**Creating AppD Policies**

The *batmon.py* script polls the REST URL-based JSON files that are registered in lookup.xml, based on the <execution-frequency-in-seconds> parameter defined in *monitor.xml*.

Each job status condition is written to one or two tables:

1. When parsing each JSON, the script auto-writes each job result to the native AppD events database.
2. When parsing each JSON, the script will auto-write WARN & ERROR conditions ONLY to the external AppD-events-push table. It is these events that are subject to policy-based alerting.

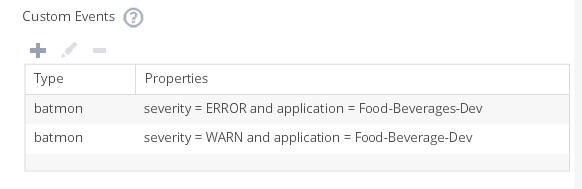
Create AppD policy using the same ‘Policies’ interface for the application connected to the server agent running the Batmon extension (example AppD-Dev container policy).

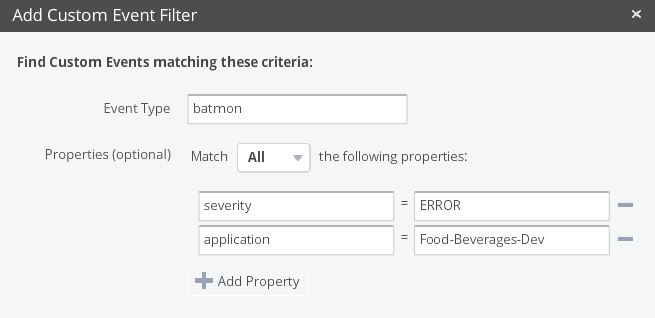
Customize the Health Rule Violation Events as usual, but do not specify any health rules. Instead, create **Custom Events** with following values:

Type = batmon

Properties = severity and application

Values = ERROR/WARN and app name defined in JSON file

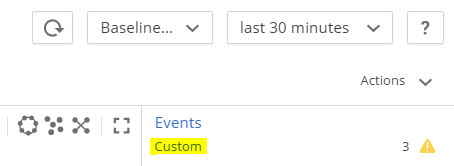


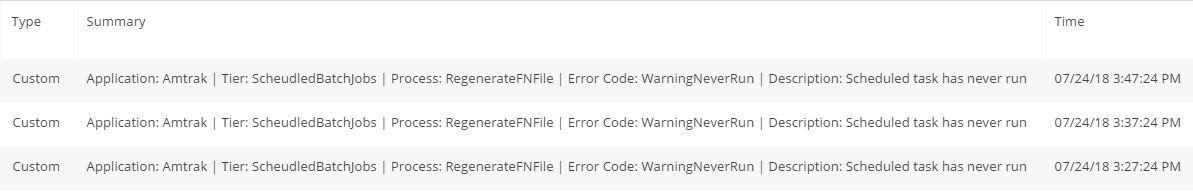


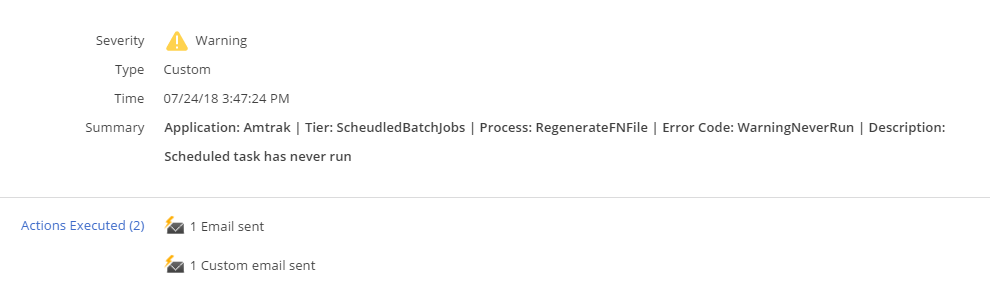
**Viewing events on AppD Application Dashboard**

All related Batmon events can be viewed by selecting ‘Application Dashboard’ interface for the application connected to the server agent running the Batmon extension.

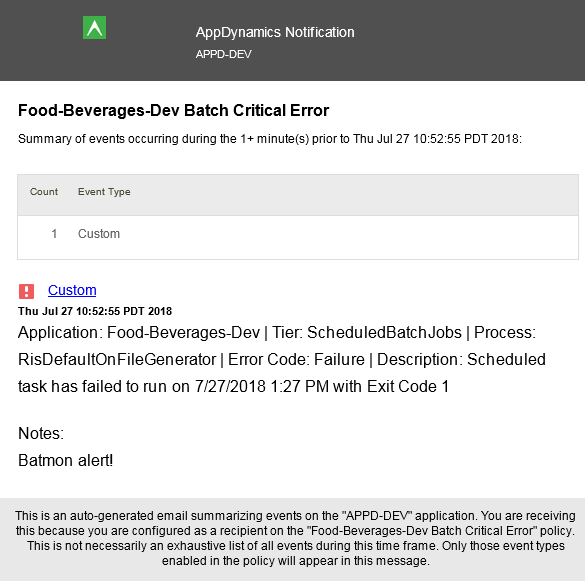
(example AppD-Dev container)



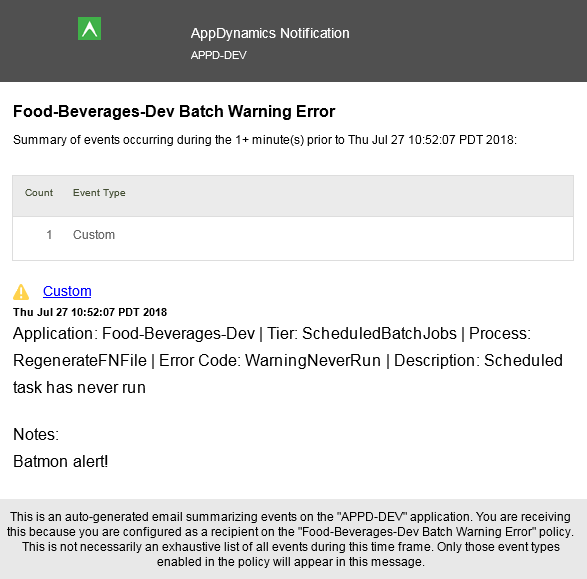




**Sample Batmon Critical Alert:**



**Sample Batmon Warning Alert:**



**\*\*\*Policy Limitation\*\*\***

20180725 – After some testing, it seems that Batmon policy has limitation of only sending one email simultaneously using a single policy. Workaround means creating a separate policy for EACH alert type!

Example:

