Interlink Software Integration with AppDynamics

Contents

Integration for AppDynamics

Overview

There are two ways to integrate BES with the AppDynamics solution:

1. Configure AppDynamics to use a webhook on the Interlink BES server. To setup a webhook use Interlinks HTTP Event tool, piHTTP.
2. Polling the AppDynamics REST API by using Interlinks REST client tool, piRestClient.

Configuration – Webhook Solution (piHTTP)

The piHTTP integration is installed on the BES and is configured to receive HTTP POST events in JSON or XML format.

Prerequisites

- BES 3.8.0 +
- AppDynamics instance configured to forward health rule violations to the piHTTP webhook
- iss-bes-pihttp RPM within a Yum repository (you may use a local repository)

Installation

To run the install, logon to the BES server as root.

Then use the following command to install piHTTP:

```
yum install iss-bes-pihttp
```

If there are any issues reported from the install process, please contact Interlink Software Support for assistance.

Configure a piHTTP Message Channel as a Webhook Listener

A default piHTTP message channel is created during installation. This channel will need to be configured as a webhook listener.

Note: For illustrative purposes, we shall use the default piHTTP message channel as a webhook for AppDynamics to communicate with BES.

Setup the configuration file

During installation of the integration a default configuration file is copied to $PPHOME/cfg/piHTTP.cfg.

As the ppadmin user, update the configuration file to setup the piHTTP channel as a webhook listener.

Below is an example that configures the piHTTP channel as a webhook listener running on port 9000.

```
# Port which the integration will listen on
http.listen.port = 9000

# Whether to use HTTPS - True/False, optional, default = false
http.useSsl = true

# if http.useSsl = true, a JKS (Java Key Store) is required
http.sslKeystore = /opt/ISS/config/security/keystore.jks
http.sslKeystorePassword = myPassword

# logging at debug level
log.debug = true


# Several formats can be set for the incoming events. Set the key to a string
which you would expect to see in the event, the corresponding format value will
then be used. The values in brackets should match JSON Keys from the event
# Note: JSON values should be in []

# The following maps the AppDynamics incoming JSON payload that is received by the
webhook
format.1.key = controllerUrl

format.1.value = APPD_WEBHOOK id = [action.id] | url = [controllerUrl] | name =
[latestEventDisplayName]    |    deepURL    =    [url]    |    applicationName    =
[latestEventDisplayName] |
```

After updating the configuration file you will need to restart the piHTTP message channel:

```
ppCycle -n piHTTP
```

Logging

Logging configuration is set within the piHTTP-logging.xml file in the /opt/ISS/POWERpack/cfg folder.

The following example shows the settings within the piHTTP-logging.xml:

```
<configuration>
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${PPLOG}/${logFileName}.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>${PPLOG}/${logFileName}.%i.log.zip</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>9</maxIndex>
    </rollingPolicy>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <maxFileSize>10MB</maxFileSize>
    </triggeringPolicy>
    <encoder>
      <pattern>%date [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>

  <root level="INFO">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

To change the logging level, update the *root level* to ERROR, WARN, INFO, DEBUG or TRACE.

Any changes to the file require a restart of the piHTTP message channel:

```
ppCycle -n piHTTP
```

Next Steps

Now that the webhook is configured please refer to the following section of this document "Receiving Alert Information from AppDynamics".

Configuration – Polling Solution (piRestClient)

Prerequisites

- BES 3.8.0 +
- AppDynamics instance running the REST API
- iss-bes-pirestclient RPM downloaded from ftp.interlistsoftware.com/rpms and added to a Yum repository (you may use a local repository)

Installation

To run the install, logon to the BES server as root.

Then use the following command to install piRestClient:

```
yum install iss-bes-pirestclient
```

If there are any issues reported from the install process, please contact Interlink Software Support for assistance.

Configuration of the piRestClient

The installation creates a default piRestClient message channel that will require configuration before use.

Note: For illustrative purposes, we shall use the default piRestClient message channel

Setup the configuration file

During installation of the integration a default configuration file is copied to the following location:

$PPHOME/cfg/piRestClient.yml

As ppadmin user, open the piRestClient.yml file and update the configuration to feed metrics data from AppDynamics into BES, as shown in the example below.

Example Configuration for the piRestClient.yml

