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

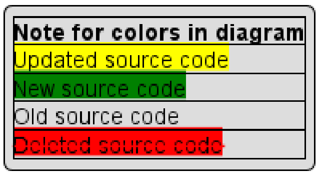
Change log

Rev.	Date	Author	Details
1	2021/12/20	GCS	Created
2	2021/12/31	GCS	Updated • Update solution for spec 205, 206, 207 • Add solution for spec 208 • Update diagram 202.1, 202.2, 202.3, 205.10, 206.1, 206.2, 206.3, 206.4, 207.3 • Add diagrams 205.9, 205.12, 205.14, 205.15, 205.16, 205.18, 205.19, 205.20, 207.1, 207.2, 208.1, 208.2 • Delete diagram 206.5
3	2022/01/12	GCS	Updated • Update solution for spec 202, 205, 206, 207, 208 • Update diagram 201.1, 202.1, 202.2, 202.3, 203.2, 203.3, 205.3, 205.9, 205.10, 205.11, 205.12, 205.13, 205.14, 205.15, 205.16, 205.23, 206.1, 206.2, 206.3, 206.4, 207.2, 208.1, 208.2 • Add diagrams 205.17, 205.18, 205.19, 205.20, 205.24, 205.25, 205.26, 205.27, 206.5, 206.6, 206.7
4	2022/02/23	GCS	Updated • Update description for spec 203 • Update solution for spec 201, 203, 205 • Delete 205.19, 205.22, 205.23, 205.24, 205.25, 205.26 (the diagram numbers are in Rev3) • Update 201.2, 205.4, 205.5, 205.10, 205.14, 205.15, 205.16, 205.17, 206.4 • Add 203.8, 205.18, 205.19, 205.20, 205.24, 205.25, 205.26, 205.27, 205.28

Target System

[TP-J560XDn]Ver1.00_base_3.50

Note for diagrams



Note for colors in diagram
Updated source code
New source code
Old source code
Deleted source code

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。
```

- In CommonDef.h, add a new value to ENUM_ERR_CODE enum to display the warning ID in UnwinderManager plugin.

CommonCommonDef.h

```
//Before
ERR_RDPMONITOR  = 17700,        //!< RDPMonitor.dll 17700 - 17799(HD/NX)
ERR_PRINT_STOP  = 17800,        //!< PrintStop.dll 17800 - 17899(HD/NX)
//次は17900.

//After
ERR_RDPMONITOR  = 17700,        //!< RDPMonitor.dll 17700 - 17799(HD/NX)
ERR_PRINT_STOP  = 17800,        //!< PrintStop.dll 17800 - 17899(HD/NX)
ERR_UNWINDER_MANAGER = 18600,   //!< UnwinderManager.dll 18600-18699 (XDn)

//次は18700.
```

- 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

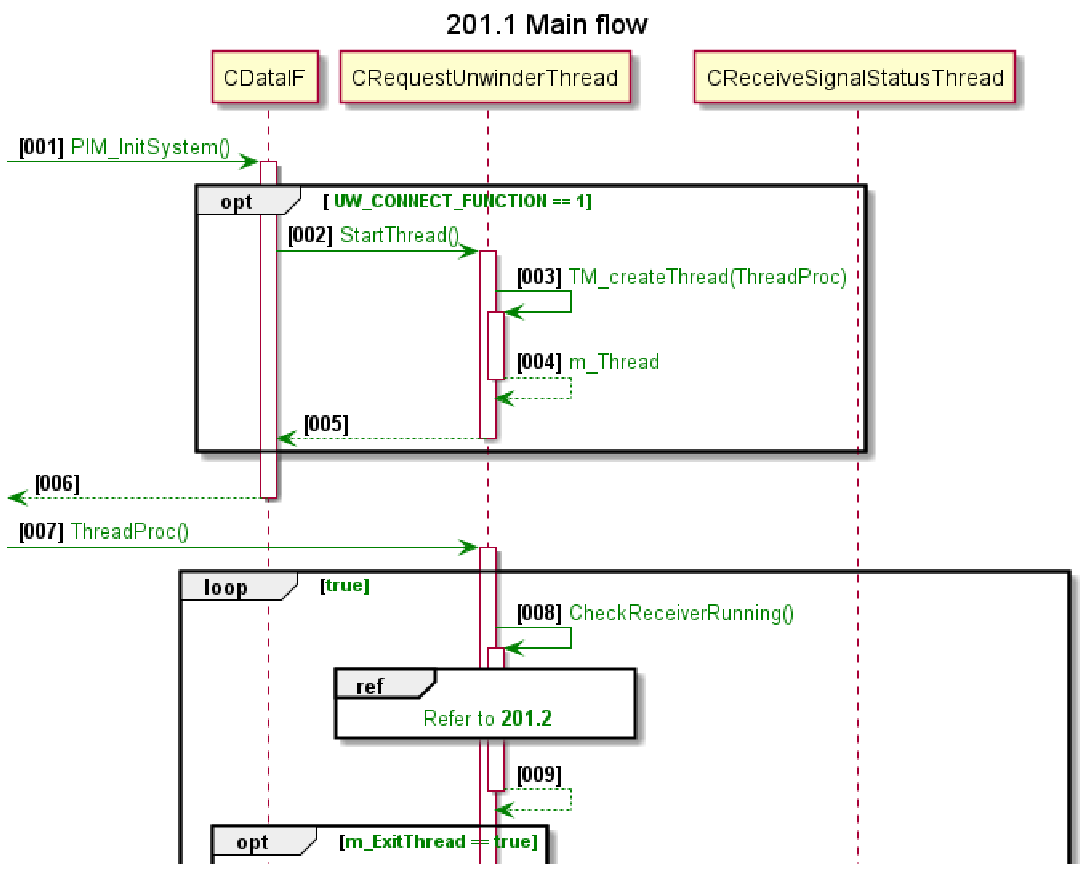
```
class CDataIF : public CBaseDataIF,
                public CMakeComposeUnwinderData
{
public:
    void Initialize();
    void Finalize();
    virtual BOOL PIM_InitSystem();
    virtual BOOL PIM_ExitSystem();
    ...
}
```

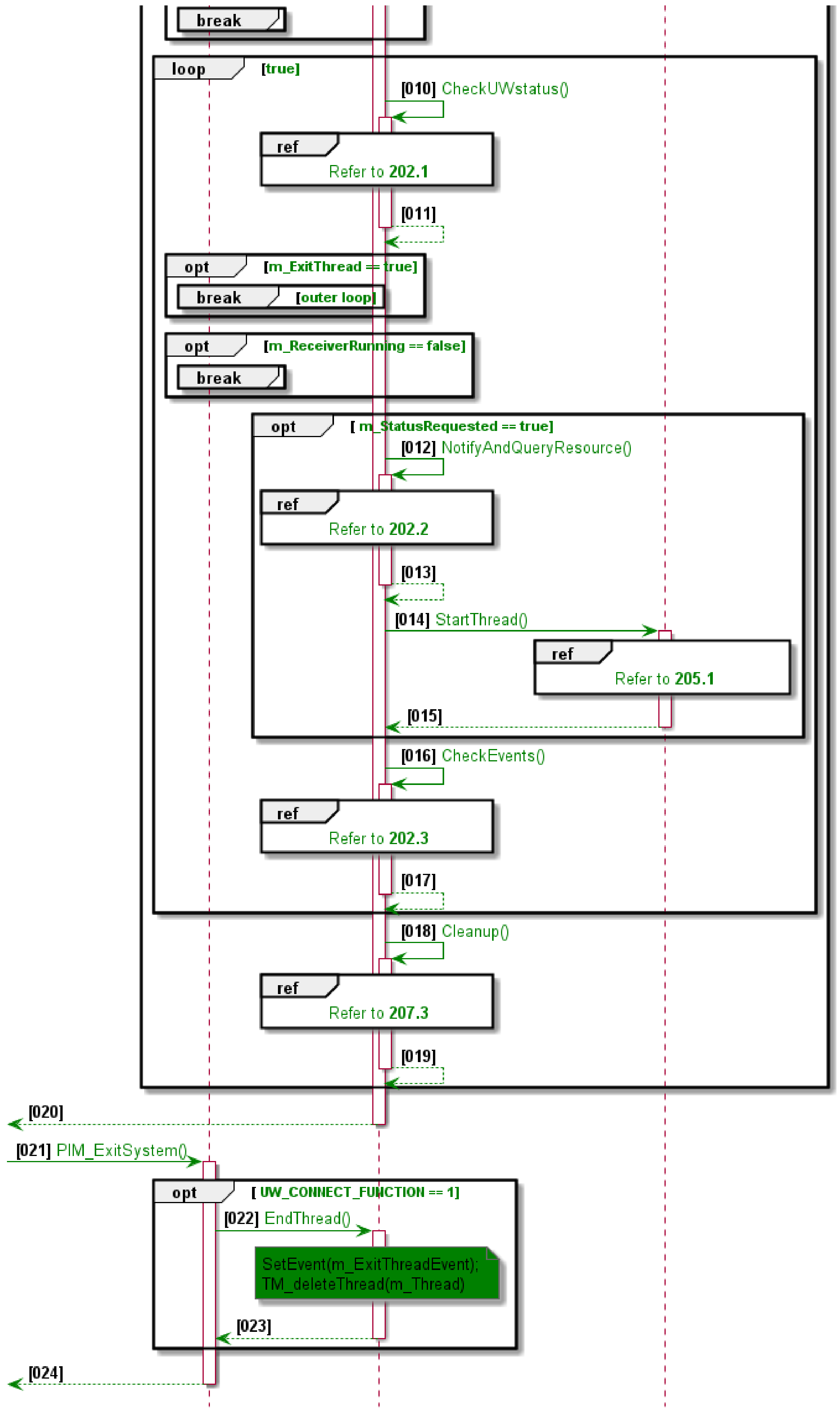
- Add class CRequestUnwinderThread() to run main process.
- At first, check for process “UWandRW_Receiver.exe” running, if not then display warning dialog.
- Add loop to wait until “UWandRW_Receiver.exe” running to continue process.

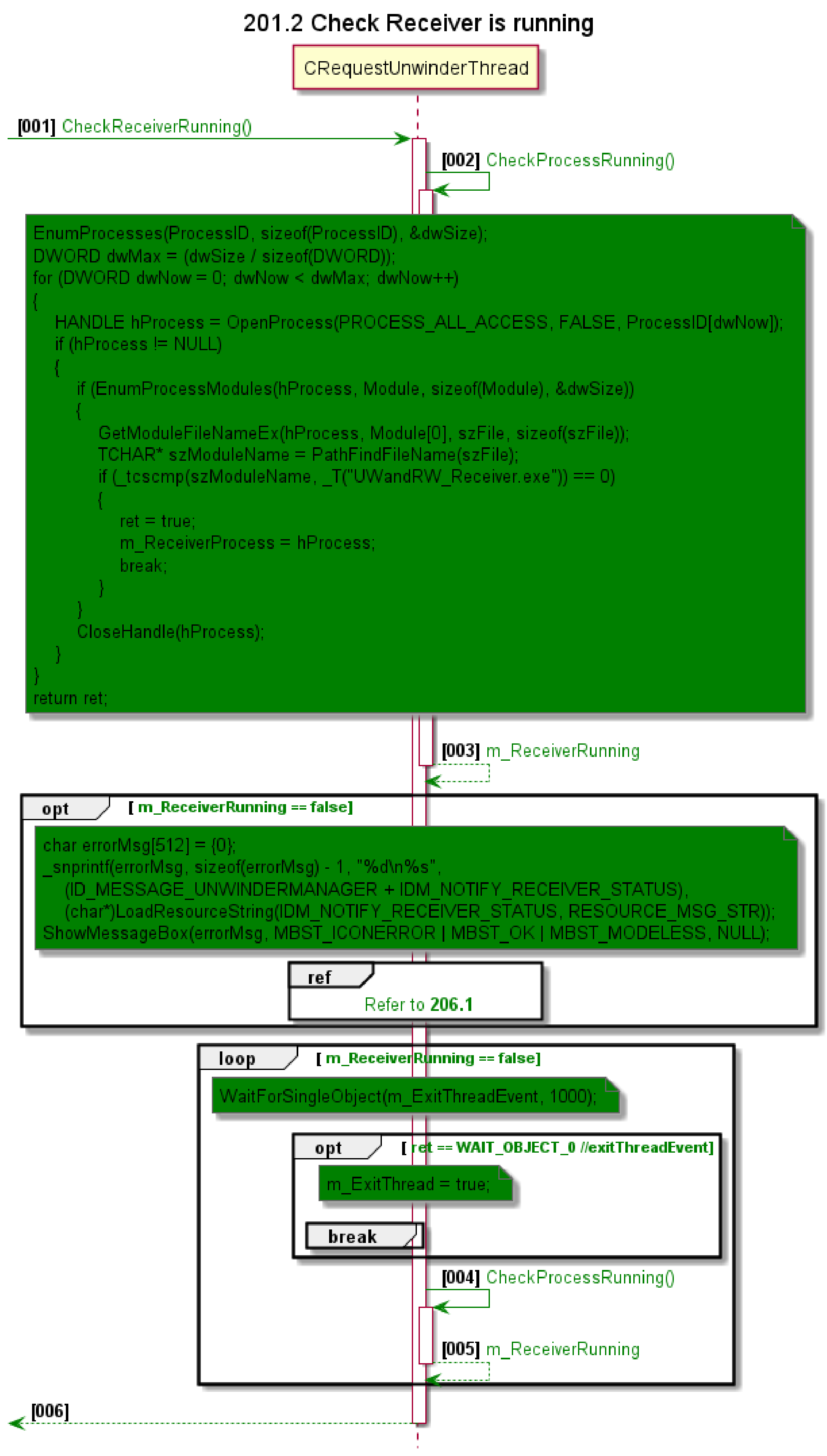
UnwinderManager\RequestUnwinderThread.h

```
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




break
loop [true]
[010] CheckUWstatus()
ref
Refer to 202.1
[011]
opt [m_ExitThread == true]
break [outer loop]
opt [m_ReceiverRunning == false]
break
opt [m_StatusRequested == true]
[012] NotifyAndQueryResource()
ref
Refer to 202.2
[013]
[014] StartThread()
ref
Refer to 205.1
[015]
[016] CheckEvents()
ref
Refer to 202.3
[017]
[018] Cleanup()
ref
Refer to 207.3
[019]
[020]
[021] PIM_ExitSystem()
opt [UW_CONNECT_FUNCTION == 1]
[022] EndThread()
SetEvent(m_ExitThreadEvent);
TM_deleteThread(m_Thread)
[023]
[024]


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, "%d\n%s",
    (ID_MESSAGE_UNWINDERMANAGER + IDM_NOTIFY_RECEIVER_STATUS),
    (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().
- 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

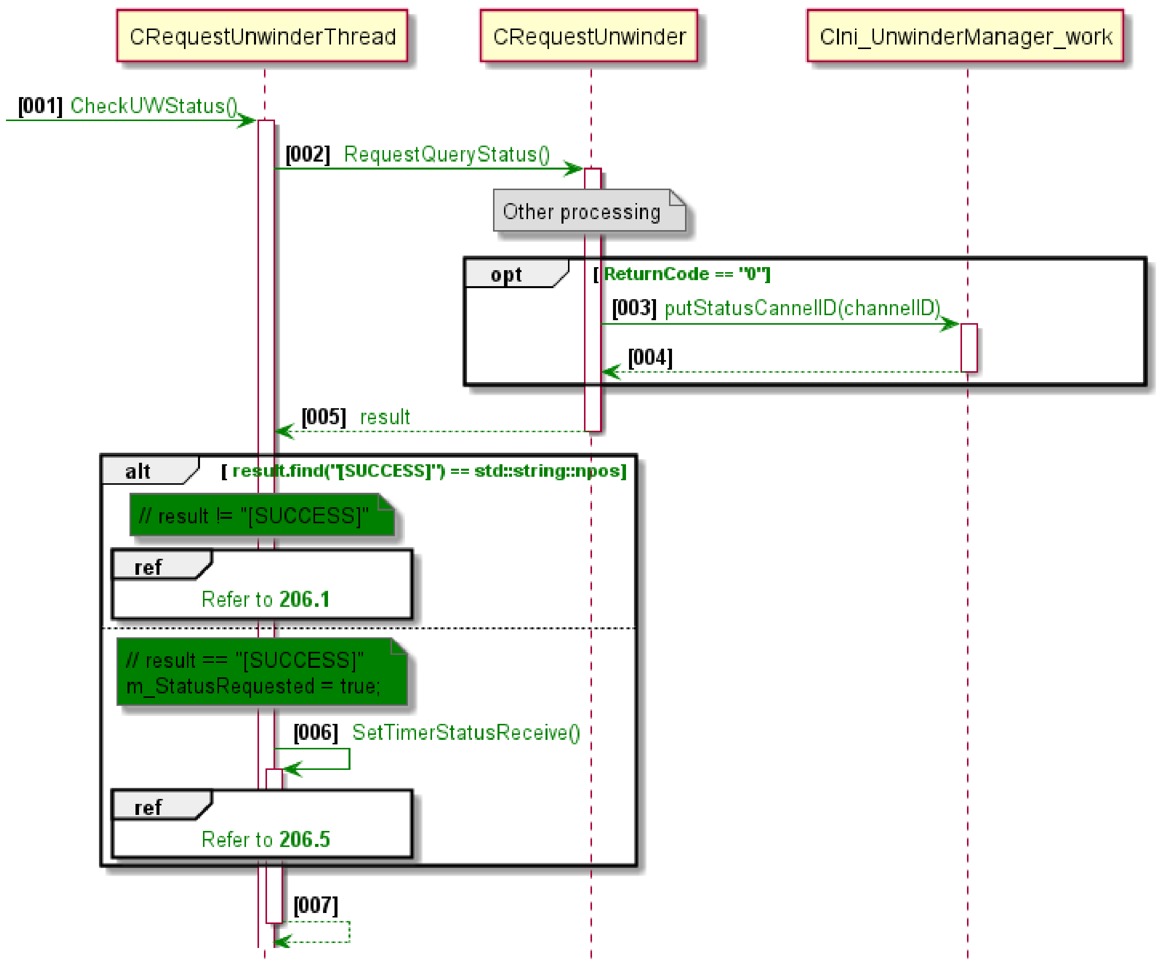
```
class CRequestUnwinderThread
{
private:
...
void CheckUWStatus();
void NotifyAndQueryResource(const std::string& inSectionId = "");
void CheckEvents();
...
bool m_StatusRequested;
bool m_ResouceRequested;
...
}
```

UnwinderManager\RequestUnwinderThread.cpp

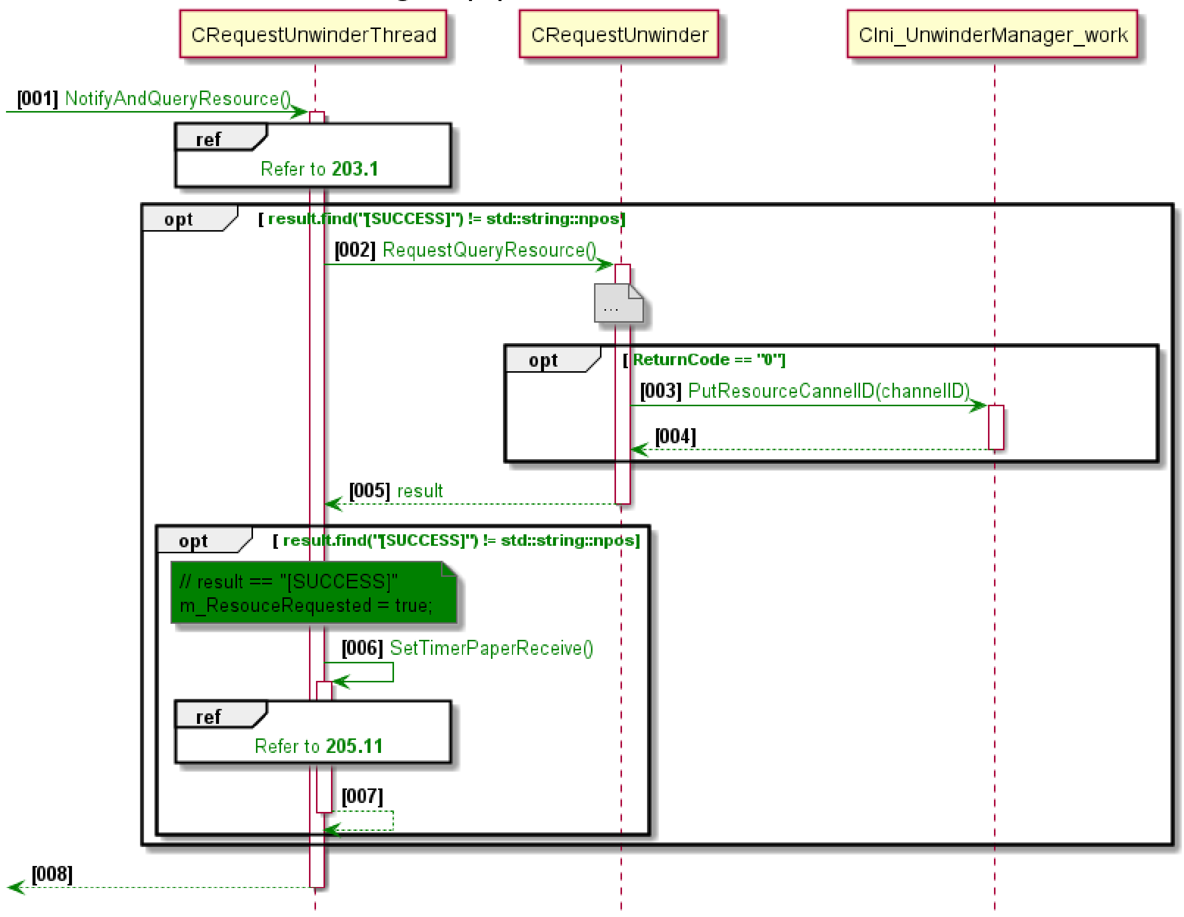
```
#define WM_USER_NOTIFY_QUERY_RESOURCE   WM_USER+102
```

3. Detail implementation

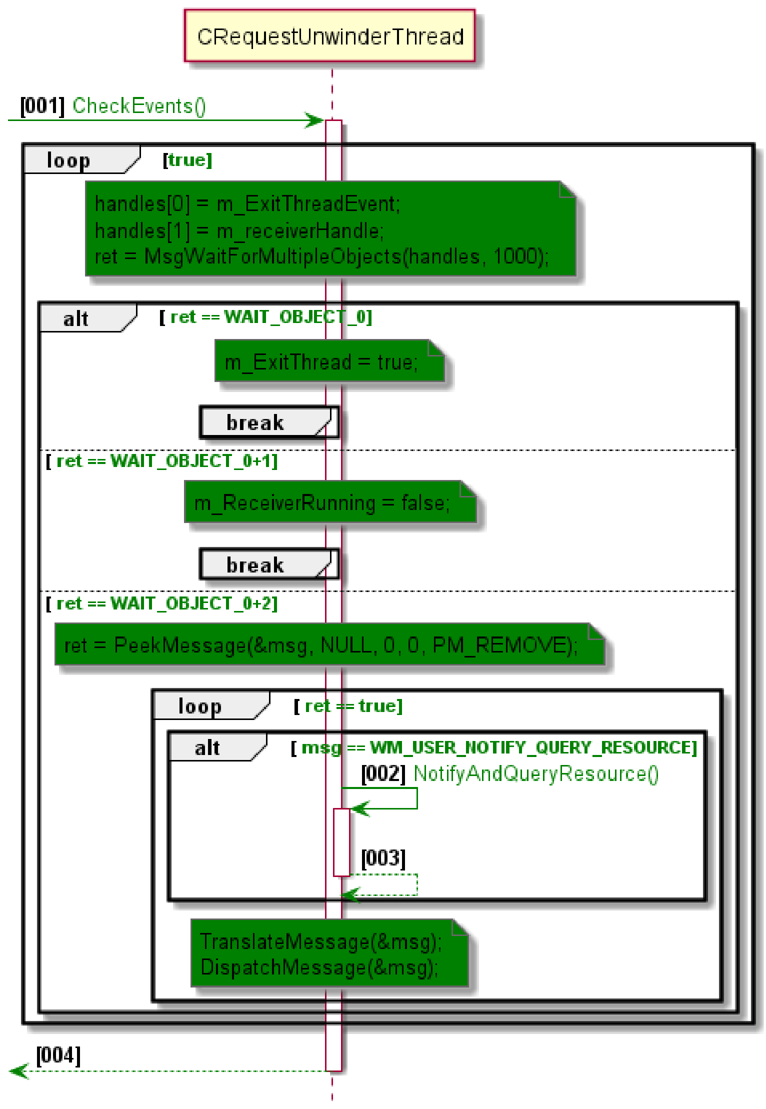
202.1 Register condition monitoring channel



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] SetTimerPaperReceive()
ref
Refer to 205.11
[007]
[008]

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_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)
5. When the job running status is released (Current print condition)

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)();
typedef void(*_OnEndJobRun)();
struct SUnwinderManager_Callbacks
{
    //Version 1
    DWORD                           StructVersion;
    _OnSetCurrentPrintCondition     OnSetCurrentPrintCondition;
    _OnUpdateCurrentPrintCondition  OnUpdateCurrentPrintCondition;
    _OnFirstJobRun                  OnFirstJobRun;

