Import alarms from WinCC to Nimbus

WinCC can export alarms to Nimbus using two different methods:

1) Using a text file created by a global *C-script*
2) Using a separate application (*WinCC2Nimbus*) developed using the Siemens *ODK.* It will connect to the *WinCC messaging service*

Wich method to choose depends of the application. It is always easier to configure and debug method 1, but it has the drawback that you have to edit a property for each alarm (check a checkbox) which is important to remember if you add new alarms in the future.

Both methods are described in this document.

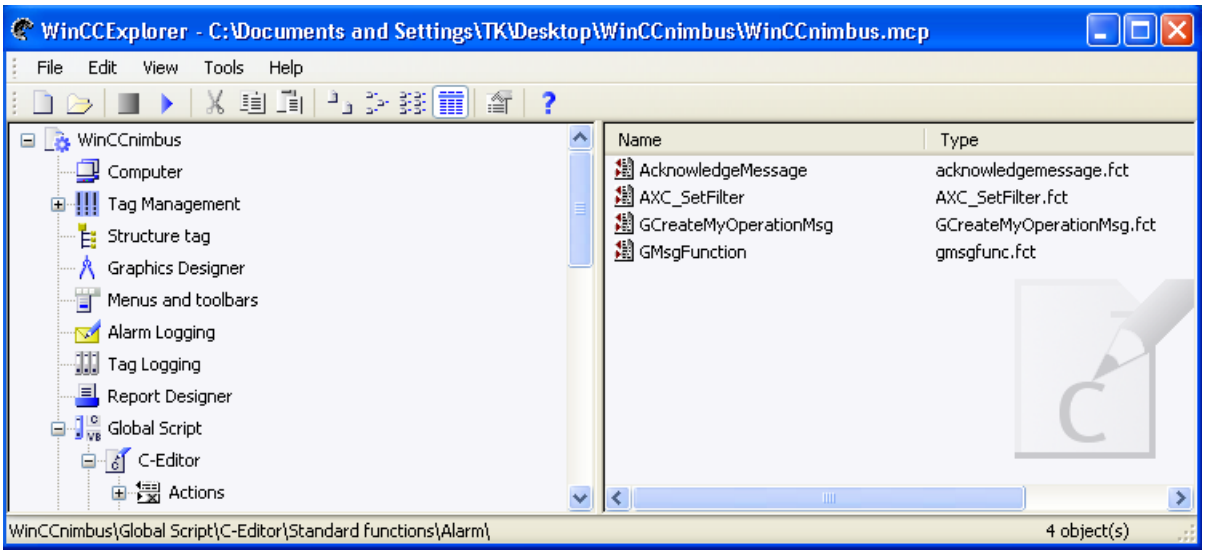
Method 1: Configure WinCC to create the Nimbus readable alarm event text

Find the file *C:\Program Files (x86)\Siemens\WinCC\aplib\Alarm\gmsgfunc.fct*. Change its name to *gmsgfunc.org*. Where the file resides depends of the *WinCC* and *OS* version.

Copy the file *gmsgfunc.fct* from the Nimbus media found in the folder *..\Tools and documents\WinCC*. The file may also be found in the *WinCC2Nimbus* package at *www.automatisera.nu*.

