#7041 [TP-J560XDn]XJMF communication between controller and UW

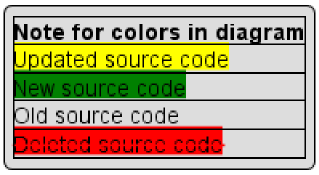
Change log

Rev.	Date	Author	Details
1	2021/12/20	GCS	Created
2	2021/12/30	GCS	Updated • Update solution for spec 205, 207 • Add solution for spec 208 • Update diagram 202.1, 202.2, 202.3, 205.10, 207.3 • Add diagrams 205.9, 205.13, 205.14, 205.15, 205.17, 205.18, 205.19, 207.1, 207.2, 208.1, 208.2

Target System

[TP-J560XDn]Ver1.00_base_3.50

Note for diagrams



200. JetDrive

The behaviors of 201 to 208 below work only when the key below is 1.
[File name] PrinterDescriptor.ini
[Section name] OPTION
[Key name] UW_CONNECT_FUNCTION
The default value of the above key is 0.

201. Check the startup of the HTTP communication service program.

1. Description

If UWandRW_Receiver.exe is not started when the controller is started, the following warning message dialog is displayed.
(Ja)前後装置の通信サービスプログラムが起動していません。
(En) The communication service program of the front and rear devices has not started.

2. Solution

- Add resource into strings_UnwinderManager.ini file to display UWandRW_Receiver.exe is not started.

Resource\English\strings_UnwinderManager.ini

```
[MSG]
IDM_NOTIFY_RECEIVER_STATUS = The communication service program of the front and rear devices
has not started.
```

Resource\Japanese\strings_UnwinderManager.ini

```
[MSG]
IDM_NOTIFY_RECEIVER_STATUS = 前後装置の通信サービスプログラムが起動していません。
```

- Add class CDataIF inherits from CMakeComposeUnwinderData.
- In function CDataIF::PIM_InitSystem(), check UW_CONNECT_FUNCTION if 1 then create a thread to run plugin main process.
- In function CDataIF::PIM_ExitSystem(), signal thread to exit.

UnwinderManager\Data_IF.h

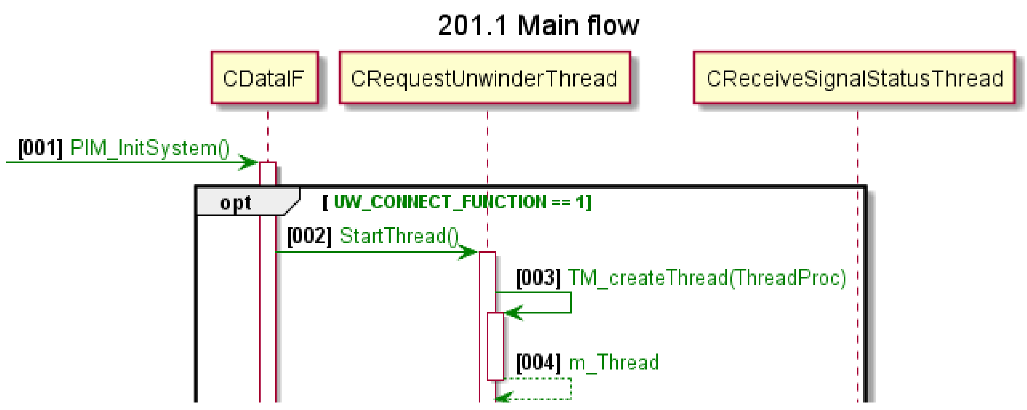
```
class CDataIF : public CBaseDataIF,
                public CMakeComposeUnwinderData
{
public:
    void Initialize();
    void Finalize();
    virtual BOOL PIM_InitSystem();
    virtual BOOL PIM_ExitSystem();
    ...
}
```

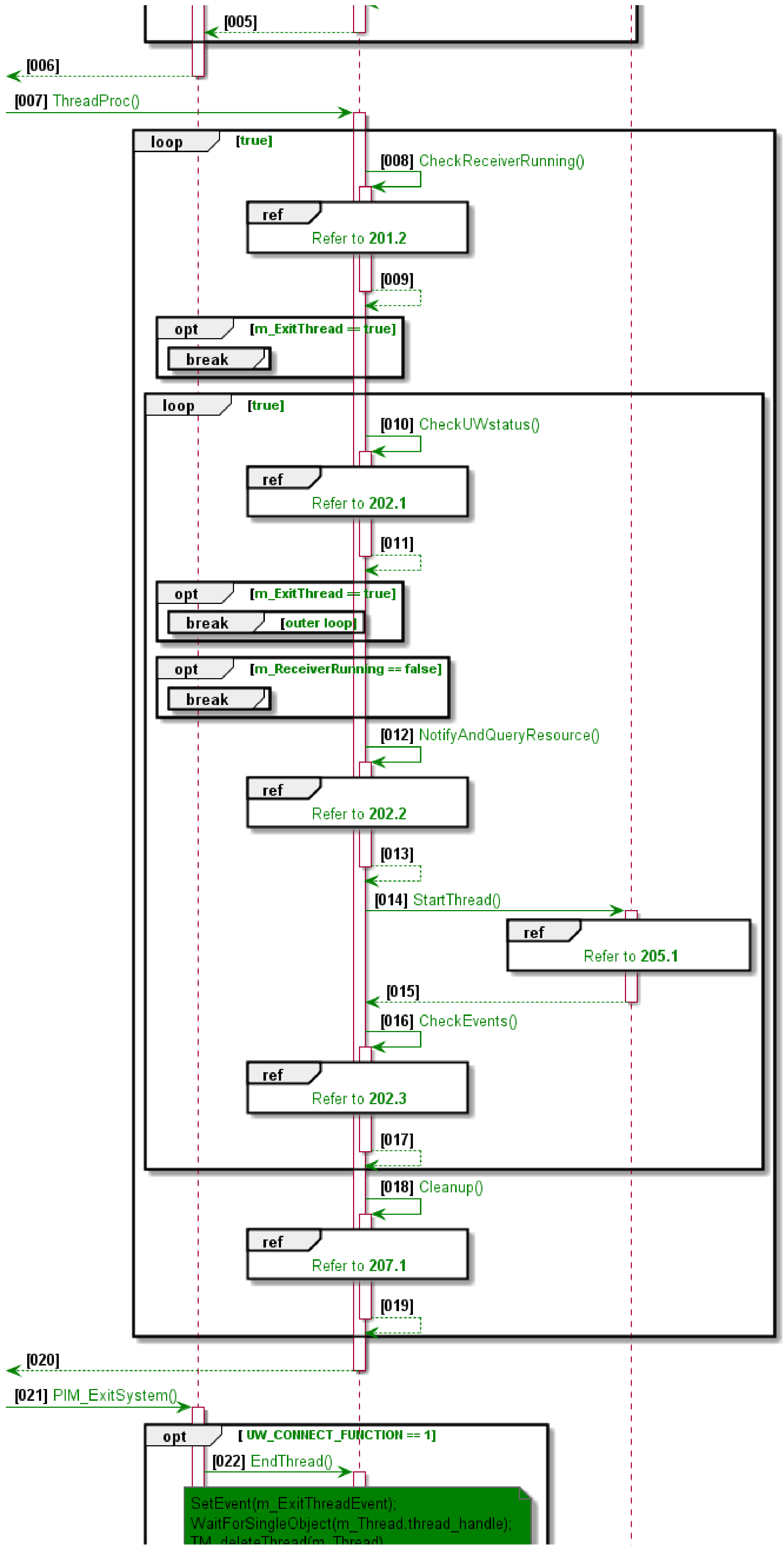
- Add class CRequestUnwinderThread() to run main process.
- At first, check for process “UWandRW_Receiver.exe” running, if not then display warning dialog.
- Add loop to wait until “UWandRW_Receiver.exe” running to continue process.

UnwinderManager\RequestUnwinderThread.h

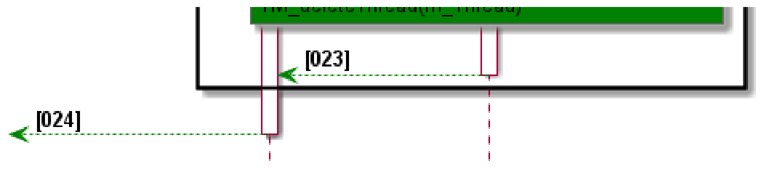
```
class CRequestUnwinderThread
{
public:
    CRequestUnwinderThread();
    virtual ~CRequestUnwinderThread();
    void Initialize(CMakeComposeUnwinderData* unwinderData);
    void Finalize();
    void StartThread();
    void EndThread();
    ...
private:
    void ThreadProc();
    void CheckReceiverRunning();
    CRequestUnwinder m_RequestUnwinder;
    ST_THREAD_INFO m_Thread;
    HANDLE m_ExitThreadEvent;
    HANDLE m_ReceiverProcess;
    bool m_ExitThread;
    bool m_ReceiverRunning;
    ...
}
```

3. Detail implementation

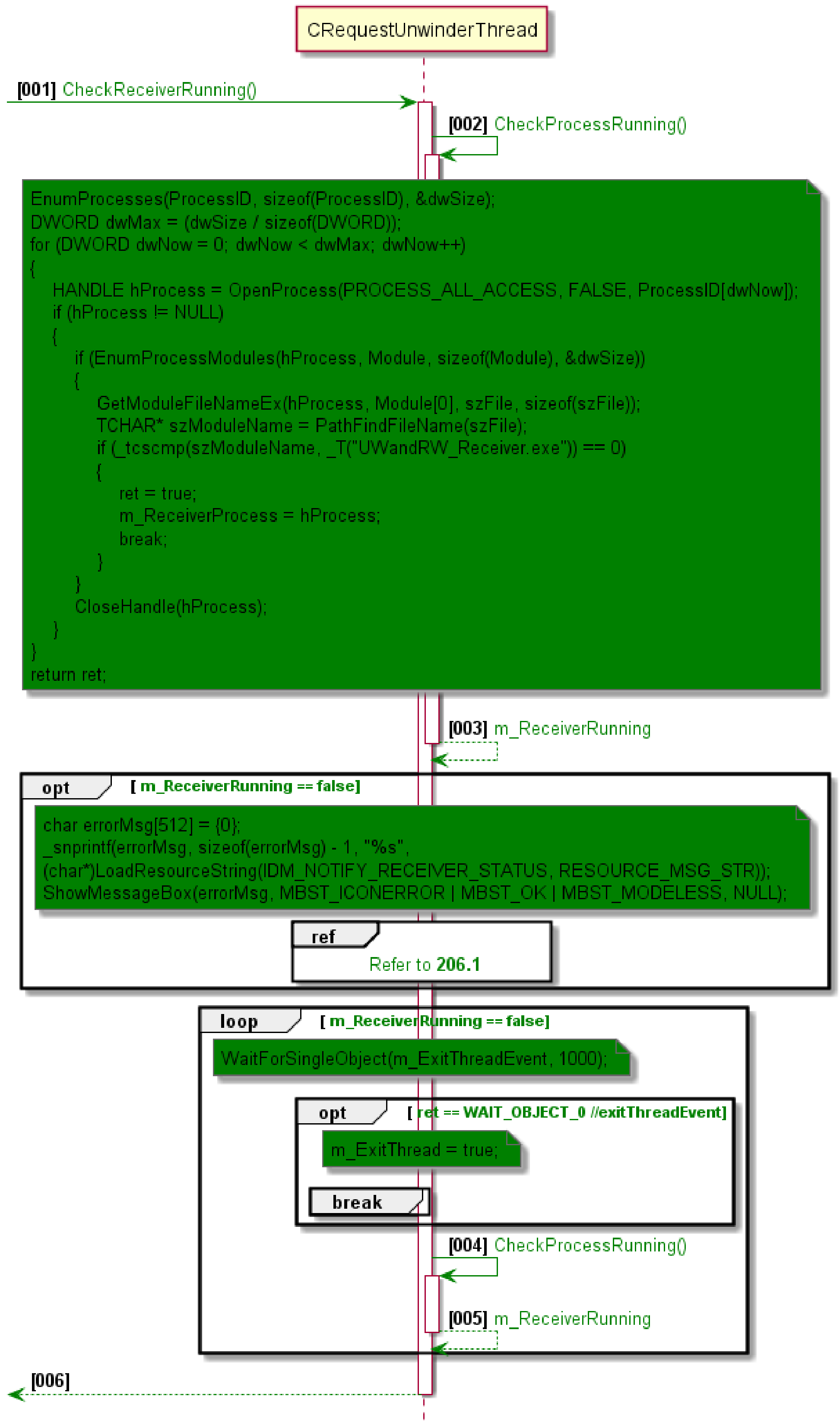




```
SetEvent(m_ExitThreadEvent);
WaitForSingleObject(m_Thread.thread_handle);
TM_deleteThread(m_Thread)
```



201.2 Check Receiver is running


CRequestUnwinderThread
[001] CheckReceiverRunning()
[002] CheckProcessRunning()
[003] m_ReceiverRunning
opt
[m_ReceiverRunning == false]
ref
Refer to 206.1
loop
[m_ReceiverRunning == false]
opt
[ret == WAIT_OBJECT_0 //exitThreadEvent]
break
[004] CheckProcessRunning()
[005] m_ReceiverRunning
[006]

```
EnumProcesses(ProcessID, sizeof(ProcessID), &dwSize);
DWORD dwMax = (dwSize / sizeof(DWORD));
for (DWORD dwNow = 0; dwNow < dwMax; dwNow++)
{
    HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, ProcessID[dwNow]);
    if (hProcess != NULL)
    {
        if (EnumProcessModules(hProcess, Module, sizeof(Module), &dwSize))
        {
            GetModuleFileNameEx(hProcess, Module[0], szFile, sizeof(szFile));
            TCHAR* szModuleName = PathFindFileName(szFile);
            if (_tcscmp(szModuleName, _T("UWandRW_Receiver.exe")) == 0)
            {
                ret = true;
                m_ReceiverProcess = hProcess;
                break;
            }
        }
        CloseHandle(hProcess);
    }
}
return ret;
```

```
char errorMsg[512] = {0};
_snprintf(errorMsg, sizeof(errorMsg) - 1, "%s",
(char*)LoadResourceString(IDM_NOTIFY_RECEIVER_STATUS, RESOURCE_MSG_STR));
ShowMessageBox(errorMsg, MBST_ICONERROR | MBST_OK | MBST_MODELESS, NULL);
```

```
WaitForSingleObject(m_ExitThreadEvent, 1000);
```

```
m_ExitThread = true;
```

202. Communication channel registration for UW

1. Description

The controller registers the communication channel in order to acquire information from the UW.
Specify the URL when registering the channel, and notify the information to that URL.
Save the response channel ID in the TP-UW_Communication.ini file.
The following two communication channels are used, and channel registration assumes that UW is running.
A. Condition monitoring channel（Channel for the controller to get the status of UW from UW）
The channel registration timing is when the controller is started.
B. Paper information notification channel（Channel for the controller to obtain the remaining amount of paper, roll diameter, and paper thickness from UW）
The channel registration timing is set immediately after the print condition information is set in the UW from the controller and the setting result response is received from the UW.
The timing of setting the print conditions will be described in 204 below.

The registered notification channel is reflected / updated in various keys of TP-UW_Communication_work.ini.
※Please refer to【TP-UW_Communication_work.ini】(See below)

2. Solution

- Condition monitoring channel:
  - In function CRequestUnwinderThread::ThreadProc(), call to RequestQueryStatus().
  - If not success, update UW status (Refer to 206) and continue to call to RequestQueryStatus() until success.
  - If success, start the thread to receive signal status from UW. (refer to diagram 205.1)
  - When there is no signal status from UW (post messsage WM_USER_UW_NO_STATUS from timer), end the thread and repeat the process. (refer to 208 section)
- Paper information notification channel:
  - Add function CRequestUnwinderThread::NotifyAndQueryResource() and call in CRequestUnwinder::ThreadProc().
  - In CRequestUnwinderThread::NotifyAndQueryResource(), call to RequestQueryResource() after RequestCommandResource() (Refer to 203).
  - When there is an event (post message WM_USER_NOTIFY_QUERY_RESOURCE from other plugins), register again.

UnwinderManager\RequestUnwinderThread.h

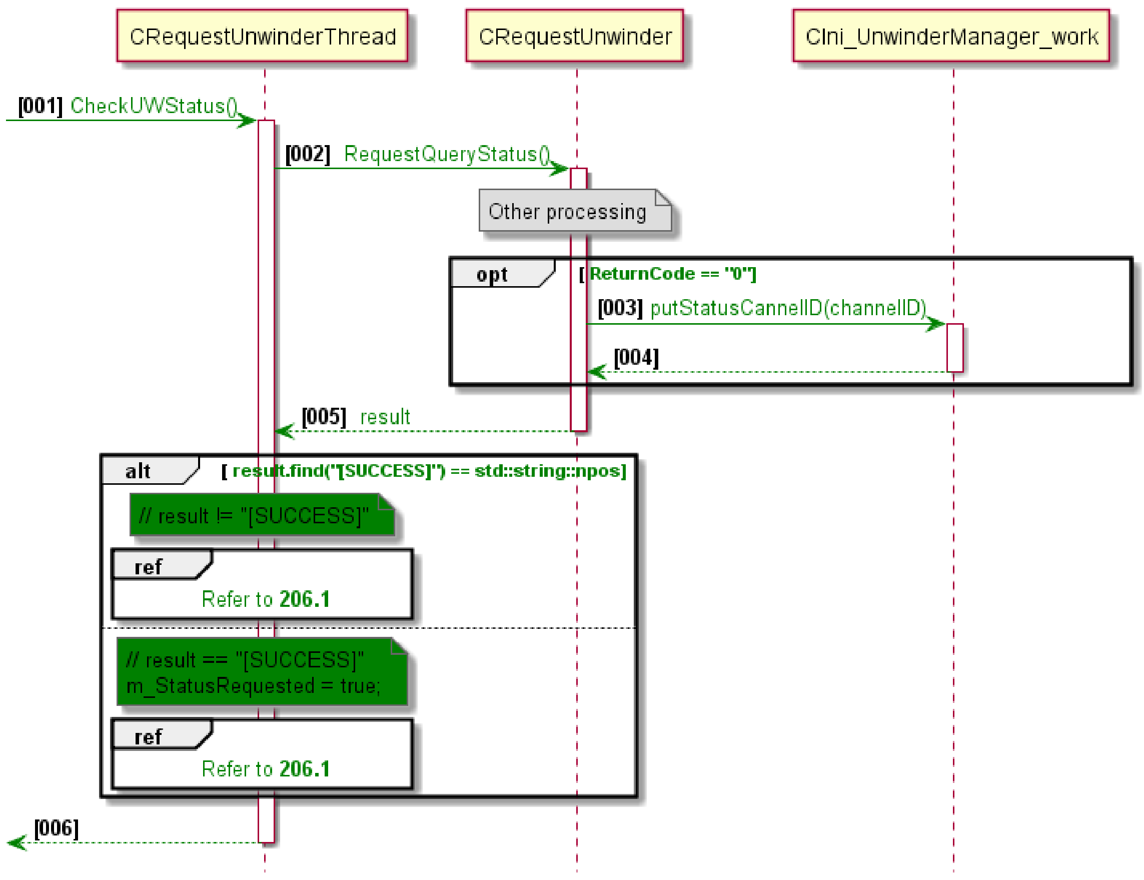
```
class CRequestUnwinderThread
{
private:
...
void CheckUWStatus();
void NotifyAndQueryResource(const std::string& inSectionId = "");
void CheckEvents();
...
bool m_StatusRequested;
bool m_ResouceRequested;
...
}
```

UnwinderManager\RequestUnwinderThread.cpp

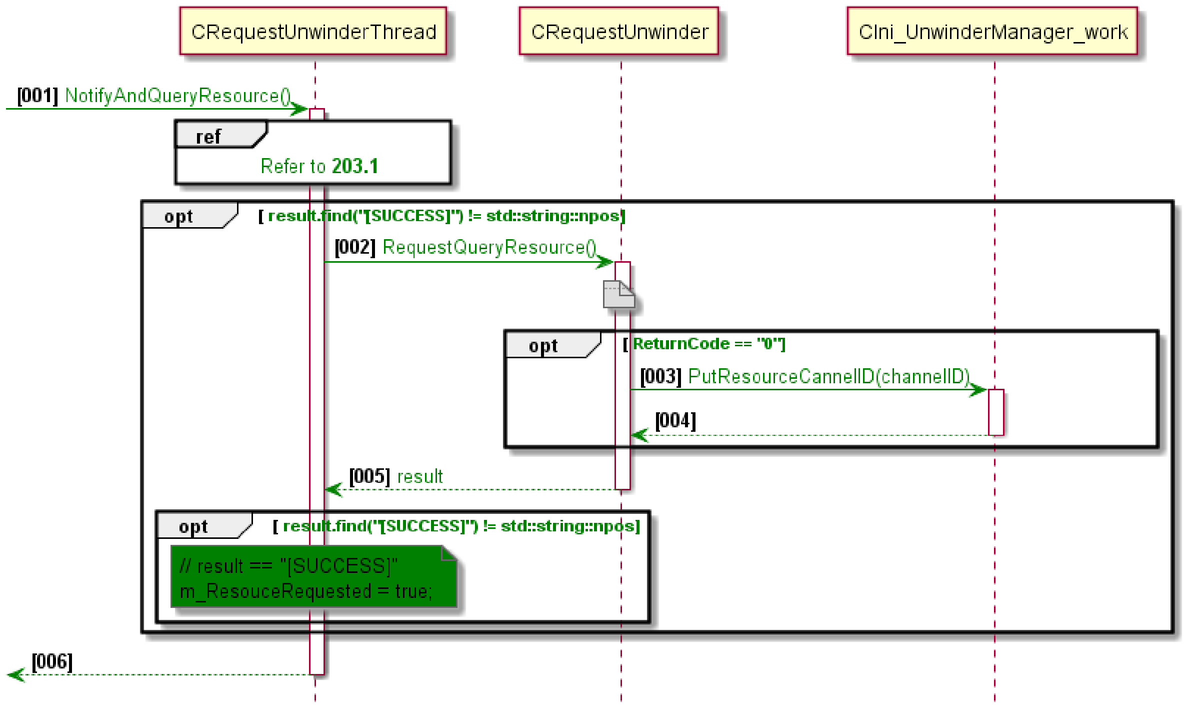
```
#define WM_USER_UW_NO_STATUS                    WM_USER+100
#define WM_USER_NOTIFY_QUERY_RESOURCE    WM_USER+101
```

