

# Import alarms from WinCC Unified to Nimbus

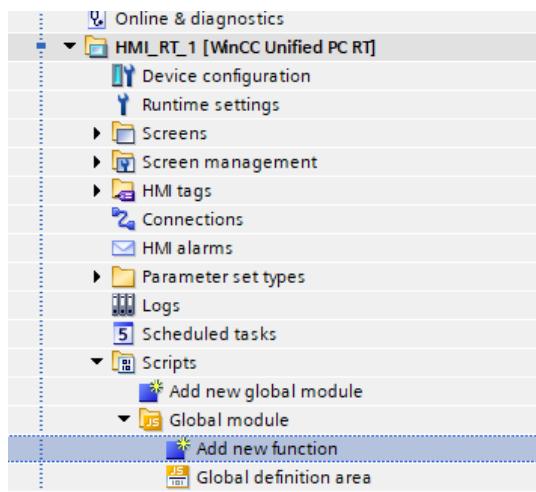
WinCC can export alarms to Nimbus using a text file created for each new alarm event. Nimbus reads/imports the file cyclically.

This instructions assume than Nimbus is already installed.

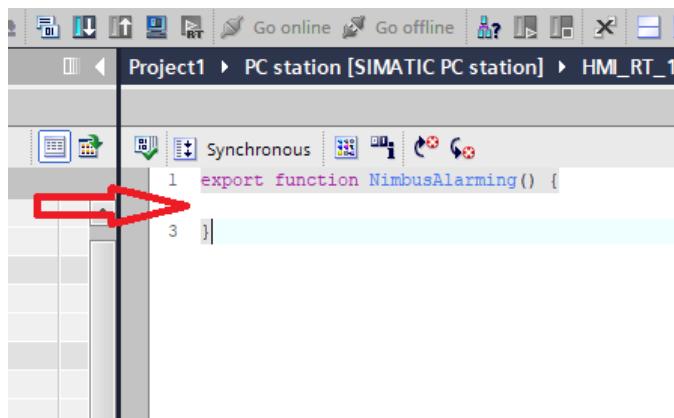
The version of WinCC Unified needs to be at least v.19 (update 2)

## Configure WinCC to create the Nimbus readable alarm event text file

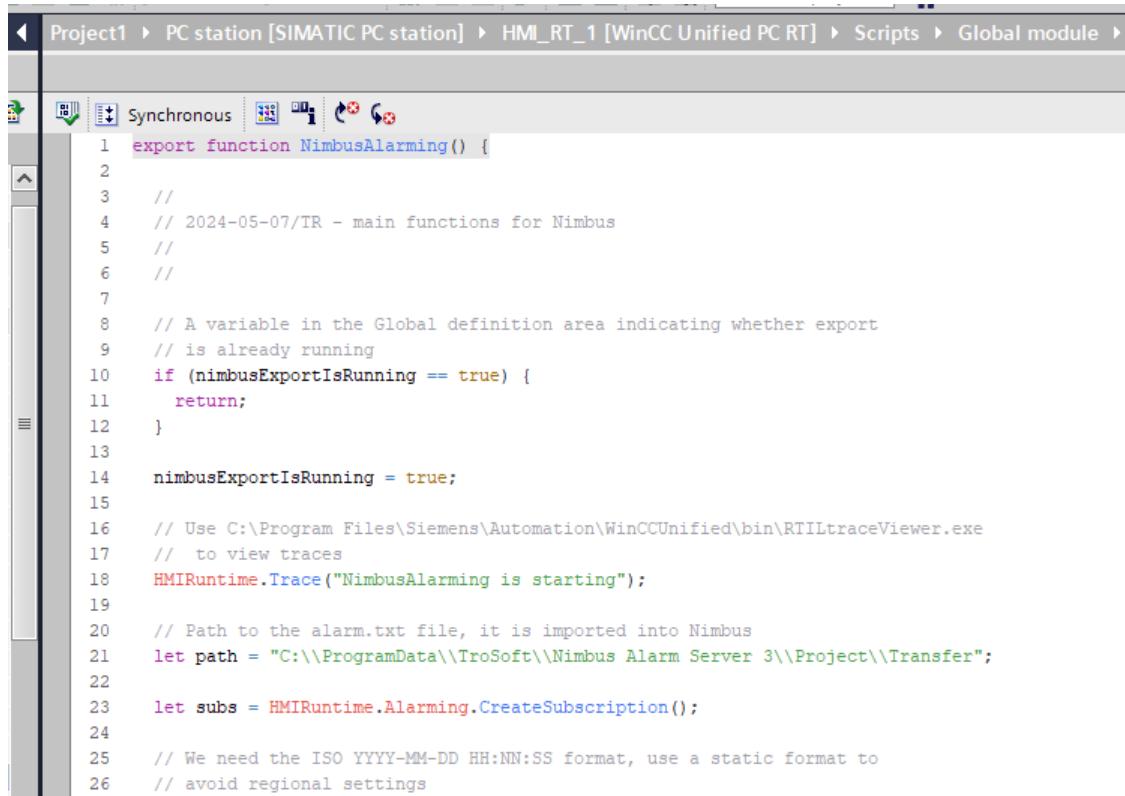
First of all create the Javascript which creates the text file. It is added to the *Scripts -> Global module* folder:



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:



```

1 export function NimbusAlarming() {
2
3     //
4     // 2024-05-07/TR - main functions for Nimbus
5     //
6     //
7
8     // A variable in the Global definition area indicating whether export
9     // is already running
10    if (nimbusExportIsRunning == true) {
11        return;
12    }
13
14    nimbusExportIsRunning = true;
15
16    // Use C:\Program Files\Siemens\Automation\WinCCUnified\bin\RTILtraceViewer.exe
17    // to view traces
18    HMIRuntime.Trace("NimbusAlarming is starting");
19
20    // Path to the alarm.txt file, it is imported into Nimbus
21    let path = "C:\\\\ProgramData\\\\TroSoft\\\\Nimbus Alarm Server 3\\\\Project\\\\Transfer";
22
23    let subs = HMIRuntime.Alarming.CreateSubscription();
24
25    // We need the ISO YYYY-MM-DD HH:NN:SS format, use a static format to
26    // avoid regional settings

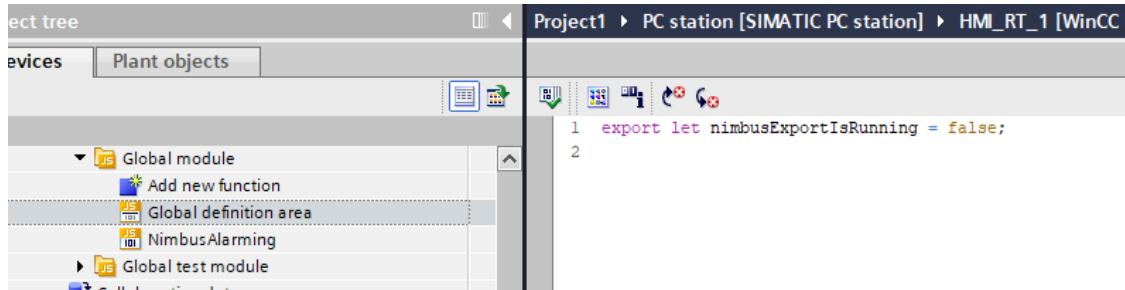
```

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:

The screenshot shows the WinCC Unified PC RT interface. On the left, the 'Project tree' pane is open, displaying a hierarchy of objects: Devices, Plant objects, Parameter set types, Logs, Scheduled tasks, Scripts, Global module, Global definition area, Nimbus Alarming, Global test module, Collaboration data, and Cycles. Under 'Scheduled tasks', there is one entry: 'Start NimbusExport'. On the right, a detailed view of this task is shown in a table:

Name	Trigger	Description
Start NimbusExport	T5s	Execute every 5000 milliseconds.

