



EMC® Watch4net®

Version 6.3u2

Generic Event Listener Admin Guide

P/N: pending

Rev : 41258

EMC²

Copyright ©2009 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published April, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>).

Contents

Preface	iii
Purpose	iii
Intended Audience	iii
Prerequisites	iii
Document Organization	iii
Where to get help	v
1 <i>Event Processing Manager</i> Configuration	1
2 <i>Generic Event Listener</i> Configuration	3

Preface

Purpose

The *Generic Event Listener* is a *Processing Element* designed to listen on a socket for incoming serialized events. It handles event de-serialization and allows the received events to be forwarded to any other *Processing Element*.

Intended Audience

This guide is intended for advanced *APG* Administrators that have a good knowledge of the *APG Event Processing* framework.

Prerequisites

The *Generic Event Listener* can only be used within an instance of *Event Processing Manager*. Thus, to be able to use this *Processing Element*, you must have a working *Event Processing Manager* ready.

Document Organization

Where to get help

EMC support, product, and licensing information can be obtained as follows:

PRODUCT INFORMATION - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

<https://support.emc.com>

TECHNICAL SUPPORT - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

1

Event Processing Manager Configuration

To use a *Generic Event Listener* instance, please follow this *Event Processing Manager* configuration file example:

```
<!-- an other element (for example a database writer) -->
<processing-element name="Database-Writer" config="..." data="Next-Element" />

<!-- Generic Event Listener -->
<processing-element name="Generic-Event-Listener" config="Generic-Event-Listener/Default/conf/
generic-event-listener.xml" data1="Database-Writer[data]" data2="Database-Writer[data]" />
```

The *Generic Event Listener* can forward its events to any kind of *Processing Element*. It can be configured to use as many output streams as required. Each output stream corresponds to a socket on which the *Generic Event Listener*. In the example above, the *Generic Event Listener* will listen on two different ports and forward all the events received on both ports to the *data* stream of the *Database Writer*.

Since the *Generic Event Listener* listens on sockets, no input streams can be configured. However, as mentioned above, any number of different output streams can be defined. Each output stream that is declared in the *Event Processing Manager* configuration must also appear in the *Generic Event Listener* configuration (see [section 2](#)).

2

Generic Event Listener Configuration

This example shows a *Generic Event Listener* configuration that could be used in the *Event Processing Manager* configuration file example in section 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/Events/GenericEventListener"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Events/GenericEventListener_generic-event-
listener.xsd">

  <!--
  The max-threads attribute defines the maximum number of simultaneous connections.
  The default value, 0, means that there is no limit.
  -->
  <event-receiver xsi:type="GenericTcpSocketReceiver" encoding="UTF-8" max-threads="0">
    <!-- Data received on port 22222 will be forwarded on the data1 OUTPUT stream. -->
    <data-stream>data1</data-stream>
    <port>22222</port>
```

```
<deserializer>
  <class>com.watch4net.events.common.serialization.GenericEventDeserializer</class>
</deserializer>
</event-receiver>

<event-receiver xsi:type="GenericTcpSocketReceiver" encoding="UTF-8" max-threads="1"
timeout="1000">
  <!-- Data received on port 22223 will be forwarded on the data2 OUTPUT stream. -->
  <data-stream>data2</data-stream>
  <port>22223</port>
  <!-- Use spaces instead of tabs to separate events in the OUTPUT stream. -->
  <deserializer separator="␣">
    <class>com.watch4net.events.common.serialization.GenericEventDeserializer</class>
  </deserializer>
</event-receiver>

</configuration>
```

A configuration file must contain at least one `event-receiver`. Each receiver listens for incoming events on a single port and redirect them to an output stream.

Receivers are defined by the elements:

- `<data-stream>` Defines the unique name of the output stream. This name will be used in the *Event Processing Manager* configuration (see section 1).
- `<port>` The port on which this receiver will listen for incoming events.
- `<deserializer>` Defines the *Deserializer* implementation that will be used to transformed serialized events back to standard events.
 - *separator* Optional attribute specifying the character that marks the end of an event and the beginning of a new one in the serialized form. Its default value (TAB) should be a good choice most of the time.
 - `<class>` The java class that corresponds to the *Deserializer* that will be used. This value should not be changed.

Receivers' behavior is also controlled by optional attributes:

- *encoding* Encoding used to serialize the events that are received. If omitted, the default value is set to *UTF-8*.

- *max-threads* Maximum number of connections that can be established at any given time. The default value is 0, meaning that there is no maximum.
- *timeout* Amount of time to wait for incoming data on a connection before it is dropped (in milliseconds). The default value is 0, disabling the timeout.

Testing the configuration The *Generic-Event-Sender* is a small tool that can be used to write serialized events on a socket. It is a good way to test the *Generic Event Listener*'s configuration.

The *Generic-Event-Sender* is provided in a separate package. Once the package is installed, events can be generated and sent on a specific port using the script `send-event.sh` on Unix or `send-event.cmd` on Windows. The script is located in the folder `<APG>/bin/`, where `<APG>` is the installation directory of APG.

RawValueDeserializer It is possible, using the deserializer class *RawValueDeserializer*, to convert raw values into generic events.

Configuration example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/Events/GenericEventListener"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Events/GenericEventListener_generic-event-
listener.xsd">

  <event-receiver xsi:type="GenericTcpSocketReceiver" encoding="UTF-8" max-threads="0">
    <data-stream>data</data-stream>
    <port>22222</port>
    <deserializer>
      <class>com.watch4net.events.common.serialization.RawValueDeserializer</class>
    </deserializer>
  </event-receiver>

</configuration>
```

For example, if the following raw value is sent to the generic event listener:

```
+r 1200000000 group var1 25 source=MySource device=MyDevice name=MyName
```

The resulting event will look like this:

```
(k) RV_Variable = var1      (DEFINITION,STRING)
    RV_GroupName = group    (DEFINITION,STRING)
    RV_Action    = r        (OCCURRENCE,STRING)
(k) RV_Timestamp = 1200000000 (OCCURRENCE,INT)
    RV_Value     = 25.0      (OCCURRENCE,FLOAT)
    source       = MySource  (DEFINITION,STRING)
    name         = MyName    (DEFINITION,STRING)
    device       = MyDevice  (DEFINITION,STRING)
```

Advanced Options Every *Deserializer* can be configured to use specific field deserializing components. Adding a field serializer is needed only when a field doesn't have a default deserialization behavior (typically, a user-defined object). A custom field deserializer can be added as follows:

```
...
<deserializer>

<class>com.watch4net.events.common.serialization.GenericEventDeserializer</class>
>
...
<field type="OBJECT">
  <class>CustomFieldSerializer</class>
  <parameter name="param1">value1</parameter>
  <parameter name="param2">value2</parameter>
  ...
</deserializer>
...
```

There are currently no predefined field deserializer implementations. Please contact Watch4Net support if you feel like a custom field deserializer is needed.