Import alarms from WinCC Unified to Nimbus

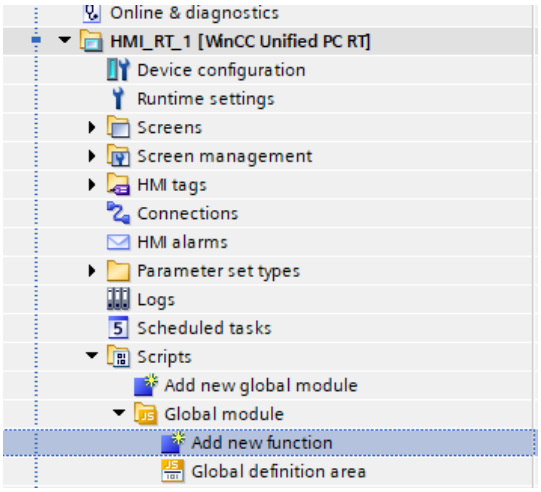
WinCC can export alarms to Nimbus using a text file created for each new alarm event. Nimbus reads/imports the file cyclically.

This instructions assume than Nimbus is already installed.

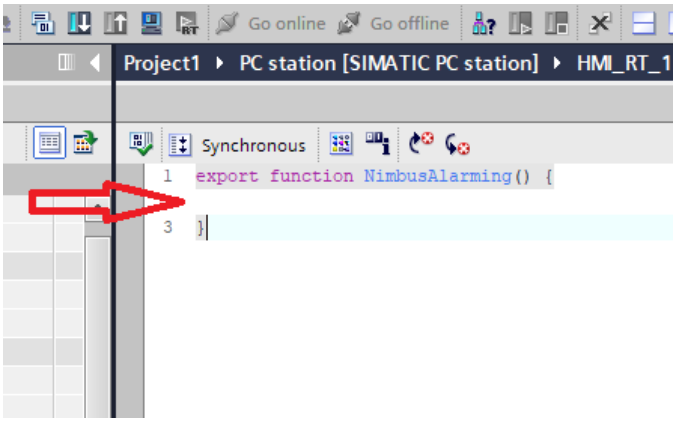
The version of WinCC Unified needs to be at least v.19 (update 2)

Configure WinCC to create the Nimbus readable alarm event text file

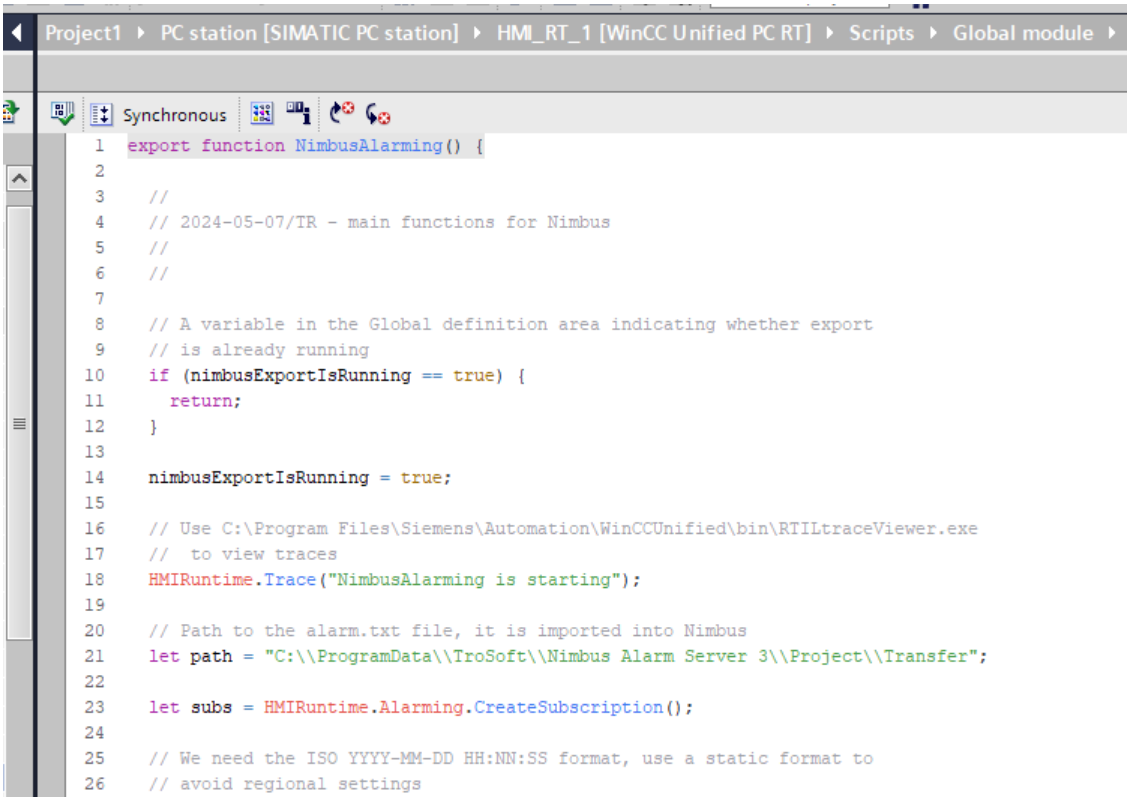
First of all create the Javascript which creates the text file. It is added to the *Scripts -> Global module* folder:



Select *Add new function*. Rename the function to *NimbusAlarming* and delete the two default parameters found in *Properties*. The *NimbusAlarming* script is found later on in this document, copy and paste it in between the brackets in the script editor.



The result will look something like this:

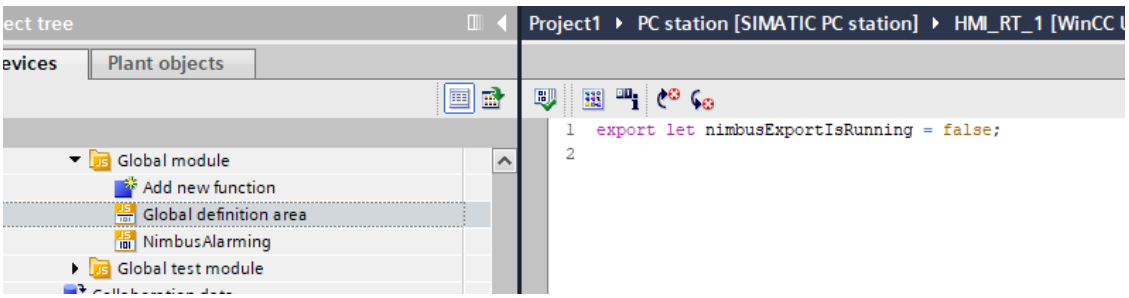


The script should be started by a *Scheduled Task*, however it is unable to start a script only once per system start so we need to create a global flag to know whether the script already is started or not.

Select the *Global definition area* and copy and paste the following row into the script editor:

export let nimbusExportIsRunning = false;

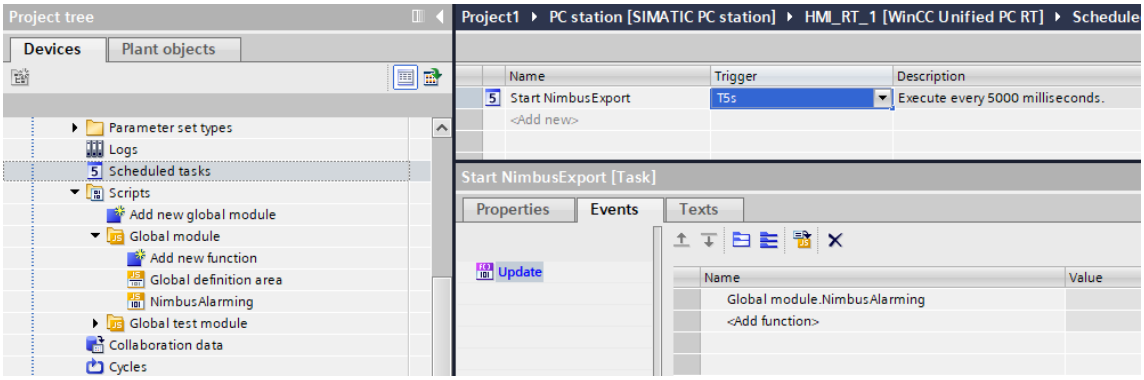
It will look like this:



Now go to *Scheduled task* and create a new task. Name it *Start NimbusExport*.

Set the *Trigger* to *T5s* and at *Events* select *<Add function>* and select the script *Global module.NimbusAlarming*.

It will look like this:

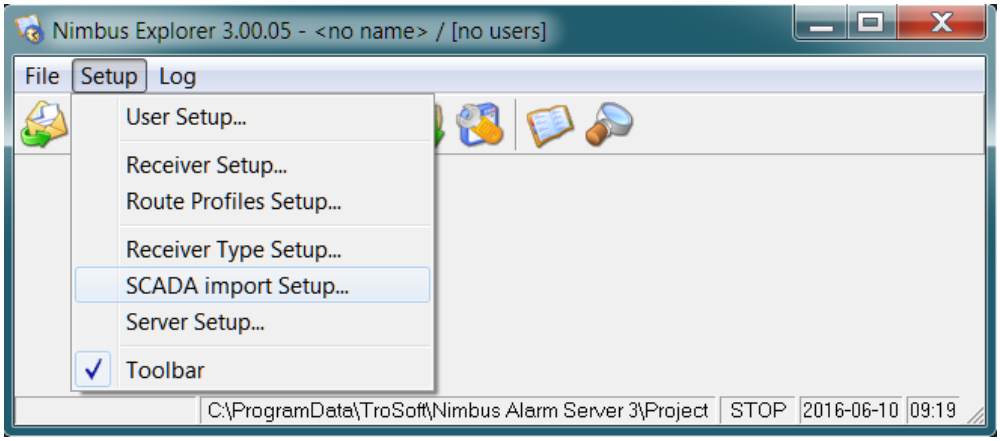


Now compile and start Runtime, create some alarm(s) and see that the script have created the folder *C:\ProgramData\TroSoft\Nimbus Alarm Server 3\Project\Transfer* and also a file named *Alarm.txt* with the alarm contents.

Where the script put the file could be changed by the path *variable in the script and could be located more or less anywhere, just ensure Nimbus have read/write/delete access rights in the folder (and of course also WWinCC Unified should have access to the folder)*

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*.

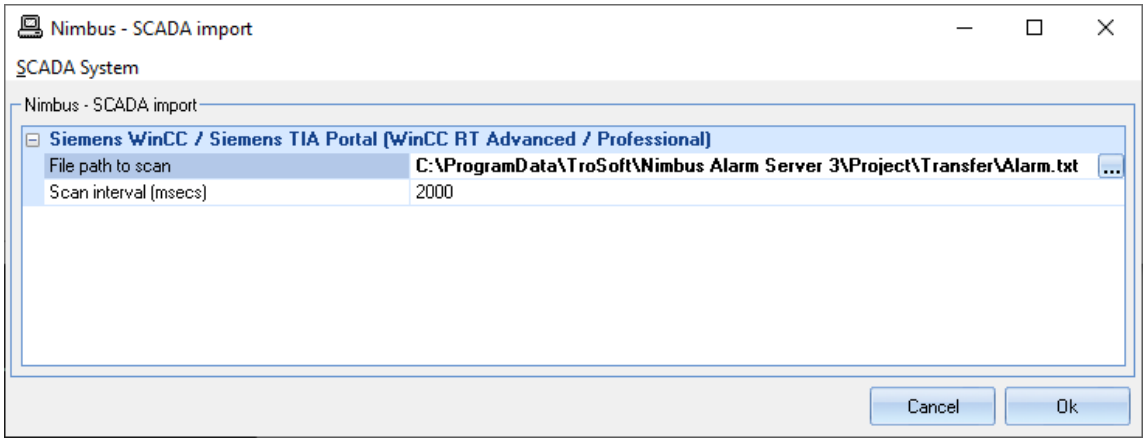


Select *Setup -> SCADA Import Setup.*

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

Find and select *Alarm.txt*. If the file does not exist, just select *Open*.

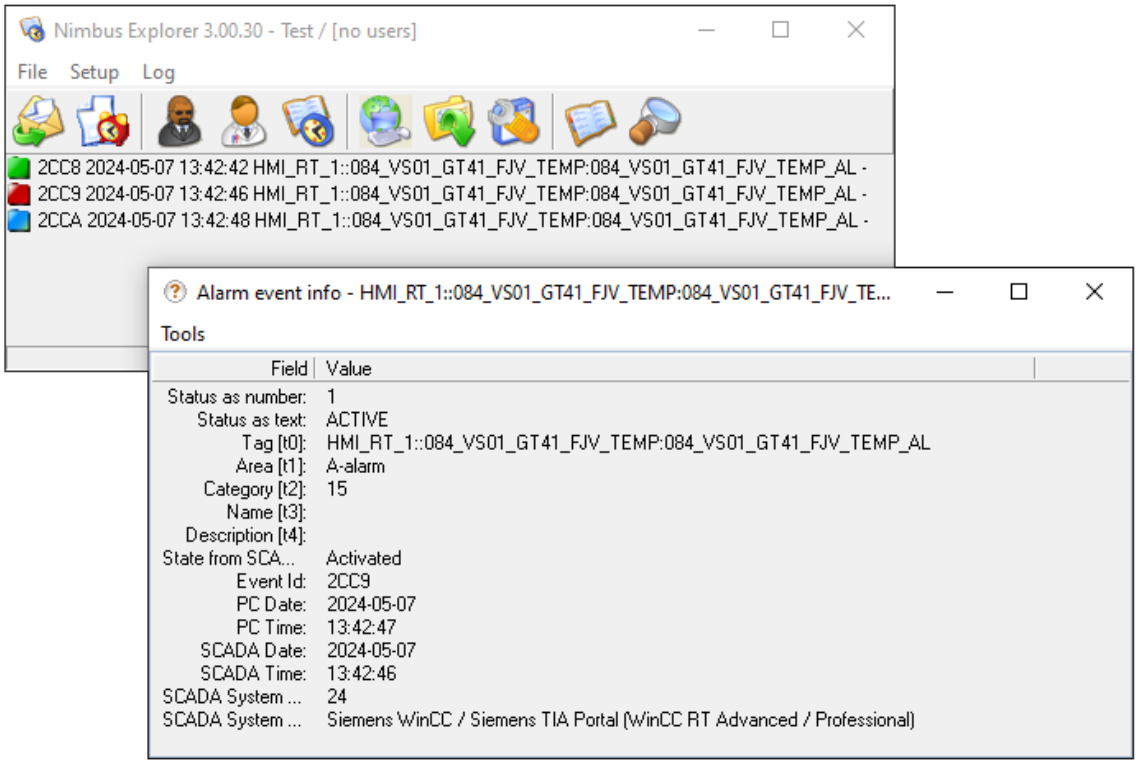
Ensure the *File path to scan* is complete, including the *Alarm.txt* file name.