3. Detail implementation

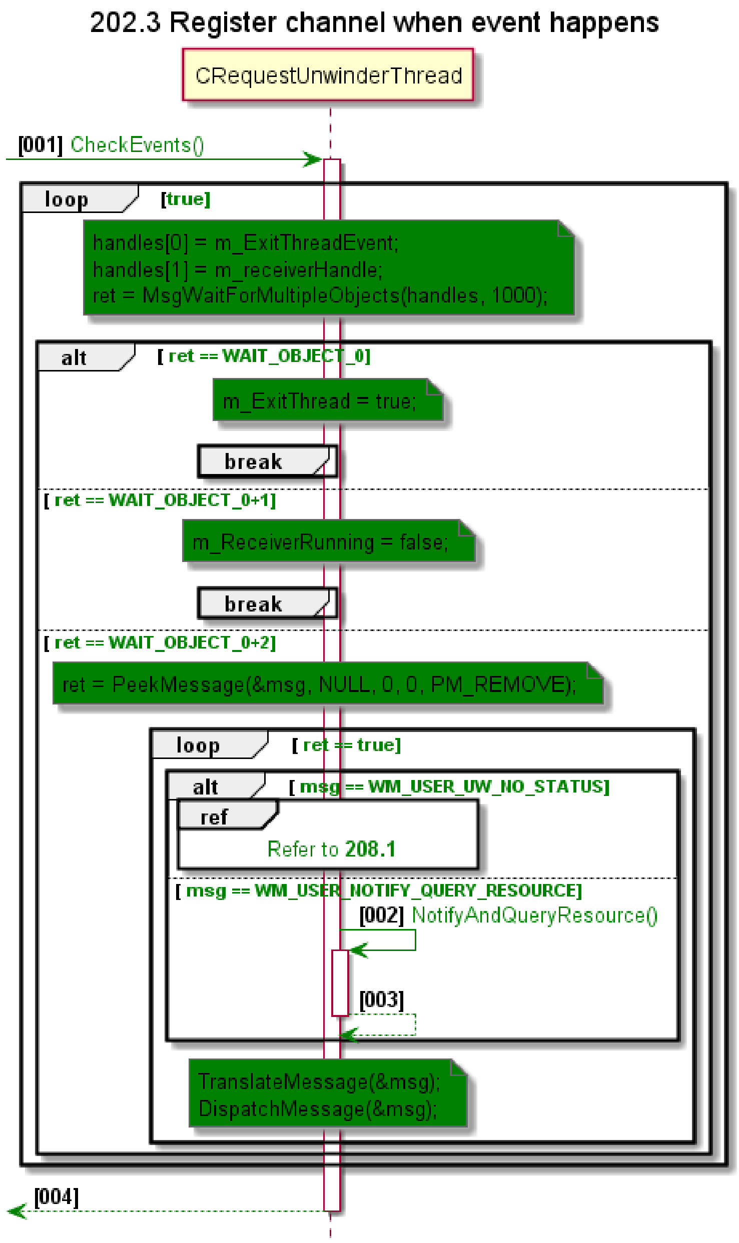
202.1 Register condition monitoring channel



202.2 Register paper information notification channel



CRequestUnwinderThread
CRequestUnwinder
CIni_UnwinderManager_work
[001] NotifyAndQueryResource()
ref
Refer to 203.1
opt
[result.find("[SUCCESS]") != std::string::npos]
[002] RequestQueryResource()
opt
[ReturnCode == "0"]
[003] PutResourceCannelID(channelID)
[004]
[005] result
opt
[result.find("[SUCCESS]") != std::string::npos]
// result == "[SUCCESS]"
m_ResouceRequested = true;
[006]


202.3 Register channel when event happens
CRequestUnwinderThread
[001] CheckEvents()
loop [true]
handles[0] = m_ExitThreadEvent;
handles[1] = m_receiverHandle;
ret = MsgWaitForMultipleObjects(handles, 1000);
alt [ret == WAIT_OBJECT_0]
m_ExitThread = true;
break
[ret == WAIT_OBJECT_0+1]
m_ReceiverRunning = false;
break
[ret == WAIT_OBJECT_0+2]
ret = PeekMessage(&msg, NULL, 0, 0, PM_REMOVE);
loop [ret == true]
alt [msg == WM_USER_UW_NO_STATUS]
ref
Refer to 208.1
[msg == WM_USER_NOTIFY_QUERY_RESOURCE]
[002] NotifyAndQueryResource()
[003]
TranslateMessage(&msg);
DispatchMessage(&msg);
[004]

203. Notify UW of print condition information.

1. Description

The print conditions to be notified are as bellow.
・Print condition name(DescriptiveName)
・Media name(MediaType)
・Paper width(Dimension X point)
・Paper remaining amount(Dimension Y point)
・Paper thickness(Thickness)
・Tension(scr:UWTension)
・Print speed(scr:MaxRunSpeed)
※For tension and speed, use the calculation result using the formula described at the time of additional update. Once, the value as it is notified.

Notify UW of printing conditions at the following timing.
1.When controller is started.(Current print condition)
2. When switching the current print condition(Current print condition)
3. When changing the current print condition setting(Current print condition)
4. When the job is running (Print conditions during job execution)

Regarding 4, in the case of continuous job printing, the content of the print conditions of the first job is notified.

2. Solution

- In CRequestUnwinderThread::NotifyAndQueryResource():
  - Depend on sectionId empty (current print condition) or not empty (job print condition), get the neccessary information.
  - call to RequestCommandResource().

- Case 1 When controller is started:
  Call to NotifyAndQueryResource() in CRequestUnwinderThread::ThreadProc(). (Refer to 202)

- Add plugin callbacks functions.

  UnwinderManager\Plugin_IF.h

  ```
  PLUGIN_MODULE_API bool _UnwinderManager_GetCallbacks(struct SUnwinderManager_Callbacks*
  outCallbacks);
  ```

  Common\UnwinderManager_Callbacks.h

  ```
  typedef void (*_OnFirstJobRun)(const std::string& inSectionId);
  typedef void (*_OnSetCurrentPrintCondition)();
  typedef void (*_OnUpdateCurrentPrintCondition)();
  struct SUnwinderManager_Callbacks
  {
      //Version 1
      DWORD                           StructVersion;
      _OnSetCurrentPrintCondition     OnSetCurrentPrintCondition;
      _OnUpdateCurrentPrintCondition  OnUpdateCurrentPrintCondition;
      _OnFirstJobRun                  OnFirstJobRun;
  }
  ```

  Common\UnwinderManager_Callbacks.h

  ```
      class CUnwinderManager_OP : public CUnwinderManager
      {
      public:
          ...
          void OnSetCurrentPrintCondition();
          void OnUpdateCurrentPrintCondition();
          void OnFirstJobRun(const std::string& inSectionId);
          ...
      }
  ```

  UnwinderManager\Data_IF.h

  ```
  class CDataIF
  {
  public:
      ...
      void OnSetCurrentPrintCondition();
      void OnUpdateCurrentPrintCondition();
      void OnFirstJobRun(const std::string& inSectionId);
      ...
  }
  ```

- Add function CRequestUnwinderThread::MsgNotifyAndQueryResource() to notify main thread of the event.

  UnwinderManager\RequestUnwinderThread.h

  ```
  class CRequestUnwinderThread
  {
  public:
      ...
      void MsgNotifyAndQueryResource(const std::string& inSectionId = "");
      ...
  }
  ```

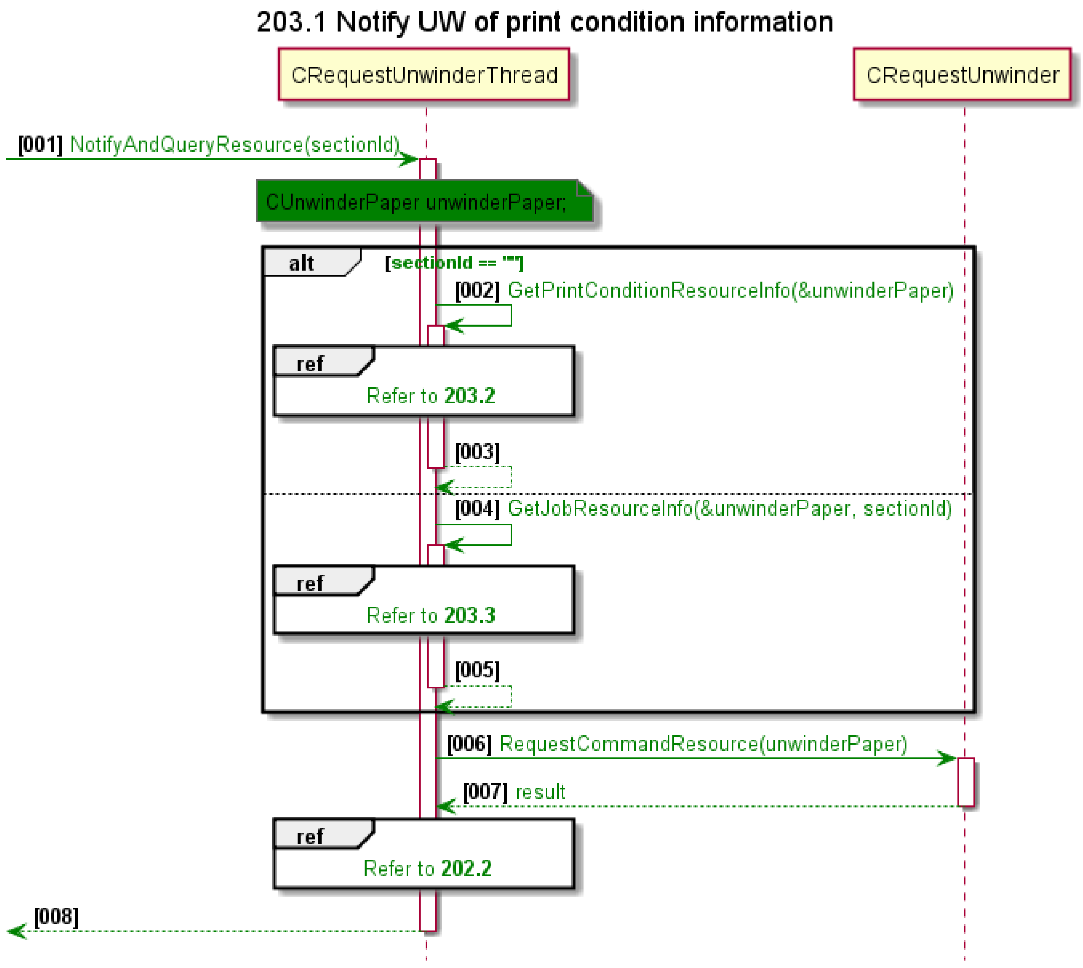
- In each callback function, call to CRequestUnwinderThread::MsgNotifyAndQueryResource().

- Case 2 When switching the current print condition:
  In plugin PrintConditionGUI: call to
  SUnwinderManager_Callbacks::OnSetCurrentPrintCondition() in

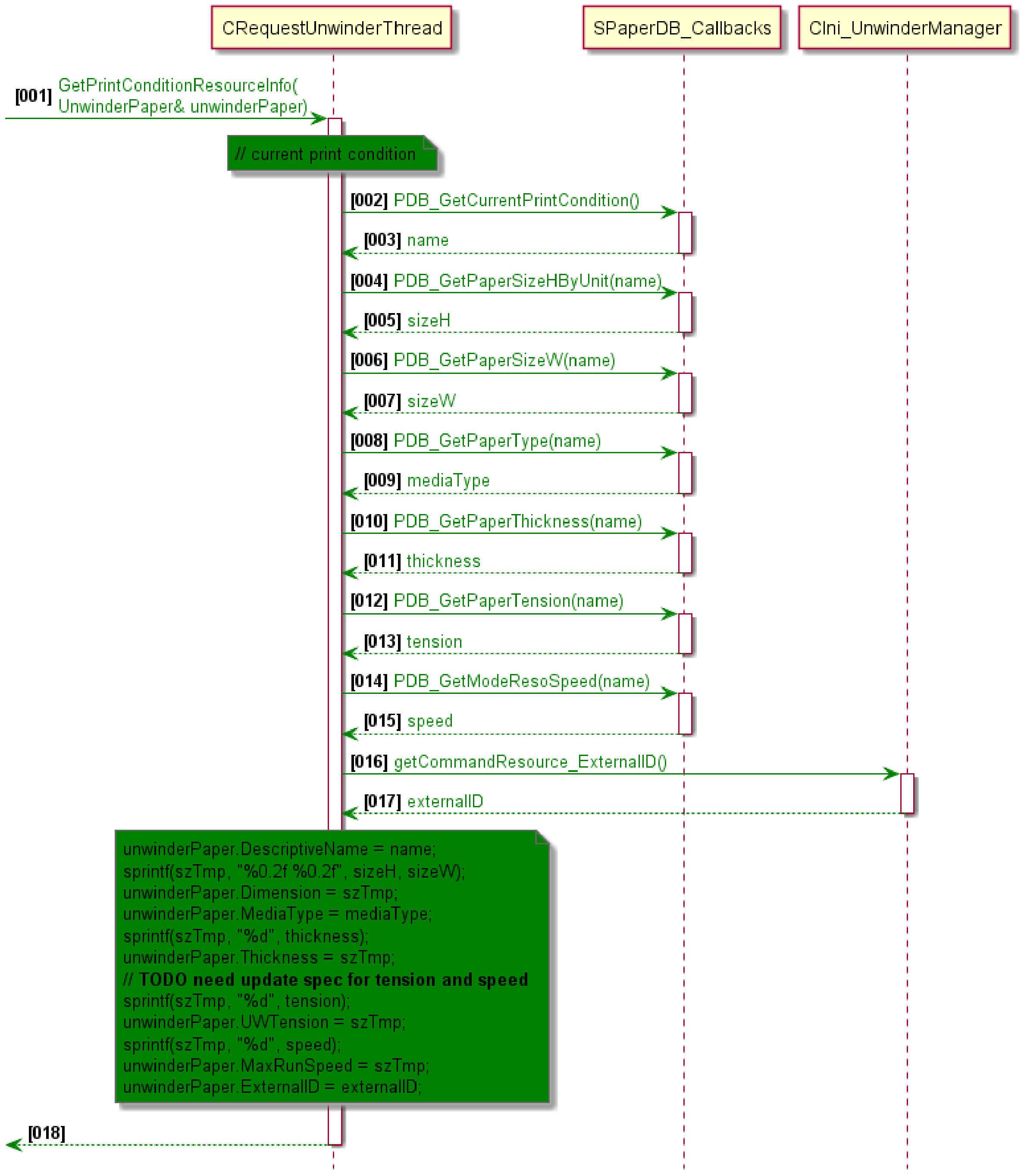
CDataIF::SetCurrentPrintCondition()

- Case 3 When changing the current print condition setting:
  In plugin PrintConditionGUI: call to
  SUnwinderManager_Callbacks::OnUpdateCurrentPrintCondition() in
  CDataIF::SaveCurrentPrintCondition()
- Case 4 When the job is running:
  In plugin JobPrintSequence: call to SUnwinderManager_Callbacks::OnFirstJobRun() in
  JobPrintManager::runJob()

3. Detail implementation

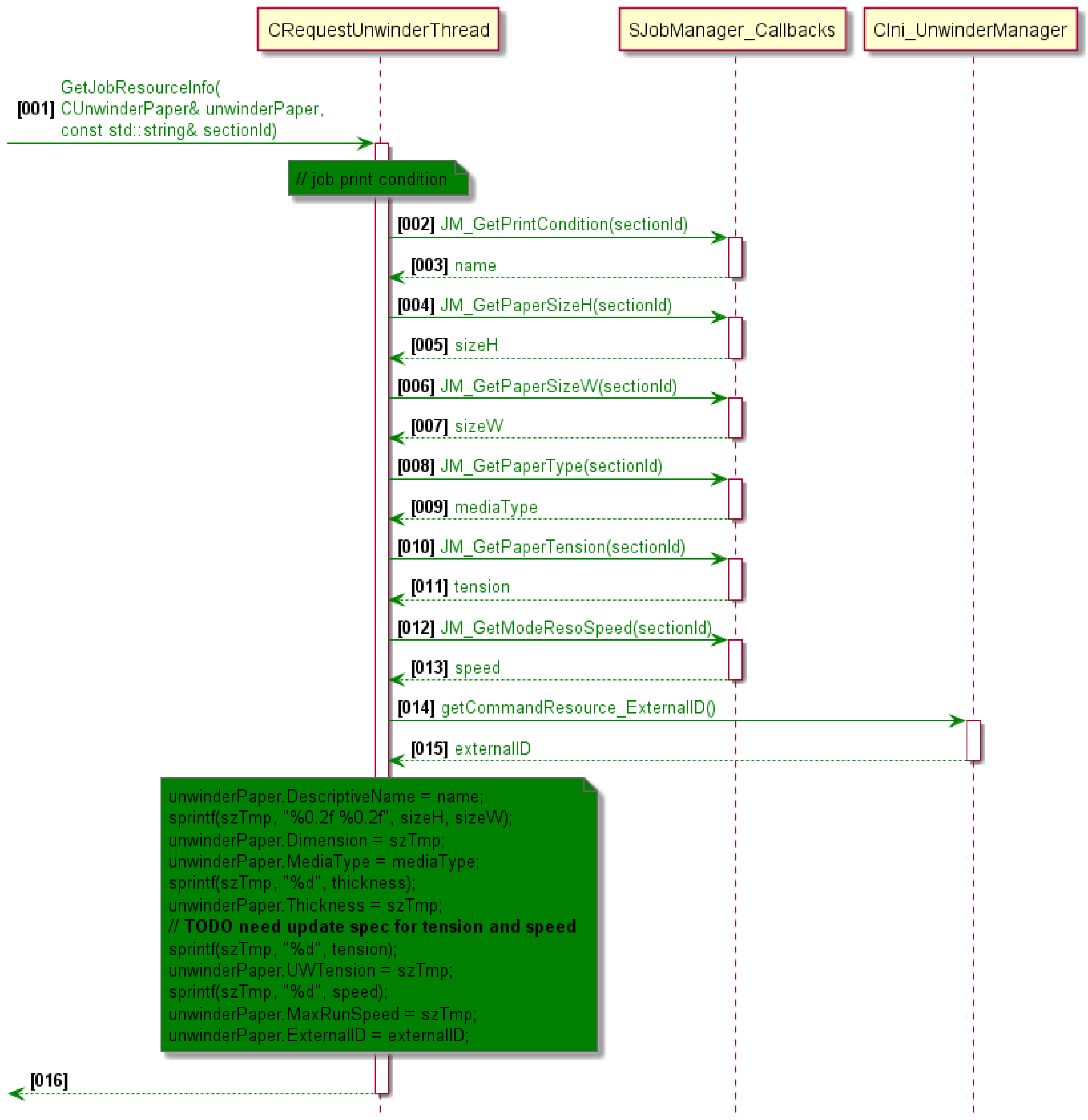

203.1 Notify UW of print condition information
CRequestUnwinderThread
CRequestUnwinder
[001] NotifyAndQueryResource(sectionId)
CUnwinderPaper unwinderPaper;
alt
[sectionId == ""]
[002] GetPrintConditionResourceInfo(&unwinderPaper)
ref
Refer to 203.2
[003]
[004] GetJobResourceInfo(&unwinderPaper, sectionId)
ref
Refer to 203.3
[005]
[006] RequestCommandResource(unwinderPaper)
[007] result
ref
Refer to 202.2
[008]

203.2 Get current print condition information


CRequestUnwinderThread
SPaperDB_Callbacks
CIni_UnwinderManager
[001] GetPrintConditionResourceInfo(UnwinderPaper& unwinderPaper)
// current print condition
[002] PDB_GetCurrentPrintCondition()
[003] name
[004] PDB_GetPaperSizeHByUnit(name)
[005] sizeH
[006] PDB_GetPaperSizeW(name)
[007] sizeW
[008] PDB_GetPaperType(name)
[009] mediaType
[010] PDB_GetPaperThickness(name)
[011] thickness
[012] PDB_GetPaperTension(name)
[013] tension
[014] PDB_GetModeResoSpeed(name)
[015] speed
[016] getCommandResource_ExternalID()
[017] externalID
[018]

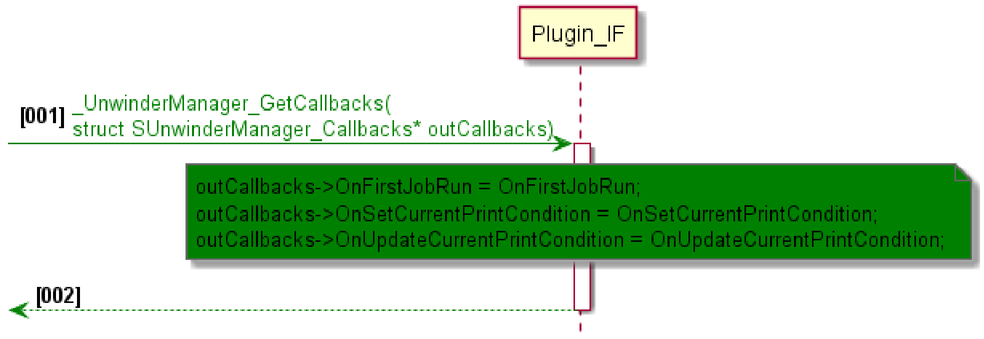
```
unwinderPaper.DescriptiveName = name;
sprintf(szTmp, "%0.2f %0.2f", sizeH, sizeW);
unwinderPaper.Dimension = szTmp;
unwinderPaper.MediaType = mediaType;
sprintf(szTmp, "%d", thickness);
unwinderPaper.Thickness = szTmp;
// TODO need update spec for tension and speed
sprintf(szTmp, "%d", tension);
unwinderPaper.UWTension = szTmp;
sprintf(szTmp, "%d", speed);
unwinderPaper.MaxRunSpeed = szTmp;
unwinderPaper.ExternalID = externalID;
```

203.3 Get job print condition information


CRequestUnwinderThread
SJobManager_Callbacks
CIni_UnwinderManager
[001] GetJobResourceInfo(CUnwinderPaper& unwinderPaper, const std::string& sectionId)
// job print condition
[002] JM_GetPrintCondition(sectionId)
[003] name
[004] JM_GetPaperSizeH(sectionId)
[005] sizeH
[006] JM_GetPaperSizeW(sectionId)
[007] sizeW
[008] JM_GetPaperType(sectionId)
[009] mediaType
[010] JM_GetPaperTension(sectionId)
[011] tension
[012] JM_GetModeResoSpeed(sectionId)
[013] speed
[014] getCommandResource_ExternalID()
[015] externalID
[016]

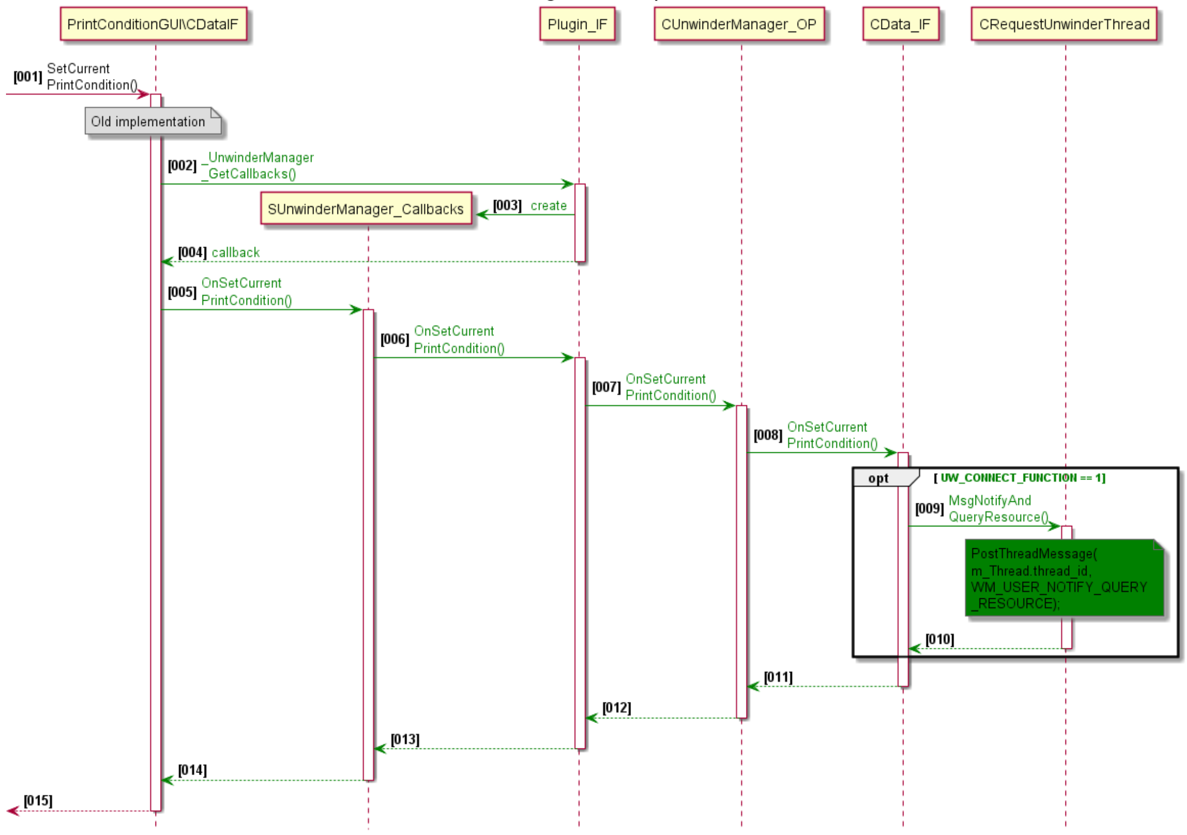
```
unwinderPaper.DescriptiveName = name;
sprintf(szTmp, "%0.2f %0.2f", sizeH, sizeW);
unwinderPaper.Dimension = szTmp;
unwinderPaper.MediaType = mediaType;
sprintf(szTmp, "%d", thickness);
unwinderPaper.Thickness = szTmp;
// TODO need update spec for tension and speed
sprintf(szTmp, "%d", tension);
unwinderPaper.UWTension = szTmp;
sprintf(szTmp, "%d", speed);
unwinderPaper.MaxRunSpeed = szTmp;
unwinderPaper.ExternalID = externalID;
```

203.4 Set callback functions


Plugin_IF
[001] _UnwinderManager_GetCallbacks(struct SUnwinderManager_Callbacks* outCallbacks)
[002]

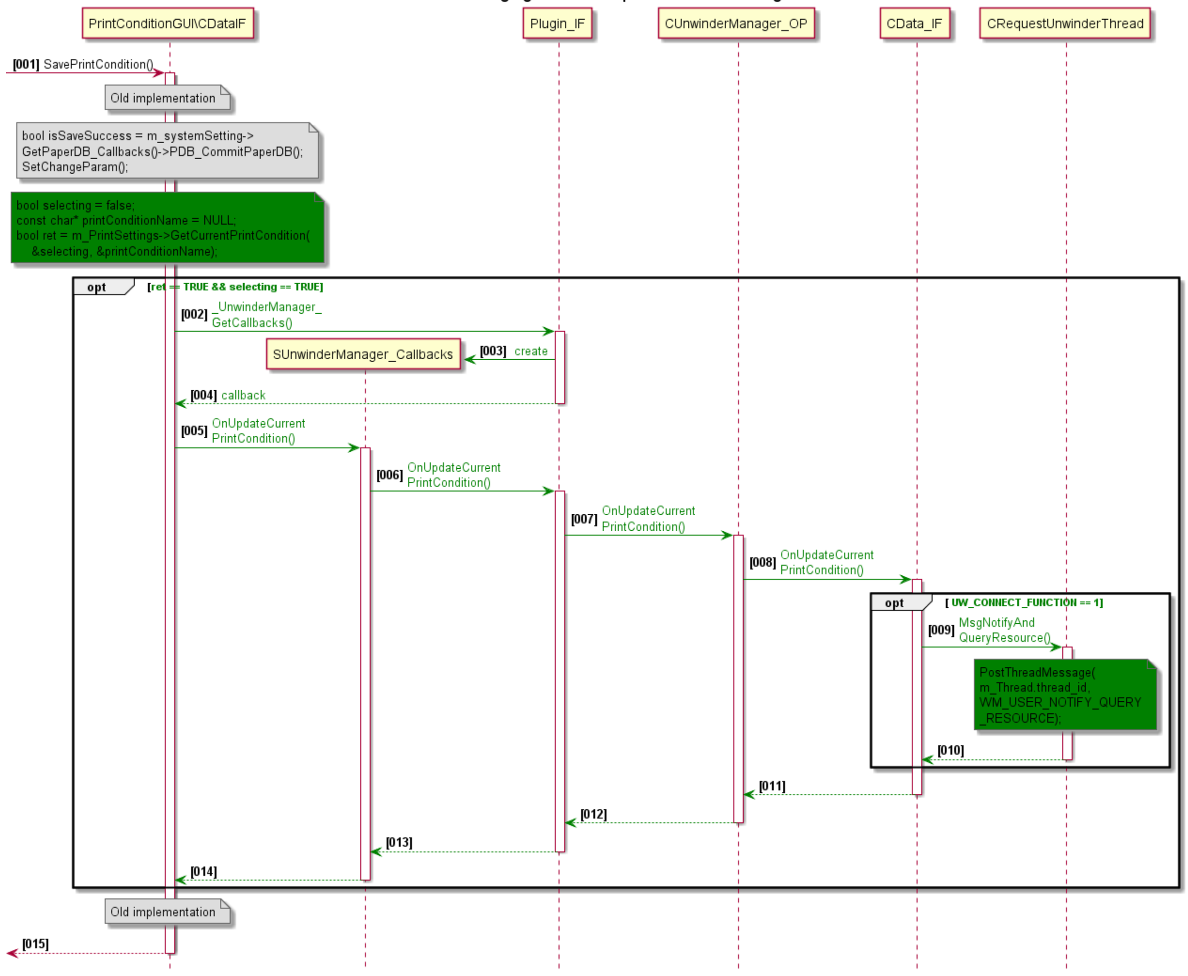
```
outCallbacks->OnFirstJobRun = OnFirstJobRun;
outCallbacks->OnSetCurrentPrintCondition = OnSetCurrentPrintCondition;
outCallbacks->OnUpdateCurrentPrintCondition = OnUpdateCurrentPrintCondition;
```

203.5 Switching the current print condition


PrintConditionGUI\CDataIF
Plugin_IF
CUnwinderManager_OP
CData_IF
CRequestUnwinderThread
[001] SetCurrent PrintCondition()
Old implementation
[002] _UnwinderManager _GetCallbacks()
SUnwinderManager_Callbacks
[003] create
[004] callback
[005] OnSetCurrent PrintCondition()
[006] OnSetCurrent PrintCondition()
[007] OnSetCurrent PrintCondition()
[008] OnSetCurrent PrintCondition()
opt
[UW_CONNECT_FUNCTION == 1]
[009] MsgNotifyAnd QueryResource()
[010]
[011]
[012]
[013]
[014]
[015]

```
PostThreadMessage(
m_Thread.thread_id,
WM_USER_NOTIFY_QUERY
_RESOURCE);
```

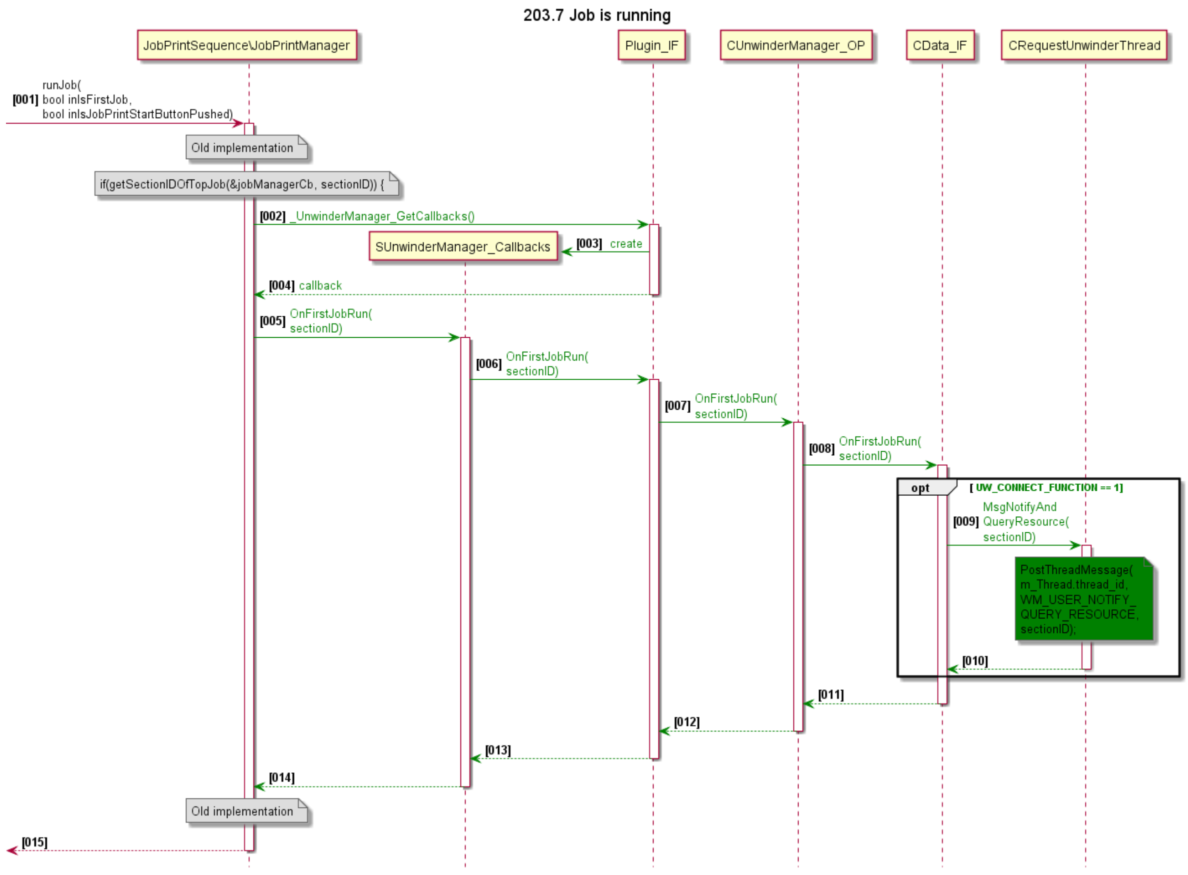
203.6 Changing the current print condition setting


PrintConditionGUI\CDataIF
Plugin_IF
CUnwinderManager_OP
CData_IF
CRequestUnwinderThread
[001] SavePrintCondition()
Old implementation
opt
[ret == TRUE && selecting == TRUE]
[002] _UnwinderManager_ GetCallbacks()
SUnwinderManager_Callbacks
[003] create
[004] callback
[005] OnUpdateCurrent PrintCondition()
[006] OnUpdateCurrent PrintCondition()
[007] OnUpdateCurrent PrintCondition()
[008] OnUpdateCurrent PrintCondition()
opt
[UW_CONNECT_FUNCTION == 1]
[009] MsgNotifyAnd QueryResource()
[010]
[011]
[012]
[013]
[014]
Old implementation
[015]

```
bool isSaveSuccess = m_systemSetting->
GetPaperDB_Callbacks()->PDB_CommitPaperDB();
SetChangeParam();
```

```
bool selecting = false;
const char* printConditionName = NULL;
bool ret = m_PrintSettings->GetCurrentPrintCondition(
    &selecting, &printConditionName);
```

```
PostThreadMessage(
m_Thread.thread_id,
WM_USER_NOTIFY_QUERY
_RESOURCE);
```


203.7 Job is running
JobPrintSequence\JobPrintManager
Plugin_IF
CUnwinderManager_OP
CData_IF
CRequestUnwinderThread
[001] runJob(bool inIsFirstJob, bool inIsJobPrintStartButtonPushed)
Old implementation
if(getSectionIDOfTopJob(&jobManagerCb, sectionID)) {
[002] _UnwinderManager_GetCallbacks()
SUnwinderManager_Callbacks
[003] create
[004] callback
[005] OnFirstJobRun(sectionID)
[006] OnFirstJobRun(sectionID)
[007] OnFirstJobRun(sectionID)
[008] OnFirstJobRun(sectionID)
opt
[UW_CONNECT_FUNCTION == 1]
[009] MsgNotifyAnd QueryResource(sectionID)
PostThreadMessage(m_Thread.thread_id, WM_USER_NOTIFY_ QUERY_RESOURCE, sectionID);
[010]
[011]
[012]
[013]
[014]
Old implementation
[015]

205. Reflect paper information notified from UW (Paper thickness, Roll diameter, Paper remaining amount)

1. Description

205-1. Management of roll diameter and remaining amount of paper

Save the roll diameter and remaining amount of paper in the TP-UW_Communication.ini file.

205-2. Notification timing of paper information from UW to the controller

After the transfer instruction by the controller, UW notifies the paper information at the interval specified at the time of registration of the paper information notification.
If no interval is specified, the paper information will be notified at the interval specified by UW.
However, if the UW is equipped with a paper thickness gauge and the paper thickness changes, the UW will promptly notify the controller.
The controller promptly reflects the paper information notified by UW.
The paper thickness will be reflected in the applicable printing conditions.
Update the TP-UW_Communication.ini file for the remaining amount of paper and roll diameter.

205-3. Reflect paper thickness information in printing condition.

The UW paper thickness is reflected in the paper thickness information of the current print condition or the print condition during job execution.

205-4. UW / RW icon and paper remaining amount display (※Additional update planned)

UW / RW display is added to the left and right of the printer icon to visually express the roll diameter and remaining amount (in meters).
When the remaining amount approaches the warning threshold, change the winding core from white to yellow to red.
Also, consider the UW / RW icon when the roll changer is installed.

2. Solution

- Create a class for receiving thread of the signal status information from UW:

ReceiveSignalStatusThread.h

```
class CReceiveSignalStatusThread
{
public:
    CReceiveSignalStatusThread();
    virtual ~CReceiveSignalStatusThread();
    // Start a thread
    void StartThread();
    void EndThread();
private:
    static UINT __stdcall ThreadFunction( void* pData );
    ST_THREAD_INFO m_Thread;
    HANDLE m_ExitThreadEvent;
};
```

- In function CRequestUnwinderThread::CheckUWStatus(), when receive the response from UW sucessfully, start a thread for receiving the signal status information.
- Create a class for receiving the signal status info and processing for the received info:

ReceiveSignalStatus.h

```
class CReceiveSignalStatus
{
public:
    CReceiveSignalStatus(CWnd* pWnd);
    virtual ~CReceiveSignalStatus();
    // Receive signal status info from UW
    void ReceiveSignalStatusInfo()
    // Receive signal status info notified from UWandRW_Receiver
    BOOL ReceiveInfo();
    // setting for timeout timer of SignalStatus(STATUS)
    void SetTimerStatusReceive();
    // setting for timeout timer of SignalStatus(PAPER)
    void SetTimerPaperReceive();
    // stop timeout timer of SignalStatus(STATUS)
    void KillTimerStatusReceive();
    // stop timeout timer of SignalStatus(PAPER)
    void KillTimerPaperReceive();
    // whether or not the paper info is received when the controller is started.
    void IsPaperInfoReceivingWhenStart();
    // whether or not the paper info has been received
    void HasPaperInfoBeenReceived();
    // get thickness value which is notified from UW
    long GetUWThickness();
private:
    // pipe reading
    BOOL ReadData( HANDLE inPipe, char* outData, DWORD inSize );
    // analyze signal status info notified from UWandRW_Receiver
    BOOL AnalyzeData( std::string& inXmldata );
    // call UWandRW_Parse_Xml.exe to parse the receiving info
    std::string ExecuteParseXml( std::string& inSignalData );
    // check whether the data returned from UWandRW_Parse_Xml.exe is valid or not
    BOOL CheckPickupData( const std::string& inData );
    // extract data from the parsed info
    std::string SelectPickupData( const std::string& inScrData,
                    const std::string& inSelectName );
    // processing when the status info is received
    BOOL ReceiveStatusInfo( const std::string& inStatus );
    // processing when the paper info is received
    BOOL ReceivePaperInfo( const std::string& inDescriptiveName,
        const std::string& inDimension,
        const std::string& inMediaType,
        const std::string& inRollDiameter,
        const std::string& inThickness );
    CWnd* m_pWnd;
    HANDLE m_ExitThreadEvent;
    HANDLE m_IoEvent;
    static bool m_isPrintNavStarted; //whether or not the
                                    // paper info is received when the controller starts.
    static bool m_receivedPaperInfo; // whether or not the paper info has been received
    long m_UWThickness; // Value of thickness notified from UW.
};
```

- In function CReceiveSignalStatus::ReceivePaperInfo:
    - Stop the current timeout timer and start a new one.
    - If the paper info is received when the controller is started (m_isPrintNavStarted is true and flag variable m_isRegisterFailed is false), compare the value of paper thickness between the current print condition and value notified from UW. If there is a difference, display a warning

message box and save the paper thickness into the print condition by using callback functions from PaperDB if “Yes” is chosen on the message box.
    - If value of m_receivedPaperInfo variable is false then set it to true to indicate that the paper info has been received from UW.
    - Save the thickness value notified from UW to variable m_UWThickness
    - Set the remaining amount of paper and roll diameter into TP-UW_Communication.ini file by new created functions: CIni_UnwinderManager::SetRollDiameter and CIni_UnwinderManager::SetPaperRemainingAmount
- Add plugin callback function to get the paper thickness notified from UW:

Common\UnwinderManager_Callbacks.h

```
typedef void (*_GetUWPaperThickness)(long &outThickness);
typedef bool (*_HasPaperInfoReceived)(void);
struct SUnwinderManager_Callbacks
{
    //Version 1
    ...
    _GetUWPaperThickness GetUWPaperThickness;
    _HasPaperInfoReceived HasPaperInfoReceived;
}
```

Common\UnwinderManager_Callbacks.h

```
    class CUnwinderManager_OP : public CUnwinderManager
    {
    public:
        ...
        void GetUWPaperThickness(long &outThickness);
        bool HasPaperInfoReceived(void);
    }
```

UnwinderManager\Data_IF.h

```
class CDataIF
{
public:
    ...
    void GetUWPaperThickness(long &outThickness);
    bool HasPaperInfoReceived(void);
}
```

- In CDataIF::SetCurrentPrintCondition and CDataIF::SavePrintCondition functions, in case there has been the paper info notified from UW, compare the paper thickness value between the current print condition and the one from UW (value is saved in m_UWThickness variable). If there is a difference then update the value from UW to the current print condition.

- In CCtlJobList::Proc, implement same as in CDataIF::SetCurrentPrintCondition and CDataIF::SavePrintCondition functions, except reflecting the paper thickness value to print condition of job, not the current print condition.

- Add a warning message to following ini files to display the warning message box for reflecting the paper thickness from UW to the current print condition or print condition of job:

strings_UnwinderManager.ini and strings_PrintConditionGUI.ini

```
[MSG]
//English
IDM_UPDATE_PAPER_THICKNESS = ??
//Japanese
IDM_UPDATE_PAPER_THICKNESS = カレント印刷条件の紙厚情報を更新しますか？
```

strings_jobmanager.ini

```
[MSG]
//English
IDM_UPDATE_PAPER_THICKNESS = ??
//Japanese
IDM_UPDATE_PAPER_THICKNESS = ジョブ実行を中止し、実行対象のジョブの紙厚情報を更新しますか？
```

- In StatusBar plugin, create a new class CCtlUnwinder for displaying of UW/RW icon and paper remaining amount:

CtlUnwinder.h

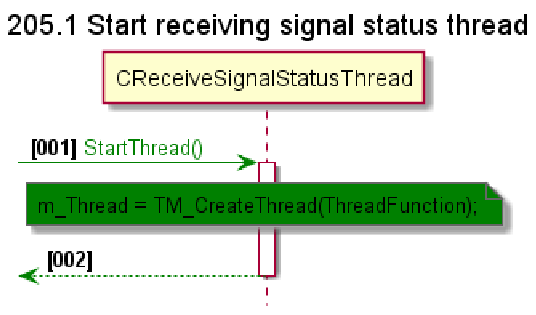
```
class CCtlUnwinder : public CBaseCtl
{
public:
    CCtlUnwinder();
    virtual ~CCtlUnwinder();
    virtual void OnUpdateState();
    virtual void OnUpdateValue();
protected:
    virtual void OnSetAttribute();
};
```

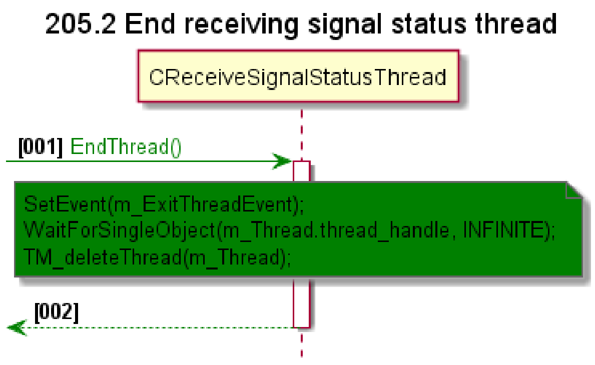
- In function CCtlUWandRW::OnUpdateValue, check the status of the UW. If it is not started, display a translucent UW icon (the top icon). If it is started, display normal icon (the bottom icon).
- In StatusBar\CDataIF class, add methods and member variable related to UW status:

StatusBarDataIF.h

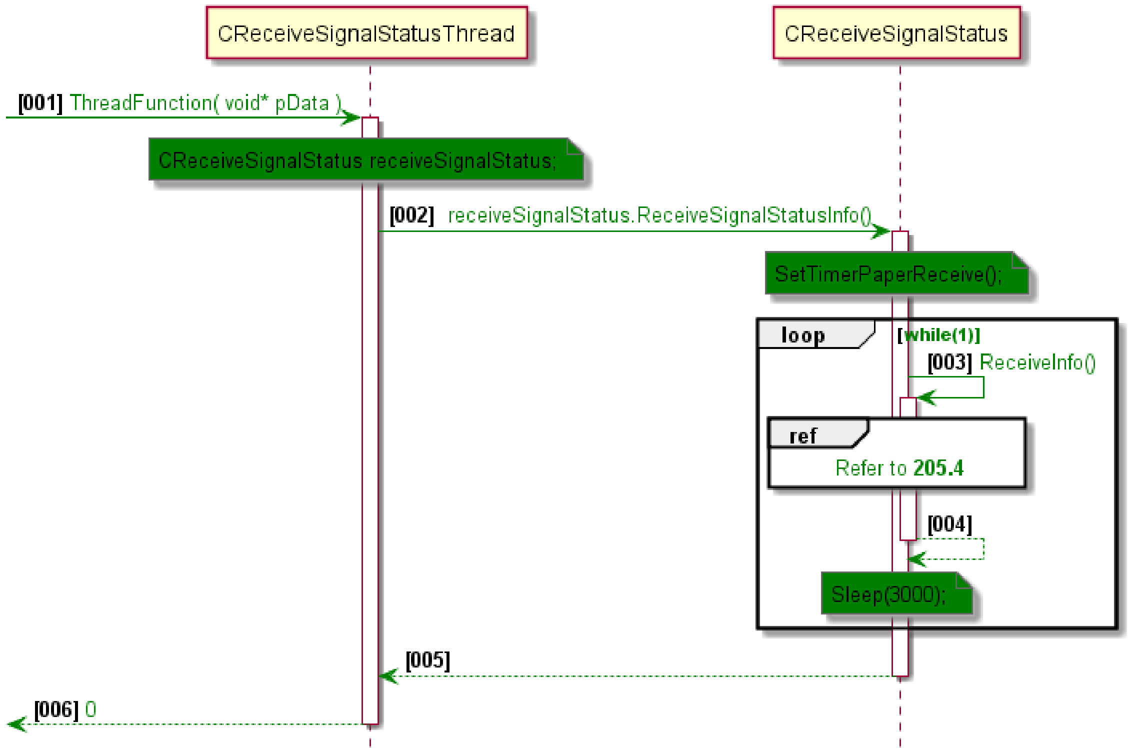
```
enum UW_STATUS{
    UW_STATUS_ON,
    UW_STATUS_OFF
};
class CDataIF
{
public:
    ...
    void SetUWStatus(UW_STATUS status);
    UW_STATUS GetUWStatus();
protected:
    UW_STATUS m_UWStatus;
};
```

3. Detail implementation


205.1 Start receiving signal status thread
CReceiveSignalStatusThread
[001] StartThread()
m_Thread = TM_CreateThread(ThreadFunction);
[002]


205.2 End receiving signal status thread
CReceiveSignalStatusThread
[001] EndThread()
SetEvent(m_ExitThreadEvent);
WaitForSingleObject(m_Thread.thread_handle, INFINITE);
TM_deleteThread(m_Thread);
[002]

205.3 Receiving thread



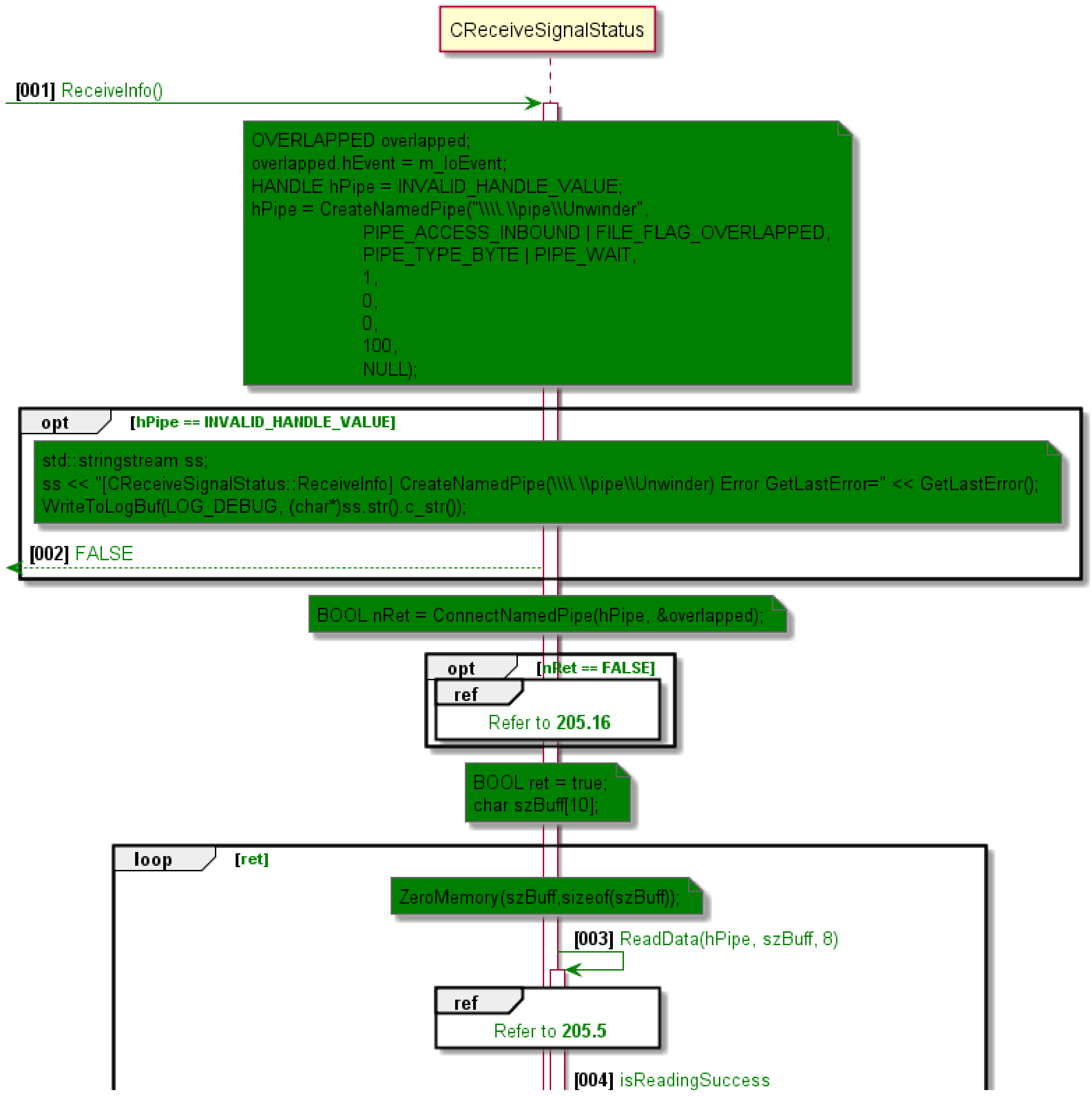
CReceiveSignalStatusThread
CReceiveSignalStatus
[001] ThreadFunction(void* pData)
[002] receiveSignalStatus.ReceiveSignalStatusInfo()
loop [while(1)]
[003] ReceiveInfo()
ref
Refer to 205.4
[004]
[005]
[006] 0

```
CReceiveSignalStatus receiveSignalStatus;
```

```
SetTimerPaperReceive();
```

```
Sleep(3000);
```

205.4 Receive signal status notified from UWandRW_Receiver



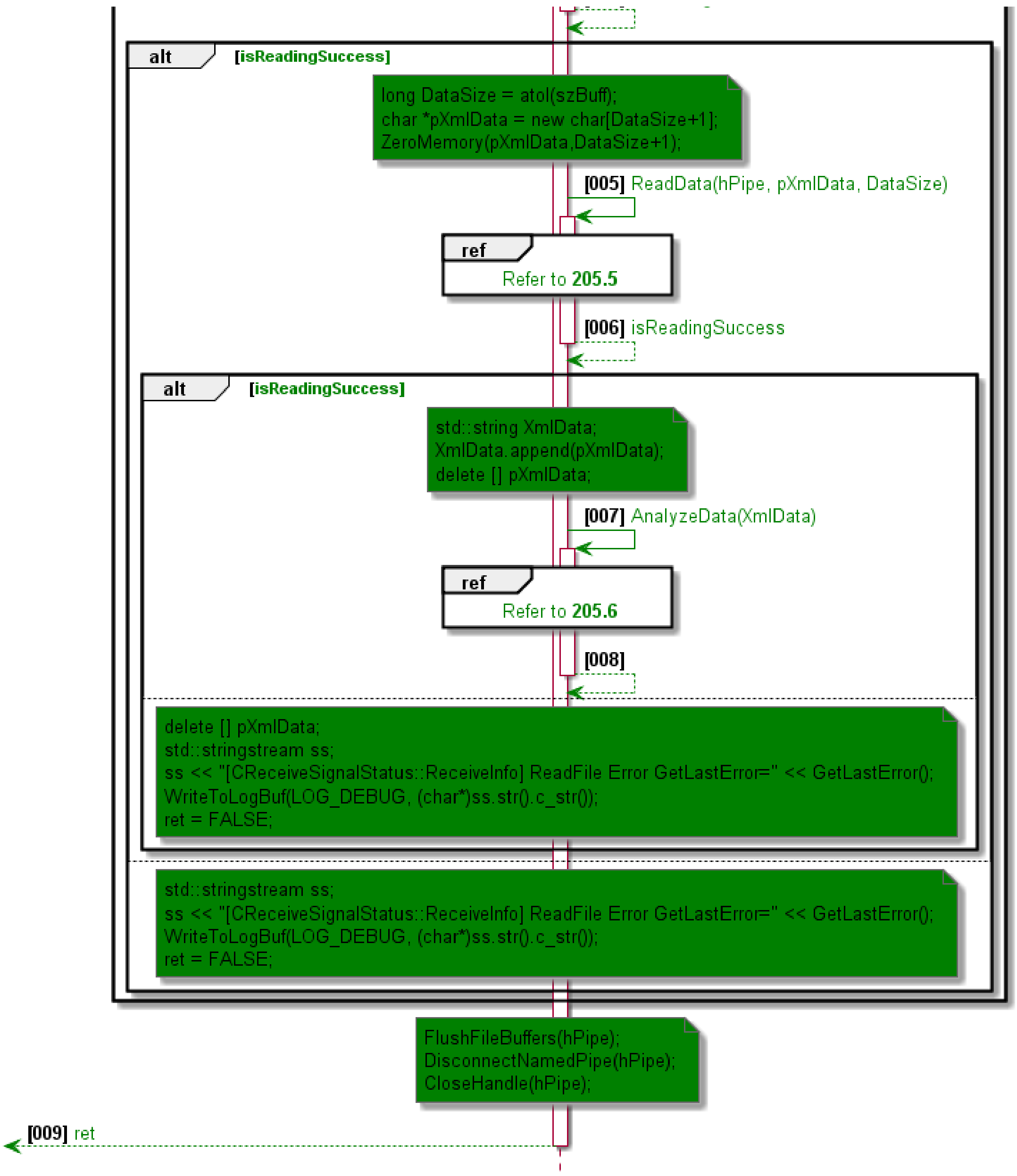
```
OVERLAPPED overlapped;
overlapped.hEvent = m_loEvent;
HANDLE hPipe = INVALID_HANDLE_VALUE;
hPipe = CreateNamedPipe("\\\\.\\pipe\\Unwinder",
                        PIPE_ACCESS_INBOUND | FILE_FLAG_OVERLAPPED,
                        PIPE_TYPE_BYTE | PIPE_WAIT,
                        1,
                        0,
                        0,
                        100,
                        NULL);
```

```
std::stringstream ss;
ss << "[CReceiveSignalStatus::ReceiveInfo] CreateNamedPipe(\\\\.\\pipe\\Unwinder) Error GetLastError=" << GetLastError();
WriteToLogBuf(LOG_DEBUG, (char*)ss.str().c_str());
```

```
BOOL nRet = ConnectNamedPipe(hPipe, &overlapped);
```

```
BOOL ret = true;
char szBuff[10];
```

```
ZeroMemory(szBuff,sizeof(szBuff));
```



```
long DataSize = atol(szBuff);
char *pXmlData = new char[DataSize+1];
ZeroMemory(pXmlData,DataSize+1);
```

[005] ReadData(hPipe, pXmlData, DataSize)

[006] isReadingSuccess

```
std::string XmlData;
XmlData.append(pXmlData);
delete [] pXmlData;
```

[007] AnalyzeData(XmlData)

[008]

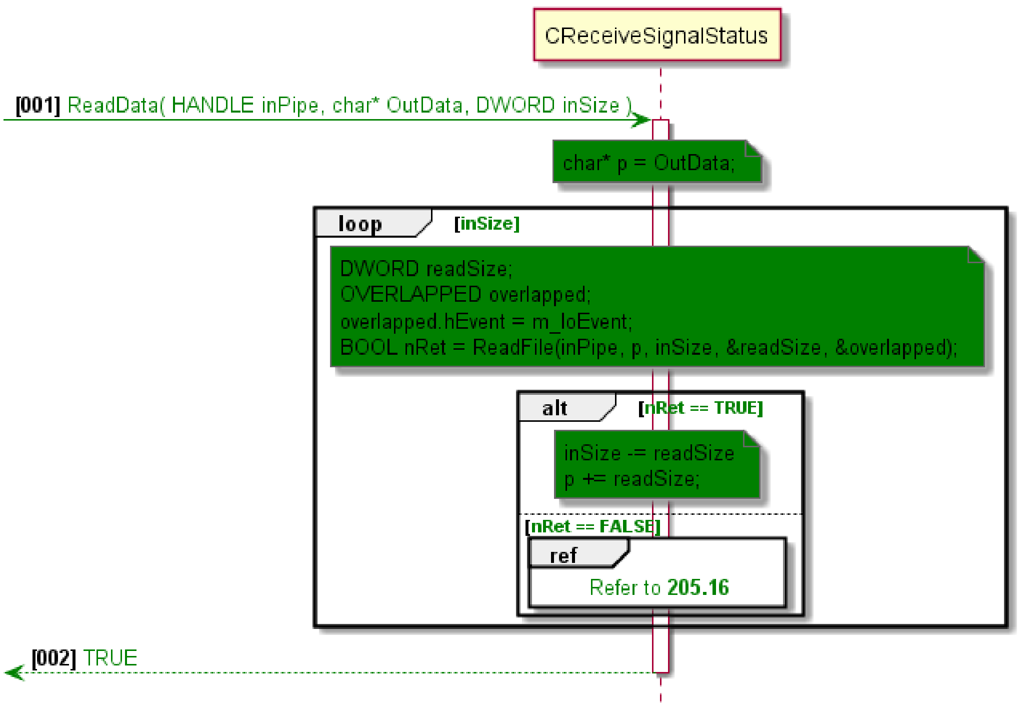
```
delete [] pXmlData;
std::stringstream ss;
ss << "[CReceiveSignalStatus::ReceiveInfo] ReadFile Error GetLastError=" << GetLastError();
WriteToLogBuf(LOG_DEBUG, (char*)ss.str().c_str());
ret = FALSE;
```

```
std::stringstream ss;
ss << "[CReceiveSignalStatus::ReceiveInfo] ReadFile Error GetLastError=" << GetLastError();
WriteToLogBuf(LOG_DEBUG, (char*)ss.str().c_str());
ret = FALSE;
```

```
FlushFileBuffers(hPipe);
DisconnectNamedPipe(hPipe);
CloseHandle(hPipe);
```

[009] ret

205.5 Pipe reading



[001] ReadData(HANDLE inPipe, char* OutData, DWORD inSize)

