 = wlan.*
- const char \* [tsIEth0IP](#)  
*ip address of device connected to eth0, 0.0.0.0 if none*

- const char \* [tslEth1IP](#)  
*ip address of device connected to eth1, 0.0.0.0 if none*
- int8\_t [tslId](#)  
*id for device in tsl network, continues in tsl, starts with 0 on first device*
- int32\_t [tslNetworkId](#)  
*id of tsl network, -1 = no TSL, all devices with same tslNetworkId belong to the same TSL*
- const char \* [tslName](#)  
*name(id) of tsl network*
- uint8\_t [deviceType](#)  
*Device type,.*
- const char \* [fwVersion](#)  
*Current firmware version, since FW 2.1.1.*

### 6.28.1 Detailed Description

Struct with information about a logger found in LAN.

### 6.28.2 Member Data Documentation

#### 6.28.2.1 uint8\_t OnlineLoggerInfo::deviceType

Device type,.

See Also

[BPNGDeviceType](#)

#### 6.28.2.2 uint8\_t OnlineLoggerInfo::loggerStatus

current logger status,

See Also

[BPNGLoggerStatus](#)

The documentation for this struct was generated from the following file:

- [BPNGDefines.h](#)

## 6.29 RdbEvent2 Struct Reference

Implementation class for a wrapper of [IRdbEvent](#) using STL classes.

```
#include <RdbEventList.hh>
```

## Public Member Functions

- **RdbEvent2** (const [IRdbEvent](#) \*rdbEvent)

## Public Attributes

- RdbEventType **type**
- uint64\_t **uniqueID**
- uint64\_t **timeStamp**
- std::string **timeZone**
- int **index**
- std::string **comment**

### 6.29.1 Detailed Description

Implementation class for a wrapper of [IRdbEvent](#) using STL classes.

To achieve a compiler independend interface for the Telemotive Client Library only pointer to complex objects are returned from some functions. The [IRdbEvent](#) class is can be wrapped by this class RdbEvent to have access to its members in the usual way. You only have to pass a [IRdbEvent](#) pointer to the constructor.

#### See Also

[IRdbEvent](#), [RdbEventList](#)

The documentation for this struct was generated from the following file:

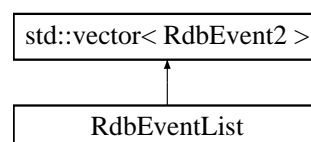
- [RdbEventList.hh](#)

## 6.30 RdbEventList Class Reference

Implementation class for a wrapper of [IRdbEventList](#) using STL classes.

```
#include <RdbEventList.hh>
```

Inheritance diagram for RdbEventList:



## Public Member Functions

- **RdbEventList** (const [IRdbEventList](#) \*list)

### 6.30.1 Detailed Description

Implementation class for a wrapper of [IRdbEventList](#) using STL classes.

To achieve a compiler independent interface for the Telemotive Client Library only pointer to complex objects are returned from some functions. The class [IRdbEventList](#) is nothing else than a vector of [IRdbEvent](#) objects. Pass a pointer to [IRdbEventList](#) to the constructor of this wrapper class [RdbEventList](#) and you get a STL vector of [RdbEvent](#) objects which by itself is a wrapper to [IRdbEvent](#)

See Also

[RdbEvent](#), [IRdbEventList](#), [IRdbEvent](#)

The documentation for this class was generated from the following file:

- [RdbEventList.hh](#)

## 6.31 TSLCluster Class Reference

```
#include <BPNGLoggerDetector.hh>
```

### Public Types

- enum [ConnectionType](#) {  
    [DOWNLOAD](#), [CONVERSION](#), [CONFIG](#), [BUGREPORT](#),  
    [FW\\_UPDATE](#) }

### Public Member Functions

- [TSLCluster](#) ()
- [TSLCluster](#) ([BPNGDevice](#) firstDevice)
- void [addDevice](#) ([BPNGDevice](#) device)
- std::string [getConnectionString](#) ([ConnectionType](#) connectionType)
- int [getNumMembers](#) ([ConnectionType](#) connectionType)
- std::string [getTSLName](#) ()
- void [print](#) ()
- std::vector< [BPNGDevice](#) >::iterator [begin](#) ()
- std::vector< [BPNGDevice](#) >::iterator [end](#) ()

### 6.31.1 Detailed Description

A simple class that represents a Telemotive System Link chain.

### 6.31.2 Member Enumeration Documentation

#### 6.31.2.1 enum TSLCluster::ConnectionType

A enumeration of types of which task the connection will be used for.

##### Enumerator

**DOWNLOAD** Download tasks.  
**CONVERSION** Conversion tasks.  
**CONFIG** Configuration tasks.  
**BUGREPORT** Create bug report.  
**FW\_UPDATE** make Firmware update

### 6.31.3 Constructor & Destructor Documentation

#### 6.31.3.1 TSLCluster::TSLCluster ( ) [inline]

##### Constructor

#### 6.31.3.2 TSLCluster::TSLCluster ( BPNGDevice *firstDevice* ) [inline]

##### Constructor

##### Parameters

<i>firstDevice</i>	the first device of the chain.
--------------------	--------------------------------

### 6.31.4 Member Function Documentation

#### 6.31.4.1 void TSLCluster::addDevice ( BPNGDevice *device* ) [inline]

Add a [BPNGDevice](#) to the [TSLCluster](#).

##### Parameters

<i>device</i>	the new device
---------------	----------------

#### 6.31.4.2 std::vector<BPNGDevice>::iterator TSLCluster::begin ( ) [inline]

Begin iterator for ranged base for loop

##### Returns

the begin iterator of internal [BPNGDevice](#) vector.



#### 6.31.4.3 `std::vector<BPNGDevice>::iterator TSLCluster::end ( )` [inline]

End iterator for ranged base for loop

##### Returns

the end iterator of internal [BPNGDevice](#) vector.

#### 6.31.4.4 `std::string TSLCluster::getConnectionString ( ConnectionType connectionType )` [inline]

get the ip string (ips separated by ';') for [IBPNGClient::connectLogger\(const char\\* ipAddress\)](#)

It is recommended to exclude the Remote Control Touch on Download and Conversion cause its not a data logging device.

##### Parameters

<i>connection-Type</i>	the type of work for which the connection will be used
------------------------	--

##### Returns

the ip string

#### 6.31.4.5 `int TSLCluster::getNumMembers ( ConnectionType connectionType )` [inline]

get the number of participants for [IBPNGClient::getTSLClient\(int numTSLMember\)](#)

It is recommended to exclude the Remote Control Touch on Download and Conversion cause its not a data logging device.

##### Parameters

<i>connection-Type</i>	the type of work for which the connection will be used
------------------------	--

##### Returns

number of participants

#### 6.31.4.6 `std::string TSLCluster::getTSLName ( )` [inline]

Get the name of TSL chain. All devices in the chain have the same TSL name.

##### Returns

the TSL name.

#### 6.31.4.7 void TSLCluster::print ( ) [inline]

Stream the [TSLCluster](#) to cout.

The documentation for this class was generated from the following file:

- [BPNGLoggerDetector.hh](#)

## Kapitel 7

# File Documentation

### 7.1 BPNGDefines.h File Reference

Defines for Telemotive Client Library.

```
#include "cstdio"
#include "stdint.h"
```

#### Classes

- struct [IFalseMeasureSignal](#)  
*False measure signal interface.*
- struct [IFalseMeasureSignalList](#)  
*False measure signal list interface.*
- struct [IChannel](#)  
*Channel interface.*
- struct [ITesttoolsChannel](#)  
*Channel interface.*
- struct [IChannelList](#)  
*Channel list interface.*
- struct [ITesttoolsChannelList](#)  
*TesttoolsChannel list interface.*
- struct [IFormatInfo](#)  
*FormatInfo interface.*
- struct [IFormatList](#)  
*Format list interface.*
- struct [IConversionSet](#)  
*A conversion set stores all conversion relevant settings.*
- struct [OnlineLoggerInfo](#)  
*Struct with information about a logger found in LAN.*
- struct [DataSpan](#)
- struct [BPNGError](#)  
*Error struct with error code and optional error message.*

- struct [LoginData](#)  
*structure for login*
- struct [CommonProperties](#)  
*Common properties,.*
- struct [CANPseudoMessagesProperties](#)  
*CAN pseudo message properties,.*
- struct [MOSTPseudoMessagesProperties](#)  
*MOST pseudo properties,.*
- struct [ConversionProperties](#)  
*Conversion Properties,.*
- struct [ClientProperties](#)  
*ClientProperties,.*
- struct [MemoryFillLevel](#)  
*stores memory fill level of a device*

## Macros

- #define **\_\_BPNGDEFINES\_H\_\_**
- #define **WINAPI**
- #define **DECLDIR**
- #define **BOOL** bool
- #define **VOID** void

## Typedefs

- typedef void(WINAPI \* [onLogRequest](#) )(const char \*logRecord)  
*Pointer to a function named onLogRequest with one parameter and no return value.*

