



EMC[®] Watch4net[®]

Version 6.3u2

Generic Event Transmitter Admin Guide

P/N: pending

Rev : 40032

EMC²

Copyright ©2011 - 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	iv
Where to get help	v
1 <i>Event Processing Manager</i> Configuration	1
2 <i>Generic Event Transmitter</i> Configuration	3

Preface

Purpose

The *Generic Event Transmitter* is a *Processing Element* designed to allow the reception and transmission of events within the same *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 Transmitter* 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

1. <i>Event Processing Manager Configuration</i>	How to configure the <i>Event Processing Manager</i> to include a <i>Generic Event Transmitter</i> .
2. <i>Generic Event Transmitter Configuration</i>	How to configure the <i>Generic Event Transmitter</i> itself.

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 Transmitter* instance please add this line in the *Event Processing Manager* configuration file:

```
<!-- Element that's send the data on the network -->
<processing-element name="Event-Transmitter" config="Generic-Event-Transmitter/Default/conf/
transmitter.xml" data="Receiver[command]" />

<!-- Element that's receive event from the transmitter -->
<processing-element name="Receiver" config="Receiver/Default/conf/test.xml" />

<!-- Element creating events -->
<processing-element name="Creator" config="Creator/Default/conf/test.xml" data="Event-
Transmitter[data]" />
```


2

Generic Event Transmitter Configuration

There's two types of socket transmitters which are both bidirectional. The server transmitter will be the one that will spend more time receiving events than sending them. Once started, it will wait for a client transmitter connection. The client transmitter will be the one spending more time sending events than receiving some. It's mandatory to have a waiting server transmitter before the first sending of the client transmitter.

The following example shows a standard *Generic Event Transmitter* server configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration ... encoding="UTF-8" flush-period="500ms">

  <serializer separator="|">
    <class>com.watch4net.events.common.serialization.GenericEventSerializer</class>
    <field>
      <class>com.watch4net.FieldSerializerExample</class>
      <parameter name="param1">value</parameter>
    </field>
  </serializer>

  <deserializer separator="|">
    <class>com.watch4net.events.common.serialization.GenericEventDeserializer</class>
    <field>
      <class>com.watch4net.FieldDeserializerExample</class>
      <parameter name="param1">value</parameter>
    </field>
  </deserializer>

  <nioServerTransmitter>
    <port>22222</port>
  </nioServerTransmitter>
</configuration>
```

- configuration: The main element contains two optional attributes. The *encoding* attribute specifies which encoding will be used when sending and receiving. It's important that every connected transmitter share the same encoding. The default value is **UTF-8**. The *flush-period* attribute specifies the period at which the buffered output will be sent even if it is not full. The default value is **500ms**.
 - serializer: This will specify the configuration of the serializer that will transform the event when sending them to the receiver transmitter. The optional *separator* attribute specifies which character will be used to separate the events fields. The default value is **tab**.
 - * class: This is the class that is implementing `com.watch4net.events.serialization.EventSerializer`
 - * field: This unbound list specifies how to serialize the special fields in the events with its configuration. Please refer to the documentation of the desired serializer to see which field serializer can be used.
 - deserializer: This will specify the configuration of the deserializer that will transform the event when receiving them from another transmitter. The optional *separator* attribute specifies which character is used to separate the events fields. The default value is **tab**.
 - * class: This is the class that is implementing `com.watch4net.events.serialization.EventDeserializer`
 - * field: This unbound list will specify how to deserialize the special fields in the events with its configuration. Please refer to the documentation of the desired deserializer to see which field deserializer can be used.
 - nioServerTransmitter: This specifies that the event transmitter is a socket receiver and the *port* tag tells which port will be used to listen for new connections.

The configuration standard *Generic Event Transmitter* client is mostly the same except for the last tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration ... encoding="UTF-8" flush-period="500ms">

    ...

    <nioClientTransmitter>
        <port>22222</port>
        <host>serverHostNameOrIp</host>
    </nioClientTransmitter>
</configuration>
```

- `nioClientTransmitter`: This specifies that the event transmitter is a socket client and the *port* and *host* tags tell the location of the server transmitter.