```
workingDirectory: /var/spool/ISS/POWERpack  ①
routing:
  routes: ②
    - name: AppDMetricsRequest  ③
      enabled: true  ④
      schedule-milliseconds: 300000 ⑤
      initialDelay: 25000  ⑥
                                                                                               ⑦
      url: https://interlinksoftware-nfr.saas.appdynamics.com/controller/rest/applications/742/metric-data
      http-method: get  ⑧
      headers:  ⑨
      query-parameters: ⑩
        - key: "metric-path"
          value: "Application Infrastructure Performance|*|Individual Nodes|*|Hardware Resources|CPU|%Busy"
        - key: "time-range-type"
          value: "BEFORE_NOW"
        - key: "duration-in-mins"
          value: "15"
      body: ⑪
      splitter-expression: /metric-datas/metric-data ⑫
      filter-expression: ⑬
      responseType: xml ⑭
      authentication: ⑮
        type: "Basic Auth"
        username: interlinksoftware-nfr@interlinksoftware-nfr
        password: ENC(SyttCmz1x/2WnHa8Vuayr7bc5fuB5iuy)
      templating:
        path: file:///opt/ISS/POWERpack/cfg/AppDMetricsRequest.vm  ⑯
```

Item	Description
1	*Integration work file location. This is where the duplicate message processor will dump the duplicate message cache*
2	*List of routes/rest endpoints that will be called*
3	*Name of the route (must not be empty and must be unique)*
4	*If false, this route will not be loaded at startup*
5	*The polling schedule on which this route will be called (default is 20000 and must be a positive number)*
6	*The initial delay in milliseconds before the first call to the route's endpoint will be made. After the first call, subsequent calls to the endpoint will be made on the schedule-milliseconds interval. So if initial-delay is 20000 and schedule-milliseconds is 10000 then the application will wait 20 seconds on application startup before making the first call and then each subsequent call will be every 10 seconds (default is 20000)*
7	*The path for the target endpoint*
8	*The method to use. Options are GET (default), POST and PUT (must be set to one of the options and is case insensitive)*
9	*A list of key value pair headers to be passed to the endpoint*
10	*The query parameters to pass to the endpoint*
11	*The body to pass to the endpoint*
12	*A json or xml expression, as per the response content-type, within the payload body that will be used to split the response into sub messages*
13	*A json or xml expression, as per the response content-type, within the payload body that will be used to filter the response*
14	*Indicates what content type to expect back from the API being queried (defaults to JSON)*
15	*Auth headers, if required*
16	*The template location used to format the endpoint response (must not be empty)*

http-method

Accepts POST, PUT or GET requests.

url

This is the url of the REST endpoint that will be called by the client application. The url must not contain any query-parameters.

headers

A list of key value pair http headers to be passed to the endpoint.

For example, the following header,

Content-Type: application/json

would be specified in the config as:

```
- key: Content-Type
  value: application/json
```

query-parameters

A list of key value pair query parameters to be passed to the endpoint.

For example, the following key value pairs:

```
url: https://interlinksoftware-nfr.saas.appdynamics.com/controller/rest/applications/742/metric-data

http-method: get

headers:

query-parameters:

  - key: "metric-path"

    value: "Application Infrastructure Performance|*|Individual Nodes|*|Hardware Resources|CPU|%Busy"

  - key: "time-range-type"

    value: "BEFORE_NOW"

  - key: "duration-in-mins"

    value: "15"
```

Would result in the following URL:

```
https://interlinksoftware-nfr.saas.appdynamics.com/controller/rest/applications/742/metric-data?metric-
path=Application%20Infrastrucuture%20Performance%7C*%7CIndividual%20Nodes%7C*%7CHardware%20Resources%7CCPU%7
C%25Busy&time-range-type=BEFORE_NOW&duration-in-mins=15
```

splitter-expression

A JSON or XPath expression to an array in the response payload body as per the returned content type.

The contents of the array will be taken as individual messages and passed into the rest of the pipeline and processed as normal.

For example, if a rest endpoint returned:

```
<metric-datas>

  <metric-data>

    <metricId>123</metricId>

    …

  </metric-data>

  <metric-data>

    <metricId>456</metricId>

    …

   </metric-data>

   <metric-data>

    <metricId>789</metricId>

    …

   </metric-data>

</metric-datas>
```

With a splitter expression of the following:

```
splitter-expression: "/metric-datas/metric-data"
```

This would split the original message into three new messages, with each being one of the contents of the *foos* array.

The messages are passed down the processing pipeline as normal and the original message is discarded.

filter-expression

A JSON or XPath expression within the payload body that will be used to filter the response.

The filter is particularly useful where a given API is not able to allow filtering on some of its fields.

This is an inclusive filter and as such will only send down to BES the data that matches the filter.

If the payload has been split using a *splitter-expression*, then this filter-expression will be applied to each split payload, otherwise it will be applied to the whole payload.

For example, if a rest endpoint returned:

```
<metric-datas>

  <metric-data>

    <metricId>123</metricId>

    …

    <metricValues>

      <metric-value>

        …

        <occurrences>0</occurrences>

        …

      </metric-value>

    </metricValues>

  </metric-data>

</metric-datas>
```

A filter expression of the following:

```
filter-expression: “/metric-datas/metric-data/metricValues/metric-value/occurrences > 0"
```

Would prevent the payload from being passed down the processing pipeline and would be discarded.

Once you have finished configuring the message channel, start it using the following command:

```
ppStart -n piRestClient
```

Templating

The template file describes the mapping of a given API response payload to a pipe delimited BES payload.

The integration supports various templating engines. For the example below, we are using the Freemarker template engine to demonstrate the field mapping.

Example templating file

```
MetricId=${body.metricId} | MetricName=${body.metricName} | Occurrences=${body.metricValues.metric\-
value.occurrences} | Current=${body.metricValues.metric\-value.current} | Min=${body.metricValues.metric\-
value.min} | Max=${body.metricValues.metric\-value.max} |
```

The key value pairs are pipe delimited and in the format of:

BES field=API Response field

Encryption

If you need to encrypt sensitive properties in the config file, use the following command:

```
piRestClient -e "somethingiwanttoencrypt"
```

This will return the encrypted property that can be added to the configuration file.

Logging

The integration has a configuration file piRestClient-logging.xml that describes the level of logging. The file is located in the /opt/ISS/POWERpack/cfg directory.

Example Logging Configuration file

```
<configuration>
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${PPLOG}/${logFileName}.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
       <fileNamePattern>${PPLOG}/${logFileName}.%i.log.zip</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>9</maxIndex>
    </rollingPolicy>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <maxFileSize>10MB</maxFileSize>
    </triggeringPolicy>
    <encoder>
      <pattern>%date [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>

  <root level="INFO">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

To change the logging level, update the *root level* field to INFO, DEBUG, WARN, ERROR or TRACE.

The log file for the integration is piRestClient.log and it is located I the /var/log/ISS/POWERpack ($PPLOG) folder.

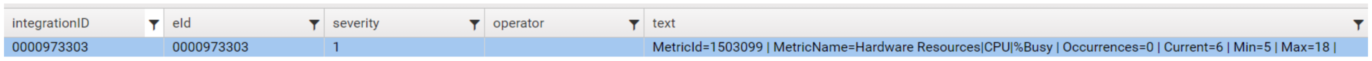
Any change to either the configuration yaml or logging xml files will require a restart of the message channel using the following command:

```
ppCycle -n piRestClient
```

Receiving Alert Information from AppDynamics

Once the configuration is complete for the webhook or polling options and the message channels are active, the solution will start to receive information from AppDynamics for events such as metrics data and health rule violations.

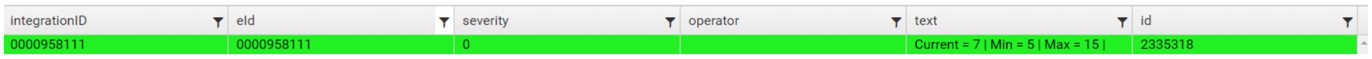
Normalization rules should be created for the message channel using Interlinks SMARTView tool. Prior to such rules being created the alerts will come through as "passthrough" alerts – the default alerts that indicate the information has not been processed by any rules. These are light blue, as shown in the following example:



integrationID	eId	severity	operator	text
0000973303	0000973303	1		MetricId=1503099 | MetricName=Hardware Resources|CPU|%Busy | Occurrences=0 | Current=6 | Min=5 | Max=18 |

