PPCS P2P API

2.0.0

API

- Common:
    - PPCS_Initialize, PPCS_DeInitialize
    - PPCS_NetworkDetect
    - PPCS_NetworkDetectByServer
    - PPCS_GetAPIVersion
    - PPCS_QueryDID
    - PPCS_Share_Bandwidth
    - PPCS_Get_ServerIP
- Device:
    - PPCS_Listen
    - PPCS_Listen_Break
    - PPCS_LoginStatus_Check
- Client:
    - PPCS_Connect
    - PPCS_ConnectByServer
    - PPCS_Connect_Break
- Session:
    - PPCS_Check
    - PPCS_Close
- Read / Write data
    - PPCS_Read
    - PPCS_Write
- Check Buffer size
    - PPCS_Check_Buffer

Return Code of API

- >=0: Successful

– #define ERROR_PPCS_SUCCESSFUL	0

- <0: Some thing wrong

– #define ERROR_PPCS_NOT_INITIALIZED	-1
– #define ERROR_PPCS_ALREADY_INITIALIZED	-2
– #define ERROR_PPCS_TIME_OUT	-3
– #define ERROR_PPCS_INVALID_ID	-4
– #define ERROR_PPCS_INVALID_PARAMETER	-5
– #define ERROR_PPCS_DEVICE_NOT_ONLINE	-6
– #define ERROR_PPCS_FAIL_TO_RESOLVE_NAME	-7
– #define ERROR_PPCS_INVALID_PREFIX	-8
– #define ERROR_PPCS_ID_OUT_OF_DATE	-9
– #define ERROR_PPCS_NO_RELAY_SERVER_AVAILABLE	-10
– #define ERROR_PPCS_INVALID_SESSION_HANDLE	-11
– #define ERROR_PPCS_SESSION_CLOSED_REMOTE	-12
– #define ERROR_PPCS_SESSION_CLOSED_TIMEOUT	-13
– #define ERROR_PPCS_SESSION_CLOSED_CALLED	-14
– #define ERROR_PPCS_REMOTE_SITE_BUFFER_FULL	-15
– #define ERROR_PPCS_USER_LISTEN_BREAK	-16
– #define ERROR_PPCS_MAX_SESSION	-17
– #define ERROR_PPCS_UDP_PORT_BIND_FAILED	-18
– #define ERROR_PPCS_USER_CONNECT_BREAK	-19
– #define ERROR_PPCS_SESSION_CLOSED_INSUFFICIENT_MEMORY	-20
– #define ERROR_PPCS_INVALID_APILICENSE	-21
– #define ERROR_PPCS_FAIL_TO_CREATE_THREAD	-22

PPCS_GetAPIVersion

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

PPCS_Initialize, PPCS_DeInitialize

- Function Declare:
  - PPCS_Initialize(CHAR *Parameter)
  - INT32 PPCS_DeInitialize()
- Description:
  - PPCS_Initialize: To initialize usage of PPCS session module.
  - PPCS_DeInitialize: To free all resource used by PPCS session module.
- Parameters:
  - **Parameter**: The parameter string that tell server information.
- Return:
  - ERROR_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_ALREADY_INITIALIZED
  - ERROR_PPCS_INSUFFICIENT_RESOURCE

PPCS_NetworkDetect

- Function Declare:
  - INT32 PPCS_NetworkDetect(st_PPCS_NetInfo *NetInfo, UINT16 UDP_Port);
  - INT32 PPCS_NetworkDetectByServer(st_PPCS_NetInfo *NetInfo, UINT16 UDP_Port, CHAR *ServerString)
- Description:
  - PPCS_NetworkDetect: To detect network related information.
  - PPCS_NetworkDetectByServer: The same as PPCS_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_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_PARAMETER
  - ERROR_PPCS_UDP_PORT_BIND_FAILED

PPCS_QueryDID

- Function Declare:
  - INT32 PPCS_QueryDID(const CHAR* DeviceName, CHAR* DID, INT32 DIDBufSize)
- Description:
  - PPCS_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.