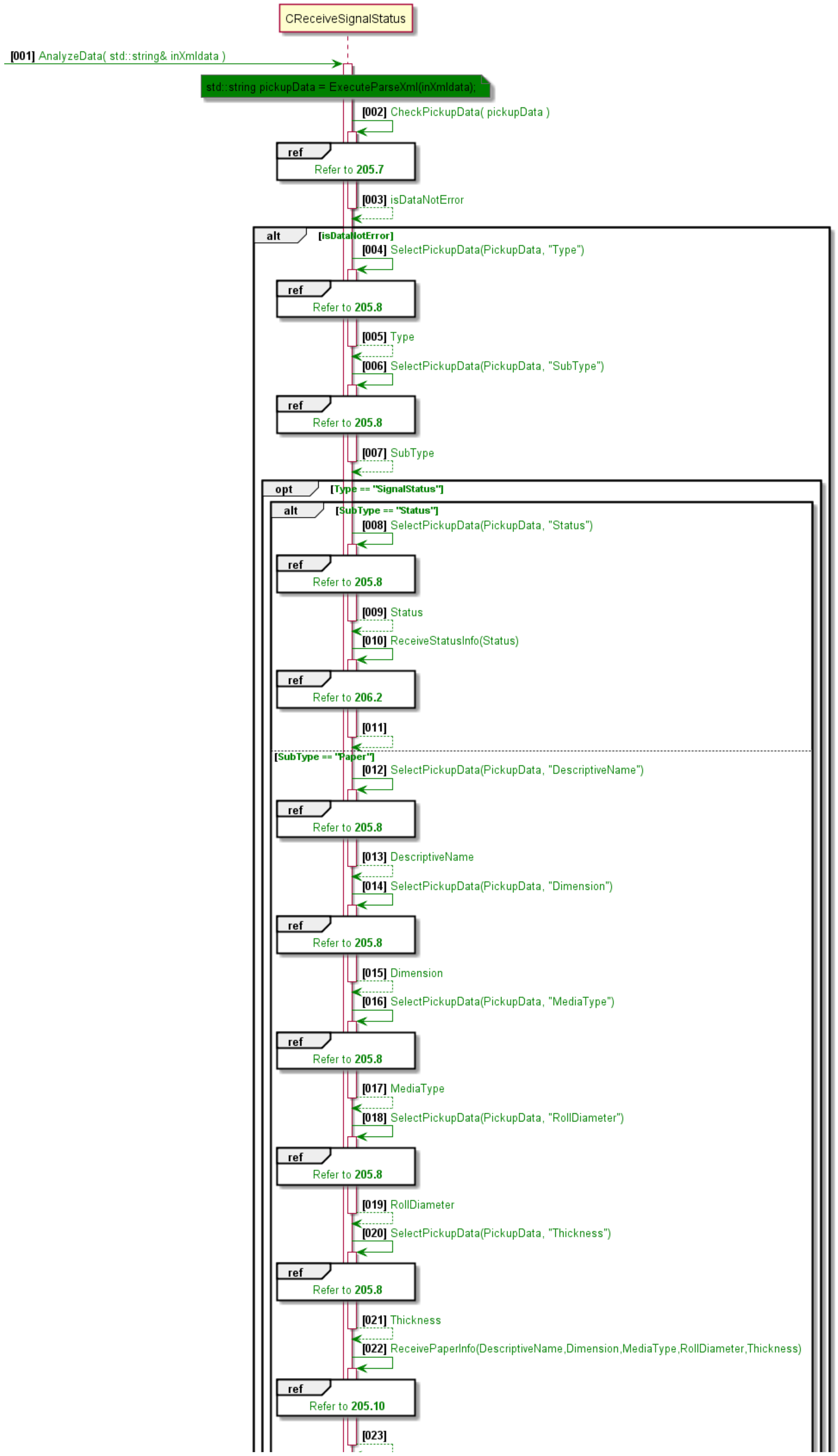
```
char* p = OutData;
```

```
DWORD readSize;
OVERLAPPED overlapped;
overlapped.hEvent = m_IoEvent;
BOOL nRet = ReadFile(inPipe, p, inSize, &readSize, &overlapped);
```

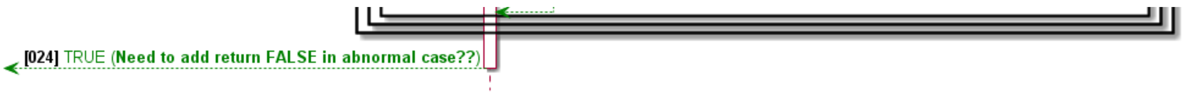
```
inSize -= readSize
p += readSize;
```

[002] TRUE

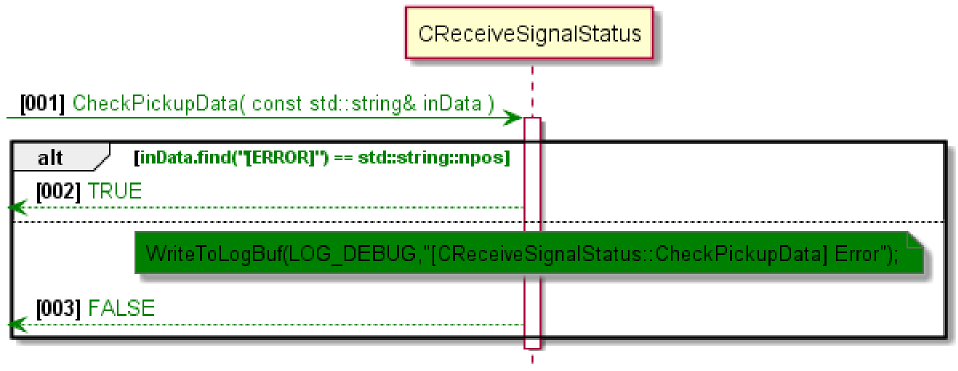
205.6 Analyze signal status info notified from UWandRW_Receiver



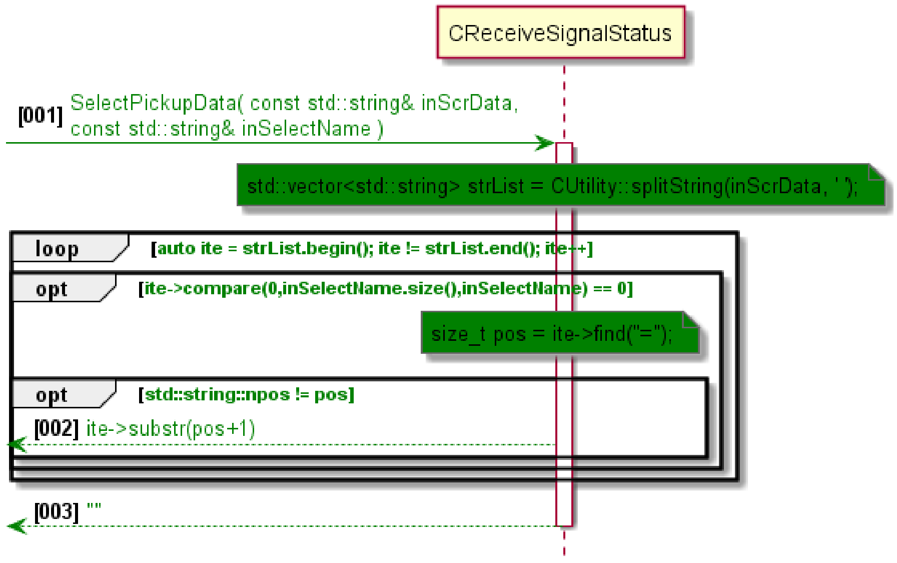
CReceiveSignalStatus
[001] AnalyzeData(std::string& inXmldata)
std::string pickupData = ExecuteParseXml(inXmldata);
[002] CheckPickupData(pickupData)
ref
Refer to 205.7
[003] isDataNotError
alt
[isDataNotError]
[004] SelectPickupData(PickupData, "Type")
ref
Refer to 205.8
[005] Type
[006] SelectPickupData(PickupData, "SubType")
ref
Refer to 205.8
[007] SubType
opt
[Type == "SignalStatus"]
alt
[SubType == "Status"]
[008] SelectPickupData(PickupData, "Status")
ref
Refer to 205.8
[009] Status
[010] ReceiveStatusInfo(Status)
ref
Refer to 206.2
[011]
[SubType == "Paper"]
[012] SelectPickupData(PickupData, "DescriptiveName")
ref
Refer to 205.8
[013] DescriptiveName
[014] SelectPickupData(PickupData, "Dimension")
ref
Refer to 205.8
[015] Dimension
[016] SelectPickupData(PickupData, "MediaType")
ref
Refer to 205.8
[017] MediaType
[018] SelectPickupData(PickupData, "RollDiameter")
ref
Refer to 205.8
[019] RollDiameter
[020] SelectPickupData(PickupData, "Thickness")
ref
Refer to 205.8
[021] Thickness
[022] ReceivePaperInfo(DescriptiveName,Dimension,MediaType,RollDiameter,Thickness)
ref
Refer to 205.10
[023]



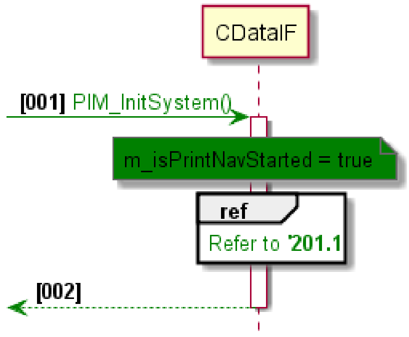
205.7 Check whether the data returned from UWandRW_Parse_Xml.exe is valid or not


CReceiveSignalStatus
[001] CheckPickupData(const std::string& inData)
alt
[inData.find("[ERROR]") == std::string::npos]
[002] TRUE
WriteToLogBuf(LOG_DEBUG,"[CReceiveSignalStatus::CheckPickupData] Error");
[003] FALSE

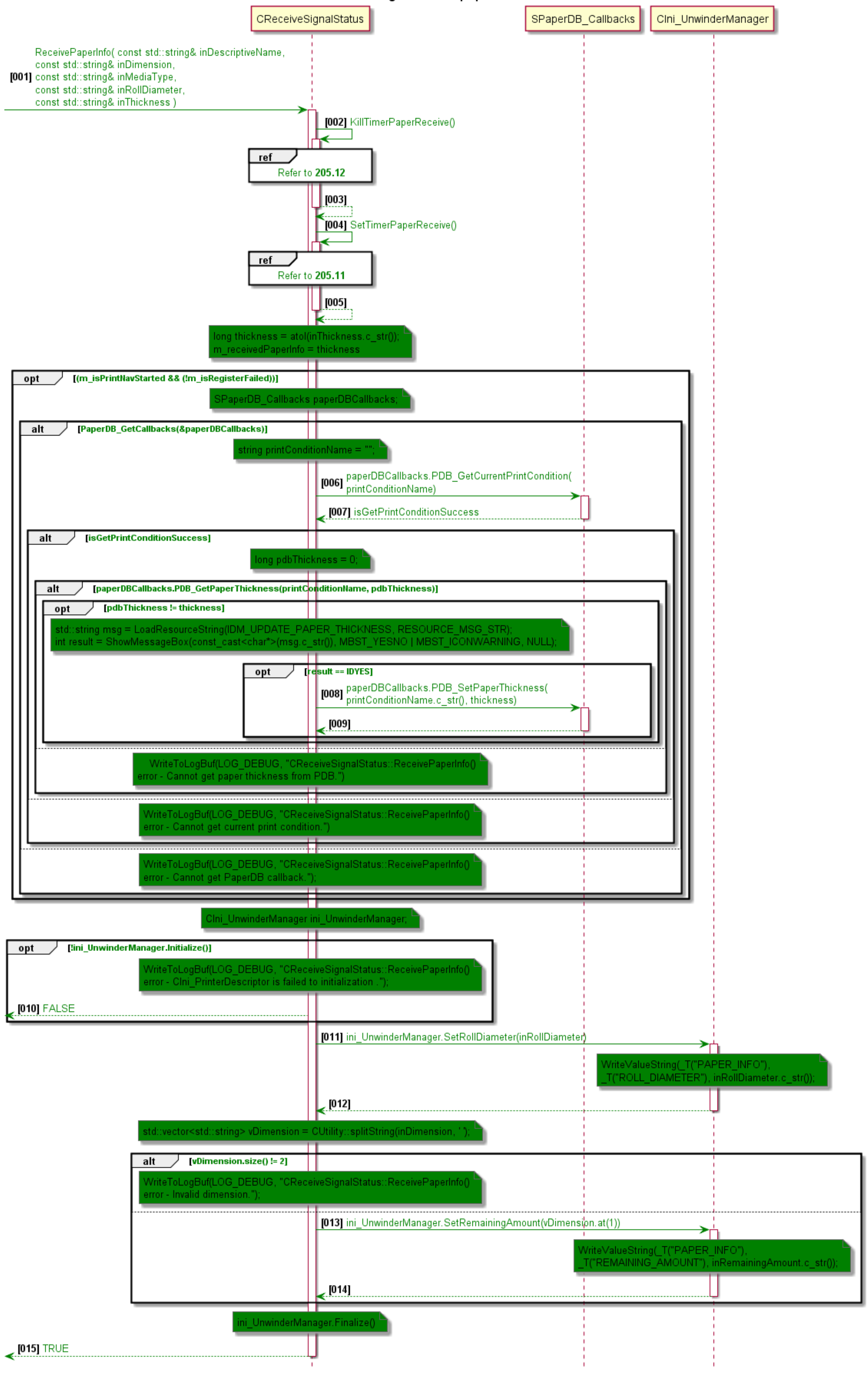
205.8 Extract data from the parsed info


CReceiveSignalStatus
[001] SelectPickupData(const std::string& inScrData, const std::string& inSelectName)
std::vector<std::string> strList = CUtility::splitString(inScrData, ' ');
loop
[auto ite = strList.begin(); ite != strList.end(); ite++]
opt
[ite->compare(0,inSelectName.size(),inSelectName) == 0]
size_t pos = ite->find("=");
opt
[std::string::npos != pos]
[002] ite->substr(pos+1)
[003] ""

205.9 Determine the time when the controller is started

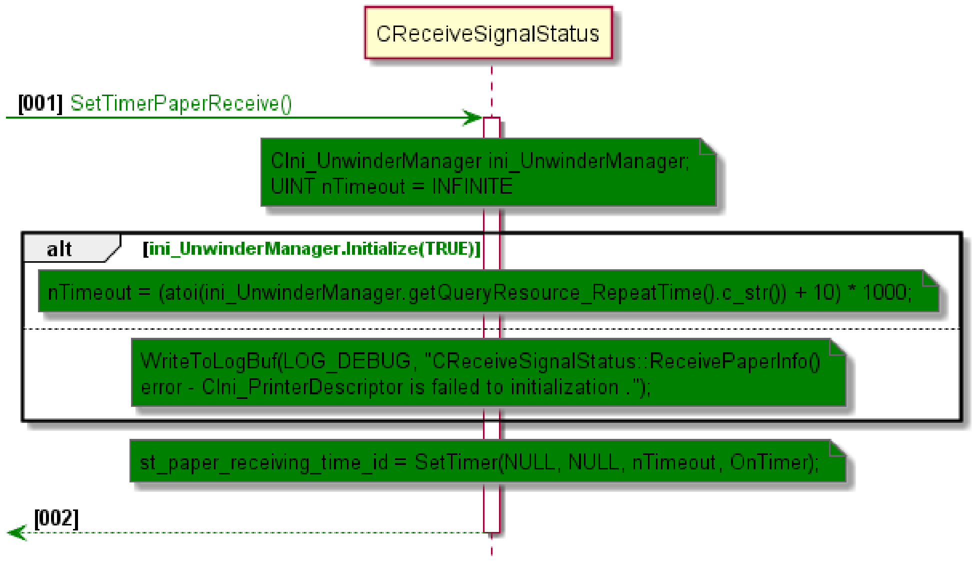


205.10 Processing when the paper info is received



CReceiveSignalStatus
SPaperDB_Callbacks
CIni_UnwinderManager
[001] ReceivePaperInfo(const std::string& inDescriptiveName, const std::string& inDimension, const std::string& inMediaType, const std::string& inRollDiameter, const std::string& inThickness)
[002] KillTimerPaperReceive()
ref
Refer to 205.12
[003]
[004] SetTimerPaperReceive()
ref
Refer to 205.11
[005]
long thickness = atol(inThickness.c_str());
m_receivedPaperInfo = thickness
opt [(m_isPrintNavStarted && (!m_isRegisterFailed))]
SPaperDB_Callbacks paperDBCallbacks;
alt [PaperDB_GetCallbacks(&paperDBCallbacks)]
string printConditionName = "";
[006] paperDBCallbacks.PDB_GetCurrentPrintCondition(printConditionName)
[007] isGetPrintConditionSuccess
alt [isGetPrintConditionSuccess]
long pdbThickness = 0;
alt [paperDBCallbacks.PDB_GetPaperThickness(printConditionName, pdbThickness)]
opt [pdbThickness != thickness]
std::string msg = LoadResourceString(IDM_UPDATE_PAPER_THICKNESS, RESOURCE_MSG_STR);
int result = ShowMessageBox(const_cast<char*>(msg.c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
opt [result == IDYES]
[008] paperDBCallbacks.PDB_SetPaperThickness(printConditionName.c_str(), thickness)
[009]
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - Cannot get paper thickness from PDB.")
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - Cannot get current print condition.")
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - Cannot get PaperDB callback.");
CIni_UnwinderManager ini_UnwinderManager;
opt [!ini_UnwinderManager.Initialize()]
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - CIni_PrinterDescriptor is failed to initialization .");
[010] FALSE
[011] ini_UnwinderManager.SetRollDiameter(inRollDiameter)
WriteValueString(_T("PAPER_INFO"), _T("ROLL_DIAMETER"), inRollDiameter.c_str());
[012]
std::vector<std::string> vDimension = CUtility::splitString(inDimension, ' ');
alt [vDimension.size() != 2]
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - Invalid dimension.");
[013] ini_UnwinderManager.SetRemainingAmount(vDimension.at(1))
WriteValueString(_T("PAPER_INFO"), _T("REMAINING_AMOUNT"), inRemainingAmount.c_str());
[014]
ini_UnwinderManager.Finalize()
[015] TRUE

205.11 Setting for timeout timer of SignalStatus(PAPER)


CReceiveSignalStatus
[001] SetTimerPaperReceive()
alt
[ini_UnwinderManager.Initialize(TRUE)]
[002]

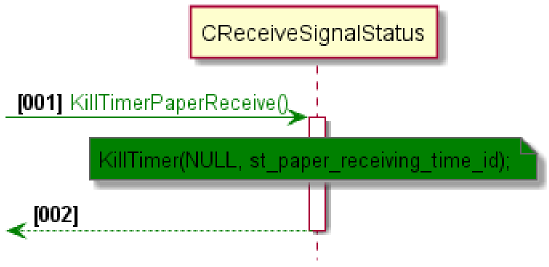
```
CIni_UnwinderManager ini_UnwinderManager;
UINT nTimeout = INFINITE
```

```
nTimeout = (atoi(ini_UnwinderManager.getQueryResource_RepeatTime().c_str()) + 10) * 1000;
```

```
WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
error - CIni_PrinterDescriptor is failed to initialization .");
```

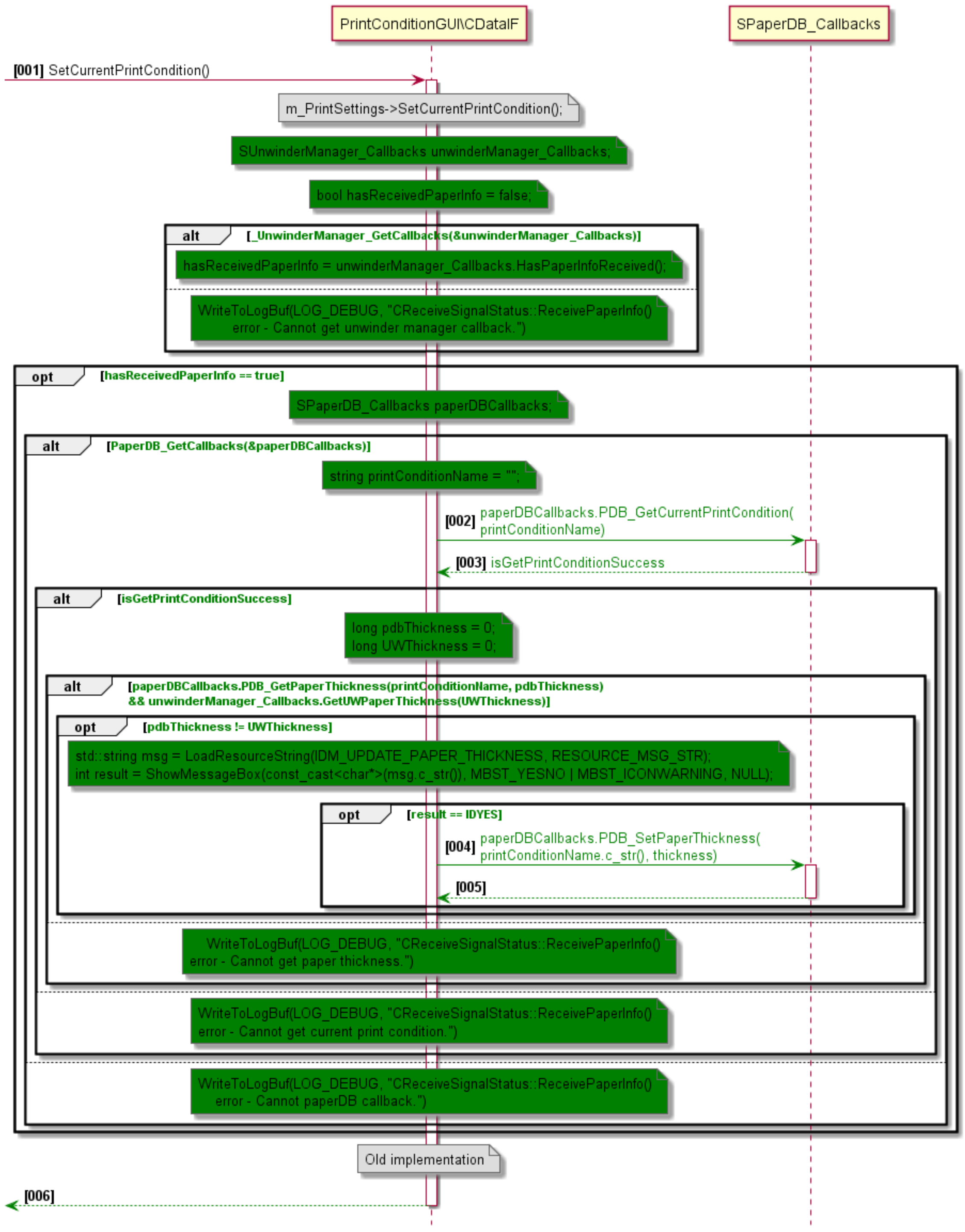
```
st_paper_receiving_time_id = SetTimer(NULL, NULL, nTimeout, OnTimer);
```

