PPPP P2P API

1.5.1

API

- Common:
  - PPPP_Initialize, PPPP_DeInitialize
  - PPPP_NetworkDetect
  - PPPP_NetworkDetectByServer
  - PPPP_GetAPIVersion
  - PPPP_QueryDID
  - PPPP_Share_Bandwidth
  - PPPP_Get_ServerIP
- Device:
  - PPPP_Listen
  - PPPP_Listen_Break
  - PPPP_LoginStatus_Check
- Client:
  - PPPP_Connect
  - PPPP_ConnectByServer
  - PPPP_Connect_Break
- Session:
  - PPPP_Check
  - PPPP_Close
- Read / Write data
  - PPPP_Read
  - PPPP_Write
- Check Buffer size
  - PPPP_Check_Buffer

Return Code of API

- >=0: Successful
  - #define ERROR_PPPP_SUCCESSFUL 0
- <0: Some thing wrong
  - #define ERROR_PPPP_NOT_INITIALIZED -1
  - #define ERROR_PPPP_ALREADY_INITIALIZED -2
  - #define ERROR_PPPP_TIME_OUT -3
  - #define ERROR_PPPP_INVALID_ID -4
  - #define ERROR_PPPP_INVALID_PARAMETER -5
  - #define ERROR_PPPP_DEVICE_NOT_ONLINE -6
  - #define ERROR_PPPP_FAIL_TO_RESOLVE_NAME -7
  - #define ERROR_PPPP_INVALID_PREFIX -8
  - #define ERROR_PPPP_ID_OUT_OF_DATE -9
  - #define ERROR_PPPP_NO_RELAY_SERVER_AVAILABLE -10
  - #define ERROR_PPPP_INVALID_SESSION_HANDLE -11
  - #define ERROR_PPPP_SESSION_CLOSED_REMOTE -12
  - #define ERROR_PPPP_SESSION_CLOSED_TIMEOUT -13
  - #define ERROR_PPPP_SESSION_CLOSED_CALLED -14
  - #define ERROR_PPPP_REMOTE_SITE_BUFFER_FULL -15
  - #define ERROR_PPPP_USER_LISTEN_BREAK -16
  - #define ERROR_PPPP_MAX_SESSION -17
  - #define ERROR_PPPP_UDP_PORT_BIND_FAILED -18
  - #define ERROR_PPPP_USER_CONNECT_BREAK -19
  - #define ERROR_PPPP_SESSION_CLOSED_INSUFFICIENT_MEMORY -20
  - #define ERROR_PPPP_INVALID_APILICENSE -21

PPPP_GetAPIVersion

- Function Declare:
  - UINT32 PPPP_GetAPIVersion()
- Description:
  - PPPP_GetAPIVersion: To retrive API version info.
- Parameters:
  - **None**
- Return:
  - 0x01020304 ➔ Version 1.2.3.4

PPPP_Initialize, PPPP_DeInitialize

- Function Declare:
  - PPPP_Initialize(CHAR *Parameter)
  - INT32 PPPP_DeInitialize()
- Description:
  - PPPP_Initialize: To initialize usage of PPPP session module.
  - PPPP_DeInitialize: To free all resource used by PPPP session module.
- Parameters:
  - **Parameter**: The parameter string that tell server information.
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_ALREADY_INITIALIZED
  - ERROR_PPPP_INSUFFICIENT_RESOURCE

PPPP_NetworkDetect

- Function Declare:
  - INT32 PPPP_NetworkDetect(st_PPPP_NetInfo *NetInfo, UINT16 UDP_Port);
  - INT32 PPPP_NetworkDetectByServer(st_PPPP_NetInfo *NetInfo, UINT16 UDP_Port, CHAR *ServerString)
- Description:
  - PPPP_NetworkDetect: To detect network related information.
  - PPPP_NetworkDetectByServer: The same as PPPP_NetworkDetect, but user can specify with which server to perform this function.
- Parameters:
  - **NetInfo** : the structure where network iformation is retrived.
  - **UDP_Port** : Specify the UDP port. if **UDP_Port** =0, a random port will be used.
  - **ServerString**: Encoded string, specifying the server address.
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALID_PARAMETER
  - ERROR_PPPP_UDP_PORT_BIND_FAILED

PPPP_Get_ServerIP

- Function Declare:
  - void PPPP_Get_ServerIP(CHAR *Server1IP, CHAR *Server2IP, CHAR *Server3IP);
- Description:
  - PPPP_Get_ServerIP: To retrive P2P Servers' IP Address.