## Enumerations

- enum [BPNGErrCode](#) {  
[BPNG\\_NOERR](#) = 0, [BPNG\\_LOGGER\\_NOT\\_FOUND](#) = 1, [BPNG\\_NOT\\_CONNECTED](#) = 2, [BPNG\\_CONNECT\\_FTP\\_FAILED](#) = 3,  
[BPNG\\_CONNECT\\_TMPBUS\\_FAILED](#) = 4, [BPNG\\_TMPBUS\\_NOT\\_CONNECTED](#) = 5, [BPNG\\_AMBIGUOUS\\_IP](#) = 66, [BPNG\\_FAILED\\_TO\\_CONNECT\\_STREAMING](#) = 67,  
[BPNG\\_FTP\\_NOT\\_CONNECTED](#) = 6, [BPNG\\_FTP\\_SERVER\\_NOT\\_FOUND](#) = 7, [BPNG\\_FTP\\_LOGIN\\_FAILED](#) = 8, [BPNG\\_FTP\\_REMOTE\\_PATH\\_NOT\\_FOUND](#) = 9,  
[BPNG\\_FTP\\_READ\\_REMOTE\\_FILE\\_ERROR](#) = 10, [BPNG\\_FTP\\_WRITE\\_REMOTE\\_FILE\\_ERROR](#) = 11, [BPNG\\_FTP\\_TRANSFER\\_USER\\_CANCELED](#) = 12, [BPNG\\_FTP\\_CREATE\\_REMOTE\\_DIR\\_ERROR](#) = 13,  
[BPNG\\_FTP\\_REMOVE\\_REMOTE\\_DIR\\_ERROR](#) = 14, [BPNG\\_FTP\\_REMOVE\\_REMOTE\\_FILE\\_ERROR](#) = 15, [BPNG\\_FTP\\_CHANGE\\_CWD\\_ERROR](#) = 16, [BPNG\\_TMPBUS\\_COPYRDB\\_ERROR](#) = 17,  
[BPNG\\_TMPBUS\\_SEND\\_MSG\\_ERROR](#) = 18, [BPNG\\_TMPBUS\\_REQUEST\\_ERROR](#) = 19, [BPNG\\_FAILED\\_TO\\_CREATE\\_LOCAL\\_FILE\\_OR\\_DIRECTORY](#) = 20, [BPNG\\_LOCAL\\_PATH\\_NOT\\_FOUND](#) = 21,  
[BPNG\\_READ\\_LOCAL\\_FILE\\_ERROR](#) = 22, [BPNG\\_WRITE\\_LOCAL\\_FILE\\_ERROR](#) = 23,

```

BPNG_FILE_EXISTS_ERROR = 24, BPNG_DIR_EXISTS_ERROR = 25,
BPNG_TARGET_PATH_TOO_LONG = 26, BPNG_ZIP_EXCEEDS_FATFS_MAX = 27, B-
PNG_XML_PARSER_ERROR = 28, BPNG_INITIALISATION_ERROR = 29,
BPNG_RDB_SQLITE_QUERY_ERROR = 30, BPNG_RDB_OPEN_FAILED = 31, BPNG-
_CONVENSION_ERRORS = 32, BPNG_CONV_SET_NOT_FOUND = 33,
BPNG_NOTHING_TO_CONVERT = 34, BPNG_TMT_FILE_ID_ERROR = 35, BPNG_TM-
T_FORMAT_ERROR_VERSION = 36, BPNG_TMT_FORMAT_ERROR_TS = 37,
BPNG_INVALID_MESSAGE_ERROR = 38, BPNG_INVALID_MESSAGE_ID = 39, BPNG-
_INVALID_MESSAGE_TS = 40, BPNG_INVALID_MESSAGE_SUBID = 41,
BPNG_INVALID_MESSAGE_LEN = 42, BPNG_CONV_FORMAT_ERROR = 43, BPNG_-
_DOWNLOAD_ERRORS = 44, BPNG_NOTHING_TO_DOWNLOAD = 45,
BPNG_INVALID_OFFLINE_SET = 46, BPNG_PARAMETER_MISMATCH = 47, BPNG_F-
W_VERSION_CHECK_ERROR = 48, BPNG_USER_CANCELLED = 49,
BPNG_MIN_VERSION_ERROR = 50, BPNG_EXCEPTION = 51, BPNG_INCOMPATIBL-
E_RDB = 52, BPNG_UNSPECIFIED_ERROR = 53,
BPNG_LOAD_DBC_FAILED = 81, BPNG_CCP_XCP_PARSER_ERROR = 54, BPNG-
_CCP_XCP_DBC_GENERATOR_ERROR = 55, BPNG_CCP_XCP_SEQUENCE_GENE-
RATOR_ERROR = 56,
BPNG_INSUFFICIENT_DISK_SPACE = 57, BPNG_FWUPDATE_FAILED = 58, BPNG_-
INDEX_OUT_OF_RANGE_ERR = 59, BPNG_READ_CONFIG_BACKUP_ERR = 60,
BPNG_INVALID_RPC_COMMAND = 61, BPNG_INVALID_TSL_CASCDING = 62, BPNG-
_LOGIN_CANCELED = 63, BPNG_USER_PWD_WRONG = 64,
BPNG_NO_ACCESS_FOR_FUNCTION = 65, BPNG_STREAMING_PROTOCOLL_ERR-
OR = 68, BPNG_STREAMING_SOCKET_ERROR = 69, BPNG_STREAMING_DISABL-
ED = 70,
BPNG_FW_DEPRECATED = 71, BPNG_STREAMING_ABORTED_BY_PEER = 72, BP-
NG_INCONSISTENT_TSL_FWVERSIONS = 80, BPNG_INVALID_TSL_CLUSTER = 82,
BPNG_DLL_NO_FORMAT_PLUGIN = 83, BPNG_FORMAT_PLUGIN_ID_EXISTS = 84,
BPNG_DLL_NO_SYSTEMCLIENTLISTENER_PLUGIN = 90, BPNG_FAILED_RENAME-
_RESUMED_OFFLINEDATASET = 85,
BPNG_FAILED_RENAME_RESUMED_RDB = 86, BPNG_RESUME_INIT_FAILURE = 87,
BPNG_SIGNAL_FILTER_INVALID_CONFIG = 88, BPNG_BAD_ALLOC = 89 }

```

*enum Error codes*

- enum FWUpdateErrorCode {
 

```

FWUPDATE_ERRORCODE_NO_ERR = 0, FWUPDATE_ERRORCODE_FW_PKT_NA-
ME_EMPTY = -2, FWUPDATE_ERRORCODE_FW_PKT_MISSING = -3, FWUPDATE_-
ERRORCODE_NAMED_PIPE_SERVER_MKNOD = -4,
FWUPDATE_ERRORCODE_NAMED_PIPE_SERVER_OPEN = -5, FWUPDATE_ERRO-
RCODE_FW_UPDATE_NOT_IN_PROGRESS_TIMEOUT = -6, FWUPDATE_ERRORCO-
DE_MISSING_LINUX_DISTR = -7, FWUPDATE_ERRORCODE_MISSING_LIBTMLIB_F-
ILE = -8,
FWUPDATE_ERRORCODE_MISSING_TMLIB_FILE = -9, FWUPDATE_ERRORCODE_-
MISSING_ATOM_FILE = -10, FWUPDATE_ERRORCODE_MISSING_CLIENT_FILE = -
11, FWUPDATE_ERRORCODE_MISSING_FPGAA_FILE = -12,
FWUPDATE_ERRORCODE_MISSING_FPGAB_FILE = -13, FWUPDATE_ERRORCOD-
E_MISSING_EXTENSION_BOARD_FPGA_FILE = -14, FWUPDATE_ERRORCODE_MI-
SSING_GBE_FILE = -15, FWUPDATE_ERRORCODE_MISSING_SBC_FILE = -16,
FWUPDATE_ERRORCODE_MISSING_SBC_FLASH_SCRIPT = -17, FWUPDATE_ERR-
ORCODE_MISSING_FPGA_FLASH_SCRIPT = -18, FWUPDATE_ERRORCODE_MISSI-
NG_RCV_FILE = -19, FWUPDATE_ERRORCODE_MISSING_LINUX_SETUP_ARCHIVE
= -20,
FWUPDATE_ERRORCODE_UNKNOWN_MB_HW_VERSION = -21, FWUPDATE_ERR-
ORCODE_UNKNOWN_EXTENSION_BOARD = -22, FWUPDATE_ERRORCODE_UNK-