205.12 Stop timeout timer of SignalStatus(PAPER)



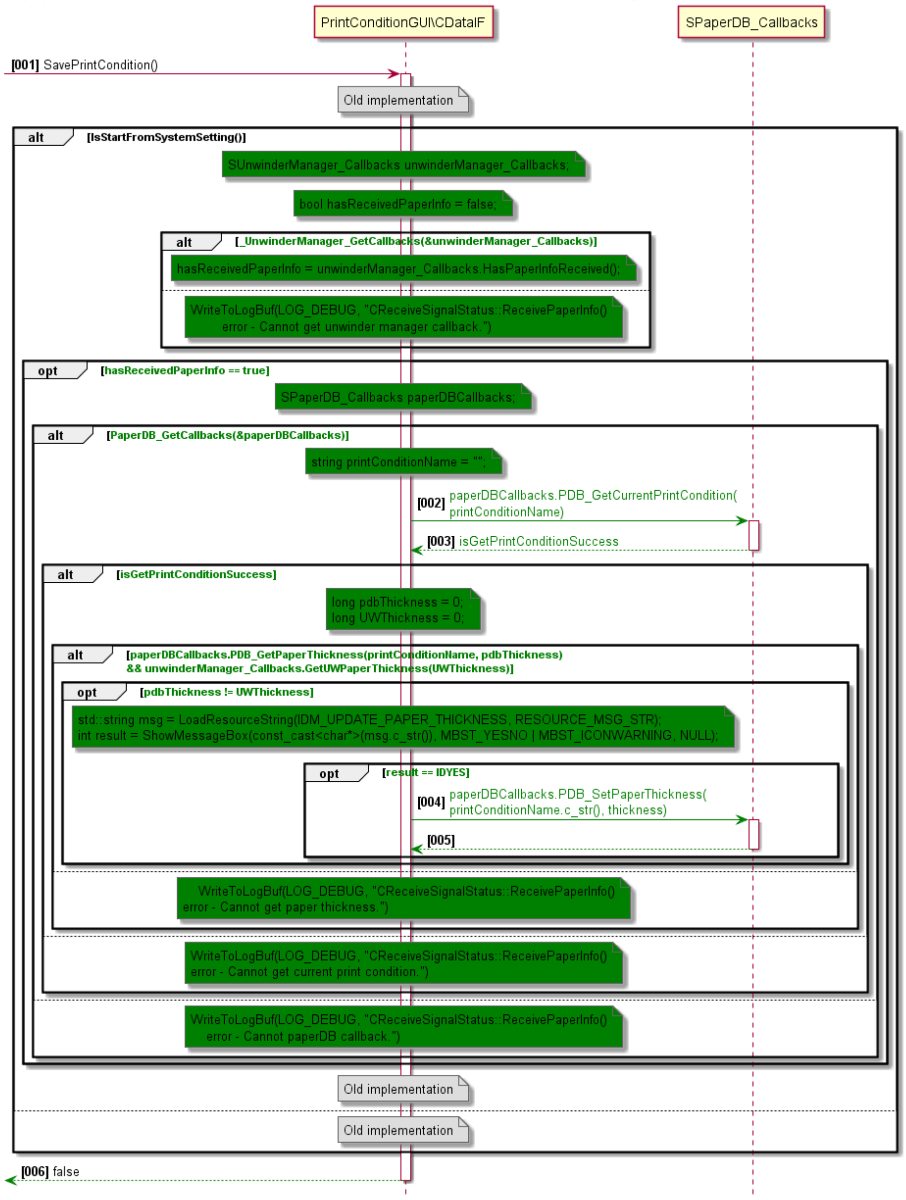
```
KillTimer(NULL, st_paper_receiving_time_id);
```

205.13 Reflect the paper thickness notified from UW to the current print condition when switch current print condition



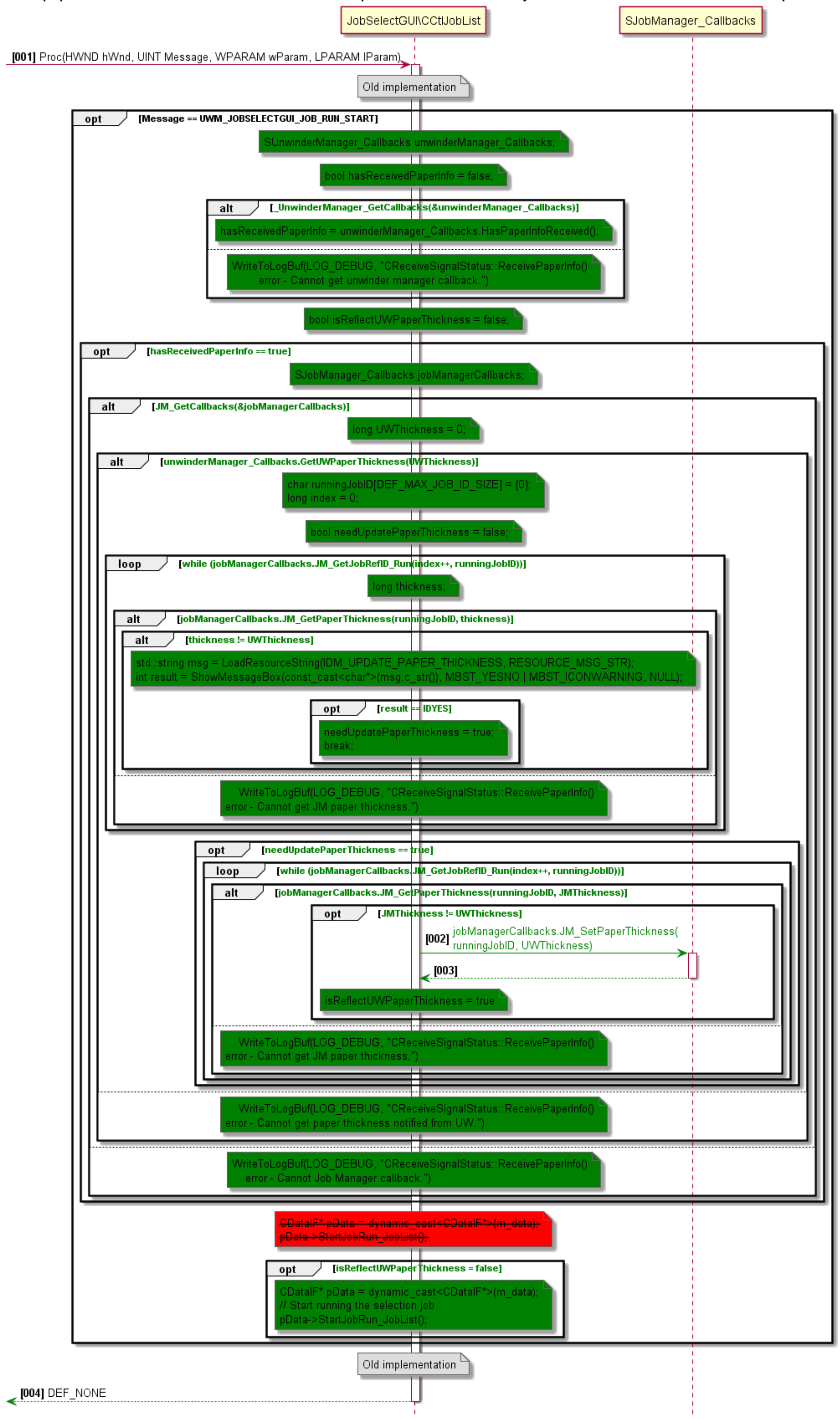
```
[001] SetCurrentPrintCondition()
m_PrintSettings->SetCurrentPrintCondition();
SUnwinderManager_Callbacks unwinderManager_Callbacks;
bool hasReceivedPaperInfo = false;
alt [_UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
    hasReceivedPaperInfo = unwinderManager_Callbacks.HasPaperInfoReceived();
    ---
    WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
         error - Cannot get unwinder manager callback.")
opt [hasReceivedPaperInfo == true]
    SPaperDB_Callbacks paperDBCallbacks;
    alt [PaperDB_GetCallbacks(&paperDBCallbacks)]
        string printConditionName = "";
        [002] paperDBCallbacks.PDB_GetCurrentPrintCondition(
              printConditionName)
        [003] isGetPrintConditionSuccess
        alt [isGetPrintConditionSuccess]
            long pdbThickness = 0;
            long UWThickness = 0;
            alt [paperDBCallbacks.PDB_GetPaperThickness(printConditionName, pdbThickness)
                 && unwinderManager_Callbacks.GetUWPaperThickness(UWThickness)]
                opt [pdbThickness != UWThickness]
                    std::string msg = LoadResourceString(IDM_UPDATE_PAPER_THICKNESS, RESOURCE_MSG_STR);
                    int result = ShowMessageBox(const_cast<char*>(msg.c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
                    opt [result == IDYES]
                        [004] paperDBCallbacks.PDB_SetPaperThickness(
                              printConditionName.c_str(), thickness)
                        [005]
                ---
                WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                error - Cannot get paper thickness.")
            ---
            WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
            error - Cannot get current print condition.")
        ---
        WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
            error - Cannot paperDB callback.")
Old implementation
[006]
```

205.14 Reflect the paper thickness notified from UW to the current print condition when the setting contents of the current print condition are changed


PrintConditionGUI\CDataIF
SPaperDB_Callbacks
[001] SavePrintCondition()
Old implementation
alt [IsStartFromSystemSetting()]
alt [_UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
opt [hasReceivedPaperInfo == true]
alt [PaperDB_GetCallbacks(&paperDBCallbacks)]
[002] paperDBCallbacks.PDB_GetCurrentPrintCondition(printConditionName)
[003] isGetPrintConditionSuccess
alt [isGetPrintConditionSuccess]
alt [paperDBCallbacks.PDB_GetPaperThickness(printConditionName, pdbThickness) && unwinderManager_Callbacks.GetUWPaperThickness(UWThickness)]
opt [pdbThickness != UWThickness]
opt [result == IDYES]
[004] paperDBCallbacks.PDB_SetPaperThickness(printConditionName.c_str(), thickness)
[005]
Old implementation
Old implementation
[006] false

```
SUnwinderManager_Callbacks unwinderManager_Callbacks;

bool hasReceivedPaperInfo = false;

hasReceivedPaperInfo = unwinderManager_Callbacks.HasPaperInfoReceived();

WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
        error - Cannot get unwinder manager callback.")

SPaperDB_Callbacks paperDBCallbacks;

string printConditionName = "";

long pdbThickness = 0;
long UWThickness = 0;

std::string msg = LoadResourceString(IDM_UPDATE_PAPER_THICKNESS, RESOURCE_MSG_STR);
int result = ShowMessageBox(const_cast<char*>(msg.c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);

    WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
error - Cannot get paper thickness.")

WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
error - Cannot get current print condition.")

WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
    error - Cannot paperDB callback.")
```

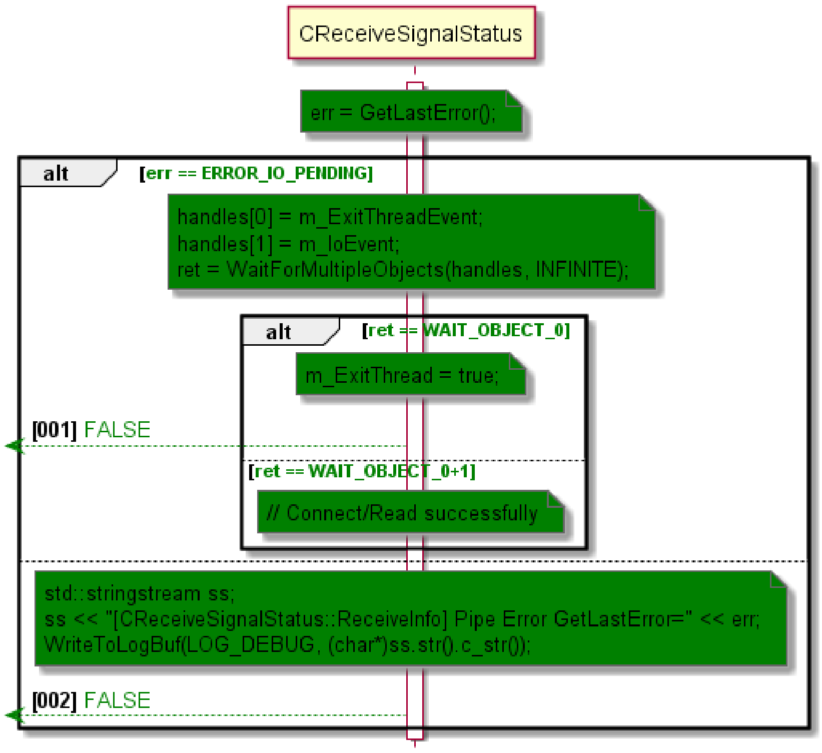
205.15 Reflect the paper thickness notified from UW to the current print condition when the job execution button or the continuous print button is pressed



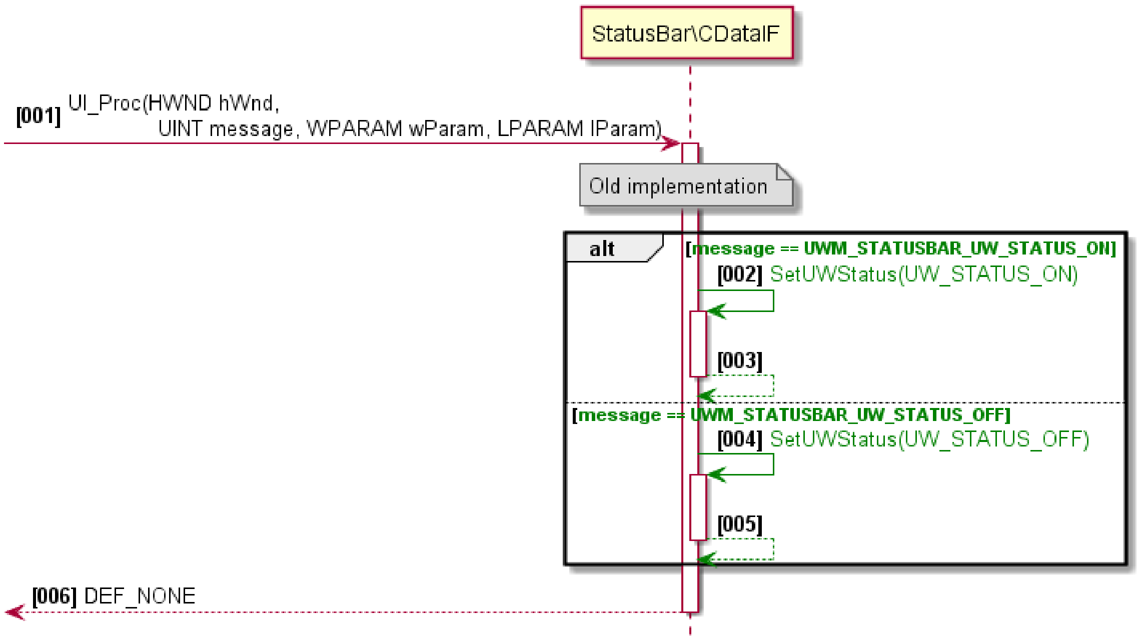
JobSelectGUI\CCtlJobList
SJobManager_Callbacks
[001] Proc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)
Old implementation

```
opt [Message == UWM_JOBSELECTGUI_JOB_RUN_START]
    SUnwinderManager_Callbacks unwinderManager_Callbacks;
    bool hasReceivedPaperInfo = false;
    alt [_UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
        hasReceivedPaperInfo = unwinderManager_Callbacks.HasPaperInfoReceived();
    ----
        WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                error - Cannot get unwinder manager callback.")
    bool isReflectUWPaperThickness = false;
    opt [hasReceivedPaperInfo == true]
        SJobManager_Callbacks jobManagerCallbacks;
        alt [JM_GetCallbacks(&jobManagerCallbacks)]
            long UWThickness = 0;
            alt [unwinderManager_Callbacks.GetUWPaperThickness(UWThickness)]
                char runningJobID[DEF_MAX_JOB_ID_SIZE] = {0};
                long index = 0;
                bool needUpdatePaperThickness = false;
                loop [while (jobManagerCallbacks.JM_GetJobRefID_Run(index++, runningJobID))]
                    long thickness;
                    alt [jobManagerCallbacks.JM_GetPaperThickness(runningJobID, thickness)]
                        alt [thickness != UWThickness]
                            std::string msg = LoadResourceString(IDM_UPDATE_PAPER_THICKNESS, RESOURCE_MSG_STR);
                            int result = ShowMessageBox(const_cast<char*>(msg.c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
                            opt [result == IDYES]
                                needUpdatePaperThickness = true;
                                break;
                    ----
                            WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                        error - Cannot get JM paper thickness.")
                opt [needUpdatePaperThickness == true]
                    loop [while (jobManagerCallbacks.JM_GetJobRefID_Run(index++, runningJobID))]
                        alt [jobManagerCallbacks.JM_GetPaperThickness(runningJobID, JMThickness)]
                            opt [JMThickness != UWThickness]
                                [002] jobManagerCallbacks.JM_SetPaperThickness(
                                      runningJobID, UWThickness)
                                [003]
                                isReflectUWPaperThickness = true
                        ----
                                WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                            error - Cannot get JM paper thickness.")
            ----
                    WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                error - Cannot get paper thickness notified from UW.")
        ----
            WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
                error - Cannot Job Manager callback.")
    ~~CDataIF* pData = dynamic_cast<CDataIF*>(m_data);~~
    ~~pData->StartJobRun_JobList();~~
    opt [isReflectUWPaperThickness = false]
        CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
        // Start running the selection job
        pData->StartJobRun_JobList();
Old implementation
[004] DEF_NONE
```

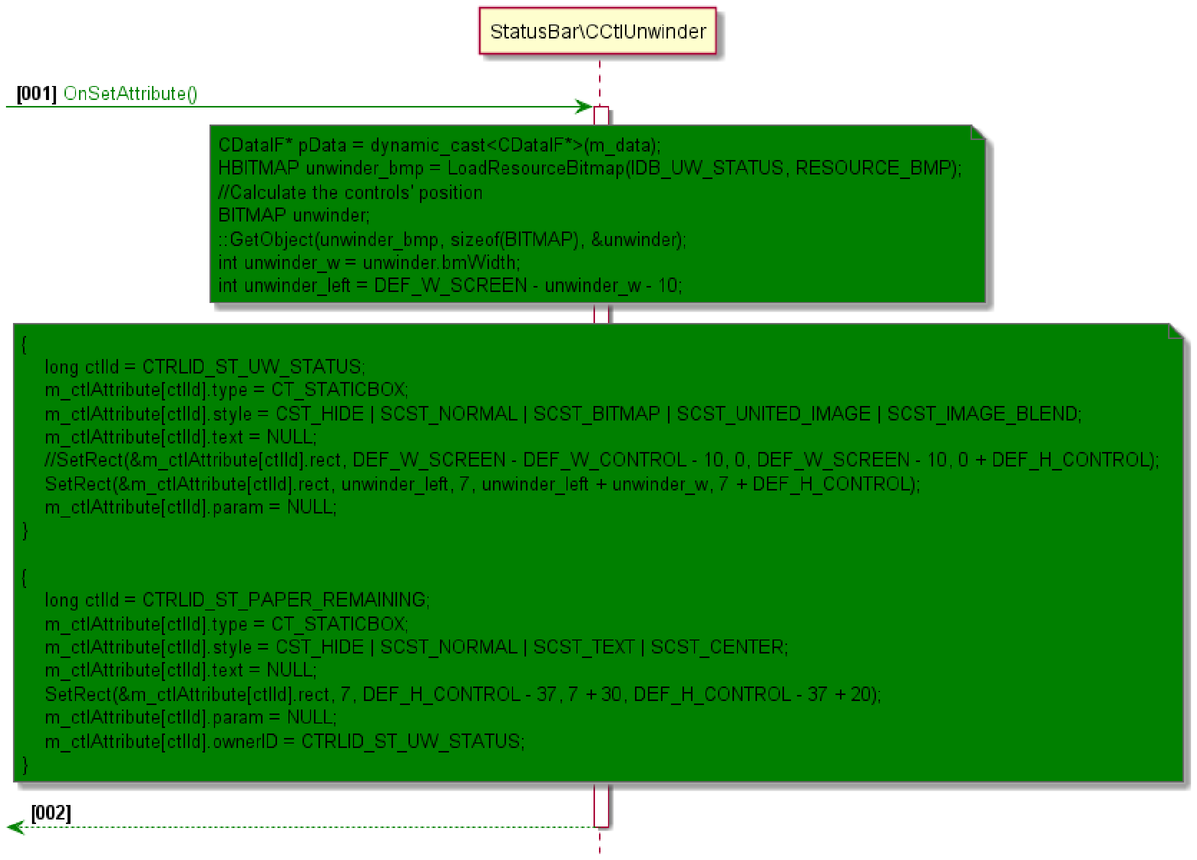
205.16 Asynchronous pipe connecting/reading


