



EMC® Watch4net®

Version 6.3u2

Text Collector Administration Guide

P/N: pending

Rev : 43463

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
Document Organization	iv
Where to get help	v
1 Product Overview	1
2 Architectural Overview	3
1 Parsing Engine Architecture	6
1.1 Command Executor	7
1.2 Text Parser	8
1.3 Data Formatter	8
1.4 Data Handler	8
3 Installation	9
1 APG 4.3 and earlier	9
2 APG 5.0 and later	10
4 Configuration	11
1 For APG 4.2 to 4.3	11
2 For APG 5.0 and later	12
2.1 Adapter configuration (textoutputcollector.xml)	13
2.2 Configuration example	16
2.3 Parser configuration	17
5 Polling Windows Hosts	19
1 Windows Platform Support and Requirements	20
1.1 APG Windows Servers	20
1.2 Polled Windows Hosts	20

- 1.3 Other Requirements 20
 - 1.4 Allowing Communication between the APG Server and the Remote Windows Hosts 20
 - 1.5 Enabling WinRM on the Server and Remote Polled Hosts 21
 - 1.6 Configuring the text-parsing-configuration.xml file 21
- 6 Start Up 23**
- 7 Troubleshooting 25**
 - 1 The adapter won't start 25
 - 2 The adapter started successfully, but no data is showing up in APG 26
 - 3 The collector ran successfully but there's no data in APG 26
- 8 Appendix A: Timestamps 27**
 - 1 Definitions 28
 - 2 Examples 28

Preface

Purpose

The purpose of this guide is to provide information about the *APG* Text Collector architecture and how to use it.

Intended Audience

This guide is intended to be read by any user of the *APG* Text Collector.

Document Organization

1. Product Overview	This section gives an overview of the product.
2. Architectural Overview	This section describes the architecture of the product.
3. Installation	This section describes how to install the product.
4. Configuration	This section describes how to configure the product.
4. Polling Windows Hosts	This section explains specific requirements and configuration steps for polling Windows hosts.
6. Start Up	This section describes how to start the product.
6. Troubleshooting	This section includes some troubleshooting procedures.

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.

Product Overview

The Text Collector is a generic module for APG that allows data to be imported from various text-based sources (command line output, text file, etc.) and pushed into APG. The Text Collector relies on a text parsing library (property of Watch4Net Solutions) that can be configured to execute the parsing task in a specific way. Therefore, this library must be installed in order for the Text Collector to function properly. Here are the requirements:

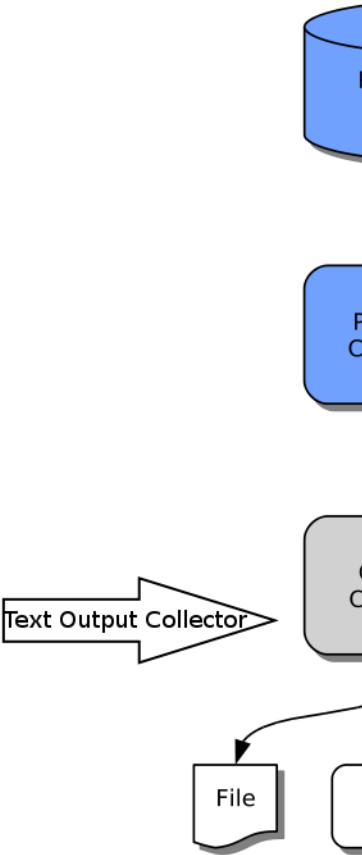
- APG 4.2 or higher

The Text Collector can also be used to poll remote Windows hosts for data to be parsed. There are other requirements in this case. See [Polling Windows Hosts](#) on page 19 for more information.

