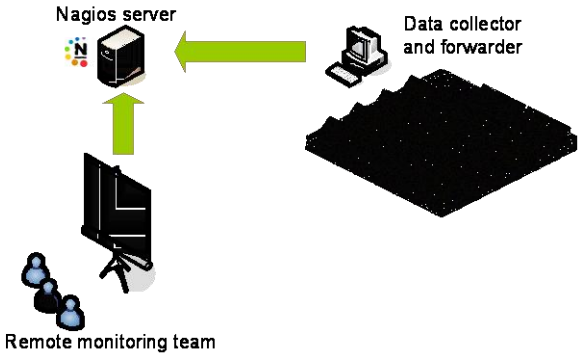
OPEN SOURCE MONITORING APPLICATION IN ENERGY INDUSTRY

Author:
Michal Lackovic , IS/IT enterprise architect at Schneider Electric; michal.lackovic@schneider-electric.com
Code modifications and data collector plugin author:
Vlastmil Kozelka, Infrastructure specialist at Schneider Electric; vlastimil.kozelka@schneider-electric.com

This article is to describe and share our ideas of an open source application monitoring tool that monitors solar plants and other automated systems with specific APIs. Today many good ideas are abandoned due to cost constraints and so affects that possibly, specific business processes are not implemented.

Companies are sometimes lumbered with an open source application where there is a lack of "adequate" support or solid roadmap for further development. Fortunately this does not apply to Nagios, an open source application We selected for our project. We have been using Nagios to monitor and remedy (event handlers) service problems in our infrastructure for many years now. I am sure that the internal logic of the system would satisfy any monitoring team with notifications, escalations or dependency models.

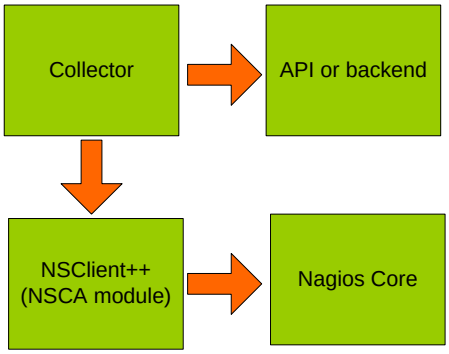
Basically all you need is an agent that provides Nagios with the relevant data.



The platform consists of 3 main components:

- Data collector and forwarder
- Nagios Core server
- Dashboard for monitoring team

Data collector is an on-site pc which receives signals from various PLCs that control a solar plant. Additionally there are two more important functions – to collect data from the PLC backend using an API and covert these figures into the Nagios format and send them to the Nagios server over internet. What if there is no API to connect to? Well, even without an official API you can use the application backend (database or logs) or you can parse its frontend (web). Once you have all figures and statuses collected you then need to translate them so that Nagios understands them. We've selected NSClient++ (www.nsclient.org) as the forwarder of passive checks to Nagios Core server. The only modification of the client is a scheduler for the NSCA checks as we needed different intervals for the service checks. In addition to a service state it is important to send performance data that you can later use for reports or data visualization.



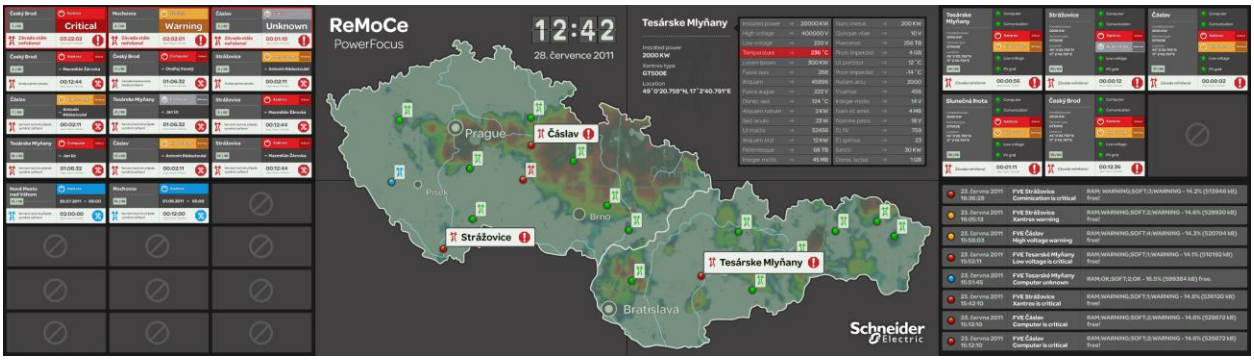
There is no modification of the **Nagios Core** (www.nagios.org) server at all.
We downloaded latest version from the web, compiled and installed it. If you are not familiar with source compiling you want to search for distributions with Nagios packages installed. The most important step here is to get **NSCA daemon** working. This daemon will listen to passive checks that come from the **Data collector** and process them. There are two options with Nagios checks – either you send checks against

collector from Nagios server (active checks) or you send data from the collector to Nagios NSCA daemon. I strongly suggest using passive method to avoid latency problems or firewall restrictions.

At this point you have all your data from your remote plant on your local Nagios server. The next step is to define the service thresholds, notifications, escalations and service check dependencies.

Please pay special attention to this step as you don't want your operators to be flooded by hundreds of alerts. If you are not lucky with the Nagios frontend and optional data visualization using RRD tools, such as Nagiosgrapher or pnp4nagios, you can also use another open source application, Nagvis, to create nice visual maps. These maps help operators to easily determine the root of cause or range of issues.

Even though there are "unlimited" options with Nagvis and other RRD tools we decided to develop our own **dashboard** with custom logic. The main dashboard is a front end, connected to aNagios backend, MK_livestatus. We have installed 8 PCs of 102" LCD screens on the wall (4x2) where the results from Nagios are interpreted in different way. We have also defined a dashboard logic so that monitoring team better understands alerts and events occurring at solar plants.



Currently the platform monitors solar plants but we are ambitious and we would like connect it to intelligent houses in the future. Basically the only limiting factor is an API. Once you succeed in getting relevant data from the application then you can benefit from Nagios logic and data interpretation. In addition to monitoring it allows operators to create solar plant efficiency reports for customers.

SUMMARY

This platform is a combination of an open source components and custom development. Selected components have been on the market for some years now. The applications such as Nagios or Nagvis supports huge communities as well as they offer commercial support programs to mitigate potential risk and ensure business continuity.

Application	Open source	Community support	Commercial support *	Original code modification
Nagios	Yes	Yes	Yes	No
NSCA	Yes	Yes	No	Yes
NSClient++	Yes	Yes	No	Yes
Nagvis	Yes	Yes	Yes	No
NagiosQL	Yes	Yes	No	No
PNP4Nagios	Yes	Yes	No	No
Dashboard	No	No	Yes	N/A
MK_livestatus	Yes	Yes	No	No
Suse Enterprise server	Yes	Yes	Yes	No
* Commercial support provided by developer or vendor				

Although there are some OSS components whose developers do not offer commercial support programs there are also other companies on the market which could provide it for you. This project is a mixture of internal IT resources, open source community and external developers which resulted in cost effective and reliable solution that is designed according remote monitoring team requirements.