CReceiveSignalStatus
err = GetLastError();
alt
[err == ERROR_IO_PENDING]
handles[0] = m_ExitThreadEvent;
handles[1] = m_IoEvent;
ret = WaitForMultipleObjects(handles, INFINITE);
alt
[ret == WAIT_OBJECT_0]
m_ExitThread = true;
[001] FALSE
[ret == WAIT_OBJECT_0+1]
// Connect/Read successfully
std::stringstream ss;
ss << "[CReceiveSignalStatus::ReceiveInfo] Pipe Error GetLastError=" << err;
WriteToLogBuf(LOG_DEBUG, (char*)ss.str().c_str());
[002] FALSE

205.17 Status bar window message procedure


StatusBar\CDataIF
[001] UI_Proc(HWND hWnd,
UINT message, WPARAM wParam, LPARAM lParam)
Old implementation
alt
[message == UWM_STATUSBAR_UW_STATUS_ON]
[002] SetUWStatus(UW_STATUS_ON)
[003]
[message == UWM_STATUSBAR_UW_STATUS_OFF]
[004] SetUWStatus(UW_STATUS_OFF)
[005]
[006] DEF_NONE

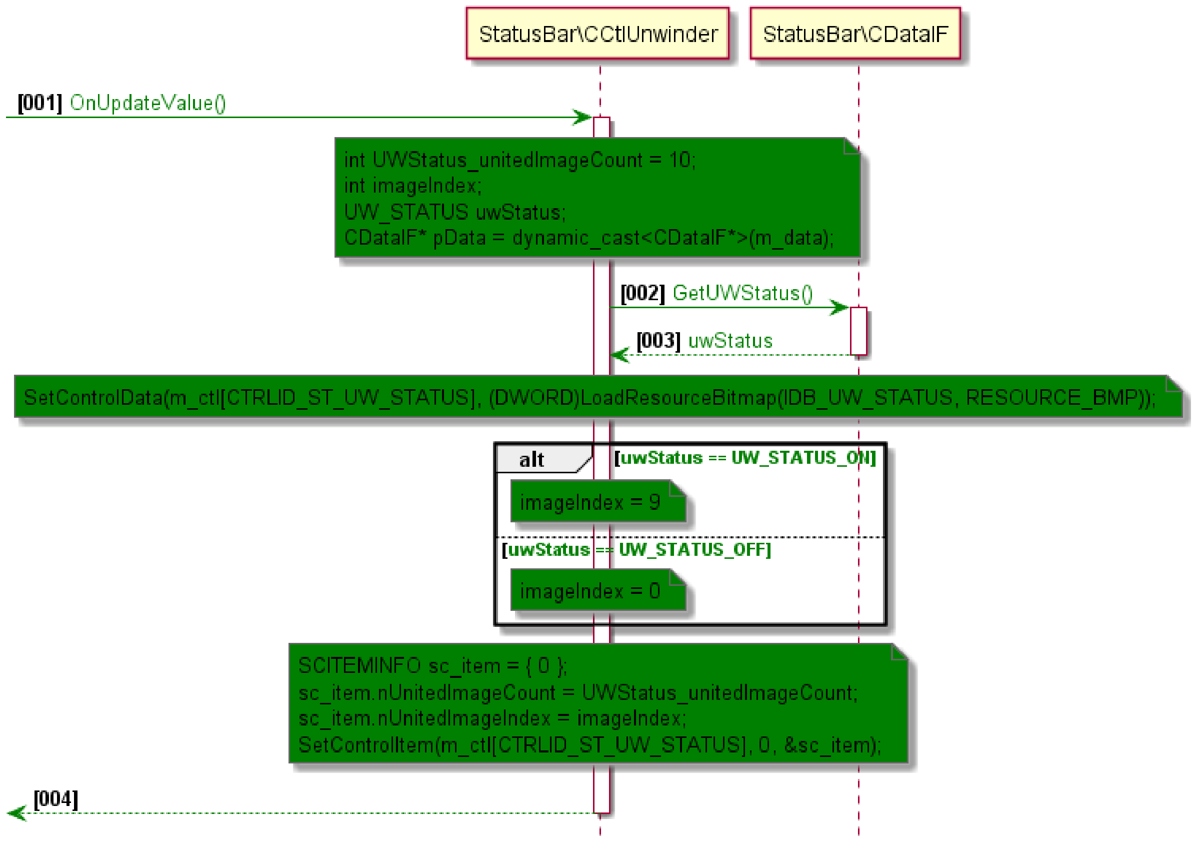
205.18 Set the unwinder control's property



```
CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
HBITMAP unwinder_bmp = LoadResourceBitmap(IDB_UW_STATUS, RESOURCE_BMP);
//Calculate the controls' position
BITMAP unwinder;
::GetObject(unwinder_bmp, sizeof(BITMAP), &unwinder);
int unwinder_w = unwinder.bmWidth;
int unwinder_left = DEF_W_SCREEN - unwinder_w - 10;
```

```
{
    long ctlId = CTRLID_ST_UW_STATUS;
    m_ctlAttribute[ctlId].type = CT_STATICBOX;
    m_ctlAttribute[ctlId].style = CST_HIDE | SCST_NORMAL | SCST_BITMAP | SCST_UNITED_IMAGE | SCST_IMAGE_BLEND;
    m_ctlAttribute[ctlId].text = NULL;
    //SetRect(&m_ctlAttribute[ctlId].rect, DEF_W_SCREEN - DEF_W_CONTROL - 10, 0, DEF_W_SCREEN - 10, 0 + DEF_H_CONTROL);
    SetRect(&m_ctlAttribute[ctlId].rect, unwinder_left, 7, unwinder_left + unwinder_w, 7 + DEF_H_CONTROL);
    m_ctlAttribute[ctlId].param = NULL;
}

{
    long ctlId = CTRLID_ST_PAPER_REMAINING;
    m_ctlAttribute[ctlId].type = CT_STATICBOX;
    m_ctlAttribute[ctlId].style = CST_HIDE | SCST_NORMAL | SCST_TEXT | SCST_CENTER;
    m_ctlAttribute[ctlId].text = NULL;
    SetRect(&m_ctlAttribute[ctlId].rect, 7, DEF_H_CONTROL - 37, 7 + 30, DEF_H_CONTROL - 37 + 20);
    m_ctlAttribute[ctlId].param = NULL;
    m_ctlAttribute[ctlId].ownerID = CTRLID_ST_UW_STATUS;
}
```

205.19 Update unwinder control display value



```
int UWStatus_unitedImageCount = 10;
int imageIndex;
UW_STATUS uwStatus;
CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
```

```
SetControlData(m_ctl[CTRLID_ST_UW_STATUS], (DWORD)LoadResourceBitmap(IDB_UW_STATUS, RESOURCE_BMP));
```

```
imageIndex = 9
```

```
imageIndex = 0
```

```
SCITEMINFO sc_item = { 0 };
sc_item.nUnitedImageCount = UWStatus_unitedImageCount;
sc_item.nUnitedImageIndex = imageIndex;
SetControlItem(m_ctl[CTRLID_ST_UW_STATUS], 0, &sc_item);
```

206. Processing according to UW status

1. Description

If there is no response from the UW for the notification channel opening result after the controller registers the status monitoring channel, it is determined that the UW has not started.

Or, if there is no status notification from the UW at the interval specified by the controller when registering the status monitoring channel, it is determined that the UW has not started.

206-1. If UW is not started when the controller is started

・Display a warning icon on the status bar.
(Ja) UWが起動していません。UWを起動してください。
(En) A communication error with UW has occurred.
・Display the translucent UW icon.
(※These icons will be added and updated in relation to 205-4)

206-2. When UW ends while the controller is starting

・The following warning message dialog is displayed.
(Ja)UWとの通信エラーが発生しました。
(En) A communication error with UW has occurred.
・Display the translucent UW icon.
(※These icons will be added and updated in relation to 205-4)

206-3. When UW starts while the controller is starting

・Set the print condition information of the current print condition and register the channel for paper information notification.
・Cancel the 206-1 warning icon display.
・Switch from the translucent UW icon of the UW icon to the normal UW icon.
(※These icon will be added and updated in relation to 205-4)

2. Solution

- Add resource into strings_UnwinderManager.ini file to display UW status warning dialog

Resource\English\strings_UnwinderManager.ini

```
[MSG]
IDS_NOTIFY_UW_STATUS = A communication error with UW has occurred.
```

Resource\Japanese\strings_UnwinderManager.ini

```
[MSG]
IDS_NOTIFY_UW_STATUS = UW との通信エラーが発生しました。
```

- In class CDataIF, create member variables and method to handle display the UW status.

Src\UnwinderManagerDataIF.h

```
class CDataIF : public CBaseDataIF,
public CMakeComposeUnwinderData
{
public:
    /...

    //methods
    void UpdateDisplayUWStatus(bool inUWstatus);

protected:

    // members
    bool m_displayWarningDialogEnable;
}
```

- In file Common\CommonUiMsg_OP.h, add new messages in enum for UW status

Common\CommonUiMsg_OP.h

```
// before
enum
{

    //...
    UWM_ADJUSTMENTGUI_CLOSE_DIALOG,         //!< Adjustment screen (Alignment/Shading/HeadCleaning) cl

    //
    UWM_OP_MAX_COUNT
};

// after
enum
{

    //...
    UWM_ADJUSTMENTGUI_CLOSE_DIALOG,         //!< Adjustment screen (Alignment/Shading/HeadCleaning) cl

    UWM_STATUSBAR_UW_STATUS_ON,             //!< UW status is online, then notify to display normal UW
    UWM_STATUSBAR_UW_STATUS_OFF,            //!< UW status is offine, then notify to display transluce

    //
    UWM_OP_MAX_COUNT
};
```

- In method CDataIF::UpdateDisplayUWStatus(), post a message about UW status and display the warning dialog when UW is offline

- In method CRequestUnwinderThread::CheckReceiverRunning(), if UWandRW_Receiver.exe is not run, call method UpdateDisplayUWStatus(false) to display UW is offline

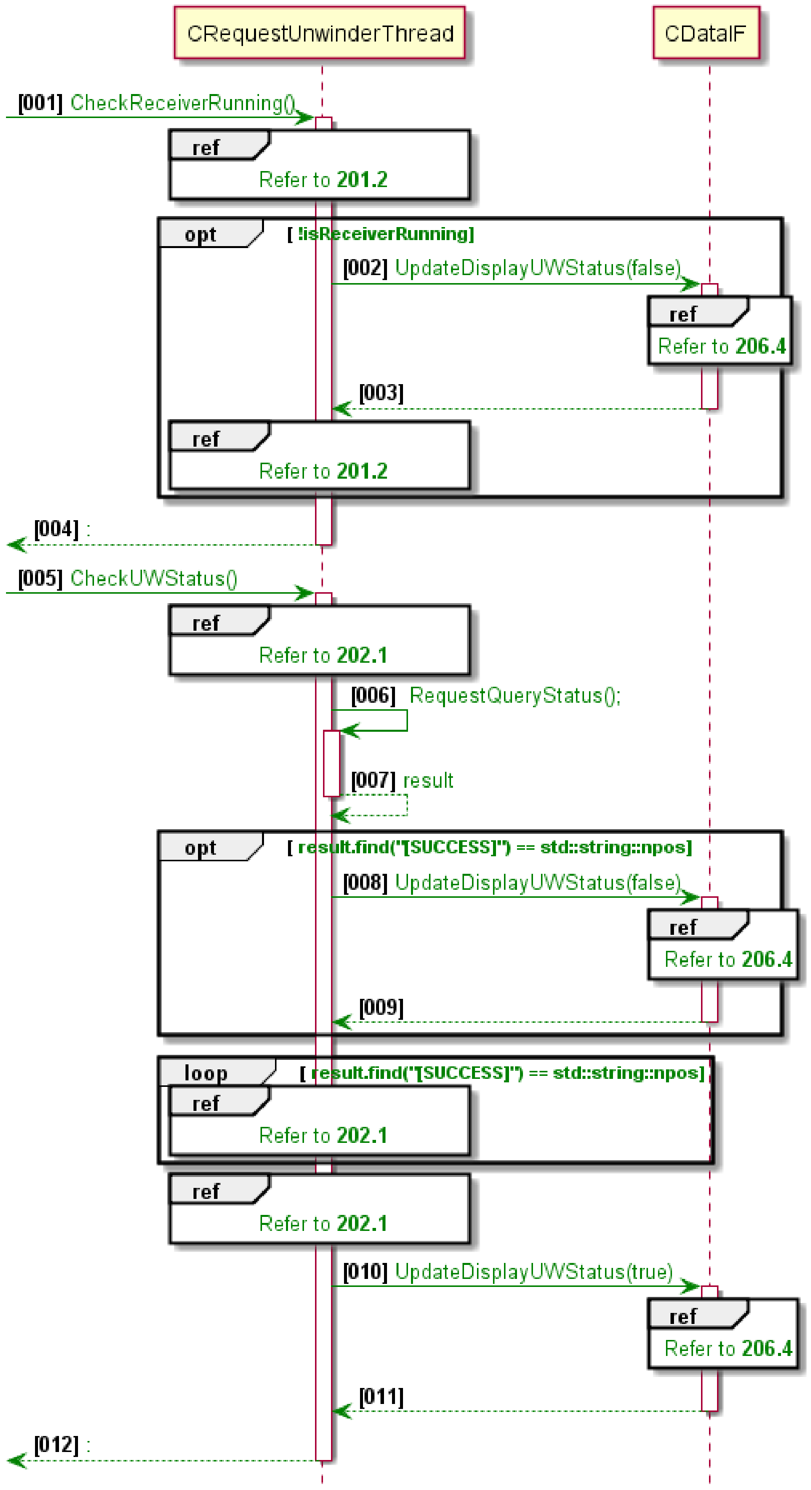
- In method CRequestUnwinderThread::CheckUWStatus(),
  - If receive the response from UW successfully, call method UpdateDisplayUWStatus(true) to display UW is online
  - Else, call method UpdateDisplayUWStatus(false) to display UW is offline
- In method CReceiveSignalStatus::ReceiveStatusInfo:
  - Stop the current timeout timer and start a new one.
  - Set the UW status into UnwinderManager_work.ini file.
  - call method UpdateDisplayUWStatus(true) to display UW is online
- In method OnStatusTimer(UINT_PTR nIDEvent):
  - When the timer for receiving the UW status timeout, call method UpdateDisplayUWStatus(false) to display UW is offline
  - Stop the current timer.
  - call CRequestUnwinderThread::MsgNotifyNoStatus().
- Add method CRequestUnwinderThread::MsgNotifyNoStatus() to notify main thread.

UnwinderManager\RequestUnwinderThread.h

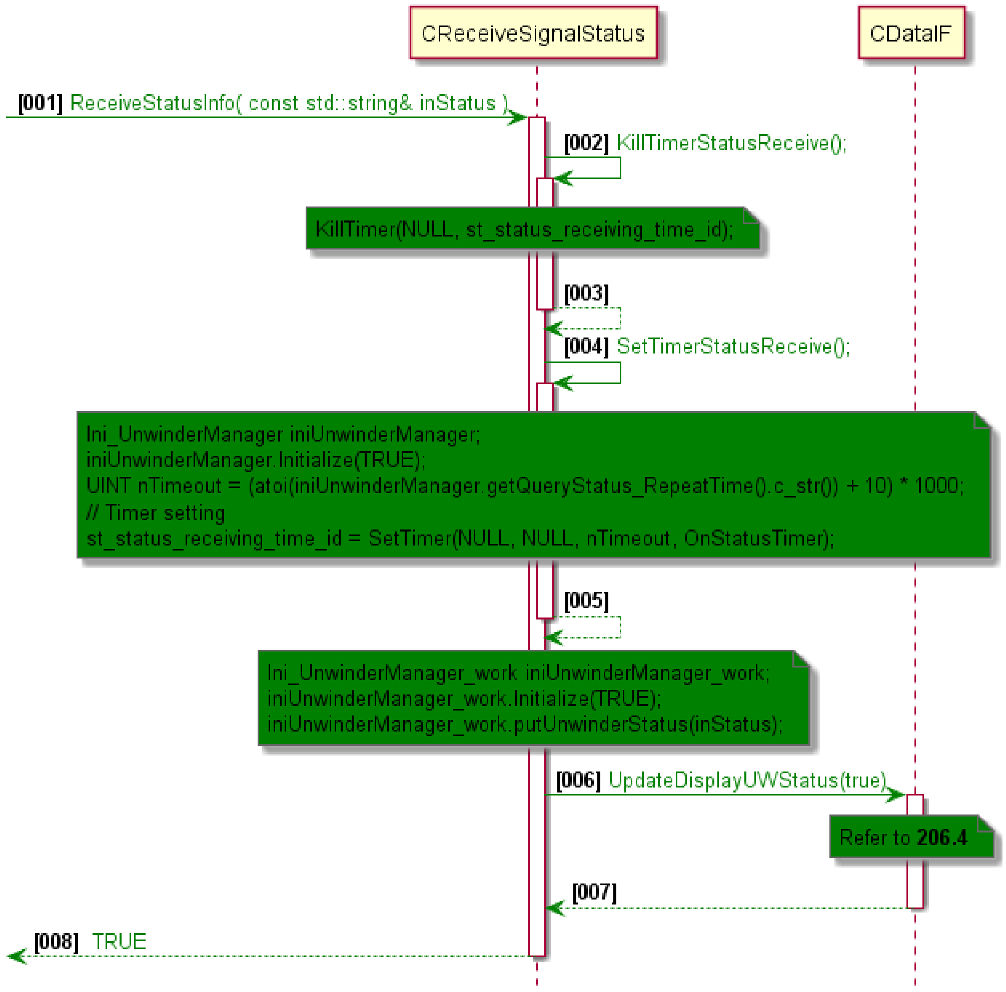
```
class CRequestUnwinderThread
{
public:
...
void MsgNotifyNoStatus();
...
}
```

3. Detail implementation

206.1 Check UW status when starting control



206.2 Receive status info

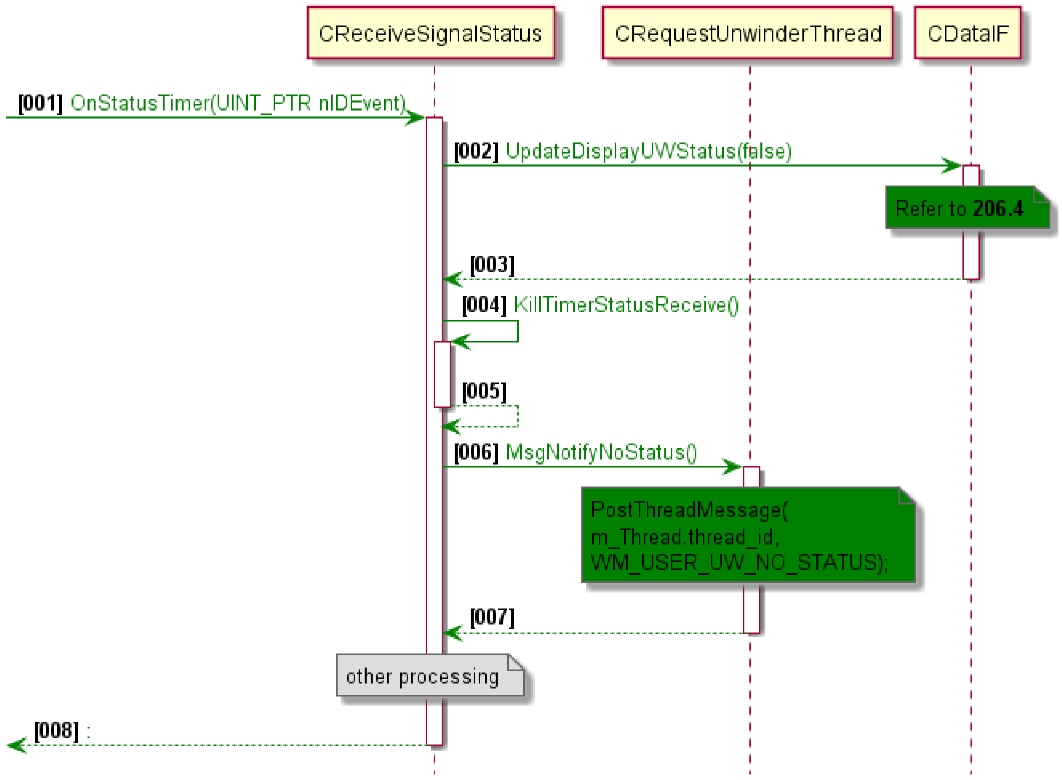

CReceiveSignalStatus
CDataIF
[001] ReceiveStatusInfo(const std::string& inStatus)
[002] KillTimerStatusReceive();
[003]
[004] SetTimerStatusReceive();
[005]
[006] UpdateDisplayUWStatus(true)
Refer to 206.4
[007]
[008] TRUE

```
KillTimer(NULL, st_status_receiving_time_id);
```

```
Ini_UnwinderManager iniUnwinderManager;
iniUnwinderManager.Initialize(TRUE);
UINT nTimeout = (atoi(iniUnwinderManager.getQueryStatus_RepeatTime().c_str()) + 10) * 1000;
// Timer setting
st_status_receiving_time_id = SetTimer(NULL, NULL, nTimeout, OnStatusTimer);
```

