# PeerSDK Programming Guide

## System Prerequisite

1. Windows XP/Vista/7 32-bit
2. iOS 4.3 or above
3. Linux 64-bit

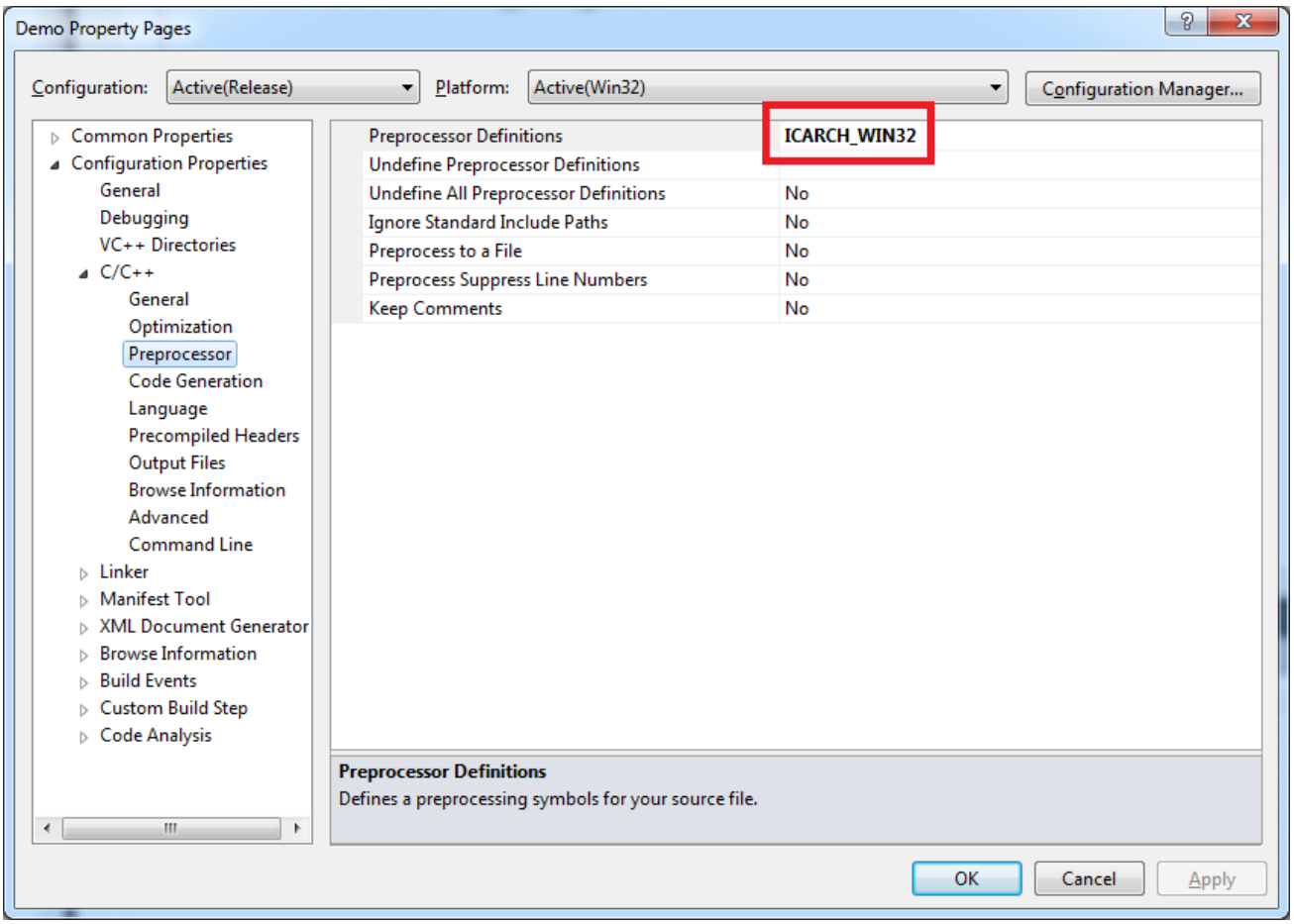
## System Preparation

### 1. Define ICARCH_WIN32 in source or IDE

Example: (see *Code/ConnectInformationEvents.cpp*)

```
#define ICARCH_WIN32     // for Windows
#define ICARCH_IOS_ARM7  // for iPhone/iPad/iOS devices
#define ICARCH_LINUX64   // for Linux x64
```

Or in Visual Studio 2010




### 2. Include <PeerSDK.h>

Example: (see *Code/ConnectInformationEvents.cpp*)

```
#include <PeerSDK.h>
using namespace PeerSDK;   // Optional, remove this line if namespace conflict
```