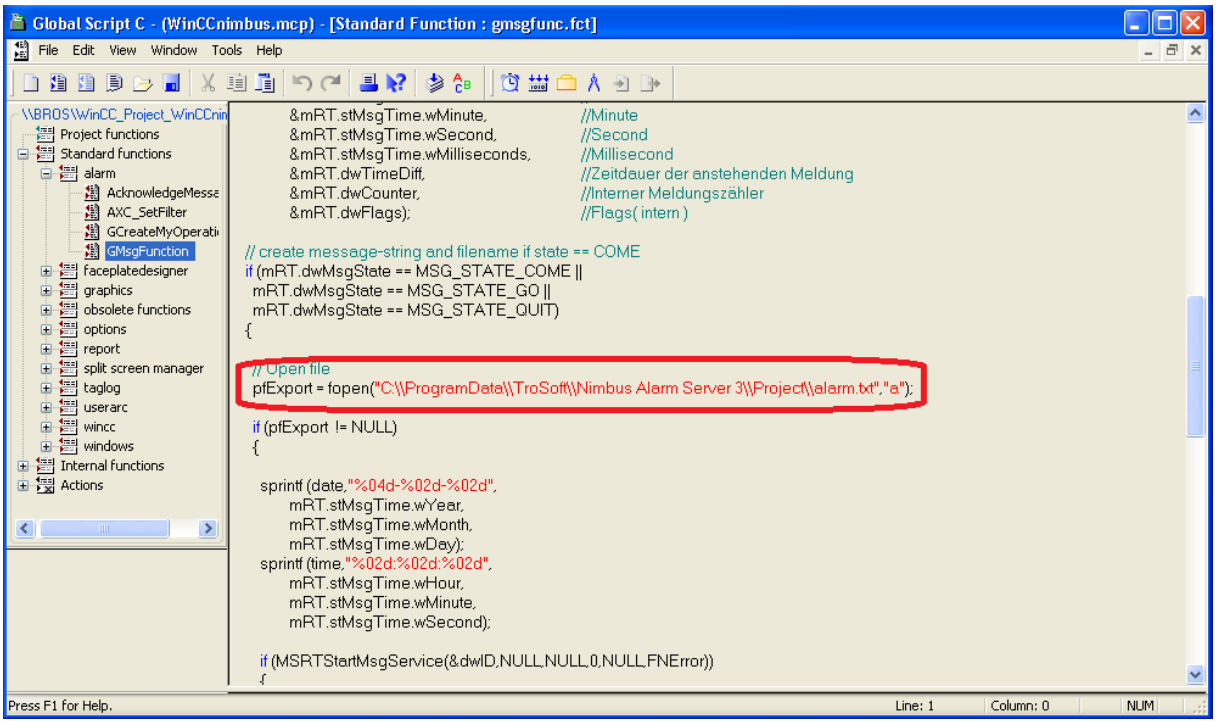
The script is also provided in *.txt-format* if you like to cut and paste the script. *fct*-files cannot be opened using a text editor, ex *Notepad*. The script is also included in this document later on.

Open *WinCC Explorer*.



Go to *Global Script -> C-Editor -> Standard functions -> Alarm*

Open *GMsgFunction*



Edit the row with the path and filename if needed. It will by default point to the *Nimbus version 3 project folder*. The folder is created when *Nimbus* is installed. You can choose any folder you like as long as both *Nimbus* and *WinCC* have security settings allowing the applications to create and delete files in that folder.

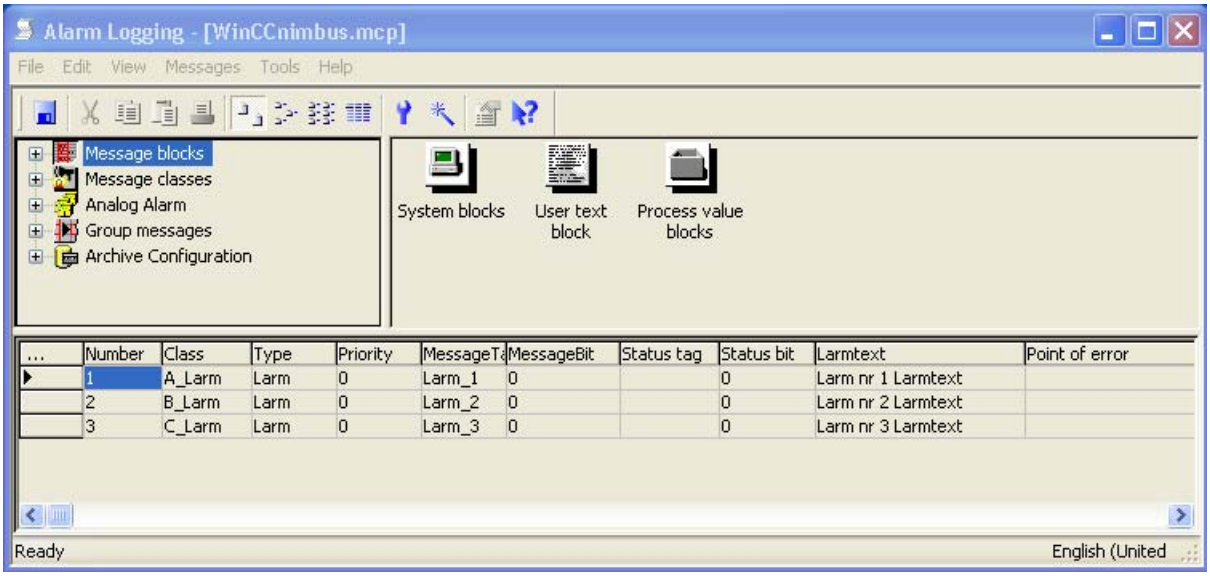
If the original *gmsgfunc.fct* already is used for something, you must manually edit the script.