```

- NOWN\_EXTENSION\_BOARD\_VARIANCE** = -23, **FWUPDATE\_ERRORCODE\_NOT\_READABLE\_EXTENSION\_BOARD\_VARIANCE** = -24,  
**FWUPDATE\_ERRORCODE\_NOT\_READABLE\_EXTENSION\_BOARD\_HW\_VERSION** = -25, **FWUPDATE\_ERRORCODE\_NOT\_READABLE\_HW\_TYPE\_VERSION** = -26, **FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_APP\_LIBS** = -27, **FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_RC** = -28,  
**FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_GBEC** = -29, **FWUPDATE\_ERRORCODE\_CONV\_CFG\_ERROR** = -30, **FWUPDATE\_ERRORCODE\_FAILED\_UNCOMPRESS\_LINUX\_KERNEL** = -31, **FWUPDATE\_ERRORCODE\_FAILED\_UNCOMPRESS\_LINUX\_KERNEL\_MODULES** = -32,  
**FWUPDATE\_ERRORCODE\_FAILED\_CPY\_LINUX\_KERNEL** = -33, **FWUPDATE\_ERRORCODE\_FAILED\_UNCOMPRESS\_CLIENT\_FILE** = -34, **FWUPDATE\_ERRORCODE\_FAILED\_CPY\_CLIENT\_FILE** = -35, **FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_LINUX\_DISTR** = -36,  
**FWUPDATE\_ERRORCODE\_FAILED\_SBC\_FLASH** = -37, **FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_CCP\_XCP** = -38, **FWUPDATE\_ERRORCODE\_FAILED\_UPDATE\_CCP\_XCP\_SEED\_KEY\_SERVERS** = -39, **FWUPDATE\_ERRORCODE\_MISSING\_CCP\_XCP\_FILE** = -40,  
**FWUPDATE\_ERRORCODE\_MISSING\_CCP\_XCP\_SEED\_KEY\_SERVER\_FILE** = -41, **FWUPDATE\_ERRORCODE\_MISSING\_SPYNIC\_FILE** = -42, **FWUPDATE\_ERRORCODE\_MISSING\_LOADING\_ISPVM** = -43, **FWUPDATE\_ERRORCODE\_MISSING\_DEVICE\_FPGAB\_FILE** = -44,  
**FWUPDATE\_ERRORCODE\_MISSING\_DEVICE\_FPGAA\_FILE** = -45, **FWUPDATE\_ERRORCODE\_UNREADY\_FPGAA** = -46, **FWUPDATE\_ERRORCODE\_LINUX\_KERNEL\_MAY\_FREEZE\_SYSTEM** = -47, **FWUPDATE\_ERRORCODE\_NOT\_SET\_DEVICE\_PATH** = -48,  
**FWUPDATE\_ERRORCODE\_NOT\_SET\_FW\_FILE** = -49, **FWUPDATE\_ERRORCODE\_NOT\_SET\_FPGA\_KEY** = -50, **FWUPDATE\_ERRORCODE\_MISSING\_DEVICE\_FILE** = -51, **FWUPDATE\_ERRORCODE\_MISSING\_FPGA\_FW\_FILE** = -52,  
**FWUPDATE\_ERRORCODE\_MISSING\_TMUDEVQ** = -53, **FWUPDATE\_ERRORCODE\_UNKNOWN\_DEVICE\_PATH** = -54, **FWUPDATE\_ERRORCODE\_FAILED\_CPY\_FPGA\_FILE** = -55, **FWUPDATE\_ERRORCODE\_FAILED\_LINKING\_FW\_UPDATE** = -56,  
**FWUPDATE\_ERRORCODE\_ERROR\_FLASH\_FPGA** = -57, **FWUPDATE\_ERRORCODE\_ERROR\_FLASH\_SPYNIC** = -58, **FWUPDATE\_ERRORCODE\_ERROR\_LOADING\_FPGA\_JTAG\_DRIVER** = -59, **FWUPDATE\_ERRORCODE\_MISSING\_EXTENSION\_BOARD\_VIA\_PCIE** = -60,  
**FWUPDATE\_ERRORCODE\_FAILED\_CONV\_CFG** = -61, **FWUPDATE\_ERRORCODE\_FAILED\_UPLOAD** = -62, **FWUPDATE\_ERRORCODE\_FWUFOLDER\_EXISTS** = 63, **FWUPDATE\_ERRORCODE\_UNDEFINED** = -1 }
- enum **BPNGWarningCode** {  
**BPNG\_NOWARNING**, **BPNG\_WARNING\_CLOSE\_TRACE\_FILES**, **BPNG\_WARNING\_MESSAGES\_NOT\_CONVERTED**, **BPNG\_WARNING\_NO\_ESO\_TRACE**,  
**BPNG\_WARNING\_TSL\_WITH\_DIFFERENT\_TIMEZONES**, **BPNG\_WARNING\_RECOVERING\_FAILED** }  
*Warning codes.*
  - enum **LanguageID** { **BPNG\_GERMAN**, **BPNG\_ENGLISH** }  
*Languages.*
  - enum **BPNGBugreportMode** {  
**BR\_FULL\_WO\_TRACES** = 0, **BR\_ONLY\_LOGS** = 1, **BR\_FDB\_RDB** = 2, **BR\_ONLY\_CLIENT** = 3,  
**BR\_FULL\_ALL\_TRACES** = 4, **BR\_FULL\_TIMESPAN\_TRACES** = 5 }  
*Mode for the `IBPNG::downloadBugReport()` function.*

- enum [ChannelType](#) {  
`CH_UNDEFINED = 0, OBSOLETE_CH_CANLS, CH_CAN, CH_LIN,  
CH_SERIAL, CH_ETHERNET, CH_FLEXRAY, CH_MOST25_CTRL,  
CH_MOST25_MDP, CH_MOST25_SYNC, CH_MOST150_CTRL, CH_MOST150_MDP,  
CH_MOST150_MEP, CH_MOST150_STREAM, CH_ANALOG_IN, CH_DIGITAL_IN,  
CH_CAMERA, CH_CCPXCP, CH_DIAG, CH_GPS,  
CH_ECL, CH_COMPLEXFILTER, CH_TTY }`  
*Currently supported interfaces.*
- enum **PwdPrivilegesFuncId** {  
`REMOVE_DATA = 0, SET_TIME, SET_EVENT, RECONFIG,  
RECONFIG_PASSWORD, RECONFIG_COMPLEX_FILTER, UPLOAD_WINE_DLLS, U-  
PDATE_FIRMWARE,  
CHANGE_LICENCES, PRIVILEGES_END }`
- enum [Reason](#) {  
`R_UNSUPPORTED_BIT_MASK, R_BIT_MASK_OVERLAP, R_UNSUPPORTED_COMP-  
U_TAB, R_FORBIDDEN_TAB_VALUE,  
R_UNKNOWN }`
- enum [BPNGLoggerStatus](#) {  
`LS_OK = 0, LS_ERROR = 1, LS_NOSYNC = 2, LS_WARNING = 3,  
LS_FWUPDATE = 4, LS_MEM = 5, LS_RING = 6, LS_UNDEFINED = -1 }`  
*Logger status.*
- enum [BPNGDeviceType](#) {  
`DEV_BP2, DEV_BPMINI, DEV_BP2_V1X, DEV_BP2_V2X,  
DEV_RC_TOUCH, DEV_BP_REMOTE, DEV_TSL = 0x80, DEV_UNKNOWN = 0xFF }`  
*Enumartion of Telemotives next generation data loggers.*
- enum [DataSpanType](#) { **DST\_IDSPAN** = 0, **DST\_TIMESPAN** = 1 }  
*Types for [DataSpan](#).*

### 7.1.1 Detailed Description

Defines for Telemotive Client Library.

#### Author

Markus van Pinxteren

#### Date

12.05.2010

### 7.1.2 Enumeration Type Documentation

#### 7.1.2.1 enum BPNGBugreportMode

Mode for the `IBPNG::downloadBugReport()` function.

#### Enumerator

- BR\_FULL\_WO\_TRACES** Full bug report without traces.
- BR\_ONLY\_LOGS** Only log files are downloaded.

**BR\_FDB\_RDB** only FDB and RDB are downloaded

**BR\_ONLY\_CLIENT** only client logs are stored

**BR\_FULL\_ALL\_TRACES** Full bug report with all traces files.

**BR\_FULL\_TIMESPAN\_TRACES** Full bugreport with trace file of a specified time span.

### 7.1.2.2 enum BPNGDeviceType

Enumartion of Telemotives next generation data loggers.

Enumerator

**DEV\_BP2** **Deprecated** For blue PiraT 2 devices use type *DEV\_BP2\_V1X*, for new blue PiraT 2 5E devices use *DEV\_BP2\_V2X*

**DEV\_BPMINI** blue PiraT mini devices

**DEV\_BP2\_V1X** standard blue PiraT 2 device

**DEV\_BP2\_V2X** blue PiraT 2 5E device

**DEV\_RC\_TOUCH** Remote Control Touch.

**DEV\_BP\_REMOTE** blue PiraT Remote

**DEV\_TSL** internal use only! don't use!

### 7.1.2.3 enum BPNGErrCode

enum Error codes

An error is identified by one of the following error codes. Additional information may be found in the [BPNGError::msg](#) field (e.g. file path that causes a *BPNG\_LOCAL\_PATH\_NOT\_FOUND* error)

