API User Manual

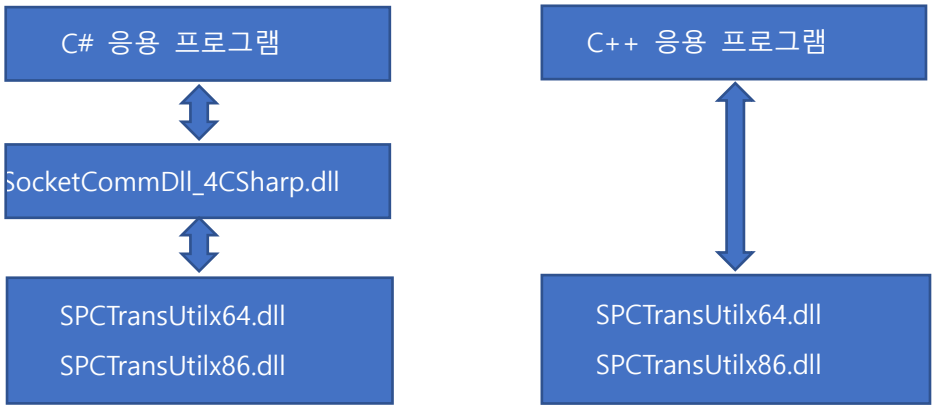
작성 : AiTech㈜

1) 내용

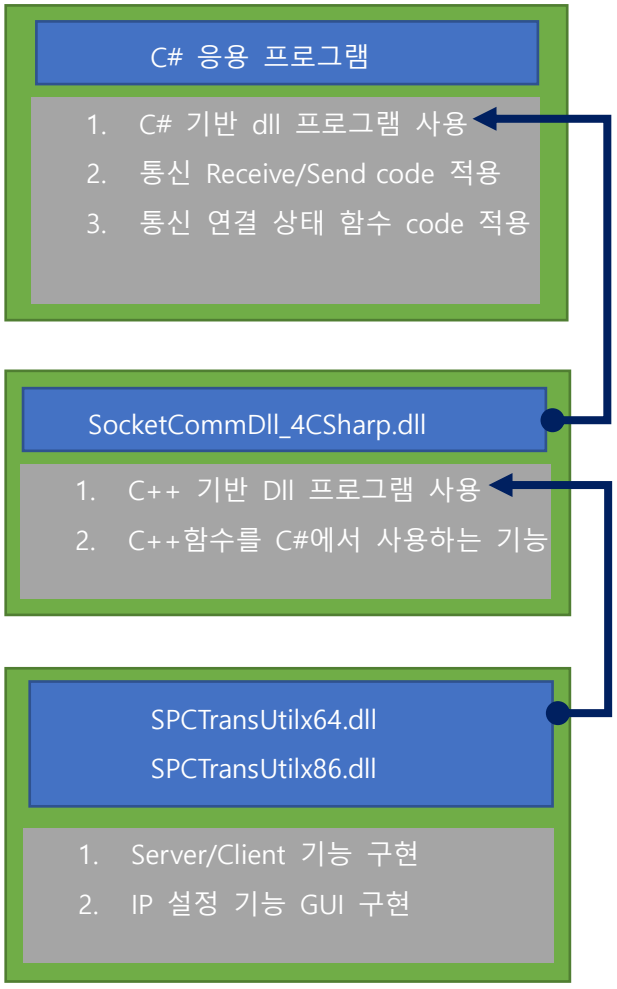
2) 매뉴얼 업데이트 내역

2021.06.21	Version 1.0	초판
2021.07.05	Version 1.1	C++ 설명 추가
2021.07.07	Version 1.2	C# 빌드 조건 명기(Any CPU→x64)
2021.08.11	Version 1.3	Trouble 항목 추가
2021.08.17	Version 1.4	시뮬레이션 환경 설정 및 시뮬레이션 디버깅 방법 추가
2021.08.20	Version 1.5	API에서 수신 받은 메시지 Queue의 첫번째 요소 호출 함수 추가
2021.08.23	Version 1.6	수신 Callback 사용➔미사용,수신 메시지 함수 명칭 및 사용법 변경
2021.08.24	Version 1.7	API←PLC CSV 파일 저장 기능 추가(API 내부 환경 설정 정보)
2021.08.26	Version 1.8	C++/C#에서 API 초기화, 종료, 수신 메시지 획득 함수 명칭 통일
2021.08.29	Version 1.9	수신 Callback 미사용➔사용
2021.09.01	Version 2.0	수신 Queue Buffer 사용 개념과 수신 Callback 사용 개념 추가
2021.09.13	Version 2.1	API 폴더 선택 옵션 적용 API 관리자 권한이 아닌 경우에도 실행 C# Delegate 종료 오류 수정
2021.10.11	Version 2.2	API DLL에서 Agent+로 보고 데이터 송신 함수 추가 (DllSocket_AddToECS)
2021.10.18	Version 2.3	"Agent+"➔"SPC+"로 이름 변경, "DllSocket_AddToECS" 주의점 추가
2021.10.20	Version 2.4	DllSocket_GetReceiveMessageInQueue 함수 반환
2021.10.28	Version 2.5	"DllSocket_AddToECS" 주의점 추가, 구분자 기호('|') 내용 추가
2022.01.21	Version 2.6	C# 문자열 깨짐 처리, 문자열 끝문자('₩0')자 처리 예제, SPC+를 Agent+로 변경
2022.02.12	Version 2.7	시리얼 키 만류 후 정식 키 입력 후 조치 사항 추가,API 창 설명추가
2022.02.17	Version 2.8	하드웨어 키 변경 업데이트 내용 추가
2022.02.22	Version 2.9	시리얼 키 입력(방법, 만기 후 입력, API Loading 순서 확인 내용 추가)
2022.06.07	Version 2.10	Heartbeat 기능 설명, 많이 묻는 질문 정리
2022.06.24	Version 2.11	Serial키 및 라이선스 키 입력 내용 제거

3) 프로그램 구성도(C# / C++)

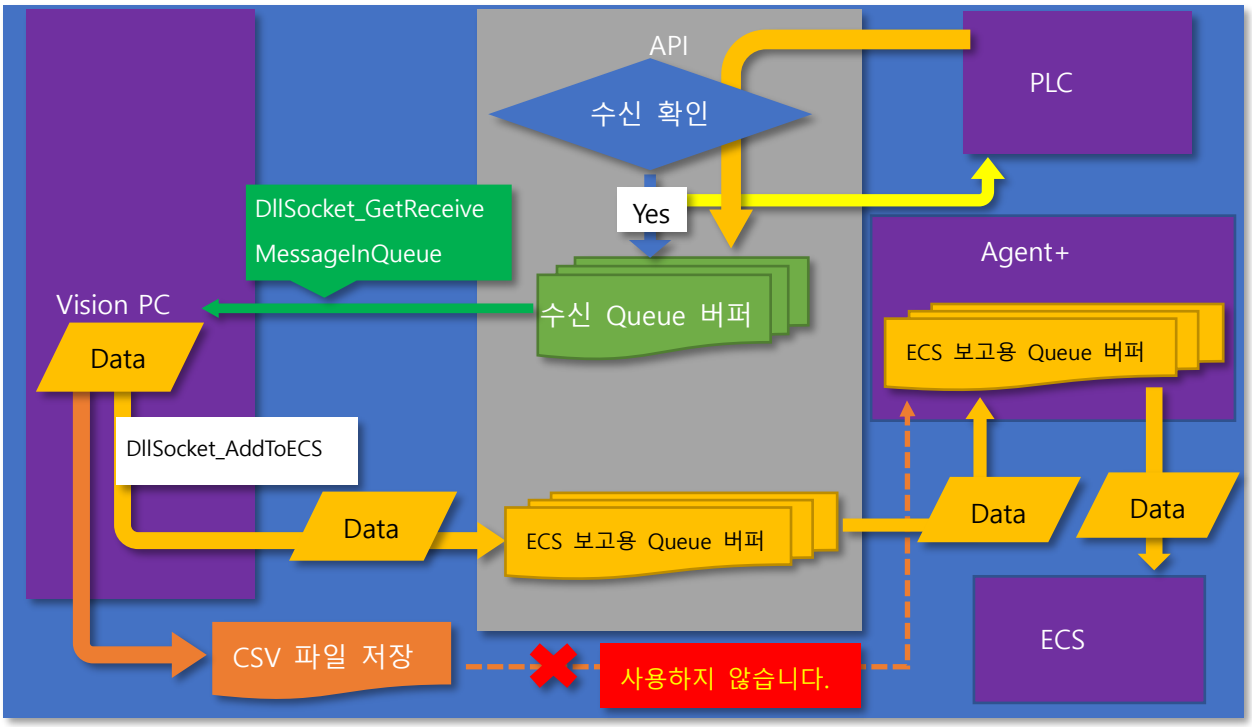


① 프로그램 기능별 구분

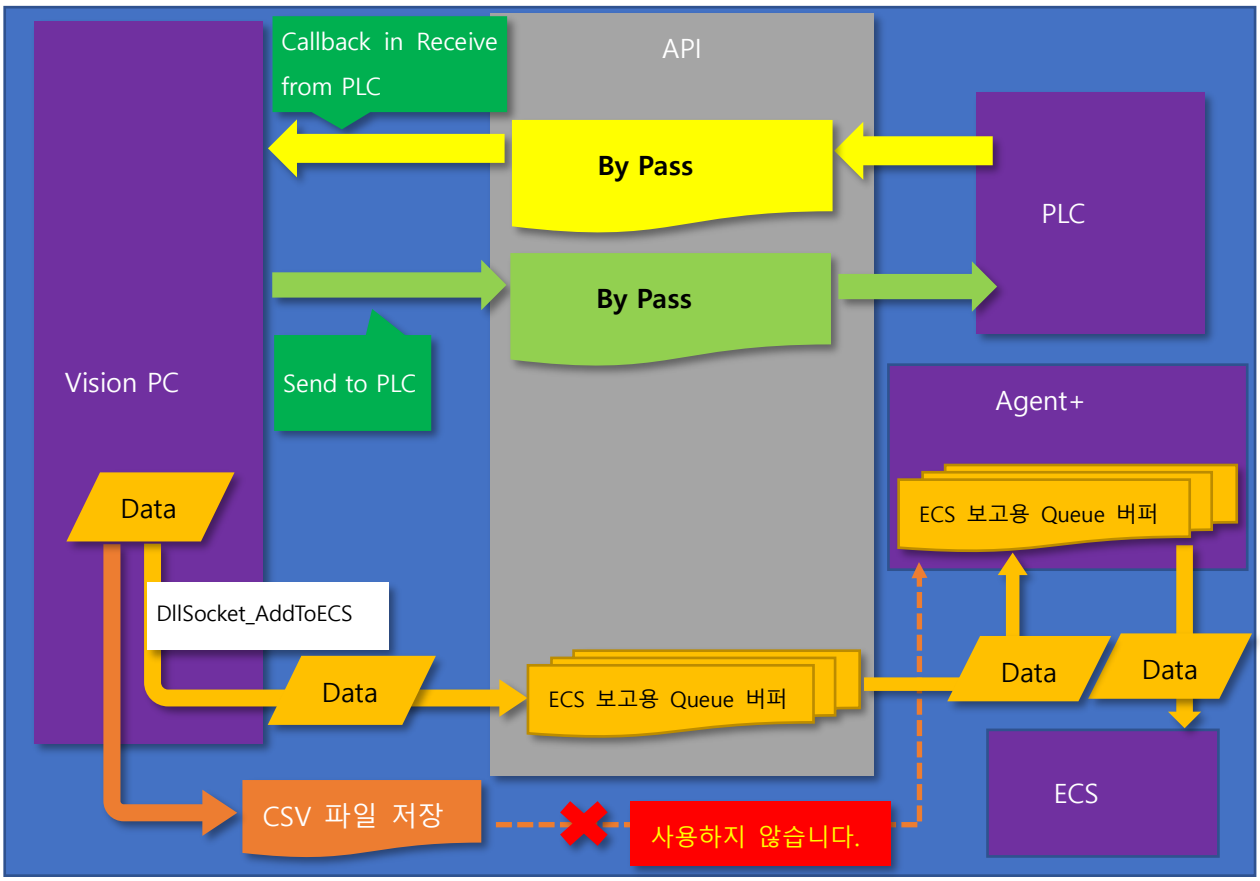


4) 수신 Queue 사용 개념과 수신 Callback 사용 개념

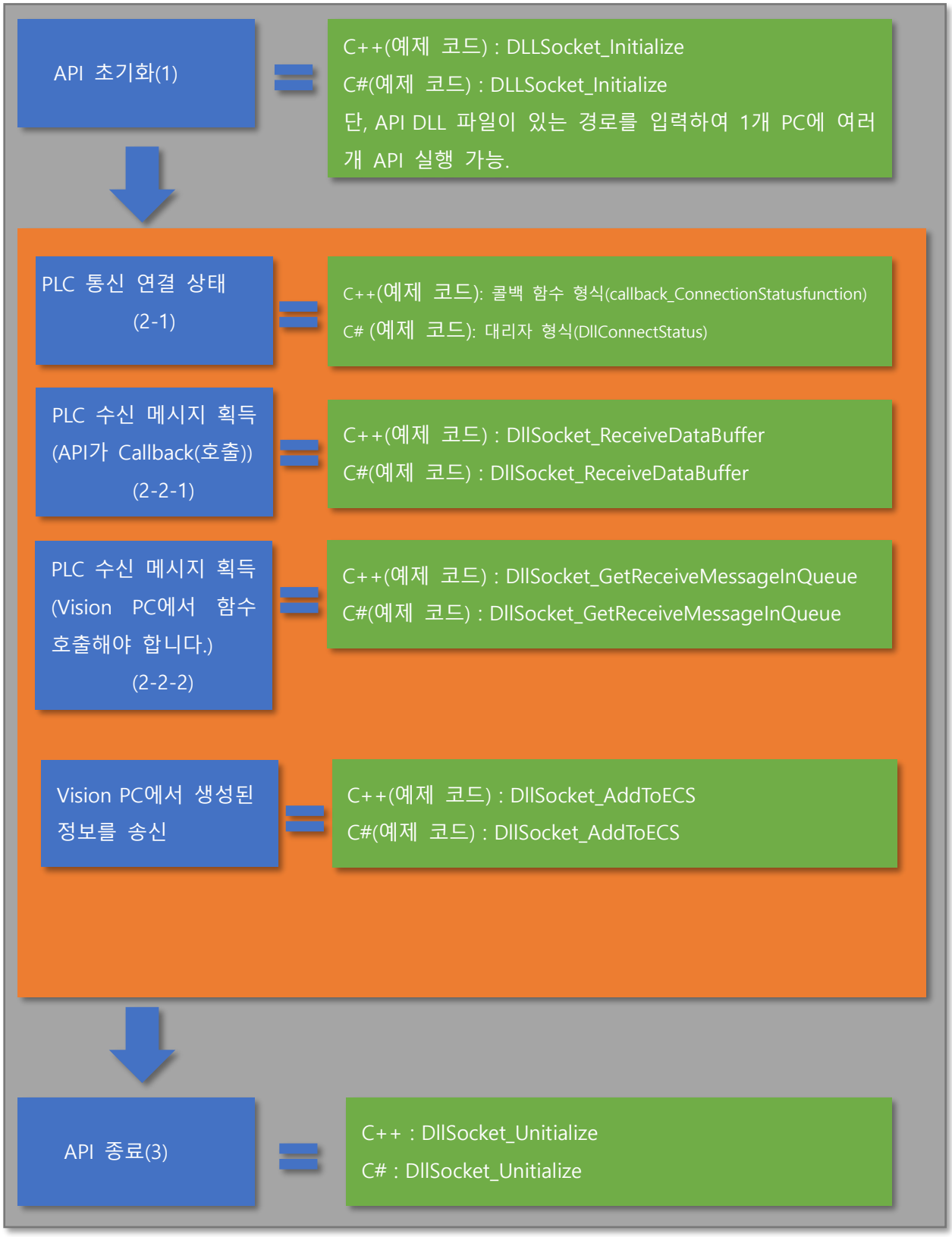
① 수신 Queue 버퍼를 사용 개념도



② 수신 Callback 사용 개념도



5) 프로그램 실행 구성


API 초기화(1)
C++(예제 코드) : DLLSocket_Initialize
C#(예제 코드) : DLLSocket_Initialize
단, API DLL 파일이 있는 경로를 입력하여 1개 PC에 여러 개 API 실행 가능.
PLC 통신 연결 상태 (2-1)
C++(예제 코드): 콜백 함수 형식(callback_ConnectionStatusfunction)
C# (예제 코드): 대리자 형식(DllConnectStatus)
PLC 수신 메시지 획득 (API가 Callback(호출)) (2-2-1)
C++(예제 코드) : DllSocket_ReceiveDataBuffer
C#(예제 코드) : DllSocket_ReceiveDataBuffer
PLC 수신 메시지 획득 (Vision PC에서 함수 호출해야 합니다.) (2-2-2)
C++(예제 코드) : DllSocket_GetReceiveMessageInQueue
C#(예제 코드) : DllSocket_GetReceiveMessageInQueue
Vision PC에서 생성된 정보를 송신
C++(예제 코드) : DllSocket_AddToECS
C#(예제 코드) : DllSocket_AddToECS
API 종료(3)
C++ : DllSocket_Unitialize
C# : DllSocket_Unitialize