Select *File -> Close*. Compile if suggested by *WinCC*.

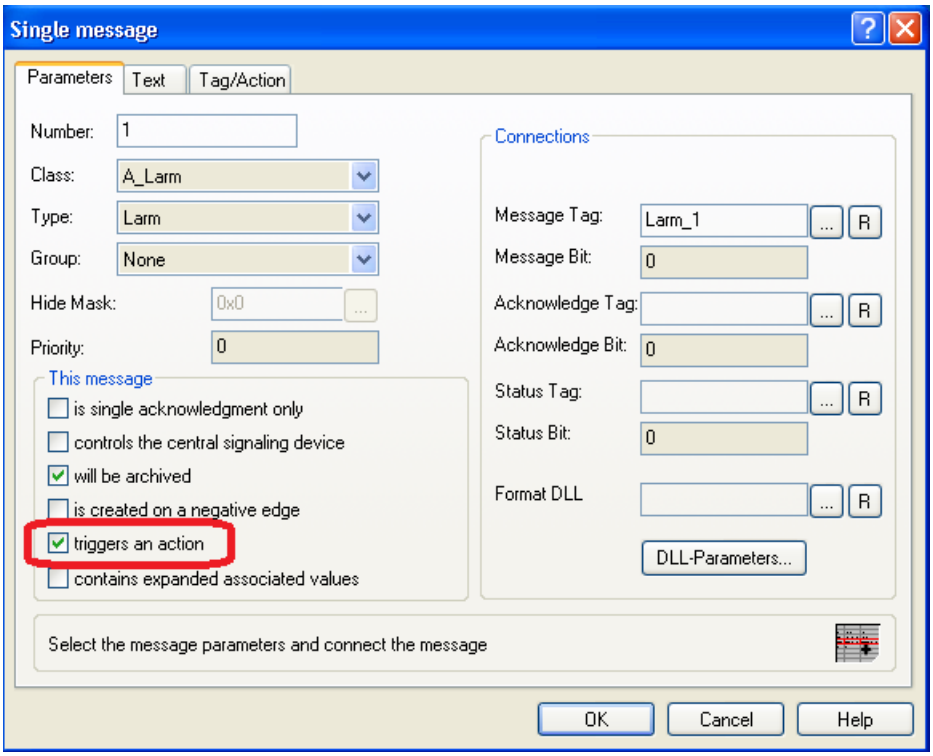
Select *Tools -> Regenerate Header*.

Close the *Script editor*.

Open *Alarm Logging*



You have to change a parameter for each alarm to be sent to *Nimbus*. Right click and select *Properties*.