2

Architectural Overview

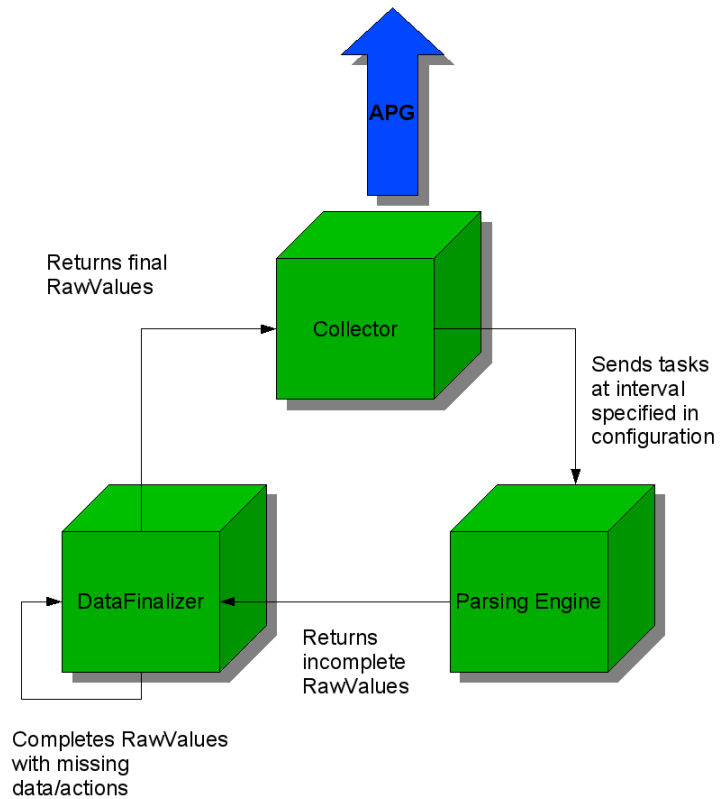
The Text Collector is built to integrate seamlessly into the APG structure. The following diagram il-



lustrates APG’s global architecture and how collectors can be attached to it.

The Text Collector is composed of three main modules: the collector itself, the parsing engine,

which is provided by the output parsing library, and the data finalizer. The figure below illustrates how these modules work together to collect data from text sources.



Here is a quick overview of every module's main responsibilities:

Collector Module

- Loads configuration
- Creates a parsing engine for every *different* parsing job that will be executed
- Sends a new parsing task to the parsing engine at a fixed interval
- Receives final data from the data finalizer
- Forwards data to APG

Parsing Engine

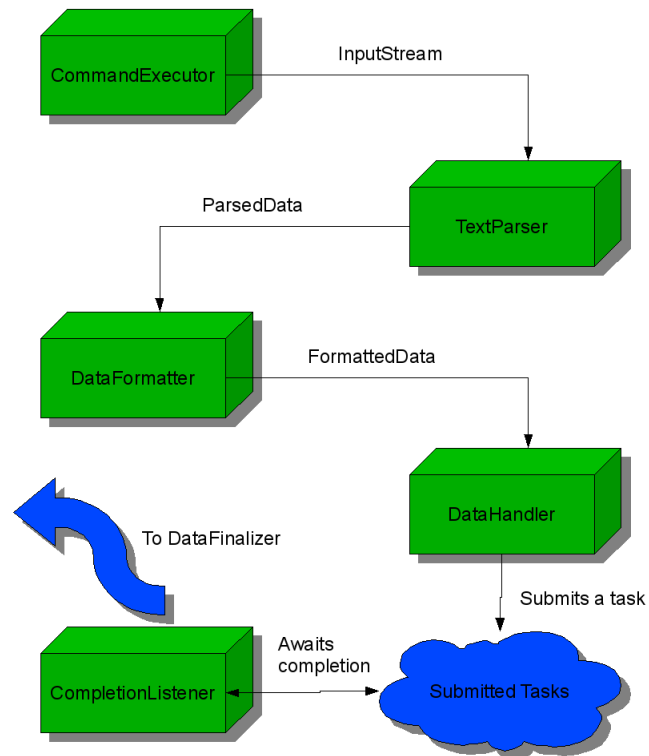
- Handles parsing tasks sent by the collector
- Parses the specified output according to its configuration
- Formats the parsed data into APG values
- Sends APG values to the data finalizer

Data Finalizer

- Checks if properties need to be refreshed
- Forces a refresh if needed
- Sends the APG values to the collector

1 Parsing Engine Architecture

While the collector and data finalizer modules are fairly simple, the parsing engine has a more complex architecture of its own. The following diagram illustrates the way its sub-modules operate:



1.1 Command Executor

Fetches the text-based source and passes it to the parser. Currently, data can be obtained from:

- A text file (local or remote, via FTP)
- A command-line output (local or remote, via SSH)

- A Windows computer via Windows Remote Management. For more information, see [Polling Windows Hosts](#) on page 19.