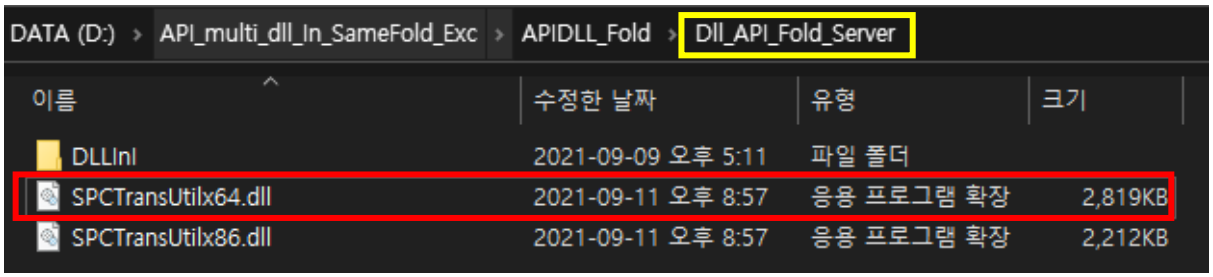
6) Dll Socket C# 프로그램 적용 방법(Visual Studio 2019)

① Dll 파일 복사

a. **실행파일 폴더에 1개의 dll 파일 복사**

✓ SocketCommDll_4CSharp.dll

b. **"SPCTransUtilx64.dll" 파일을 복사한 폴더("Dll_API_Fold_Server", 사용자 정의 폴더, 경로의 글자수는 200자이내)와 설정을 저장한 파일을 가진 폴더("DllIni") 복사**



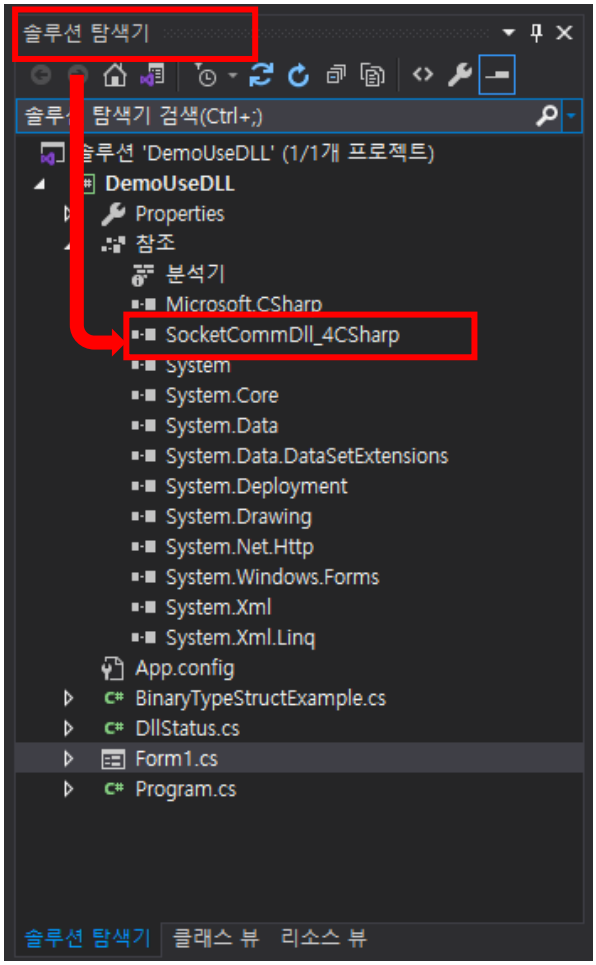
✓ IPConfig.ini : "**DllIni" 폴더와 같이 복사(통신 설정 환경 정보)**

✓ **GetFromAPI_CSV.ini : Vision PC←API 로 전달된 셀 ID 정보를 저장하는 CSV 설정**

✓ **ReceiveFromPLC_CSV.ini : PLC→API로 전달된 셀 ID 정보를 저장하는 CSV 설정**

② 참조 추가

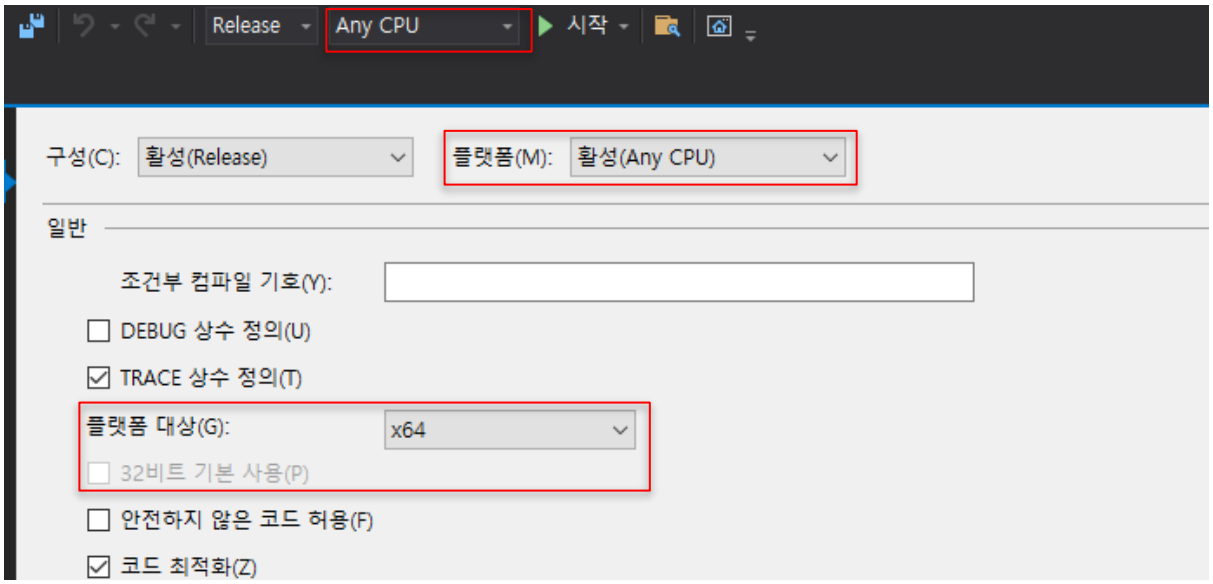
a. **Visual Studio "솔루션 탐색기➔참조" 창에 "SocketCommDll_4CSharp.dll" 추가**



③ 빌드 조건 설정

a. **대상 Framework : .NET Framework 4.6.1 이상**

b. **플랫폼 대상 : x64**



④ Name space 정의

```
//dll 통신 제어 tool
using SocketCommDll_4CSharp.LoadControl;
```

⑤ Dll Socket 초기화

a. **통신 연결 대리자(DllConnectStatus dllConnectStatus), null이면 사용하지 않습니다.**

b. **통신 수신 대리자(DllReceiveMessage dllReceive), null이면 사용하지 않습니다.**

c. **API←PLC, 수신 받은 메시지를 저장하는 Queue 버퍼 사용유무(단, 정의된 통신 프로토콜이면 API는 수신받은 메시지를 해석하여 문자열 형태로 처리됩니다.)**

```
/// <summary>
/// DllSocket 초기화
/// </summary>
/// <param name="strDllPath">dll path</param>
/// <param name="dllConnectStatus">DllSocket 연결상태 대리자</param>
/// <param name="dllReceive">DllSocket 통신 수신 대라자</param>
/// <param name=" bUseRcvQueueBuffer ">수신 메세지를 저장하는 Receive Queue 버퍼 사용 유무 설정 true이면 사용</param>
/// <returns>정상적으로 초기화 되면 true 반환</returns>
private bool DllSocket_Initialize(string strDllPath, DllConnectStatus dllConnectStatus, DllReceiveMessage dllReceive, bool bUseRcvQueBuff)
{
    lock (m_Locking)
    {
        if (null == DllSocket)
        {
            DllSocket = new LoadDllTool();
        }
        if (null != DllSocket)
        {
            if (!DllSocket.Initialize(strDllPath + "W0", dllConnectStatus, dllReceive, bUseReceiveQueueBuffer))
            {
                DllSocket.UnloadingDll();
                return false;
            }
        }
        return (null != DllSocket);
    }
}
```

⑥ 통신 수신과 통신 연결 상태가 변경되면 호출하는 대리자 처리 코드

a. **연결 상태가 변경되면 호출되는 대리자(Delgate) 예제**

```
        /// <summary>
        /// 통신 수신 상태 호출 함수
        /// </summary>
        /// <param name="strEventContents">통신 수신 상태 설명 문자열</param>
        /// <param name="nStatus">상태값</param>
        /// <param name="nErrorCode">오류 코드번호(Socket 오류 코드 참조)
        /// "https://docs.microsoft.com/ko-kr/windows/win32/winsock/windows-sockets-error-codes-2"</param>
        public void DllSocket_ConnectStatus(string strEventContents, int nStatus, int nErrorCode)
        {
            if (LABEL_DLL_CONNECTED.InvokeRequired)
            {
                // 작업쓰레드인 경우
                LABEL_DLL_CONNECTED.BeginInvoke(
                    new Action(() => LABEL_DLL_CONNECTED.Text = $"{strEventContents} :{nStatus}, {nErrorCode}"));
            }
            else
            {
                // UI 쓰레드인 경우
                LABEL_DLL_CONNECTED.Text = $"{strEventContents} :{nStatus}, {nErrorCode}";
            }
        }
```

b. 수신이 되면 호출되는 대리자(Delegate) 예제

```
/// <summary>
/// 통신 수신 처리 함수
/// </summary>
/// <param name="ReceiveBuffer">수신받은 데이터 버퍼 배열</param>
/// <returns>정상 처리되면 true</returns>
public void DllSocket_ReceiveDataBuffer(byte[] ReceiveBuffer)
{
    switch (m_ASCIIType)
    {
        case enASCIIBinaryType.ASCIITypeDll:
            if (null != ReceiveBuffer && ReceiveBuffer.Length > 0)
            {
                Trace.WriteLine($"{Encoding.Default.GetString(ReceiveBuffer)}[{ReceiveBuffer.Length}]");
            }
            break;
        case enASCIIBinaryType.BinaryTypeDll:
            if (null != ReceiveBuffer && ReceiveBuffer.Length > 0)
            {
                Trace.WriteLine($"Receive Binary Type[{ReceiveBuffer.Length}]");
            }
            break;
    }
}
```

⑦ Queue 버퍼 수신 데이터 획득(DllSocket_GetReceiveMessageInQueue)

```
/// <summary>
/// 수신받은 Qeueu 첫번째 message를 문자열로 수신받는 함수
/// </summary>
/// <param name="bLastReceiveReturn">가장 최근 수신 Message Return이면 true, 가장 과거 수신 Message Return 이면 false</param>
/// <param name="bDeleteRemainQueueBuffer">함수 호출과 동시에 남겨진 message가 있다면 모두 제거조건이면 true</param>
/// <returns>수신받은 메세지를 문자열 형태로 반환</returns>
public string DllSocket_GetReceiveMessageInQueue(bool bLastReceiveReturn, bool bDeleteRemainQueueBuffer)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.GetReceiveMessageInQueueBuffer(bLastReceiveReturn, bDeleteRemainQueueBuffer);
        }
        return "";
    }
}
```

⑧ API ➔ Agent+를 통해 ECS보고용 데이터 송신(DllSocket_AddToECS)

```
/// <summary>
/// 비젼 데이터는 오른쪽 괄호안과 같이 정보 넘어갑니다.(->DLL->SPC 중계 프로그램->ECS )
/// </summary>
/// <param name="strCellID">셀 ID 정보</param>
/// <param name="strFilePathAnaName">csV 파일이 저장된 경로와 파일 이름</param>
/// <param name="strTitleAndData">Title과 데이터가 합쳐진 문자열</param>
/// <returns>정상적으로 추가되면 true</returns>
public bool DllSocket_AddToECS(string strCellID, string strFilePathAnaName, string strTitleAndData)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.AddToECS(strCellID+'WO', strFilePathAnaName+'WO', strTitleAndData+'WO');
        }
        return false;
    }
}
```

```
DateTime time = DateTime.Now;
string strDate = time.ToString("yyyyMMdd");
string strTime = time.ToString("HHmmss");
string strCellID = string.Format("C{0}", time.ToString("HHmmssfff"));
string strParam1 = string.Format("c:\\test\\test_oc_{0}_{1}.csv", strDate, time.ToString("HH"));
string strParam2 = string.Format("Date={0},time={1},Cell ID={2},judge=ok", strDate, strTime, strCellID);
bool bReturn = m_DllSocketCtrl.DllSocket_AddToECS(strCellID, strParam1, strParam2);
```

⑨ Dll Socket 통신 송신(API➔PLC)

```
/// <summary>
/// 통신 송신 함수(ASCII Type)
/// </summary>
/// <param name="strSend">보내려는 문자열</param>
/// <returns>송신 성공이면 true</returns>
public bool DllSocket_Send(string strSend)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.Send(strSend+'W0');
        }
        return false;
    }
}

/// <summary>
/// 통신 송신 함수(BINARY Type)
/// </summary>
/// <param name="byteSend">보내려는 바이너리 타입</param>
/// <returns>송신 성공이면 true</returns>
public bool DllSocket_Send(byte[] byteSend)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.Send(byteSend);
        }
        return false;
    }
}
```

⑩ Dll 종료

```
/// <summary>
/// 종료자
/// </summary>
~DllSocketExample()
{
    DllSocket_Unitialize();
}

/// <summary>
/// DllSocket 종료 처리
/// </summary>
/// <returns></returns>
public bool DllSocket_Unitialize()
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.UnloadingDll();
        }
        return true;
    }
}
```

⑪ 기타 : IP 주소 설정 Dialog window show/hide

```
/// <summary>
/// Dll Socket의 IP Config 설정창 Show/Hide
/// </summary>
/// <param name="bShowHide">show와 hide는 true와 false, 단, bToggleModeOnlstrue값이 false이면 적용.</param>
/// <param name="bToggleModeOnlstrue">show hide toggle 모드</param>
/// <returns>정상처리되면 true</returns>
public bool DllSocket_IPConfigToggleShow(bool bShowHide, bool bToggleModeOnlstrue)
{
    lock(m_Locking)
    {
        if(null!=DllSocket)
        {
            return DllSocket.ToggleViewHideIPConfigDll(bShowHide, bToggleModeOnlstrue);
        }
        return false;
    }
}
```

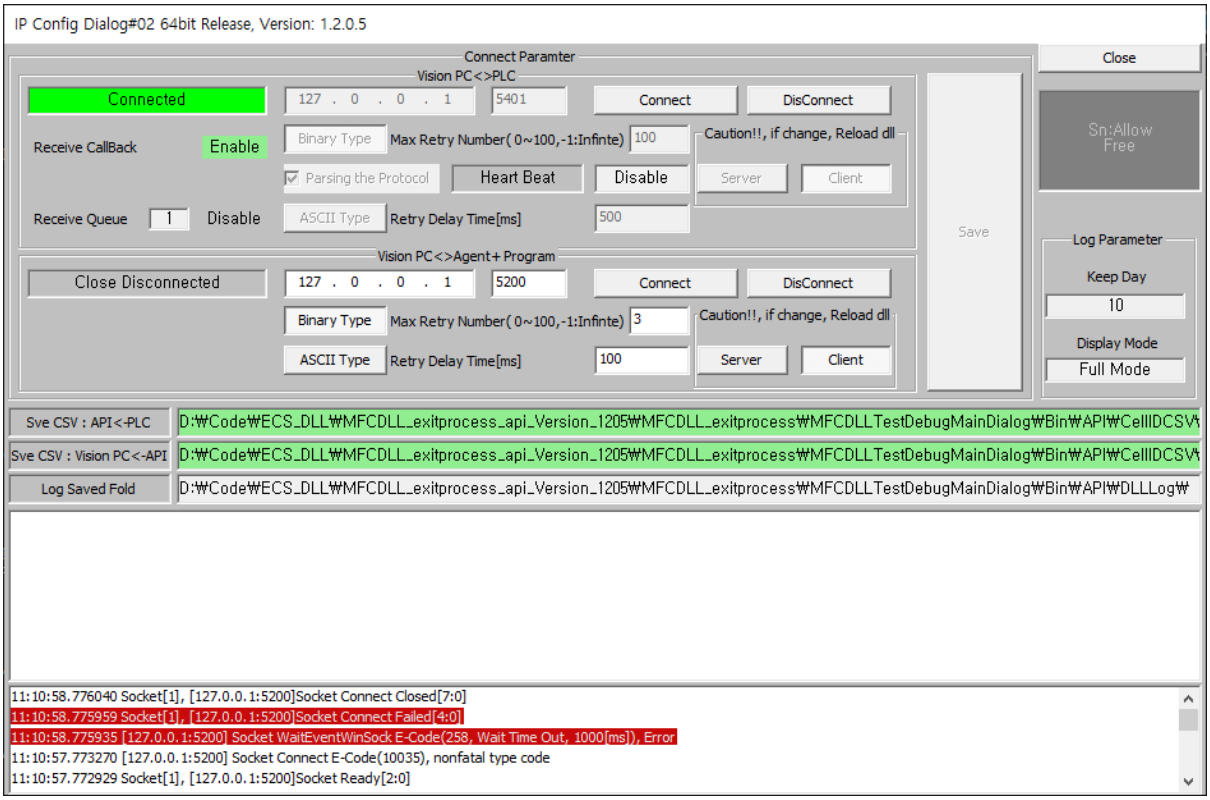


Figure 1. API Config 화면

⑫ Dll Socket 예제 코드(C#, 예제 프로그램의 "DllSocketExample.cs")

```
//dll 통신 제어 tool
using SocketCommDll_4CSharp.LoadControl;
namespace DemoUseDLL.DllSocket_Example
{
    public class DllSocketExample
    {
        /// <summary>
        /// dll socket 제어변수
        /// </summary>
        private LoadDllTool DllSocket = null;

        /// <summary>
        /// 인터럽터 처리(DllSocket)
        /// </summary>
        private static object m_Locking = null;

        /// <summary>
        /// 생성자
        /// </summary>
        /// <param name="dllConnectStatus">연결 상태 정보 호출 대리자 함수</param>
        /// <param name="dllReceive">수신시 메세지를 전달 호출 대리자</param>
        /// <param name="bUseReceiveQueueBuffer">수신 메세지 저장 Queue 버퍼 사용 유무, 사용은 true</param>
        public DllSocketExample(DllConnectStatus dllConnecStatus, DllReceiveMessage dllReceive, bool bUseReceiveQueueBuffer)
        {
            m_Locking = new object();
            DllSocket_Initialize(dllConnecStatus, dllReceive, bUseReceiveQueueBuffer);
        }


        /// <summary>
        /// 생성자
        /// </summary>
        /// <param name="strDllPath">API dll 경로</param>
        /// <param name="dllConnecStatus">연결 상태 정보 호출 대리자 함수</param>
        /// <param name="dllReceive">수신시 메세지를 전달 호출 대리자</param>
        /// <param name="bUseRecQueBuffer">수신 메세지 저장 Queue 버퍼 사용 유무, 사용은 true</param>
        public DllSocketExample(string strDllPath, DllConnectStatus dllConnecStatus, DllReceiveMessage dllReceive, bool bUseRecQueBuffer)
        {
            m_Locking = new object();

            DllSocket_Initialize(strDllPath + 'W0', dllConnecStatus, dllReceive, bUseRecQueBuffer);
        }


        /// <summary>
        /// 종료자
        /// </summary>
        ~DllSocketExample()
        {
            DllSocket_Unitialize();
        }

        /// <summary>
        /// DllSocket 종료 처리
        /// </summary>
        /// <returns></returns>
        public bool DllSocket_Unitialize()
        {
            lock (m_Locking)
            {
                if (null != DllSocket)
                {
                    return DllSocket.UnloadingDll();
                }
                return true;
            }
        }
```

```
        /// <summary>
        /// DllSocket 초기화
        /// </summary>
        /// <param name="strDllPath">dll path</param>
        /// <param name="dllConnectStatus">DllSocket 연결상태 대리자</param>
        /// <param name="dllReceive">DllSocket 통신 수신 대라자</param>
        /// <param name="bUseRcvQueBuff">수신 메세지를 저장하는 Receive Queue 버퍼 사용 유무 설정 true이면 사용</param>
        /// <returns>정상적으로 초기화 되면 true 반환</returns>
        private bool DllSocket_Initialize(string strDllPath, DllConnectStatus dllConnect, DllReceiveMessage dllReceive, bool bUseRcvQueBuff)
        {
            lock (m_Locking)
            {
                if (null == DllSocket)
                {
                    DllSocket = new LoadDllTool();
                }
                if (null != DllSocket)
                {
                    if (!DllSocket.Initialize(strDllPath + "WO", dllConnect, dllReceive, bUseRcvQueBuff))
                    {
                        DllSocket.UnloadingDll();
                        return false;
                    }
                }
                return (null != DllSocket);
            }
        }

        /// <summary>
        /// DllSocket 초기화 여부 확인 함수
        /// </summary>
        /// <returns></returns>
        public bool DllSocket_IsIntialized()
        {
            lock (m_Locking)
            {
                if (null == DllSocket)
                {
                    return false;
                }
                return DllSocket.GetDllStatus();
            }
        }

        /// <summary>
        /// DllSocket ASCII Type, Binary Type, Invalid Type 반환
        /// </summary>
        /// <returns>ASCII Type(1), Binary Type(0), Invalid Type(-1)</</returns>
        public bool DllSocket_IsASCIITypeDll()
        {
            lock (m_Locking)
            {
                if (null != DllSocket)
                {
                    return enASCIIBinaryType.ASCIITypeDll == DllSocket.IsASCIITypeDll();
                }
                return false;
            }
        }

        /// <summary>
        /// DllSocket 연결 상태 반환
        /// </summary>
        /// <returns>연결이면 true</returns>
        public bool DllSocket_IsConnected()
        {
            lock(m_Locking)
            {
                return DllSocket.IsConnectedDll();
            }
        }
```

```
/// <summary>
/// Dll Socket의 IP Config 설정창 Show/Hide
/// </summary>
/// <param name="bShowHide">show와 hide는 true와 false, 단, bToggleModeOnIstrue값이 false이면 적용됩니다.</param>
/// <param name="bToggleModeOnIstrue">show hide toggle 모드</param>
/// <returns>정상처리되면 true</returns>
public bool DllSocket_IPConfigToggleShow(bool bShowHide, bool bToggleModeOnIstrue)
{
    lock(m_Locking)
    {
        if(null!=DllSocket)
        {
            return DllSocket.ToggleViewHideIPConfigDll(bShowHide, bToggleModeOnIstrue);
        }
        return false;
    }
}

/// <summary>
/// 통신 송신 함수(ASCII Type)
/// </summary>
/// <param name="strSend">보내려는 문자열</param>
/// <returns>송신 성공이면 true</returns>
public bool DllSocket_Send(string strSend)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.Send(strSend+'W0');
        }
        return false;
    }
}

/// <summary>
/// 통신 송신 함수(BINARY Type)
/// </summary>
/// <param name="byteSend">보내려는 바이너리 타입</param>
/// <returns>송신 성공이면 true</returns>
public bool DllSocket_Send(byte[] byteSend)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.Send(byteSend);
        }
        return false;
    }
}

/// <summary>
/// 비젼 데이터는 오른쪽 괄호안과 같이 정보 넘어갑니다.(->DLL->SPC 중계 프로그램->ECS )
/// </summary>
/// <param name="strCellID">셀 ID 정보</param>
/// <param name="strFilePathAnaName">csV 파일이 저장된 경로와 파일 이름</param>
/// <param name="strTitleAndData">Title과 데이터가 합쳐진 문자열</param>
/// <returns>정상적으로 추가되면 true</returns>
public bool DllSocket_AddToECS(string strCellID, string strFilePathAnaName, string strTitleAndData)
{
    lock (m_Locking)
    {
        if (null != DllSocket)
        {
            return DllSocket.AddToECS(strCellID, strFilePathAnaName, strTitleAndData);
        }
        return false;
    }
}
```

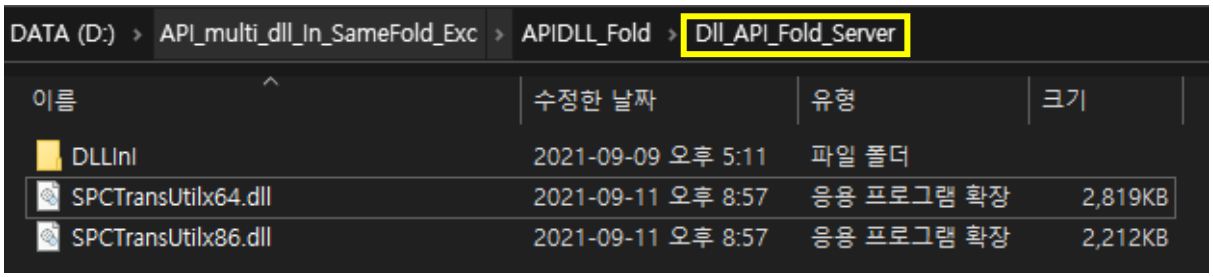
```
        /// <summary>
        /// 수신받은 Qeueu 첫번째 message를 문자열로 수신받는 함수
        /// </summary>
        /// <param name="bLastReceiveReturn">가장 최근에 수신받은 Message Return이면 true, 가장 과거에 수신받은 Message Return 이면
false</param>
        /// <param name="bDeleteRemainQueueBuffer">함수 호출과 동시에 남겨진 message가 있다면 모두 제거조건이면 true</param>
        /// <returns>수신받은 메세지를 문자열 형태로 반환</returns>
        public string DllSocket_GetReceiveMessageInQueue(bool bLastReceiveReturn, bool bDeleteRemainQueueBuffer)
        {
            lock(m_Locking)
            {
                if(null!=DllSocket)
                {
                    return DllSocket.GetReceiveMessageInQueueBuffer(bLastReceiveReturn, bDeleteRemainQueueBuffer);
                }
                return "";
            }
        }

        ////////////////////////////////////////////////////////////////////// End Class Functions
    }
    ////////////////////////////////////////////////////////////////////////////// End namespace
}
```

7) Dll Socket C++ 프로그램 적용 방법(Visual Studio 2019)

① Dll Socket C++ 프로그램 적용 방법(Visual Studio 2019)

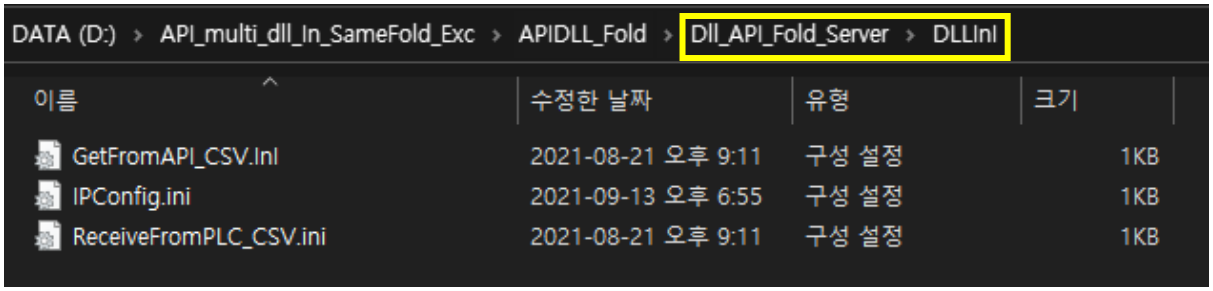
a. **Dll 파일("SPCTransUtilx64.dll"와 "SPCTransUtilx86.dll")을 특정 폴더("Dll_API_Fold_Server", 사용자 정의 폴더, 경로의 글자수는 200자이내)에 복사**

- ✓ 64비트 실행파일에 적용 : "SPCTransUtilx64.dll"
- ✓ 32비트 실행파일에 적용 : "SPCTransUtilx86.dll"



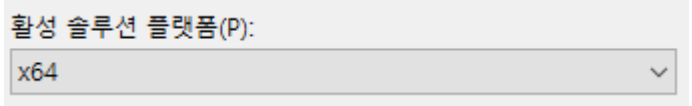
b. **설정을 저장한 파일을 가지고 있는 DllIni 파일 복사**

- ✓ IPConfig.ini : "DllIni" 폴더와 같이 복사
- ✓ **GetFromAPI_CSV.ini : Vision PC←API 로 전달된 셀 ID 정보를 저장하는 CSV 설정**
- ✓ **ReceiveFromPLC_CSV.ini : PLC→API로 전달된 셀 ID 정보를 저장하는 CSV 설정**



② 예제 프로그램 폴더 "VTECSLoad"의 파일을 적용하려는 프로젝트로 복사

③ 솔루션 플랫폼(x64 혹은 x86 지원)



④ Namespace 사용 정의(헤드 파일)

```
using namespace LOADVTECSDll_GROUP;
```

⑤ Dll load 제어기 변수 선언 및 초기화

```
LoadVTECSDll                m_LoadVTECSDll;
```

```
/// <summary>
/// dll 초기화-실행파일
/// </summary>
/// <param name="Callback_Connect">통신 연결 상태가 변경되면 호출하는 call back 함수 접근주소</param>
/// <param name="Callback_Receive">통신 수신이 되면 호출하는 call back 함수 접근 주소</param>
/// <param name="bUseReceiveQueueBuffer">수신 Queue 버퍼 사용이면 true, 사용하지 않으면 false</param>
/// <returns>정상 처리되면 true</returns>
bool DllSocket_Initialize(callback_ConnectionStatusfunction Callback_Connect, callback_Bufferfunction Callback_Receive = NULL,
const bool bUseReceiveQueueBuffer = true);

/// <summary>
/// dll 초기화-동일한 폴더에서 실행파일이 dll 초기화 할 때 사용
/// </summary>
/// <param name="strDllFold">dll 파일이 있는 경로 문자열</param>
/// <param name="Callback_Connect">통신 연결 상태가 변경되면 호출하는 call back 함수 접근주소</param>
/// <param name="Callback_Receive">통신 수신이 되면 호출하는 call back 함수 접근 주소</param>
/// <param name="bUseReceiveQueueBuffer">수신 Queue 버퍼 사용이면 true, 사용하지 않으면 false</param>
/// <returns></returns>
bool DllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect, callback_Bufferfunction
Callback_Receive = NULL, const bool bUseReceiveQueueBuffer = true);
```

a. **PLC에서 수신되는 메시지를 Callback 함수에서 직접 받아서 처리하는 초기화**

- ✓ 통신 연결상태 변경 Callback: 사용
- ✓ 통신 수신이 되면 호출되는 Callback : 사용
- ✓ 수신 Queue버퍼 : 미사용

```
//인자1 : 통신 연결 상태 변경시 호출되는 call back 함수 : 사용
//인자2 : 수신이 되면 호출되는 call back 함수 : 사용
//인자3 : API 자체 수신 Queue 버퍼 : 미사용
if (m_LoadVTECSDll.DllSocket_Initialize(DllSocket_ConnectStatus, DllSocket_ReceiveDataBuffer, false))
{
        m_CurrentDllStatus.bInitialzed = true;
        SubDisplayUpdate();//화면에 dll 상태표시
}
```

b. **PLC수신 메시지를 API에서 Queue버퍼에 저장하여, Vision PC에서 수신 메시지 요청시에 문자열로 해석 값을 처리하는 초기화**

- ✓ 통신 연결상태 변경 Callback: 사용
- ✓ 통신 수신이 되면 호출되는 Callback : 미사용
- ✓ 수신 Queue버퍼 : 사용

```
bool LoadVTECSDll::DllSocket_Initialize(callback_ConnectionStatusfunction Callback_Connect, callback_Bufferfunction
Callback_Receive, const bool bUseReceiveQueueBuffer)
{
        m_CriticalLoadUnload.Lock();
        CString strPath;
        GetCurrentDirectory(MAX_PATH, strPath.GetBuffer(MAX_PATH));
        strPath.ReleaseBuffer();
        strPath += _T('WW');
        const bool bReturn = SubDllSocket_Initialize(strPath, Callback_Connect, Callback_Receive, bUseReceiveQueueBuffer);
        m_CriticalLoadUnload.Unlock();
        return bReturn;
}

bool LoadVTECSDll::DllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect,
callback_Bufferfunction Callback_Receive, const bool bUseReceiveQueueBuffer)
{
        m_CriticalLoadUnload.Lock();
        const bool bReturn = SubDllSocket_Initialize(strDllFold, Callback_Connect, Callback_Receive, bUseReceiveQueueBuffer);
        m_CriticalLoadUnload.Unlock();
        return bReturn;
}
```

c. **통신 연결 상태가 변경되면 호출되는 콜백 함수 정의**

- ✓ Window Message 기반 처리 함수 사용금지
- ✓ 예 : LRESULT SendMessage(HWND,UINT,WPARAM,LPARAM)

```
/// <summary>
/// 통신 연결가 변경되면 호출되는 함수
/// </summary>
/// <param name="strIPAddress4">통신 이벤트 발생 내역</param>
/// <param name="nPortNumber">포트 번호</param>
/// <param name="nStatus">연결 상태를 int로 표현</param>
/// <param name="nErrorCode">오류 코드 번호</param>
/// <returns>0</returns>
static UINT WINAPI DllSocket_ConnectStatus(char* strEventContents, int nStatus, int nErrorCode);
```

d. **통신 수신이 되면 호출되는 콜백 함수 정의(Window Message 처리 함수 사용금지)**

- ✓ Window Message 기반 처리 함수 사용금지
- ✓ 예 : LRESULT SendMessage(HWND,UINT,WPARAM,LPARAM)

```
/// <summary>
/// 통신 수신이 되면 호출되는 함수
/// </summary>
/// <param name="pReceiveBuffer">수신받은 버퍼 접근 주소</param>
/// <param name="nReceiveLengthWithByte">수신받은 버퍼 개수</param>
/// <returns>0</returns>
static UINT WINAPI DllSocket_ReceiveDataBuffer(unsigned char* pReceiveBuffer, int nReceiveLengthWithByte);
```

⑥ Queue 버퍼 수신 데이터 획득("DllSocket_GetReceiveMessageInQueue")

```
char* LoadVTECSDll::DllSocket_GetReceiveMessageInQueue(const bool bLastReceiveMsgReturn, const bool
bDeleteRemainReceiveQueue)
{
        m_CriticalLoadUnload.Lock();

        if (NULL != m_fGetReceiveMessageInQueueBuffer)
        {
                char* pReturn = m_fGetReceiveMessageInQueueBuffer(bLastReceiveMsgReturn,
bDeleteRemainReceiveQueue);
                m_CriticalLoadUnload.Unlock();
                return pReturn;
        }
        m_CriticalLoadUnload.Unlock();
        return NULL;
}
```