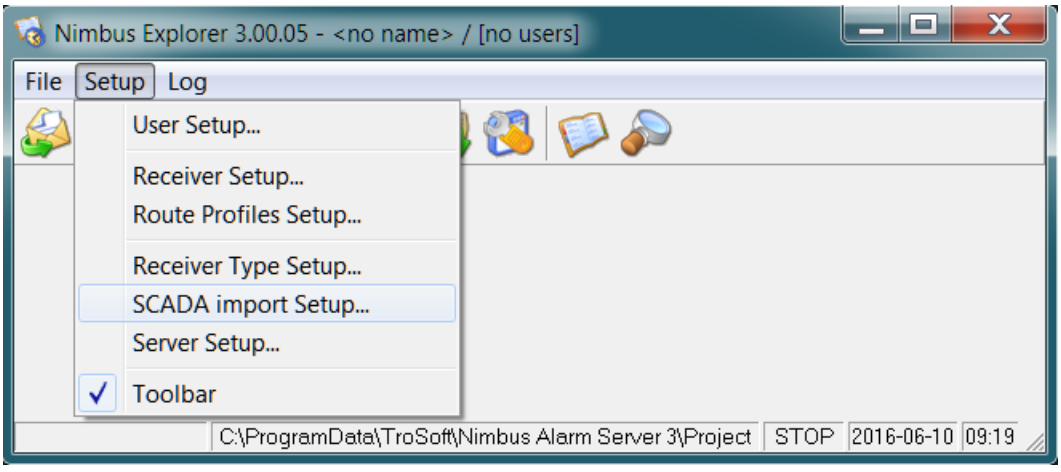
Check *Triggers an action*. This will cause *WinCC* to activate the script.

If you are using *PCS7* and *CFC* then the check box is found in there. The settings are copied from *PCS7/CFC* at compile/download, hence overwriting the above settings.

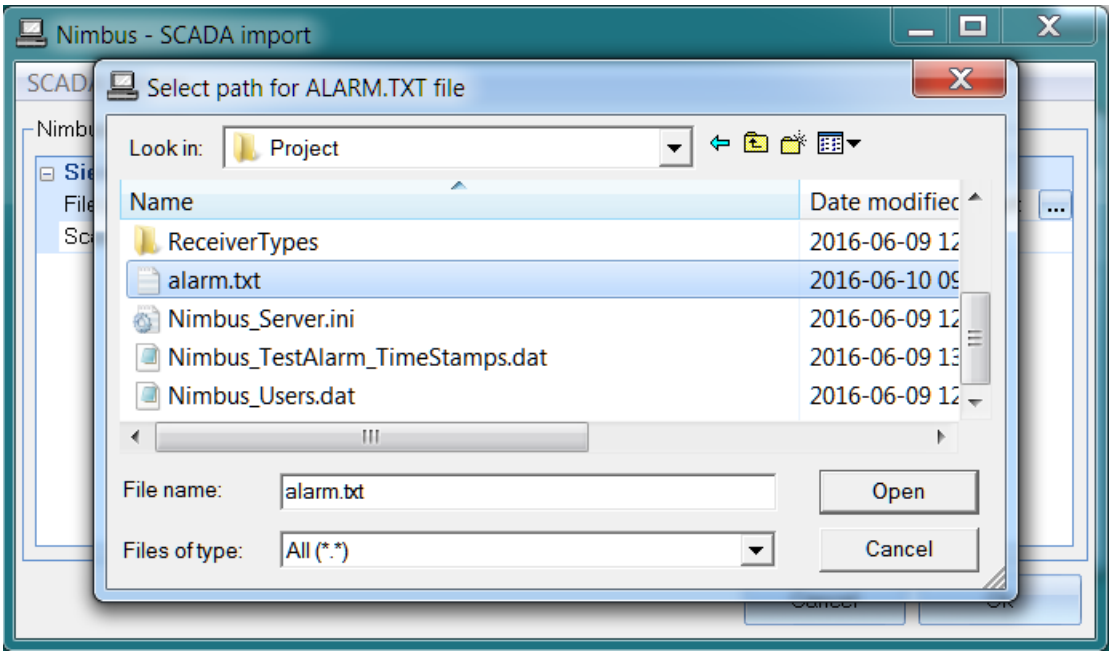
Start *WinCC runtime* and try to set an alarm, the textfile should be created.

Method 1: Configure Nimbus to import the alarm event text file

Run *Nimbus Explorer* (right click and select *Run as Administrator*) using its shortcut. *Nimbus Explorer* shall always be run as *Administrator*.



Select *Setup -> SCADA Import Setup.*



Select *SCADA System -> Add SCADA System Import -> Siemens WinCC*

Select *alarm.txt* in the same folder used in the script. If the file does not exist, just select *Open*. Ensure the *File path to scan* also included the textfile name.