Enumerator

**BPNG\_NOERR** no error

**BPNG\_LOGGER\_NOT\_FOUND** The IP address the lib wanted to connect was not found.

**BPNG\_NOT\_CONNECTED** A function call failed because the logger was not connected.

**BPNG\_CONNECT\_FTP\_FAILED** Establishing the ftp connection failed.

**BPNG\_CONNECT\_TMPBUS\_FAILED** Establishing the TMP (Telemotive Protocol) bus connection failed.

**BPNG\_TMPBUS\_NOT\_CONNECTED** TMP bus is not connected.

**BPNG\_AMBIGUOUS\_IP** multiple devices with same IP available

**BPNG\_FAILED\_TO\_CONNECT\_STREAMING** Streaming feature could not be connected.

**BPNG\_FTP\_NOT\_CONNECTED** FTP is not connected.

**BPNG\_FTP\_SERVER\_NOT\_FOUND** FTP server is not found.

**BPNG\_FTP\_LOGIN\_FAILED** FTP login failed.

**BPNG\_FTP\_REMOTE\_PATH\_NOT\_FOUND** A requested path on the FTP server is not found.



- BPNG\_FTP\_READ\_REMOTE\_FILE\_ERROR** Can't read a file on the FTP server.
- BPNG\_FTP\_WRITE\_REMOTE\_FILE\_ERROR** Can't write a file on the FTP server.
- BPNG\_FTP\_TRANSFER\_USER\_CANCELED** FTP file transfer was canceled by the user.
- BPNG\_FTP\_CREATE\_REMOTE\_DIR\_ERROR** Can't create the directory on the FTP server.
- BPNG\_FTP\_REMOVE\_REMOTE\_DIR\_ERROR** Can't remove the directory on the FTP server.
- BPNG\_FTP\_REMOVE\_REMOTE\_FILE\_ERROR** Can't remove the file on the FTP server.
- BPNG\_FTP\_CHANGE\_CWD\_ERROR** Can't change the current working directory on the FTP server.
- BPNG\_TMPBUS\_COPYRDB\_ERROR** Failed to copy the reference data base to the logger's tmp directory.
- BPNG\_TMPBUS\_SEND\_MSG\_ERROR** Failed to send a TMP bus request message.
- BPNG\_TMPBUS\_REQUEST\_ERROR** The TMP bus request execution failed.
- BPNG\_FAILED\_TO\_CREATE\_LOCAL\_FILE\_OR\_DIRECTORY** Failed to create local file or directory.
- BPNG\_LOCAL\_PATH\_NOT\_FOUND** Local path not found.
- BPNG\_READ\_LOCAL\_FILE\_ERROR** Failed to read local file.
- BPNG\_WRITE\_LOCAL\_FILE\_ERROR** Failed to write local file.
- BPNG\_FILE\_EXISTS\_ERROR** Local file already exists.
- BPNG\_DIR\_EXISTS\_ERROR** Local directory already exists.
- BPNG\_TARGET\_PATH\_TOO\_LONG** Specified path exceeds the max. valid length (e.g. 260 for Windows systems)
- BPNG\_ZIP\_EXCEEDS\_FATFS\_MAX** ZIP file exceeds max size for FAT32 file systems.
- BPNG\_XML\_PARSER\_ERROR** Error while parsing xml file.
- BPNG\_INITIALISATION\_ERROR** BPNGClient instance is not initialised or with the wrong function. Use [IBPNGClient::initOnline](#) for data download or conversion directly from the device and [IBPNGClient::iniOffline](#) for data conversion from an offline data set.
- BPNG\_RDB\_SQLITE\_QUERY\_ERROR** Error when trying to read data from the rdb.
- BPNG\_RDB\_OPEN\_FAILED** Failed to open the reference data base.
- BPNG\_CONVERSION\_ERRORS** Multiple conversion errors. Use [IBPNGClient::getNumConversionErrors\(\)](#) and [IBPNGClient::getConversionError\(\)](#) for further information
- BPNG\_CONV\_SET\_NOT\_FOUND** The passed conversion set pointer was not created with this [IBPNGClient](#) instance and thus could not be found.
- BPNG\_NOTHING\_TO\_CONVERT** There is no data available that could be converted. Check the specified time/id spans.
- BPNG\_TMT\_FILE\_ID\_ERROR** Invalid TMT/XTMT file id while trying to convert data.
- BPNG\_TMT\_FORMAT\_ERROR\_VERSION** The TMT/XTMT version of the trace file is not supported by this lib version.
- BPNG\_TMT\_FORMAT\_ERROR\_TS** Missing FileTimeMessage in header of TMT/XTMT file.
- BPNG\_INVALID\_MESSAGE\_ERROR** Invalid messages found in trace file(s).
- BPNG\_INVALID\_MESSAGE\_ID** Invalid message id found in trace file(s).
- BPNG\_INVALID\_MESSAGE\_TS** Invalid message ts found in trace file(s).

- BPNG\_INVALID\_MESSAGE\_SUBID** Invalid message sub id found in trace file(s).
- BPNG\_INVALID\_MESSAGE\_LEN** Invalid message length found in trace file(s).
- BPNG\_CONV\_FORMAT\_ERROR** Invalid format assignment or mismatching recorded trace data for the specified conversion format.
- BPNG\_DOWNLOAD\_ERRORS** Multiple download errors. Use [IBPNGClient::getNumDownloadErrors\(\)](#) and [IBPNGClient::getDownloadError\(\)](#) for further information
- BPNG\_NOTHING\_TO\_DOWNLOAD** There is no data available that could be downloaded. Check the specified time/id spans.
- BPNG\_INVALID\_OFFLINE\_SET** Failed to initialise the [IBPNGClient](#) from the passed offline data set.
- BPNG\_PARAMETER\_MISMATCH** currently not used
- BPNG\_FW\_VERSION\_CHECK\_ERROR** The verification of the new firmware at the end of a firmware update failed.
- BPNG\_USER\_CANCELLED** currently not used
- BPNG\_MIN\_VERSION\_ERROR** The current library version does not suffice the the required min version written to [BPNGError::msg](#).
- BPNG\_EXCEPTION** Some kind of unhandled exception was thrown.
- BPNG\_INCOMPATIBLE\_RDB** The logger's or offline data set's RDB-Version is incompatible to this library version.
- BPNG\_UNSPECIFIED\_ERROR** An unspecified error occurred.
- BPNG\_INVALID\_RPC\_COMMAND** if a rpc command for tsl is wrong
- BPNG\_INVALID\_TSL\_CASCDING** if cascading of tsl is invalid
- BPNG\_INCONSISTENT\_TSL\_FWVERSIONS** if fw versions on tsl clusters are inconsistent
- BPNG\_INVALID\_TSL\_CLUSTER** in case of different TSLNetwork IDs

#### 7.1.2.4 enum BPNGLoggerStatus

Logger status.

Enumerator

- LS\_OK** Device is ok.
- LS\_ERROR** Device has at least one active error.
- LS\_NOSYNC** Device is configured as slave but no master is found.
- LS\_WARNING** Device has at least one active warning.
- LS\_FWUPDATE** Firmware update in progress.
- LS\_MEM** Internal storage of device is full. Ring buffer deactivated or full with protected trace files.
- LS\_RING** Internal storage of device is full. Ring buffer is activated.

## 7.1.2.5 enum BPNGWarningCode

Warning codes.

Warnings are notified by listener calls to the function [IBPNGClientListener::onWarning\(\)](#)

Enumerator

- BPNG\_WARNING\_CLOSE\_TRACE\_FILES** no warning Failed to close the current trace files on the logger device when trying to execute [IBPNGClient::initOnline\(\)](#)
- BPNG\_WARNING\_MESSAGES\_NOT\_CONVERTED** In case of protocol mismatch between recorded data and target format or unsupported message sub types, it is possible that some messages can not be converted to the selected format.
- BPNG\_WARNING\_NO\_ESO\_TRACE** ethernet data for eso trace conversion is not logged in eso trace format
- BPNG\_WARNING\_TSL\_WITH\_DIFFERENT\_TIMEZONES** A TSL cluster with loggers with different time zones is in undefined state. It's not defined which time zone will be used for time zone dependent processes.
- BPNG\_WARNING\_RECOVERING\_FAILED** Recovering trace files from a previous power down failed.

## 7.1.2.6 enum ChannelType

Currently supported interfaces.

Enumerator

- CH\_UNDEFINED** undefined channel type
- OBSOLETE\_CH\_CANLS** CAN low speed interface.
- CH\_CAN** CAN high speed interface.
- CH\_LIN** LIN interface.
- CH\_SERIAL** Serial interface.
- CH\_ETHERNET** Ethernet interface.
- CH\_FLEXRAY** Flexray interface.
- CH\_MOST25\_CTRL** MOST 25 control channel.
- CH\_MOST25\_MDP** MOST 25 data packet channel (MDP)
- CH\_MOST25\_SYNC** MOST 25 synchronous channel (streaming data)
- CH\_MOST150\_CTRL** MOST 150 control channel.
- CH\_MOST150\_MDP** MOST 150 data packet channel (MDP)
- CH\_MOST150\_MEP** MOST 150 ethernet packet channel (MEP)
- CH\_MOST150\_STREAM** MOST 150 synchronous channel (streaming data)
- CH\_ANALOG\_IN** Analog in.
- CH\_DIGITAL\_IN** Digital in.
- CH\_CAMERA** Camera channel.
- CH\_CCPXCP** CCP XCP.
- CH\_DIAG** Diagnose, currently not used.
- CH\_GPS** Global Positioning System.
- CH\_ECL** Electronic Control Line.
- CH\_TTY** TTY channel for QXDM.

### 7.1.2.7 enum LanguageID

Languages.

ID for specifying the language in that the library handles process and error information. Default language is english.

Enumerator

**BPNG\_GERMAN** english  
**BPNG\_ENGLISH** german

### 7.1.2.8 enum Reason

Enumerator

**R\_UNSUPPORTED\_BIT\_MASK** DBC file don't support bit operations with a bit mask.  
**R\_BIT\_MASK\_OVERLAP** Bit mask is incorrect and cause a overlap with at least one other signal.  
**R\_UNSUPPORTED\_COMPU\_TAB** DBC file don't support all compu tab types; only tab will ignored, not the signal itself!  
**R\_FORBIDDEN\_TAB\_VALUE** DBC file don't support all possible values of a compu tab; only tab will ignored, not the signal itself!  
**R\_UNKNOWN** Unknown reason.