⑦ 통신 송신(ASCII와 Binary 예제)

```
/// <summary>
/// 바이트 버퍼와 바이트 개수를 대입하여 이더넷으로 송신
/// </summary>
/// <param name="pSendBuffer">송신할 바이트 버퍼</param>
/// <param name="nSendLength">송신할 바이트 개수</param>
/// <returns>정상적으로 송신할 경우 true</returns>
bool DllSocket_Send(unsigned char* pSendBuffer, const int nSendLength);
```

```
            //ASCII Type
            if (m_LoadVTECSDll.DllSocket_IsASCIITypeDll())
            {
#ifdef _UNICODE
                        CStringW strTitleW = _T("C:₩₩Test₩₩oc_vision_20210429_11.csv");
                        CStringA strTitle;
                        strTitle = strTitleW;
                        m_LoadVTECSDll.DllSocket_Send((byte*)strTitle.GetString(), strTitle.GetLength());
#else
                        CStringA strTitle = "C:₩₩Test₩₩oc_vision_20210429_11.csv";
                        m_LoadVTECSDll.DllSocket_Send((byte*)strTitle.GetString(), strTitle.GetLength());
#endif // _UNICODE
            }
            else
            {
                        //Binary Type
                        /*
#pragma pack(1)
                                    struct typeTest
                                    {
                                                char        strID[10]; //문자열 128
                                                INT32       intCount;    //숫자형
                                                BYTE        byteData[70]; //바이트형 배열
                                                UINT32      uintCount[4];  //유니트형 배열
                                    };
#pragma pack()
                        typeTest                            m_stBinaryTest;
                        */
                        const size_t nSize = sizeof(typeTest);
                        CTime time = CTime::GetCurrentTime();
                        CStringA strA;
                        strA.Format("ABCDEF%02u%02u", time.GetMonth(), time.GetSecond());
                        memcpy_s(m_stBinaryTest.strID, 10, strA.GetString(), 10);

                        m_LoadVTECSDll.DllSocket_Send((byte*)&m_stBinaryTest, nSize);
            }
```

⑧ API ➔ Agent+를 통해 ECS보고용 데이터 송신(DllSocket_AddToECS)

```
//송신할 메세지를 추가하는 함수
//인자1                 :           셀 id 문자열
//인자1                 :           문자열이 저장되는 파일 경로 및 파일 이름 포함
//인자2                 :           추가할 Data 메세지 문자열
//반환                  :           정상이면 TRUE
bool DllSocket_AddToECS(CString& strCellID, CString& strFilePathName, CString& strData);
```

```
bool LoadVTECSDll::DllSocket_AddToECS(CString& strCellID, CString& strFilePathName, CString& strData)
{
            m_CriticalLoadUnload.Lock();
            if (NULL != m_fAddLine && !strFilePathName.IsEmpty() && !strData.IsEmpty())
            {
#ifdef _UNICODE
                        CStringA strA, strB;
                        strA = strFilePathName;
                        strB = strData;
                        const bool bReturn = m_fAddLine((PCHAR)strCellID.GetString(), (PCHAR)strA.GetString(),
(PCHAR)strB.GetString());
#else
                        const bool bReturn = m_fAddLine((PCHAR)strCellID.GetString(), (PCHAR)strFilePathName.GetString(),
            (PCHAR)strData.GetString());
#endif // _UNICODE

                        m_CriticalLoadUnload.Unlock();
                        return bReturn;
            }
            m_CriticalLoadUnload.Unlock();
            return false;
}
```

```
CString strCellID, strFileName, strData, strDate, strTime;
CTime time = CTime::GetCurrentTime();
DWORD nT = (DWORD)(GetTickCount64() % 1000);
strCellID.Format(_T("KAMM0%02d%03u"), time.GetSecond(), nT);
strDate = time.Format(_T("%Y%m%d"));
strTime = time.Format(_T("%H%M%S"));
strFileName.Format(_T("D:\\ECS 중계 프로그램\\csv 분석\\오창\\14-2\\%s\\GapVision_CellData_%s_%02d.csv"), strDate, strDate, time.GetHour());
strData.Format(_T("Date=%s,Time=%s,CellID=%s,TBiCell=1,TCell=0,TOKCell=0,CHKG0=0.00,Type=0.00,OKNG=1"), strDate, strTime, strCellID);
m_LoadVTECSDll.DllSocket_AddToECS(strCellID, strFileName, strData);
```

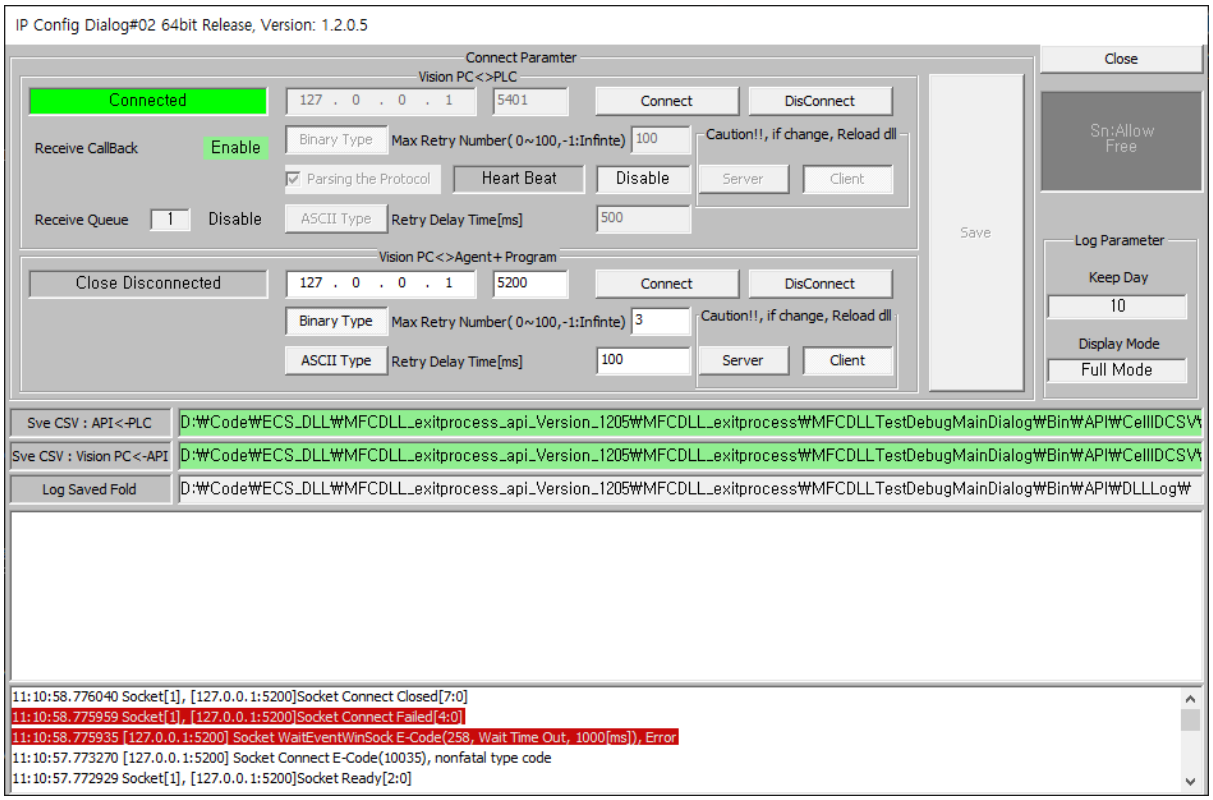
⑨ Dll 종료 처리

```
//dll 파일 Unloading 처리
//인자1                 :          없음
//반환                  :          정상이면 TRUE, 그렇지 않으면 FALSE
bool DllSocket_Unitialize(void);
```

⑩ 기타 : IP 주소 설정 Dialog window show/hide

```
/// <summary>
/// DLL IP Config 창 보이기/숨기기
/// </summary>
/// <param name="bShow">창 보이기 true, 숨기기 false</param>
/// <param name="bToggleModeOnlsTrue">창 보이기 토글 모드 사용이면 true, 토글모드에서는 bShow 값은 무시 됩니다.</param>
/// <returns></returns>
bool DllSocket_IPConfigToggleShow(bool bShow, bool bToggleModeOnlsTrue);
```

⑪ IP 주소 설정창 Pop-up 이미지



⑫ 예제 코드

a. "DefineDllFunctionPoint.h"

```
#pragma once
#ifndef CONNECT_EVENT_LIST
#define    CONNECT_EVENT_LIST
/// <summary>
/// 통신 연결 상태값 정의
/// </summary>
enum enConnectEvent : signed char
{
            OnError = -1,
            OnInvalid,
            OnStart_CreateSocketMemory,     //소켓 메모리 생성하려고 시작
            OnReady,                        //서버 타입인 경우 client에서 통신 접속을 기다리는 상태
            OnConnected,                    //통신 연결상태
            OnConnectFail,                  //통신 연결 실패 상태
            OnDisConnected,                 //통신 연결 해제 상태
            OnTerminated,                   //통신 연결 제어 이벤트 처리부 해제 상태
            OnCloseComm                     //통신 control close 상태
}
#endif // !ENUM_CONNECT_EVENT_LIST

/// <summary>
/// 통신이 수신되면 수신받은 버퍼 접근주소와 수신받은 바이트 개수를 입력하여 호출하는 콜백함수 선언부
/// <param name="BYTE*">수신받은 버퍼의 접근 주소</param>
/// <param name="int">수신받은 바이트 개수</param>
/// </summary>
typedef UINT(__stdcall* callback_Bufferfunction)(unsigned char*, int);

/// <summary>
/// 통신 연결 상태가 변경되면 호출되는 콜백함수 선언부
/// </summary>
/// <param name="char*">이벤트 발생 내용 문자열</param>
/// <param name="int">연결 상태 int 값</param>
/// <param name="int">오류 코드 값</param>
/// <returns>정상처리 되면 true</returns>
typedef UINT(__stdcall* callback_ConnectionStatusfunction)(char*, int, int);
namespace LOADVTECSDll_GROUP
{
            //dll에 송신, 바이트 버퍼와 바이트 개수
            typedef bool(*pfSendData)(byte*, int);

            //dll에 송신하려는 문자열 추가함수
            typedef bool(*pfAddLineDLL)(PCHAR, PCHAR, PCHAR);

            //dll에서 수신받은 메세지를 Queue에서 한개씩 수신받는 함수
            typedef char*(*fGetReceiveMessageInQueueBuffer)(bool,bool);

            //Dll 초기화 처리 함수
            //plc로부터 받은 메세지를 저장하는 Queue 사용유무
            typedef bool(*pfInitializeDll)(bool bUseReceiveQueueBufferFromPLC);

            //dll 통신 연결 상태 반환
            typedef bool(*pfIsConnected)(void);

            //ascii 타입 통신 타입이면 true, binary 타입이면 false
            typedef bool(*pfIsASCIIType)(void);

            //Dll 종료 호출 함수
            typedef bool(*pfExitDll)(void);

            // IP Config Dialog 화면 보이기/숨기기
            typedef bool(*pfToggleViewHideIPConfigDialog)(bool, bool);

            //Dll에서 통신 연결 상태 CallBack 되는 함수 등록
            typedef void(*fRegister_CallBackConnectStatusFuntion)(callback_ConnectionStatusfunction);

            //Dll에서 통신 수신 후 callback 되는 함수 등록
            typedef void(*fRegister_CallBackReceiveFunction)(callback_Bufferfunction);
}
```

b. **"LoadVTECSDll.h"**

```
#pragma once
#include "./DefineDllFunctionPoint.h"
#include <afxmt.h>

namespace LOADVTECSDll_GROUP
{
class LoadVTECSDll//:private stDllStatus
{
public:
	LoadVTECSDll(void);
	~LoadVTECSDll(void);

private:
	//dll HANDLE 변수
	HMODULE m_hdll;

	//dll 파일 및 경로 정보를 저장한 변수
	CString m_strDllFile;

	//dll load/unload/call process/attack process 인터럽터
	CCriticalSection	m_CriticalLoadUnload;

private:

	//dll 초기화 처리 함수
	pfInitializeDll			m_fInitializeDll;

	//dll 초기화가 되어 있는지 확인하는 함수
	pfIsConnected			m_fIsConnected;

	/// <summary>
	/// ASCII 타입통신이면 true, binary 타입 통신이면 false
	/// </summary>
	pfIsASCIIType			m_fIsASCIIType;

	//dll 종료처리 함수
	pfExitDll			m_fExitDll;

	//바이트 단위로 통신 송신 함수
	pfSendData			m_fSendData;

	//문자열을 추가하는 함수
	pfAddLineDLL			m_fAddLine;

	//수신받은 메세지 첫번째 Queue 요소 메세지를 반환받는 함수
	fGetReceiveMessageInQueueBuffer	m_fGetReceiveMessageInQueueBuffer;

	//Dll 통신 설정 화면
	pfToggleViewHideIPConfigDialog	m_fToggleViewHideIPConfigDialog;

	//Dll에서 통신 연결 상태 CallBack 되는 함수 등록
	fRegister_CallBackConnectStatusFuntion		m_fRegister_CallBackConnectStatusFuntion;

	//DLL에서 통신 수신이 되면 Callback 되는 함수 등록
	fRegister_CallBackReceiveFunction		m_fRegister_CallBackReceiveFunction;

public:

	//dll 파일 Unloading 처리
	//인자1		:	없음
	//반환		:	정상이면 TRUE, 그렇지 않으면 FALSE
	bool DllSocket_Unitialize(void);

	//dll 초기화 및 버전 반환
	//인자1		:통신 연결 상태가 변경되면 호출하는 call back 함수 접근주소
	//인자2		:통신 수신이 되면 호출하는 call back 함수 접근 주소
	//인자3		:수신 Queue 버퍼 사용이면 true, 사용하지 않으면 false
	//반환		:정상적으로 초기화 되면 true, 그렇지 않으면 false
	bool DllSocket_Initialize(callback_ConnectionStatusfunction Callback_Connect, callback_Bufferfunction Callback_Receive
= NULL, const bool bUseReceiveQueueBuffer = true);
```

```
		/// <summary>
		/// dll 초기화-동일한 폴더에서 실행파일이 dll 초기화 할 때 사용
		/// </summary>
		/// <param name="strDllFold">dll 파일이 있는 경로 문자열</param>
		/// <param name="Callback_Connect">통신 연결 상태가 변경되면 호출하는 call back 함수 접근주소</param>
		/// <param name="Callback_Receive">통신 수신이 되면 호출하는 call back 함수 접근 주소</param>
		/// <param name="bUseReceiveQueueBuffer">수신 Queue 버퍼 사용이면 true, 사용하지 않으면 false</param>
		/// <returns></returns>
		bool DllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect,
callback_Bufferfunction Callback_Receive = NULL, const bool bUseReceiveQueueBuffer = true);

		//dll 연결이 되면 TRUE, 그렇지 않으면 FALSE
		//인자1	:		없음
		//반환	:		초기화 되어 있으면 TRUE
		bool DllSocket_IsConnected(void);

		//dll에서 수신받은 Queue 버퍼에서 첫번째 메세지를 획득하는 함수
		//인자1	:		가장 최근에 받은 수신 메세지를 받고 싶으면 true, 가장 과거에 받은 수신 메세지를 받고 싶으면 false
		//인자1	:		함수 호출과 동시에 수신 메세지 Queue 버퍼를 제거(기본 제거: 권장)
		//반환	:		수신받은 메세지를 문자열로 변환한 것, NULL이면 수신받은 것이 없습니다.
		char* DllSocket_GetReceiveMessageInQueue(const bool bLastReceiveMsgReturn, const bool bDeleteRemainReceiveQueue = true);

		/// <summary>
		/// ASCII 타입 통신이면 true, binary 타입
		/// </summary>
		/// <param name=""></param>
		/// <returns>ASCII 타입이면 true, Binary 타입이면 false</returns>
		bool DllSocket_IsASCIITypeDll(void);

		//송신할 메세지를 추가하는 함수
		//인자1		:		셀 id 문자열
		//인자1		:		문자열이 저장되는 파일 경로 및 파일 이름 포함
		//인자2		:		추가할 Data 메세지 문자열
		//반환		:		정상이면 TRUE
		bool DllSocket_AddToECS(CString& strCellID, CString& strFilePathName, CString& strData);

		/// <summary>
		/// 바이트 버퍼와 바이트 개수를 대입하여 이더넷으로 송신
		/// </summary>
		/// <param name="pSendBuffer">송신할 바이트 버퍼</param>
		/// <param name="nSendLength">송신할 바이트 개수</param>
		/// <returns>정상적으로 송신할 경우 true</returns>
		bool DllSocket_Send(unsigned char* pSendBuffer, const int nSendLength);

		/// <summary>
		/// DLL IP Config 창 보이기/숨기기
		/// </summary>
		/// <param name="bShow">창 보이기 true, 숨기기 false</param>
		/// <param name="bToggleModeOnIsTrue">창 보이기 토글 모드 사용이면 true, 토글모드에서는 bShow 값은 무시 됩니다.</param>
		/// <returns></returns>
		bool DllSocket_IPConfigToggleShow(bool bShow, bool bToggleModeOnIsTrue);

		//dll 상태를 반환한다.
		//인자1		:		없음
		//반환		:		정상이면 TRUE
		bool DllSocket_IsIntialized(void);

private:

		/// <summary>
		/// dll 초기화-동일한 폴더에서 실행파일이 dll 초기화 할 때 사용
		/// </summary>
		/// <param name="strDllFold"></param>
		/// <param name="Callback_Connect">통신 연결 상태가 변경되면 호출하는 call back 함수 접근주소</param>
		/// <param name="Callback_Receive">통신 수신이 되면 호출하는 call back 함수 접근 주소</param>
		/// <param name="bUseReceiveQueueBuffer">수신 Queue 버퍼 사용이면 true, 사용하지 않으면 false</param>
		/// <returns></returns>
		bool SubDllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect,
callback_Bufferfunction Callback_Receive = NULL, const bool bUseReceiveQueueBuffer = true);
```

```
        //dll 파일 Loading 처리
        //인자1                 :         dll파일이 있는 파일 경로 지정 마지막 문자열에 'WW' 추가 필요
        //반환                  :         정상이면 TRUE, 그렇지 않으면 FALSE
        bool LoadingDll(const CString strToLoadDllPathName);

        //포인터 함수 연결 함수
        //인자1                 :         없음
        //반환                  :         정상이면 TRUE, 그렇지 않으면 FALSE
        bool LinkFuntion(void);

        //포인터 함수를 모두 NULL 처리하는 함수
        //인자1                 :         없음
        //반환                  :         없음
        void LinkFuntionToAllNULL(void);

        ////////////////////////////////////////////////////////////////////////// End Class
};
////////////////////////////////////////////////////////////////////////////// End Namespace
```

