



EMC® Watch4net®

Version 6.3u2

Transaction Listener Administration 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
Introduction	iii
Terminology	iii
Document Organization	iv
Where to get help	v
1 System Prerequisites	1
2 Installation Overview	3
3 Configuration Details	5
1 Listening to Packets	5
2 Triggering Actions	6
2.1 Actions	7
2.2 Polling	7
2.3 Events	7
3 Client Sites	9
3.1 Thresholds	11
4 Applications	11
A Duration Notation	13
B Timeseries Values	15
C Event Properties	17
D System Properties	19
E Additional Files	21

Preface

Introduction

The Transaction Listener listens to packets from a router via a device's network adapter. It can be configured to process timeseries data, and create events.

It includes functionalities like defining metric thresholds for triggering events, and the ability to identify, and aggregate, metrics by client site.

Terminology

The Transaction Listener works similarly to any *APG* collector, in that it processes data that is output to timeseries raw data. It is configured in *APG* as if it is a collector, and fits into the overall pipeline like other collectors. The reason it is called a listener is that instead of actively gathering data it listens to traffic received on a network adapter.

The Transaction Listener can also output events via a Generic Event Writer into the event processing pipeline.

Although it is called a listener, other *APG* documentation that generically discusses collectors is applicable to the Transaction Listener, except, as mentioned, in so far as its method of assembling its source data.

Document Organization

1. System Prerequisites	Contains information about the module requirements.
2. Installation Overview	Gives a global overview of the <i>Transaction Listener</i> after its installation.
3. Configuration Details	Contains details about the configuration of the <i>Transaction Listener</i> .

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

System Prerequisites

This module is only supported on the following platforms:

- Linux x86 & x64
- Solaris 10 x64
- Windows x86 & x64

The listener requires access to a timeseries backend to send raw values at every polling period, and an event listener to send alerts and ICMP/SNMP authorization. As such, this is one of the few collecting components that requires root privileges, or at least raw socket access privileges.

WARNING: The module also strongly depends on the `libpcap` library, so please make sure it has been installed first. On windows, the library is called `winpcap`

See figure below for a simplified overview of the different inter-connections.

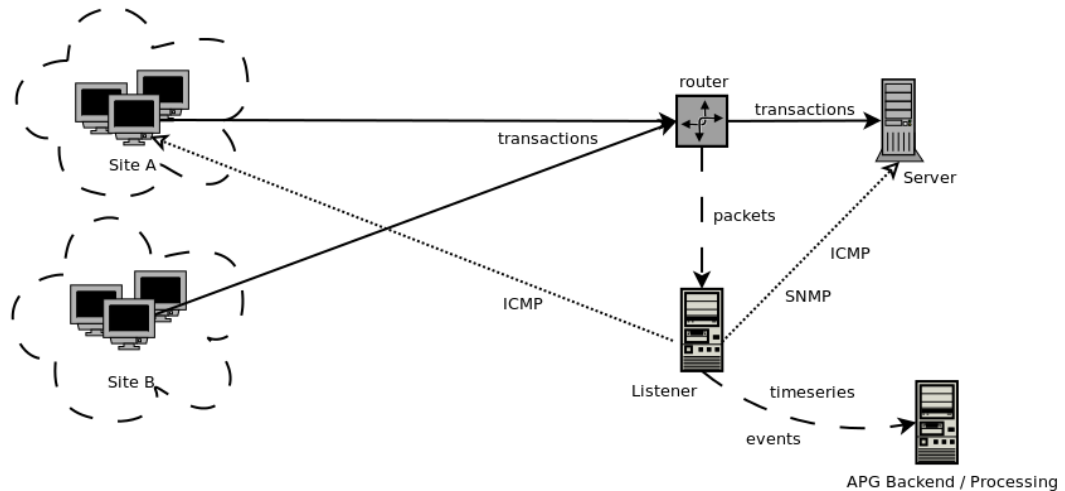


Figure 1: Overview

The listener works on both IPv4 and IPv6 networks, but configuration is a lot easier on IPv4 networks as subnet masks can be used. They can't with IPv6, so each client IP has to be manually specified.

2

Installation Overview

Once the installation is complete, the installation directory will contain the following entries.

- `conf` which contains the main configuration file
- `ddl` which contains SQL files used to create the related event schema
- `doc` which contains this documentation file
- `lib` which contains the Java binaries
- `misc` which contains additional configuration for the event listener, datastore writer and frontend mapping

See appendix [Additional Files](#) on page 21 for a list of additional configuration files and what part of the process they are used to configure.