PPCS_Share_Bandwidth

- Function Declare:
  - INT32 PPCS_Share_Bandwidth(CHAR bOnOff)
- Description:
  - PPCS_Share_Bandwidth: Allow device(those call PPCS_Listen) to provide device relay service, if it is in suitable network condition.
- Parameters:
  - **bOnOff** :
    - bOnOff = 0: Not share, or stop sharing (if is on sharing).
    - bOnOff = 1: Allow bandwidth sharing.
- Return:
  - ERROR_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED

PPCS_Listen/PPCS_Listen_Break

- Function Declare:
  - INT32 PPCS_Listen(const CHAR *MyID, UINT32 TimeOut_Sec, UINT16 UDP_Port , CHAR bEnableInternet, const CHAR *APILicense)
  - INT32 PPCS_Listen_Break();
- Description:
  - PPCS_Listen: To login to server and wait until some client to connect with. The calling thread will be blocked, till Client connection or timeout.
  - PPCS_Listen_Break: to break PPCS_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.

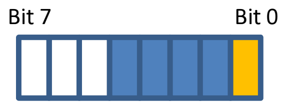
    case 2: APILicense is like "ABCDE", Empty CRC Key is used.
- Return:
  - >=0 , successful and the Session handle is returned.
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_PARAMETER
  - ERROR_PPCS_TIME_OUT
  - ERROR_PPCS_INVALID_ID
  - ERROR_PPCS_INVALID_PREFIX
  - ERROR_PPCS_ID_OUT_OF_DATE
  - ERROR_PPCS_MAX_SESSION
  - ERROR_PPCS_USER_LISTEN_BREAK
  - ERROR_PPCS_UDP_PORT_BIND_FAILED
  - ERROR_PPCS_INVALID_APILICENSE

PPCS_LoginStatus_Check

- Function Declare:
  - INT32 PPCS_LoginStatus_Check(CHAR* bLoginStatus)
- Description:
  - PPCS_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_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_PARAMETER

PPCS_Connect/PPCS_Connect_Break

- Function Declare:
  - INT32 PPCS_Connect(const CHAR *TargetID,CHAR bEnableLanSearch, UINT16 UDP_Port)
  - INT32 PPCS_Connect_Break()
  - INT32 PPCS_ConnectByServer(const CHAR *TargetID, CHAR bEnableLanSearch, UINT16 UDP_Port, CHAR *ServerString)
- Description:
  - PPCS_Connect: To look for target device and connect it.
  - PPCS_Connect_Break: to break PPCS_Connect.
  - PPCS_ConnectByServer: The same as PPCS_ConnectByServer, but user can specify with which server to perform this function.
- Parameters:
  - **TargetID** : The target device ID
  - **bEnableLanSearch:**
- refer to next page
  - **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_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_TIME_OUT
  - ERROR_PPCS_INVALID_ID
  - ERROR_PPCS_INVALID_PREFIX
  - ERROR_PPCS_DEVICE_NOT_ONLINE
  - ERROR_PPCS_NO_RELAY_SERVER_AVAILABLE
  - ERROR_PPCS_MAX_SESSION
  - ERROR_PPCS_UDP_PORT_BIND_FAILED
  - ERROR_PPCS_USER_CONNECT_BREAK