c. **"LoadVTECSDll.cpp"**

```
#include "Pch.h"
#include "LoadVTECSDll.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

namespace LOADVTECSDll_GROUP
{
	LoadVTECSDll::LoadVTECSDll(void)
	{
		//dll 모듈 초기화
		m_hdll = NULL;

#ifdef _WIN64
		m_strDllFile = _T("SPCTransUtilx64.dll");
#else
		m_strDllFile = _T("SPCTransUtilx86.dll");
#endif // WIN32

		//함수 포인터 초기화
		LinkFuntionToAllNULL();
	}

	LoadVTECSDll::~LoadVTECSDll(void)
	{
		DllSocket_Unitialize();
	}

	bool LoadVTECSDll::LoadingDll(void)
	{
		try
		{
			if (NULL == m_hdll)
			{
				m_hdll = GetModuleHandle(m_strDllFile);
				if (NULL == m_hdll)
				{
					m_hdll = LoadLibrary(m_strDllFile);
				}
			}
		}
		catch (CException* exc)
		{
			CString str;
			exc->GetErrorMessage(str.GetBuffer(MAX_PATH), MAX_PATH);
			exc->Delete();
			str.ReleaseBuffer();

			TRACE(_T("Error in LoadingDll[%s]Wn"), str);
			m_hdll = NULL;
			exc->Delete();
			m_CriticalLoadUnload.Unlock();
			return false;
		}
		return NULL != m_hdll;
	}

	bool LoadVTECSDll::DllSocket_Unitialize(void)
	{
		m_CriticalLoadUnload.Lock();

		bool bExitOK = true;
		try
		{
			if (NULL == m_hdll && !m_strDllFile.IsEmpty())
			{
				m_hdll = GetModuleHandle(m_strDllFile);
			}
```

```
                                if (NULL != m_hdll)
                                {
                                        if (NULL != m_fExitDll)
                                        {
                                                bExitOK = m_fExitDll();
                                        }

                                        if (FreeLibrary(m_hdll))
                                        {
                                                m_hdll = NULL;
                                        }

                                        LinkFuntionToAllNULL();
                                }
                        }
                        catch (CException* exc)
                        {
                                TRACE(_T("Error in UnloadingDllWn"));
                                exc->Delete();
                                bExitOK = false;
                                m_hdll = NULL;
                        }
                        bExitOK &= (NULL == m_hdll);
                        m_CriticalLoadUnload.Unlock();
                        return bExitOK;
                }

                bool LoadVTECSDll::LinkFuntion(void)
                {
                        if (NULL != m_hdll)
                        {
                                if (NULL == m_fInitializeDll)
                                {
                                        m_fInitializeDll = (pfInitializeDll)GetProcAddress(m_hdll, "InitializeDll");
                                }
                                if (NULL == m_fIsConnected)
                                {
                                        m_fIsConnected = (pfIsConnected)GetProcAddress(m_hdll, "IsConnected");
                                }
                                if (NULL == m_fIsASCIIType)
                                {
                                        m_fIsASCIIType = (pfIsASCIIType)GetProcAddress(m_hdll, "IsASCIIType");
                                }
                                if (NULL == m_fExitDll)
                                {
                                        m_fExitDll = (pfExitDll)GetProcAddress(m_hdll, "Exitdll");
                                }
                                if (NULL == m_fSendData)
                                {
                                        m_fSendData = (pfSendData)GetProcAddress(m_hdll, "SendData");
                                }
                                if (NULL == m_fAddLine)
                                {
                                        m_fAddLine = (pfAddLineDLL)GetProcAddress(m_hdll, "AddLine");
                                }
                                if (NULL == m_fGetReceiveMessageInQueueBuffer)
                                {
                                        m_fGetReceiveMessageInQueueBuffer =
(fGetReceiveMessageInQueueBuffer)GetProcAddress(m_hdll, "GetReceiveMessageInQueueBuffer");
                                }

                                if (NULL == m_fToggleViewHideIPConfigDialog)
                                {
                                        m_fToggleViewHideIPConfigDialog = (pfToggleViewHideIPConfigDialog)GetProcAddress(m_hdll,
"ToggleViewHideIPConfigDialog");
                                }

                                if (NULL == m_fRegister_CallBackConnectStatusFuntion)
                                {
                                        m_fRegister_CallBackConnectStatusFuntion =
(fRegister_CallBackConnectStatusFuntion)GetProcAddress(m_hdll, "RegCallBackFun_ConnectStatus");
                                }
```

```
                                if (NULL == m_fRegister_CallBackReceiveFunction)
                                {
                                        m_fRegister_CallBackReceiveFunction =
(fRegister_CallBackReceiveFunction)GetProcAddress(m_hdll, "RegCallBackFun_ReceiveBuffer");
                                }
                        }
                        else
                        {
                                LinkFuntionToAllNULL();
                        }
                        return      NULL != m_fAddLine && NULL != m_fInitializeDll && NULL != m_flsConnected && NULL != m_flsASCIIType &&
                                NULL != m_fToggleViewHideIPConfigDialog && NULL != m_fSendData && NULL !=
m_fGetReceiveMessageInQueueBuffer &&
                                NULL != m_fExitDll && NULL != m_fRegister_CallBackConnectStatusFuntion && NULL !=
m_fRegister_CallBackReceiveFunction;
        }

        bool LoadVTECSDll::DllSocket_Initialize(callback_ConnectionStatusfunction Callback_Connect, callback_Bufferfunction
Callback_Receive, const bool bUseReceiveQueueBuffer)
        {
                m_CriticalLoadUnload.Lock();
                CString strPath;
                GetCurrentDirectory(MAX_PATH, strPath.GetBuffer(MAX_PATH));
                strPath.ReleaseBuffer();
                strPath += _T('WW');
                const bool bReturn = SubDllSocket_Initialize(strPath, Callback_Connect, Callback_Receive,
bUseReceiveQueueBuffer);
                m_CriticalLoadUnload.Unlock();
                return bReturn;
        }

        bool LoadVTECSDll::DllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect,
callback_Bufferfunction Callback_Receive, const bool bUseReceiveQueueBuffer)
        {
                m_CriticalLoadUnload.Lock();
                const bool bReturn = SubDllSocket_Initialize(strDllFold, Callback_Connect, Callback_Receive,
bUseReceiveQueueBuffer);
                m_CriticalLoadUnload.Unlock();
                return bReturn;
        }

        bool LoadVTECSDll::SubDllSocket_Initialize(const CString strDllFold, callback_ConnectionStatusfunction Callback_Connect,
callback_Bufferfunction Callback_Receive, const bool bUseReceiveQueueBuffer)
        {
                if (LoadingDll(strDllFold) &&  LinkFuntion())
                {
#ifdef _UNICODE
                        CStringA strDll;
                        strDll = strDllFold;
#else
                        CString strDll = strDllFold;
#endif // _UNICODE

                        //초기화
                        if (m_fInitializeDll((char*)strDll.GetString(), bUseReceiveQueueBuffer))
                        {
                                if (NULL != m_fRegister_CallBackConnectStatusFuntion && NULL != Callback_Connect)
                                {
                                        m_fRegister_CallBackConnectStatusFuntion(Callback_Connect);
                                }

                                if (NULL != m_fRegister_CallBackReceiveFunction && NULL != Callback_Receive)
                                {
                                        m_fRegister_CallBackReceiveFunction(Callback_Receive);
                                }
                                return true;
                        }
                }
                return false;
        }
```

```
bool LoadVTECSDll::DllSocket_IsConnected(void)
{
	m_CriticalLoadUnload.Lock();

	if (NULL != m_flsConnected)
	{
		const bool bReturn = m_flsConnected();
		m_CriticalLoadUnload.Unlock();
		return bReturn;
	}

	m_CriticalLoadUnload.Unlock();
	return false;
}

char* LoadVTECSDll::DllSocket_GetReceiveMessageInQueue(const bool bLastReceiveMsgReturn, const bool
bDeleteRemainReceiveQueue)
{
	m_CriticalLoadUnload.Lock();

	if (NULL != m_fGetReceiveMessageInQueueBuffer)
	{
		char* pReturn = m_fGetReceiveMessageInQueueBuffer(bLastReceiveMsgReturn,
bDeleteRemainReceiveQueue);
		m_CriticalLoadUnload.Unlock();
		return pReturn;
	}
	m_CriticalLoadUnload.Unlock();
	return NULL;
}

bool LoadVTECSDll::DllSocket_IsASCIITypeDll(void)
{
	m_CriticalLoadUnload.Lock();

	if (NULL != m_flsASCIIType)
	{
		const bool bReturn = m_flsASCIIType();
		m_CriticalLoadUnload.Unlock();
		return bReturn;
	}
	m_CriticalLoadUnload.Unlock();
	return false;
}

bool LoadVTECSDll::DllSocket_AddToECS(CString& strCellID, CString& strFilePathName, CString& strData)
{
	m_CriticalLoadUnload.Lock();

	if (NULL != m_fAddLine && !strFilePathName.IsEmpty() && !strData.IsEmpty())
	{
#ifdef _UNICODE
		CStringA strA, strB;
		strA = strFilePathName;
		strB = strData;
		const bool bReturn = m_fAddLine((PCHAR)strCellID.GetString(), (PCHAR)strA.GetString(),
(PCHAR)strB.GetString());
#else
		const bool bReturn = m_fAddLine((PCHAR)strCellID.GetString(), (PCHAR)strFilePathName.GetString(),
(PCHAR)strData.GetString());
#endif // _UNICODE

		m_CriticalLoadUnload.Unlock();
		return bReturn;
	}

	m_CriticalLoadUnload.Unlock();
	return false;
}
```

```
	bool LoadVTECSDll::DllSocket_Send(unsigned char* pSendBuffer, const int nSendLength)
	{
		m_CriticalLoadUnload.Lock();

		if (NULL != m_fSendData && NULL!= pSendBuffer && nSendLength>0)
		{
			const bool bReturn = m_fSendData(pSendBuffer, nSendLength);
			m_CriticalLoadUnload.Unlock();
			return bReturn;
		}
		m_CriticalLoadUnload.Unlock();
		return false;
	}

	bool LoadVTECSDll::DllSocket_IPConfigToggleShow(bool bShow, bool bToggleModeOnIsTrue)
	{
		m_CriticalLoadUnload.Lock();

		if (NULL != m_fToggleViewHideIPConfigDialog)
		{
			const bool bReturn = m_fToggleViewHideIPConfigDialog(bShow, bToggleModeOnIsTrue);
			m_CriticalLoadUnload.Unlock();
			return bReturn;
		}
		m_CriticalLoadUnload.Unlock();
		return false;
	}

	bool LoadVTECSDll::DllSocket_IsIntialized(void)
	{
		m_CriticalLoadUnload.Lock();

		if	(NULL != m_fAddLine && NULL != m_fInitializeDll && NULL != m_fIsConnected && NULL != m_fIsASCIIType
&&
			NULL != m_fToggleViewHideIPConfigDialog && NULL != m_fSendData && NULL !=
m_fGetReceiveMessageInQueueBuffer &&
			NULL != m_fExitDll && NULL != m_fRegister_CallBackConnectStatusFuntion && NULL !=
m_fRegister_CallBackReceiveFunction)
		{
			m_CriticalLoadUnload.Unlock();
			return true;
		}
		m_CriticalLoadUnload.Unlock();

		return false;
	}

	void LoadVTECSDll::LinkFuntionToAllNULL(void)
	{
		m_fInitializeDll = NULL;

		m_fExitDll = NULL;

		m_fIsConnected = NULL;

		m_fIsASCIIType = NULL;

		m_fSendData = NULL;

		m_fAddLine = NULL;

		m_fGetReceiveMessageInQueueBuffer = NULL;

		m_fToggleViewHideIPConfigDialog = NULL;

		m_fRegister_CallBackConnectStatusFuntion = NULL;

		m_fRegister_CallBackReceiveFunction = NULL;
	}
	//////////////////////////////////////////////////////////////////////////////// End Namespace
}
```

8) Dll Socket 통신 기타 정의

① 통신 연결 상태 값 정의

```
/// <summary>
/// 통신 이벤트 정의
/// </summary>
enum enConnectEvent : signed char
{
    OnError = -1,
    OnInvalid,
    OnStart_CreateSocketMemory,     //소켓 메모리 생성하려고 시작
    OnReady,                        //서버 타입인 경우 client에서 통신 접속을 기다리는 상태
    OnConnected,                    //통신 연결상태
    OnConnectFail,                  //통신 연결 실패 상태
    OnDisConnected,                 //통신 연결 해제 상태
    OnTerminated,                   //통신 연결 제어 이벤트 처리부 해제 상태
    OnCloseComm                     //통신 control close 상태
}
```

② 통신 데이터 형식 정의

```
/// <summary>
/// 통신 데이터 방식
/// </summary>
public enum enASCIIBinaryType : int
{
    InvalidType = -1,               //통신 연결 제어기나 기타 이유로 dll 함수가 할당되어 있지 않을 겨우
    BinaryTypeDll,                  //Binary 통신 타입
    ASCIITypeDll                    //ASCII 통신 타입
}
```

9) IP 주소 설정 파일과 로그 저장 파일 위치

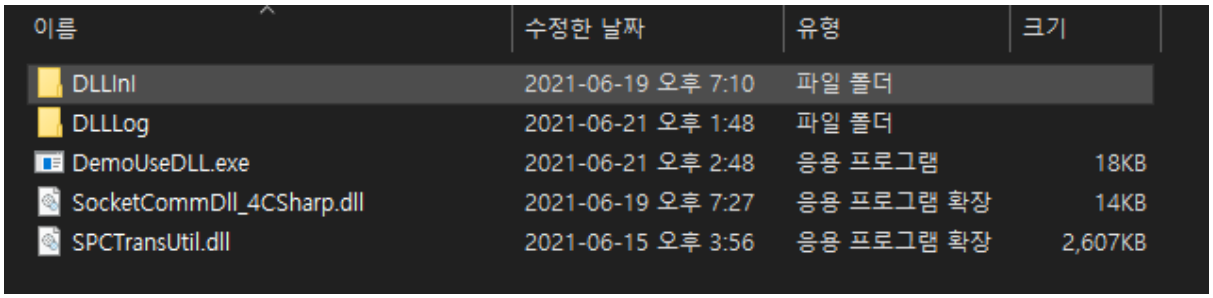
① 설정 정보 파일

a. **SPCTransUtil.dll 파일이 있는 폴더의 "DllIni"에 존재**

b. **"IPConfig.ini" 파일에 IP 설정값을 저장한다.**

② 로그 파일 저장 위치와 내용

a. **SPCTransUtil.dll 파일이 있는 폴더의 "DllLog"에 존재**



b. **로그 파일 내용(Binary 통신 예, 16진수 2자리(00~ff), 4바이트 구분자는 ‘_’)**

SPCTransUtilDllLog_20210621.log

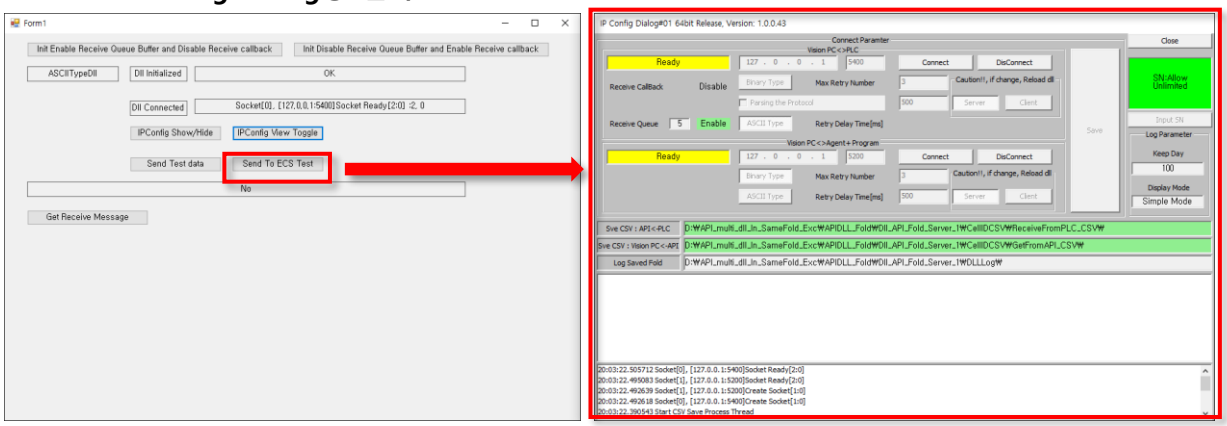
```
13:48:41.503000 >>>>>>>>> Initialized D:\Code\ECS_DLL\DemoUseDLL\bin\SPCTransUtil.Dll Version[1.0.0.4] 
<<<<<<<<<
13:48:41.504543 the IP Address Setting Ini file loading : Success
13:48:41.507600 [192.168.1.201:4002]Create Socket[1:0]
13:48:41.511298 [192.168.1.201:4002]Socket Ready[2:0]
13:48:41.511816 Socket Connect E-Code(10035), nonfatal type code
13:48:41.515717 [192.168.1.201:4002]Socket Connected[3:0]
13:49:03.900272 DLL->PC : HEX : 
00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000
000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_0
0000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_000000
00_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00
000000_00000000_009874b6_f77f0000_203d2490_1d020000_503d2490_1d020000_183e2490_1d020000_00000000_00000000_0000000
0_00000000_103e638e_1d020000_00000000_00000000_00000000_00000000_68aa74b6_f77f0000_78801d90_1d020000_00000000_000
00000_500585b6_f77f0000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_20ac74b6_f77f0000
_78801d90_1d020000_00000000_00000000_400585b6_f77f0000_00000000_00000000_00000000_00000000_00000000_00000000[200]

13:49:03.928098 DLL<-PC : HEX : 
00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000
000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_0
0000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_000000
00_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00
000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_0000000
0_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_000
00000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000
_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000[200]