When you run *Nimbus Alarm Server*, the program will remove the textfile because it contains just old alarm events. Create a new alarm in *WinCC*. It should now appear in *Nimbus Explorer*. *Nimbus* always remove the file when it has been read.

Metod 1: gmsgfunc.fct

```
//
// Date     / Vers  / Sign / Comment
// ------------------------------------------------------------------------------------
// 16.02.26 / 1.0.0.0 /  TR  / Major changes
// 16.02.27 / 1.0.0.1 /  TR  / Added priority and Class moved to  Area
// 16.03.18 / 1.0.0.2 /  TR  / Added Text block 2
//
//
//
//
//
// ------------------------------------------------------------------------------------

#include "msrtapi.h";

BOOL GMsgFunction( char* pszMsgData)
{

  extern char g_Msg[];

  MSG_RTDATA_STRUCT mRT;
  PCMN_ERROR pError;
  DWORD dwID = 0;
  CMN_ERROR Error;
  MSG_CSDATA_STRUCT MsgData;
  MSG_CLASS_STRUCT MsgClass;
  MSG_TEXT_STRUCT mtsClass;

  MSG_TEXT_STRUCT mtsBlock1;
  MSG_TEXT_STRUCT mtsBlock2;
  MSG_TEXT_STRUCT mtsBlock3;

  char lpszMsgState[256];
```

```
char date[20];
char time[20];
char errcode[20] = "\r\n";
long lPriority = 0;

FILE *pfExport;
pError=&Error;

memset (&mRT, 0, sizeof(MSG_RTDATA_STRUCT));
memset (&MsgData, 0, sizeof(MSG_CSDATA_STRUCT));
memset (&MsgClass, 0, sizeof(MSG_CLASS_STRUCT));

if (pszMsgData != NULL )
{
  //Read messagedata
  sscanf( pszMsgData, "%ld,%ld,%04d.%02d.%02d,%02d:%02d:%02d:%03d,%ld, %ld, %ld",
    &mRT.dwMsgNr,                  //Messagenr
    &mRT.dwMsgState,               //Status MSG_STATE_COME, .._GO, .._QUIT, .._QUIT_SYSTEM
    &mRT.stMsgTime.wYear,          //Day
    &mRT.stMsgTime.wMonth,         //Month
    &mRT.stMsgTime.wDay,           //Year
    &mRT.stMsgTime.wHour,          //Hour
    &mRT.stMsgTime.wMinute,        //Minute
    &mRT.stMsgTime.wSecond,        //Second
    &mRT.stMsgTime.wMilliseconds,  //Millisecond
    &mRT.dwTimeDiff,               //Zeitdauer der anstehenden Meldung
    &mRT.dwCounter,                //Interner Meldungszähler
    &mRT.dwFlags);                 //Flags( intern )

  // create message-string and filename if state == COME
  if (mRT.dwMsgState == MSG_STATE_COME ||
    mRT.dwMsgState == MSG_STATE_GO ||
    mRT.dwMsgState == MSG_STATE_QUIT)
  {

    // Open file, this file must be located somewhere where we have read/write/delete
    // access rights
    pfExport =
      fopen("C:\\ProgramData\\TroSoft\\Nimbus Alarm Server 3\\Project\\alarm.txt","a");

    if (pfExport  != NULL)
    {

      sprintf (date,"%04d-%02d-%02d",
      mRT.stMsgTime.wYear,
      mRT.stMsgTime.wMonth,
      mRT.stMsgTime.wDay);
      sprintf (time,"%02d:%02d:%02d",
      mRT.stMsgTime.wHour,
      mRT.stMsgTime.wMinute,
      mRT.stMsgTime.wSecond);

      if (MSRTStartMsgService(&dwID, NULL, NULL, 0, NULL, pError))
      {

        MSRTGetMsgCSData (mRT.dwMsgNr, &MsgData, pError);
        // These are the Texts as they appear in the Text-tab in alarm logging,
        // block 1 -> dwTextID[0] etc
        MSRTGetMsgText (0, MsgData.dwTextID[0], &mtsBlock1, pError);
        MSRTGetMsgText (0, MsgData.dwTextID[1], &mtsBlock2, pError);
        MSRTGetMsgText (0, MsgData.dwTextID[2], &mtsBlock3, pError);
        MSRTGetClassInfo (mRT.dwMsgNr, &MsgClass, pError);
        MSRTGetMsgText (0,MsgClass.dwName, &mtsClass, pError);
        MSRTGetMsgPriority (MsgData.dwMsgNr, (long*)&lPriority, pError);

        fprintf(pfExport,"%s#", date);
        fprintf(pfExport,"%s#", time);
        fprintf(pfExport,"%s#", mtsBlock1.szText);

        // This text is the only cruical text for Nimbus
        switch (mRT.dwMsgState)
        {
        case MSG_STATE_COME:
          fprintf(pfExport,"%s","Aktivt");
          break;
        case MSG_STATE_GO:
          fprintf(pfExport,"%s","Avgått");
```

```
            break;
          case MSG_STATE_QUIT:
            fprintf(pfExport,"%s","Kvitterat");
            break;
          }
          fprintf(pfExport,"#");

          fprintf(pfExport,"%ld#", lPriority);
          fprintf(pfExport,"%s#", mtsClass.szText);

          // You may add any texts you wish but Nimbus will only import them if you
          // select them in the file found in
          // Nimbus Project folder ..\Project\Import\Import_WinCC.imp
          fprintf(pfExport,"%s#", mtsBlock3.szText);

          fprintf(pfExport,"\n");

          MSRTStopMsgService( dwID, pError);

        } // MSRTStartMsgService

        //Close and save File
        fclose( pfExport );

      } // pfExport  != NULL

    } // mRT.dwMsgState

  } // pszMsgData != NULL

  return TRUE;

}
```