## 7.2 BPNGLoggerDetector.hh File Reference

Logger Detector Sample.

```
#include "IBPNGClient.h"  
#include "IBPNGClientListener.h"  
#include <vector>  
#include <string>  
#include <iostream>
```

### Classes

- struct [BPNGDevice](#)
- class [TSLCluster](#)
- class [BPNGLoggerDetector](#)

### 7.2.1 Detailed Description

Logger Detector Sample.

## 7.3 IBPNGClient.h File Reference

Interface class for the BPNGClient DLL.

```
#include "BPNGDefines.h"
#include "RdbDefines.h"
#include "IClientProperties.h"
#include "IBPNGClientListener.h"
```

### Classes

- struct [IBPNGClient](#)  
*Interface class for the Telemotive Client Library.*

### Functions

- DECLDIR [IBPNGClient](#) \*WINAPI [getBPNGClient](#) (const char \*name="")  
*Factory function that creates instances of BPNGClient giving away ownership.*
- DECLDIR [IBPNGClient](#) \*WINAPI [getTSLClient](#) (int numTSLMember)  
*Factory function returning an [IBPNGClient](#) instance for working with a TSL logger cluster.*
- DECLDIR [BPNGErrCode](#) WINAPI [getNumTSLMemberFromOfflineDataSet](#) (const char \*offline-Path, int \*numMember)
- DECLDIR void WINAPI [setTempDir](#) (const char \*tmp)  
*Sets the directory where all temporary files are created. If not called, the default system's tmp dir is used.*
- DECLDIR const char \*WINAPI [getTempDir](#) ()
- DECLDIR void WINAPI [setLanguageID](#) ([LanguageID](#) id)  
*Sets the language for status messages.*
- DECLDIR [IConversionSet](#) \*WINAPI [createNewConversionSet](#) ()  
*returns a new created conversionset*
- DECLDIR void WINAPI [freeConversionSetMemory](#) ([IConversionSet](#) \*convSet)
- DECLDIR [IClientProperties](#) \*WINAPI [createNewClientProperties](#) ()
- DECLDIR void WINAPI [freeClientPropertiesMemory](#) ([IClientProperties](#) \*prop)
- DECLDIR void WINAPI [writeLogFile](#) (const char \*path, int maxSizeInByte, int numBackup-Files)
- DECLDIR void WINAPI [writeLogToCout](#) (bool flag)
- DECLDIR void WINAPI [writeLogToDebugView](#) (bool flag)
- DECLDIR void WINAPI [addLogListener](#) ([onLogRequest](#) logFunc)  
*Adds a log listener to the library.*
- DECLDIR void WINAPI [removeLogListener](#) ([onLogRequest](#) logFunc)  
*Removes a log listener from the library.*

### 7.3.1 Detailed Description

Interface class for the BPNGClient DLL.

**Author**

Markus van Pinxteren

**Date**

21.04.2010

**7.3.2 Function Documentation****7.3.2.1 DECLDIR void WINAPI addLogListener ( onLogRequest *logFunc* )**

Adds a log listener to the library.

If you want to receive the debug outputs from the client library, you can set a log listener to the lib. All set listeners get the log outputs from all BPNGClient instances.

All log outputs are forwarded to the registered listeners by calling the onLogRequest function that was added.

**See Also**

[onLogRequest](#)

**7.3.2.2 DECLDIR IClientProperties\* WINAPI createNewClientProperties ( )**

After modifying the properties, you can set them to an instance of [IBPNGClient](#) with setClientProperties();

**See Also**

[IClientProperties](#), setClientProperties()

**7.3.2.3 DECLDIR void WINAPI freeClientPropertiesMemory ( IClientProperties \* *prop* )**

To free memory of [IClientProperties](#), use this method. Otherwise the memory will be freed when detaching the DLL from process. Never call any function of an [IClientProperties](#) pointer after passing the pointer to the freeClientPropertiesMemory function. This would cause a heap corruptions.

**7.3.2.4 DECLDIR void WINAPI freeConversionSetMemory ( IConversionSet \* *convSet* )**

To free memory of a [IConversionSet](#), use this method. Otherwise the memory will be freed when detaching the DLL from process. Never call any function of an [IConversionSet](#) after passing the pointer to the freeConversionSetMemory function. This would cause a heap corruptions.

**7.3.2.5 DECLDIR IBPNGClient\* WINAPI getBPNGClient ( const char \* *name* = "" )**

Factory function that creates instances of BPNGClient giving away ownership.

The instance is created on the heap and the allocated memory must be freed by the calling application. You can pass a name to this function. This will be the name of the created instance.

See Also

[IBPNGClient::release\(\)](#), [IBPNGClient::getInstanceName\(\)](#)

**7.3.2.6 DECLDIR BPNGErrCode WINAPI getNumTSLMemberFromOfflineDataSet ( const char \* offlinePath, int \* numMember )**

Read out the number of TSL members from a offline data set. Needed for [getTSLClient\(int num-TSLMember\)](#).

**7.3.2.7 DECLDIR IBPNGClient\* WINAPI getTSLClient ( int numTSLMember )**

Factory function returning an [IBPNGClient](#) instance for working with a TSL logger cluster.

The instance is created on the heap and the allocated memory must be freed by the calling application. You must pass the number of devices in TSL as argument. For online operations you can use the number of devices in a TSL chain from logger detection. On offline operations you can use the function [getNumTSLMemberFromOfflineDataSet\(const char\\* offlinePath, int\\* numMember\)](#) to get the number of TSL members.

See Also

[BPNGLoggerDetector](#)  
[TSLCluster](#)  
[IBPNGClient::release\(\)](#)

**7.3.2.8 DECLDIR void WINAPI writeLogFile ( const char \* path, int maxSizeInByte, int numBackupFiles )**

From version 2.1.1 on the client library doesn't write log messages to std::cout by default. The lib actually doesn't write a log at all unless this function is called. A log file is created under the passed *path*. If the file already exists, the logs will be appended. The file will be closed when the DLL is detached from the process.

**7.3.2.9 DECLDIR void WINAPI writeLogToCout ( bool flag )**

The library's log output can also be written to std::cout. If this is required activate cout log with this function. Default is no cout output.

## 7.4 IBPNGClientListener.h File Reference

Interface class for the BPNGClient listener.

```
#include <iostream>
#include "BPNGDefines.h"
```

### Classes

- struct [IBPNGClientListener](#)

### 7.4.1 Detailed Description

Interface class for the BPNGClient listener.

**Author**

Markus van Pinxteren

**Date**

12.05.2010

## 7.5 IClientProperties.h File Reference

Interface for client properties.

```
#include "BPNGDefines.h"
```

**Classes**

- struct [IClientProperties](#)

*The [IClientProperties](#) interface replaces the deprecated [ClientProperties](#) struct.*

### 7.5.1 Detailed Description

Interface for client properties.

**Author**

Markus van Pinxteren

**Date**

20.03.2014

## 7.6 RdbDefines.h File Reference

Public interfaces for Telemotive Reference Database access.

```
#include <atom-config.h>
#include <cstdlib>
#include <stdint.h>
```



## Classes

- struct [IRdbEvent](#)
- struct [IRdbEventList](#)
- struct [IRdbTraceBlock](#)
- struct [IRdbTraceBlockList](#)

## Macros

- `#define WINAPI`
- `#define ATOM_CLIENTLIB 0`

### 7.6.1 Detailed Description

Public interfaces for Telemotive Reference Database access.

## 7.7 RdbEventList.hh File Reference

[IRdbEvent](#) wrapper.

```
#include <vector>
#include <string>
#include "BPNGDefines.h"
```

## Classes

- struct [RdbEvent2](#)  
*Implementation class for a wrapper of [IRdbEvent](#) using STL classes.*
- class [RdbEventList](#)  
*Implementation class for a wrapper of [IRdbEventList](#) using STL classes.*

### 7.7.1 Detailed Description

[IRdbEvent](#) wrapper.

