



EMC® Watch4net®

Version 6.3u2

Remote Shell Collector Admin Guide

P/N: pending

Rev : 42665

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
Document Organization	iii
Where to get help	v
1 Product Overview	1
2 Installation	3
3 Configuration	5
1 Configuration files	5
2 Configuring the remote-shell-collector.xml file	6
3 Configuring the hosts-definition.xml file	7
3.1 Hosts section	8
3.2 Groups section	9
4 Configuring the scripts-definition.xml file	10
4.1 Referencing script files	10
5 Configuring the poller-definition.xml file	11
5.1 Tags and attributes	12
4 Scripts	15
1 Header	16
2 Metric format	17
5 Testing Scripts	19
6 Polling Windows Hosts	21
1 Windows Platform Support and Requirements	21
1.1 APG Windows Servers	22

1.2 Other Requirements	22
1.3 Allowing Communication between the APG Server and the Remote Windows Hosts	22
1.4 Enabling WinRM on the Server and Remote Polled Hosts	22
7 Glossary	25

Preface

Purpose

This guide describes the configuration and usage of the Remote Shell Collector.

Intended Audience

This guide is meant to be read by any person who will configure and/or use the Remote Shell Collector.

Document Organization

1. Product Overview	This section quickly describes the product
3. Configuration	This section describes how to configure the collector

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

Product Overview

The Remote Shell Collector can run scripts on remote hosts to collect performance data from the timeseries pipeline.

In the Remote Shell Collector, you configure one or more hosts to poll, and can easily group hosts so that you can run common scripts on them.

You can compose scripts, organized by operating system, to return the desired data, each line of which contains a timestamp, value, and related properties. You can test the scripts before you deploy them, and create mappings between scripts and hosts or host groups, so that scripts can be run on the hosts.

2

Installation

The Remote Shell Collector module ships with *APG* (since *APG* 6.0) but is not installed by default. You can install it via the Module Manager or by using the Centralized Management. The Remote Shell Collector module name is `Remote-Shell-Collector`.

In order to activate the Remote Shell Collector, you must edit the `collecting.xml` configuration file to add the collector declaration.

```
<collectors>
...
  <collector enabled="true" name="RemoteShellCollector" next="Backend" config="Remote-
Shell-Collector/Default/conf/remote-shell-collector.xml" />
...
</collectors>
```

Then, you can edit the collector configuration file in the `conf` folder of the Remote Shell Collector module. The default file name for this configuration file is `remote-shell-collector.xml`.

3

Configuration

1 Configuration files

There are four main configuration files that you must configure to use the Remote Shell Collector, in addition to any scripts that you have to compose. They are available in the directory `<APG>/Collecting/Remote-Shell-Collector/<Instance>/conf`.

File name	Description
remote-shell-collector.xml	Points to the other configuration files and the scripts directory, and contains collector-specific parameters. This is the file that is pointed to from the <i>collecting.xml</i> when referencing the Remote Shell Collector in the collecting pipeline
hosts-definition.xml	Defines the hosts and host groups that scripts will be executed on to collect performance data
scripts-definition.xml	Defines all the scripts to be run on the hosts defined in the <i>hosts-definition.xml</i> file. The scripts filenames are included as references, and are taken from the scripts directory defined in the <i>remote-shell-collector.xml</i> configuration file
poller-definition.xml	Maps hosts and host groups defined in <i>hosts-definition.xml</i> to scripts listed in <i>script-definition.xml</i> , with their polling frequency. This enables you to easily assign scripts to be run on hosts, collect metrics at each polling interval and create logical polling groups