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.

It should look something like this (double click an alarm to see all alarm properties):



The NimbusAlarming script

```
//
// 2024-05-07/TR - main functions for Nimbus
//
//

// A variable in the Global definition area indicating whether export
// is already running
if (nimbusExportIsRunning == true) {
  return;
}

nimbusExportIsRunning = true;

// Use C:\Program Files\Siemens\Automation\WinCCUnified\bin\RTILtraceViewer.exe
//  to view traces
HMIRuntime.Trace("Nimbus Alarming is starting");

// Path to the alarm.txt file, it is imported into Nimbus
let path = "C:\\ProgramData\\TroSoft\\Nimbus Alarm Server 3\\Project\\Transfer";

HMIRuntime.FileSystem.CreateDirectory(path).then(
  function() {
    HMIRuntime.Trace("Nimbus directory successfully created");
});

let subs = HMIRuntime.Alarming.CreateSubscription();

// We need the ISO YYYY-MM-DD HH:NN:SS format, use a static format to
// avoid regional settings
let dateFormat = new Intl.DateTimeFormat("sv-SE", {
  dateStyle: 'short'
});

let timeFormat = new Intl.DateTimeFormat("sv-SE", {
  timeStyle: 'medium'
});

// Filter changereason does not seem to work...
subs.Filter = "ChangeReason IN (3, 5, 7)";
subs.Language = 1033;

subs.OnAlarm = function(Errorcode, SystemNames, ResultSet)
{

  let nimbusString = "";

  for (let index = 0; index < ResultSet.length; index++)
  {

    HMIRuntime.Trace("Nimbus Alarm [" + index + "]: Name: " + ResultSet[index].Name +
      ", State: " + ResultSet[index].State +
      ", ChangeReason: " + ResultSet[index].ChangeReason +
      ", NotificationReason: " + ResultSet[index].NotificationReason +
      ", AlarmText" + ResultSet[index].AlarmText);

    let nimbusState;
    let timeStamp;

    // Add || Change
    if (ResultSet[index].NotificationReason == 1 ||
      ResultSet[index].NotificationReason == 2)
    {
      switch (ResultSet[index].ChangeReason) {
        case 3: // RaisedEvent
          nimbusState = "Activated";
          timeStamp = ResultSet[index].RaiseTime;
          break;
        case 5: //  ClearEvent
          nimbusState = "Inactivated";
          timeStamp = ResultSet[index].ClearTime;
          break;
        case 7: // AcknowledgeEvent
```

```
          nimbusState = "Acknowledged";
          timeStamp = ResultSet[index].AcknowledgementTime;
          break;
        default:
          continue;
      }
    }

    nimbusString += dateFormat.format(timeStamp) + "#" + timeFormat.format(timeStamp) + "#";

    // This field will end up in the Nimbus Tag (T0) field
    nimbusString += ResultSet[index].Name + "#";

    // This field will end up in the Nimbus Status (T5) field
    // (needs to be located exactly here and with the status contents)
    nimbusString += nimbusState + "#";

   // This field will end up in the Nimbus Category (T2) field
    nimbusString += ResultSet[index].Priority + "#";

   // This field will end up in the Nimbus Area (T1) field
    nimbusString += ResultSet[index].AlarmClassName + "#";

   // This field will end up in the Nimbus Description (T4) field
    nimbusString += ResultSet[index].AlarmText[0];

    nimbusString += "\r\n";

  }

  // Write only once to avoid async write problems
  if (nimbusString != "")
  {
    HMIRuntime.FileSystem.AppendFile(path + "\\" + "alarm.txt",
      nimbusString, "ISO-8859-1").then(
      function() {
        HMIRuntime.Trace("Nimbus file append succeeded");
      }).catch(function(errCode) {
        HMIRuntime.Trace("Nimbus file append failed errorcode=" + errCode);

      });
  }

}

subs.Start();

HMIRuntime.Trace("Nimbus Alarming was started");
```