    Hint: PPPP_Get_ServerIP() shall return correct Server IP, only when PPPP_NetworkDetect(), PPPP_Listen() or PPPP_Connect() is called in advance.
- Parameters:
  - **Server1IP**: IP Address of P2P Serevr 1.
  - **Server2IP**: IP Address of P2P Serevr 2.
  - **Server3IP**: IP Address of P2P Serevr 3.
- Return:
  - NONE

PPPP_QueryDID

- Function Declare:
  - INT32 PPPP_QueryDID(const CHAR* DeviceName, CHAR* DID, INT32 DIDBufSize)
- Description:
  - PPPP_QueryDID: To Query device' DID by Name
- Parameters:
  - **DeviceName**: Name of Device
  - **DID**: The DID of Device
  - **DIDBufSize** : Buffer size of DID
- Return:
  - 0: Query successfully
  - -1: Query Failed

This API use a random UDP port to send/recv query packet.
For better performance, remember the DID for furture usage.

PPPP_Share_Bandwidth

- Function Declare:
  - INT32 PPPP_Share_Bandwidth(CHAR bOnOff)
- Description:
  - PPPP_Share_Bandwidth: Allow devices(call PPPP_Listen) to perform relay service.
- Parameters:
  - **bOnOff** :
    - bOnOff = 0: Not share, or stop sharing (if is on sharing).
    - bOnOff = 1: Allow bandwidth sharing.
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED

PPPP_Listen/PPPP_Listen_Break

- Function Declare:
  - INT32 PPPP_Listen(const CHAR *MyID, UINT32 TimeOut_Sec, UINT16 UDP_Port , CHAR bEnableInternet, const CHAR *APILicense)
  - INT32 PPPP_Listen_Break();
- Description:
  - PPPP_Listen: To login to server and wait until some client to connect with. The calling thread will be blocked, till Client connection or timeout.
  - PPPP_Listen_Break: to break PPPP_Listen
- Parameters:
  - **MyID**: My ID
  - **TimeOut_Sec**: Block until Client connection or Time out in Second. Valid timeout value: 60~86400
  - **UDP_Port** : Specify the UDP port. if **UDP_Port** =0, a random port will be used.
  - **bEnableInternet** : If allow Client connection from Internet.
  - **APILicense**: The License string for using this API. Also, used to define CRCKey.

    case 1: APILicense is like "ABCDE:CRCKey", the CRCKey is the CRC Key string that user set in P2P Server.

    case 2: APILicense is like "ABCDE", Empty CRC Key is used.
- Return:
  - >=0 , successful and the Session handle is returned.
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALID_PARAMETER
  - ERROR_PPPP_TIME_OUT
  - ERROR_PPPP_INVALID_ID
  - ERROR_PPPP_INVALID_PREFIX
  - ERROR_PPPP_ID_OUT_OF_DATE
  - ERROR_PPPP_MAX_SESSION
  - ERROR_PPPP_USER_LISTEN_BREAK
  - ERROR_PPPP_UDP_PORT_BIND_FAILED
  - ERROR_PPPP_INVALID_APILICENSE

PPPP_LoginStatus_Check

- Function Declare:
  - INT32 PPPP_LoginStatus_Check(CHAR* bLoginStatus)
- Description:
  - PPPP_LoginStatus_Check: To Check login status of device
- Parameters:
  - **bLoginStatus** : To receive Login status
    - 0, Not login to Server
    - 1, Successfully login to Server (get server's login ack response in last 60 sec)
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALID_PARAMETER