```

10) 시뮬레이션 환경 설정 및 디버깅

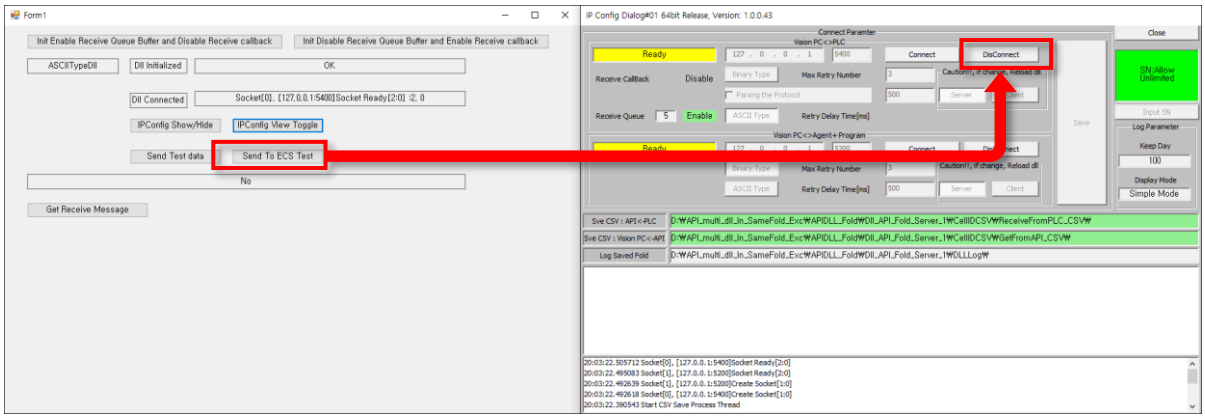
① PLC 역할을 하는 서버(Server)용 API 실행- Client➔Server 변경되는 경우

a. **API 예제 프로그램을 관리자 권한으로 실행**

b. **IP Config Dialog창 열기**

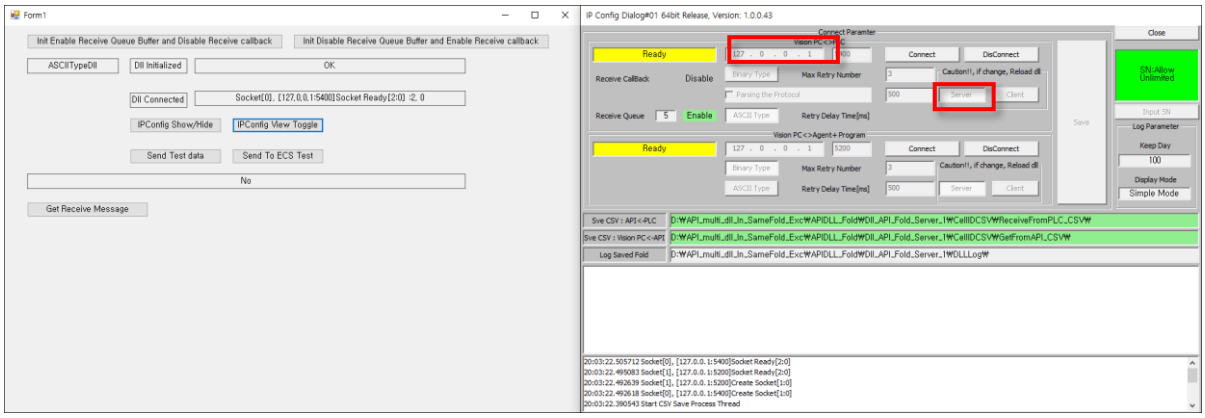


c. **통신 Disconnect 선택**

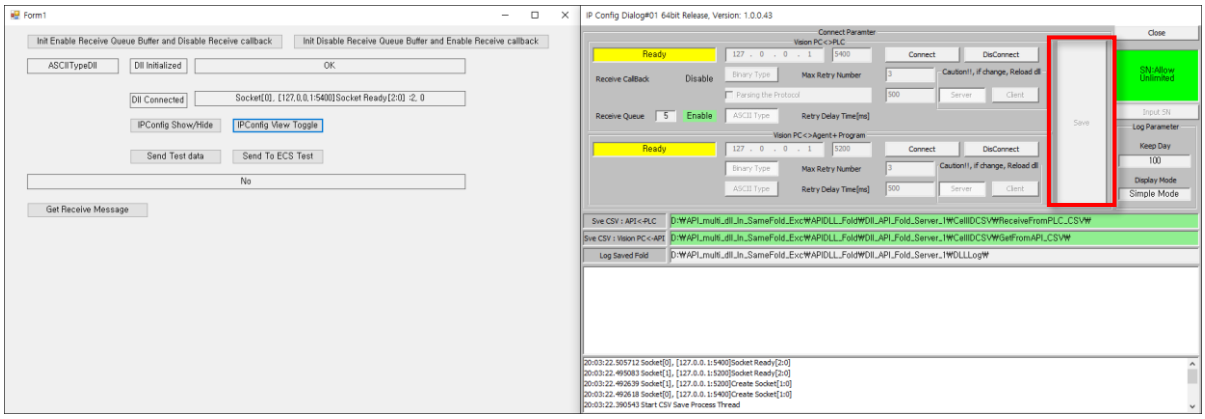


d. **IP 주소, 포트번호 편집 및 서버(Server) 선택**

단, IP 주소값이 "0.0.0.0"은 Any Address, 127.0.0.1은 가상 IP Address입니다.



e. **저장**

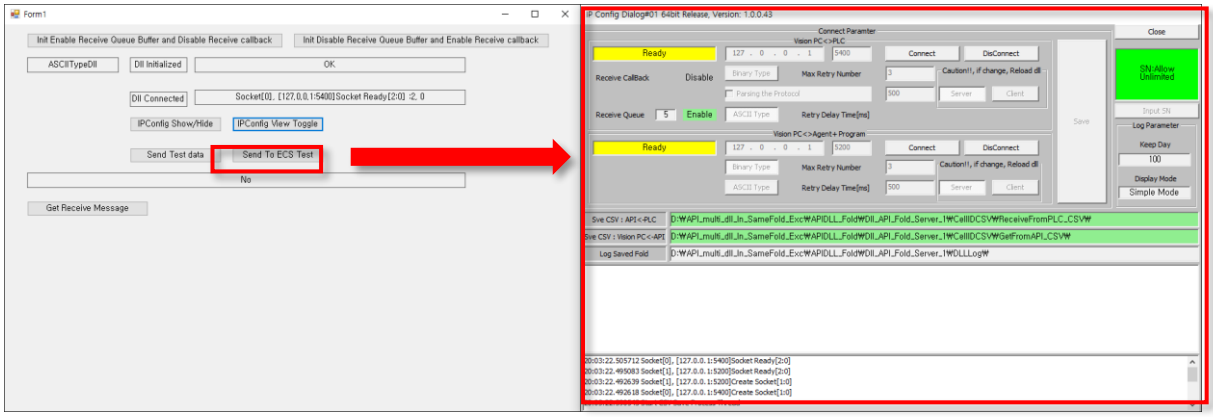


f. **API 예제 프로그램을 Visual Studio 관리자 권한으로 디버깅 모드 재시작**

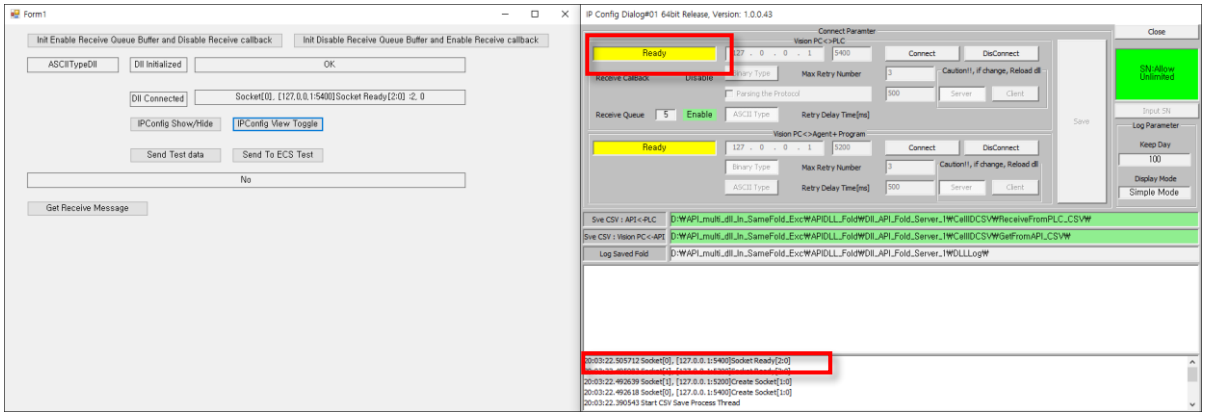
② PLC 역할을 하는 서버(Server)용 API 실행-서버 설정이 되어 있는 경우

a. **API 예제 프로그램을 관리자 권한으로 실행**

b. **IP Config Dialog창 열기**



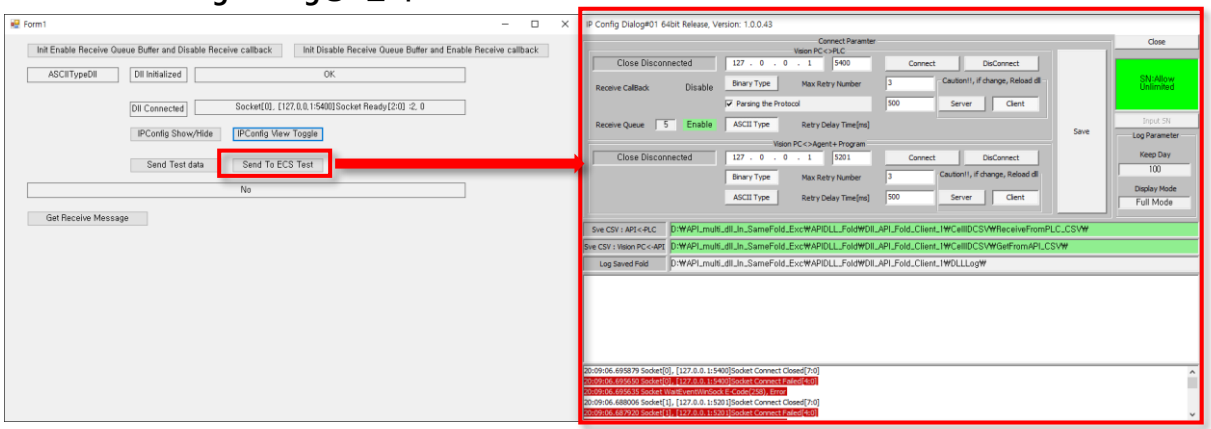
c. **서버(Server) IP 주소와 포트 번호 확인**



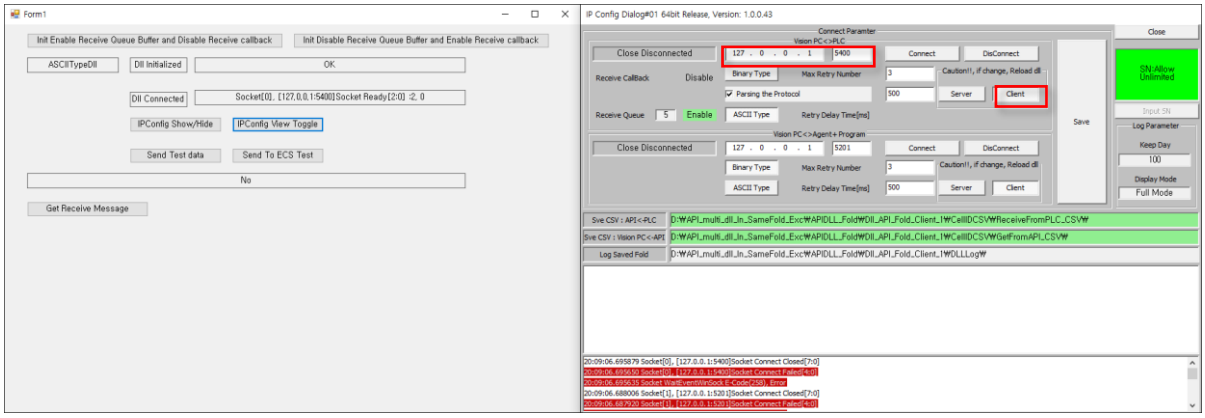
③ 개발 및 디버깅 프로그램 클라이언트(Client)용 API 실행

a. **API 포함 프로그램을 Visual Studio 관리자 권한으로 디버깅 모드 시작**

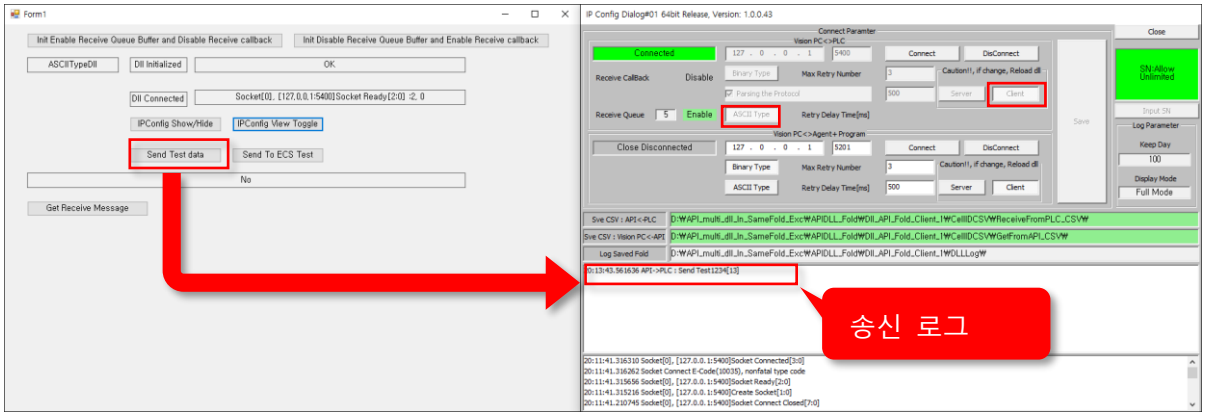
b. **IP Config Dialog창 열기**

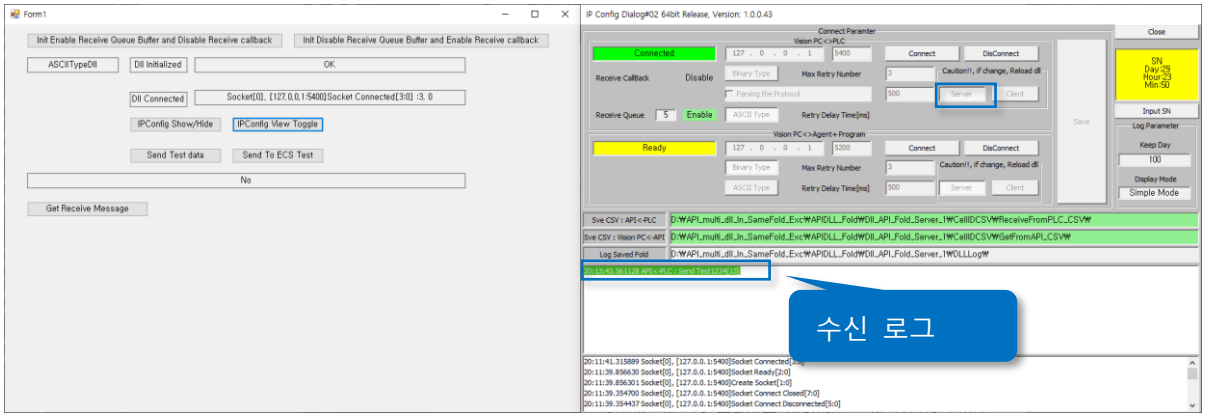


c. **시뮬레이션 서버(Server)의 IP 주소, 포트 번호, Client 선택 확인(red box)**



d. **ASCII 타입에서 통신 송/수신 테스트**





④ Visual Studio에서 "DllSocket_ReceiveDataBuffer" 함수내 중단점을 선언하여 수신 데이터 디버깅 합니다.