# Index

- addAnalogPortSettings
  - IClientProperties, [55](#)
- addChannel
  - IConversionSet, [58](#)
- addDevice
  - TSLCluster, [69](#)
- addLogListener
  - IBPNGClient.h, [83](#)
- addRdbldRange
  - IConversionSet, [58](#)
- addTimeSpan
  - IConversionSet, [59](#)
- assignDBCFile
  - IBPNGClient, [34](#)
- BPNG\_AMBIGUOUS\_IP
  - BPNGDefines.h, [77](#)
- BPNG\_CONNECT\_FTP\_FAILED
  - BPNGDefines.h, [77](#)
- BPNG\_CONNECT\_TMPBUS\_FAILED
  - BPNGDefines.h, [77](#)
- BPNG\_CONV\_FORMAT\_ERROR
  - BPNGDefines.h, [79](#)
- BPNG\_CONV\_SET\_NOT\_FOUND
  - BPNGDefines.h, [78](#)
- BPNG\_CONVERSION\_ERRORS
  - BPNGDefines.h, [78](#)
- BPNG\_DIR\_EXISTS\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_DOWNLOAD\_ERRORS
  - BPNGDefines.h, [79](#)
- BPNG\_ENGLISH
  - BPNGDefines.h, [81](#)
- BPNG\_EXCEPTION
  - BPNGDefines.h, [79](#)
- BPNG\_FAILED\_TO\_CONNECT\_STREAMING
  - BPNGDefines.h, [77](#)
- BPNG\_FAILED\_TO\_CREATE\_LOCAL\_FILE\_OR\_DIRECTORY
  - BPNGDefines.h, [78](#)
- BPNG\_FILE\_EXISTS\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_CHANGE\_CWD\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_CREATE\_REMOTE\_DIR\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_LOGIN\_FAILED
  - BPNGDefines.h, [77](#)
- BPNG\_FTP\_NOT\_CONNECTED
  - BPNGDefines.h, [77](#)
- BPNG\_FTP\_READ\_REMOTE\_FILE\_ERROR
  - BPNGDefines.h, [77](#)
- BPNG\_FTP\_REMOTE\_PATH\_NOT\_FOUND
  - BPNGDefines.h, [77](#)
- BPNG\_FTP\_REMOVE\_REMOTE\_DIR\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_REMOVE\_REMOTE\_FILE\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_SERVER\_NOT\_FOUND
  - BPNGDefines.h, [77](#)
- BPNG\_FTP\_TRANSFER\_USER\_CANCELED
  - BPNGDefines.h, [78](#)
- BPNG\_FTP\_WRITE\_REMOTE\_FILE\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_FW\_VERSION\_CHECK\_ERROR
  - BPNGDefines.h, [79](#)
- BPNG\_GERMAN
  - BPNGDefines.h, [81](#)
- BPNG\_INCOMPATIBLE\_RDB
  - BPNGDefines.h, [79](#)
- BPNG\_INCONSISTENT\_TSL\_FWVERSIONS
  - BPNGDefines.h, [79](#)
- BPNG\_INITIALISATION\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_INVALID\_MESSAGE\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_INVALID\_MESSAGE\_ID
  - BPNGDefines.h, [78](#)
- BPNG\_INVALID\_MESSAGE\_LEN
  - BPNGDefines.h, [79](#)
- BPNG\_INVALID\_MESSAGE\_SUBID
  - BPNGDefines.h, [78](#)
- BPNG\_INVALID\_MESSAGE\_TS
  - BPNGDefines.h, [78](#)
- BPNG\_INVALID\_OFFLINE\_SET

- BPNGDefines.h, [79](#)
- BPNG\_INVALID\_RPC\_COMMAND
  - BPNGDefines.h, [79](#)
- BPNG\_INVALID\_TSL\_CASCDING
  - BPNGDefines.h, [79](#)
- BPNG\_INVALID\_TSL\_CLUSTER
  - BPNGDefines.h, [79](#)
- BPNG\_LOCAL\_PATH\_NOT\_FOUND
  - BPNGDefines.h, [78](#)
- BPNG\_LOGGER\_NOT\_FOUND
  - BPNGDefines.h, [77](#)
- BPNG\_MIN\_VERSION\_ERROR
  - BPNGDefines.h, [79](#)
- BPNG\_NOERR
  - BPNGDefines.h, [77](#)
- BPNG\_NOT\_CONNECTED
  - BPNGDefines.h, [77](#)
- BPNG\_NOTHING\_TO\_CONVERT
  - BPNGDefines.h, [78](#)
- BPNG\_NOTHING\_TO\_DOWNLOAD
  - BPNGDefines.h, [79](#)
- BPNG\_PARAMETER\_MISMATCH
  - BPNGDefines.h, [79](#)
- BPNG\_RDB\_OPEN\_FAILED
  - BPNGDefines.h, [78](#)
- BPNG\_RDB\_SQLITE\_QUERY\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_READ\_LOCAL\_FILE\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_TARGET\_PATH\_TOO\_LONG
  - BPNGDefines.h, [78](#)
- BPNG\_TMPBUS\_COPYRDB\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_TMPBUS\_NOT\_CONNECTED
  - BPNGDefines.h, [77](#)
- BPNG\_TMPBUS\_REQUEST\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_TMPBUS\_SEND\_MSG\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_TMT\_FILE\_ID\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_TMT\_FORMAT\_ERROR\_TS
  - BPNGDefines.h, [78](#)
- BPNG\_TMT\_FORMAT\_ERROR\_VERSION
  - BPNGDefines.h, [78](#)
- BPNG\_UNSPECIFIED\_ERROR
  - BPNGDefines.h, [79](#)
- BPNG\_USER\_CANCELLED
  - BPNGDefines.h, [79](#)
- BPNG\_WARNING\_CLOSE\_TRACE\_FILES
  - BPNGDefines.h, [80](#)
- BPNG\_WARNING\_MESSAGES\_NOT\_CONVERTED
  - BPNGDefines.h, [80](#)
- BPNG\_WARNING\_NO\_ESO\_TRACE
  - BPNGDefines.h, [80](#)
- BPNG\_WARNING\_RECOVERING\_FAILED
  - BPNGDefines.h, [80](#)
- BPNG\_WARNING\_TSL\_WITH\_DIFFERENT\_TIMEZONES
  - BPNGDefines.h, [80](#)
- BPNG\_WRITE\_LOCAL\_FILE\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_XML\_PARSER\_ERROR
  - BPNGDefines.h, [78](#)
- BPNG\_ZIP\_EXCEEDS\_FATFS\_MAX
  - BPNGDefines.h, [78](#)
- BPNGDefines.h
  - BPNG\_AMBIGUOUS\_IP, [77](#)
  - BPNG\_CONNECT\_FTP\_FAILED, [77](#)
  - BPNG\_CONNECT\_TMPBUS\_FAILED, [77](#)
  - BPNG\_CONV\_FORMAT\_ERROR, [79](#)
  - BPNG\_CONV\_SET\_NOT\_FOUND, [78](#)
  - BPNG\_CONVERSION\_ERRORS, [78](#)
  - BPNG\_DIR\_EXISTS\_ERROR, [78](#)
  - BPNG\_DOWNLOAD\_ERRORS, [79](#)
  - BPNG\_ENGLISH, [81](#)
  - BPNG\_EXCEPTION, [79](#)
  - BPNG\_FAILED\_TO\_CONNECT\_STREAMING, [77](#)
  - BPNG\_FAILED\_TO\_CREATE\_LOCAL\_FILE\_OR\_DIRECTORY, [78](#)
  - BPNG\_FILE\_EXISTS\_ERROR, [78](#)
  - BPNG\_FTP\_CHANGE\_CWD\_ERROR, [78](#)
  - BPNG\_FTP\_CREATE\_REMOTE\_DIR\_ERROR, [78](#)
  - BPNG\_FTP\_LOGIN\_FAILED, [77](#)
  - BPNG\_FTP\_NOT\_CONNECTED, [77](#)
  - BPNG\_FTP\_READ\_REMOTE\_FILE\_ERROR, [77](#)
  - BPNG\_FTP\_REMOTE\_PATH\_NOT\_FOUND, [77](#)
  - BPNG\_FTP\_REMOVE\_REMOTE\_DIR\_ERROR, [78](#)
  - BPNG\_FTP\_REMOVE\_REMOTE\_FILE\_ERROR, [78](#)
  - BPNG\_FTP\_SERVER\_NOT\_FOUND, [77](#)
  - BPNG\_FTP\_TRANSFER\_USER\_CANCELLED, [78](#)
  - BPNG\_FTP\_WRITE\_REMOTE\_FILE\_ERROR, [78](#)
  - BPNG\_FW\_VERSION\_CHECK\_ERROR, [79](#)
  - BPNG\_GERMAN, [81](#)
  - BPNG\_INCOMPATIBLE\_RDB, [79](#)
  - BPNG\_INCONSISTENT\_TSL\_FWVERSIONS, [79](#)
  - BPNG\_INITIALISATION\_ERROR, [78](#)
  - BPNG\_INVALID\_MESSAGE\_ERROR, [78](#)