PPPP_Connect/PPPP_Connect_Break

- Function Declare:
  - INT32 PPPP_Connect(const CHAR *TargetID,CHAR bEnableLanSearch, UINT16 UDP_Port)
  - INT32 PPPP_Connect_Break()
  - INT32 PPPP_ConnectByServer(const CHAR *TargetID, CHAR bEnableLanSearch, UINT16 UDP_Port, CHAR *ServerString)
- Description:
  - PPPP_Connect: To look for target device and connect it.
  - PPPP_Connect_Break: to break PPPP_Connect.
  - PPPP_ConnectByServer: The same as PPPP_ConnectByServer, but user can specify with which server to perform this function.
- Parameters:
  - **TargetID** : The target device ID
  - **bEnableLanSearch:**

    for Verion 1.0.0 and before, bEnableLanSearch: 0: Disable Lancearch, 1: Enable Lan Search

    for Verion 1.0.1 and after, The bit 1~4 of bEnableLanSearch is used to define timeout of P2P trying stage,

    bEnableLanSearch = 0 Disable Lancearch, P2P timeout = (default) 5 sec
    bEnableLanSearch = 1 Enable Lancearch, P2P timeout = (default) 5 sec
    bEnableLanSearch = 2 (0x02) Disable Lancearch, P2P timeout = 1 sec
    bEnableLanSearch = 3 (0x03) Enable Lancearch, P2P timeout = 1 sec
    bEnableLanSearch = 4 (0x04) Disable Lancearch, P2P timeout = 2 sec
    bEnableLanSearch = 5 (0x05) Enable Lancearch, P2P timeout = 2 sec
    bEnableLanSearch = 6 (0x06) Disable Lancearch, P2P timeout = 3 sec
    bEnableLanSearch = 7 (0x07) Enable Lancearch, P2P timeout = 3 sec
    .....
    bEnableLanSearch = 28 (0x1C) Disable Lancearch, P2P timeout = 14 sec
    bEnableLanSearch = 29 (0x1D) Enable Lancearch, P2P timeout = 14 sec
    bEnableLanSearch = 30 (0x1E) Disable Lancearch, P2P timeout = (No P2P trying stage) 0 sec
    bEnableLanSearch = 31 (0x1F) Enable Lancearch, P2P timeout = (No P2P trying stage) 0 sec

  - **UDP_Port** : Specify the UDP port. if **UDP_Port** =0, a random port will be used.
  - **ServerString**: Encoded string, specifying the server address.
- Return:
  - >=0 , successful and the Session handle is returned.
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_TIME_OUT
  - ERROR_PPPP_INVALID_ID
  - ERROR_PPPP_INVALID_PREFIX
  - ERROR_PPPP_DEVICE_NOT_ONLINE
  - ERROR_PPPP_NO_RELAY_SERVER_AVAILABLE
  - ERROR_PPPP_MAX_SESSION
  - ERROR_PPPP_UDP_PORT_BIND_FAILED
  - ERROR_PPPP_USER_CONNECT_BREAK

PPPP_Check

- Function Declare:
  - INT32 PPPP_Check(INT32 SessionHandle, struct ST_Session *SessionInfo)
- Description:
  - PPPP_Check : To check session information.
- Parameters:
  - **SessionHandle** : The session handle
  - **SessionInfo**: the structure where session iformation is retrived.
- Return:
  - ERROR_PPPP_SUCCESSFUL;
  - ERROR_PPPP_NOT_INITIALIZED;
  - ERROR_PPPP_INVALID_PARAMETER;
  - ERROR_PPPP_INVALID_SESSION_HANDLE;
  - ERROR_PPPP_INVALID_SESSION_HANDLE;
  - ERROR_PPPP_SESSION_CLOSED_CALLED;
  - ERROR_PPPP_SESSION_CLOSED_TIMEOUT;
  - ERROR_PPPP_SESSION_CLOSED_REMOTE;

PPPP_Close / PPPP_ForceClose

- Function Declare:
  - INT32 PPPP_Close(INT32 SessionHandle)
  - INT32 PPPP_ForceClose(INT32 SessionHandle)
- Description:
  - PPPP_Close : To release resource used by specified SessionHandle.
  - PPPP_ForceClose : To release resource used by specified SessionHandle. Don’t care if remote site received data written.
- Parameters:
  - **SessionHandle** : The session handle
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALIED_SESSION_HANDLE

struct st_PPPP_NetInfo

- CHAR bFlagInternet; // Internet Reachable? 1: YES, 0: NO
- CHAR bFlagHostResolved; // P2P Server IP resolved? 1: YES, 0: NO
- CHAR bFlagServerHello; // P2P Server Hello? 1: YES, 0: NO
- CHAR NAT_Type; // NAT type,