1.2 Text Parser

- Parses the data obtained by the command executor according to its configuration and puts it in a common data structure.
- Passes the parsed data to the data formatter.

1.3 Data Formatter

To this date, the only available formatter is the

```
com.watch4net.apg.v2.collector.plugins.textoutputcollector.formatting.RawValueFormatter
```

- Formats the data structure into APG values
- Passes the APG values to the data handler

1.4 Data Handler

To this date, the only available handler is the

```
com.watch4net.apg.v2.collector.plugins.textoutputcollector.handling.DeltaRateRawValueHandler
```

- Performs mathematical operations on the collected values (delta, rate, etc)
- Creates a new finalizing task on the data finalizer stack

Installation

1 APG 4.3 and earlier

On 4.3 and earlier, some libraries must be updated in order for the Text Collector to work correctly. The following files must be copied to APG-Backend/lib/ under the *APG* installation directory:

- backend-collector-dca-textoutput.jar (overwrite the previous version)
- output-parsing-library.jar (revision 720 or higher)

Some other third parties libraries could be missing in your version installation. Check with Watch4Net support to see if your third parties libraries must be updated. The following file must be copied to APG-Backend/conf/ under the *APG* installation directory:

- textoutputcollector.xsd (overwrite the previous version if present)
- parserconfiguration.xsd (overwrite the previous version if present)
- collectingconfiguration.xsd (overwrite the previous version if present)
- commanddefinition.xsd (overwrite the previous version if present)

- hostgroupconfig.xsd (overwrite the previous version if present)

IMPORTANT: Make sure that the apg user has permission to access the newly copied files.

2 APG 5.0 and later

An up-to-date version of the Text Collector and all its dependencies is already included with these releases, but not installed by default. Refer to the module manager documentation in order to install the text-collector.

4

Configuration

This section describes the different configuration files for the adapter.

1 For APG 4.2 to 4.3

All the configuration files are located in the APG-Backend/conf directory under the *APG* installation directory. In case some of these files do not exist, they must be created in this folder as well. The Text Collector for *APG* must be added in the main collecting configuration file (collecting.xml) in the APG-Backend/conf directory under the *APG* installation directory.

```
<collectors>
...
  <collector enabled="true" name="TextCollector" next="Cache">
    <classname>com.watch4net.apg.v2.collector.plugins.TextOutputCollector</classname>
    <config-file>conf/textcollector.xml</config-file>
  </collector>
...
</collectors>
```

2 For APG 5.0 and later

The Text Collector for *APG* must be added in the main collecting configuration file (collecting.xml) in the Collecting/Collector-Manager/<Instance>/conf directory under the *APG* installation directory.

```
<collectors>
...
  <collector name="TextCollector" enabled="true" next="Backend" conf="Text-Collector/Default/
conf/textcollector.xml" />
...
</collectors>
```

All the other configuration files for the collector are located in Collecting/Text-Collector/Default/conf directory under the *APG* installation directory.

2.1 Adapter configuration (textoutputcollector.xml)

This file is the main configuration file for the adapter. It contains information about how data will be interpreted, as well as links to other configuration files for specific modules. The hierarchy illustrates the global structure of the configuration file:

```
<configuration>The main container for the collector configuration. Every other tag must be
nested inside this one.

<simultaneous-collecting>The number of simultaneous requests that will be sent to the device.
<polling-interval>The time, in seconds, between each polling.
<refresh>Property refresh periodicity. Forces property refresh every 't' seconds if set to a
positive value. Do not use if properties are to be refreshed when their values change (default
behavior).
<collecting-group>The group in which the variables will be collected. This group must be
defined in config.xml, or values will be ignored.

<source>The value of the source property, as it will appear in APG. The source property is used
to identify the collector that pushed the given metric in APG.

<collecting-configuration name="name" timeout="timeout">This tag acts as a container for
configuration options that relate to the same parsing operation (i.e.: parsing of a given
command or text file). The mandatory parameter name must be unique and is used to identify all
variables that were obtained by the same request. The optional parameter timeout is the maximum
time to wait (in seconds) for the parsing operation to be completed. The default value is 120
seconds. There is usually one collecting configuration per different file or command to parse.

<parsing-configuration-file>The location of the configuration file that configures the parsing
options.
<raw-value-group-list master-node="master-node" delete-after-group="true" delete-parent-after-
group="false" variable-id="source_device_module_part_name">This tag acts as a container for the
group of values that comes from the same master-node. It's recommended to activate the
deletion after the group is done for a substantial performance improvement. The variable-id
attribute is optional (default value listed above) and is used to specify which properties will
be used to create the variable ID of each raw values.

<timestamp-id format="format">The id that represents the timestamp in the source data. This
parameter is optional, and current system time will be used if it is omitted. If the field
represented by the ID isn't in a numeric format, the format attribute must be used with the
Java SimpleDateFormat style.

<property-list name="name" use-leaf-name="true_or_false">This tag is the container of an APG
```

property. The name field will be the one appearing in APG.

`<hardcoded>`The hardcoded property.

OR

`<position>`The position of the property in the tree under the master node. An execution property can be referred with @propertyName

`<matching-property name="name">`This optional field can be used in two different ways. If the property is hardcoded, it will ensure that the property will be set only if the previously set property defined in "name" is matching the value in the tag. For properties with position, if the property is not in the scope of the master node and if the position can apply to multiple other nodes, this tag will ensure that the right node will be retrieved. The name attribute refer to an above property. The value is the position of the field to compare under the previous position path.

`<value-list name="name" leaf="leaf" unit="unit" type="type">`This tag will contain the parameters of each value. The name field is the name of the desired value in APG while leaf is the position of the leaf in the tree under the master node and unit is the unit of the value if available. The type attribute is optional. It can take one of the following values: delta or rate. If it is not specified, the metric will be pushed as is, without any transformation. If it is set to delta, the metric that will be pushed is the difference between the current value and the previous value. If it is set to rate, the metric that will be pushed is the difference between the current value and the previous value, divided by the difference between the current timestamp and the previous timestamp.

`<replace value="value" by="by">` This tag will change an invalid hardcoded value for a valid one if the value in the text isn't in a numeric format or for any other replacement needs.

Note: Remember that the position for properties and leaves are interpreted as regexes for more conveniences.

Substitutions are helpful when a polled property or metric needs to be replaced before it is pushed into APG. They can be nested inside value or property tags. They are often used when a value appears as text in the collected data. If the data is left unchanged, APG will not be able to interpret the value, which needs to be a number. Substitutions will allow the administrator to configure the adapter so that when a particular string is collected as the value for a given metric, the string will be converted to a number. In the following example, a substitution is used for the *WriteCacheEnabled* metric. If the collected value is *ENABLED*, the value for the metric will be set to 1. If the collected value is *DISABLED*, the value will be set to 0.

```
<value-list name="WriteCacheEnabled" leaf="Write_cache" unit="">
  <replace value="ENABLED" by="1"/>
  <replace value="DISABLED" by="0"/>
</value-list>
```

When working with properties, it's possible to extract part of the leaf name or value depending on the *use-leaf-name* attribute. For that, a substitution providing a regex is needed with the desired group defined in the *by* attribute with \$ as prefix.

```
<property-list name="device" use-leaf-name="true" >
  <position>./result</position>
  <replace value="(\w+)\_current\_value" by="$1"/>
</property-list>
```

2.2 Configuration example

Let's create *APG* values with the following input given by our parser configuration:

```
10.128.25.69
-IFCONFIG_0
  -Name: eth0
  -Mac address: 00:0c:29:f8:79:6e
  -IP address: 10.128.2.38
  -RX bytes: 1895232
  -TX bytes: 2333651
-IFCONFIG_1
  -Name: lo
  -Mac address: N/A
  -IP address: 127.0.0.1
  -RX bytes: 34410
  -TX bytes: 34410
```

We should be able to create valid values with the following collecting configuration:

```
<collecting-configuration name="IfConfig">
  <parsing-configuration-file>./conf/IfConfigParsing.xml</parsing-configuration-file>
  <raw-value-group-list master-node="IFCONFIG_[\d]+" delete-after-group="true" delete-parent-
after-group="false">
    <property-list name="devtype"> <hardcoded>host</hardcoded></property-list>
    <property-list name="parttype"> <hardcoded>interface</hardcoded></property-list>
    <property-list name="device"> <position>/</position></property-list>
    <property-list name="part"> <position>Name</position></property-list>
    <property-list name="ip"> <position>IP address</position></property-list>
    <property-list name="mac"> <position>Mac address</position></property-list>

    <value-list name="ifInOctets" type="delta" leaf="RX_bytes" unit="bytes"/>
    <value-list name="ifOutOctets" type="delta" leaf="TX_bytes" unit="bytes"/>
  </raw-value-group-list>
</collecting-configuration>
```