- BPNG\_INVALID\_MESSAGE\_ID, 78
- BPNG\_INVALID\_MESSAGE\_LEN, 79
- BPNG\_INVALID\_MESSAGE\_SUBID, 78
- BPNG\_INVALID\_MESSAGE\_TS, 78
- BPNG\_INVALID\_OFFLINE\_SET, 79
- BPNG\_INVALID\_RPC\_COMMAND, 79
- BPNG\_INVALID\_TSL\_CASCDING, 79
- BPNG\_INVALID\_TSL\_CLUSTER, 79
- BPNG\_LOCAL\_PATH\_NOT\_FOUND, 78
- BPNG\_LOGGER\_NOT\_FOUND, 77
- BPNG\_MIN\_VERSION\_ERROR, 79
- BPNG\_NOERR, 77
- BPNG\_NOT\_CONNECTED, 77
- BPNG\_NOTHING\_TO\_CONVERT, 78
- BPNG\_NOTHING\_TO\_DOWNLOAD, 79
- BPNG\_PARAMETER\_MISMATCH, 79
- BPNG\_RDB\_OPEN\_FAILED, 78
- BPNG\_RDB\_SQLITE\_QUERY\_ERROR, 78
- BPNG\_READ\_LOCAL\_FILE\_ERROR, 78
- BPNG\_TARGET\_PATH\_TOO\_LONG, 78
- BPNG\_TMPBUS\_COPYRDB\_ERROR, 78
- BPNG\_TMPBUS\_NOT\_CONNECTED, 77
- BPNG\_TMPBUS\_REQUEST\_ERROR, 78
- BPNG\_TMPBUS\_SEND\_MSG\_ERROR, 78
- BPNG\_TMT\_FILE\_ID\_ERROR, 78
- BPNG\_TMT\_FORMAT\_ERROR\_TS, 78
- BPNG\_TMT\_FORMAT\_ERROR\_VERSION, 78
- BPNG\_UNSPECIFIED\_ERROR, 79
- BPNG\_USER\_CANCELLED, 79
- BPNG\_WARNING\_CLOSE\_TRACE\_FILES, 80
- BPNG\_WARNING\_MESSAGES\_NOT\_CONVERTED, 80
- BPNG\_WARNING\_NO\_ESO\_TRACE, 80
- BPNG\_WARNING\_RECOVERING\_FAILED, 80
- BPNG\_WARNING\_TSL\_WITH\_DIFFERENT\_TIMEZONES, 80
- BPNG\_WRITE\_LOCAL\_FILE\_ERROR, 78
- BPNG\_XML\_PARSER\_ERROR, 78
- BPNG\_ZIP\_EXCEEDS\_FATFS\_MAX, 78
- BR\_FDB\_RDB, 76
- BR\_FULL\_ALL\_TRACES, 77
- BR\_FULL\_TIMESPAN\_TRACES, 77
- BR\_FULL\_WO\_TRACES, 76
- BR\_ONLY\_CLIENT, 77
- BR\_ONLY\_LOGS, 76
- CH\_ANALOG\_IN, 80
- CH\_CAMERA, 80
- CH\_CAN, 80
- CH\_CCPXCP, 80
- CH\_DIAG, 80
- CH\_DIGITAL\_IN, 80
- CH\_ECL, 80
- CH\_ETHERNET, 80
- CH\_FLEXRAY, 80
- CH\_GPS, 80
- CH\_LIN, 80
- CH\_MOST150\_CTRL, 80
- CH\_MOST150\_MDP, 80
- CH\_MOST150\_MEP, 80
- CH\_MOST150\_STREAM, 80
- CH\_MOST25\_CTRL, 80
- CH\_MOST25\_MDP, 80
- CH\_MOST25\_SYNC, 80
- CH\_SERIAL, 80
- CH\_TTY, 80
- CH\_UNDEFINED, 80
- DEV\_BP2, 77
- DEV\_BP2\_V1X, 77
- DEV\_BP2\_V2X, 77
- DEV\_BP\_REMOTE, 77
- DEV\_BPMINI, 77
- DEV\_RC\_TOUCH, 77
- DEV\_TSL, 77
- LS\_ERROR, 79
- LS\_FWUPDATE, 79
- LS\_MEM, 79
- LS\_NOSYNC, 79
- LS\_OK, 79
- LS\_RING, 79
- LS\_WARNING, 79
- OBSOLETE\_CH\_CANLS, 80
- R\_BIT\_MASK\_OVERLAP, 81
- R\_FORBIDDEN\_TAB\_VALUE, 81
- R\_UNKNOWN, 81
- R\_UNSUPPORTED\_BIT\_MASK, 81
- R\_UNSUPPORTED\_COMPU\_TAB, 81
- BR\_FDB\_RDB
  - BPNGDefines.h, 76
- BR\_FULL\_ALL\_TRACES
  - BPNGDefines.h, 77
- BR\_FULL\_TIMESPAN\_TRACES
  - BPNGDefines.h, 77
- BR\_FULL\_WO\_TRACES
  - BPNGDefines.h, 76
- BR\_ONLY\_CLIENT
  - BPNGDefines.h, 77
- BR\_ONLY\_LOGS
  - BPNGDefines.h, 76
- BUGREPORT
  - TSLCluster, 69
- BPNGBugreportMode
  - BPNGDefines.h, 76
- BPNGDefines.h, 72
  - BPNGBugreportMode, 76
  - BPNGDeviceType, 77

- BPNGErrCode, [77](#)
- BPNGLoggerStatus, [79](#)
- BPNGWarningCode, [79](#)
- ChannelType, [80](#)
- LanguageID, [80](#)
- Reason, [81](#)
- BPNGDevice, [20](#)
  - status, [21](#)
- BPNGDeviceType
  - BPNGDefines.h, [77](#)
- BPNGErrCode
  - BPNGDefines.h, [77](#)
- BPNGError, [21](#)
- BPNGLoggerDetector, [21](#)
  - BPNGLoggerDetector, [23](#)
  - BPNGLoggerDetector, [23](#)
  - getLoggerList, [23](#)
  - getOverwritingPermission, [23](#)
  - getTSLs, [23](#)
  - onBPNGDeviceDetected, [24](#)
  - onBPNGDeviceDisappeared, [24](#)
  - onBPNGDeviceStateChange, [24](#)
  - onConversionStart, [24](#)
  - onCriticalDiskSpace, [24](#)
  - onDownloadStart, [25](#)
  - onGetLogReportProgress, [25](#)
  - onInvalidPwConfigFound, [25](#)
  - onLogInDataRequired, [25](#)
  - onProgressConversion, [26](#)
  - onProgressDataDownload, [26](#)
  - onStatusMessage, [26](#)
  - onTargetPathTooLong, [27](#)
  - onWarning, [27](#)
- BPNGLoggerDetector.hh, [81](#)
- BPNGLoggerStatus
  - BPNGDefines.h, [79](#)
- BPNGWarningCode
  - BPNGDefines.h, [79](#)
- begin
  - TSLCluster, [69](#)
- CH\_ANALOG\_IN
  - BPNGDefines.h, [80](#)
- CH\_CAMERA
  - BPNGDefines.h, [80](#)
- CH\_CAN
  - BPNGDefines.h, [80](#)
- CH\_CCPXCP
  - BPNGDefines.h, [80](#)
- CH\_DIAG
  - BPNGDefines.h, [80](#)
- CH\_DIGITAL\_IN
  - BPNGDefines.h, [80](#)
- CH\_ECL
  - BPNGDefines.h, [80](#)
- CH\_ETHERNET
  - BPNGDefines.h, [80](#)
- CH\_FLEXRAY
  - BPNGDefines.h, [80](#)
- CH\_GPS
  - BPNGDefines.h, [80](#)
- CH\_LIN
  - BPNGDefines.h, [80](#)
- CH\_MOST150\_CTRL
  - BPNGDefines.h, [80](#)
- CH\_MOST150\_MDP
  - BPNGDefines.h, [80](#)
- CH\_MOST150\_MEP
  - BPNGDefines.h, [80](#)
- CH\_MOST150\_STREAM
  - BPNGDefines.h, [80](#)
- CH\_MOST25\_CTRL
  - BPNGDefines.h, [80](#)
- CH\_MOST25\_MDP
  - BPNGDefines.h, [80](#)
- CH\_MOST25\_SYNC
  - BPNGDefines.h, [80](#)
- CH\_SERIAL
  - BPNGDefines.h, [80](#)
- CH\_TTY
  - BPNGDefines.h, [80](#)
- CH\_UNDEFINED
  - BPNGDefines.h, [80](#)
- CONFIG
  - TSLCluster, [69](#)
- CONVERSION
  - TSLCluster, [69](#)
- CANPseudoMessagesProperties, [27](#)
- ChannelType
  - BPNGDefines.h, [80](#)
- ClientProperties, [28](#)
- CommonProperties, [29](#)
- connectLogger
  - IBPNGClient, [34](#)
- ConnectionType
  - TSLCluster, [69](#)
- ConversionProperties, [29](#)
- convertData
  - IBPNGClient, [34](#)
- createNewClientProperties
  - IBPNGClient.h, [83](#)
- DEV\_BP2
  - BPNGDefines.h, [77](#)
- DEV\_BP2\_V1X
  - BPNGDefines.h, [77](#)
- DEV\_BP2\_V2X
  - BPNGDefines.h, [77](#)