  0: Unknow, 1: IP-Restricted Cone type, 2: Port-Restricted Cone type, 3: Symmetric
- CHAR MyLanIP[16]; // My LAN IP.
  - If (bFlagInternet==0) || (bFlagHostResolved==0) || (bFlagServerHello==0), MyLanIP will be "0.0.0.0“
- CHAR MyWanIP[16]; // My Wan IP.
  - If (bFlagInternet==0) || (bFlagHostResolved==0) || (bFlagServerHello==0), MyWanIP will be "0.0.0.0”

struct st_PPPP_Session

- INT32 Skt; // Sockfd
- struct sockaddr_in RemoteAddr; // Remote IP:Port
- struct sockaddr_in MyLocalAddr; // My Local IP:Port
- struct sockaddr_in MyWanAddr; // My Wan IP:Port
- UINT32 ConnectTime; // Connection build in ? Sec Before
- CHAR DID[24]; // Device ID
- CHAR bCorD; // I am Client or Device, 0: Client, 1: Device
- CHAR bMode; // Connection Mode: 0: P2P, 1:Relay Mode

PPPP_Read

- Function Declare:
  - INT32 PPPP_Read(INT32 SessionHandle, UCHAR Channel, CHAR *DataBuf, INT32 *DataSize, UINT32 TimeOut_ms)
- Description:
  - PPPP_Read: To Read form data from specified **Channel** of specified **SessionHandle**. Execution is blocked untill **DataSizeToRead** bytes are read, or TimeOut_ms.
- Parameters:
  - **SessionHandle** : The session handle
  - **Channel**: The Channel ID, 7.
  - **DataBuf**: The data buffer
  - **DataSize**: Speciy how many byte to read. And, after return, it carry number of byte read.
  - **TimeOut_ms**: Time out value, in mini second.
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_TIME_OUT
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALID_SESSION_HANDLE
  - ERROR_PPPP_SESSION_CLOSED_REMOTE
  - ERROR_PPPP_SESSION_CLOSED_TIMEOUT