3

Configuration Details

The configuration takes place in four parts: where to capture packets, which actions should be taken, what defines the client sites and what defines the applications to monitor. It's all located inside one single configuration file named `transaction-listener.xml`, located in the module's `conf` directory.

In order to register this Transaction Listener inside a Collector-Manager, you need to add the following line to its `collecting.xml` file:

```
<collector enabled="true" name="TransactionListener" next="Backend" type="TransactionListener"
config="Transaction-Listener/Default/conf/transaction-listener.xml" />
```

You can change `Backend` for any next component you'd like to use. Also update the transaction listener configuration path if necessary.

1 Listening to Packets

This defines to which network device the listener will attach to, and with which settings.

```
<listen iface="192.168.1.1" dump="logs/packets.pcap" />
```

The `iface` attribute is used to select which network adapter to bind to. This network device will be used to receive the transaction TCP packets from the router. This should contain the name of the requested interface. However, as network adapter names might be really complex on Windows systems, the selection can also use an IP address to find the corresponding network device. This attribute is optional. If not set, the first one found will be selected.

The `filter` attribute is optional and can be used to change the default capture filter (which can be visible in the corresponding `xsd`). The filter syntax is the same as the `tcpdump` tool.

The `dump` attribute can be used to specify a path to a file in which packets will be dumped. Use this for network debugging purposes only as there is no limitation on the file size.

2 Triggering Actions

Actions are specific processes that run when a set of conditions are met. They can be divided into two categories. The polling action is defined by the `<polling>` tag and regularly recovers the summarized timeseries data and sends it into the collecting process. The event-based actions are defined by the `<events>` tag and are triggered when the configured thresholds are reached. Thresholds are defined by site whose definition is covered in section [Client Sites](#) on page 9.

```
<actions transaction-idle-timeout="1m" transaction-idle-check-period="1m">
  <polling group="group" period="5m" />
  <events config="../../Event-Processing/Generic-Event-Writer/Default/conf/transaction-
event-writer.xml" iface="0.0.0.0">
    <ping timeout="10s" />
    <snmp community="read-community">
      <cpu idle=".1.3.6.1.4.1.2021.11.11" />
      <memory available=".1.3.6.1.4.1.2021.4.6" total=".1.3.6.1.4.1.2021.4.5" />
    </snmp>
  </events>
</actions>
```

2.1 Actions

```
<actions transaction-idle-timeout="1m" transaction-idle-check-period="1m">
  [...]
</action>
```

The attributes from the `actions` elements configure the way idle transactions are handled. Idle transactions are transactions for which packets have been received at some time, but then the communication went silent without being closed properly. By properly setting `transaction-idle-timeout`, you can configure how long a transaction will remain idle before being forcefully closed. See appendix [Duration Notation](#) on page 13 for an explanation of the notation used to define durations.

The `transaction-idle-check-period` attribute contains the period at which the idle transaction status will be checked. It is a good practice to have it lower or equal to the timeout parameter `transaction-idle-timeout`.

2.2 Polling

```
<polling group="group" period="5m" />
```

The `polling` element defines the way captured data will be sent to the timeseries database. Two attributes are needed here: the `group` which is the name of the backend group inside which data should be pushed (see the Backend Administration Guide for more details on this), and the `period` on which to push the collected data. Packets are captured and analyzed in real-time, but aggregated data is stored in memory and only sent at defined regular intervals.

See appendix [Timeseries Values](#) on page 15 for a detailed overview of the timeseries values and their properties.

2.3 Events

```
<events config="../../Event-Processing/Generic-Event-Writer/Default/conf/transaction-event-
writer.xml" iface="0.0.0.0">
  [...]
```

```
</event>
```

Events are sent when one (or more) metrics of a specific transaction are higher than the corresponding specified threshold. Those thresholds are a site-related notion as different sites are very likely to have different connection performance.

The attribute `config` is mandatory and should point to the installed transaction writer that will be instantiated, configured and used to send those events. The path can be relative to the *Transaction Listener* installation directory.

`iface` contains the IP address of the interface to bind to, when applicable. For instance, this proves particularly useful when ICMP and SNMP requests need to be sent through a specific interface due to access control list restrictions. This setting allows the selection of a specific outgoing network interface when sending the ICMP and SNMP commands. This attribute can be removed to let the Transaction Listener select a default one.

Event Writer

```
"../../../../Event-Processing/Generic-Event-Writer/Default/conf/transaction-event-writer.xml"
```

This module (*Generic Event Writer*) is a direct dependency of the *Transaction Listener*. So it needs to be installed first. It will be used independently and as such does not require an *Event Processing Manager*. Its configuration however needs to be edited so that it sends data to the right receiver (usually on a different server).