Method 2: Configurera WinCC to export alarm events using WinCC2Nimbus

Install the *WinCC2Nimbus* application, it will be found in the media folder ..*Tools and documents**WinCC*. It can also be found at *www.automatisera.nu*.

WinCC2Nimbus may be run as a normal Windows application or as a service. If run as a normal Windows application it should always be run as *Administrator*.

To add the application as service, run *WinCC2Nimbus* using the *-i* command line parameter

Example:

"C:\Pogram Files (x86)\WinCC2Nimbus\WinCC2Nimbus.exe" -i

Obs! For the installation to succeed properly, it is very important this command is executed in an elevated CMD-prompt (run as Administrator).

Remove from services using the command line parameter *-u*. Ensure the service first has been stopped.

First time the service has to be manually started using the *Service Control Manager*. The service will automatically start when the the computer is rebooted. To ensure WinCC has started before *WinCC2Nimbus* during reboot, set the service *Startup Type* to *Automatic (Delayed start).*

To ensure everything works ok, run the application as a normal Windows application before installing as a service. Alarm events are shown in the *WinCC2Nimbus* window. The program will also create a *LogFiles* subfolder in its installation folder, where all events will be stored in logfiles (textfiles). The logfiles will automatically be deleted when they are older than 90 days (default setting)

WinCC2Nimbus may also create events and send to Nimbus when it is started and stopped or when WinCC is started and stopped. This functionality is configured in *WinCC2Nimbus.ini* located in the installation folder. Each event type has its own section, ex to create an alarm event when WinCC is stopped, edit the *[NimbusMessageWhenWinCCIsStopped]* section:

EventType=1

T0=WinCC
T1=
T2=99
T3=
T4=WinCC2 has stopped

EventType sets the event type as follows*: 0 = Inactive (normal), 1 = Active (Alarm), 2 = Acknowledge*

If *EventType* is set to 1 as above, *WinCC2Nimbus* will create an alarm and send to Nimbus when WinCC is stopped. All fields, *T0 (Tag)* to *T4 (Description)* may be set to any text. This enables the possibility to create route profiles matching this alarm or use any existing route profile.

Tip: Processes (ex the WinCC-services) may be monitored using Nimbus. Use the *Watchdog* functionality in Nimbus, it will be found in *Nimbus Explorer -> Setup -> Server setup -> Watchdog*

WinCC2Nimbus will subscribe to all alarm events in *WinCC* and send them to *Nimbus Alarm Server* using one of two different variants:

a) *Queing the alarm events directly in the internal Nimbus Alarm Server event queue*

b) using a TCP-connection (Generic TCP)

Variant A is default and does not require any *SCADA import* in Nimbus, however it requires that both *Nimbus Alarm Server* and WinCC are installed in the same server.