    //Version 2
    _OnEndJobRun                                        OnEndJobRun;
}
```

  UnwinderManager\UnwinderManager_OP.h

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

  UnwinderManager\Data_IF.h

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

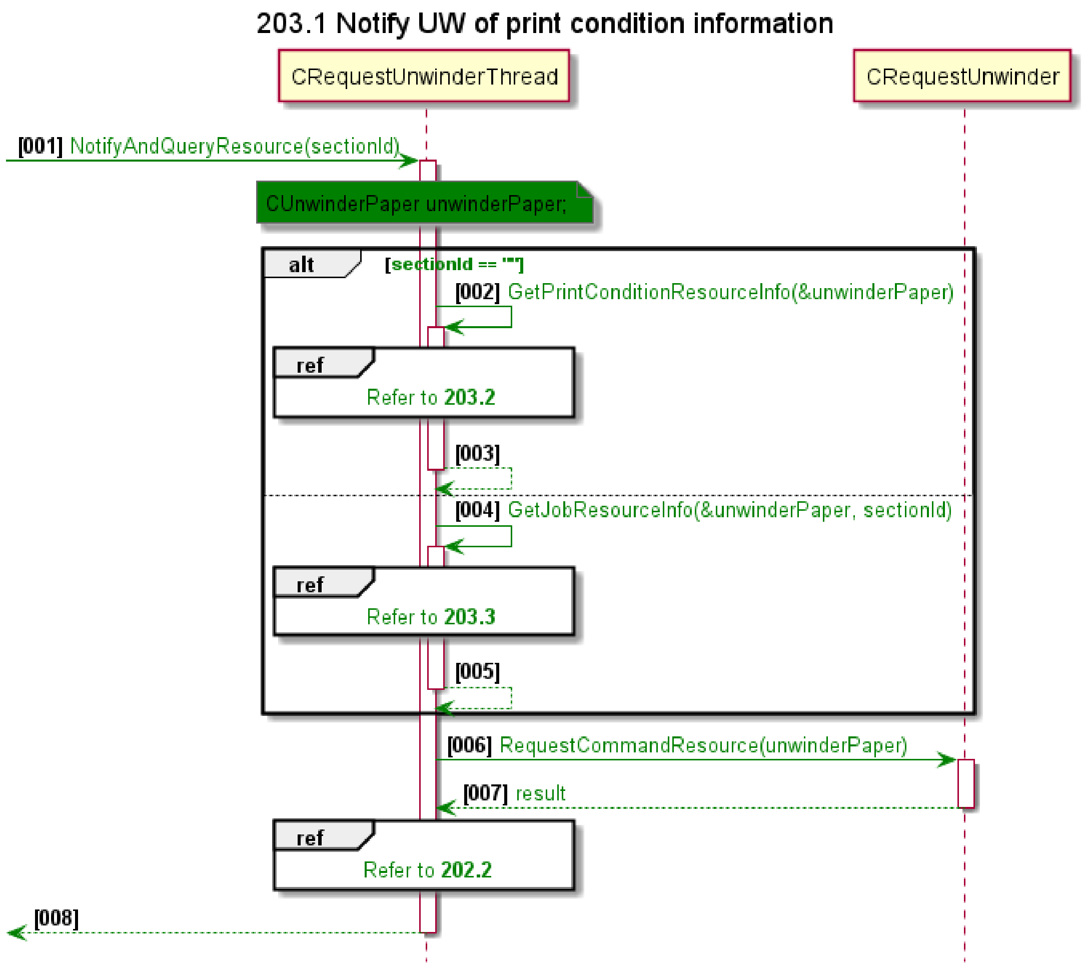
- Add function CRequestUnwinderThread::MsgNotifyAndQueryResource() to notify main thread of the event.

  UnwinderManager\RequestUnwinderThread.h

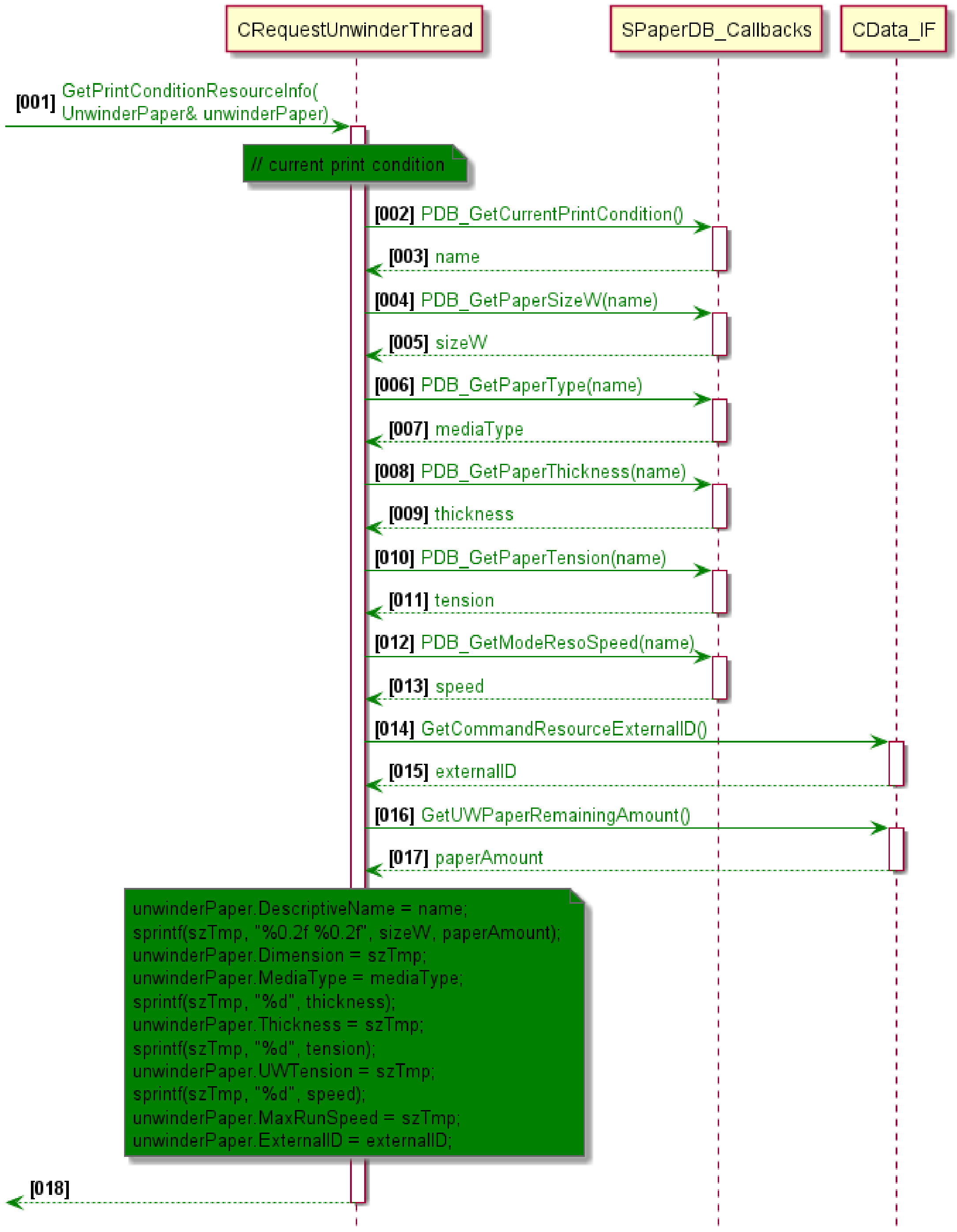
```
class CRequestUnwinderThread
{
public:
    ...
    void MsgNotifyAndQueryResource(const std::string& inSectionId = "");
    ...
}
```

- 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()
- Case 5 When the job running status is released:
  In plugin JobPrintSequence: call to SUnwinderManager_Callbacks::OnEndJobRun() in
  JobPrintManager::ProStartJobPrintSeq()

3. Detail implementation

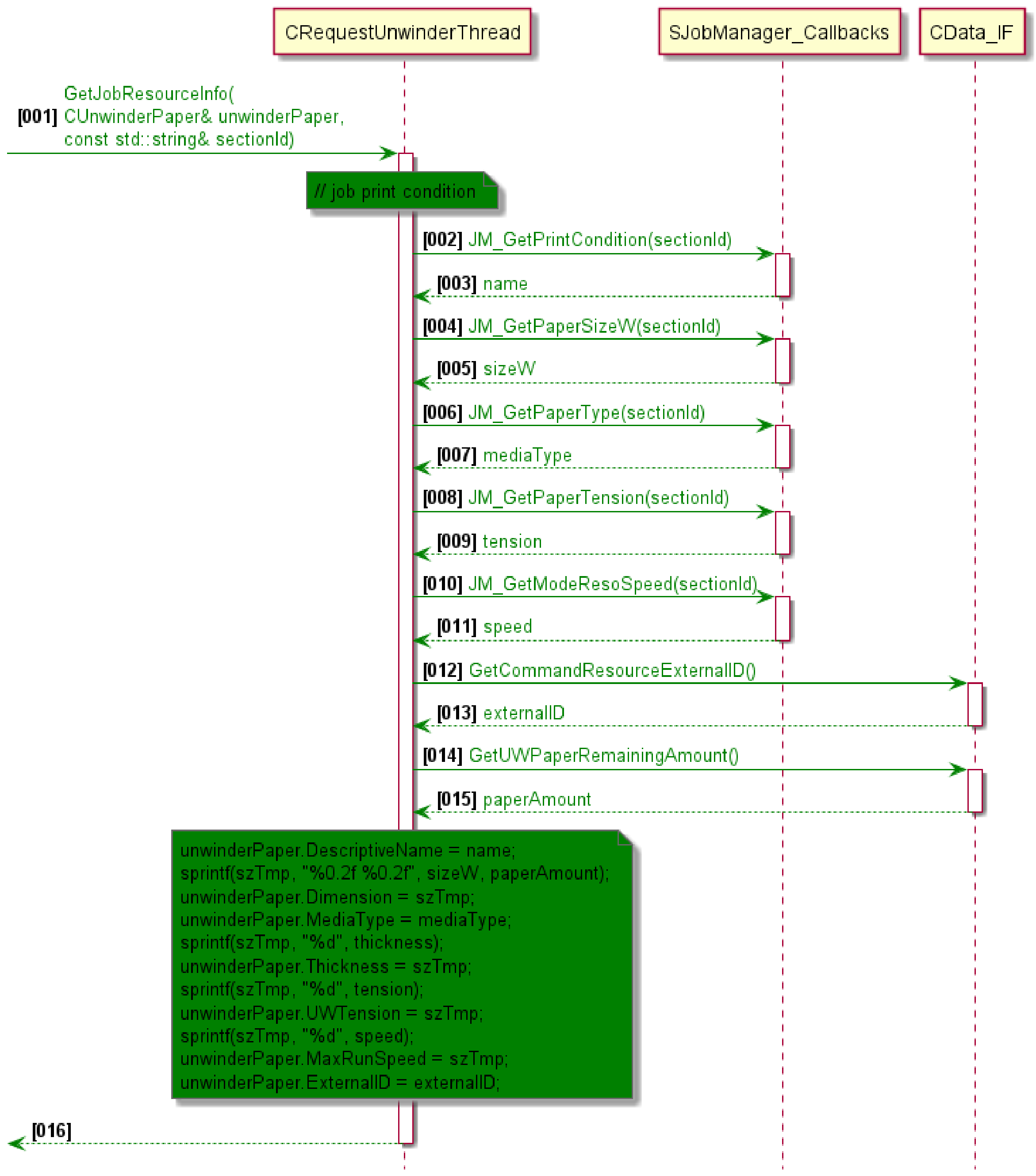

203.1 Notify UW of print condition information
CRequestUnwinderThread
CRequestUnwinder
[001] NotifyAndQueryResource(sectionId)
CUnwinderPaper unwinderPaper;
alt
[sectionId == ""]
[002] GetPrintConditionResourceInfo(&unwinderPaper)
ref
Refer to 203.2
[003]
[004] GetJobResourceInfo(&unwinderPaper, sectionId)
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
CData_IF
[001] GetPrintConditionResourceInfo(UnwinderPaper& unwinderPaper)
// current print condition
[002] PDB_GetCurrentPrintCondition()
[003] name
[004] PDB_GetPaperSizeW(name)
[005] sizeW
[006] PDB_GetPaperType(name)
[007] mediaType
[008] PDB_GetPaperThickness(name)
[009] thickness
[010] PDB_GetPaperTension(name)
[011] tension
[012] PDB_GetModeResoSpeed(name)
[013] speed
[014] GetCommandResourceExternalID()
[015] externalID
[016] GetUWPaperRemainingAmount()
[017] paperAmount
[018]

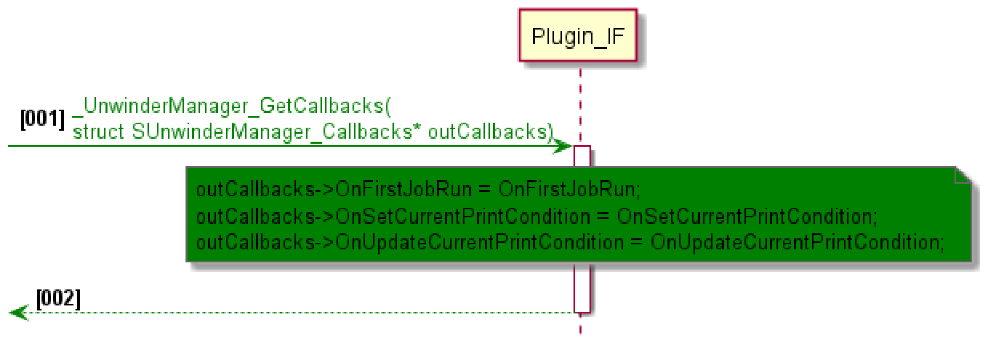
```
unwinderPaper.DescriptiveName = name;
sprintf(szTmp, "%0.2f %0.2f", sizeW, paperAmount);
unwinderPaper.Dimension = szTmp;
unwinderPaper.MediaType = mediaType;
sprintf(szTmp, "%d", thickness);
unwinderPaper.Thickness = szTmp;
sprintf(szTmp, "%d", tension);
unwinderPaper.UWTension = szTmp;
sprintf(szTmp, "%d", speed);
unwinderPaper.MaxRunSpeed = szTmp;
unwinderPaper.ExternalID = externalID;
```

203.3 Get job print condition information


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

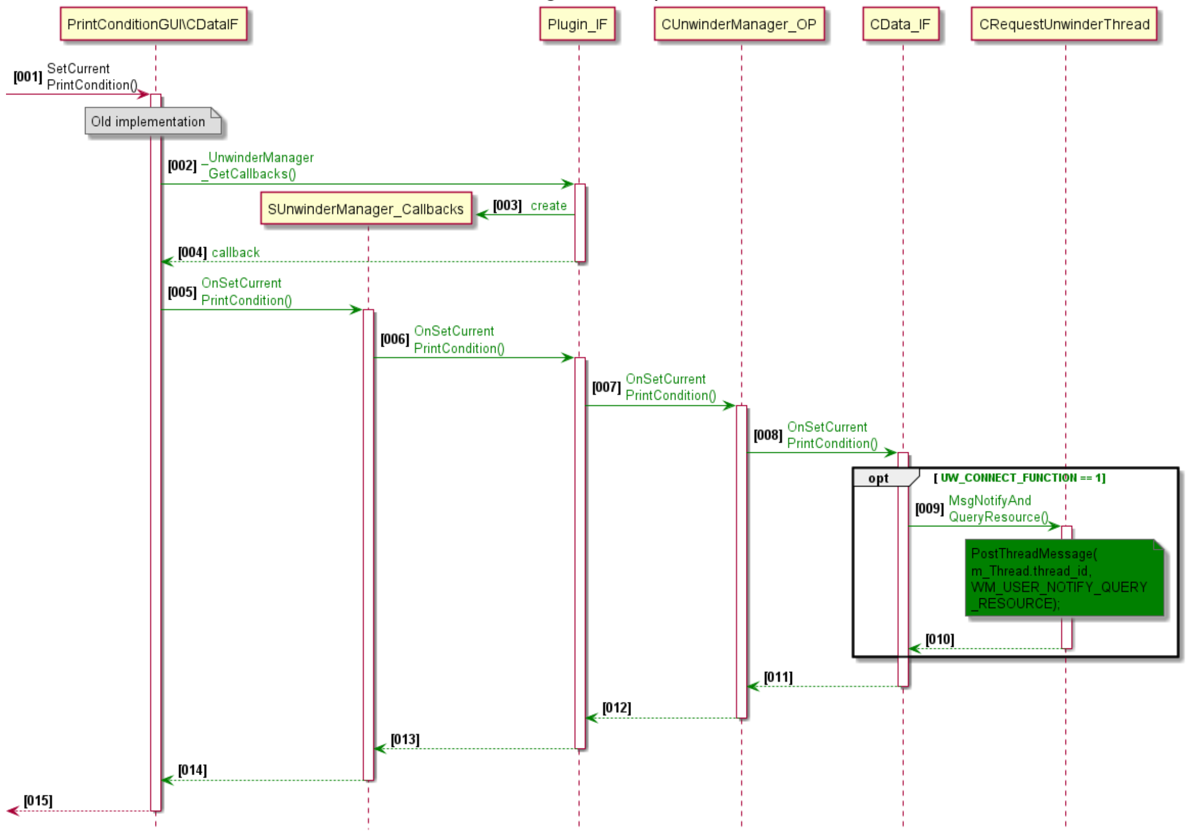
```
unwinderPaper.DescriptiveName = name;
sprintf(szTmp, "%0.2f %0.2f", sizeW, paperAmount);
unwinderPaper.Dimension = szTmp;
unwinderPaper.MediaType = mediaType;
sprintf(szTmp, "%d", thickness);
unwinderPaper.Thickness = szTmp;
sprintf(szTmp, "%d", tension);
unwinderPaper.UWTension = szTmp;
sprintf(szTmp, "%d", speed);
unwinderPaper.MaxRunSpeed = szTmp;
unwinderPaper.ExternalID = externalID;
```

203.4 Set callback functions


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

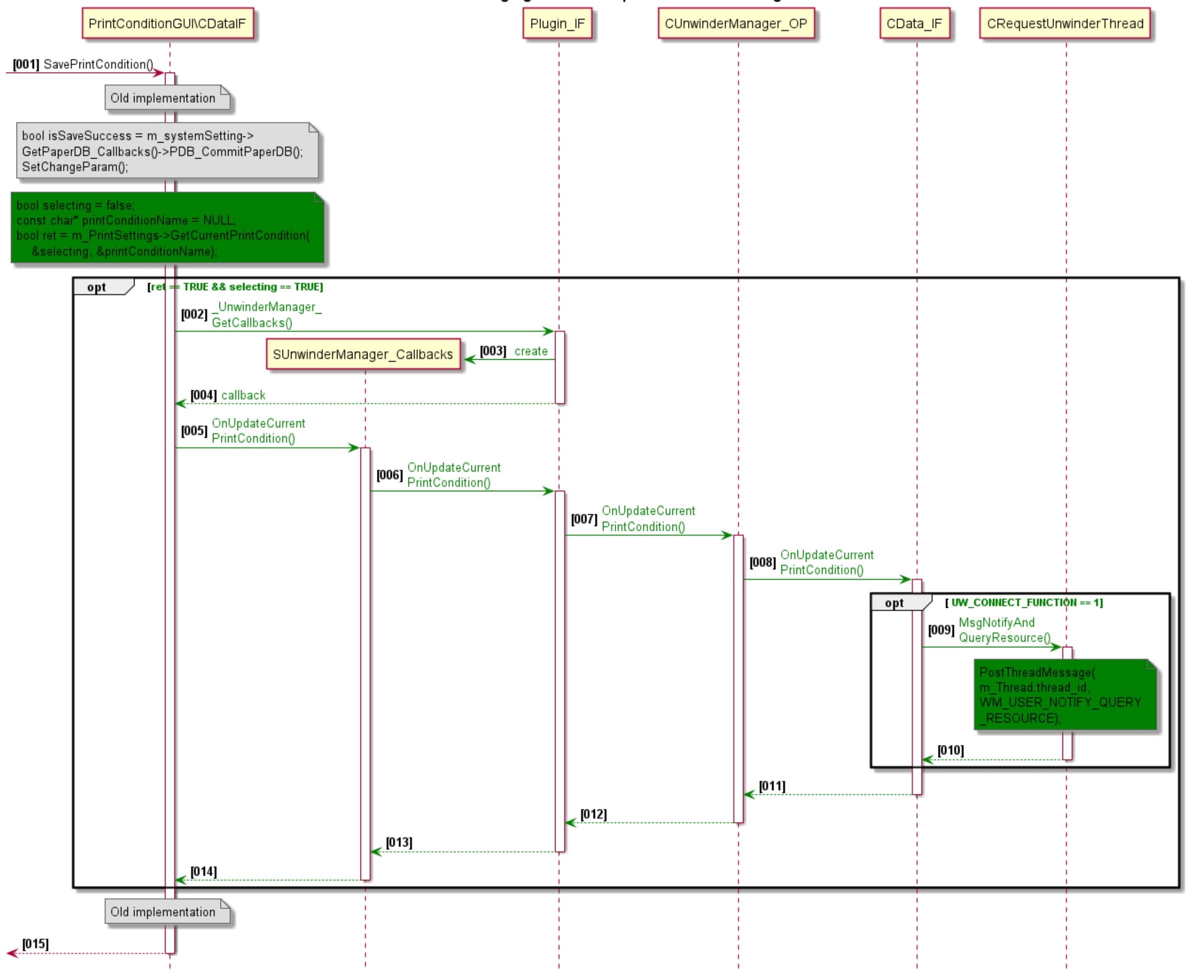
```
outCallbacks->OnFirstJobRun = OnFirstJobRun;
outCallbacks->OnSetCurrentPrintCondition = OnSetCurrentPrintCondition;
outCallbacks->OnUpdateCurrentPrintCondition = OnUpdateCurrentPrintCondition;
```

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

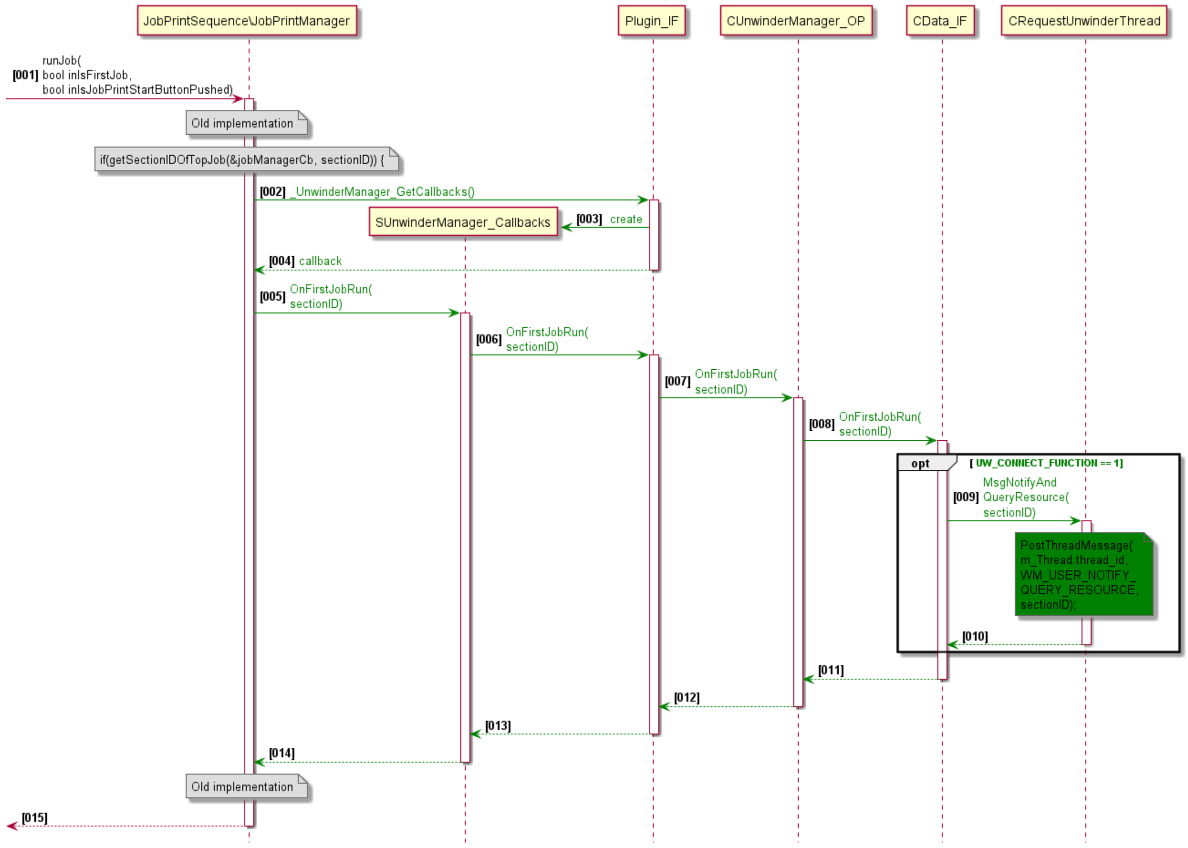

PrintConditionGUI\CDataIF
Plugin_IF
CUnwinderManager_OP
CData_IF
CRequestUnwinderThread
[001] SavePrintCondition()
Old implementation
opt
[ret == TRUE && selecting == TRUE]
[002] _UnwinderManager_ GetCallbacks()
SUnwinderManager_Callbacks
[003] create
[004] callback
[005] OnUpdateCurrent PrintCondition()
[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);
```

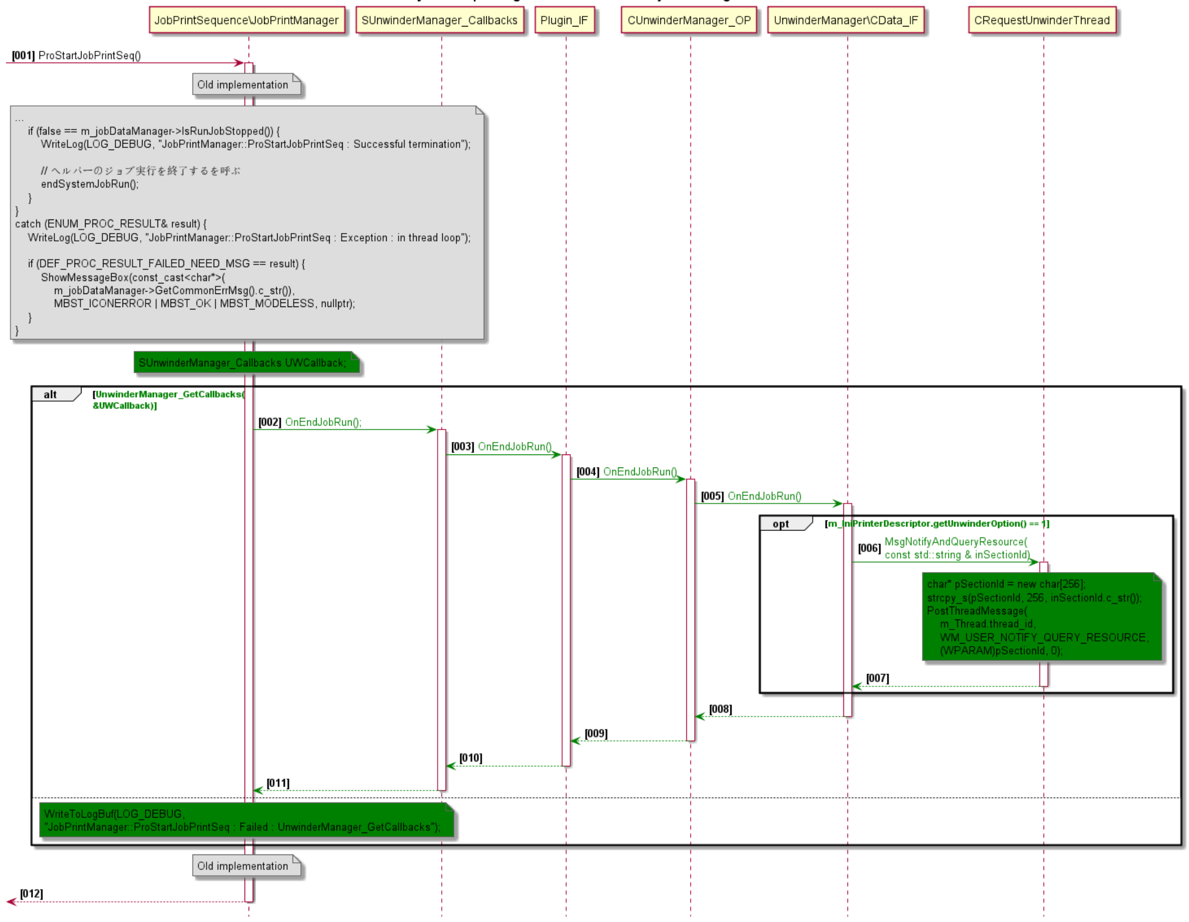
```
PostThreadMessage(
m_Thread.thread_id,
WM_USER_NOTIFY_QUERY
_RESOURCE);
```

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)
[010]
[011]
[012]
[013]
[014]
Old implementation
[015]

```
PostThreadMessage(
m_Thread.thread_id,
WM_USER_NOTIFY_
QUERY_RESOURCE,
sectionID);
```

203.8 Notify UW of printing conditions when the job running status is released


JobPrintSequence\JobPrintManager
SUnwinderManager_Callbacks
Plugin_IF
CUnwinderManager_OP
UnwinderManager\CData_IF
CRequestUnwinderThread
[001] ProStartJobPrintSeq()
Old implementation
SUnwinderManager_Callbacks UWCallback;
alt
[UnwinderManager_GetCallbacks(&UWCallback)]
[002] OnEndJobRun();
[003] OnEndJobRun()
[004] OnEndJobRun()
[005] OnEndJobRun()
opt
[m_IniPrinterDescriptor.getUnwinderOption() == 1]
[006] MsgNotifyAndQueryResource(const std::string & inSectionId)
[007]
[008]
[009]
[010]
[011]
Old implementation
[012]

```
...
    if (false == m_jobDataManager->IsRunJobStopped()) {
        WriteLog(LOG_DEBUG, "JobPrintManager::ProStartJobPrintSeq : Successful termination");

        // ヘルパーのジョブ実行を終了するを呼ぶ
        endSystemJobRun();
    }
}
catch (ENUM_PROC_RESULT& result) {
    WriteLog(LOG_DEBUG, "JobPrintManager::ProStartJobPrintSeq : Exception : in thread loop");

    if (DEF_PROC_RESULT_FAILED_NEED_MSG == result) {
        ShowMessageBox(const_cast<char*>(
            m_jobDataManager->GetCommonErrMsg().c_str()),
            MBST_ICONERROR | MBST_OK | MBST_MODELESS, nullptr);
    }
}
```

```
char* pSectionId = new char[256];
strcpy_s(pSectionId, 256, inSectionId.c_str());
PostThreadMessage(
    m_Thread.thread_id,
    WM_USER_NOTIFY_QUERY_RESOURCE,
    (WPARAM)pSectionId, 0);
```

```
WriteToLogBuf(LOG_DEBUG,
"JobPrintManager::ProStartJobPrintSeq : Failed : UnwinderManager_GetCallbacks");
```

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

1. Description

205-1. 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.