The raw value group is searching for each node starting with IFCONFIG_ and followed by a number. Remember that each position given inside this tag will be relative to the position of the node IFCONFIG_[\d]+. The devtype and parttype are always the same and hardcoded to the host and interface. Since the device is the address of the polled host, the position can simply be referring to the root node (/). The next properties ip and mac can easily be found under the master node at IP address and Mac address. Finally, the 2 values generated will be ifInOctets and ifOutOctets with the leaves RX bytes

and TX bytes. Since there's 2 master nodes in the given input, this configuration will output 4 different values for *APG*.

2.3 Parser configuration

Refer to the [APG-Text-Parsing-Library.pdf](#) documentation for a full explanation of the parsing options and configuration.

5

Polling Windows Hosts

You can use the Text Collector to interface with and poll metric data from Windows hosts.

In these cases, both the APG server and Windows hosts being polled must have Windows Remote Management (WinRM) installed.

Windows Remote Shell (WinRS), is used by the server to be able to poll the remote Windows hosts.

Further, PowerShell commands are invoked on the remote hosts to gather the actual metrics that will be parsed by the Text Collector so PowerShell must be installed on both the server and remote Windows hosts.

Please be aware that while with the Text Collector in general you can use a local or remote command executor, since specifically when polling Windows hosts you are actually using WinRM and WinRS for the server-client communication, you can only use a local command executor.

1 Windows Platform Support and Requirements

The following are the platform and software requirements for polling Windows hosts with APG.

1.1 APG Windows Servers

The following are prerequisites for APG servers running the Text Collector to poll Windows hosts:

- APG must be installed on a server running Windows Server 2008, or Windows Server 2003.
- WinRM, WinRS, and PowerShell must be installed.
These components are installed in Windows Server 2008 by default. On Windows Server 2003, you must manually install the necessary components.
- Because the `winrs` command you will invoke to communicate via PowerShell is a *privileged* command, the APG collection process in this case must be run with a user that has Administrative privileges. You can do this by running the Collector-Manager with a user that has administrative privileges. These user rights are then propagated to the Text Collector and the `winrs` command, as explained below in [Configuring the text-parsing-configuration.xml file](#).

1.2 Polled Windows Hosts

You can poll Windows hosts that can run WinRM, and must install it if it is not installed by default. Similarly, because it is recommended to use PowerShell to gather the collected metrics, PowerShell must also be installed on the Windows host being polled.

Note: You can use commands on other tools to gather metrics besides PowerShell, but these are not covered in this document, nor were they tested by Watch4net.

1.3 Other Requirements

Commands invoked by WinRS cannot prompt the user for any input.

1.4 Allowing Communication between the APG Server and the Remote Windows Hosts

On the APG server, run the following command from PowerShell:

```
Set-Item wsman:\localhost\client\trustedhosts -value <trusted hosts>
```

This allows the Text Collector to connect to all hosts on the network, and run `winrs`. For the syntax of how to refer to trusted hosts, consult the relevant Windows PowerShell documentation. You can now verify that all necessary hosts are reachable by running the following command from PowerShell:

```
winrm get winrm/config/client
```

This returns a list of hosts that can be polled by the Text Collector.

1.5 Enabling WinRM on the Server and Remote Polled Hosts

WinRM must be enabled on both the Server and any remote Windows computer that is being polled, even if it has been installed by default. Consult the appropriate Windows documentation on how to enable WinRM.

Note: WinRM in many cases will already be enabled as it is used for many other typical remote management purposes.

1.6 Configuring the text-parsing-configuration.xml file

You must add a custom command which invokes `winrs` in the `text-parsing-configuration.xml` file. An example is as follows:

```
<command type="custom">winrs -r:10.100.5.1 -u:administrator -p:passxyz powershell "Get-WmiObject -Class Win32_OperatingSystem -Namespace root/cimv2 -ComputerName .|.Format-List FreePhysicalMemory"</command>
```

Notice the following about the above command tag:

- You must use the `winrs` command followed by the remote host location, and login credentials.
- A PowerShell command is then to be executed on the remote computer.
- In this case, a `get` command is used to fetch the `FreePhysicalMemory` metric using the `Win32_OperatingSystem` class. Multiple metrics can be fetched for the class by separating them with commas.