2 Configuring the remote-shell-collector.xml file

This is a description of the XML tags in the *remote-shell-collector.xml* file. This is an example configuration which correspond to the module defaults.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/Remote-Shell-Collector"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Remote-Shell-Collector_remote-shell-collector.
xsd">
  <poller-definition-file>conf/poller-definition.xml</poller-definition-file>
  <scripts-defintion-file>conf/scripts-definition.xml</scripts-defintion-file>
  <hosts-definition-file>conf/hosts-definition.xml</hosts-definition-file>
  <scripts-directory>scripts/</scripts-directory>
  <property-refresh-frequency>600</property-refresh-frequency>
  <maximum-polling-thread>10</maximum-polling-thread>
</configuration>
```

<poller-definition-file> The path to the file where hosts are mapped to scripts to be executed on them. By default, this file is named *poller-definition.xml*

<scripts-defintion-file> The path to, and name of, the file which includes the names of scripts that will be executed on hosts to retrieve performance data. By default, this file is named *scripts-definition.xml*

<hosts-defintion-file> The path to, and name of, the file defines hosts, and groups of hosts, upon which scripts will be run to retrieve performance data. By default, this file is named *hosts-definition-file.xml*

<scripts-directory> The directory path where scripts referenced in the *scripts-definition-file.xml* are located. Scripts can be placed in subdirectories on this main reference point, to any level. By default, scripts are divided into directories by operating system.

<property-refresh-frequency> The *remote-shell-collector.xml* caches the properties of each raw value sent to the next component. This cache is used so the backend does not have to refresh the properties of a variable when they have not changed. However, it needs to be reset occasionally so the cache does not grow too large, and does so at the specified value. The frequency is in seconds and the default is 600, or 10 minutes.

<maximum-polling-thread> The maximum number of threads that can run concurrently in order to execute the scripts on the remote hosts. The default value is 10.

3 Configuring the hosts-definition.xml file

This is a description of the XML tags in the *hosts-definition.xml* file. This is a typical example configuration. The constituent tags in the *hosts-definition.xml* file are wrapped in **<hosts-definition>** tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<hosts-definition xmlns="http://www.watch4net.com/Remote-Shell-Collector"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Remote-Shell-Collector_hosts-definition.xsd">
  <hosts>
    <host id="Host-1" pollingType="ssh" os="linux" hostname="10.128.5.100" port="22"
      username="root" password="{60CF6D571152AA30F80A565D7B43208C970A014F1980111482617542839}" />
    <host id="Host-2" pollingType="windows" os="windows" hostname="10.128.5.234" username="
      admin" password="examplePass" />
    <host id="Host-3" pollingType="ssh" os="windows" hostname="10.128.5.234" port="22"
      username="admininstrator" password="examplePass2" />
  </hosts>
  <host-groups>
    <host-group name="group1">
      <host alias="Host-1" />
      <host alias="Host-2" />
    </host-group>
  </host-groups>
</hosts-definition>
```

```
<host alias="Host-3" />
</host-group>
</host-groups>
</hosts-definition>
```

3.1 Hosts section

The first section of the *hosts-definition.xml* file, enclosed in `<hosts>` tags, lists hosts on which scripts are to be executed to collect performance data, and their connection parameters.

<host> Each `<host>` tag encloses the definition and connection parameters for a single host, upon which common scripts are to be run.

id A unique ID for the host. This ID can be used in two places:

- in the *poller-definition.xml* file, which indicates that this is the individual host that is to be polled.
- in the *hosts-definition.xml* file, to define the groups of hosts to be collectively polled.

pollingType The type of communication method used between the server and the remote host. Typically, **ssh** is used for Linux hosts, and **windows** for Windows hosts.

Note: In order to poll Windows hosts, your Remote Shell Collector instance must be installed on a Windows server. However, SSH can also be used for Windows hosts, as long as they have SSH capabilities, for example by using Cygwin or OpenSSH. For further details regarding pre-requisites, please refer to the section [6. Polling Windows Hosts](#)

os The operating system of the host to be polled. This can be any name as long as the directory has the exact same name.

hostname The hostname or IP address of the host to be polled.

port The port on the remote host to communicate on. For example, for SSH, this is 22 by default regardless if it is a Linux or Windows host (although this can be changed by the administrator). The port attribute is optional as it will not be used in some cases, such as with the Windows pollingType.

username The username for the user account you want to use to access the remote host. You can use the *ondemand-script-execution* tool to validate how your scripts are behaving on the remote host with a selected user account to check results.

password The password corresponding to the username. You can enter the password directly in plain text or use the *crypt-password.sh/cmd* script available from <APG>/<bin> to enter an encrypted one.

pemFile The file containing a DSA or RSA private key of the user (in PEM format). When using this parameter the *password* is optional. It is used as the *passphrase* when the PEM file is encrypted. (*This parameter is only available for the linux os.*)

3.2 Groups section

The second section of the *hosts-definition.xml* file groups hosts defined in the first section.

<host-group> Each <host-group> tag encloses a list of hosts, defined in the first section of the *hosts-definition.xml* file, upon which common scripts are to be run.

name The name of the group of hosts. The value of this attribute will be what is used in the *poller-definition.xml* file to indicate this group.

<host> Each <host> tag encloses a single host, defined in the first section of the *hosts-definition.xml* file, upon which common scripts are to be run.

alias A reference to the ID attribute of a host defined in the first section of the *hosts-definition.xml* file.