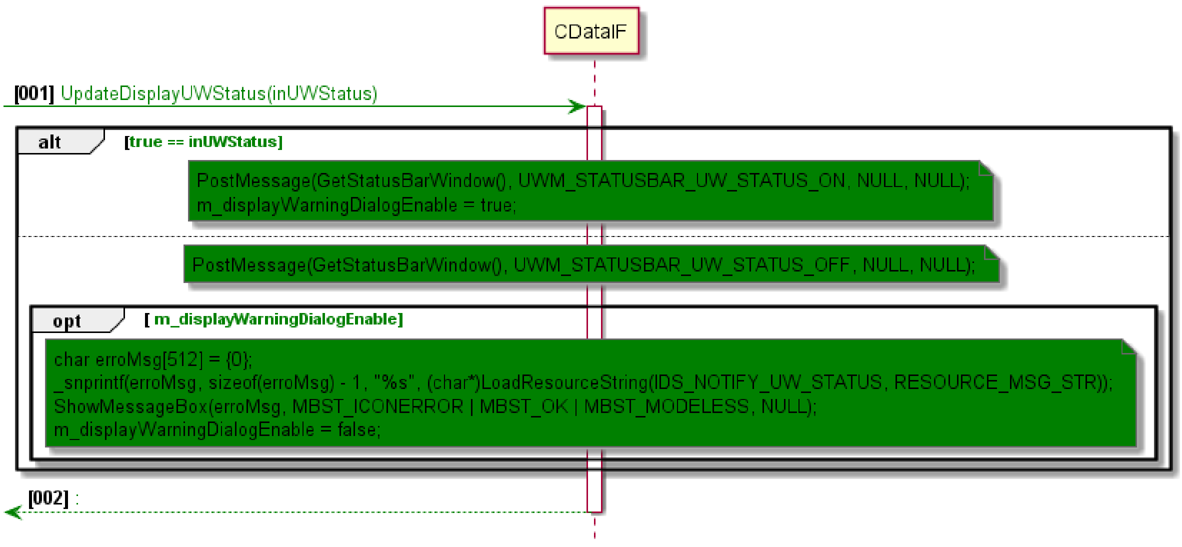
```
Ini_UnwinderManager_work iniUnwinderManager_work;
iniUnwinderManager_work.Initialize(TRUE);
iniUnwinderManager_work.putUnwinderStatus(inStatus);
```

206.3 Handle when receive UW status is timeout


CReceiveSignalStatus
CRequestUnwinderThread
CDataIF
[001] OnStatusTimer(UINT_PTR nIDEvent)
[002] UpdateDisplayUWStatus(false)
Refer to 206.4
[003]
[004] KillTimerStatusReceive()
[005]
[006] MsgNotifyNoStatus()
[007]
other processing
[008] :

```
PostThreadMessage(
m_Thread.thread_id,
WM_USER_UW_NO_STATUS);
```

206.4 Update display the UW status (in plugin UnwinderManager)


CDataIF
[001] UpdateDisplayUWStatus(inUWStatus)
alt
[true == inUWStatus]
opt
[m_displayWarningDialogEnable]
[002] :

```
PostMessage(GetStatusBarWindow(), UWM_STATUSBAR_UW_STATUS_ON, NULL, NULL);
m_displayWarningDialogEnable = true;
```

```
PostMessage(GetStatusBarWindow(), UWM_STATUSBAR_UW_STATUS_OFF, NULL, NULL);
```

```
char erroMsg[512] = {0};
_snprintf(erroMsg, sizeof(erroMsg) - 1, "%s", (char*)LoadResourceString(IDS_NOTIFY_UW_STATUS, RESOURCE_MSG_STR));
ShowMessageBox(erroMsg, MBST_ICONERROR | MBST_OK | MBST_MODELESS, NULL);
m_displayWarningDialogEnable = false;
```

207. Delete channel

1. Description

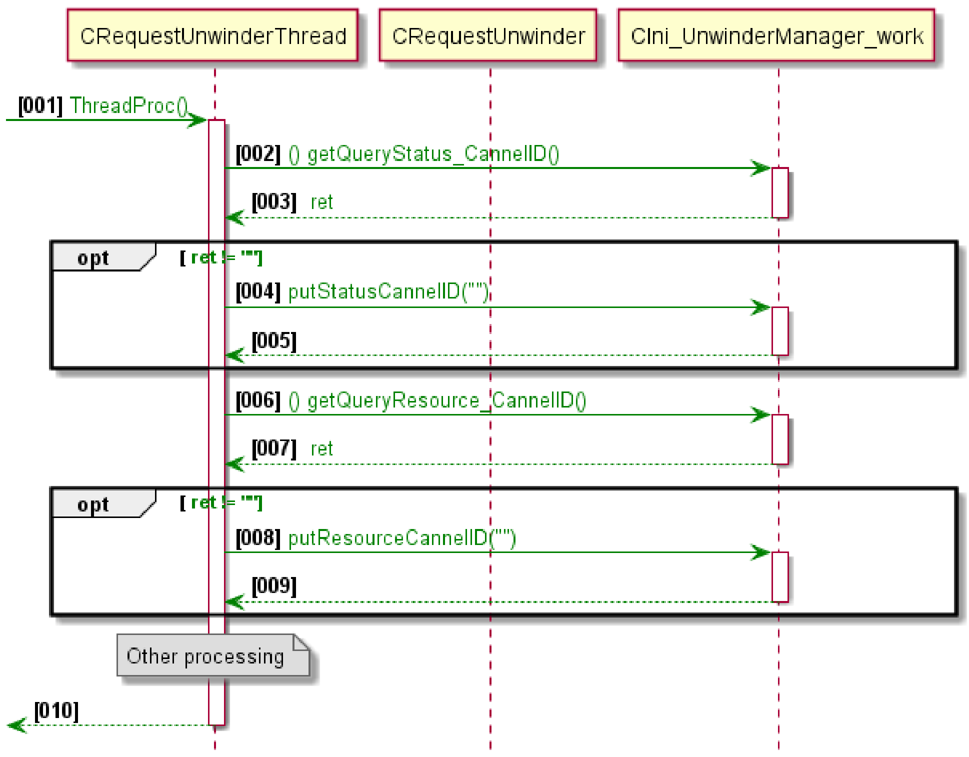
The communication channel is deleted by the channel ID notified in the response at the time of channel registration at the following timing.
- When channel information is already registered in the TP-UW_Communication_work.ini file at the time of channel registration(Described in 202).
- When Signal Status notifications for the status notification channel or paper information notification channel can no longer be received (at this time, a reconnection request is required from the controller (described later in 208)).
- When StopPresChParams is sent (sent when the controller ends)

2. Solution

- In CRequestUnwinderThread::ThreadProc(), check for existing channel IDs in TP-UW_Communication_work.ini and delete them.
- In CRequestUnwinderThread::CheckEvents(), when WM_USER_UW_NO_STATUS message is received, delete the channel IDs in TP-UW_Communication_work.ini.
- In CRequestUnwinderThread::ThreadProc(), wait for process “UWandRW_Receiver.exe” to end or m_ExitThread set. (see 201.1 Main flow)
- Call to RequestStopPersChParams() for channel which has been registered.
- If m_ExitThread not set yet, repeat the main loop. (see 201.1 Main flow)

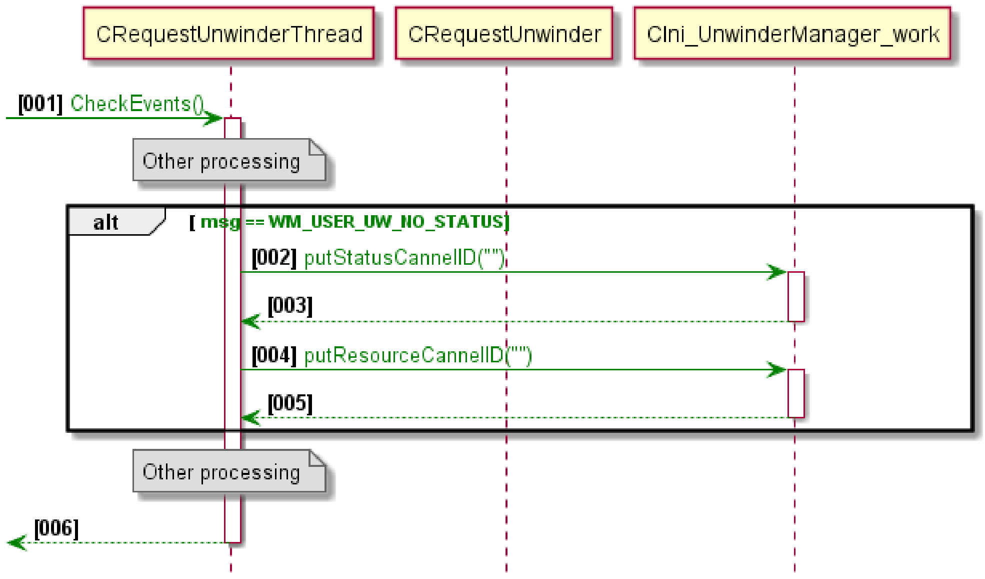
3. Detail implementation

207.1 Delete channels when controller starts

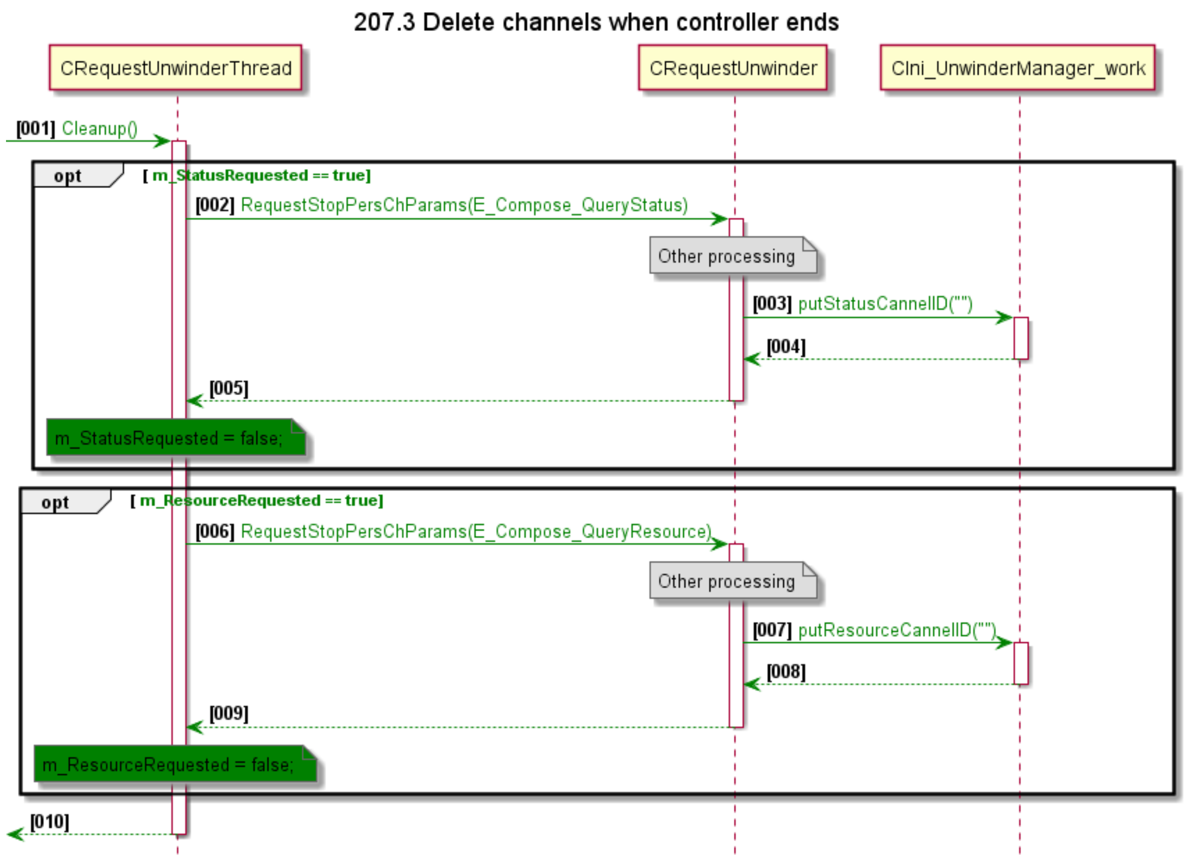


CRequestUnwinderThread
CRequestUnwinder
CIni_UnwinderManager_work
[001] ThreadProc()
[002] () getQueryStatus_CannelID()
[003] ret
opt
[ret != ""]
[004] putStatusCannelID("")
[005]
[006] () getQueryResource_CannelID()
[007] ret
opt
[ret != ""]
[008] putResourceCannelID("")
[009]
Other processing
[010]

207.2 Delete channels when not receiving signal status



CRequestUnwinderThread
CRequestUnwinder
CIni_UnwinderManager_work
[001] CheckEvents()
Other processing
alt
[msg == WM_USER_UW_NO_STATUS]
[002] putStatusCannelID("")
[003]
[004] putResourceCannelID("")
[005]
Other processing
[006]


207.3 Delete channels when controller ends
CRequestUnwinderThread
CRequestUnwinder
CIni_UnwinderManager_work
[001] Cleanup()
opt
[m_StatusRequested == true]
[002] RequestStopPersChParams(E_Compose_QueryStatus)
Other processing
[003] putStatusCannelID("")
[004]
[005]
m_StatusRequested = false;
opt
[m_ResourceRequested == true]
[006] RequestStopPersChParams(E_Compose_QueryResource)
Other processing
[007] putResourceCannelID("")
[008]
[009]
m_ResourceRequested = false;
[010]

208. Channel reconnection request

1. Description

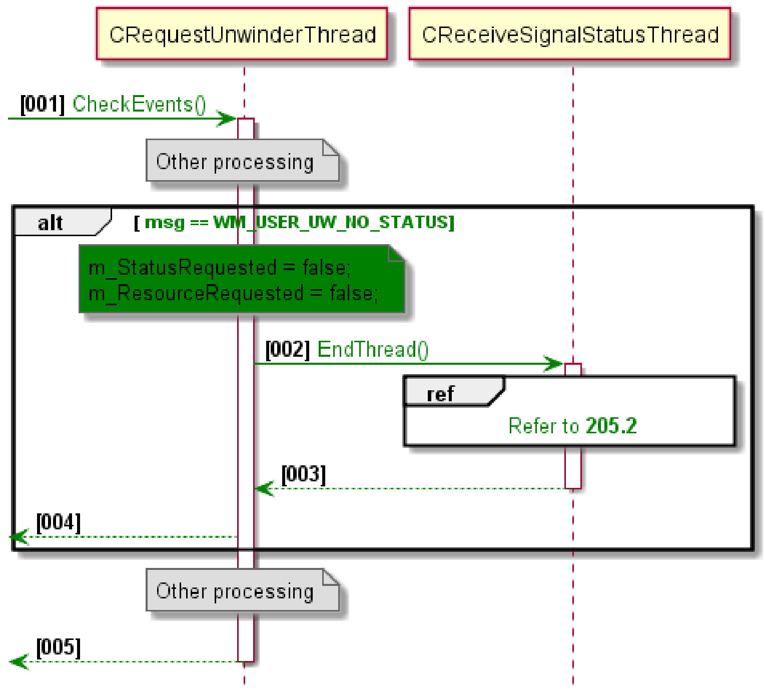
Issue a UWPing confirmation timer when SignalStatus notifications for the status notification channel or paper information notification channel can no longer be received.
If Ping passes, channel registration will be performed (until channel registration is possible).

2. Solution

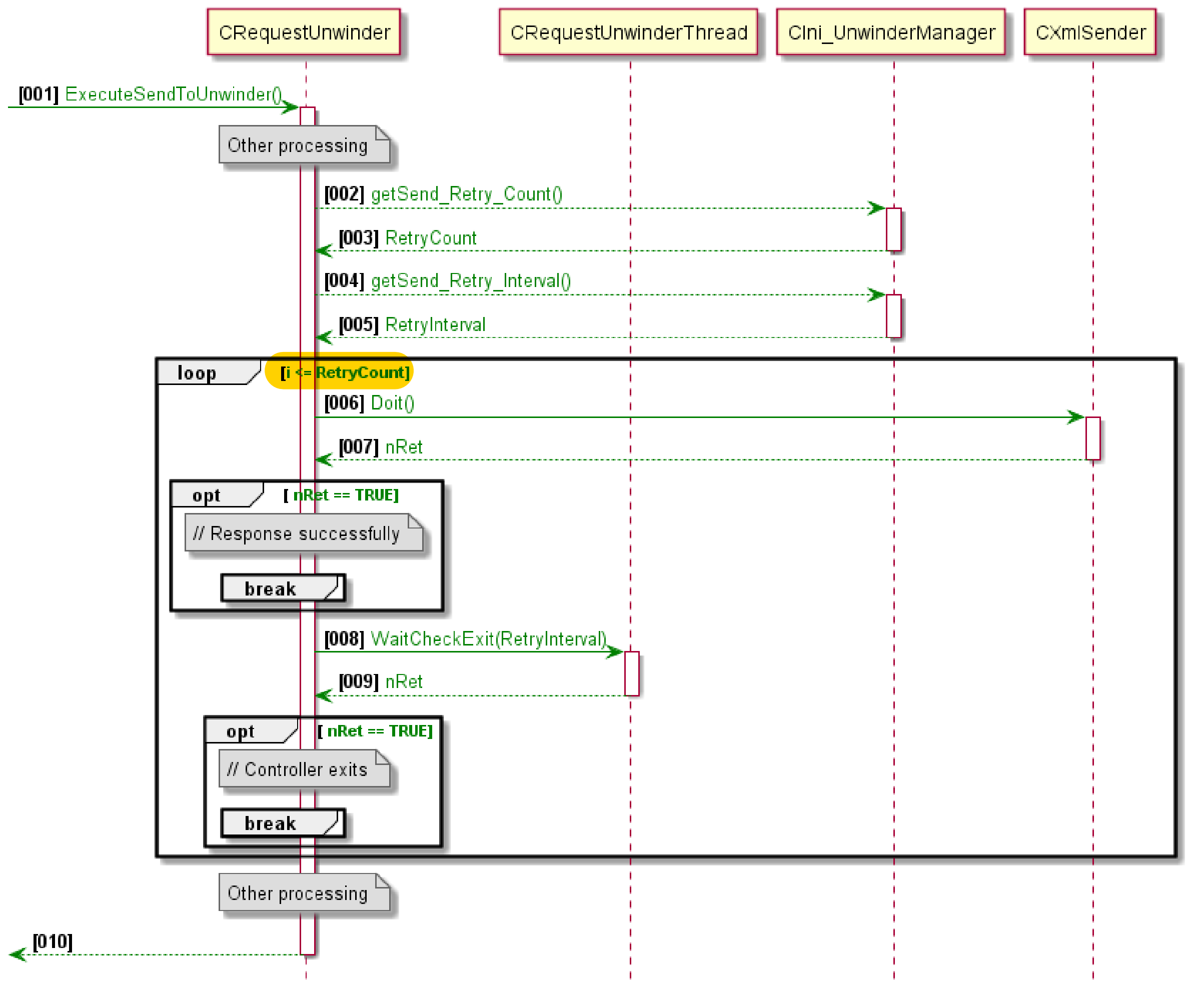
- In CRequestUnwinderThread::CheckEvents(), when WM_USER_UW_NO_STATUS message is received, end CReceiveSignalStatusThread and return, then the outer loop in CRequestUnwinderThread::ThreadProc() will repeat the registration process. (Refer to diagram 201.1)
- In function CRequestUnwinder::ExecuteSendToUnwinder(), get SEND_RETRY_COUNT and SEND_RETRY_INTERVAL values from TP-UW_Communication.ini.
- If sending XML fail, repeat until successfully or SEND_RETRY_COUNT times, each time waits SEND_RETRY_INTERVAL milliseconds.

3. Detail implementation

208.1 when SignalStatus is not received



208.2 Retry registration process


CRequestUnwinder
CRequestUnwinderThread
CIni_UnwinderManager
CXmlSender
[001] ExecuteSendToUnwinder()
Other processing
[002] getSend_Retry_Count()
[003] RetryCount
[004] getSend_Retry_Interval()
[005] RetryInterval
loop
[i <= RetryCount]
[006] Doit()
[007] nRet
opt
[nRet == TRUE]
// Response successfully
break
[008] WaitCheckExit(RetryInterval)
[009] nRet
opt
[nRet == TRUE]
// Controller exits
break
Other processing
[010]