bEnableLanSearch

- if bEnableLanSearch = 0x7F --> Connect() is used to detect if Device is on-line.
  - Return Value:
    - ERROR_PPCS_SUCCESSFUL ---> Device of DID is on line
    - ERROR_PPCS_INVALID_PREFIX ---> Invalid Prefix of DID
    - ERROR_PPCS_INVALID_ID ---> Invalid DID
    - ERROR_PPCS_DEVICE_NOT_ONLINE ---> Device is not On-line (Not Login in last 5 minute)
    - ERROR_PPCS_TIME_OUT ---> No Response from Server
- else Connect() Connect is used to connect Device.
  - Bit 0 [LanSearch] , 0: Disable Lan search, 1: Enable Lan Search
  - Bit 1~4 [P2P Try time]:
    - 0 (0b0000): 5 second (default)
    - 1 (0b0001): 1 second
    - 2 (0b0010): 2 second
    - 3 (0b0011): 3 second
    - ....
    - 14 (0b1110): 14 second
    - 15 (0b1111): 0 second, No P2P trying
  - Bit 5 [RelayOff], 0: Relay mode is allowed, 1: No Relay connection
  - Bit 6 [ServerRelayOnly], 0: Device Relay is allowed, 1: Only Server relay (if Bit 5 = 1, this value is ignored)
- example:
  - bEnableLanSearch = 0 (0b00000000): LanSearch Off, P2P for 5 sec, device relay then server relay
  - bEnableLanSearch = 1 (0b00000001): LanSearch On , P2P for 5 sec, device relay then server relay
  - bEnableLanSearch = 7 (0b00000111): LanSearch On , P2P for 3 sec, device relay then server relay
  - bEnableLanSearch = 16(0b00010000): LanSearch Off, P2P for 8 sec, device relay then server relay
  - bEnableLanSearch = 30(0b00011110): LanSearch Off, No P2P , device relay then server relay
  - bEnableLanSearch = 31(0b00011111): LanSearch On , No P2P , device relay then server relay
  - bEnableLanSearch = 32(0b00100000): LanSearch Off, P2P for 5 sec, relay Off
  - bEnableLanSearch = 33(0b00100001): LanSearch On , P2P for 5 sec, relay Off
  - bEnableLanSearch = 37(0b00100101): LanSearch On , P2P for 2 sec, relay Off
  - bEnableLanSearch = 64(0b01000000): LanSearch Off, P2P for 5 sec, server relay only
  - bEnableLanSearch = 65(0b01000001): LanSearch On , P2P for 5 sec, server relay only
  - bEnableLanSearch =127(0b01111111): Connect() is used to detect if Device is on-line. (Note define of Return value is different)

PPCS_Check

- Function Declare:
  - INT32 PPCS_Check(INT32 SessionHandle, struct ST_Session *SessionInfo)
- Description:
  - PPCS_Check : To check session information.
- Parameters:
  - **SessionHandle** : The session handle
  - **SessionInfo**: the structure where session iformation is retrived.
- Return:
  - ERROR_PPCS_SUCCESSFUL;
  - ERROR_PPCS_NOT_INITIALIZED;
  - ERROR_PPCS_INVALID_PARAMETER;
  - ERROR_PPCS_INVALID_SESSION_HANDLE;
  - ERROR_PPCS_INVALID_SESSION_HANDLE;
  - ERROR_PPCS_SESSION_CLOSED_CALLED;
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT;
  - ERROR_PPCS_SESSION_CLOSED_REMOTE;

PPCS_Close / PPCS_ForceClose

- Function Declare:
  - INT32 PPCS_Close(INT32 SessionHandle)
  - INT32 PPCS_ForceClose(INT32 SessionHandle)
- Description:
  - PPCS_Close : To release resource used by specified SessionHandle.
  - PPCS_ForceClose : To release resource used by specified SessionHandle. Don’t care if remote site received data written.
- Parameters:
  - **SessionHandle** : The session handle
- Return:
  - ERROR_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALIED_SESSION_HANDLE

struct st_PPCS_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_PPCS_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

PPCS_Read

- Function Declare:
  - INT32 PPCS_Read(INT32 SessionHandle, UCHAR Channel, CHAR *DataBuf, INT32 *DataSize, UINT32 TimeOut_ms)
- Description:
  - PPCS_Read: To Read data from specified **Channel** of specified **SessionHandle**. Execution of PPCS_Read will block untill **DataSizeToRead** bytes are read, or TimeOut_ms expired.
- 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_PPCS_SUCCESSFUL
  - ERROR_PPCS_TIME_OUT
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_SESSION_HANDLE
  - ERROR_PPCS_SESSION_CLOSED_REMOTE
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT

PPCS_Write

- Function Declare:
  - INT32 PPCS_Write(INT32 SessionHandle, UCHAR Channel, CHAR *DataBuf, INT32 DataSizeToWrite)
- Description:
  - PPCS_Write: To write data into specified **Channel** of specified **SessionHandle**. Execution is no-blocked. (Important: Don't write any more data, if **WriteSize** from PPCS_Check_Buffer() is larger then 2MB, otherwise it will cause malfunction.)
- 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 writen
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_PARAMETER
  - ERROR_PPCS_INVALID_SESSION_HANDLE
  - ERROR_PPCS_SESSION_CLOSED_REMOTE
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT
  - ERROR_PPCS_REMOTE_SITE_BUFFER_FULL

PPCS_Check_Buffer

- Function Declare:
  - INT32 PPCS_Check_Buffer(INT32 SessionHandle, UCHAR Channel, UINT32 *WriteSize, UINT32 *ReadSize)
- Description:
  - PPCS_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_PPCS_SUCCESSFUL
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALIED_SESSION_HANDLE
  - ERROR_PPCS_SESSION_CLOSED_REMOTE
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT

PPCS_PktSend

(For Ver 2.0.0 later ONLY!)

- Function Declare:
  - INT32 PPCS_PktSend(INT32 SessionHandle, UCHAR Channel, CHAR *PktBuf, INT32 PktSize)
- Description:
  - PPCS_PktSend: To send a data packet to specified **Channel** of specified **SessionHandle**. Execution of PPCS_PktSend is no-blocked. (Important: PPCS_PktSend is not reliable, that means remote site MAY NOT receive it, even PPCS_PktSend return >=0).
- Parameters:
  - **SessionHandle** : The session handle
  - **Channel**: The Channel ID, 0~7.
  - **PktBuf**: The data buffer
  - **PktSize** : Speciy how many byte to send to remote site, max: 1240 Byte
- Return:
  - >= 0, Number of byte writen
  - ERROR_PPCS_INVALID_PARAMETER
  - ERROR_PPCS_INVALID_SESSION_HANDLE
  - ERROR_PPCS_SESSION_CLOSED_REMOTE
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT

PPCS_PktRecv

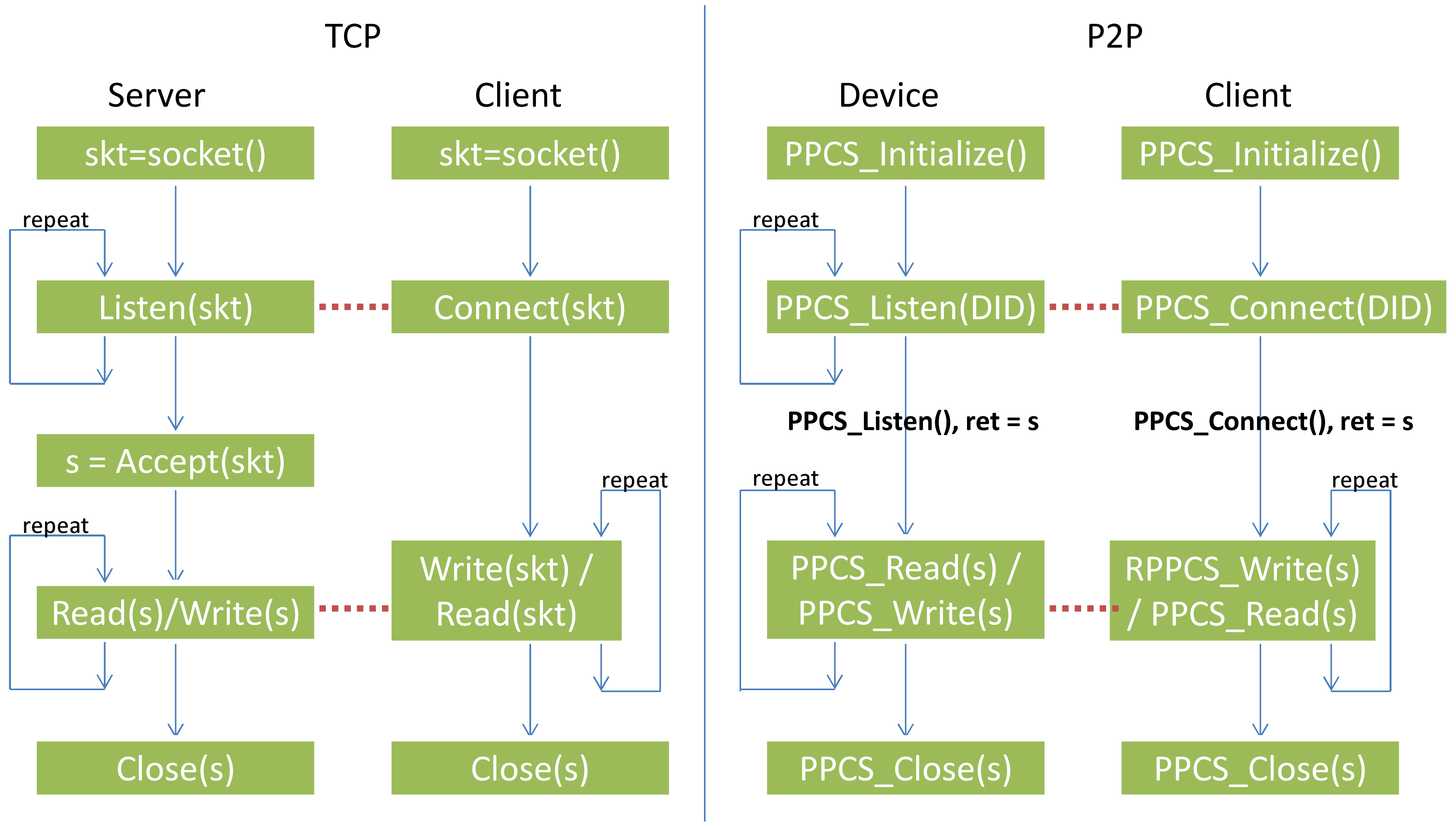
(For Ver 2.0.0 later ONLY!)