### 3. Initialize PeerSDK

Example: (see *Code/ConnectInformationEvents.cpp*)

```
PeerResult r;
r = Peer::Startup();
if (!r)
    printf("*** failed to initialize\n");
```

# Topic

## 1. Connect, Information, and Events

*Create a new Peer instance*

```
m_peer = new Peer();
```

*Actively check channel video status*

```
m_peer-> Channels()[ch]->Video()->IsLoss()  // get videoloss status
m_peer-> Channels()[ch]->Video()->IsMotion();  // get video motion status
```

*Attach event sink point*

```
m_peer->ErrorOccurred().Add(NULL, OnErrorOccurred);
m_peer->VideoLossChanged().Add(NULL, OnVideoLossChanged);
m_peer->VideoMotionChanged().Add(NULL, OnVideoMotionChanged);
```

*Connect to DVR*

```
r = m_peer->Connect("192.168.3.38", 80, "Admin", "123456", "", true);
if (!r)
{
    printf("*** failed to connect\n");
    return Finalize(-1);
}
```

Peer class contains a property which name is **Type**. Application could store this value, and use it to connect at next time.

Example: try ***peertype*** first, if can't connect, try other types.

```
r = m_peer->Connect("192.168.3.38", 80, "Admin", "123456", peertype, true);
```

Example: try ***peertype*** first, if can't connect, return failure.

```
r = m_peer->Connect("192.168.3.38", 80, "Admin", "123456", peertype, false);
```

## 2. Live View

*Connect to DVR*

*See Topic 1: Connect, Information, and Events*

*Create a live stream*

```
r = m_peer->CreateLiveStream(&m_stream);
```

*Attach event sink point*

```
m_stream->ErrorOccurred().Add(NULL, OnErrorOccurred);
m_stream->VideoArrived().Add(NULL, OnVideoArrived);
m_stream->AudioArrived().Add(NULL, OnAudioArrived);
```

*Set active stream and channel mask*

```
m_stream->SetActive(0);
m_stream->SetChannelMask(0xFFFF, 0xFFFF);
```

*Start to receive stream data*

```
m_stream->Start();
```

## 3. Get Record List

*Create new record list*

```
PeerRecordList* recordList = NULL;
r = m_peer->CreateRecordList(&recordList);
if (!r)
    printf("*** failed to get record list\n");
```

*Get record segments*

```
TimeSpanList tsList;
r = recordList->GetRecordedMinutesOfDay(year, month, day, tsList);
if (!r)
    printf("*** failed to get record\n");
```

***PeerRecordList*** has three methods (***GetRecordedDaysOfMonth, GetRecordedMinutesOfDay, GetList***) to get record segments. Application can use them in different situation.

## 4. Playback

*Connect to DVR*

*See Topic 1: Connect, Information, and Events*

*Create a record stream for playback*

```
r = m_peer->CreateRecordedStream(DateTime(year, month, day, tsList[0].Hours(), tsList[0].Minutes(), 0, m_peer), &m_stream);
```

*Attach event sink point*

```
m_stream->ErrorOccurred().Add(NULL, OnErrorOccurred);
m_stream->VideoArrived().Add(NULL, OnVideoArrived);
m_stream->AudioArrived().Add(NULL, OnAudioArrived);
```

*Set active stream and channel mask*

```
m_stream->SetActive(0);
m_stream->SetChannelMask(0xFFFF, 0xFFFF);
```

DVR could support multiple stream (main stream and network stream, etc…). Application can switch by calling SetActive(). In general, bigger value has better quality and higher resolution (by DVR configuration).

*Start to receive stream data*

```
m_stream->Start();
```

## 5. Backup

*Connect to DVR*

*See Topic 1: Connect, Information, and Events*

*Create a record stream for backup*

```
r = m_peer->CreateRecordedStream(
    DateTime(year, month, day, tsList[0].Hours(), tsList[0].Minutes(), 0, m_peer),
    DateTime(year, month, day, tsList[0].Hours(), tsList[0].Minutes() + 1, 0, m_peer),
    &m_stream);
```

*Attach event sink point*

```
m_stream->ErrorOccurred().Add(NULL, OnErrorOccurred);
m_stream->VideoArrived().Add(NULL, OnVideoArrived);
m_stream->AudioArrived().Add(NULL, OnAudioArrived);
m_stream->Completed().Add(NULL, OnCompleted);
```

Backup has an extra event: ***Completed***. When stream reaches end of stream, the event is raised. Application should finalize the backup job at this time.

*Set active stream and channel mask*

```
m_stream->SetActive(0);
m_stream->SetChannelMask(0xFFFF, 0xFFFF);
```

*Start to receive stream data*

```
m_stream->Start();
```