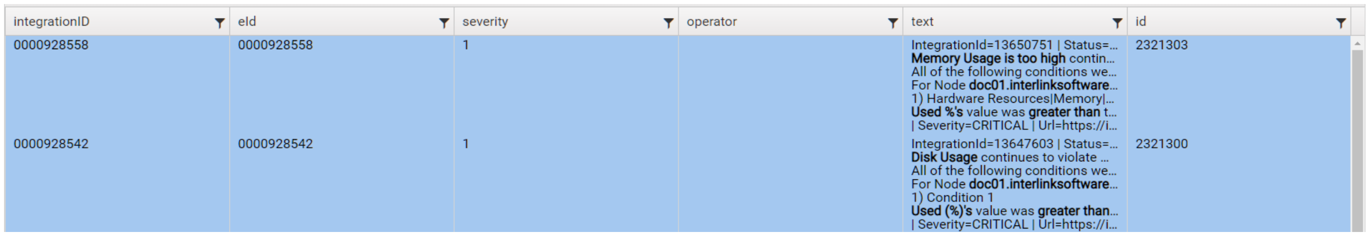
Once the normalization rules are in place, the alerts will use colors to visually represent the severity of the original alert from AppDynamics.

Below, we see a non-critical alert in green:



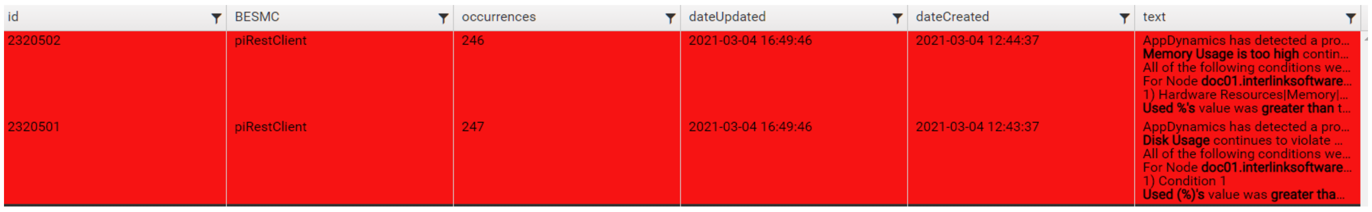
integrationID	eId	severity	operator	text	id
0000958111	0000958111	0		Current = 7 | Min = 5 | Max = 15 |	2335318

As well as feeding metrics data to BES we can also feed health-rule violations. The following is an example of two "passthrough" alerts originating from AppDynamics Memory and Disk health-rule violations.



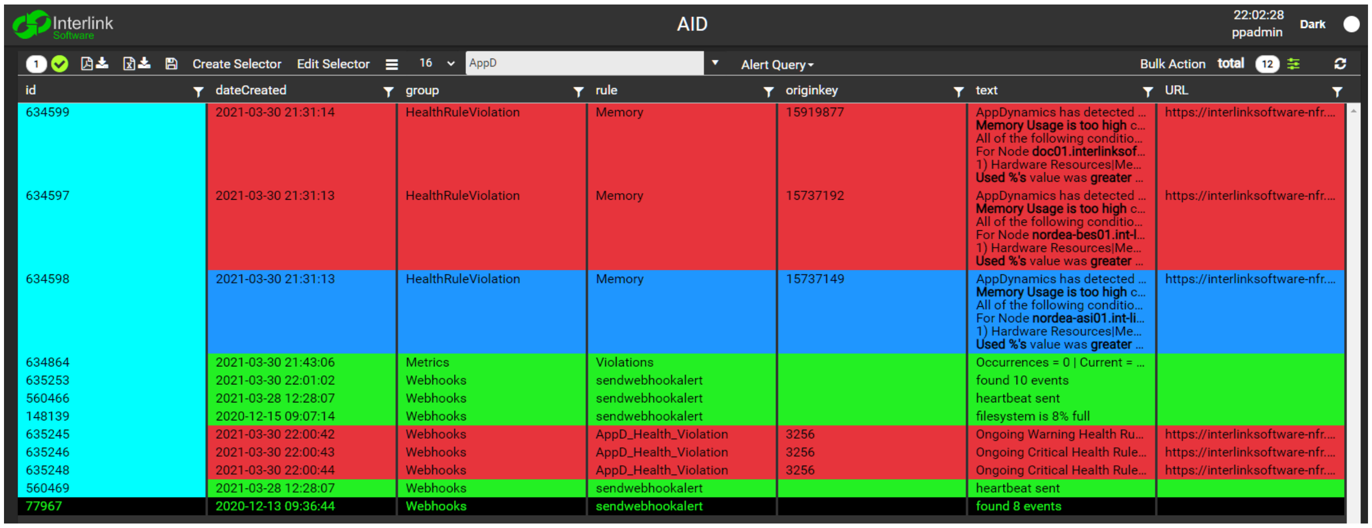
integrationID	eId	severity	operator	text	id
0000928558	0000928558	1		IntegrationId=13650751 | Status=... **Memory Usage is too high** contin... All of the following conditions we... For Node **doc01.interlinksoftware...** 1) Hardware Resources|Memory|... **Used %'s** value was **greater than** t... | Severity=CRITICAL | Url=https://i...	2321303
0000928542	0000928542	1		IntegrationId=13647603 | Status=... **Disk Usage** continues to violate ... All of the following conditions we... For Node **doc01.interlinksoftware...** 1) Condition 1 **Used (%)'s** value was **greater than...** | Severity=CRITICAL | Url=https://i...	2321300