- Function Declare:
  - INT32 PPCS_PktRecv(INT32 SessionHandle, UCHAR Channel, CHAR *PktBuf, INT32 *PktSize, UINT32 TimeOut_ms)
- Description:
  - PPCS_PktRecv: To receive a data packet of specified **Channel** of specified **SessionHandle**. Execution of PPCS_PktRecv will block untill a data packet arrived, or TimeOut_ms expired. (Important: after return of PPCS_PktRecv, user **MUST** call another PPCS_PktRecv **As Soon As Possible** to prevent lose of data packet).
- Parameters:
  - **SessionHandle** : The session handle
  - **Channel**: The Channel ID, 7.
  - **PktBuf**: The data buffer
  - **PktSize**: When calling, **PktSize** speciy the size of PktBuf. And, after return, it carry number of byte received. (Warning, if **PktSize** is small then received packet size, the return value will be ERROR_PPCS_SUCCESSFUL. However, the final extra bytes in received packet will be truncated.)
  - **TimeOut_ms**: Time out value, in mini second.
- Return:
  - ERROR_PPCS_SUCCESSFUL
  - ERROR_PPCS_TIME_OUT
  - ERROR_PPCS_NOT_INITIALIZED
  - ERROR_PPCS_INVALID_SESSION_HANDLE
  - ERROR_PPCS_SESSION_CLOSED_REMOTE
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT
  - ERROR_PPCS_SESSION_CLOSED_TIMEOUT

Defference Between PPCS_PktSend and PPCS_Write

	PPCS_Write	PPCS_PktSend
Receiving API	PPCS_Read (PPCS_Read can not read Data packet from PPCS_PktSend)	PPCS_PktRecv (PPCS_PktRecv can not receive data from PPCS_Write)
Buffering	Written data is stored in under-layer buffer. User may call PPCS_Check_Buffer to check buffer size used.	No Buffering. PktData is sent to remote size immediately.
Reliability	Writen data will be sent to remote site reliably.	PktData may be lost during transmission on Network.
TCP or UDP style	TCP-Like. User may call 1 PPCS_Read to read data writen by serveral times of PPCS_Write call, or call PPCS_Read serveral times to read data by 1 PPCS_Write.	UDP-Like. Every PPCS_PktRecv receives data packet from exactly single PPCS_PktSend.
Data Size	PPCS_Write can write more than 1240 Byte (Up to 2MB, if current write buffer is 0) in one call.	PPCS_PktSend can only send at most 1240 Byte in one call.

PPCS Read / Write API flows


TCP
P2P
Server
Client
Device
Client
skt=socket()
skt=socket()
PPCS_Initialize()
PPCS_Initialize()
repeat
Listen(skt)
Connect(skt)
PPCS_Listen(DID)
PPCS_Connect(DID)
PPCS_Listen(), ret = s
PPCS_Connect(), ret = s
s = Accept(skt)
repeat
Read(s)/Write(s)
Write(skt) / Read(skt)
PPCS_Read(s) / PPCS_Write(s)
RPPCS_Write(s) / PPCS_Read(s)
Close(s)
Close(s)
PPCS_Close(s)
PPCS_Close(s)