



EMC® Watch4net®

Version 6.3u2

Generic Event Writer Admin Guide

P/N: pending

Rev : 40032

EMC²

Copyright ©2010 - 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 Writer</i> Configuration	3

Preface

Purpose

The *Generic Event Writer* is a *Processing Element* designed to write events on a socket. It handles event serialization and allows the incoming events to be forwarded to another *Processing Element* on a remote server.

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 Writer* 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 Writer</i> .
2. <i>Generic Event Writer Configuration</i>	How to configure the <i>Generic Event Writer</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 Writer* instance, please add this line in the *Event Processing Manager* configuration file:

```
<!-- Generic Event Writer -->
<processing-element name="Generic-Event-Writer" config="Generic-Event-Writer/Default/conf/
generic-event-writer.xml"/>

<!-- Some other element -->
<processing-element name="Some-Element-That-Sends-Data" config="Test/Default/conf/test.xml"
data="Generic-Event-Writer[data]" />
```

The *Generic Event Writer* can receive events on as many input streams as required. Events received on each input stream will be written the corresponding socket which is bound to the server and port specified in the configuration. In the example above, the *Processing Element* called "Some-Element-That-Sends-Data" will forward all its data events on the *Generic Event Writer*'s incoming data

stream.

Since the *Generic Event Writer* writes events on sockets, it has no output streams. Therefore, it cannot be configured to forward its data to other *Processing Elements* in this *Event Processing Manager*. However, as mentioned above, any number of input streams can be defined. Each input stream defined in the *Event Processing Manager* configuration must also be defined in the *Generic Event Writer* configuration (see section [2](#)).

2

Generic Event Writer Configuration

The following example shows a standard *Generic Event Writer* configuration with 2 input streams:

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

    <!-- Data received on the specified INPUT stream will be sent on the specified port. -->
    <data-stream>data</data-stream>
    <hostname>localhost</hostname>
    <port>22222</port>
    <serializer>
      <class>com.watch4net.events.common.serialization.GenericEventSerializer</class>
    </serializer>
  </sender>

  <!--
  <sender>
    <data-stream>command</data-stream>
```

```
<hostname>localhost</hostname>
<port>22223</port>
<serializer>
  <class>com.watch4net.events.common.serialization.GenericEventSerializer</class>
</serializer>
</sender>
-->
</configuration>
```

- *sender* Defines a new sender, which sends events on the socket. Every sender defined here corresponds to an input stream. The *encoding* attribute refers to the encoding used to serialize the events that are received. If omitted, the default value will be set to *UTF-8*. The *flush-period* parameter defines the interval at which the socket is flushed (i.e.: the data inside buffers is forcibly sent). If omitted, the default value (500ms) will be used.
 - *data-stream* Defines the unique name of the input stream. This name will be used in the *Event Processing Manager* configuration (see section 1).
 - *port* The port on which this sender will write incoming events.
 - *serializer* Defines the *Serializer* implementation that will be used to transform events into readable strings.
 - * *separator* The character that marks the end of an event and the beginning of a new one in the serialized form. This parameter is optional, and the default value (TAB) should be a good choice most of the time.
 - * *class* The java class that corresponds to the *Serializer* that will be used. This value should not be changed.

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

```
...
<serializer>

<class>com.watch4net.events.common.serialization.GenericEventSerializer</class>
>
...
```

```
<field type="OBJECT">
  <class>CustomFieldSerializer</class>
  <parameter name="param1">value1</parameter>
  <parameter name="param2">value2</parameter>
  ...
</serializer>
...
```

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