After creating normalization rules for these, we can see the entries represented as critical alerts in red:



id	BESMC	occurrences	dateUpdated	dateCreated	text
2320502	piRestClient	246	2021-03-04 16:49:46	2021-03-04 12:44:37	AppDynamics has detected a pro... **Memory Usage is too high** contin... All of the following conditions we... For Node **doc01.interlinksoftware...** 1) Hardware Resources|Memory|... **Used %'s** value was **greater than** t...
2320501	piRestClient	247	2021-03-04 16:49:46	2021-03-04 12:43:37	AppDynamics has detected a pro... **Disk Usage** continues to violate ... All of the following conditions we... For Node **doc01.interlinksoftware...** 1) Condition 1 **Used (%)'s** value was **greater tha...**

The below image shows a combination of the webhook events and the polled events within the AID.



id	dateCreated	group	rule	originkey	text	URL
634599	2021-03-30 21:31:14	HealthRuleViolation	Memory	15919877	AppDynamics has detected ... **Memory Usage is too high** c... All of the following conditio... For Node **doc01.interlinksof...** 1) Hardware Resources|Me... **Used %'s** value was **greater** ...	https://interlinksoftware-nfr....
634597	2021-03-30 21:31:13	HealthRuleViolation	Memory	15737192	AppDynamics has detected ... **Memory Usage is too high** c... All of the following conditio... For Node **nordea-bes01.int-l...** 1) Hardware Resources|Me... **Used %'s** value was **greater** ...	https://interlinksoftware-nfr....
634598	2021-03-30 21:31:13	HealthRuleViolation	Memory	15737149	AppDynamics has detected ... **Memory Usage is too high** c... All of the following conditio... For Node **nordea-asi01.int-li...** 1) Hardware Resources|Me... **Used %'s** value was **greater** ...	https://interlinksoftware-nfr....
634864	2021-03-30 21:43:06	Metrics	Violations		Occurrences = 0 | Current = ...	
635253	2021-03-30 22:01:02	Webhooks	sendwebhookalert		found 10 events	
560466	2021-03-28 12:28:07	Webhooks	sendwebhookalert		heartbeat sent	
148139	2020-12-15 09:07:14	Webhooks	sendwebhookalert		filesystem is 8% full	
635245	2021-03-30 22:00:42	Webhooks	AppD_Health_Violation	3256	Ongoing Warning Health Ru...	https://interlinksoftware-nfr....
635246	2021-03-30 22:00:43	Webhooks	AppD_Health_Violation	3256	Ongoing Critical Health Rule...	https://interlinksoftware-nfr....
635248	2021-03-30 22:00:44	Webhooks	AppD_Health_Violation	3256	Ongoing Critical Health Rule...	https://interlinksoftware-nfr....
560469	2021-03-28 12:28:07	Webhooks	sendwebhookalert		heartbeat sent	
77967	2020-12-13 09:36:44	Webhooks	sendwebhookalert		found 8 events	

For more information about the AppDynamics integration visit.

To discover more about Interlink Software's products visit.