4 Configuring the scripts-definition.xml file

This is a description of the XML tags in the scripts-definition.xml file. It is a typical example of referencing three script files explicitly. All constituent tags in the *scripts-definition.xml* file are wrapped in `<scripts-definition>` tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<scripts-definition xmlns="http://www.watch4net.com/Remote-Shell-Collector"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Remote-Shell-Collector_scripts-definition.xsd"
>
  <script id="script1" file="get-load" />
  <script id="script2" file="get-processes" />
  <script id="script3" file="get-sensors" />
  <glob-script id="script4" glob="*load*" />
  <pattern-script id="script5" pattern=".*" />
</scripts-definition>
```

4.1 Referencing script files

You can reference script files from the scripts-definition.xml file, which must be located in the scripts directory as defined in the *remote-shell-collector.xml* file.

<script> Each script file referenced is enclosed in its own `<script>` tags.

id The unique id corresponding to the script file referenced. This is the value used in the poller-definition.xml file to map the script to the host or host group to be polled.

file The name of the script file located in the scripts directory that this `<script>` tag references. All subdirectories under the scripts directory defined in the *remote-shell-collector.xml* file are searched. Script filenames do not take file extensions. If there are two files with the same name, for example, but with different file extensions, the first one that is come across is used and a warning is issued.

forced-script Forces a script to be used. The script path needs to specify the file extension. This can be useful for using a single script for all the hosts, even if they don't all have the same OS.

<glob-script> Each glob-script finds scripts that match the glob.

id This is the same as above.

glob The glob to use to find scripts.

<pattern-script> Each pattern-script finds scripts that match the regex.

id This is the same as above.

pattern The regex pattern to use to find scripts.

5 Configuring the poller-definition.xml file

This is a description of the XML tags in the *poller-definition.xml* file. It is a typical example defining two polling-groups, each of which executes three separate scripts on an individual host and group of hosts.

The constituent tags in the *poller-definition.xml* file are wrapped in **<polling-groups>** tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<poller-definition name="main-poller" xmlns="http://www.watch4net.com/Remote-Shell-Collector"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.watch4net.com/Remote-Shell-Collector_poller-definition.xsd">
  <polling-groups>
    <polling-group name="polling-group1" pollingPeriod="60">
      <scripts>
        <script alias="script1" />
        <script alias="script2" />
        <script alias="script3" />
      </scripts>
      <polled-hosts>
        <host alias="acme-server1" />
        <host group-alias="group1" />
      </polled-hosts>
    </polling-group>
    <polling-group name="polling-group2" pollingPeriod="120">
      <scripts>
        <script alias="script4" />
      </scripts>
    </polling-group>
  </polling-groups>
</poller-definition>
```

```
<script alias="script5" />
<script alias="script1" />
</scripts>
<polled-hosts>
  <host alias="acme-server2" />
  <host group-alias="group2" />
</polled-hosts>
</polling-group>
</polling-groups>
</poller-definition>
```

5.1 Tags and attributes

The tags in the *poller-definition.xml* file map scripts defined in the *scripts-definition.xml* file, to hosts and host groups defined in the *hosts-definition.xml*, indicating which scripts should be executed, and how often, on which hosts, to collect performance data.

<polling-group> The **<polling-group>** tag defines scripts that will run on the hosts and groups of hosts indicated.

name The unique name of the polling-group.

pollingPeriod The interval in seconds when the indicated scripts are executed on the indicated hosts or group of hosts.

<scripts> The **<scripts>** tag wraps multiple instances of the **<script>** tag, each of which references a script defined in the *scripts-definition.xml* file.

<script> Each **<script>** tag references a single script defined in the *scripts-definition.xml* file. Each script will be executed on all host and host groups defined in the polling-group.

alias The alias attribute references a single script defined in the *scripts-definition.xml* file through its ID indicated there.

<polled-hosts> The **<polled-hosts>** tag wraps multiple instances of the **<host>** tag, each of which references a host or host group defined in the *hosts-definition.xml* file.

<host> Each **<host>** tag references a single host or host group defined in the *hosts-definition.xml* file. Each script will be executed on all host and host groups defined in the polling-group. Each **<host>** tag must include either an alias or group-alias attribute.

alias The alias attribute references a single host defined in the *hosts-definition.xml* file through its ID indicated there.

group-alias The group-alias attribute references a single host group defined in the *hosts-definition.xml* file through its ID indicated there.