205-2. 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.

205-3. Reflect paper thickness information

The UW paper thickness is reflected in the paper thickness information of the current print condition or the print condition during job execution.
The update timing and print conditions to be updated are as follows.

1. Current print condition

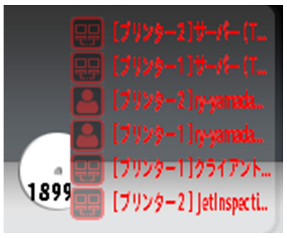
   If there is a difference between the paper thickness of the current print condition and the paper thickness of UW when the controller is started, the current print condition is switched, or the setting contents of the current print condition are changed, the following warning dialog is displayed.
   (Ja) カレント印刷条件の紙厚情報を更新しますか？
   (En) Undecided
   If press the “Yes” button, the paper thickness of the current print conditions will be updated.
   If press the “No” button, nothing is done.

2. Print condition during job execution

   When the job execution button or the continuous print button is pressed, the following warning dialog is displayed when there is a difference between the paper thickness of the print condition to be executed and the paper thickness of UW.
   (Ja) 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？
   (En) Undecided
   If press the “Yes” button is clicked, the job execution is stopped and the print condition paper thickness of the job to be executed is updated.
   If press the “No” button, the job execution status is entered as it is.

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

As shown in the figure below, the remaining amount of paper (in meters) is visually expressed in the area on the right side of the status bar. (Use UW2.bmp)
If the warning message and the remaining roll icon overlap, the former is given priority and displayed in the foreground.


[プリンター2]サーバー (T...
[プリンター1]サーバー (T...
[プリンター2]ry-yamada...
[プリンター1]ry-yamada...
[プリンター1]クライアント...
[プリンター2]JetInspecti...

In the future, when the remaining amount approaches the warning threshold, the winding core will be changed from white to yellow to red.
Currently, there is no way for the controller to know the initial remaining amount of the roll.
At one time, use only the design of the top row (when UW is not started) and the bottom row (when UW is started).

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:
    // Get instance of the class
    static CReceiveSignalStatus& GetInstance()
    {
        static CReceiveSignalStatus receiveSignalStatusInstance;
        return receiveSignalStatusInstance;
    }

    CReceiveSignalStatus(CReceiveSignalStatus const&) = delete;
    void operator=(CReceiveSignalStatus const&) = delete;

    virtual ~CReceiveSignalStatus();

    // Receive signal status info from UW
    void ReceiveSignalStatusInfo(CDataIF* inDataIF);

    // Receive signal status info notified from UWandRW_Receiver
    BOOL ReceiveInfo();

    // Set a CRequestUnwinderThread object
    void SetRequestThread(CRequestUnwinderThread* inRequestThread);

    // Get the paper info receiving status (whether or not the info has been received)
    void HasPaperInfoReceived(bool &outHasPaperInfoReceived);
    // Get value of paper thickness which is notified from UW
    void GetUWPaperThickness(long &outThickness);
    /// Get value of paper remaining amount which is notified from UW
    void GetUWPaperRemainingAmount(long &outPaperRemainingAmount);
    // Set status of paper info receiving
    void SetStatusOfPaperInfoReceiving(bool status);
    // Set the status for starting of paper info receiving
    void SetRecvPaperInfoStartingStatus(bool status);
    // Call from CReceiveSignalStatusThread to set/reset break loop condition
    void SetExit(bool inVal);

private:
    // CReceiveSignalStatus constructor
    CReceiveSignalStatus();

    // pipe reading
    BOOL ReadData(HANDLE inPipe, char* outData, DWORD inSize);

    // analyze signal status info notified from UWandRW_Receiver
    BOOL AnalyzeData(const std::string& inXmldata);

    // parse xml data
    std::string ExecuteParseXml( const std::string& inSignalData, UwXjmfDataMap&
                                 outUwXjmfDataMap );

    // 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);

    CRequestUnwinderThread* m_RequestThread;
    CDataIF* m_dataIF;

    bool m_paperInfoReceivingStatus; // status of paper info receiving
    bool m_receivedPaperInfo; // whether or not the paper info has been received
    bool m_isRecvPaperInfoStarting; // whether it is the first time of paper
                //info receiving from UW since the controller was started
    long m_UWThickness; // Value of thickness notified from UW.
    long m_paperRemainingAmount; // Value of paper remaining
                        // amount notified from UW.
    bool m_IsExit; // set when thread exit
};
```

- In class CRequestUnwinderThread, add function to delete/set timer to check signal timeout.

RequestUnwinderThread.h

```
#define WM_USER_SET_PAPERINFO_TIMER             WM_USER+101
class CRequestUnwinderThread
{
    ...
    // sent msg from ReceiveSignalStatus thread
        void MsgSetTimerPaperReceive();
    // setting for timeout timer of SignalStatus(PAPER)
    void SetTimerPaperReceive();
    // stop timeout timer of SignalStatus(PAPER)
    void KillTimerPaperReceive();
    ...
}
```

- 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_isRecvPaperInfoStarting is true and flag variable m_paperInfoReceivingStatus is true), 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 and paper remaining amount values notified from UW to variables m_UWThickness and m_paperRemainingAmount.
    - Set the remaining amount of paper and roll diameter into TP-UW_Communication.ini file by new created functions: CIni_UnwinderManager_work::SetRollDiameter and CIni_UnwinderManager_work::SetPaperRemainingAmount
- Add plugin callback functions:

Common\UnwinderManager_Callbacks.h

```
typedef void(*_GetUWPaperThickness)(long &outThickness);
typedef void(*_GetUWPaperRemaingAmount)(long &outPaperRemaingAmount);
typedef void(*_HasPaperInfoReceived)(bool &outHasPaperInfoReceived);
typedef void(*_UpdateUWPaperThicknessForJob)(
    const std::string &inSectionID, bool &outIsUpdateRequested);
typedef void(*_UpdateUWPaperThicknessForConsecutiveJobs)(
    const std::vector<std::string> &inConsecutiveJobSectionIDs, bool &outIsUpdateRequested);
typedef void(*_CheckUpdateUWPaperThicknessForCurrentPrintCondition)(
    long inCurrentPaperThickness, bool &outIsUpdateRequested);

struct SUnwinderManager_Callbacks
{
    //Version 1
    ...
    _GetUWPaperThickness                        GetUWPaperThickness;
    _GetUWPaperRemaingAmount            GetUWPaperRemaingAmount;
    _HasPaperInfoReceived                       HasPaperInfoReceived;

    //Version 2
    _UpdateUWPaperThicknessForJob UpdateUWPaperThicknessForJob;
    _UpdateUWPaperThicknessForConsecutiveJobs UpdateUWPaperThicknessForConsecutiveJobs;
    _CheckUpdateUWPaperThicknessForCurrentPrintCondition CheckUpdateUWPaperThicknessForCurre
}
```

Common\UnwinderManager_OP.h

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

        void UpdateUWPaperThicknessForJob(
            const std::string &inSectionID, bool &outIsUpdateRequested);
        void UpdateUWPaperThicknessForConsecutiveJobs(
            const std::vector<std::string> &inConsecutiveJobSectionIDs,
            bool &outIsUpdateRequested);
        void CheckUpdateUWPaperThicknessForCurrentPrintCondition(
            long inCurrentPaperThickness, bool &outIsUpdateRequested);
    }
```

UnwinderManager\Data_IF.h

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

    void UpdateUWPaperThicknessForJob(
        const std::string &inSectionID, bool &outIsUpdateRequested);
    void UpdateUWPaperThicknessForConsecutiveJobs(
        const std::vector<std::string> &inConsecutiveJobSectionIDs,
        bool &outIsUpdateRequested);
    void CheckUpdateUWPaperThicknessForCurrentPrintCondition(
        long inCurrentPaperThickness, bool &outIsUpdateRequested);
}
```

- In PrintConditionGUI plugin, in CDataPrintSettings::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 and CCtlJobList::OnCommand functions of JobSelectGUI plugin, implement same as in CDataIF::SetCurrentPrintCondition and CDataIF::SavePrintCondition functions, except reflecting the paper thickness value to print condition of jobs, 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

```
[MSG]
//English
IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION = (*The content is undecided)
IDM_UPDATE_PAPER_THICKNESS_TO_JOB = (*The content is undecided)
//Japanese
IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION = カレント印刷条件の紙厚情報を更新しますか？
IDM_UPDATE_PAPER_THICKNESS_TO_JOB = 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 warning messages and UW icon:

CtlUnwinder.h

```
class CCtlWarningIconAndUW : public CBaseCtl
{
public:
    CCtlWarningIconAndUW();
    virtual ~CCtlWarningIconAndUW();
    virtual long Proc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam);
    virtual void OnUpdateState();
    virtual void OnUpdateValue();
protected:
    virtual void OnSetAttribute();
    virtual void OnCreateItem();
private:
    long m_inItemID;    //!< id of warning, which is displaying in HintDlg
    DEF_PRINTER_WARNING m_outWarning; //!< type of warning is displaying in HintDlg
    std::string m_outMessage; //!< message of warning, which is dispalying in HintDlg
    long m_hWndHintDlg;         //!< id of HintDlg
};
```

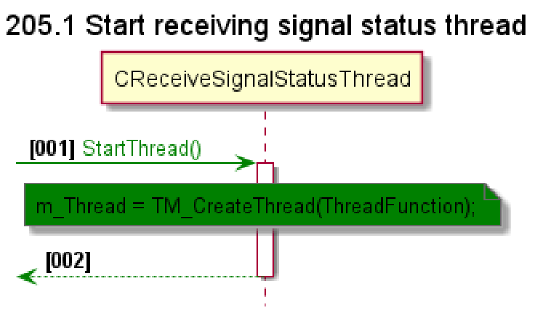
- Refer to CCtlWarningIcon to handle for the warning messages.
- In function CCtlWarningIconAndUW::OnUpdateValue, check the status of the UW. If it is not started, display a translucent UW icon (the top icon). If it is started, display normal icon (the bottom icon).
- In function CCtlWarningIconAndUW::OnUpdateState, if the status of UW is OFF then hide the paper remaining static box.
- In StatusBar\CDataIF class, add methods and member variable related to UW status:

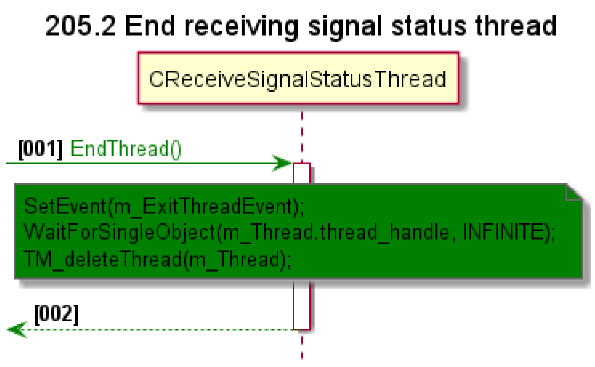
StatusBarDataIF.h

```
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;
};
```

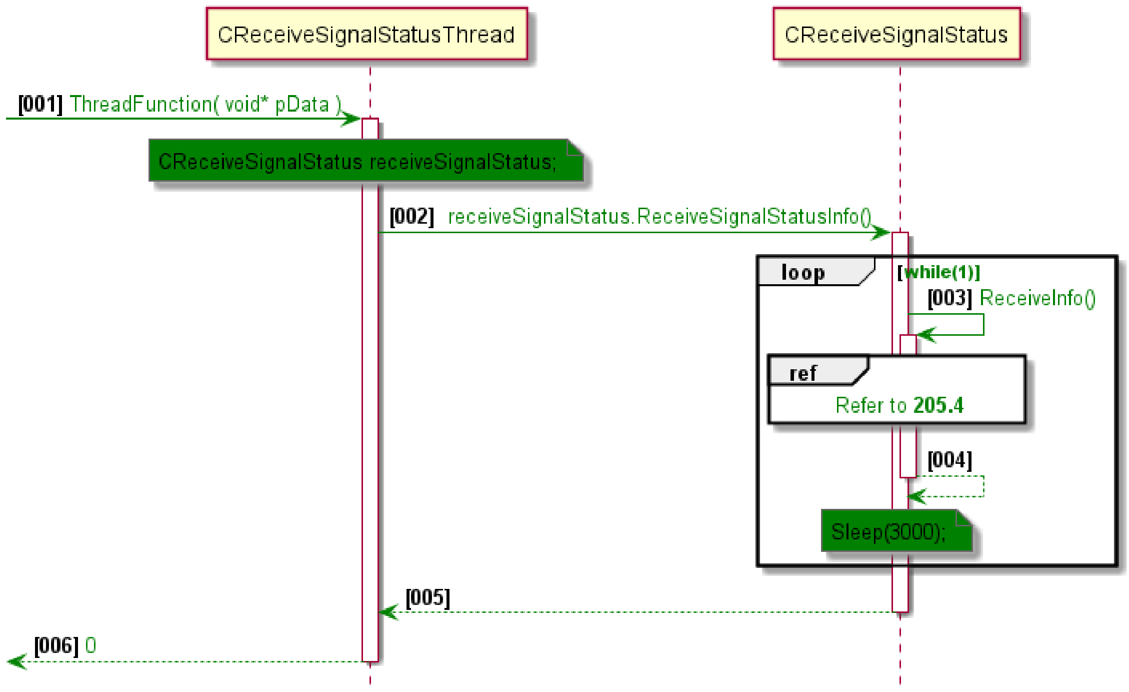
- Modify size of displaying area of data and time control when UW icon is displayed.

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]

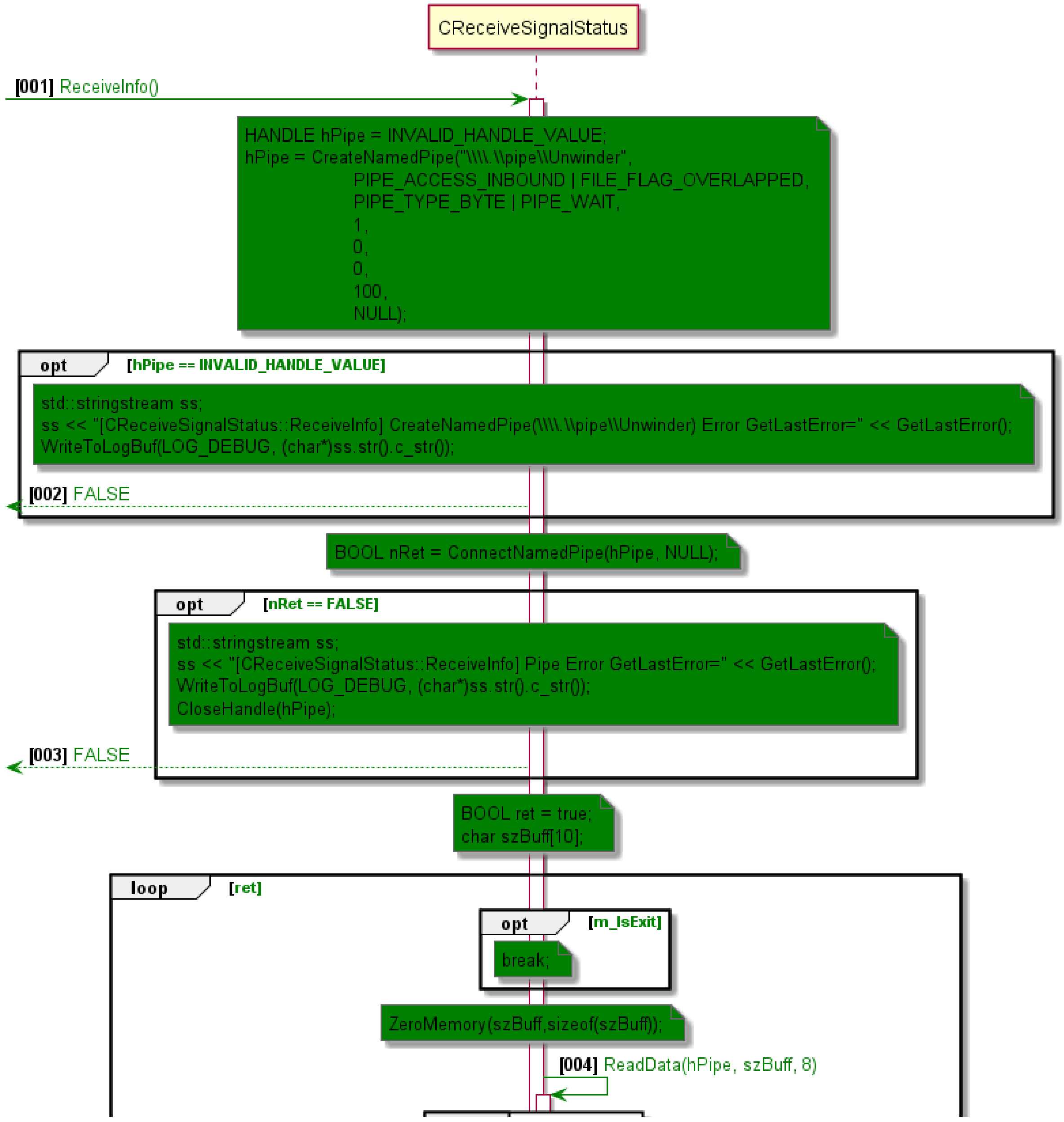
205.3 Receiving thread


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;
```

```
Sleep(3000);
```

205.4 Receive signal status notified from UWandRW_Receiver



```
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());
```

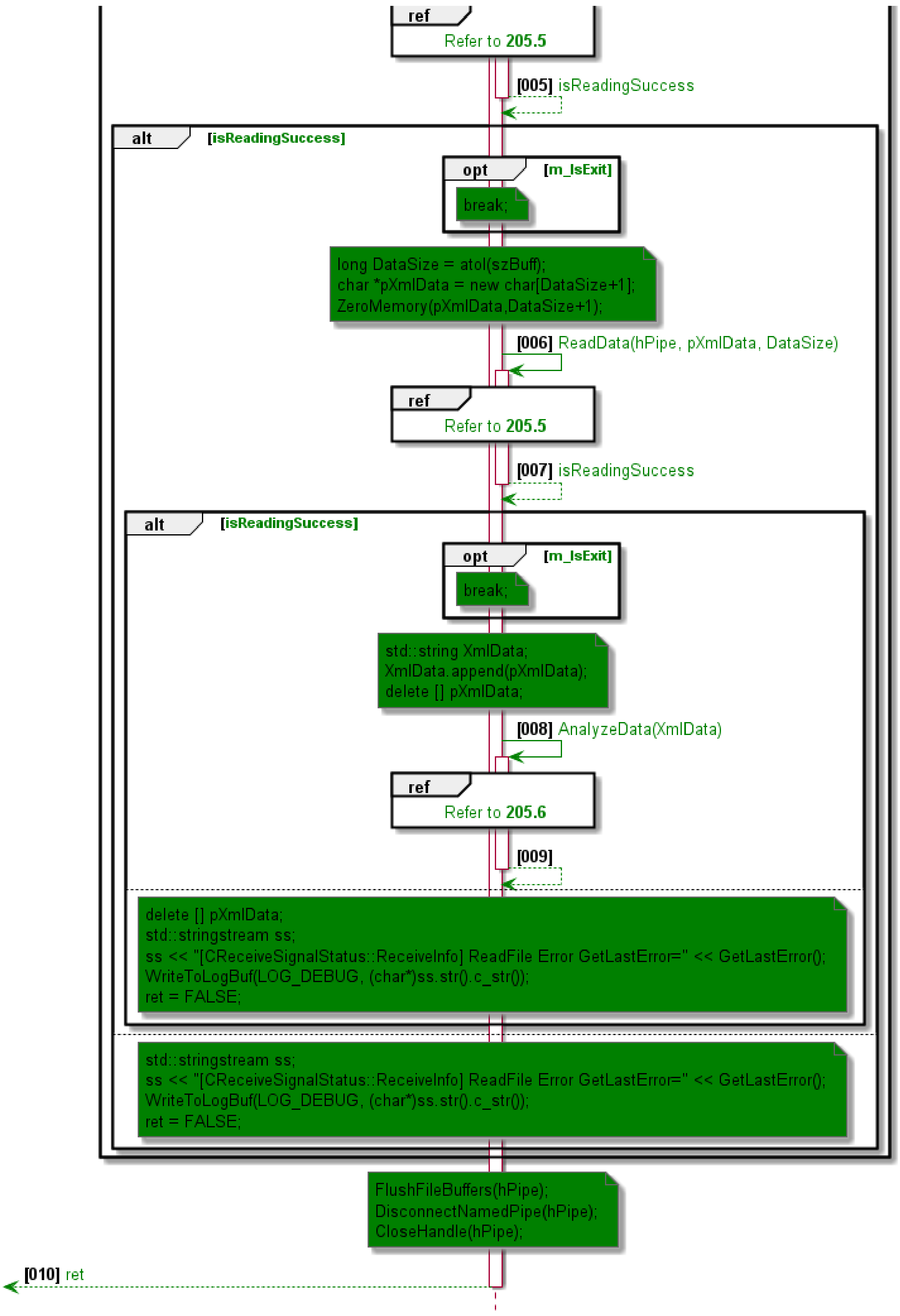
```
BOOL nRet = ConnectNamedPipe(hPipe, NULL);
```

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

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

```
break;
```

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



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

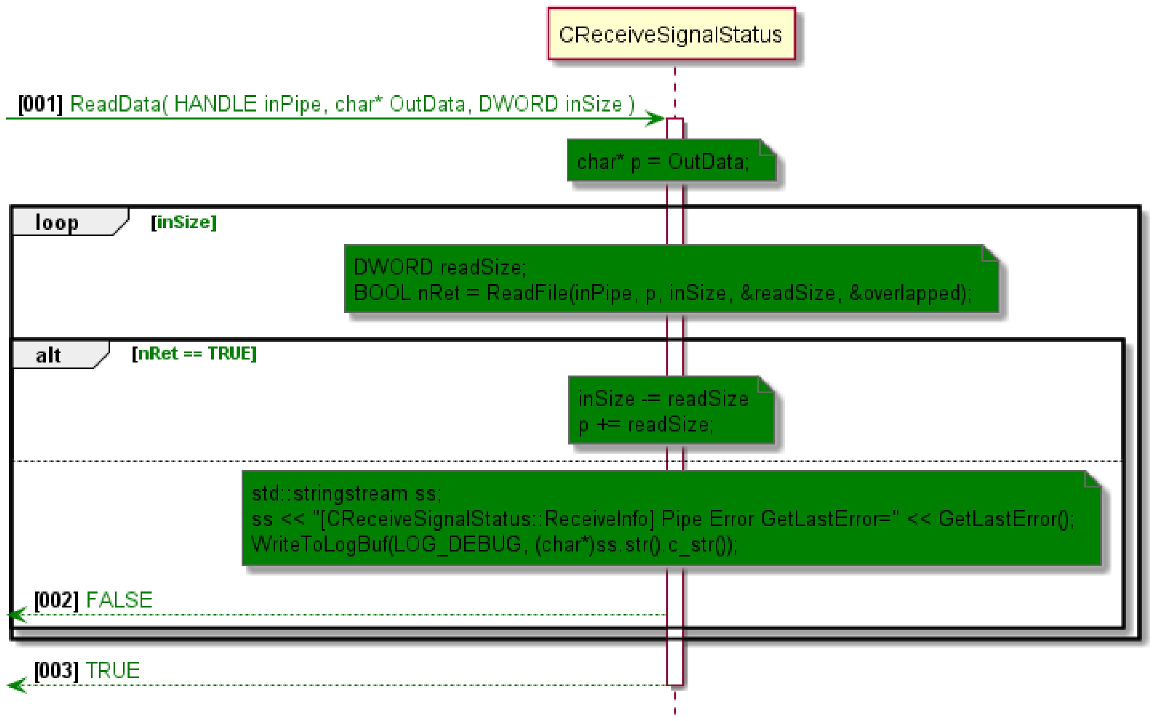
```
std::string XmlData;
XmlData.append(pXmlData);
delete [] pXmlData;
```