Below this table, the 'Start NimbusExport [Task]' dialog is open with three tabs: Properties, Events, and Texts. The 'Properties' tab is selected, showing an 'Update' button. The 'Texts' tab contains a table with one row:

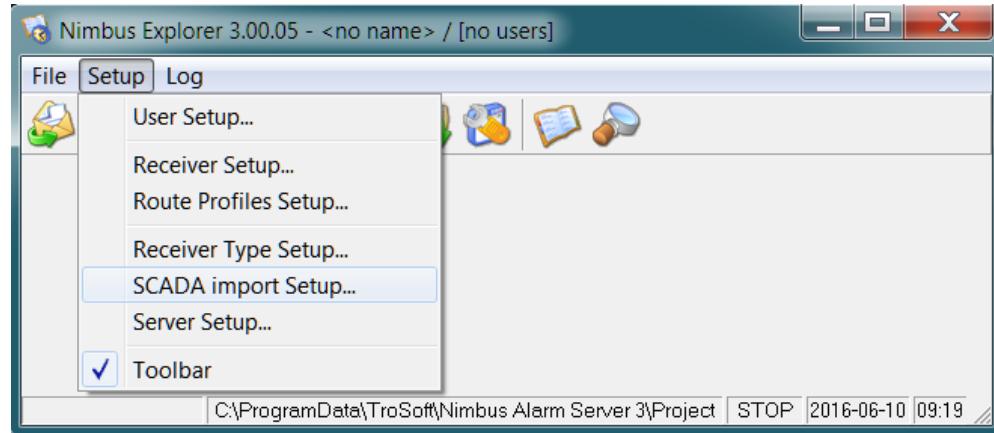
Name	Value
Global module.NimbusAlarming	<Add function>

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 WinCC 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*.

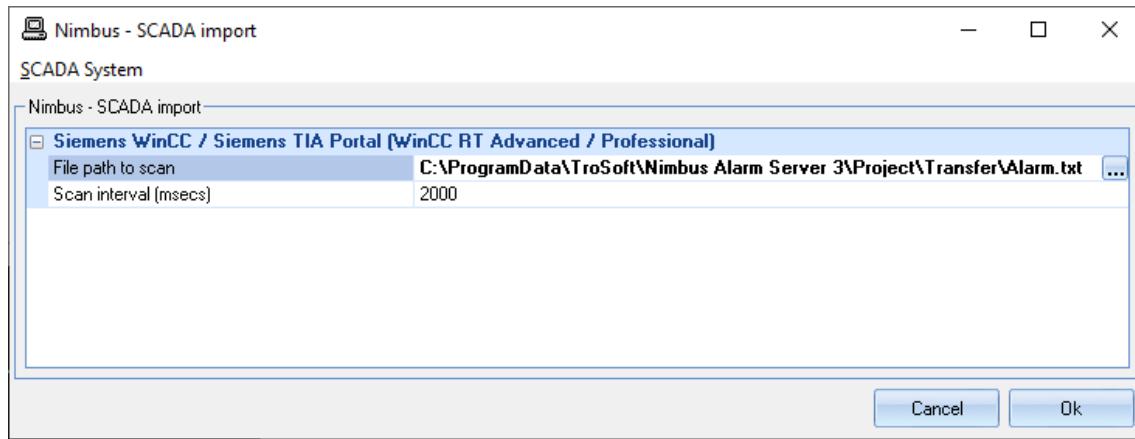


Select *Setup* -> *SCADA Import Setup*.

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 screenshot shows the "Nimbus Explorer 3.00.30 - Test / [no users]" application. The main window displays three alarms: 2CC8, 2CC9, and 2CA, each with a timestamp of 2024-05-07 13:42:42. Below the alarms is a toolbar with icons for file operations and user management. A secondary window titled "Alarm event info - HMI\_RT\_1::084\_VS01\_GT41\_FJV\_TEMP:084\_VS01\_GT41\_FJV\_TE..." is open, showing detailed properties for the selected alarm. The properties include:

Field	Value
Status as number:	1
Status as text:	ACTIVE
Tag [t0]:	HMI_RT_1::084_VS01_GT41_FJV_TEMP:084_VS01_GT41_FJV_TEMP_AL
Area [t1]:	A-alarm
Category [t2]:	15
Name [t3]:	
Description [t4]:	
State from SCA...	Activated
Event Id:	2CC9
PC Date:	2024-05-07
PC Time:	13:42:47
SCADA Date:	2024-05-07
SCADA Time:	13:42:46
SCADA System ...	24
SCADA System ...	Siemens WinCC / Siemens TIA Portal (WinCC RT Advanced / Professional)

# 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].AckTime;  
                    break;  
            }  
            nimbusString += nimbusState + " at " + timeStamp + "  
";  
        }  
    }  
}
```

```

        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");

```