If using variant A, run *WinCC* and *Nimbus Alarm Server* and create an alarm in *WinCC*. It should show up both in *WinCC2Nimbus* and *Nimbus Explorer*.

Method 2 - variant B: Configure WinCC2Nimbus for TCP

Variant B is suitable if *Nimbus Alarm Server* is installed in a different server. To enable TCP the parameter *ConnectionType* in *WinCC2Nimbus.ini* needs to be changed from 0 to 1.

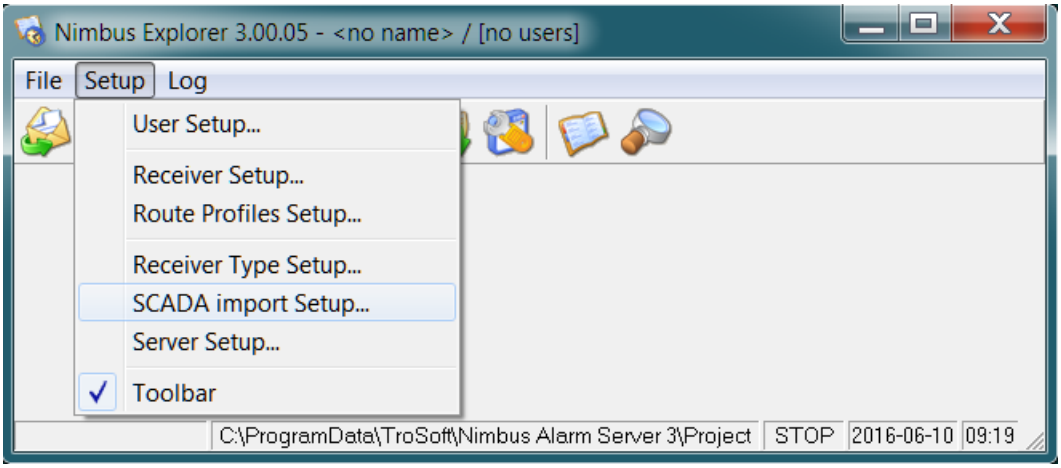
WinCC2Nimbus will by default listen at TCP port *14000*. The port number may be changed using the *PortNumber* parameter.

Restart *WinCC2Nimbus* for the changes in *WinCC2Nimbus.ini* to take effect.

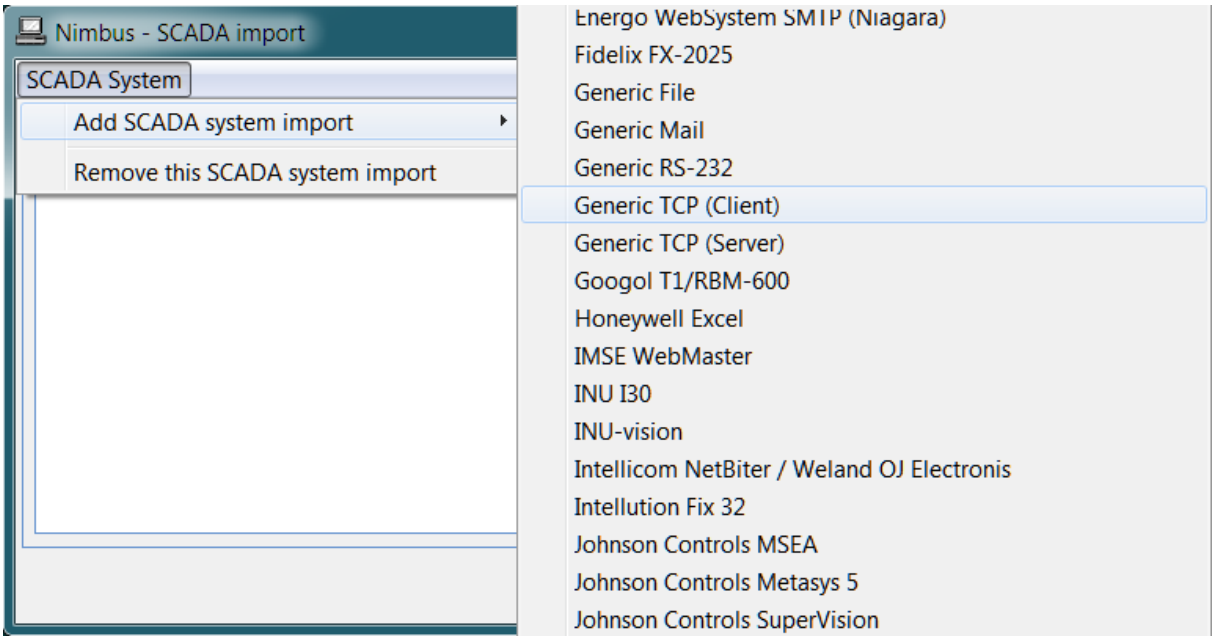
Obs! Ensure the computer firewalls allow this traffic. WinCC2Nimbus act as a TCP socket server and Nimbus Alarm Server act as a TCP socket client.