See appendix [Event Properties](#) on page 17 for a detailed overview of the events that are sent along with their properties.

Ping The values of these requests will be recovered from both the client and the server, and then included inside every event sent so as to provide additional information on both ends state.

```
<ping timeout="10s" />
```

ICMP requests are used to collect the client and server latencies. The `timeout` attribute is used to know when to give up on a device that does not respond. See appendix [Duration Notation](#) on page 13

for an explanation of the notation used to define durations. A specific interface can also be specified for binding purposes through the `iface` attribute.

SNMP The values of the configured OIDs will be recovered from the server and included inside every event sent so as to provide additional information on the server state.

```
<snmp community="read-community">
  <cpu idle=".1.3.6.1.4.1.2021.11.11" />
  <memory available=".1.3.6.1.4.1.2021.4.6" total=".1.3.6.1.4.1.2021.4.5" />
</snmp>
```

This holds the SNMP settings, and more particularly the common SNMP attributes: `port` (defaults to 161), `version` (defaults to v2c), `protocol` (defaults to UDP), `community` string (also used for username when using SNMPv3), `priv-proto` (v3 privacy protocol), `priv-pass` (v3 privacy password), `auth-proto` (v3 authentication protocol), `auth-pass` (v3 authentication password), number of retries, and response timeout. A specific interface can also be specified for binding purposes through the `iface` attribute.

In this version, CPU and Memory recovery only support single OIDs. Two sets of information will be recovered: CPU (from `idle` CPU OID) and Memory (from `available` and `total` memory). This list cannot be changed at this time.

Also, please note that this version only supports one single set of settings (SNMP community or v3 authentication that is).

3 Client Sites

```
<site name="Site_A">
  <host ip="192.168.3.1" />
  <host ip="192.168.3.2" />
  <network net="10.128.200.0" mask="255.255.255.0" />
  <net cidr="192.168.4/24" />
  <thresholds>
    <on metric="ServerResponseTime" value="200" />
    <on metric="NetworkRoundtripTime" value="100" />
    <on metric="RetransmissionDelay" value="100" />
  </thresholds>
```

```
</site>
```

Sites are groups of IPs or networks that will be identified as belonging to a site of clients with a particular name. Computed metrics will then be aggregated by client site and application, resulting in average indicators for the connections of a whole site to a given application. Three notations are allowed to define a subset of clients.

It is important to note that the order in which sites are defined has a role to play in the process of determining to which site a given IP belongs to.

The list is read in the given order and then the IPs are checked against the sites in precisely the same order. The first site to match will be the selected one. As a consequence, it is possible to have larger networks at the bottom of the list in order to select some very specific IPs of a network in the first sites, and then catch all the others in sites lower in the list.

A site can have any number of `host`, `network`, and `net` elements.

Host defines a single host name or IP

```
<host ip="192.168.3.1" />
```

Network defines a subnet based on the net IP and the associated mask

```
<network net="10.128.200.0" mask="255.255.255.0" />
```

Net defines a subnet based on a CIDR notation

```
<net cidr="192.168.4/24" />
```

WARNING: You cannot specify masks or CIDR notations on IPv6 networks.

3.1 Thresholds

For each site, a set of thresholds can be defined in order to check some specific metric against precise values. These thresholds are optional though, so if they are not needed, you can skip their definition.

The goal of these thresholds is to trigger events as soon as a value exceeds (strictly greater) the specified threshold. These events are sent as configured in section [Events](#) on page 7.

```
<thresholds>
  <on metric="ServerResponseTime" value="200" />
  <on metric="NetworkRoundtripTime" value="100" />
  <on metric="RetransmissionDelay" value="100" />
</thresholds>
```

The definition of thresholds is quite simple: you need the metric to check and the value against which to check it. All possible values are described in the [Timeseries Values](#) appendix on page 15. Values have to be given in the corresponding unit. There can be from one to all metrics defined here.

In the provided example, an event will be created and sent if *ServerResponseTime* > 200 *OR* *NetworkRoundtripTime* > 100 *OR* *RetransmissionDelay* > 100.

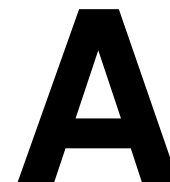
4 Applications

```
<application name="APG">
  <host ip="192.168.2.103" />
  <port value="80" />
  <port-range from="58080" to="58090" />
</application>
```

Applications are the end point of the monitored transactions. Their IPs can be configured using the same elements as the ones in a site, which are `host`, `network`, and `net`. The two additional elements available are `port` which allows the definition of a single target port and `port-range` when monitoring is to be performed on multiple ports.