```
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;
```

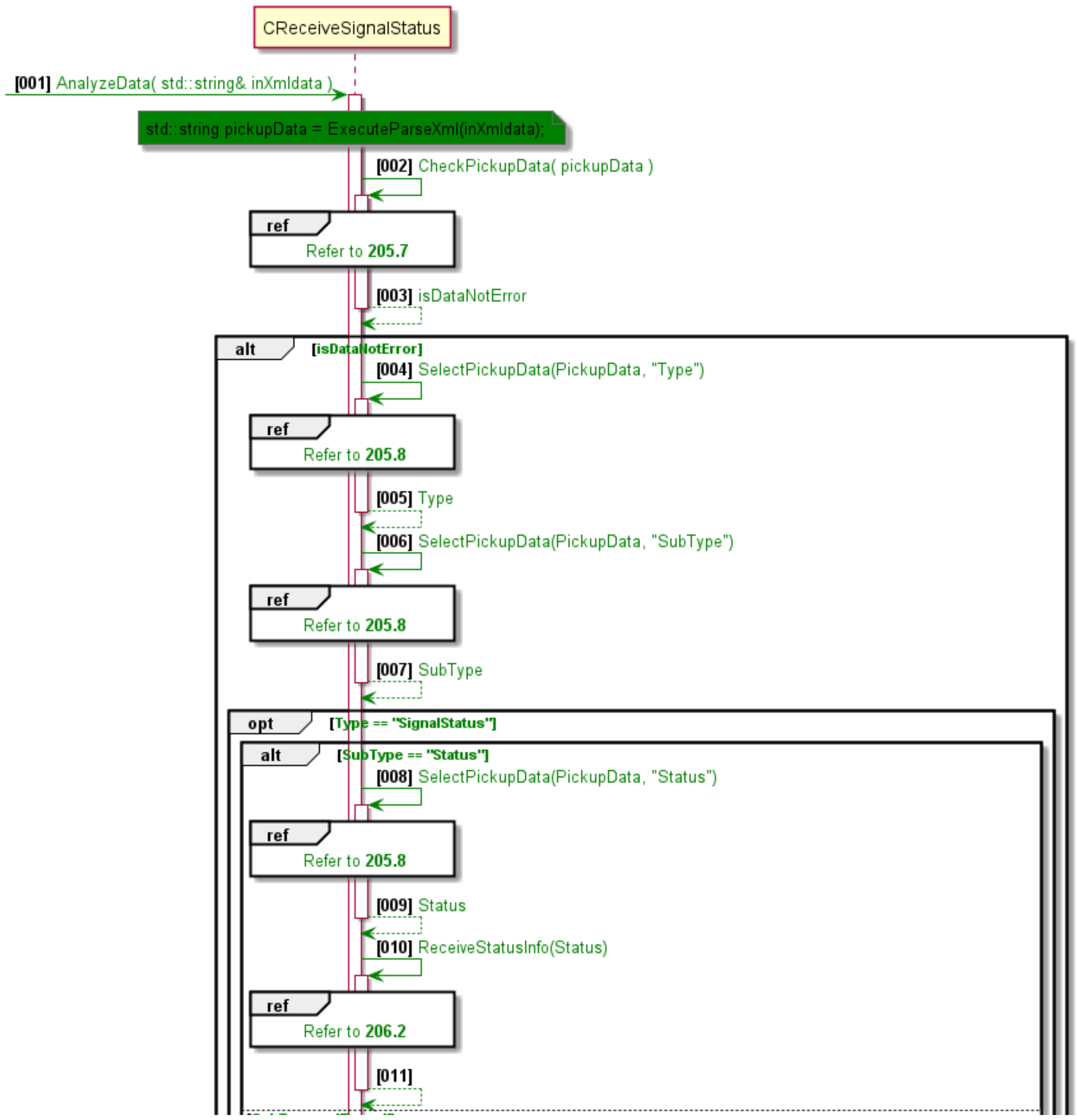
```
FlushFileBuffers(hPipe);
DisconnectNamedPipe(hPipe);
CloseHandle(hPipe);
```

205.5 Pipe reading

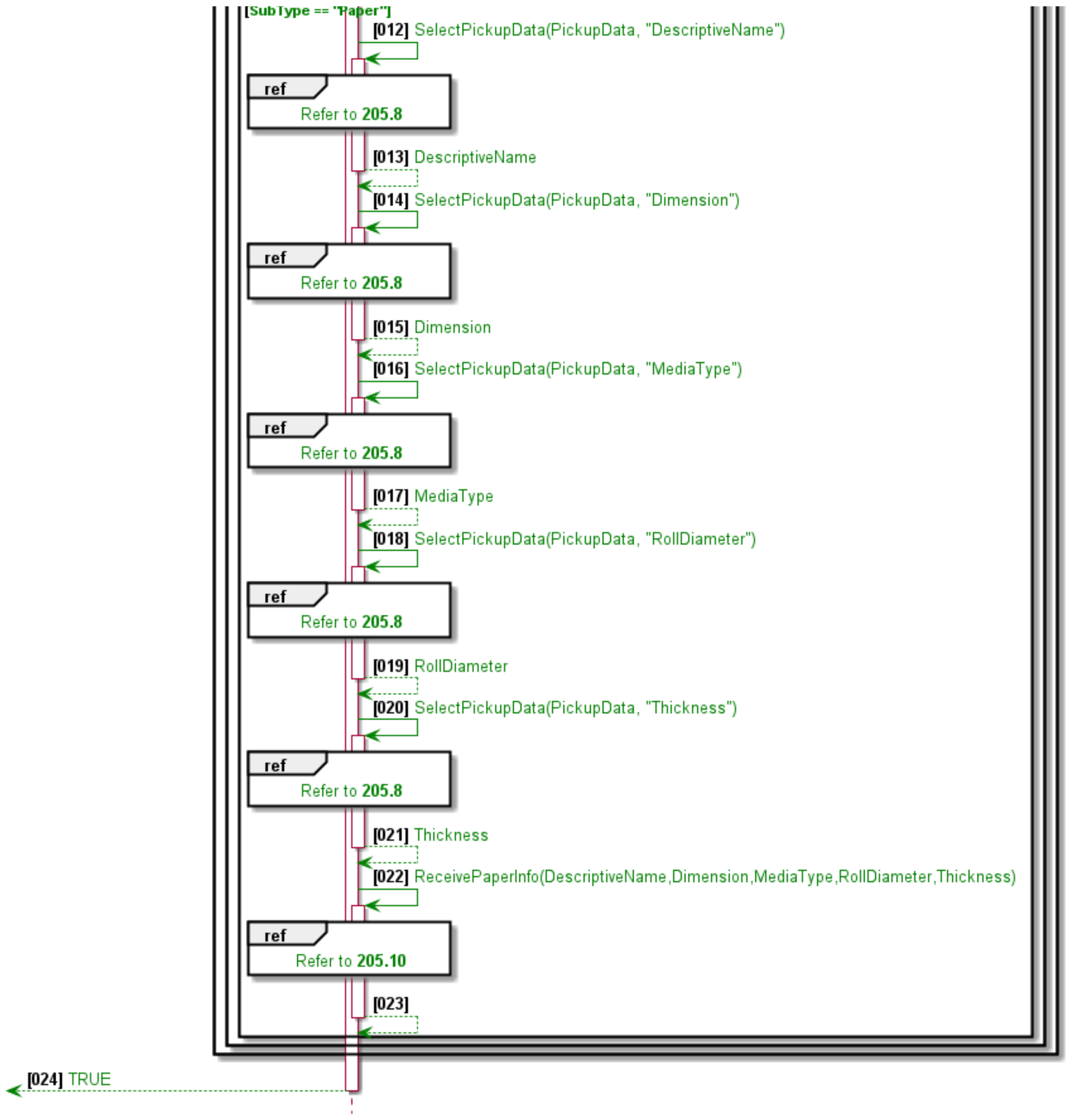


CReceiveSignalStatus
[001] ReadData(HANDLE inPipe, char* OutData, DWORD inSize)
char* p = OutData;
loop [inSize]
DWORD readSize;
BOOL nRet = ReadFile(inPipe, p, inSize, &readSize, &overlapped);
alt [nRet == TRUE]
inSize -= readSize
p += readSize;
std::stringstream ss;
ss << "[CReceiveSignalStatus::ReceiveInfo] Pipe Error GetLastError=" << GetLastError();
WriteToLogBuf(LOG_DEBUG, (char*)ss.str().c_str());
[002] FALSE
[003] TRUE

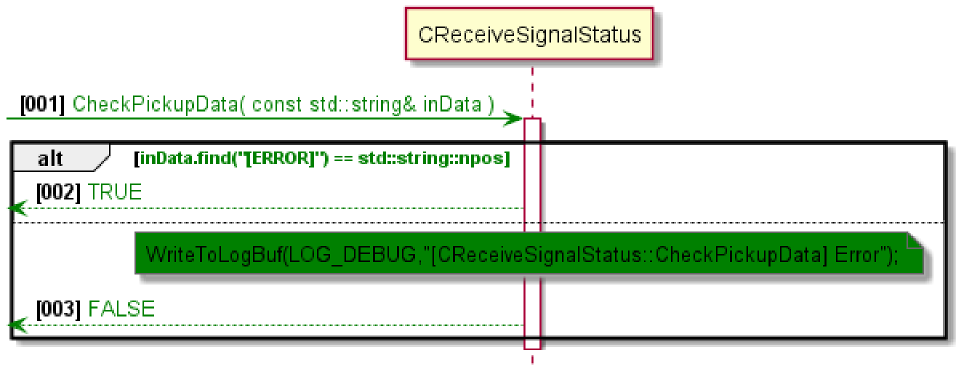
205.6 Analyze signal status info notified from UWandRW_Receiver



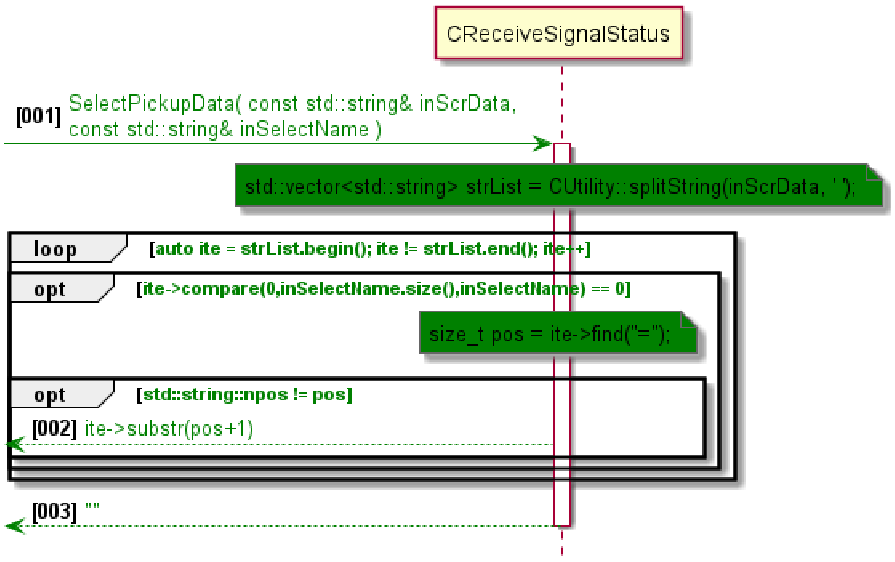
CReceiveSignalStatus
[001] AnalyzeData(std::string& inXmldata)
std::string pickupData = ExecuteParseXml(inXmldata);
[002] CheckPickupData(pickupData)
ref
Refer to 205.7
[003] isDataNotError
alt [isDataNotError]
[004] SelectPickupData(PickupData, "Type")
ref
Refer to 205.8
[005] Type
[006] SelectPickupData(PickupData, "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]
[024] TRUE

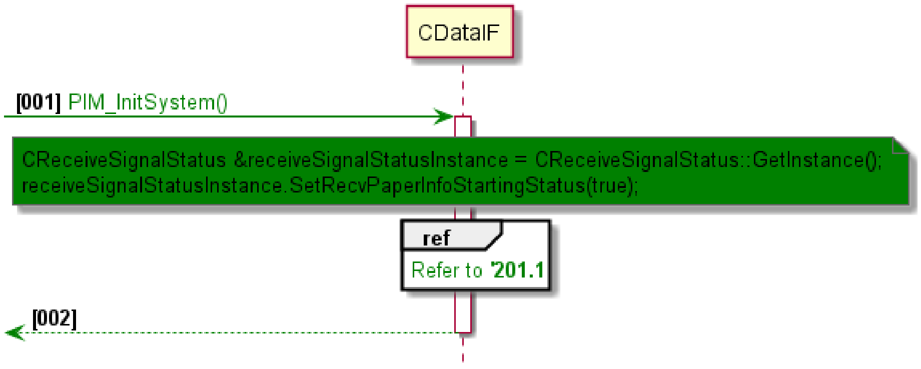
205.7 Check whether the data returned from UWandRW_Parse_Xml.exe is valid or not


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

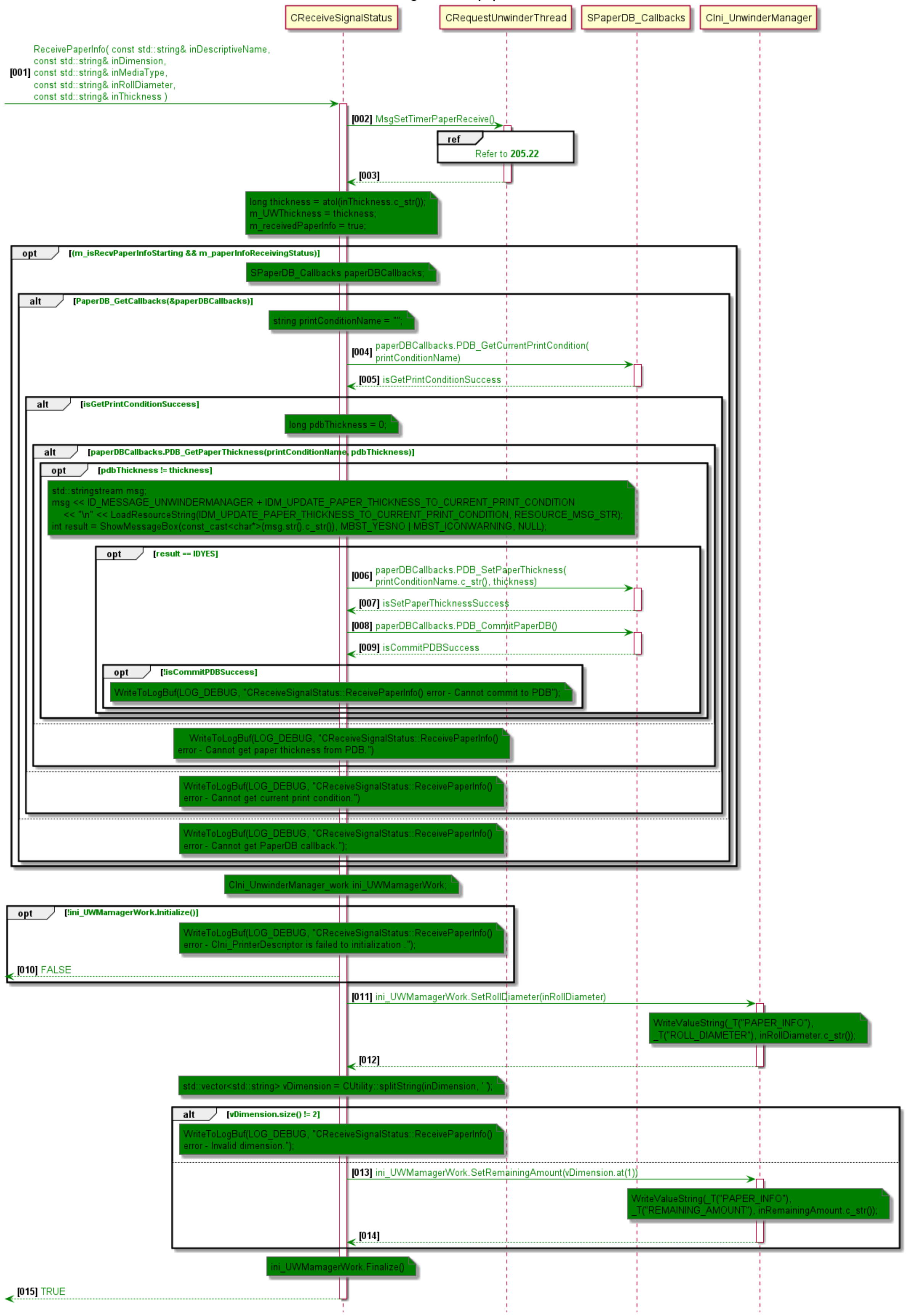
205.8 Extract data from the parsed info


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

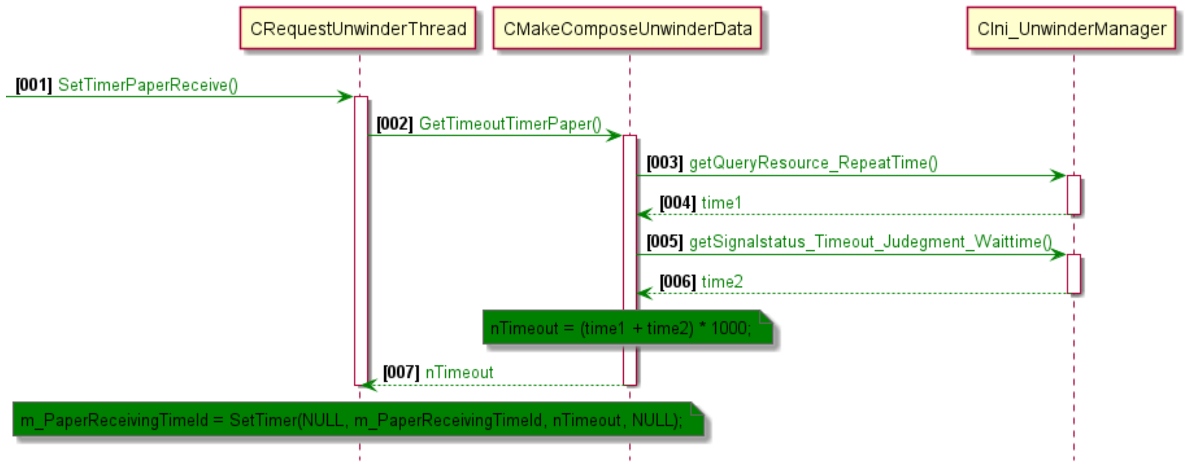

CDataIF
[001] PIM_InitSystem()
CReceiveSignalStatus &receiveSignalStatusInstance = CReceiveSignalStatus::GetInstance();
receiveSignalStatusInstance.SetRecvPaperInfoStartingStatus(true);
ref
Refer to '201.1
[002]

205.10 Processing when the paper info is received

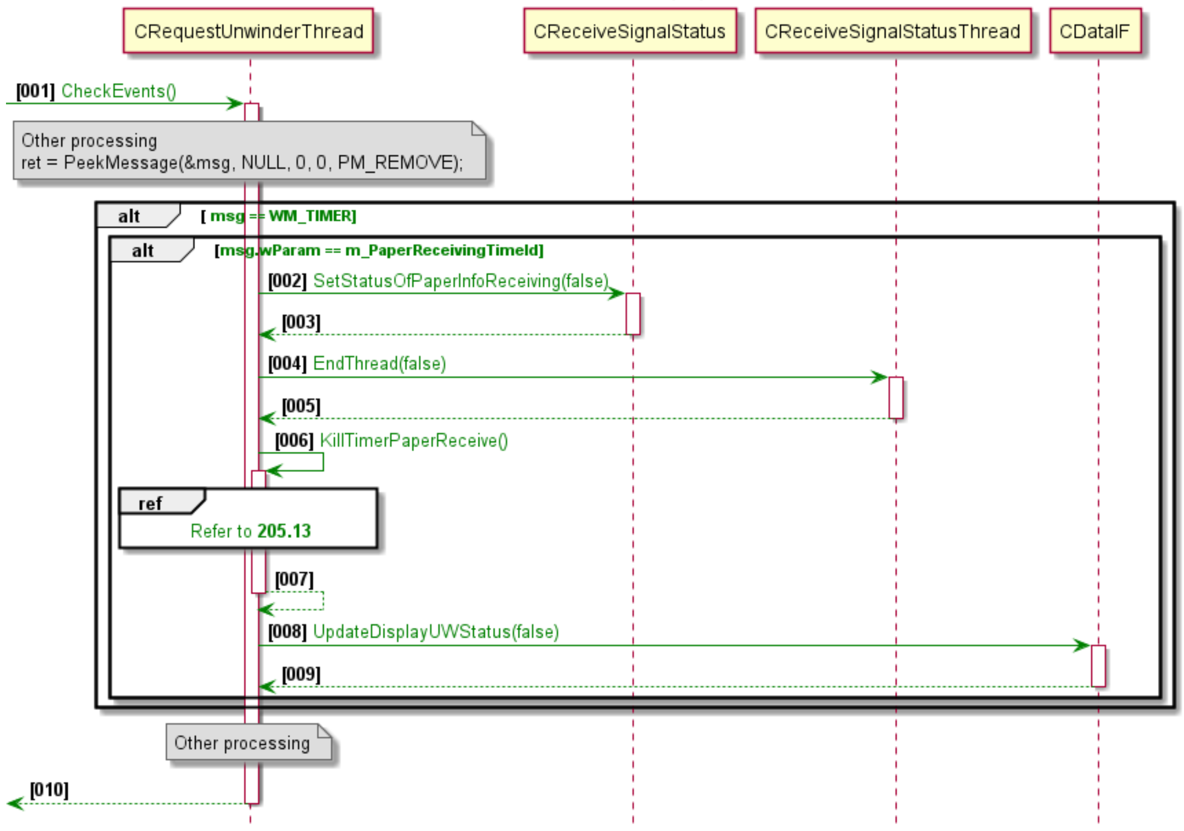


```
[001] ReceivePaperInfo( const std::string& inDescriptiveName,
      const std::string& inDimension,
      const std::string& inMediaType,
      const std::string& inRollDiameter,
      const std::string& inThickness )
[002] MsgSetTimerPaperReceive()
[003]
long thickness = atol(inThickness.c_str());
m_UWThickness = thickness;
m_receivedPaperInfo = true;
opt [(m_isRecvPaperInfoStarting && m_paperInfoReceivingStatus)]
  SPaperDB_Callbacks paperDBCallbacks;
  alt [PaperDB_GetCallbacks(&paperDBCallbacks)]
    string printConditionName = "";
    [004] paperDBCallbacks.PDB_GetCurrentPrintCondition(
          printConditionName)
    [005] isGetPrintConditionSuccess
    alt [isGetPrintConditionSuccess]
      long pdbThickness = 0;
      alt [paperDBCallbacks.PDB_GetPaperThickness(printConditionName, pdbThickness)]
        opt [pdbThickness != thickness]
          std::stringstream msg;
          msg << ID_MESSAGE_UNWINDERMANAGER + IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION
              << "\n" << LoadResourceString(IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION, RESOURCE_MSG_STR);
          int result = ShowMessageBox(const_cast<char*>(msg.str().c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
          opt [result == IDYES]
            [006] paperDBCallbacks.PDB_SetPaperThickness(
                  printConditionName.c_str(), thickness)
            [007] isSetPaperThicknessSuccess
            [008] paperDBCallbacks.PDB_CommitPaperDB()
            [009] isCommitPDBSuccess
            opt [!isCommitPDBSuccess]
              WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo() error - Cannot commit to PDB");
      ----
        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_work ini_UWMamagerWork;
opt [!ini_UWMamagerWork.Initialize()]
  WriteToLogBuf(LOG_DEBUG, "CReceiveSignalStatus::ReceivePaperInfo()
  error - CIni_PrinterDescriptor is failed to initialization .");
  [010] FALSE
[011] ini_UWMamagerWork.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_UWMamagerWork.SetRemainingAmount(vDimension.at(1))
    WriteValueString(_T("PAPER_INFO"),
    _T("REMAINING_AMOUNT"), inRemainingAmount.c_str());
  [014]
ini_UWMamagerWork.Finalize()
[015] TRUE
```

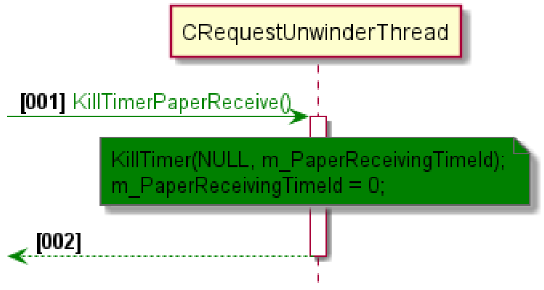
205.11 Setting for timeout timer of SignalStatus(PAPER)


CRequestUnwinderThread
CMakeComposeUnwinderData
CIni_UnwinderManager
[001] SetTimerPaperReceive()
[002] GetTimeoutTimerPaper()
[003] getQueryResource_RepeatTime()
[004] time1
[005] getSignalstatus_Timeout_Judegment_Waittime()
[006] time2
nTimeout = (time1 + time2) * 1000;
[007] nTimeout
m_PaperReceivingTimeId = SetTimer(NULL, m_PaperReceivingTimeId, nTimeout, NULL);

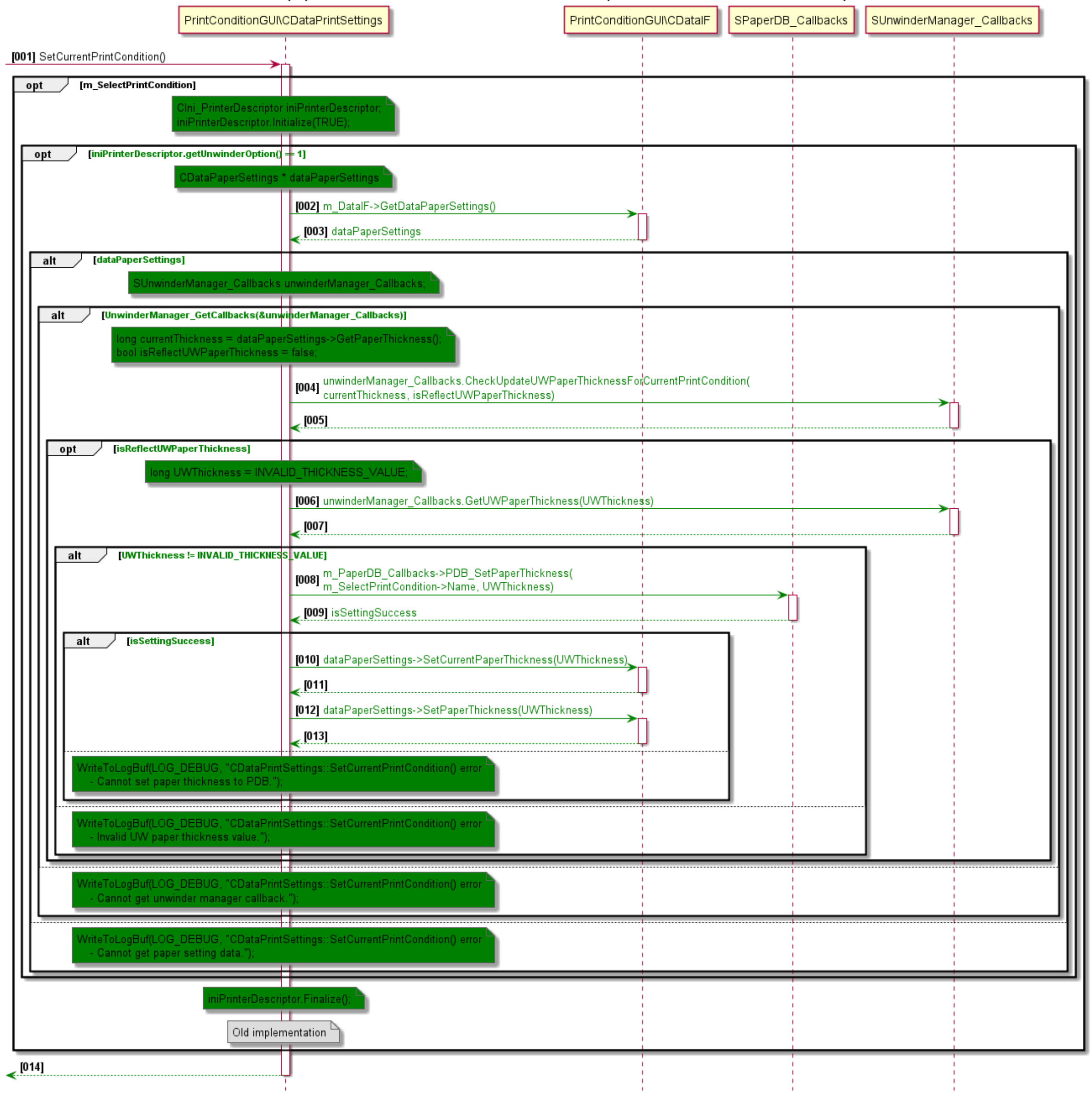
205.12 Handle when receive UW paper info is timeout


CRequestUnwinderThread
CReceiveSignalStatus
CReceiveSignalStatusThread
CDataIF
[001] CheckEvents()
Other processing
ret = PeekMessage(&msg, NULL, 0, 0, PM_REMOVE);
alt
[msg == WM_TIMER]
alt
[msg.wParam == m_PaperReceivingTimeId]
[002] SetStatusOfPaperInfoReceiving(false)
[003]
[004] EndThread(false)
[005]
[006] KillTimerPaperReceive()
ref
Refer to 205.13
[007]
[008] UpdateDisplayUWStatus(false)
[009]
Other processing
[010]

205.13 Stop timeout timer of SignalStatus(PAPER)


CRequestUnwinderThread
[001] KillTimerPaperReceive()
KillTimer(NULL, m_PaperReceivingTimeId);
m_PaperReceivingTimeId = 0;
[002]

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


PrintConditionGUI\CDataPrintSettings
PrintConditionGUI\CDataIF
SPaperDB_Callbacks
SUnwinderManager_Callbacks
[001] SetCurrentPrintCondition()
opt
[m_SelectPrintCondition]
opt
[iniPrinterDescriptor.getUnwinderOption() == 1]
[002] m_DataIF->GetDataPaperSettings()
[003] dataPaperSettings
alt
[dataPaperSettings]
alt
[UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
[004] unwinderManager_Callbacks.CheckUpdateUWPaperThicknessForCurrentPrintCondition(currentThickness, isReflectUWPaperThickness)
[005]
opt
[isReflectUWPaperThickness]
[006] unwinderManager_Callbacks.GetUWPaperThickness(UWThickness)
[007]
alt
[UWThickness != INVALID_THICKNESS_VALUE]
[008] m_PaperDB_Callbacks->PDB_SetPaperThickness(m_SelectPrintCondition->Name, UWThickness)
[009] isSettingSuccess
alt
[isSettingSuccess]
[010] dataPaperSettings->SetCurrentPaperThickness(UWThickness)
[011]
[012] dataPaperSettings->SetPaperThickness(UWThickness)
[013]
Old implementation
[014]

```
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);
```

```
CDataPaperSettings * dataPaperSettings
```

```
SUnwinderManager_Callbacks unwinderManager_Callbacks;
```

```
long currentThickness = dataPaperSettings->GetPaperThickness();
bool isReflectUWPaperThickness = false;
```

```
long UWThickness = INVALID_THICKNESS_VALUE;
```

```
WriteToLogBuf(LOG_DEBUG, "CDataPrintSettings::SetCurrentPrintCondition() error
    - Cannot set paper thickness to PDB.");
```

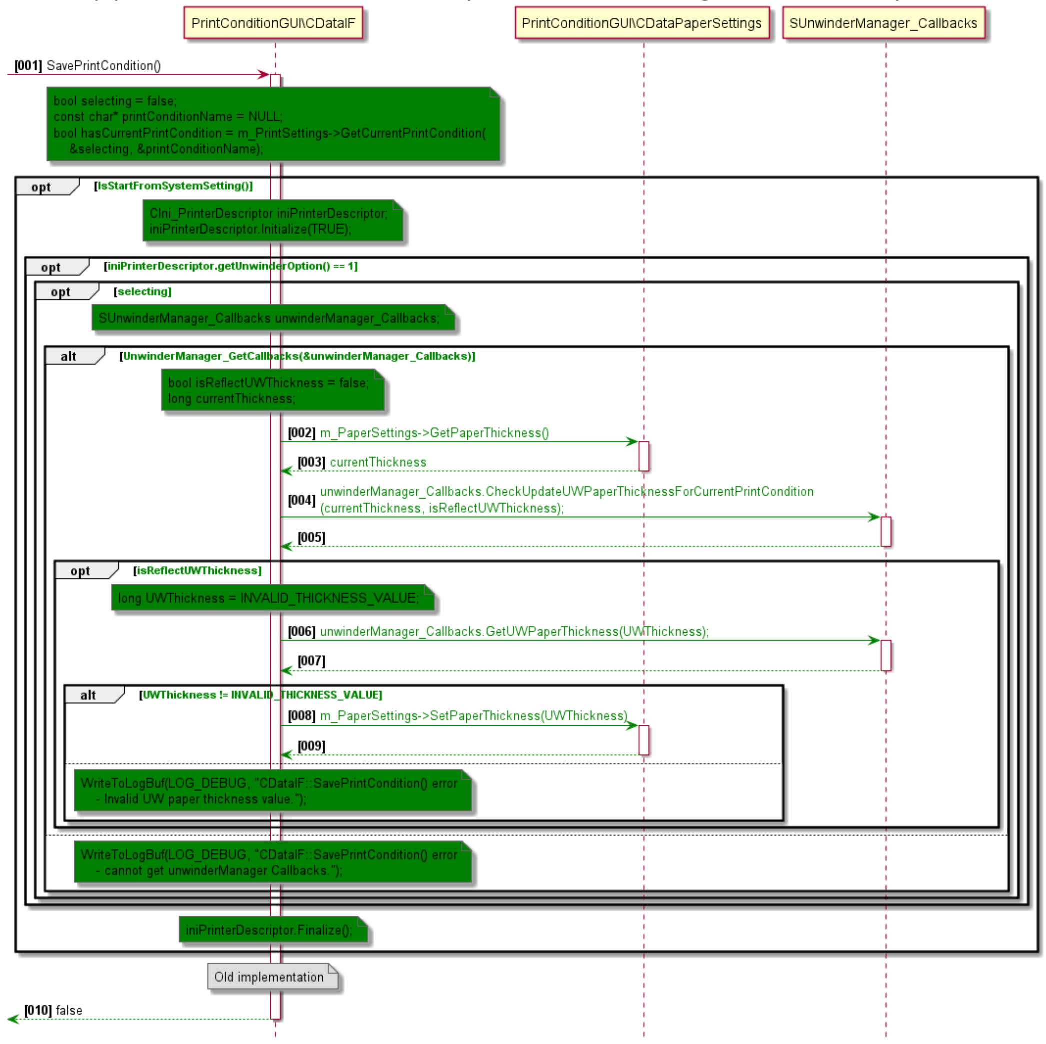
```
WriteToLogBuf(LOG_DEBUG, "CDataPrintSettings::SetCurrentPrintCondition() error
    - Invalid UW paper thickness value.");
```

```
WriteToLogBuf(LOG_DEBUG, "CDataPrintSettings::SetCurrentPrintCondition() error
    - Cannot get unwinder manager callback.");
```

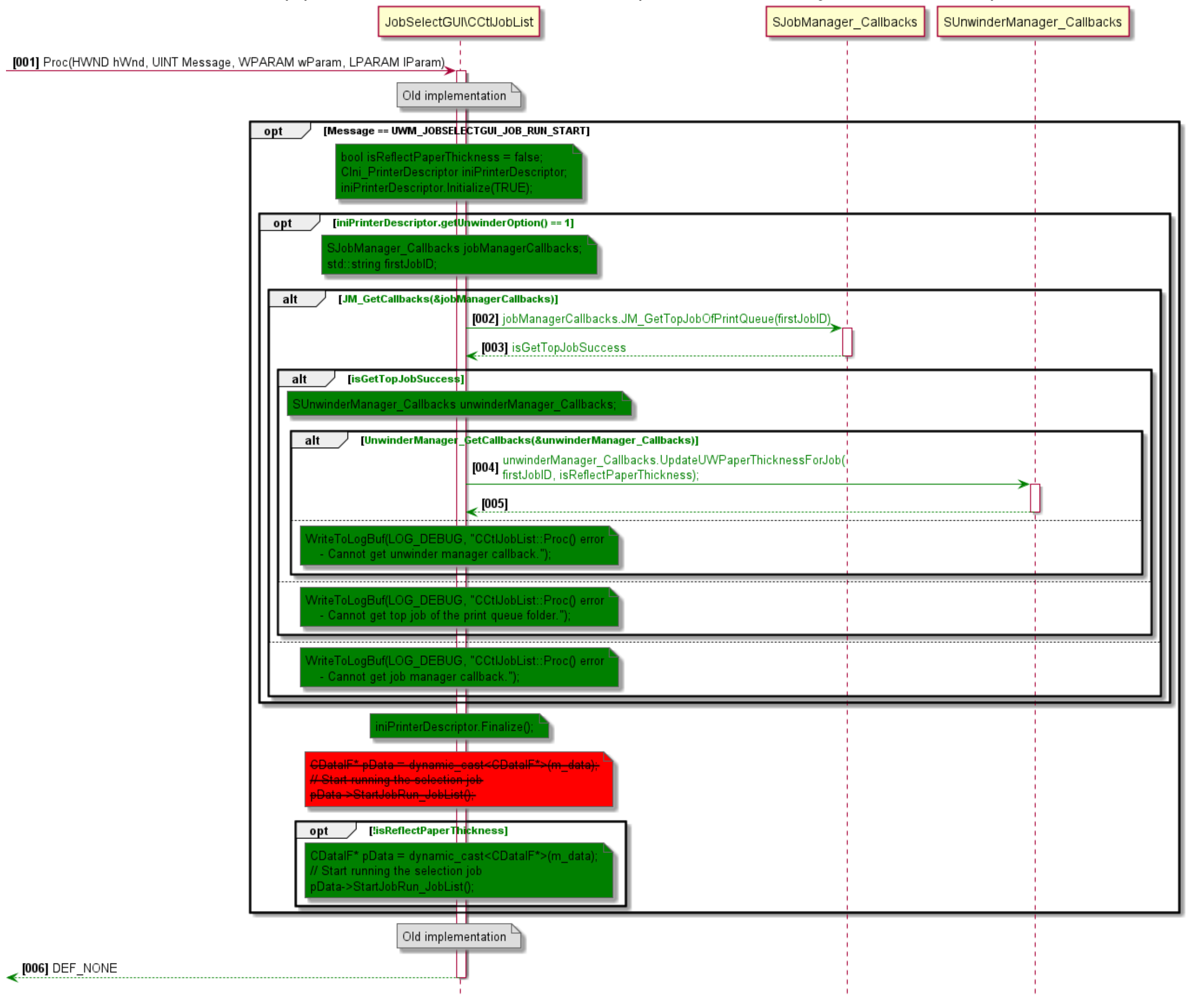
```
WriteToLogBuf(LOG_DEBUG, "CDataPrintSettings::SetCurrentPrintCondition() error
    - Cannot get paper setting data.");
```

```
iniPrinterDescriptor.Finalize();
```

205.15 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
PrintConditionGUI\CDataPaperSettings
SUnwinderManager_Callbacks
[001] SavePrintCondition()
bool selecting = false;
const char* printConditionName = NULL;
bool hasCurrentPrintCondition = m_PrintSettings->GetCurrentPrintCondition(
&selecting, &printConditionName);
opt [IsStartFromSystemSetting()]
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);
opt [iniPrinterDescriptor.getUnwinderOption() == 1]
opt [selecting]
SUnwinderManager_Callbacks unwinderManager_Callbacks;
alt [UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
bool isReflectUWThickness = false;
long currentThickness;
[002] m_PaperSettings->GetPaperThickness()
[003] currentThickness
[004] unwinderManager_Callbacks.CheckUpdateUWPaperThicknessForCurrentPrintCondition
(currentThickness, isReflectUWThickness);
[005]
opt [isReflectUWThickness]
long UWThickness = INVALID_THICKNESS_VALUE;
[006] unwinderManager_Callbacks.GetUWPaperThickness(UWThickness);
[007]
alt [UWThickness != INVALID_THICKNESS_VALUE]
[008] m_PaperSettings->SetPaperThickness(UWThickness)
[009]
WriteToLogBuf(LOG_DEBUG, "CDataIF::SavePrintCondition() error
- Invalid UW paper thickness value.");
WriteToLogBuf(LOG_DEBUG, "CDataIF::SavePrintCondition() error
- cannot get unwinderManager Callbacks.");
iniPrinterDescriptor.Finalize();
Old implementation
[010] false

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



[001] Proc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)

Old implementation

opt [Message == UWM_JOBSELECTGUI_JOB_RUN_START]

```
bool isReflectPaperThickness = false;
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);
```

opt [iniPrinterDescriptor.getUnwinderOption() == 1]

```
SJobManager_Callbacks jobManagerCallbacks;
std::string firstJobID;
```

alt [JM_GetCallbacks(&jobManagerCallbacks)]

[002] jobManagerCallbacks.JM_GetTopJobOfPrintQueue(firstJobID)

[003] isGetTopJobSuccess

alt [isGetTopJobSuccess]

```
SUnwinderManager_Callbacks unwinderManager_Callbacks;
```

alt [UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]

[004] unwinderManager_Callbacks.UpdateUWPaperThicknessForJob(
firstJobID, isReflectPaperThickness);

[005]

```
WriteToLogBuf(LOG_DEBUG, "CCtlJobList::Proc() error
    - Cannot get unwinder manager callback.");
```

```
WriteToLogBuf(LOG_DEBUG, "CCtlJobList::Proc() error
    - Cannot get top job of the print queue folder.");
```

```
WriteToLogBuf(LOG_DEBUG, "CCtlJobList::Proc() error
    - Cannot get job manager callback.");
```

```
iniPrinterDescriptor.Finalize();
```

~~CDataIF* pData = dynamic_cast<CDataIF*>(m_data);~~
~~// Start running the selection job~~
~~pData->StartJobRun_JobList();~~

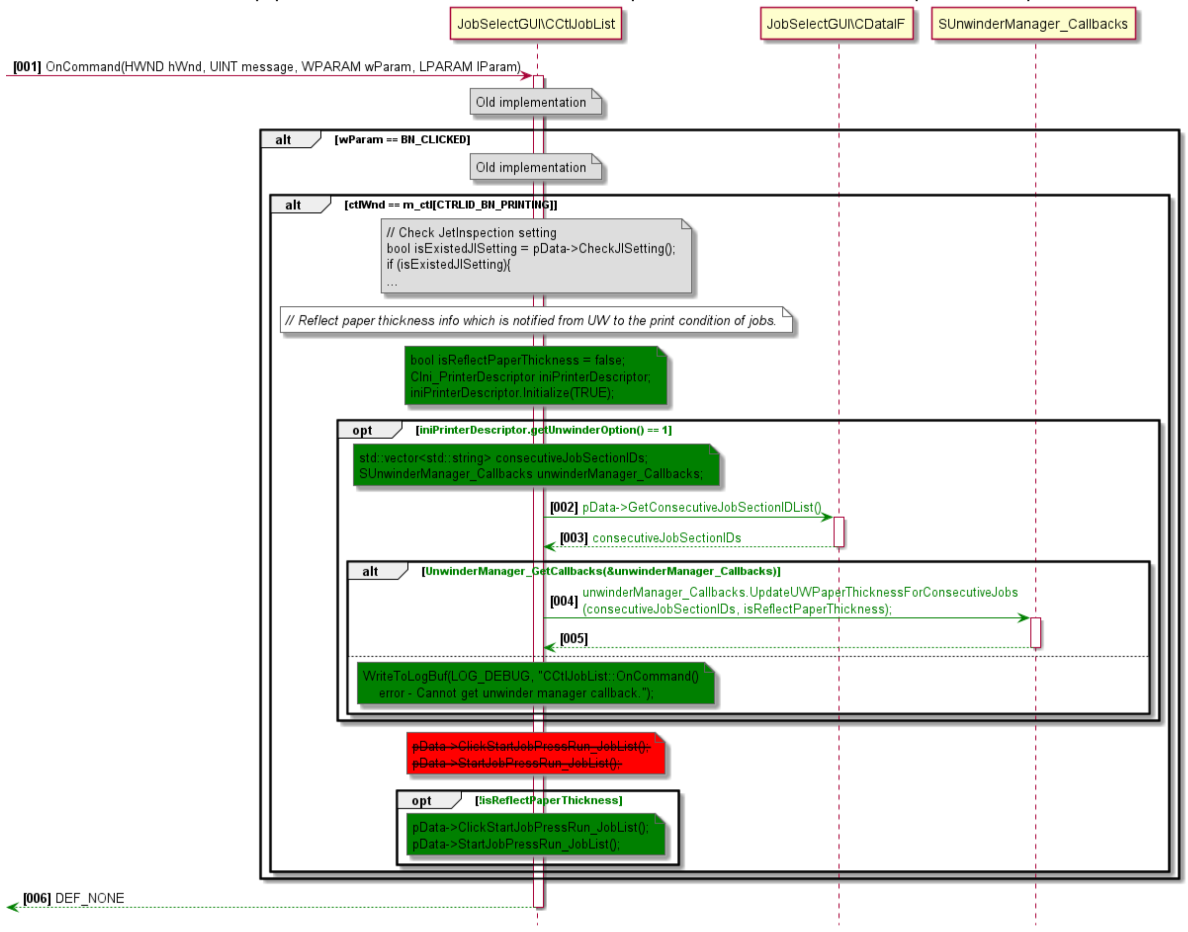
opt [!isReflectPaperThickness]

```
CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
// Start running the selection job
pData->StartJobRun_JobList();
```

Old implementation

[006] DEF_NONE

205.17 Reflect the paper thickness notified from UW to the current print condition when the continuous print button is pressed



[001] OnCommand(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

Old implementation

alt [wParam == BN_CLICKED]

Old implementation

alt [ctlWnd == m_ctl[CTRLID_BN_PRINTING]]

```
// Check JetInspection setting
bool isExistedJISetting = pData->CheckJISetting();
if (isExistedJISetting){
...
```

// Reflect paper thickness info which is notified from UW to the print condition of jobs.

```
bool isReflectPaperThickness = false;
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);
```

opt [iniPrinterDescriptor.getUnwinderOption() == 1]

```
std::vector<std::string> consecutiveJobSectionIDs;
SUnwinderManager_Callbacks unwinderManager_Callbacks;
```

[002] pData->GetConsecutiveJobSectionIDList()

[003] consecutiveJobSectionIDs

alt [UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]

[004] unwinderManager_Callbacks.UpdateUWPaperThicknessForConsecutiveJobs
(consecutiveJobSectionIDs, isReflectPaperThickness);

[005]

```
WriteToLogBuf(LOG_DEBUG, "CCtlJobList::OnCommand()
    error - Cannot get unwinder manager callback.");
```

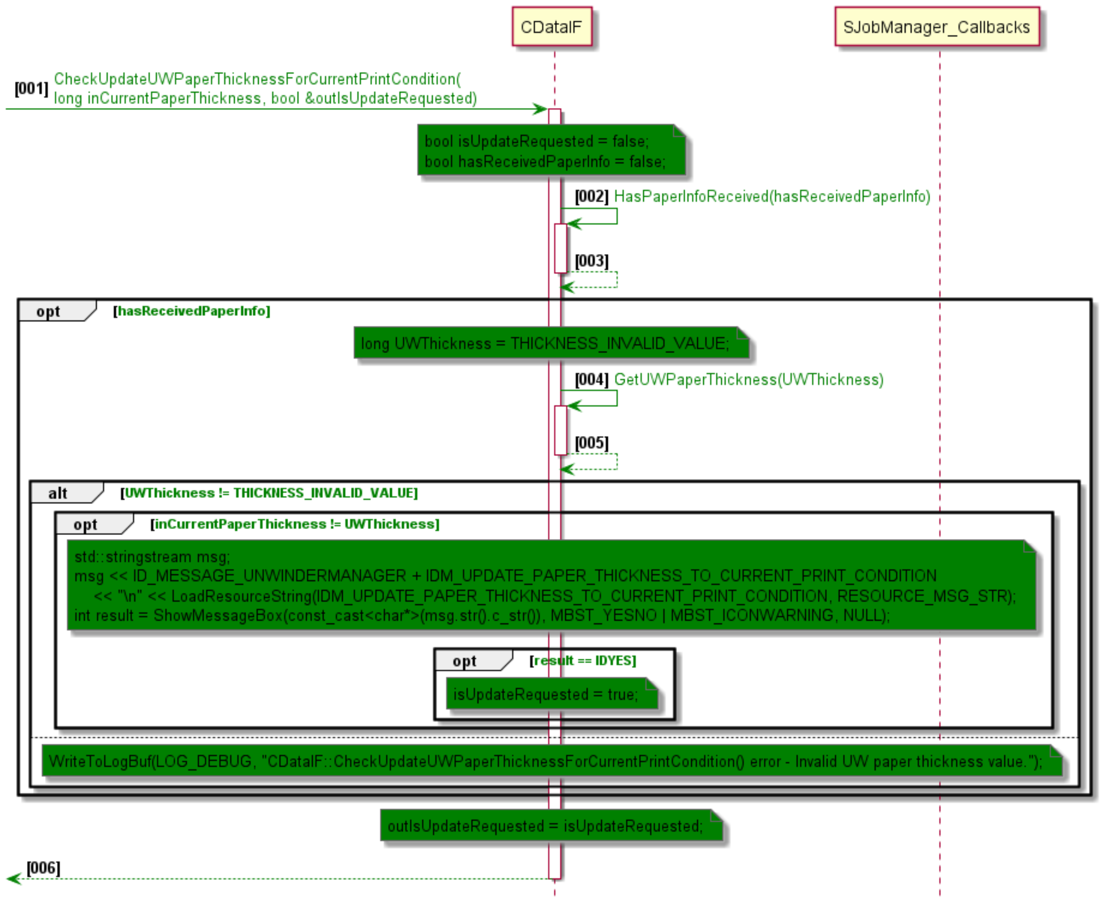
~~pData->ClickStartJobPressRun_JobList();~~
~~pData->StartJobPressRun_JobList();~~

opt [!isReflectPaperThickness]

```
pData->ClickStartJobPressRun_JobList();
pData->StartJobPressRun_JobList();
```

[006] DEF_NONE

205.18 Check whether or not update of paper thickness value notified from UW to the current print condition is requested.


CDataIF
SJobManager_Callbacks
[001] CheckUpdateUWPaperThicknessForCurrentPrintCondition(
long inCurrentPaperThickness, bool &outIsUpdateRequested)
[002] HasPaperInfoReceived(hasReceivedPaperInfo)
[003]
opt [hasReceivedPaperInfo]
[004] GetUWPaperThickness(UWThickness)
[005]
alt [UWThickness != THICKNESS_INVALID_VALUE]
opt [inCurrentPaperThickness != UWThickness]
opt [result == IDYES]
[006]

```
bool isUpdateRequested = false;
bool hasReceivedPaperInfo = false;
```

```
long UWThickness = THICKNESS_INVALID_VALUE;
```

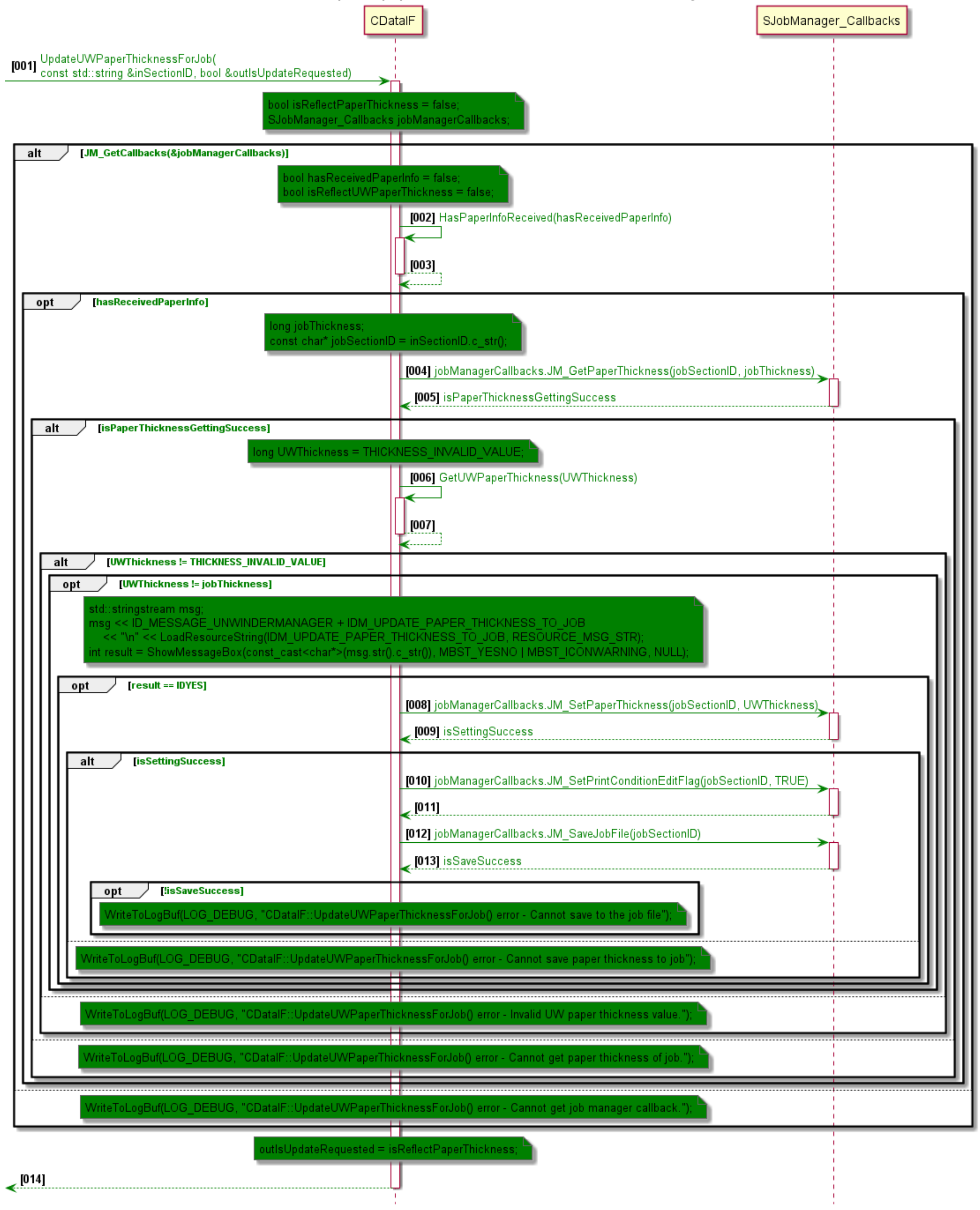
```
std::stringstream msg;
msg << ID_MESSAGE_UNWINDERMANAGER + IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION
    << "\n" << LoadResourceString(IDM_UPDATE_PAPER_THICKNESS_TO_CURRENT_PRINT_CONDITION, RESOURCE_MSG_STR);
int result = ShowMessageBox(const_cast<char*>(msg.str().c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
```