DEV\_BP\_REMOTE  
     BPNGDefines.h, 77  
 DEV\_BPMINI  
     BPNGDefines.h, 77  
 DEV\_RC\_TOUCH  
     BPNGDefines.h, 77  
 DEV\_TSL  
     BPNGDefines.h, 77  
 DOWNLOAD  
     TSLCluster, 69  
 DataSpan, 30  
 deleteAllData  
     IBPNGClient, 35  
 deleteData  
     IBPNGClient, 35  
 deviceType  
     OnlineLoggerInfo, 66  
 downloadBugReport  
     IBPNGClient, 36  
 downloadDataSpans  
     IBPNGClient, 36  
 end  
     TSLCluster, 69  
 FW\_UPDATE  
     TSLCluster, 69  
 filterSignals  
     IBPNGClient, 37  
 filterSignalsFromOfflineData  
     IBPNGClient, 37  
 flashDeviceLED  
     IBPNGClient, 37  
 freeClientPropertiesMemory  
     IBPNGClient.h, 83  
 freeConversionSetMemory  
     IBPNGClient.h, 83  
 getAvailableFormats  
     IBPNGClient, 37  
 getBPNGClient  
     IBPNGClient.h, 83  
 getClientProperties  
     IBPNGClient, 38  
 getConfig  
     IBPNGClient, 38  
 getConnectionString  
     TSLCluster, 70  
 getConversionError  
     IBPNGClient, 38  
 getDeviceName  
     IBPNGClient, 38  
 getDownloadError  
     IBPNGClient, 39  
 getEventList  
     IBPNGClient, 39  
 getFalseMeasureSignals  
     IBPNGClient, 39  
 getLastError  
     IBPNGClient, 39  
 getLoggerChannels  
     IBPNGClient, 40  
 getLoggerList  
     BPNGLoggerDetector, 23  
 getMemoryFillLevel  
     IBPNGClient, 40  
 getNumConversionErrors  
     IBPNGClient, 40  
 getNumDownloadErrors  
     IBPNGClient, 40  
 getNumMembers  
     TSLCluster, 70  
 getNumTSLMemberFromOfflineDataSet  
     IBPNGClient.h, 84  
 getOverwritingPermission  
     BPNGLoggerDetector, 23  
     IBPNGClientListener, 48  
 getProperties  
     IBPNGClient, 41  
 getReferenceDataBasePath  
     IBPNGClient, 41  
 getTSLClient  
     IBPNGClient.h, 84  
 getTSLName  
     TSLCluster, 70  
 getTSLs  
     BPNGLoggerDetector, 23  
 getTraceBlockList  
     IBPNGClient, 41  
 IBPNGClient, 30  
     assignDBCFile, 34  
     connectLogger, 34  
     convertData, 34  
     deleteAllData, 35  
     deleteData, 35  
     downloadBugReport, 36  
     downloadDataSpans, 36  
     filterSignals, 37  
     filterSignalsFromOfflineData, 37  
     flashDeviceLED, 37  
     getAvailableFormats, 37  
     getClientProperties, 38  
     getConfig, 38  
     getConversionError, 38  
     getDeviceName, 38  
     getDownloadError, 39  
     getEventList, 39

- getFalseMeasureSignals, 39
- getLastError, 39
- getLoggerChannels, 40
- getMemoryFillLevel, 40
- getNumConversionErrors, 40
- getNumDownloadErrors, 40
- getProperties, 41
- getReferenceDataBasePath, 41
- getTraceBlockList, 41
- initOffline, 41
- initOnline, 42
- keepLoggerAlive, 42
- reconfigLogger, 43
- release, 43
- removeAllLicenses, 43
- restartDevice, 43
- scanNetworkForLogger, 44
- setClientProperties, 44
- setDefaultConfig, 44
- setInfoEvent, 44
- setMarker, 45
- setProperties, 45
- setTime, 45
- shutdownDevice, 45
- synchronizeRdb, 45
- updateFirmware, 45
- updateLicenses, 46
- IBPNGClient.h, 82
  - addLogListener, 83
  - createNewClientProperties, 83
  - freeClientPropertiesMemory, 83
  - freeConversionSetMemory, 83
  - getBPNGClient, 83
  - getNumTSLMemberFromOfflineDataSet, 84
  - getTSLClient, 84
  - writeLogFile, 84
  - writeLogToCout, 84
- IBPNGClientListener, 46
  - getOverwritingPermission, 48
  - onBPNGDeviceDetected, 48
  - onBPNGDeviceDisappeared, 48
  - onBPNGDeviceStateChange, 48
  - onConversionStart, 48
  - onCriticalDiskSpace, 48
  - onDownloadStart, 49
  - onGetLogReportProgress, 49
  - onInvalidPwConfigFound, 49
  - onLogInDataRequired, 50
  - onProgressConversion, 50
  - onProgressDataDownload, 50
  - onStatusMessage, 51
  - onTargetPathTooLong, 51
  - onWarning, 51
- IBPNGClientListener.h, 84
- IChannel, 51
- IChannelList, 52
- IClientProperties, 53
  - addAnalogPortSettings, 55
  - setCANPseudoMsgTimeStampProperties, 55
  - setCANPseudoMsgTriggerProperties, 56
  - setCommonProperties, 56
  - setMOSTPseudoMsgProperties, 57
- IClientProperties.h, 85
- IConversionSet, 57
  - addChannel, 58
  - addRdbldRange, 58
  - addTimeSpan, 59
- IFalseMeasureSignal, 59
- IFalseMeasureSignalList, 60
- IFormatInfo, 60
- IFormatList, 61
- IRdbEvent, 61
- IRdbEventList, 62
- IRdbTraceBlock, 62
- IRdbTraceBlockList, 62
- ITesttoolsChannel, 62
- ITesttoolsChannelList, 63
- initOffline
  - IBPNGClient, 41
- initOnline
  - IBPNGClient, 42
- keepLoggerAlive
  - IBPNGClient, 42
- LS\_ERROR
  - BPNGDefines.h, 79
- LS\_FWUPDATE
  - BPNGDefines.h, 79
- LS\_MEM
  - BPNGDefines.h, 79
- LS\_NOSYNC
  - BPNGDefines.h, 79
- LS\_OK
  - BPNGDefines.h, 79
- LS\_RING
  - BPNGDefines.h, 79
- LS\_WARNING
  - BPNGDefines.h, 79
- LanguageID
  - BPNGDefines.h, 80
- LogInData, 63
- loggerStatus
  - OnlineLoggerInfo, 66
- MOSTPpseudoMessagesProperties, 64
- MemoryFillLevel, 64
- OBSOLETE\_CH\_CANLS



- BPNGDefines.h, 80
- onBPNGDeviceDetected
  - BPNGLoggerDetector, 24
  - IBPNGClientListener, 48
- onBPNGDeviceDisappeared
  - BPNGLoggerDetector, 24
  - IBPNGClientListener, 48
- onBPNGDeviceStateChange
  - BPNGLoggerDetector, 24
  - IBPNGClientListener, 48
- onConversionStart
  - BPNGLoggerDetector, 24
  - IBPNGClientListener, 48
- onCriticalDiskSpace
  - BPNGLoggerDetector, 24
  - IBPNGClientListener, 48
- onDownloadStart
  - BPNGLoggerDetector, 25
  - IBPNGClientListener, 49
- onGetLogReportProgress
  - BPNGLoggerDetector, 25
  - IBPNGClientListener, 49
- onInvalidPwConfigFound
  - BPNGLoggerDetector, 25
  - IBPNGClientListener, 49
- onLoginDataRequired
  - BPNGLoggerDetector, 25
  - IBPNGClientListener, 50
- onProgressConversion
  - BPNGLoggerDetector, 26
  - IBPNGClientListener, 50
- onProgressDataDownload
  - BPNGLoggerDetector, 26
  - IBPNGClientListener, 50
- onStatusMessage
  - BPNGLoggerDetector, 26
  - IBPNGClientListener, 51
- onTargetPathTooLong
  - BPNGLoggerDetector, 27
  - IBPNGClientListener, 51
- onWarning
  - BPNGLoggerDetector, 27
  - IBPNGClientListener, 51
- OnlineLoggerInfo, 65
  - deviceType, 66
  - loggerStatus, 66
- print
  - TSLCluster, 70
- R\_BIT\_MASK\_OVERLAP
  - BPNGDefines.h, 81
- R\_FORBIDDEN\_TAB\_VALUE
  - BPNGDefines.h, 81
- R\_UNKNOWN
  - BPNGDefines.h, 81
- R\_UNSUPPORTED\_BIT\_MASK
  - BPNGDefines.h, 81
- R\_UNSUPPORTED\_COMPU\_TAB
  - BPNGDefines.h, 81
- RdbDefines.h, 85
- RdbEvent2, 66
- RdbEventList, 67
- RdbEventList.hh, 86
- Reason
  - BPNGDefines.h, 81
- reconfigLogger
  - IBPNGClient, 43
- release
  - IBPNGClient, 43
- removeAllLicenses
  - IBPNGClient, 43
- restartDevice
  - IBPNGClient, 43
- scanNetworkForLogger
  - IBPNGClient, 44
- setCANPseudoMsgTimeStampProperties
  - IClientProperties, 55
- setCANPseudoMsgTriggerProperties
  - IClientProperties, 56
- setClientProperties
  - IBPNGClient, 44
- setCommonProperties
  - IClientProperties, 56
- setDefaultConfig
  - IBPNGClient, 44
- setInfoEvent
  - IBPNGClient, 44
- setMOSTPseudoMsgProperties
  - IClientProperties, 57
- setMarker
  - IBPNGClient, 45
- setProperties
  - IBPNGClient, 45
- setTime
  - IBPNGClient, 45
- shutdownDevice
  - IBPNGClient, 45
- status
  - BPNGDevice, 21
- synchronizeRdb
  - IBPNGClient, 45
- TSLCluster
  - BUGREPORT, 69
  - CONFIG, 69
  - CONVERSION, 69



- DOWNLOAD, [69](#)
- FW\_UPDATE, [69](#)
- TSLCluster, [68](#)
  - addDevice, [69](#)
  - begin, [69](#)
  - ConnectionType, [69](#)
  - end, [69](#)
  - getConnectionString, [70](#)
  - getNumMembers, [70](#)
  - getTSLName, [70](#)
  - print, [70](#)
  - TSLCluster, [69](#)
  - TSLCluster, [69](#)
- updateFirmware
  - IBPNGClient, [45](#)
- updateLicenses
  - IBPNGClient, [46](#)
- writeLogFile
  - IBPNGClient.h, [84](#)
- writeLogToCout
  - IBPNGClient.h, [84](#)