MTConnect Student Challenge: Idea Creation

MTMonitor: An IFTTT Channel for MTConnect

Steven Hearndon
Clemson University
(864) 360-1952
shearnd@clemson.edu

12/13/2015

Section I: Abstract

The MTConnect standard allows machine manufacturers to make a wealth of information available to end users in common XML format. But while XML can be easily consumed by custom or off-the-shelf applications, there's not an easy way for end users to simply choose the data fields they'd like to monitor and use them to trigger some action, such as alert them by email.

In this paper I propose a simple tool called MTMonitor that would allow users to browse the data elements made available by their machines and select a subset to be published through a channel on the software service IFTTT. This service would then give them a variety of ways to respond to that data, such as send an email or text message, notify them on their smart phone, or even trigger another device.

Section II: Proposed Idea

The MTConnect standard has made it easier to acquire useful data from manufacturing machinery, but this task is still not easy enough. To make use of this valuable information, users still need software to properly digest these feeds, which means companies must either hire developers to write custom applications or buy off-the-shelf software, hoping it will give them what they need. Both of these options represent a significant expense, and yet neither of them typically allow for truly flexible experimentation with the data. ***The goal of this project is to develop software that makes MTConnect data more accessible to end users by providing a simple interface to show users what information is available from their machines and allow them to select the information they would like to monitor. The primary purpose of this software would be to direct the selected information to IFTTT, a free internet service which allows users to respond to data from a variety of sources and take action in a variety of ways.***

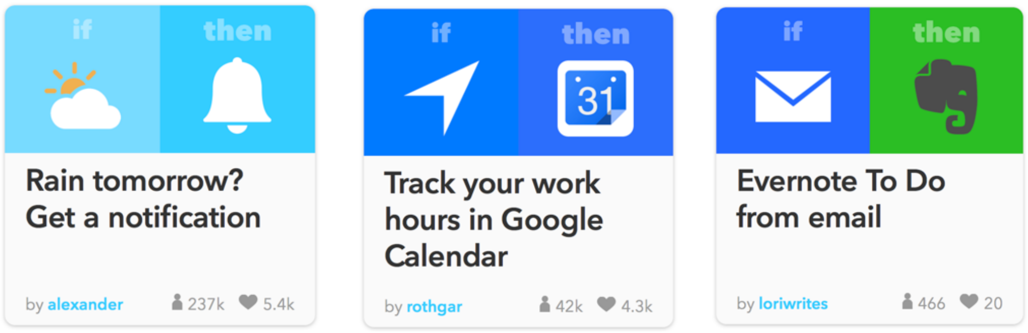
The MTConnect standard requires machine data to be made available as an XML stream. While this allows the data to be easily consumed by custom or off-the-shelf applications, it's not practical for human users to monitor this feed directly. As a result, users must rely on available or custom software solutions for monitoring their machines. And while either of these options can be a viable solution when a monitoring plan has already been clearly established, neither promote rapid prototyping at the shop floor level.

Say, for instance, that a shop manager has an idea for improving workflow through his area, but it depends on collecting certain timely information from the machines he's responsible for. First of all, he's not even sure if these machines provide the data he needs, and to find out means putting a request in with the IT department (who don't understand what he's asking for half of the time). This request costs money and time, and the response he gets is usually questionable. But even if he confirms that the information is available, in order to monitor this data and receive timely notifications, he'd have to request that software be purchased or developed (he doesn't care which as long as it works). Even if this costly request is granted (which seems unlikely), by the time he receives what he's requested (turnaround time may be 3 to 6 months), his situation has changed and the data is no longer quite as relevant. Instead, he tries to develop a solution himself using clipboards and Excel, but ultimately it doesn't work consistently. And so he just goes about business as usual.

Now let's imagine the same scenario, but this time the manager has MTMonitor, a software program that is either running locally on his company's servers or as a cloud service. In this case he only needs the URLs (network addresses) for his machines (which he acquires from the IT department in about a week or so, but he only needs to get them once). Once his machines are added to MTMonitor, it listens to the MTConnect feeds and shows him what information is available in plain English. He selects the data fields that he's interested in and then makes them available to his IFTTT account. Now he's getting somewhere.

IFTTT, which stands for "IF This Then That" (ifttt.com), is a free web service that allows a user to create *recipes* that do some *action* based on some *trigger*. These recipes run on their servers in the background and make use of any of the almost 250 *channels* IFTTT currently has available, including 3rd-party channels (like Evernote, Facebook, Google, Salesforce, and Microsoft Office) and built-in channels (like weather, email, phone, and their iOS and Android apps). Each of these channels offer possible triggers and/or actions.