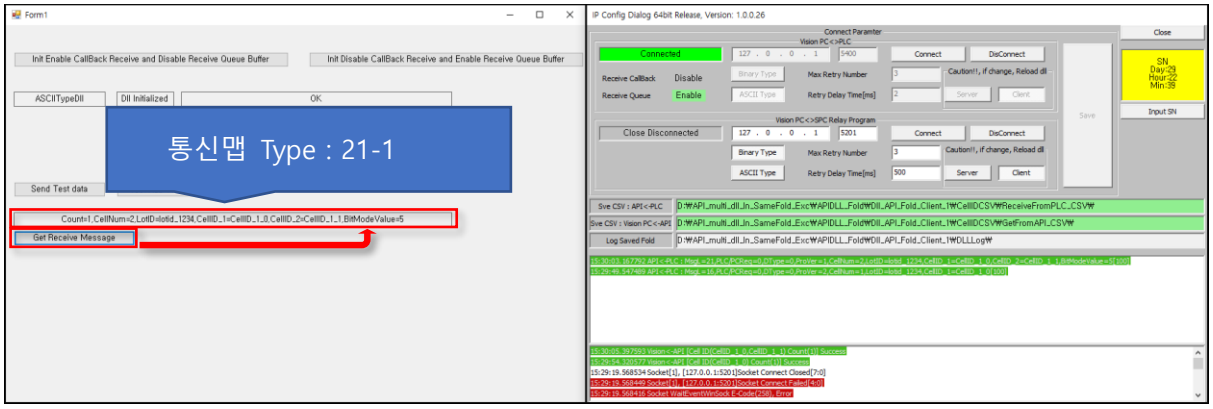
⑤ Visual Studio에서 "DllSocket_ConnectStatus" 함수내 중단점을 선언하여 통신 연결 상태 디버깅 합니다.

11) API에서 수신 받은 Queue 버퍼의 첫번째 요소(Element) 정보를 획득하는 함수 추가

① C# 함수

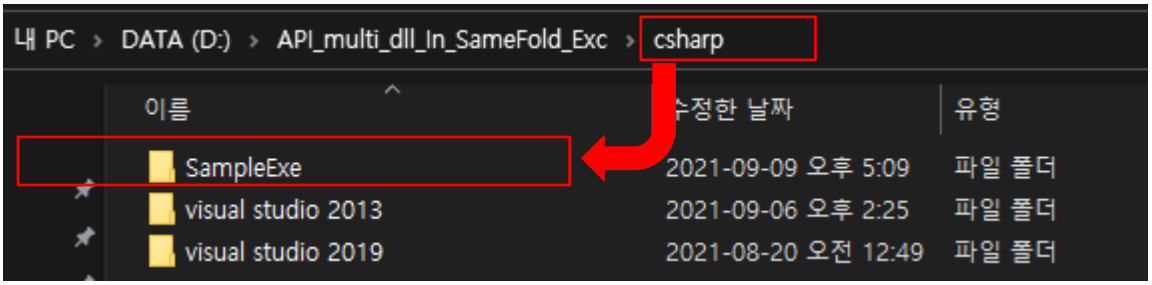
```
        /// <summary>
        /// 수신받은 Qeueu 첫번째 message를 문자열로 수신받는 함수
        /// </summary>
        /// <param name="bLastReceiveReturn">가장 최근 수신 Message 반환이면 true, 가장 과거 수신 Message 반환이면 false</param>
        /// <param name="bDeleteRemainQueueBuffer">함수 호출과 동시에 남겨진 message가 있다면 모두 제거조건이면 true</param>
        /// <returns>수신받은 메세지를 문자열 형태로 반환</returns>
        public string DllSocket_GetReceiveMessageInQueue(bool bLastReceiveReturn, bool bDeleteRemainQueueBuffer)
        {
            lock(m_Locking)
            {
                if(null!=DllSocket)
                {
                    return DllSocket.GetReceiveMessageInQueueBuffer(bLastReceiveReturn, bDeleteRemainQueueBuffer);
                }
                return "";
            }
        }
```

a. **예제 프로그램에서 정상적으로 수신 받은 문자열 표시**

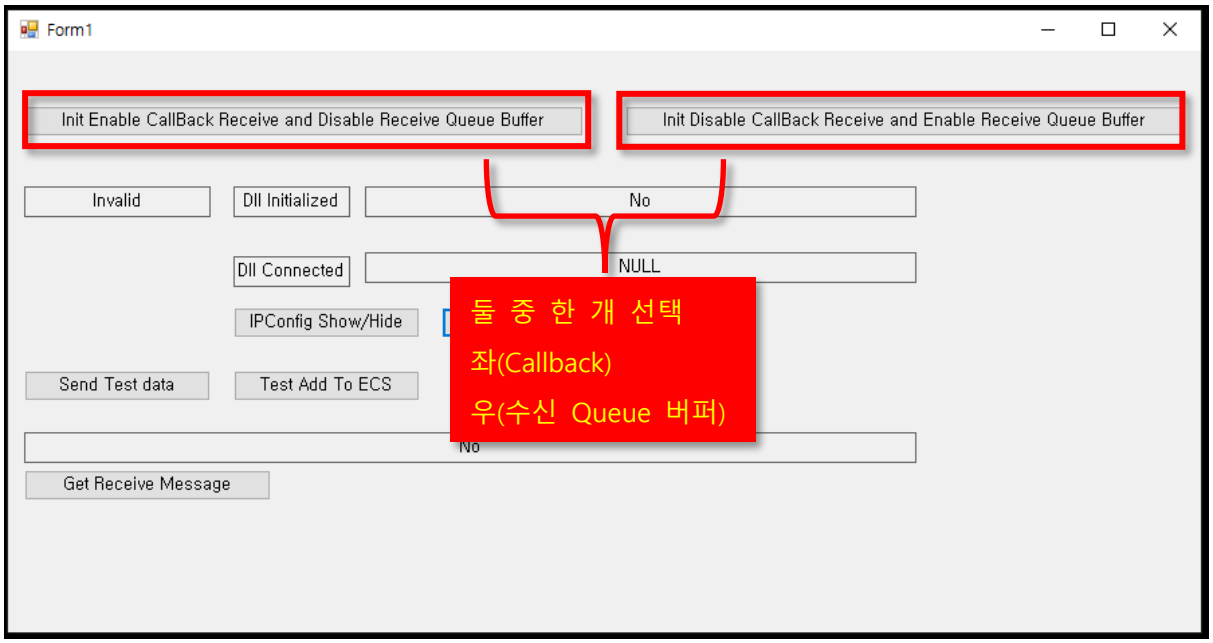


단, 기존 비트별 표시를 BitModeValue값으로 변경, BitmodeValue값은 16비트를 양수로 표시

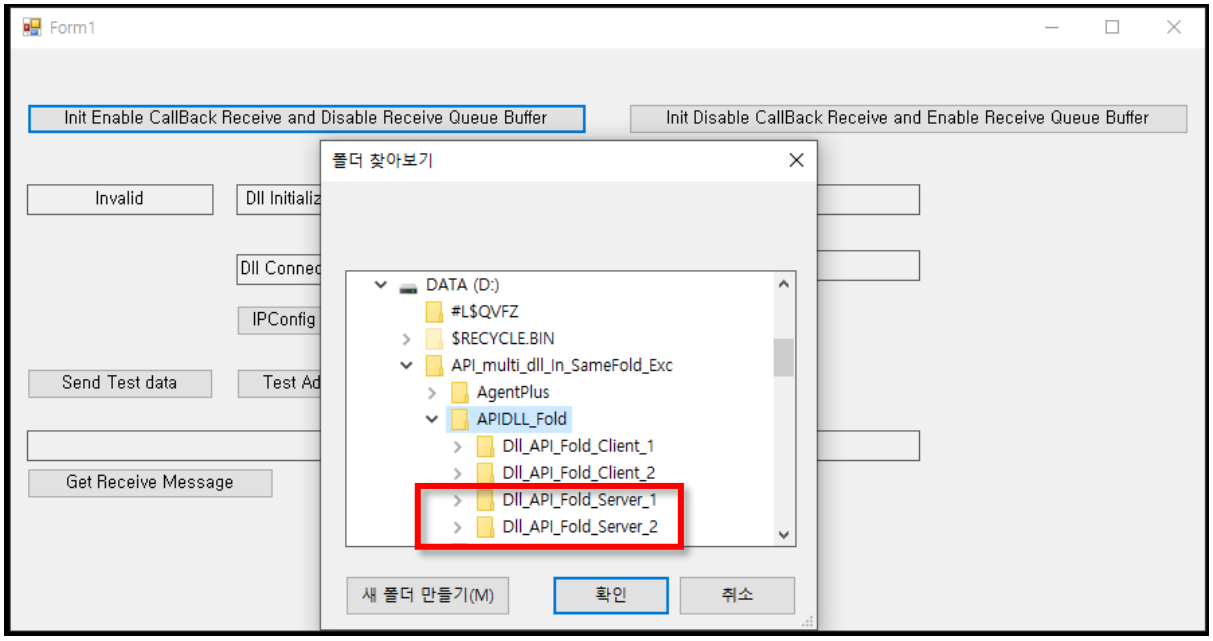
b. **예제 폴더 "API_multi_dll_In_SameFold_Exc₩csharp"에서 "SampleExe" 폴더 내 실행파일을 먼저 실행한다.(실행 테스트)**



c. **Callback 방식이나 Queue Buffer 방식 둘 중 한 개를 선택하여 초기화 버튼을 클릭한다.**

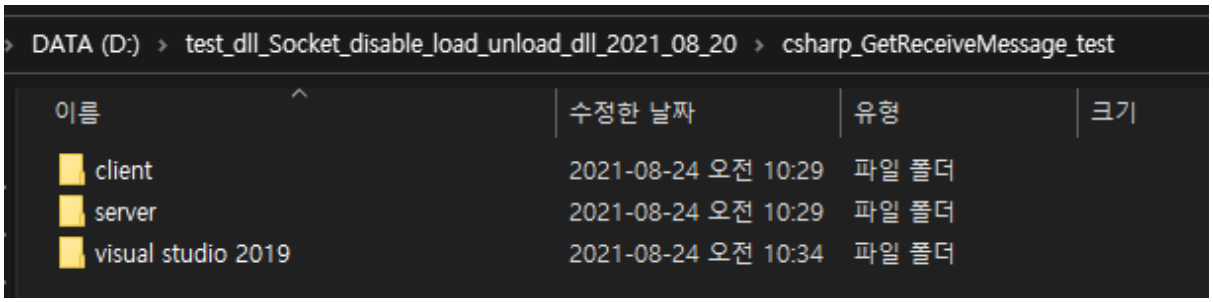


d. **폴더 선택창에서 "API_multi_dll_In_SameFold_Exc₩APIDLL_Fold"에 있는 "Server_1 혹은 Server_2"를 선택한다.**

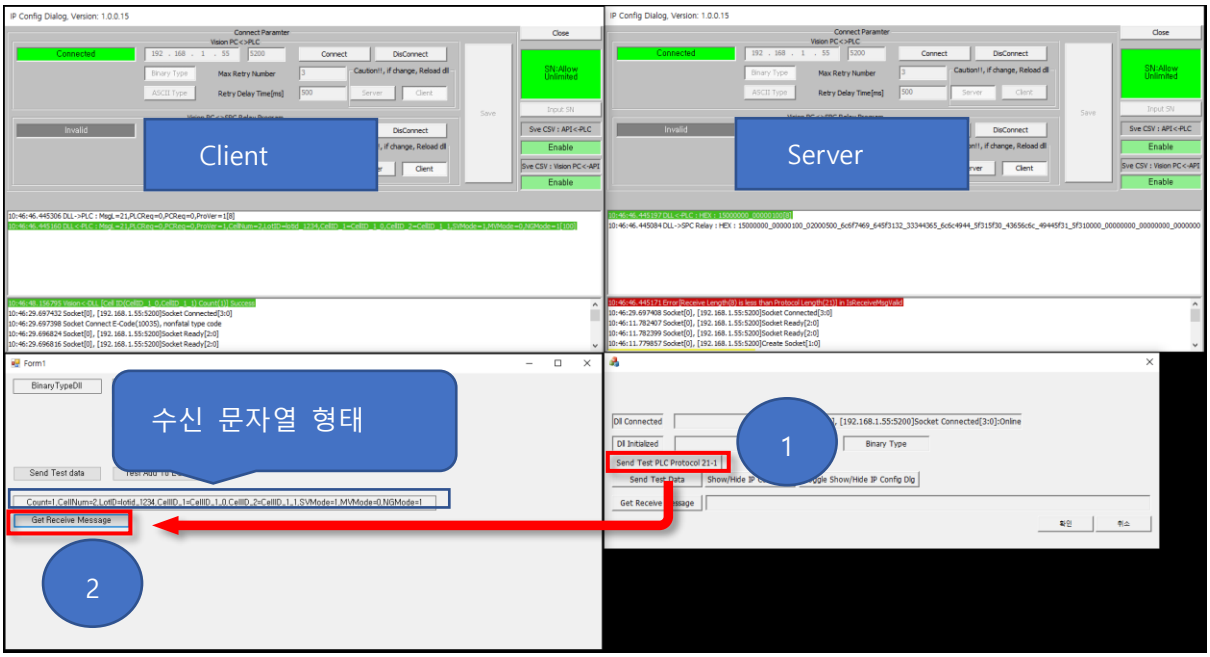


e. **예제 폴더 "csharp_GetReceiveMessage_test"에서 "Server"내 실행파일을 먼저 실행한다.**

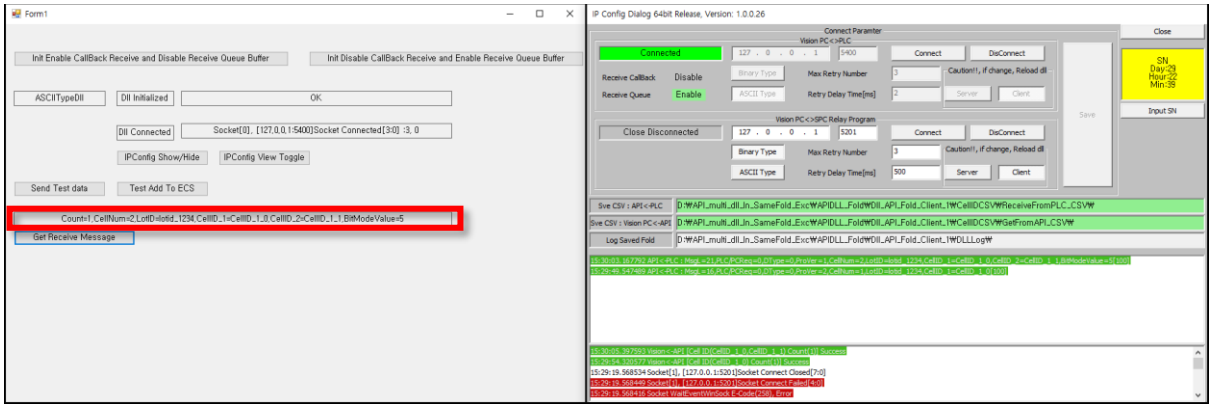
"Visual Studio 2019"폴더 내 예제 코드를 Loading한다.(디버그 테스트)



f. **PLC에서 수신 받은 첫번째 Message를 획득을 확인하는 방법**



단, 위 그림에서 SVMode=1,MyMode=0,NGMode=1은 BitModeValue로 통합 변경되었습니다.



② C++

```
//dll에서 수신받은 Queue 버퍼에서 첫번째 메세지를 획득하는 함수
//인자1                 :          가장 최근에 받은 수신 메세지를 받고 싶으면 true, 가장 과거에 받은 수신 메세지를 받고 싶으면 false
//인자1                 :          함수 호출과 동시에 수신 메세지 Queue 버퍼를 제거(기본 제거: 권장)
//반환                  :          수신받은 메세지를 문자열로 변환한 것, NULL이면 수신받은 것이 없습니다.
char* DllSocket_GetReceiveMessageInQueue(const bool bLastReceiveMsgReturn, const bool bDeleteRemainReceiveQueue = true);
```

```
char* LoadVTECSDll::DllSocket_GetReceiveMessageInQueue(const bool bLastReceiveMsgReturn, const bool bDeleteRemainReceiveQueue)
{
	m_CriticalLoadUnload.Lock();
	if (NULL != m_fGetReceiveMessageInQueueBuffer)
	{
		char* pReturn = m_fGetReceiveMessageInQueueBuffer(bLastReceiveMsgReturn, bDeleteRemainReceiveQueue);
		m_CriticalLoadUnload.Unlock();
		return pReturn;
	}
	m_CriticalLoadUnload.Unlock();
	return NULL;
}
```

12) API CSV 파일 저장

① 파일 저장 정보

a. **API←PLC : Lot ID, Cell ID 정보**

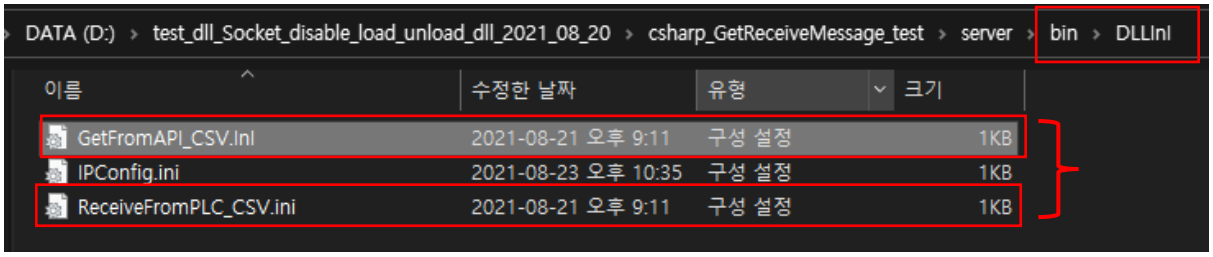
b. **Vision PC←API : Lot ID, Cell ID 정보**

② CSV 파일 저장 설정 파일 위치

a. **API 파일이 있는 폴더의 SubFold("DllIni")에 환경 설정 파일 존재**

b. **API←PLC : "ReceiveFromPLC_CSV.ini"**

c. **Vision PC←API : "GetFromAPI_CSV.Inl"**



③ 설정 파일을 Open(window notepad에서 편집 가능)

a. **Base Fold(사용자 편집 가능)➔저장 위치 편집하지 마세요**

- ✓ 항목이 파일 저장 위치(사용자가 편집가능합니다.)
- ✓ CSV 파일이 저장되는 Root Fold 입니다.
- ✓ **단, 마지막 문자는 반드시 '₩'표기되어 있어야 합니다.**

b. **Factory Line Datatype Information(사용자 편집 가능)**

- ✓ CSV 파일 이름입니다.(예: "OCPilot5-PKG_Notching", 오창 Pilot5-PKG 라인의)
- ✓ 날짜, 시간, 확장자는 자동 추가됩니다.(예: "OCPilot5-PKG_Notching_20210824_11.csv")

c. **Keep Day(사용자 편집 가능)**

- ✓ CSV 파일 보존 기간입니다. 보존 기간이 넘어서면 자동으로 제거합니다.

d. **Title(사용자 편집 하지 마세요)**

- ✓ CSV 파일 저장 항목입니다.
- ✓ 항목의 개수는 구분자(',')로 Counting합니다.
- ✓ **저장 항목의 개수를 변경할 경우 프로그램 수정이 필요합니다.**
- ✓ '//'은 문자열의 첫번째에 있어야 합니다

13) API에서 "DllSocket_AddToECS" 사용시 주의점(모든 항목은 문자열)

① Cell ID

a. **1개만 입력합니다.**

② CSV파일이 저장되는 전체 경로와 파일 이름

a. **파일 경로는 전체 경로**

b. **파일 이름은 확장자 포함 경로**

c. **예: "C:₩Data₩20211018₩OC_Vision_20211018_12.csv"**

③ ECS 보고용 데이터는 항목(Title)과 값(Value)의 짝(Pair)을 맞추어 대입합니다.

a. **Title 항목에서 "Date"와 "Time" 항목은 필수입니다.**

- ✓ Date : YYYYMMDD 형식 (예 20211018), 총8자리
- ✓ Time : HHmmSS 형식 (예 131011), 24시간 기준 총 6자리

b. **Title과 Value는 "=" 문자로 구분하여 짝을 맞춥니다.**

c. **Title과 Value로 구성된 단위는 ","로 구분합니다.**

④ 예: "**Date**=20211018,**Time**=131011,Cell ID=C123456789,Judeg=OK"

⑤ 항목(Title)과 값(Value)은 CSV 파일과 일치해야 합니다.