```
isUpdateRequested = true;
```

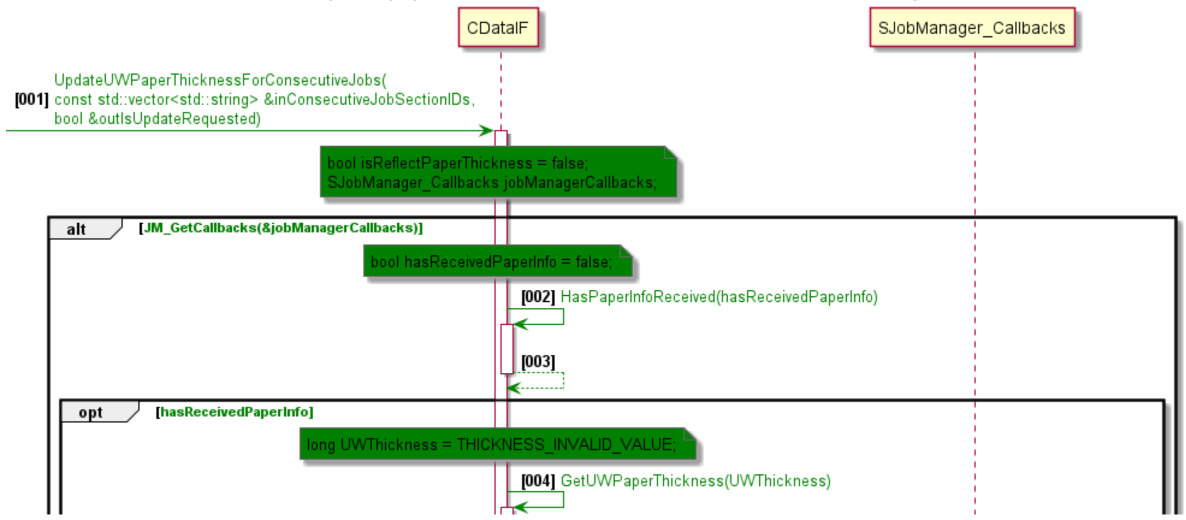
```
WriteToLogBuf(LOG_DEBUG, "CDataIF::CheckUpdateUWPaperThicknessForCurrentPrintCondition() error - Invalid UW paper thickness value.");
```

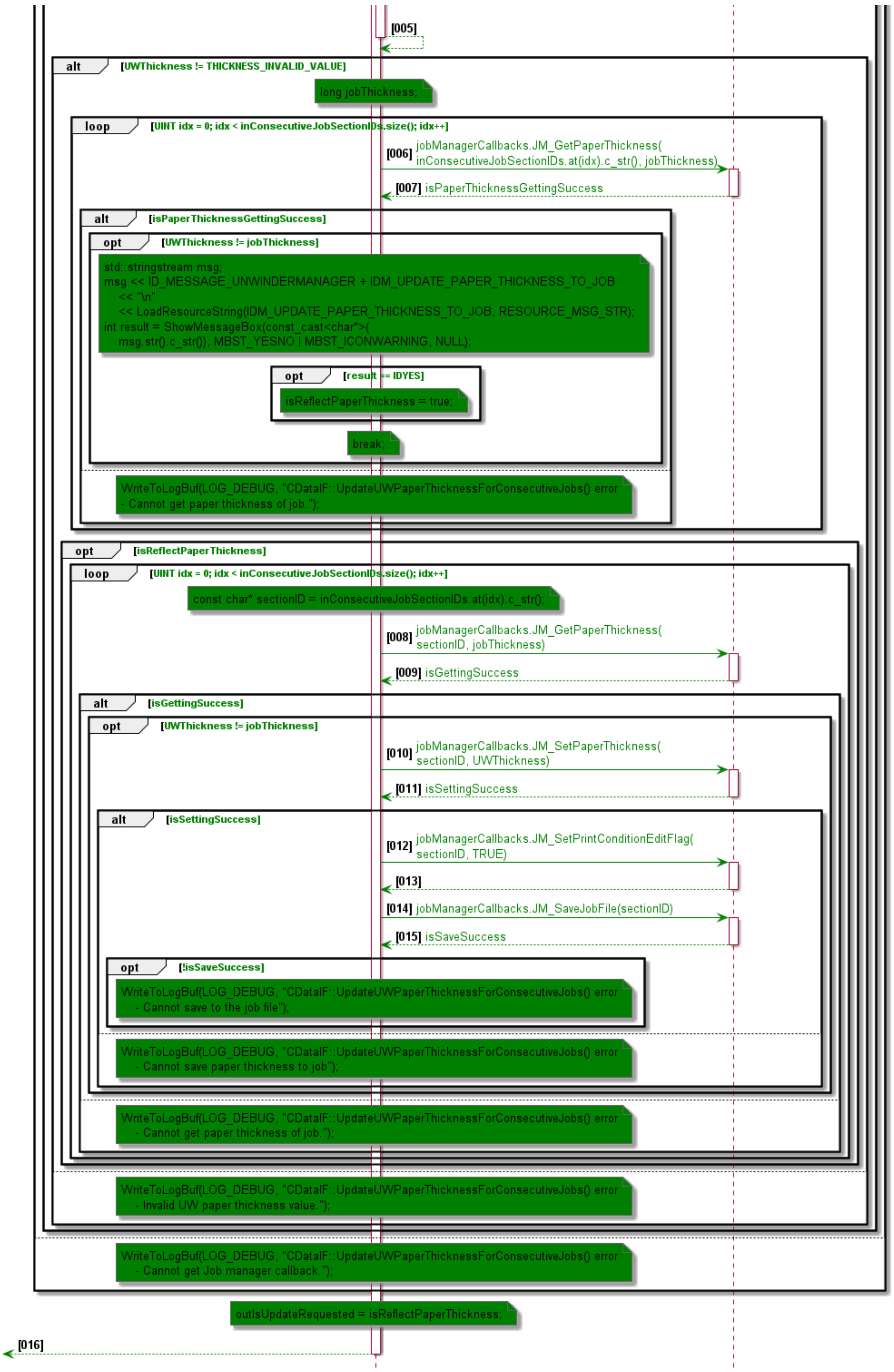
```
outIsUpdateRequested = isUpdateRequested;
```

205.19 Update paper thickness value notified from UW to a job


CDataIF
SJobManager_Callbacks
[001] UpdateUWPaperThicknessForJob(
const std::string &inSectionID, bool &outIsUpdateRequested)
bool isReflectPaperThickness = false;
SJobManager_Callbacks jobManagerCallbacks;
alt [JM_GetCallbacks(&jobManagerCallbacks)]
bool hasReceivedPaperInfo = false;
bool isReflectUWPaperThickness = false;
[002] HasPaperInfoReceived(hasReceivedPaperInfo)
[003]
opt [hasReceivedPaperInfo]
long jobThickness;
const char* jobSectionID = inSectionID.c_str();
[004] jobManagerCallbacks.JM_GetPaperThickness(jobSectionID, jobThickness)
[005] isPaperThicknessGettingSuccess
alt [isPaperThicknessGettingSuccess]
long UWThickness = THICKNESS_INVALID_VALUE;
[006] GetUWPaperThickness(UWThickness)
[007]
alt [UWThickness != THICKNESS_INVALID_VALUE]
opt [UWThickness != jobThickness]
std::stringstream msg;
msg << ID_MESSAGE_UNWINDERMANAGER + IDM_UPDATE_PAPER_THICKNESS_TO_JOB
<< "\n" << LoadResourceString(IDM_UPDATE_PAPER_THICKNESS_TO_JOB, RESOURCE_MSG_STR);
int result = ShowMessageBox(const_cast<char*>(msg.str().c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
opt [result == IDYES]
[008] jobManagerCallbacks.JM_SetPaperThickness(jobSectionID, UWThickness)
[009] isSettingSuccess
alt [isSettingSuccess]
[010] jobManagerCallbacks.JM_SetPrintConditionEditFlag(jobSectionID, TRUE)
[011]
[012] jobManagerCallbacks.JM_SaveJobFile(jobSectionID)
[013] isSaveSuccess
opt [!isSaveSuccess]
WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForJob() error - Cannot save to the job file");
WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForJob() error - Cannot save paper thickness to job");
WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForJob() error - Invalid UW paper thickness value.");
WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForJob() error - Cannot get paper thickness of job.");
WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForJob() error - Cannot get job manager callback.");
outIsUpdateRequested = isReflectPaperThickness;
[014]

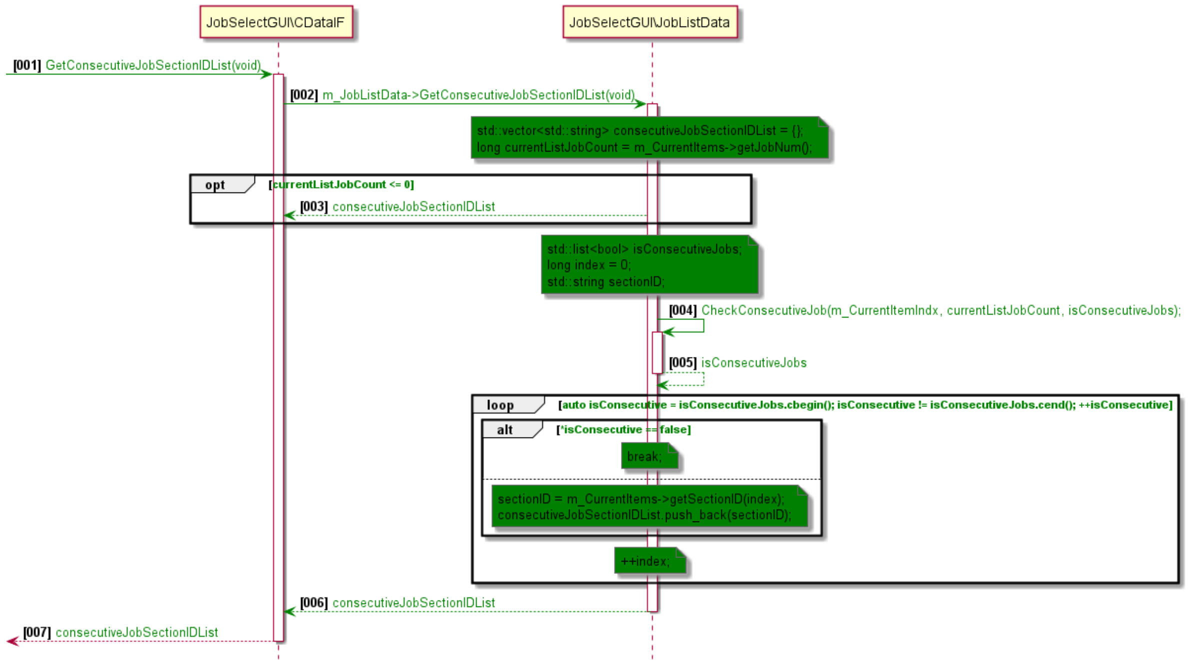
205.20 Update paper thickness value notified from UW to consecutive jobs


CDataIF
SJobManager_Callbacks
[001] UpdateUWPaperThicknessForConsecutiveJobs(
const std::vector<std::string> &inConsecutiveJobSectionIDs,
bool &outIsUpdateRequested)
bool isReflectPaperThickness = false;
SJobManager_Callbacks jobManagerCallbacks;
alt [JM_GetCallbacks(&jobManagerCallbacks)]
bool hasReceivedPaperInfo = false;
[002] HasPaperInfoReceived(hasReceivedPaperInfo)
[003]
opt [hasReceivedPaperInfo]
long UWThickness = THICKNESS_INVALID_VALUE;
[004] GetUWPaperThickness(UWThickness)

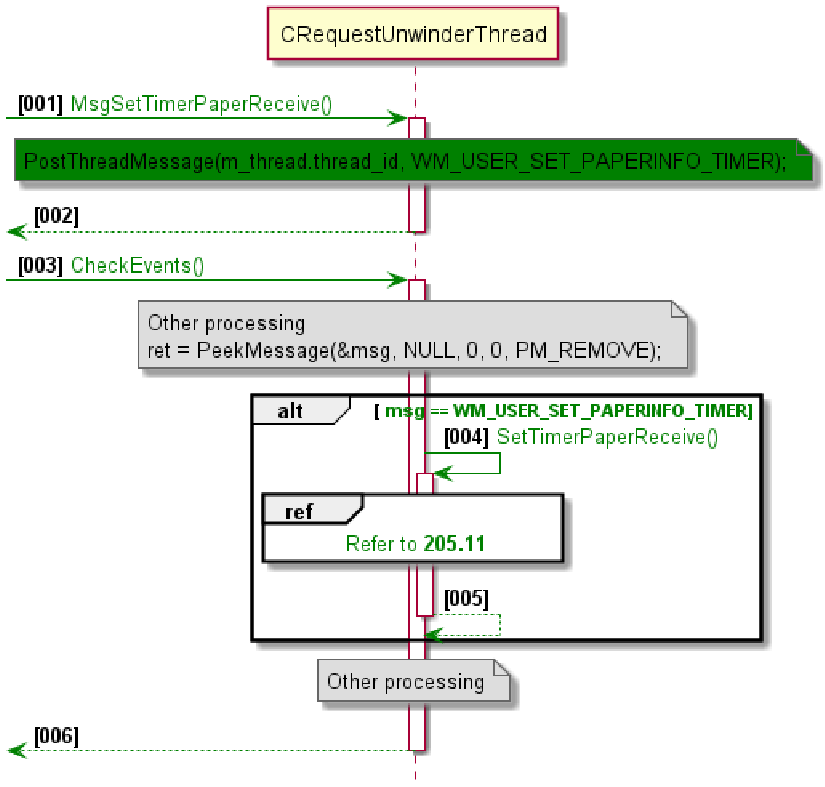


```
[005]
alt [UWThickness != THICKNESS_INVALID_VALUE]
    long jobThickness;
    loop [UINT idx = 0; idx < inConsecutiveJobSectionIDs.size(); idx++]
        [006] jobManagerCallbacks.JM_GetPaperThickness(
              inConsecutiveJobSectionIDs.at(idx).c_str(), jobThickness)
        [007] isPaperThicknessGettingSuccess
        alt [isPaperThicknessGettingSuccess]
            opt [UWThickness != jobThickness]
                std::stringstream msg;
                msg << IDM_MESSAGE_UNWINDERMANAGER + IDM_UPDATE_PAPER_THICKNESS_TO_JOB
                    << "\n"
                    << LoadResourceString(IDM_UPDATE_PAPER_THICKNESS_TO_JOB, RESOURCE_MSG_STR);
                int result = ShowMessageBox(const_cast<char*>(
                    msg.str().c_str()), MBST_YESNO | MBST_ICONWARNING, NULL);
                opt [result == IDYES]
                    isReflectPaperThickness = true;
                break;
        ----
            WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
            - Cannot get paper thickness of job.");
    opt [isReflectPaperThickness]
        loop [UINT idx = 0; idx < inConsecutiveJobSectionIDs.size(); idx++]
            const char* sectionID = inConsecutiveJobSectionIDs.at(idx).c_str();
            [008] jobManagerCallbacks.JM_GetPaperThickness(
                  sectionID, jobThickness)
            [009] isGettingSuccess
            alt [isGettingSuccess]
                opt [UWThickness != jobThickness]
                    [010] jobManagerCallbacks.JM_SetPaperThickness(
                          sectionID, UWThickness)
                    [011] isSettingSuccess
                    alt [isSettingSuccess]
                        [012] jobManagerCallbacks.JM_SetPrintConditionEditFlag(
                              sectionID, TRUE)
                        [013]
                        [014] jobManagerCallbacks.JM_SaveJobFile(sectionID)
                        [015] isSaveSuccess
                        opt [!isSaveSuccess]
                            WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
                              - Cannot save to the job file");
                    ----
                        WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
                          - Cannot save paper thickness to job");
            ----
                WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
                  - Cannot get paper thickness of job.");
----
    WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
      - Invalid UW paper thickness value.");
----
    WriteToLogBuf(LOG_DEBUG, "CDataIF::UpdateUWPaperThicknessForConsecutiveJobs() error
      - Cannot get Job manager callback.");
outIsUpdateRequested = isReflectPaperThickness;
[016]
```

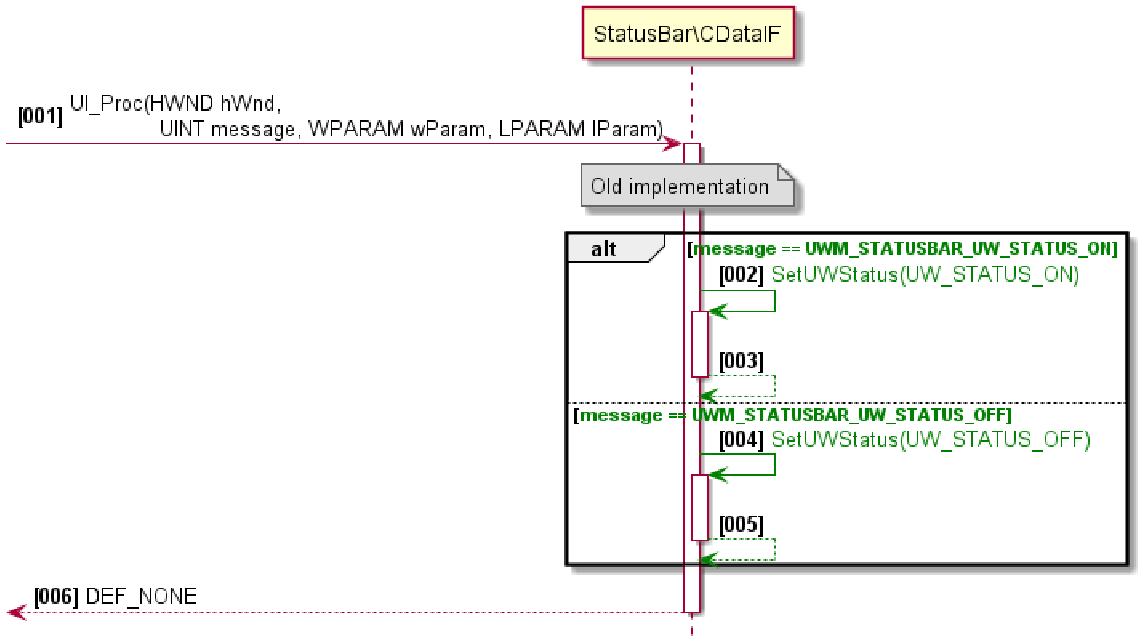
205.21 Get a list of consecutive job section IDs


JobSelectGUI\CDataIF
JobSelectGUI\JobListData
[001] GetConsecutiveJobSectionIDList(void)
[002] m_JobListData->GetConsecutiveJobSectionIDList(void)
std::vector<std::string> consecutiveJobSectionIDList = {};
long currentListJobCount = m_CurrentItems->getJobNum();
opt
[currentListJobCount <= 0]
[003] consecutiveJobSectionIDList
std::list<bool> isConsecutiveJobs;
long index = 0;
std::string sectionID;
[004] CheckConsecutiveJob(m_CurrentItemIndx, currentListJobCount, isConsecutiveJobs);
[005] isConsecutiveJobs
loop
[auto isConsecutive = isConsecutiveJobs.cbegin(); isConsecutive != isConsecutiveJobs.cend(); ++isConsecutive]
alt
[*isConsecutive == false]
break;
sectionID = m_CurrentItems->getSectionID(index);
consecutiveJobSectionIDList.push_back(sectionID);
++index;
[006] consecutiveJobSectionIDList
[007] consecutiveJobSectionIDList

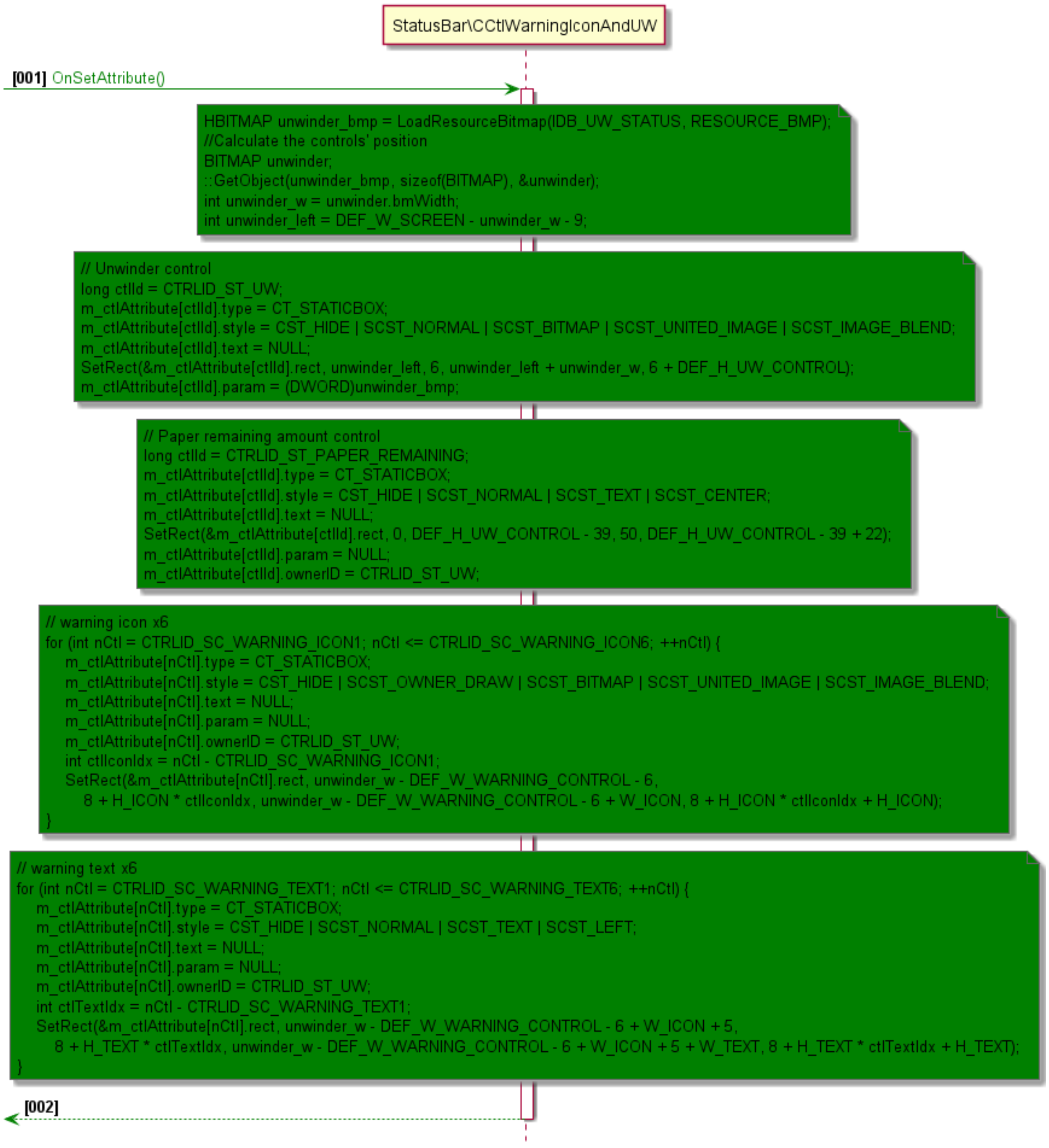
205.22 Notify main thread to update timer when receiving SignalStatus(PAPER)


CRequestUnwinderThread
[001] MsgSetTimerPaperReceive()
PostThreadMessage(m_thread.thread_id, WM_USER_SET_PAPERINFO_TIMER);
[002]
[003] CheckEvents()
Other processing
ret = PeekMessage(&msg, NULL, 0, 0, PM_REMOVE);
alt
[msg == WM_USER_SET_PAPERINFO_TIMER]
[004] SetTimerPaperReceive()
ref
Refer to 205.11
[005]
Other processing
[006]

205.23 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

205.24 Set the unwinder control's property



```
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 - 9;
```

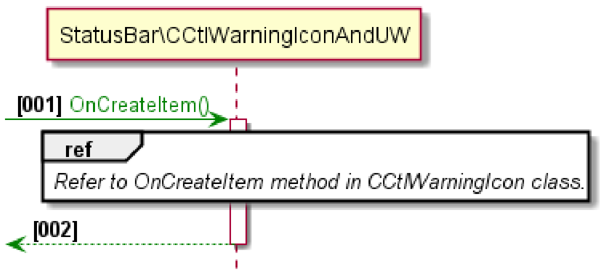
```
// Unwinder control
long ctlId = CTRLID_ST_UW;
m_ctlAttribute[ctlId].type = CT_STATICBOX;
m_ctlAttribute[ctlId].style = CST_HIDE | SCST_NORMAL | SCST_BITMAP | SCST_UNITED_IMAGE | SCST_IMAGE_BLEND;
m_ctlAttribute[ctlId].text = NULL;
SetRect(&m_ctlAttribute[ctlId].rect, unwinder_left, 6, unwinder_left + unwinder_w, 6 + DEF_H_UW_CONTROL);
m_ctlAttribute[ctlId].param = (DWORD)unwinder_bmp;
```

```
// Paper remaining amount control
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, 0, DEF_H_UW_CONTROL - 39, 50, DEF_H_UW_CONTROL - 39 + 22);
m_ctlAttribute[ctlId].param = NULL;
m_ctlAttribute[ctlId].ownerID = CTRLID_ST_UW;
```

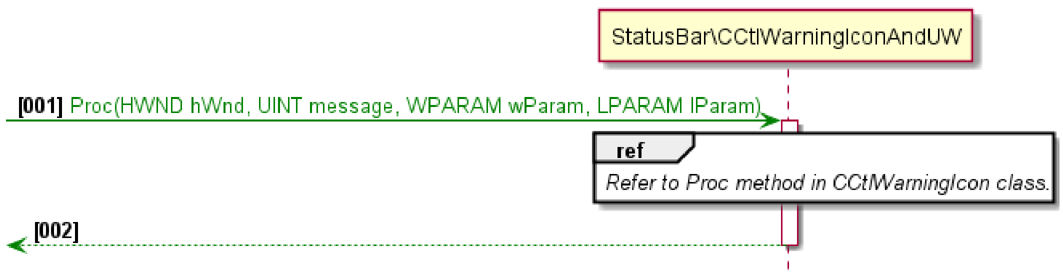
```
// warning icon x6
for (int nCtl = CTRLID_SC_WARNING_ICON1; nCtl <= CTRLID_SC_WARNING_ICON6; ++nCtl) {
    m_ctlAttribute[nCtl].type = CT_STATICBOX;
    m_ctlAttribute[nCtl].style = CST_HIDE | SCST_OWNER_DRAW | SCST_BITMAP | SCST_UNITED_IMAGE | SCST_IMAGE_BLEND;
    m_ctlAttribute[nCtl].text = NULL;
    m_ctlAttribute[nCtl].param = NULL;
    m_ctlAttribute[nCtl].ownerID = CTRLID_ST_UW;
    int ctlIconIdx = nCtl - CTRLID_SC_WARNING_ICON1;
    SetRect(&m_ctlAttribute[nCtl].rect, unwinder_w - DEF_W_WARNING_CONTROL - 6,
        8 + H_ICON * ctlIconIdx, unwinder_w - DEF_W_WARNING_CONTROL - 6 + W_ICON, 8 + H_ICON * ctlIconIdx + H_ICON);
}
```