A trigger is the "if this" part of the recipe, such as "If I receive an email from dan@mywork.com…" or "If it's going to rain tomorrow…" Each channel that includes triggers (most but not all do) has various options depending on what the source for the channel does. Likewise, the action is the "then that" part of the recipe, such as "send an email with this info", "add an entry to this Google Spreadsheet", or "notify my iPhone". With all of the available triggers and actions from so many channels, there are lots of possibilities.


if
then
Rain tomorrow? Get a notification
by alexander
237k
5.4k
if
then
Track your work hours in Google Calendar
by rothgar
42k
4.3k
if
then
Evernote To Do from email
by loriwrites
466
20

To continue our scenario, imagine the shop manager uses the MTMonitor channel on IFTTT to log the start and stop of all of his machines to a Google Spreadsheet to analyze their current workflow. He also has a hunch that a certain piece of data on one of his machines might be an early indicator of a problem that can bring the machine down, so he sets up a recipe to log any time that field goes over a certain value. Sure enough, after a week or two of collecting data, he sees that that information correlated with most of the times the machine went down. Based on this, he sets up a new recipe to send him and his operator an iPhone notification anytime it happens going forward. They are then able to respond to the issue quicker, saving time and money by preventing downtime.

Section III: Technical Requirements

The main element in this plan is the MTMonitor software itself, which serves as a bridge between the machines streaming MTConnect data and the IFTTT service. This software would have a web interface and, ideally, be a cloud hosted solution. This way, machine shops of all sizes would have access to the software as a service (SaaS) without the need for installing and maintaining a server. For this option, though, the MTConnect data would have to be made available over the internet via a public IP address for each machine. There would still be security measures and credentials involved, but this exposure would be enough to make some companies wary.

Another option then would need to be a self-hosted solution, where MTMonitor is offered as a software product that can be installed on a company's own server. With that said, the end goal of MTMonitor is still to send the data to an IFTTT channel, which is a 3rd-party hosted solution. So if tight security is of upmost concern, this MTMonitor/IFTTT combination might not be the best choice. The reality is that this setup is best for companies that need the flexibility to experiment and innovate quickly, and who are willing to trade tight restrictions for the versatility this solution can provide.

The software itself would not need to be elaborate. The earliest version would simply need to be able to receive the XML stream from a machine, parse it (separate the stream into understandable pieces), and then present it to the user in a meaningful way. It would not need to understand the data, only the patterns, such as how often a field gets repeated in relation to others (which denotes a low-level detail item versus a high-level data field). This is fairly trivial due to XML's hierarchical structure. Essentially, the software would simply tell the user, "These are the data fields I observe, this is how often they get updated, and this is the type of information they hold (number ranges and such)." It would then be up to the user to decide what information is important. Future versions of the product could even allow users to contribute their knowledge (such as which fields are important on which machines and why) and then make that knowledge available to all other users.

From there, a user would setup up Monitoring Groups by creating a named group, selecting fields to add to it, and setting how often it would get updated (e.g. every second, minute, hour, etc.). In IFTTT then, it would be as simple as making a recipe that said, "When this *field* in this *group* has this *value* (or a *value* in this *range*), then…" and then add an IFTTT *action* to it. That action could be one of the many available on IFTTT, including the several mentioned in the scenario above. There are even physical devices that can interact with IFTTT, including a variety of "smart" light bulbs. So it would be very easy at that point to have a light near the machine turn on or start flashing when a field went above a certain value. There would be so many possibilities to explore.

The final challenge at this point would be creating an IFTTT channel. An API is currently in development to allow anyone to create a channel, but it is not yet available. Since there are many 3rd-party channels on the IFTTT service though, it should be possible to partner with them to create a channel for the MTMonitor product. Also, according to their website, "Most IF Recipes check for new Trigger data every 15 minutes. Some Recipes run even faster." So the MTMonitor channel would have to be setup with them in such a way to allow its triggers to respond much quicker (every second or so).

Section IV: Benefits and Impact

Hopefully by now the benefits are fairly obvious, especially as seen in the scenario in Section II. But it's certainly worth mentioning some of the major advantages this system would have over others.

First of all, this system would allow for flexible and rapid prototyping like no other. With a minimal amount of training, shop floor managers and machine operators could generate a recipe within minutes and see almost immediate results. In most shops today, shop managers would either have to go through another department to get access to the machine data or access some elaborate software that would require extensive training. This hassle and slow turnaround time discourages these end users from coming up with creative ideas to improve their processes.

Secondly, this system could be put in place at a lower cost than most alternatives. The ideal pricing model for MTMonitor would be to charge a reasonable monthly fee for each machine being monitored. From there, users could create as many Monitoring Groups and IFTTT recipes as they needed. (Did I mention that IFTTT is free?) And if the cloud-based solution is chosen, there would be no installation or maintenance costs (a rarity for most manufacturing technology solutions).

Finally, based on the above two benefits, the real return on such a minimal investment would be from the specific gains in productivity and efficiency that empowered shop managers and operators can make using this system. As it stands now, how many ideas and gains are being lost because these users have no good means of implementing them? A system like this enables them to improve the processes they're responsible for and make a positive impact on the bottom line. As such, this simple solution could produce profound results.