⑥ API⇔Agent+간 통신에서 '|'를 구분자로 사용합니다. Cell ID|파일 경로와 파일 이름|보고 문자열

14) DllSocket_GetReceiveMessageInQueue 함수 설명

```
/// <param name="bLastReceiveReturn">가장 최근 수신 Message Return이면 true, 가장 과거 수신 Message Return 이면 false</param>
/// <param name="bDeleteRemainQueueBuffer">함수 호출과 동시에 남겨진 message가 있다면 모두 제거 조건이면 true</param>
/// <returns>수신받은 메세지를 문자열 형태로 반환</returns>
public string DllSocket_GetReceiveMessageInQueue(bool bLastReceiveReturn, bool bDeleteRemainQueueBuffer)
```

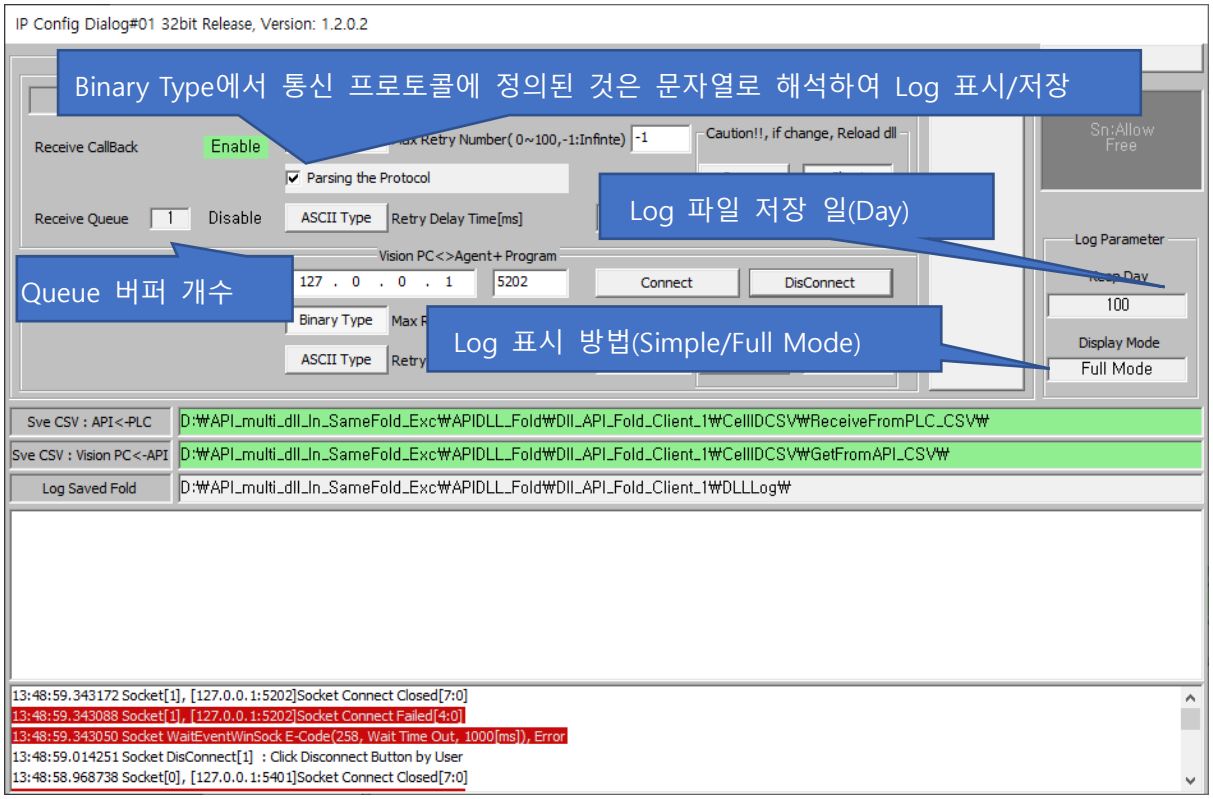
① 반환 문자열 형식

a. **Title1=Value1,Title2=Value2,...**

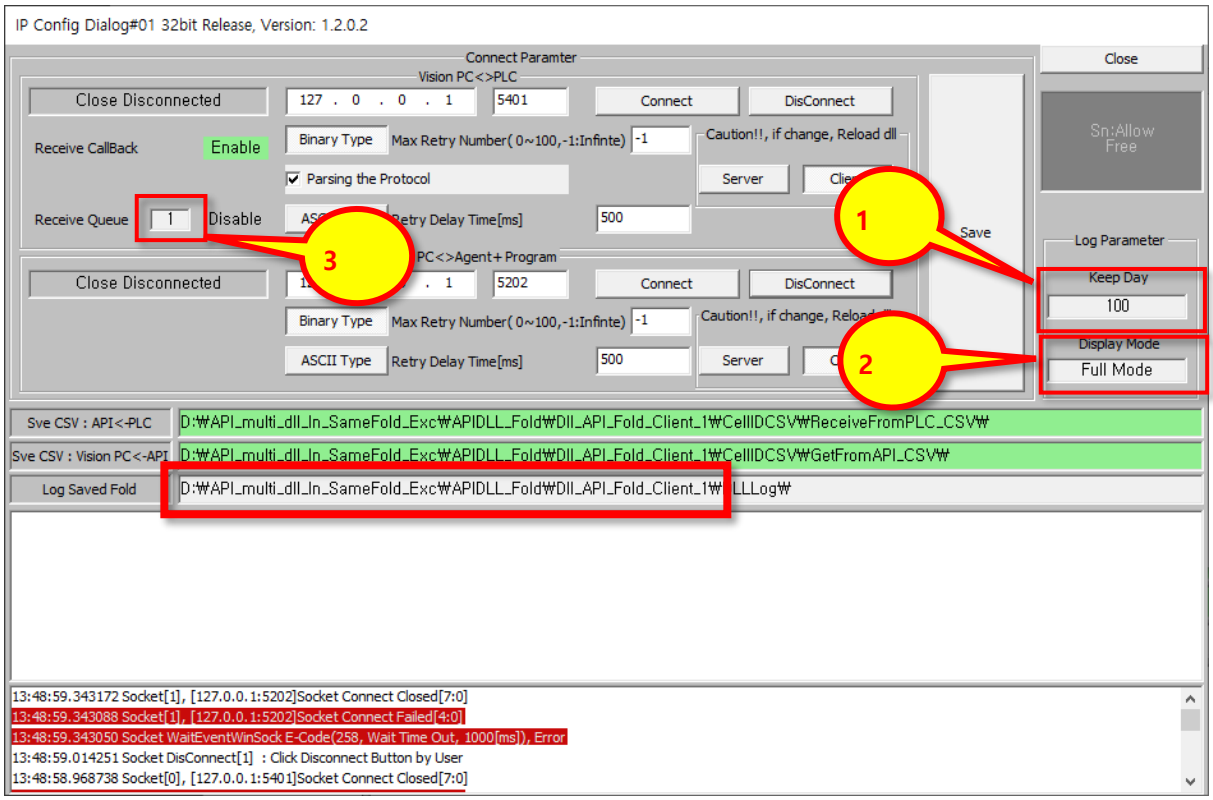
b. **Title 리스트(통신 프로토콜 2.1이상버전, 엑셀파일 참조)**

- ✓ Count : 현재 Queue 버퍼의 개
- ✓ CellNumber : Cell ID Block의 C1
- ✓ LotID : Cell ID Block의 C3
- ✓ CellID_1 : Cell ID Block의 C4
- ✓ CellID_2 : Cell ID Block의 C5
- ✓ BitModeValue : Cell ID Block의 C2

15) API Widow 창 표시 설명



16) API 설정



① Log 파일 보존 기간("Keep Day", 위 그림에서 ① 표시)

a. **파일 Open : API 폴더(위 그림 Red Box)에서 "Dllini" 폴더에 있는 "IPConfig.ini"**

✓ 예:"D:₩API_multi_dll_In_SameFold_Exc₩APIDLL_Fold₩Dll_API_Fold_Client_1₩DLLIni₩"

b. **Window program(Notepad.exe)으로 "IPConfig.ini" 파일을 편집하기로 Open합니다.**

✓ "Keep Day = 100"항목에서 100의 값을 변경(1~100)하면 됩니다.(단위는 Day)

② Log 표시 및 저장 옵션("Display Mode", 위 그림에서 ② 표시)

a. **파일 Open : API 폴더(위 그림 Red Box)에서 "Dllini" 폴더에 있는 "IPConfig.ini"**

✓ 예:"D:₩API_multi_dll_In_SameFold_Exc₩APIDLL_Fold₩Dll_API_Fold_Client_1₩DLLIni₩"

b. **Window program(Notepad.exe)으로 "IPConfig.ini" 파일을 편집하기로 Open합니다.**

✓ "Simple Log = false"항목에서 false값을 "true"/"false"의 값으로 변경하면 됩니다.

③ Queue 버퍼 개수 설정(위 그림에서 ③ 표시)

a. **파일 Open : API 폴더(위 그림 Red Box)에서 "Dllini" 폴더에 있는 "IPConfig.ini"**

✓ 예:"D:₩API_multi_dll_In_SameFold_Exc₩APIDLL_Fold₩Dll_API_Fold_Client_1₩DLLIni₩"

b. **Window program(Notepad.exe)으로 "IPConfig.ini" 파일을 편집하기로 Open합니다.**

✓ "Queue Limit Number = 1"항목에서 1의 값을 (1~5)의 값으로 변경하면 됩니다.

17) API⇔PLC 통신 연결 점검(Heartbeat)

PLC(Server)에서 API(Client) 간 통신 상태(연결/수신/송신 정상) 확인을 하기 위한 송수신

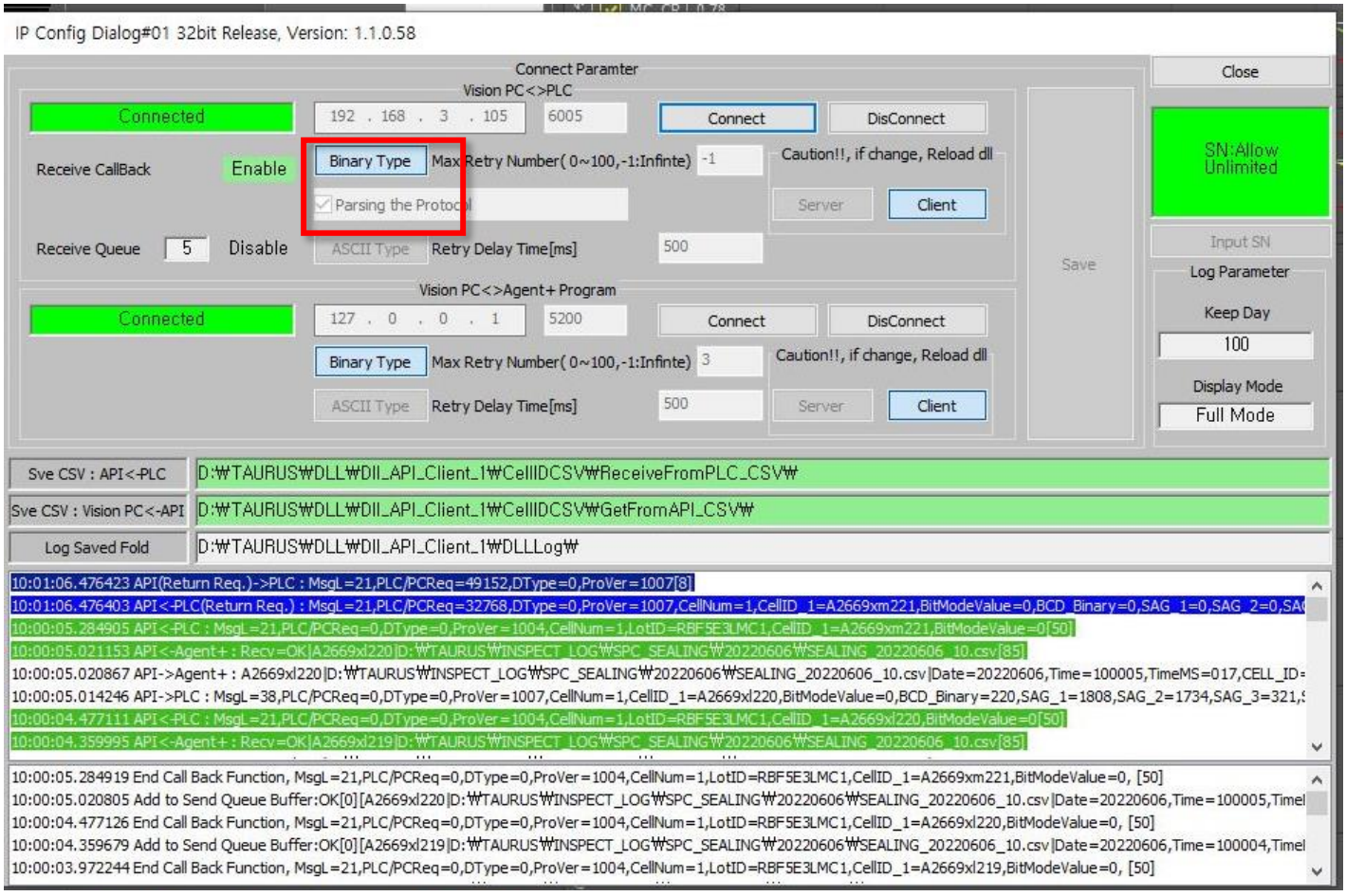
① API 설정-API⇔PLC 통신 설정에만 적용

a. **API Binary 통신**

b. **Parsing the protocol**

c. **자동 재접속 회수 설정 확인**

✓ -1 : 무한반복, 0 : 접속하지 않음, 1~100 (해당 회수 이내에 접속 시도)



18) 문제 발생 및 조치

① Dll API 초기화 오류 메시지 창 발생

a. **관리자 권한으로 프로그램 개발/디버깅을 실행 합니다.**

② 프로그램 예제 프로그램 실행 시 Callback(대리자) 오류발생(실제 발생 사례가 있습니다.)

a. **파일 경로를 250자 이하로 줄이세요**

③ "DllSocket_Initialize" 함수를 Thread 처리 함수에서 호출한 경우 IP Config 창 show/hide 작동이에 문제가 발생하였습니다(C#에서 발생).

④ "DllSocket_IsConnected" 함수는 Connect callback 함수내에서 호출하지 마세요

⑤ 콜백 함수 내에서 window message 발생 함수(예: SendMessage 함수)를 사용하지 마세요.Deal Lock이 발생합니다.

⑥ 라이선스 키 입력 후 "Fail" 발생

a. **API 실행 파일을 관리자 권한으로 실행하세요**

b. **Window Defence 보안 설정은 모두 해제하세요**

19) API 오류 표시 설명

① "Error[Delete Queue(셀 id), Count(최대 개수)] in SubOverQueueCountToDelete_Protocol"

A. Queue 버퍼 방식일 때 API에서 수신 받은 메시지가 설정값보다 많이 저장되면 가장 과거에 받은 것을 지우고, 지운 내용을 표시한 로그입니다.

B. 셀 ID는 지운 셀 ID 문자열입니다.

C. 최대 개수는 사용자가 설정한 Queue 버퍼 개수보다 많은 수입니다.

- ✓ 예: Queue 사용자 설정값이 5이면, 최대 개수 6
- ✓ 예: Queue 사용자 설정값이 1이면, 최대 개수 2

D. 비전←API로 데이터를 가져가지 않으면 발생합니다.