Metod 2 - variant B: Configure Nimbus connection to WinCC2Nimbus using TCP

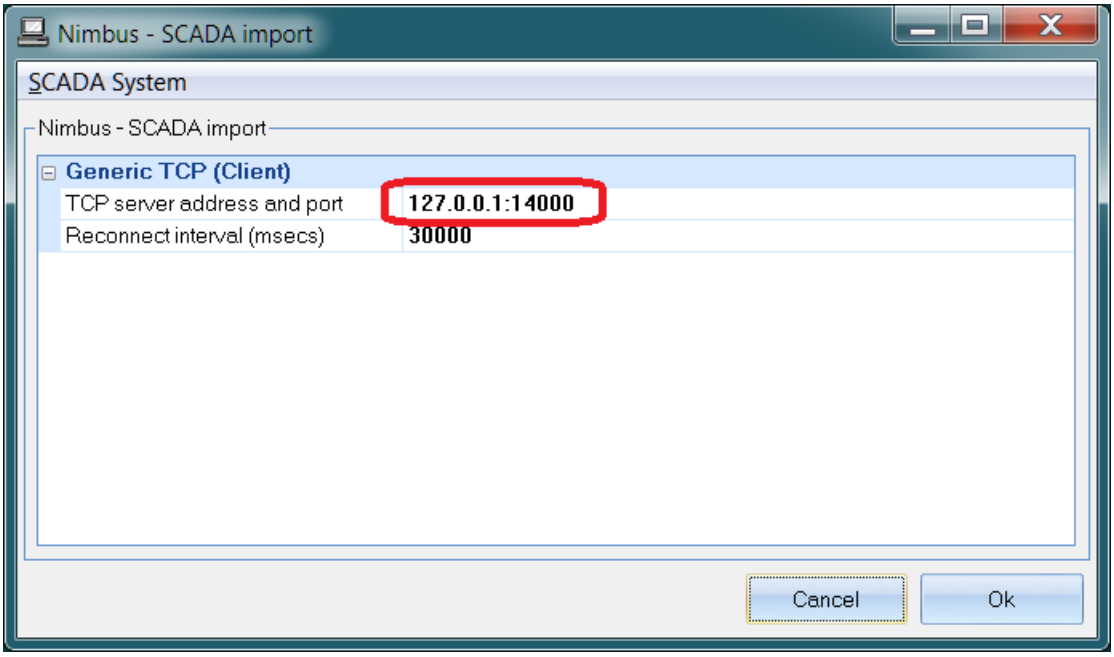
Run *Nimbus Explorer* (right click and select *Run as Administrator*) using its shortcut. *Nimbus Explorer* shall always be run as *Administrator*.



Select *Setup -> SCADA Import Setup.*



Select *SCADA System -> Add SCADA system import -> Generic TCP (Client)*



Select adress and port to the *WinCC*-server. *Nimbus Alarm Server* act as a socket client.

Run *WinCC* and *Nimbus Alarm Server*, create an alarm. It should show up both in *WinCC2Nimbus* and *Nimbus Explorer*.