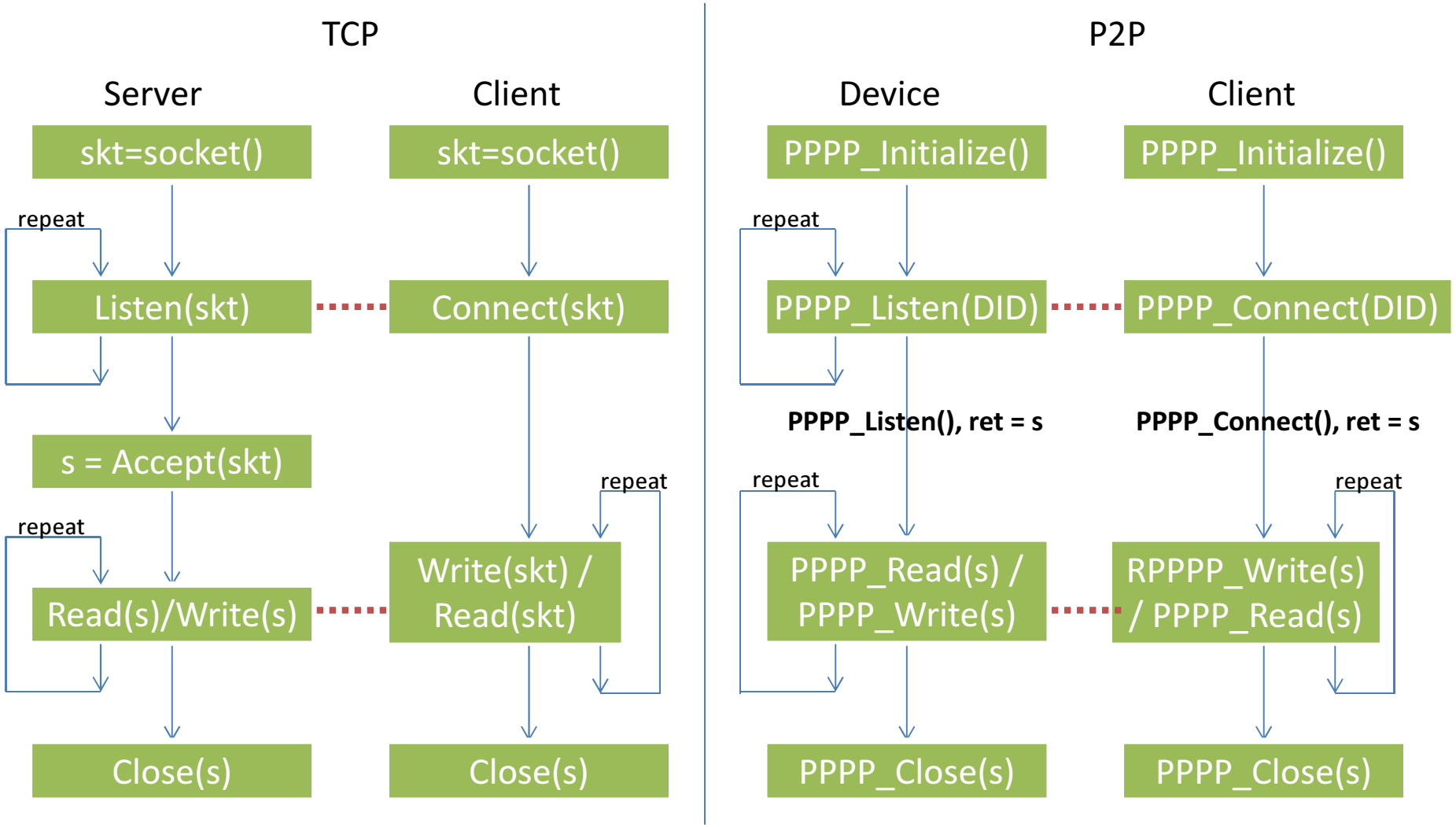
PPPP_Write

- Function Declare:
  - INT32 PPPP_Write(INT32 SessionHandle, UCHAR Channel, CHAR *DataBuf, INT32 DataSizeToWrite)
- Description:
  - PPPP_Write: To write data into specified **Channel** of specified **SessionHandle**. Execution is no-blocked, unless the sending data buffer is full. The writing buffer max size is 2MB.
- Parameters:
  - **SessionHandle** : The session handle
  - **Channel**: The Channel ID, 0~7.
  - **DataBuf**: The data buffer
  - **DataSizeToWrite** : Speciy how many byte to writeto remote site
- Return:
  - >= 0, Number of byte read
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALID_PARAMETER
  - ERROR_PPPP_INVALID_SESSION_HANDLE
  - ERROR_PPPP_SESSION_CLOSED_REMOTE
  - ERROR_PPPP_SESSION_CLOSED_TIMEOUT
  - ERROR_PPPP_REMOTE_SITE_BUFFER_FULL

PPPP_Check_Buffer

- Function Declare:
  - INT32 PPPP_Check_Buffer(INT32 SessionHandle, UCHAR Channel, UINT32 *WriteSize, UINT32 *ReadSize)
- Description:
  - PPPP_Check_Buffer: To Chek current write buffer and read buffer size. Write buffer are data to send to remote, read buffer are data received from remote site.
- Parameters:
  - **SessionHandle** : The session handle
  - **Channel**: The Channel ID, 7.
  - **WriteSize**: The write buffer size, in byte
  - **ReadSize**: The read buffer size, in byte
- Return:
  - ERROR_PPPP_SUCCESSFUL
  - ERROR_PPPP_NOT_INITIALIZED
  - ERROR_PPPP_INVALIED_SESSION_HANDLE
  - ERROR_PPPP_SESSION_CLOSED_REMOTE
  - ERROR_PPPP_SESSION_CLOSED_TIMEOUT

API flows


TCP
Server
Client
skt=socket()
skt=socket()
repeat
Listen(skt)
Connect(skt)
s = Accept(skt)
repeat
repeat
Read(s)/Write(s)
Write(skt) / Read(skt)
Close(s)
Close(s)
P2P
Device
Client
PPPP_Initialize()
PPPP_Initialize()
repeat
PPPP_Listen(DID)
PPPP_Connect(DID)
PPPP_Listen(), ret = s
PPPP_Connect(), ret = s
repeat
repeat
PPPP_Read(s) / PPPP_Write(s)
RPPPP_Write(s) / PPPP_Read(s)
PPPP_Close(s)
PPPP_Close(s)