```
// warning text x6
for (int nCtl = CTRLID_SC_WARNING_TEXT1; nCtl <= CTRLID_SC_WARNING_TEXT6; ++nCtl) {
    m_ctlAttribute[nCtl].type = CT_STATICBOX;
    m_ctlAttribute[nCtl].style = CST_HIDE | SCST_NORMAL | SCST_TEXT | SCST_LEFT;
    m_ctlAttribute[nCtl].text = NULL;
    m_ctlAttribute[nCtl].param = NULL;
    m_ctlAttribute[nCtl].ownerID = CTRLID_ST_UW;
    int ctlTextIdx = nCtl - CTRLID_SC_WARNING_TEXT1;
    SetRect(&m_ctlAttribute[nCtl].rect, unwinder_w - DEF_W_WARNING_CONTROL - 6 + W_ICON + 5,
        8 + H_TEXT * ctlTextIdx, unwinder_w - DEF_W_WARNING_CONTROL - 6 + W_ICON + 5 + W_TEXT, 8 + H_TEXT * ctlTextIdx + H_TEXT);
}
```

205.25 Events that set the properties of the control

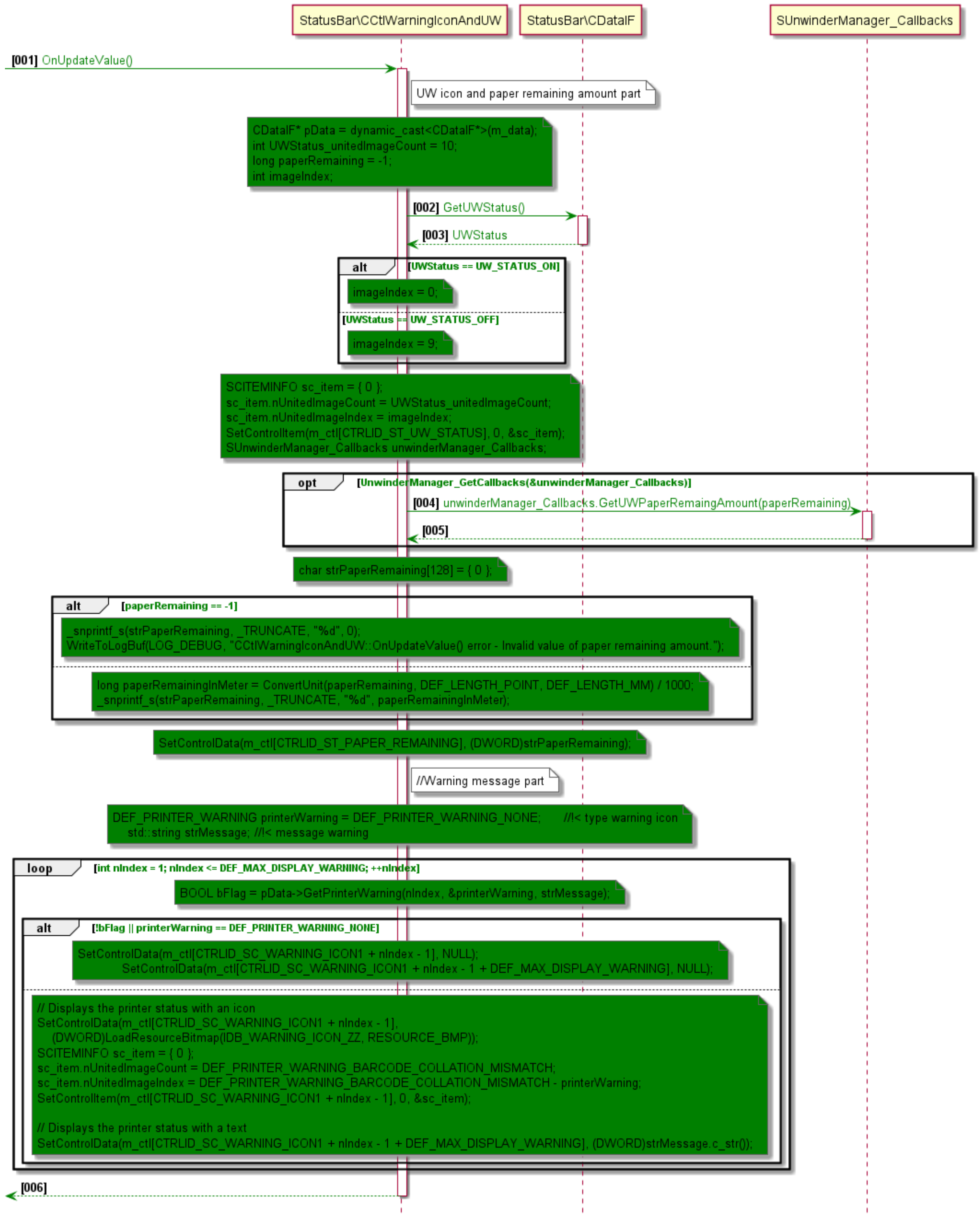

StatusBar\CCtlWarningIconAndUW
[001] OnCreateItem()
ref
Refer to OnCreateItem method in CCtlWarningIcon class.
[002]

205.26 Window procedure



StatusBar\CCtlWarningIconAndUW
[001] Proc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
ref
Refer to Proc method in CCtlWarningIcon class.
[002]

205.27 Update unwinder control display value



StatusBar\CCtlWarningIconAndUW
StatusBar\CDataIF
SUnwinderManager_Callbacks
[001] OnUpdateValue()
UW icon and paper remaining amount part
[002] GetUWStatus()
[003] UWStatus
alt
[UWStatus == UW_STATUS_ON]
[UWStatus == UW_STATUS_OFF]
opt
[UnwinderManager_GetCallbacks(&unwinderManager_Callbacks)]
[004] unwinderManager_Callbacks.GetUWPaperRemaingAmount(paperRemaining)
[005]
alt
[paperRemaining == -1]
//Warning message part
loop
[int nIndex = 1; nIndex <= DEF_MAX_DISPLAY_WARNING; ++nIndex]
alt
[!bFlag || printerWarning == DEF_PRINTER_WARNING_NONE]
[006]

```
CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
int UWStatus_unitedImageCount = 10;
long paperRemaining = -1;
int imageIndex;
```

```
imageIndex = 0;
```

```
imageIndex = 9;
```

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

```
char strPaperRemaining[128] = { 0 };
```

```
_snprintf_s(strPaperRemaining, _TRUNCATE, "%d", 0);
WriteToLogBuf(LOG_DEBUG, "CCtlWarningIconAndUW::OnUpdateValue() error - Invalid value of paper remaining amount.");
```

```
long paperRemainingInMeter = ConvertUnit(paperRemaining, DEF_LENGTH_POINT, DEF_LENGTH_MM) / 1000;
_snprintf_s(strPaperRemaining, _TRUNCATE, "%d", paperRemainingInMeter);
```

```
SetControlData(m_ctl[CTRLID_ST_PAPER_REMAINING], (DWORD)strPaperRemaining);
```

```
DEF_PRINTER_WARNING printerWarning = DEF_PRINTER_WARNING_NONE;      //!< type warning icon
    std::string strMessage; //!< message warning
```

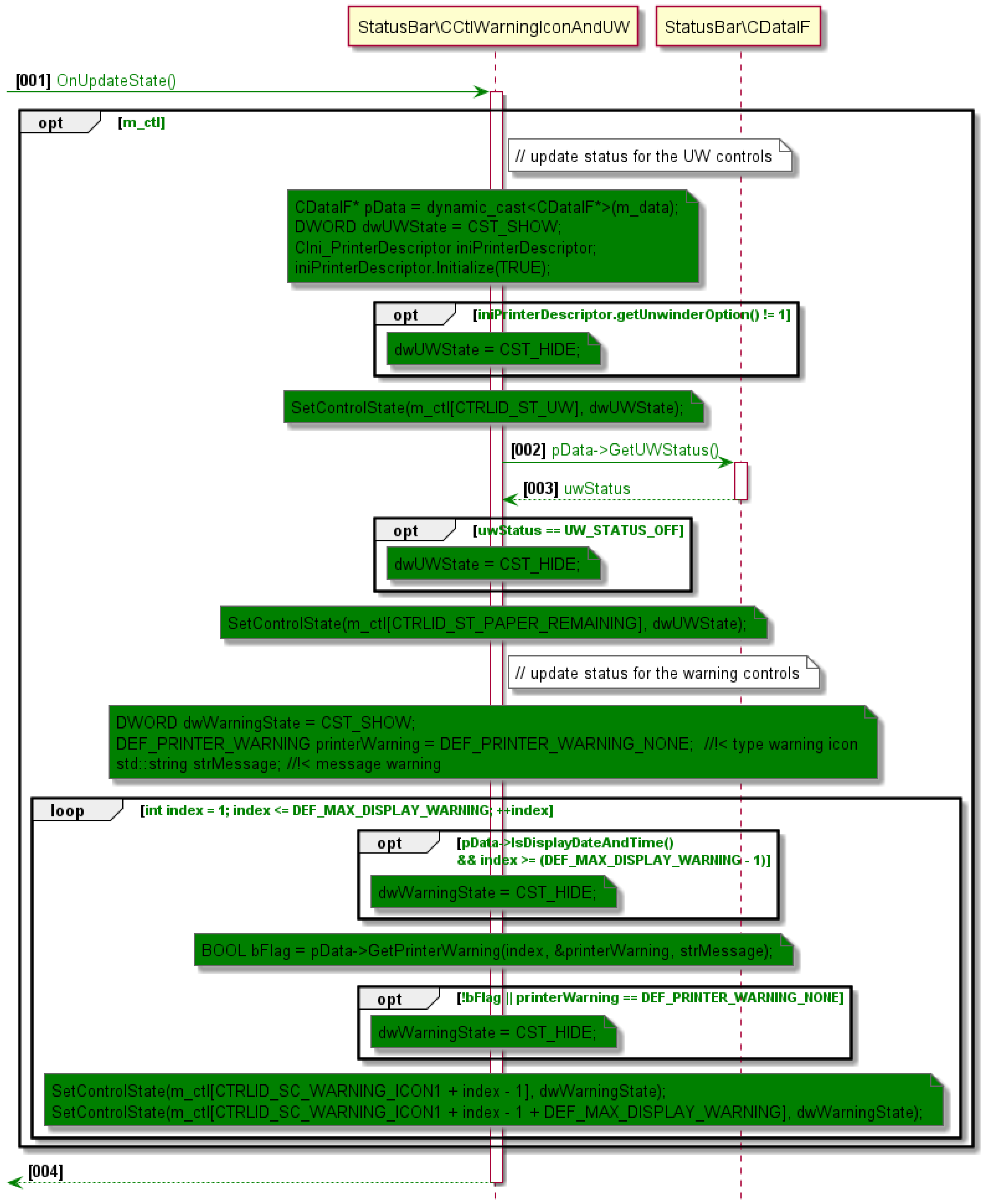
```
BOOL bFlag = pData->GetPrinterWarning(nIndex, &printerWarning, strMessage);
```

```
SetControlData(m_ctl[CTRLID_SC_WARNING_ICON1 + nIndex - 1], NULL);
            SetControlData(m_ctl[CTRLID_SC_WARNING_ICON1 + nIndex - 1 + DEF_MAX_DISPLAY_WARNING], NULL);
```

```
// Displays the printer status with an icon
SetControlData(m_ctl[CTRLID_SC_WARNING_ICON1 + nIndex - 1],
    (DWORD)LoadResourceBitmap(IDB_WARNING_ICON_ZZ, RESOURCE_BMP));
SCITEMINFO sc_item = { 0 };
sc_item.nUnitedImageCount = DEF_PRINTER_WARNING_BARCODE_COLLATION_MISMATCH;
sc_item.nUnitedImageIndex = DEF_PRINTER_WARNING_BARCODE_COLLATION_MISMATCH - printerWarning;
SetControlItem(m_ctl[CTRLID_SC_WARNING_ICON1 + nIndex - 1], 0, &sc_item);

// Displays the printer status with a text
SetControlData(m_ctl[CTRLID_SC_WARNING_ICON1 + nIndex - 1 + DEF_MAX_DISPLAY_WARNING], (DWORD)strMessage.c_str());
```

205.28 Update unwinder control status update event


StatusBar\CCtlWarningIconAndUW
StatusBar\CDataIF
[001] OnUpdateState()
opt [m_ctl]
// update status for the UW controls
opt [iniPrinterDescriptor.getUnwinderOption() != 1]
[002] pData->GetUWStatus()
[003] uwStatus
opt [uwStatus == UW_STATUS_OFF]
// update status for the warning controls
loop [int index = 1; index <= DEF_MAX_DISPLAY_WARNING; ++index]
opt [pData->IsDisplayDateAndTime()
&& index >= (DEF_MAX_DISPLAY_WARNING - 1)]
opt [!bFlag || printerWarning == DEF_PRINTER_WARNING_NONE]
[004]

```
CDataIF* pData = dynamic_cast<CDataIF*>(m_data);
DWORD dwUWState = CST_SHOW;
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);
```

```
dwUWState = CST_HIDE;
```

```
SetControlState(m_ctl[CTRLID_ST_UW], dwUWState);
```

```
dwUWState = CST_HIDE;
```

```
SetControlState(m_ctl[CTRLID_ST_PAPER_REMAINING], dwUWState);
```

```
DWORD dwWarningState = CST_SHOW;
DEF_PRINTER_WARNING printerWarning = DEF_PRINTER_WARNING_NONE;  //!< type warning icon
std::string strMessage; //!< message warning
```

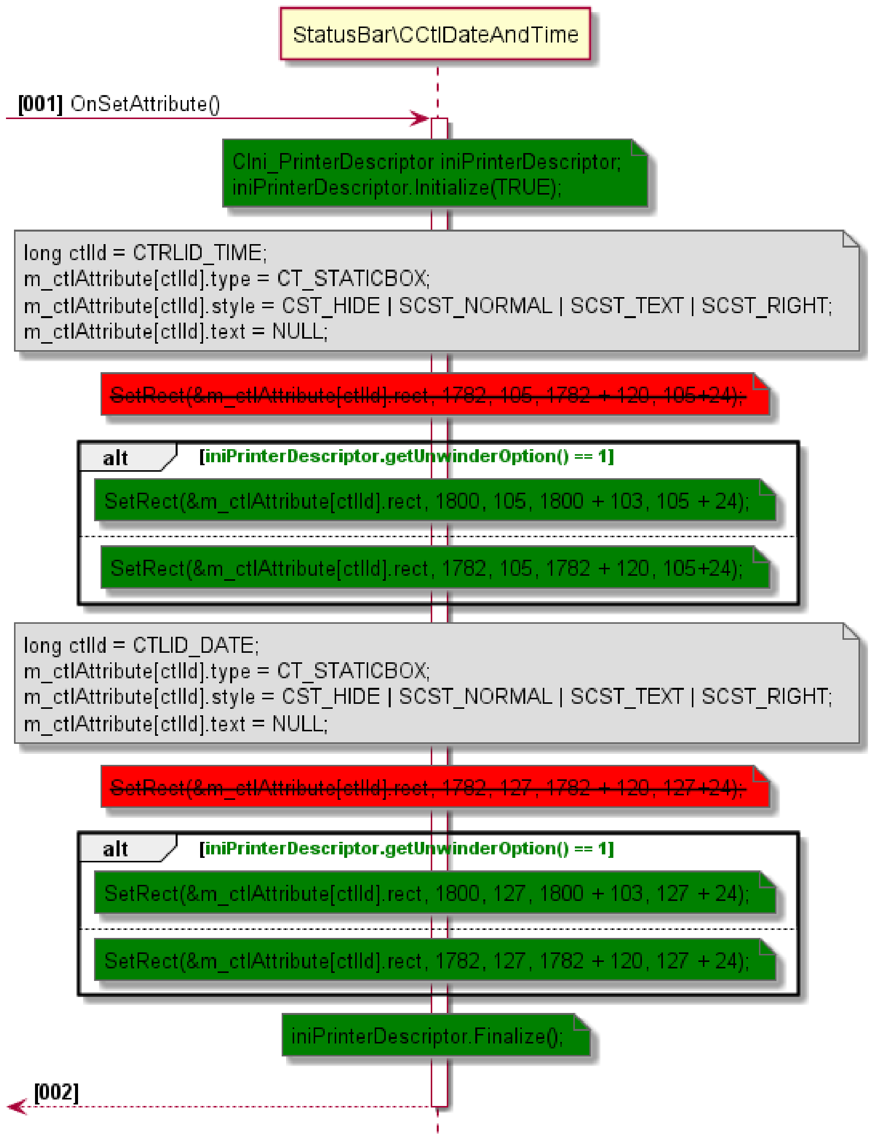
```
dwWarningState = CST_HIDE;
```

```
BOOL bFlag = pData->GetPrinterWarning(index, &printerWarning, strMessage);
```

```
dwWarningState = CST_HIDE;
```

```
SetControlState(m_ctl[CTRLID_SC_WARNING_ICON1 + index - 1], dwWarningState);
SetControlState(m_ctl[CTRLID_SC_WARNING_ICON1 + index - 1 + DEF_MAX_DISPLAY_WARNING], dwWarningState);
```

205.29 Set data and time control's property


StatusBar\CCtlDateAndTime
[001] OnSetAttribute()
alt
[iniPrinterDescriptor.getUnwinderOption() == 1]
alt
[iniPrinterDescriptor.getUnwinderOption() == 1]
[002]

```
CIni_PrinterDescriptor iniPrinterDescriptor;
iniPrinterDescriptor.Initialize(TRUE);

long ctlId = CTRLID_TIME;
m_ctlAttribute[ctlId].type = CT_STATICBOX;
m_ctlAttribute[ctlId].style = CST_HIDE | SCST_NORMAL | SCST_TEXT | SCST_RIGHT;
m_ctlAttribute[ctlId].text = NULL;
```

~~SetRect(&m_ctlAttribute[ctlId].rect, 1782, 105, 1782 + 120, 105+24);~~

```
SetRect(&m_ctlAttribute[ctlId].rect, 1800, 105, 1800 + 103, 105 + 24);
SetRect(&m_ctlAttribute[ctlId].rect, 1782, 105, 1782 + 120, 105+24);

long ctlId = CTLID_DATE;
m_ctlAttribute[ctlId].type = CT_STATICBOX;
m_ctlAttribute[ctlId].style = CST_HIDE | SCST_NORMAL | SCST_TEXT | SCST_RIGHT;
m_ctlAttribute[ctlId].text = NULL;
```

~~SetRect(&m_ctlAttribute[ctlId].rect, 1782, 127, 1782 + 120, 127+24);~~

```
SetRect(&m_ctlAttribute[ctlId].rect, 1800, 127, 1800 + 103, 127 + 24);
SetRect(&m_ctlAttribute[ctlId].rect, 1782, 127, 1782 + 120, 127 + 24);

iniPrinterDescriptor.Finalize();
```

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 +α seconds, it is determined that the UW has not started.
+α seconds are defined in the TP-UW_Communication.ini file

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

・Display the translucent UW icon(Top icon of UW2.bmp).

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(Top icon of UW2.bmp)

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 (Top icon of UW2.bmp) of the UW icon to the normal UW icon.

2. Solution

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

Resource\English\strings_UnwinderManager.ini

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

Resource\Japanese\strings_UnwinderManager.ini

```
[MSG]
IDM_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/HeadCleanir

    //
    UWM_OP_MAX_COUNT
};

// after
enum
{

    //...
    UWM_ADJUSTMENTGUI_CLOSE_DIALOG,         //!< Adjustment screen (Alignment/Shading/HeadCleanir

    UWM_STATUSBAR_UW_STATUS_ON,             //!< UW status is online, then notify to display norm
    UWM_STATUSBAR_UW_STATUS_OFF,            //!< UW status is offine, then notify to display trar

    //
    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

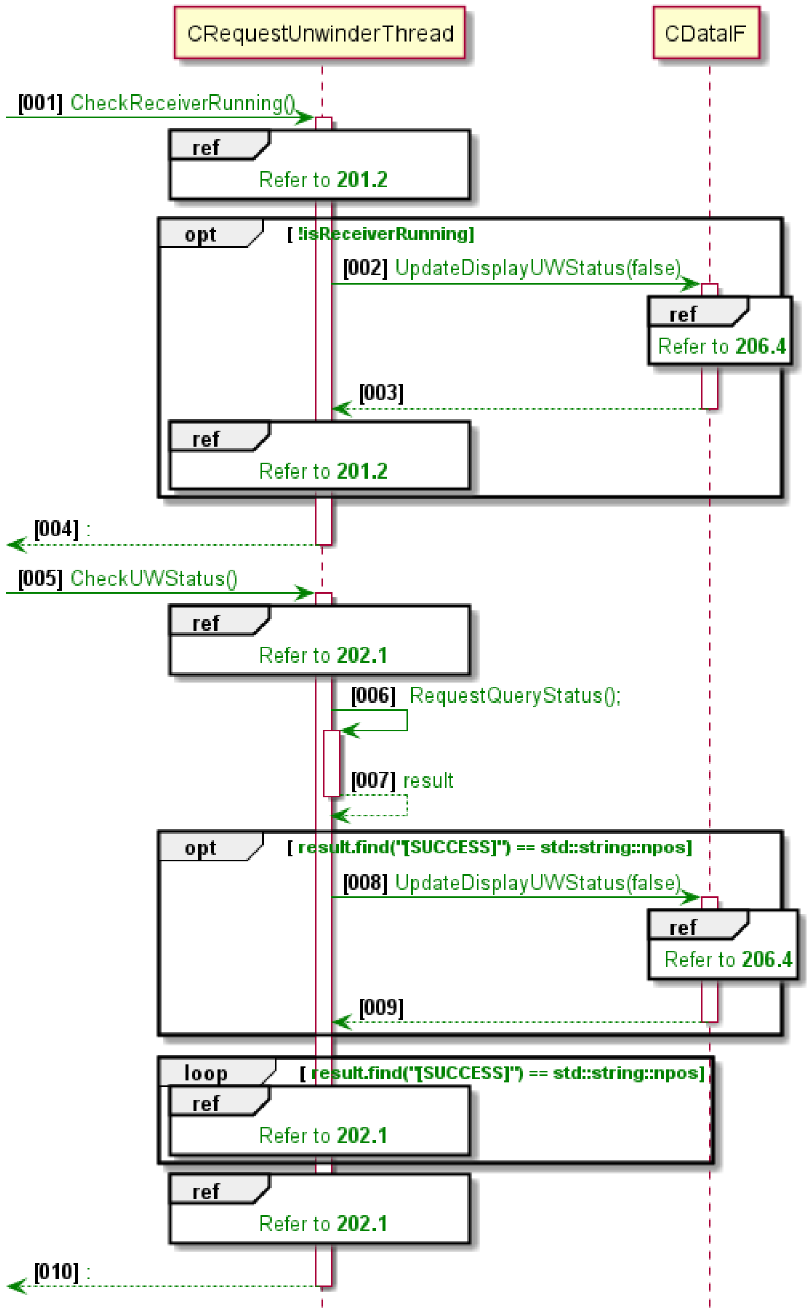
- In method CReceiveSignalStatus::ReceiveStatusInfo:
    - Stop the current timeout timer and start a new one.
    - Set the UW status into UnwinderManager_work.ini file.
    - call method UpdateDisplayUWStatus(true) to display UW is online
- In class CRequestUnwinderThread, add function to delete/set timer to check signal timeout.