Also, the application can be given here through the `name` attribute, but it is optional. When not set, a lookup will be performed inside the `etc/services` file of the operating system in order to apply those translations to the port itself and return this as application name. The collectors knows most of the

default paths to this file, but just in case the file is not located at a standard location, this path can be specified through a property. See appendix [System Properties](#) on page 19.



Duration Notation

This specific notation allows the definition of duration in a more readable way than the usual *number of seconds*. Using this, you can give values and units at the same time, thus reducing the amount of computation required to configure and read any duration.

For instance, in order to define a duration of 2 minutes and 30 seconds, you simply need to put `2m30s` instead of `150`. Note that setting `150s` would also have the same effect. However, only integer values are accepted, so you can't set `2.5m`.

Valid units are:

- `d` for days
- `h` for hours
- `m` for minutes
- `s` for seconds
- `ms` for milliseconds, where applicable (see the configuration XSD for that)



Timeseries Values

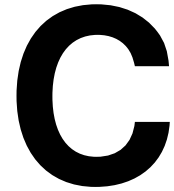
The following values are computed and sent by the *Transaction Listener*:

- `ServerConnectionSetupTime` which is the time it took for the connection to establish on the server side, in milliseconds
- `ClientConnectionSetupTime` which is the time it took for the connection to establish on the client side, in milliseconds
- `ServerResponseTime` which is the time it took for the server to send data after a request, in milliseconds
- `NetworkRoundtripTime` which is the time it took for the data to be sent to the client and acknowledged, in milliseconds
- `RetransmissionDelay` which is the amount of time lost due to data retransmissions, in milliseconds
- `RequestDuration` which is the complete duration of a request, in milliseconds
- `RequestDataSent` which is the amount of data sent by the server, in octets
- `RequestCount` which is the number of captured requests

- `ClientIPCount` which is the number of captured client IPs
- `TransactionCount` which is the number of captured transactions

The following properties are automatically assigned to every computed metric:

- `source` static value 'Transaction Listener'
- `devtype` static value 'Host'
- `device` the IP address of the server
- `port` the targeted port of the server
- `site` the name of the client site
- `parttype` static value 'Application'
- `part` the name of the application if defined or if found in services
- `name` the name of the metric
- `unit` the unit of the metric

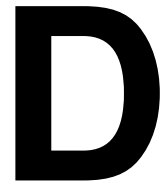


Event Properties

The following properties are set on every event sent by the *Transaction Listener*:

- `Id` an automatically generated unique event ID
- `Name` the event name - static value 'TransactionThresholdReached'
- `TransactionOpenedAt` the transaction start time, in seconds
- `TransactionClosedAt` the transaction end time, in seconds
- `ClientAddress` the client IP address
- `ClientPort` the client port
- `Site` the name of the site the client belongs to
- `ServerAddress` the server IP address
- `ServerPort` the port on the server
- `Service` the name of the service - could be null if not defined or not found
- `Metric` the name of the metric that reached its threshold
- `Message` a quick description of what happened

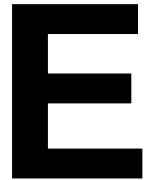
- `ClientLatency` client latency - PING
- `ServerLatency` server latency - PING
- `ServerCpuUtilization` server CPU utilization - SNMP
- `ServerMemoryUtilization` server memory utilization - SNMP



System Properties

These properties are used to configure some behaviors of the *Transaction Listener*. They need to be set as system properties for the JVM that runs the Transaction Listener. It can usually be performed by adding an entry inside the `Collector-Manager module.properties` file.

- `etc.services.file.path` defines the path (including filename) that points to the services file used to look up port translations



Additional Files

Some additional configuration files are provided with the Transaction Listener. They are not required by the module itself, but by other components of the collecting process, or by the rendering engine.

- `misc/transaction-event-writer.xml` use this file to configure the associated event writer (see Generic-Event-Writer module) that will send events to the APG event processing module
- `misc/transaction-event-listener.xml` use this file to configure the remote event listener (see Generic-Event-Listener module) (part of the APG event processing module) that will receive events sent by this module
- `ddl/transaction-events-datastore.sql` modify and execute this file in order to create the required events schema in the database; you may not need the two first lines of this file if the `events` database already exists
- `misc/transaction-datastore-writer.xml` use this file as configuration for a Datastore event writer (see Datastore-Writer module)
- `misc/transaction-mapping.xml` this is the mapping file used by the frontend to access transaction events data and display it in reports (see Frontend Property Mapping documentation)