Note: You must use either an alias or a group-alias attribute, but not both.

4

Scripts

In general, there are two types of data that can be returned and parsed by APG:

- Raw Value data
- CSV data

This is an example of data returned from a script that can be parsed by APG using the RAWVALUE dataformat:

```
APGLOG: Success polling
APGVERSION: 5.0
DATAVERSION: 1.0
DATATYPE: TIMESERIES
DATAFORMAT: RAWVALUE
```

—

```
1224251948 GROUP-TEST DEFAULT 1Min 0.01 unit=load source=DEFAULT name=1MinLoad
1224251948 GROUP-TEST DEFAULT 5Min 0.16 unit=load source=DEFAULT name=5MinLoad
1224251948 GROUP-TEST DEFAULT 15Min 0.24 unit=load source=DEFAULT name=15MinLoad
```

—

The following is an example of data returned from a script that can be parsed by APG using the RAWVALUE dataformat:

```
APGLOG: Success polling
APGVERSION: 5.0
DATAVERSION: 1.0
DATATYPE: TIMESERIES
DATAFORMAT: CSV
-
1234567890;group;variable name;111222333;prop1=val1;prop2=val2;prop3=val3
-
```

1 Header

You can include optional header information in scripts, for example by formatting it at the beginning of the file:

```
echo 'DATAFORMAT: CSV'
```

Not all of this information is currently used, but it is a good idea to include it to ensure future compatibility.

The only one of these header information elements with a default value is DATAFORMAT, which will be set to RAWVALUE is not explicitly indicated.

APGLOG This information will appear in the log of the collector. Can be useful to debug

APGVERSION The current version of APG you are using.

DATAVERSION The current version of the data, which currently only applies to the timeseries data type since this is the only data type supported with release of the Remote-Shell-Collector; in future releases this will be expanded.

DATATYPE Currently only TIMESERIES is available. This will expand to include EVENTS in the future, for example.

DATAFORMAT Both RAWVALUE and CSV are supported.

2 Metric format

Metrics can be returned in one of two formats, raw value, or CSV.

In the case of raw values, they must be tab delimited (␣) like in the following:

```
1224251948 GROUP-TEST DEFAULT 15Min 0.24 unit=load source=DEFAULT name=15MinLoad
```

This format is identical to how raw values are passed through the timeseries pipeline, so you can refer to the Raw value format Appendix in the Backend Administration Guide for more information.

For quick reference, the components of the raw value format in reference to the example above are Unix timestamp (in seconds): 1224251948, aggregation group: GROUP-TEST, variable name: DEFAULT 15Min, variable value: DEFAULT 0.24 followed by an unlimited number of property names and values: unit=load, source=DEFAULT, name=15MinLoad, etc.

In the case of CSV values, you can set the delimiter, like in the following case using the same example values as above:

```
1234567890;GROUP-TEST;DEFAULT 15Min;0.24;unit=load;source=DEFAULT;name=15MinLoad
```


5

Testing Scripts

The Remote Shell Collector is provided with a command line tool to test your scripts against your various hosts. This tool is very useful to ensure that the scripts are executed properly on the remote hosts and that the returned results is properly parsed in order to generate some valid raw values.

The *ondemand-script-execution.sh* (Linux) and *ondemand-script-execution.bat* (Windows) scripts are the entry point to test your scripts. They rely on a running instance of a Remote Shell Collector to execute the scripts on remote hosts. The available remote hosts are the one defined in the *hosts-definition.xml* corresponding to the running instance of the Remote Shell Collector that you are using. A *ondemand-script-execution.sh* located on *machine A* can reach the hosts defined through a Remote Shell Collector located on *machine B* in a transparent way.

No credentials nor hostname/port number need to be specified in order to reach those remote hosts, since the Remote Shell Collector use to execute the request on the remote hosts (either a local or remote one) is already aware of those details (from it's corresponding *hosts-definition.xml* file).

Once the script is executed on the remote host, the parsed raw values from the output is sent back by the Remote Shell Collector. They are not send to the next collecting element.

6

Polling Windows Hosts

In order to use the Remote Shell Collector to poll Windows hosts, both the APG server and Windows hosts being polled must have Windows Remote Management (WinRM) installed.

Windows Remote Shell (WinRS), is used by the Remote Shell Collector 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.

1 Windows Platform Support and Requirements

The following are the platform and software requirements for polling Windows hosts with the Remote Shell Collector.

1.1 APG Windows Servers

The following are prerequisites for APG servers running the Remote Shell 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 Remote Shell Collector.