RequestUnwinderThread.h

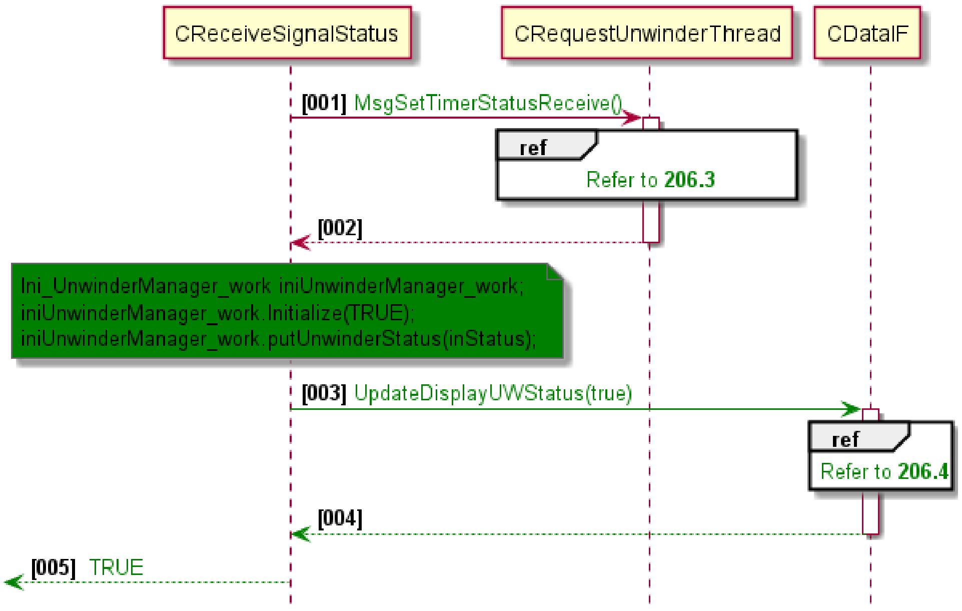
```
#define WM_USER_SET_STATUS_TIMER                WM_USER+100
class CRequestUnwinderThread
{
    ...
    // sent msg from ReceiveSignalStatus thread
        void MsgSetTimerStatusReceive();
    // setting for timeout timer of SignalStatus(STATUS)
    void SetTimerStatusReceive();
    // stop timeout timer of SignalStatus(STATUS)
    void KillTimerStatusReceive();
    ...
}
```

3. Detail implementation

206.1 Check UW status when starting control

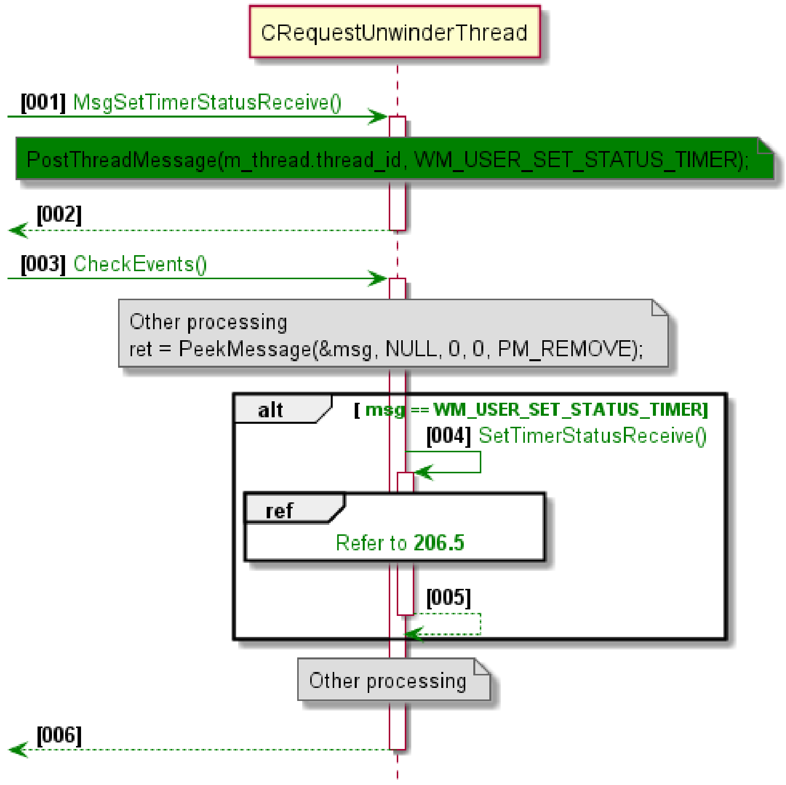


206.2 Receive status info



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

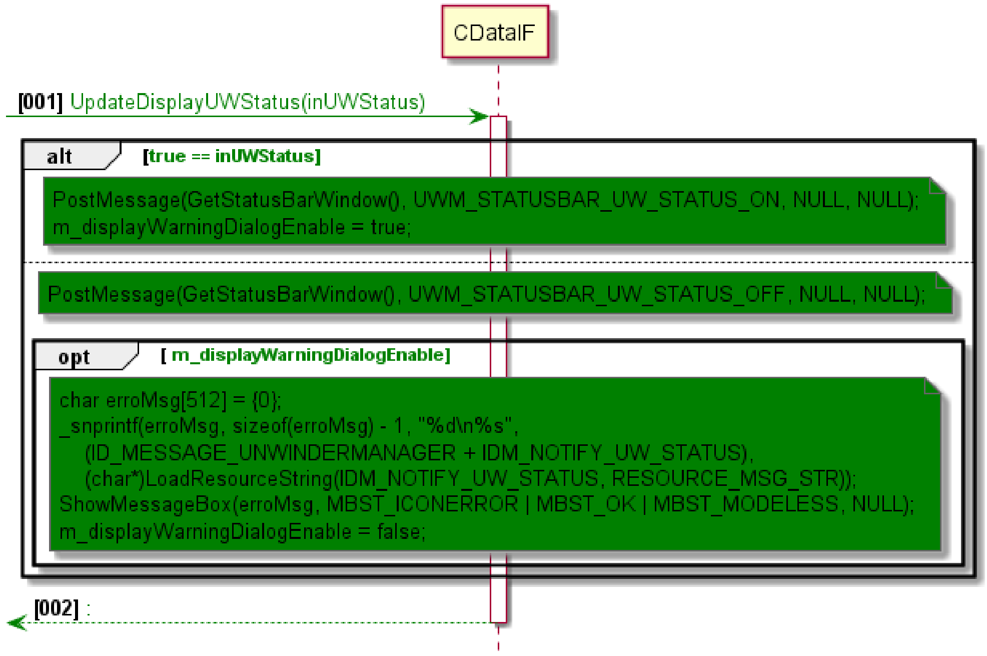
206.3 Notify main thread to update timer when receiving SignalStatus(STATUS)


CRequestUnwinderThread
[001] MsgSetTimerStatusReceive()
[002]
[003] CheckEvents()
alt
[msg == WM_USER_SET_STATUS_TIMER]
[004] SetTimerStatusReceive()
ref
Refer to 206.5
[005]
Other processing
[006]

```
PostThreadMessage(m_thread.thread_id, WM_USER_SET_STATUS_TIMER);
```

```
Other processing
ret = PeekMessage(&msg, NULL, 0, 0, PM_REMOVE);
```

206.4 Update display the UW status (in plugin UnwinderManager)

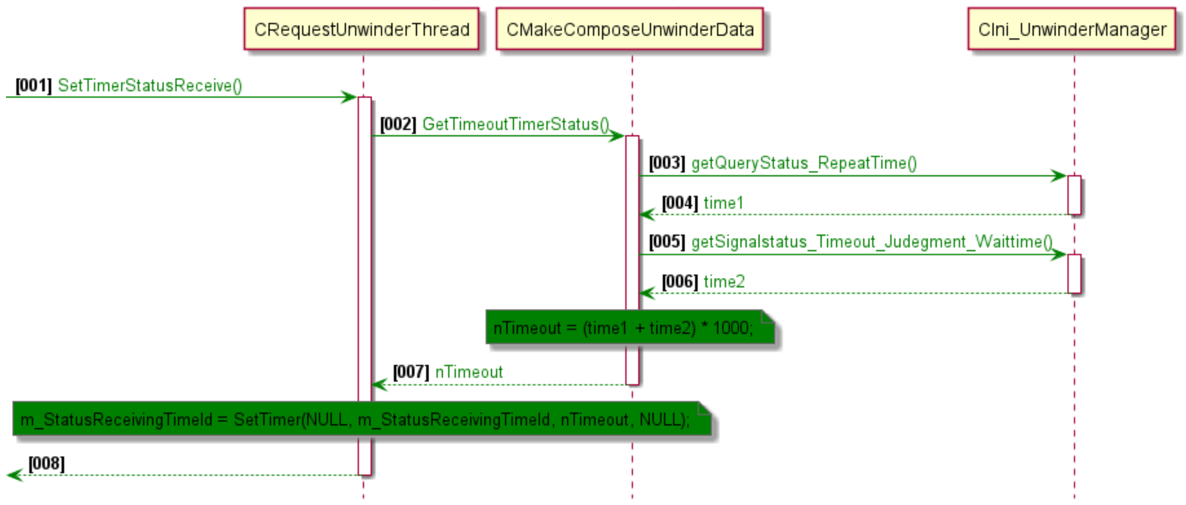

CDataIF
[001] UpdateDisplayUWStatus(inUWStatus)
alt
[true == inUWStatus]
opt
[m_displayWarningDialogEnable]
[002] :

```
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, "%d\n%s",
    (ID_MESSAGE_UNWINDERMANAGER + IDM_NOTIFY_UW_STATUS),
    (char*)LoadResourceString(IDM_NOTIFY_UW_STATUS, RESOURCE_MSG_STR));
ShowMessageBox(erroMsg, MBST_ICONERROR | MBST_OK | MBST_MODELESS, NULL);
m_displayWarningDialogEnable = false;
```

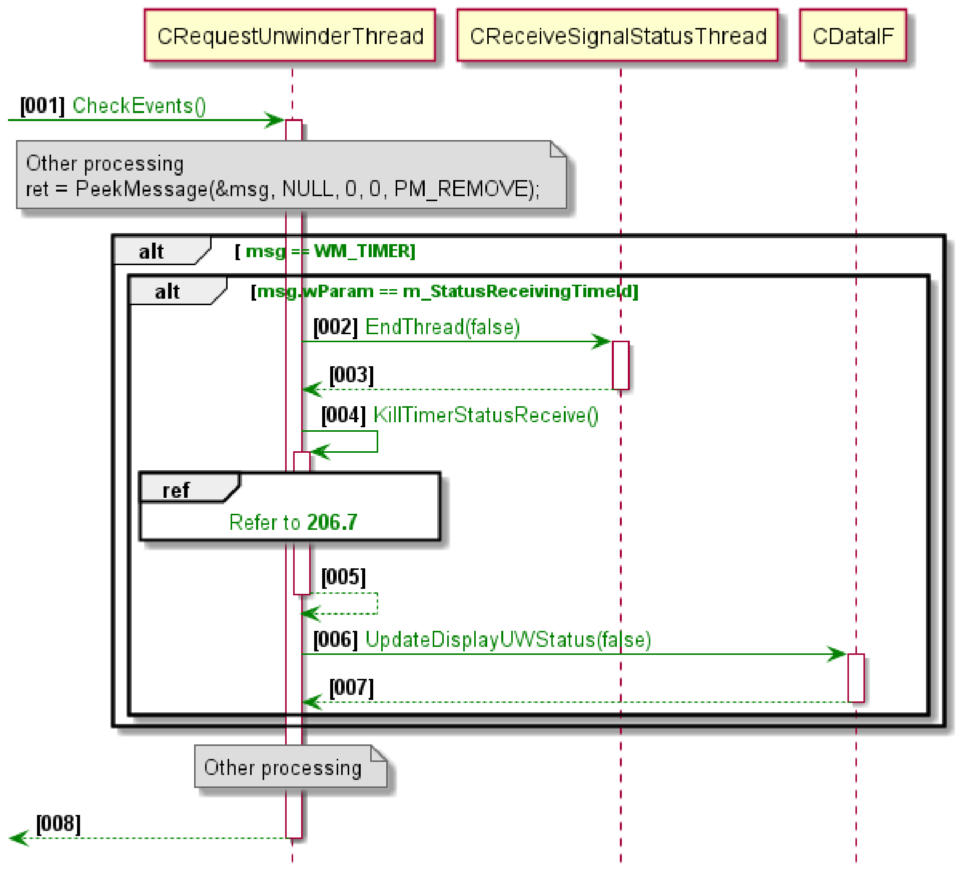
206.5 Setting for timeout timer of SignalStatus(STATUS)


CRequestUnwinderThread
CMakeComposeUnwinderData
CIni_UnwinderManager
[001] SetTimerStatusReceive()
[002] GetTimeoutTimerStatus()
[003] getQueryStatus_RepeatTime()
[004] time1
[005] getSignalstatus_Timeout_Judegment_Waittime()
[006] time2
[007] nTimeout
[008]

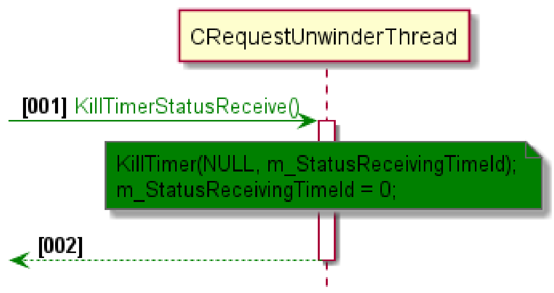
```
nTimeout = (time1 + time2) * 1000;
```

```
m_StatusReceivingTimeId = SetTimer(NULL, m_StatusReceivingTimeId, nTimeout, NULL);
```

206.6 Handle when receive UW status is timeout


CRequestUnwinderThread
CReceiveSignalStatusThread
CDataIF
[001] CheckEvents()
Other processing
ret = PeekMessage(&msg, NULL, 0, 0, PM_REMOVE);
alt [msg == WM_TIMER]
alt [msg.wParam == m_StatusReceivingTimeId]
[002] EndThread(false)
[003]
[004] KillTimerStatusReceive()
ref
Refer to 206.7
[005]
[006] UpdateDisplayUWStatus(false)
[007]
Other processing
[008]

206.7 Stop timeout timer of SignalStatus(STATUS)


CRequestUnwinderThread
[001] KillTimerStatusReceive()
KillTimer(NULL, m_StatusReceivingTimeId);
m_StatusReceivingTimeId = 0;
[002]

207. Delete channel

1. Description

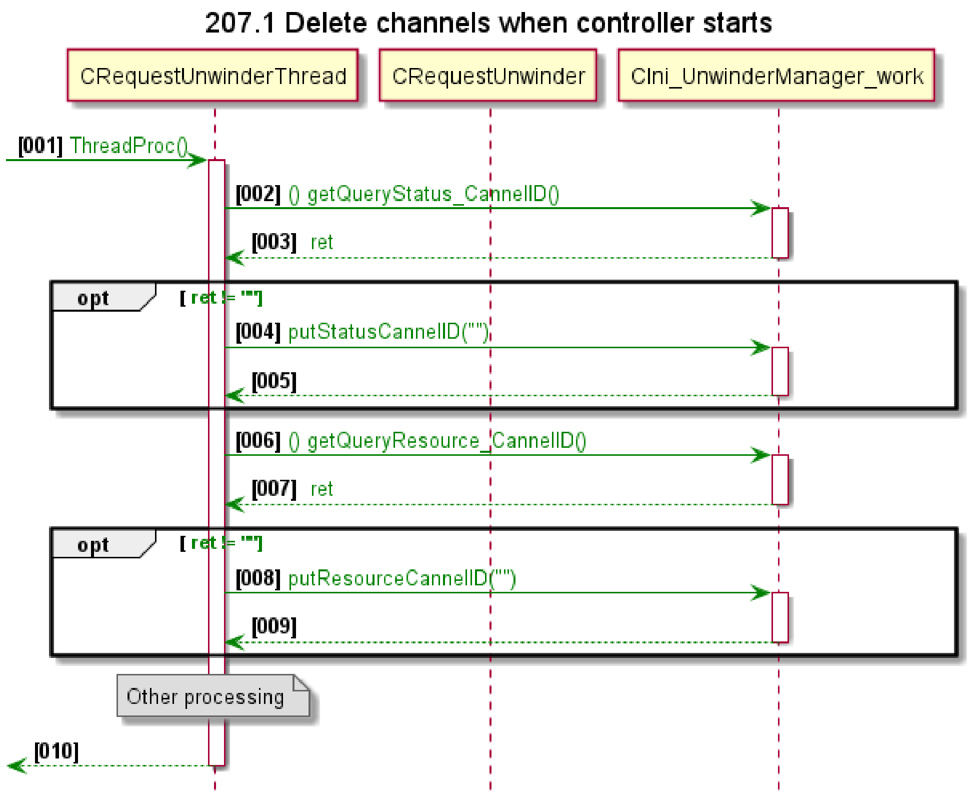
The communication channel is deleted by the channel ID notified in the response at the time of channel registration at the following timing.
- When channel information is already registered in the TP-UW_Communication_work.ini file at the time of channel registration(Described in 202).
- When Signal Status 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

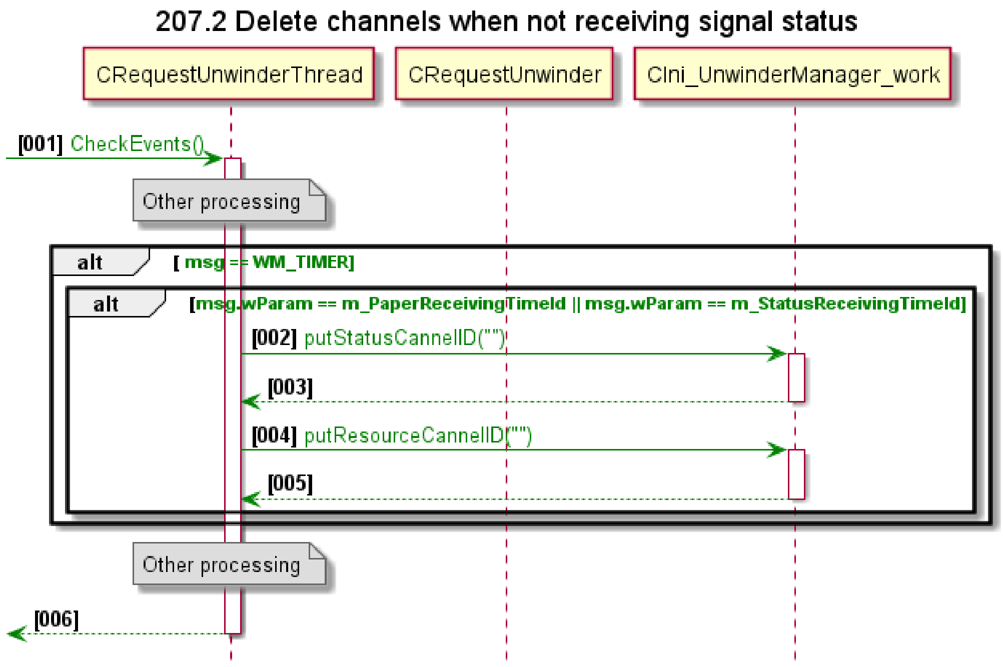
- In CRequestUnwinderThread::ThreadProc(), check for existing channel IDs in TP-UW_Communication_work.ini and delete them.
- In CRequestUnwinderThread::CheckEvents(), when WM_TIMER message is received, delete the channel IDs in TP-UW_Communication_work.ini.
- In CRequestUnwinderThread::ThreadProc(), 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

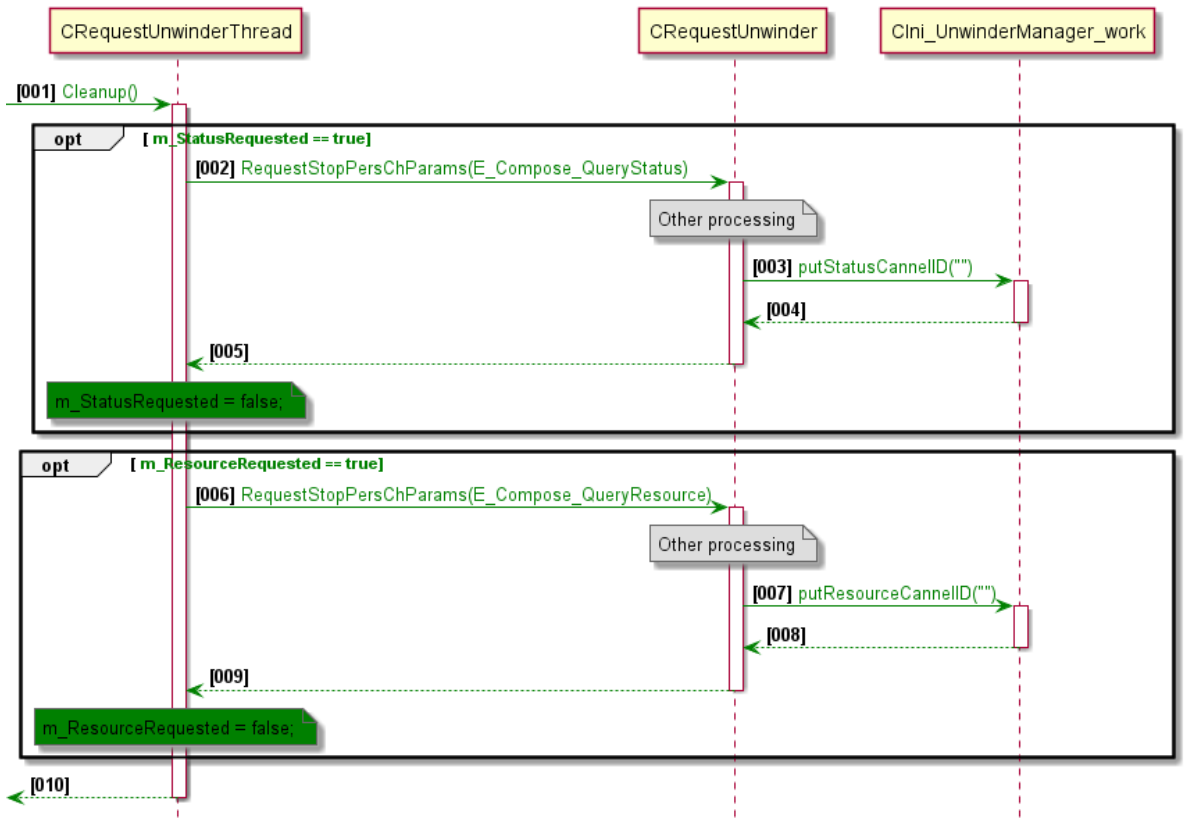


207.1 Delete channels when controller starts
CRequestUnwinderThread
CRequestUnwinder
CIni_UnwinderManager_work
[001] ThreadProc()
[002] () getQueryStatus_CannelID()
[003] ret
opt
[ret != ""]
[004] putStatusCannelID("")
[005]
[006] () getQueryResource_CannelID()
[007] ret
opt
[ret != ""]
[008] putResourceCannelID("")
[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_TIMER]
alt
[msg.wParam == m_PaperReceivingTimeId || msg.wParam == m_StatusReceivingTimeId]
[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

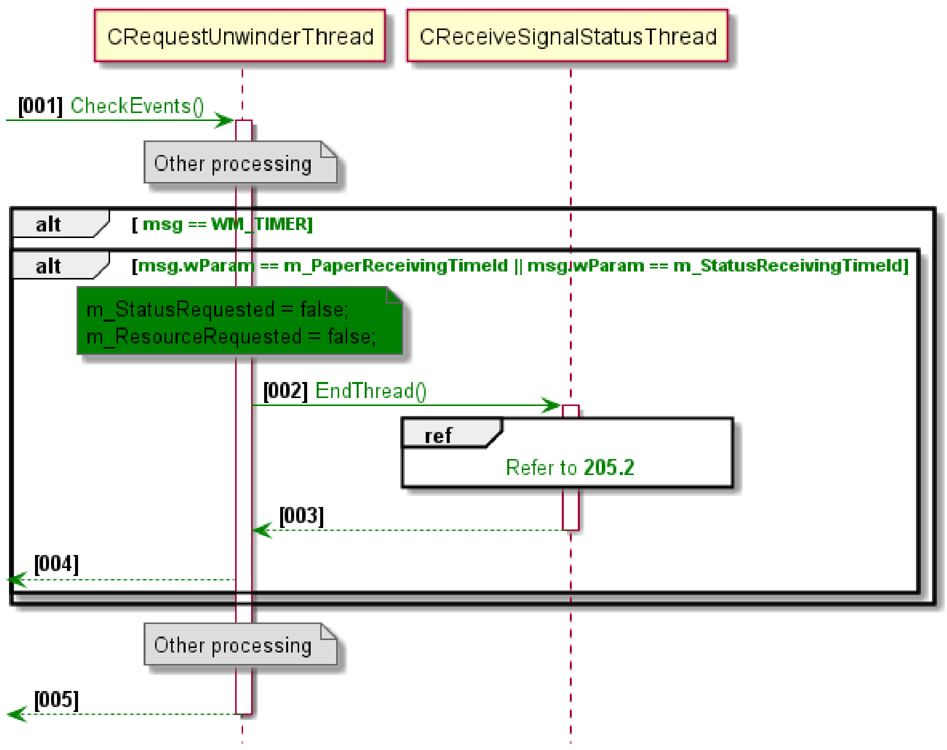
Issue a UWPing confirmation timer when SignalStatus notifications for the status notification channel or paper information notification channel can no longer be received.
If Ping passes, channel registration will be performed (until channel registration is possible).

2. Solution

- In CRequestUnwinderThread::CheckEvents(), when WM_TIMER event happens, which means SignalStatus wasn't received on time, 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.

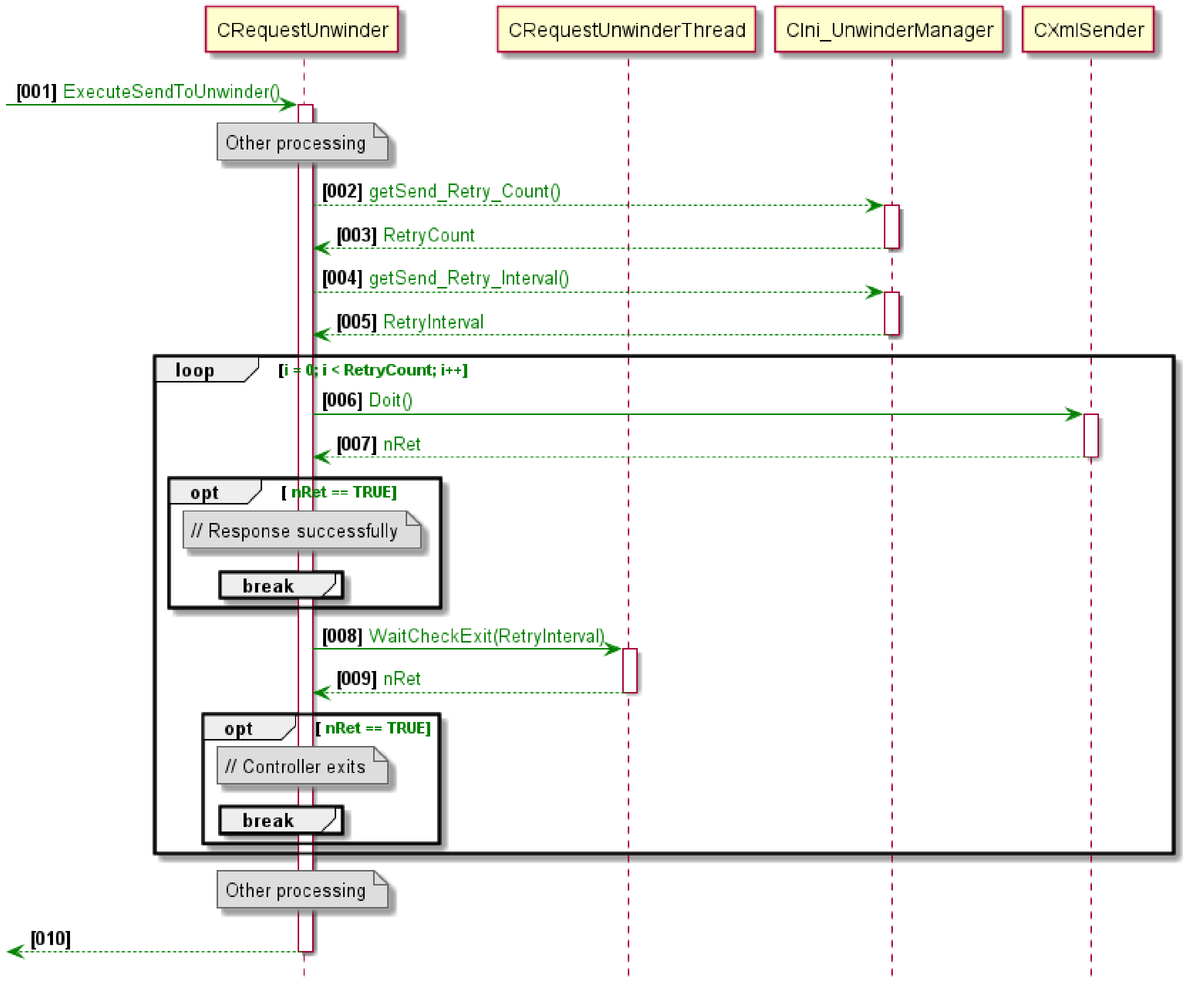
3. Detail implementation

208.1 when SignalStatus is not received



CRequestUnwinderThread
CReceiveSignalStatusThread
[001] CheckEvents()
Other processing
alt [msg == WM_TIMER]
alt [msg.wParam == m_PaperReceivingTimeId || msg.wParam == m_StatusReceivingTimeId]
m_StatusRequested = false;
m_ResourceRequested = false;
[002] EndThread()
ref
Refer to 205.2
[003]
[004]
Other processing
[005]

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 = 0; i < RetryCount; i++]
[006] Doit()
[007] nRet
opt [nRet == TRUE]
// Response successfully
break
[008] WaitCheckExit(RetryInterval)
[009] nRet
opt [nRet == TRUE]
// Controller exits
break
Other processing
[010]