To poll multiple hosts, include multiple instances of this tag configured for each host.

6

Start Up

When the collector is installed and configured, APG is ready to collect new data from the desired source. The following steps can be taken:

1. Start the APG collecting service.
2. Look at the collecting logs to make sure that everything is starting up as it should.

Troubleshooting

The first things to do while troubleshooting a configuration is to run the parsing-test script that will only test the parsing of the input text and output the resulting tree to see if the formatting or parsing configuration is good. The script is located in the bin folder of the Text-Collector instance and use 2 parameters:

- Argument 1: (Mandatory) The output parsing configuration file
- Argument 2: (Optional) A file containing an example of the data to parse

1 The adapter won't start

- Make sure that the contents of the root APG folder is accessible for the apg user, especially the files that were copied for the installation of the adapter (see section).
- Look at the collecting logs that are located either in APG-Backend/logs/ for *APG* 4.3 and earlier and in Collecting/Collector-Manager/Default/logs for *APG* 5.0 and later. These logs contain helpful messages that might help solving the problem.
- If you are using the defined commands feature, be sure to have the parameter command-definition-directory in the parser-configuration telling where the OS commands definition are

located and be sure to have a commands definition file per different OS (Ex: Linux.xml, Solaris.xml, etc.) based on the schema commanddefinition.xsd.

2 The adapter started successfully, but no data is showing up in APG

If the collector was just started, wait after a few collecting cycles have passed. Data usually takes some time to show up in APG for the first time. If there is still no data, check the following:

- In *collecting.xml*, make sure that the Text Collector is configured and that it is enabled.
- In *collecting.xml*, check the *next* attribute for the Text Outout Collector. If the value for the *next* attribute is a filter, make sure that the corresponding filter has its *enabled* attribute set to either *true* or *skip*.
- If the data source is on a remote server, make sure that the APG server is able to access it. If a connectivity issue occurs, a trace will be available in the collecting logs.

3 The collector ran successfully but there's no data in APG

If the collector is parsing a file, make sure that the file is in the same format as the collector host (Unix or DOS). Otherwise, put the logging to FINEST in the conf/collecting.logging.properties in the collector manager directory before running the collector again. There should be an entry like this one in the logs/collecting-0-0.log in the backend directory:

```
PARSER NAME parsed X elements
PARSER NAME formatted Y elements
```

1. If X is greater than 0, it means that the parser successfully did the job of parsing the entry from the configuration file depending on the parserconfiguration.xsd schema. If not, check the configuration file again.
2. When X is greater than 0 and Y equal 0, it means that the formatter couldn't create raw values from the parser output with the given configuration based on the textoutputcollector.xsd schema, check the configuration file again.
3. If Y is greater than 0 but there's still no data, it's possible that the timestamp of the created raw values aren't valid (-1). Check the timestamp-id tag of the formatter configuration.

Appendix A: Timestamps

1 Definitions

The format string can be composed of any of the following symbols:

Letter	Date/Time Component	Example
G	Era designator	AD; BC
y	Year	1998; 98
M	Month in year	January; Jan; 01
w	Week in year	38
W	Week in month	3
D	Day in year	352
d	Day in month	4
F	Day of week in month	2
E	Day in week	Friday; Fri
a	Am/Pm marker	AM; PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in Am/Pm (0-11)	0
h	Hour in Am/Pm (1-12)	12
m	Minute in hour	34
s	Second in minute	42
S	Millisecond	987
z	Time zone	PST; GMT-08:00
Z	Time zone	-0800

The number of times a letter is repeated determines the way data will be interpreted. A letter that is repeated four times or more signifies that the full representation is used, while fewer than four repetitions usually result in an abbreviation.

2 Examples

Timestamp in data	Format used
Monday 27 Oct, 2008 3:30 PM	EEEE d MMM, yyyy h:m a
2008/27/10 15:30	yyyy/d/M H:m
27-10-08 AD, 15:30:54 GMT+04:00	d-M-y G, H:m:s zzzz