1.2 Other Requirements

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

1.3 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 Remote Shell 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 Remote Shell Collector.

1.4 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.

Glossary

Automated Performance Grapher (APG) APG is the global name of Watch4net's software suite used to gather, process, and display performance and event data as well as alarming, all related to your infrastructure. Data is gathered and flows through a variety of collecting and processing components that can be distributed on a network, as well as a Frontend Web Portal responsible for the data display. APG is also comprised of a set of complementary tools and modules that aid in the processing, organization, and display of the data collected from across your network, including its hardware, applications, storage, databases, mobile-related, and more.

Backend The APG Backend is the facility which transforms and pushes to the database the time-series raw data that has been collected and manipulated through the processing components. It determines how data is stored, for example, the aggregates that are computed and kept. The data that flows through the collecting and processing components ends and interfaces with one or more databases through the Backend. There is another Backend module, called the Alerting Backend, which deals with the necessary backend communication for the Alerting. See the Alerting documentation for more information. Event data is conversely written directly to the database and does not go through the Backend.

module.

Collectors Collectors are lightweight modules which gather timeseries data from third-party technologies and ingest it into the APG processing pipeline as raw values. Examples of collectors are the Smarts-Collector and the SNMP-Collector.

Connectors Connectors are components at the end of the processing pipeline that output time-series data to a specific location, for example the MySQL Storage Connector to a MySQL database, or an Oracle Storage Connector to an Oracle database. You can also use a connector, for example, to output to human readable form for debugging using a File Connector, for instance.

Events Events are discrete or durable situations that arise from a given piece of technology, such as when virtual machines are created, removed, or migrated on a VMWare vSphere. These events can be captured, for example in this case, via the VMWare Listener, and this information can appear in reports similarly to raw value data.

The composition of events can vary, for example, some may have multiple associated time values for durable events which would be used in an event-based report (specifically overlays), and each type of event can have different associated properties which are organized through event mapping files. An event mapping file maps events as recorded in the events database, to how they will appear in the Frontend, allowing proper differentiation and organization of events from multiple sources.

Events have a separate processing pipeline than raw values, and use different processing components, for example, listeners are responsible for gathering events, rather than the collectors used for timeseries. Events are consequently written to a different database, and can make use of, for example, the Datastore, which is a high-performance database specifically for events.

Filters Filters are components used to manipulate data in the timeseries pipeline. For example, the Property Tagging filter can add new properties to raw values based on properties the raw values already have, or the Cross Referencing filter can make sure two different indicator series are tagged with the same properties.

Frontend The Frontend refers to the Web Portal (also referred to as a web application) used to display timeseries-based and event-based data in reports, and which is covered in this guide. The Frontend interfaces with the same database(s) as the Backend, retrieving data that has been stored there, for display purposes, for example, in APG reports.

Processing Pipeline The APG processing pipeline is comprised of the individual module components that run on your network to gather, process, and store timeseries-based performance and event-based data. These components include collectors and listeners, which are the data entry point from (third-party) technology, filters and processing elements, which somehow manipulate the data en-route, connectors and database writers which output the data to their final destinations, as well as one or more databases that stores the data, and makes it available in the Frontend.

In general, there are separate pipelines for timeseries and event data, with different kinds of modules available in each, and which terminate at different database types, either storing performance or event data. These flows can interact however on some occasions, for example, using the Event Aggregator, you can retrieve events in the event pipeline, manipulate them somehow, then push these events into the timeseries pipeline. Installation and configuration information is available from the individual module administration guides, and from the APG Backend Admin Guide.

Raw values A raw value in APG is a floating point value, at a specific date and for a specific indicator. This is how timeseries-based performance data is processed in the APG Backend and recorded in the database. Raw values can have meta-data associated to them, which is referred to as their properties. A series of raw values, for example, can be plotted in an APG graph, by selecting a property associated to it.

ReportPack A ReportPack is a collection of settings and report templates prepared by Watch4net's professional services customized for a specific piece of technology. Reports from ReportPacks can often be used out-of-the-box to monitor their targets and can as well be edited in APG to further hone-in on your needs. ReportPacks, containing their individual report templates, can be imported and exported in APG. ReportPacks can rely on specific collecting and processing components for their data, which may be included and/or pre-configured when receiving the ReportPack from Watch4net.

Reports Reports are displayed in the APG Web Portal (Frontend), and convey information about timeseries-based performance and/or event-based data related to your infrastructure in a variety of formats, including tables and graphs.