



EMC® Watch4net®

Version 6.3u2

SQL Collector Administration Guide

P/N: pending

Rev : 40030

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
Prerequisites	iii
Common Abbreviations and Acronyms	iv
Where to get help	v
1 Product Overview	1
2 Architectural Overview	3
3 SQL Data Sources	5
1 Microsoft SQL Server	6
1.1 Required user privileges	6
1.2 Microsoft SQL Server JDBC Driver	6
1.3 jTDS JDBC Driver (for Microsoft SQL Server)	7
2 Oracle	8
2.1 Required user privileges	8
2.2 Oracle JDBC Driver	8
3 MySQL	9
3.1 Required user privileges	9
3.2 MySQL Connector/J	9
4 Other DBMS	10
4.1 Any JDBC Drivers with API version 4.0 or higher	10
5 Logging with log4jdbc	11
4 Installation	13

- 5 Configuration 15**
 - 1 Collecting service configuration 16
 - 2 Collector configuration 16
 - 2.1 Global structure 17
 - 2.2 The SQL queries configuration 22
- 6 Start Up 25**
- 7 Troubleshooting 27**
 - 1 The collector won't start 27
 - 2 The collector started successfully, but no data is showing up in APG's frontend 28
 - 3 Enable SQL queries logging 28
- 8 Configuration example 29**

Preface

Purpose

This guide describes the common usage of the SQL Collector for APG, which includes all the necessary information to install and configure the collector.

Intended Audience

This guide is intended to be read by any *APG* administrator who will configure the SQL Collector to import data from SQL sources into APG.

Prerequisites

It is assumed that the *APG* application has been properly installed and configured on the server. Make sure you have acquired and installed the license for the SQL Collector.

Common Abbreviations and Acronyms

<i>DBMS</i>	DataBase Management System
<i>JDBC</i>	Java DataBase Connectivity
<i>SQL</i>	Structured Query Language
<i>XML</i>	eXtensible Markup Language

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 SQL Collector is a generic module for APG that allows metrics and properties to be collected from any supported SQL data source and pushed into APG.

Requirements are as follows:

- APG 5.x

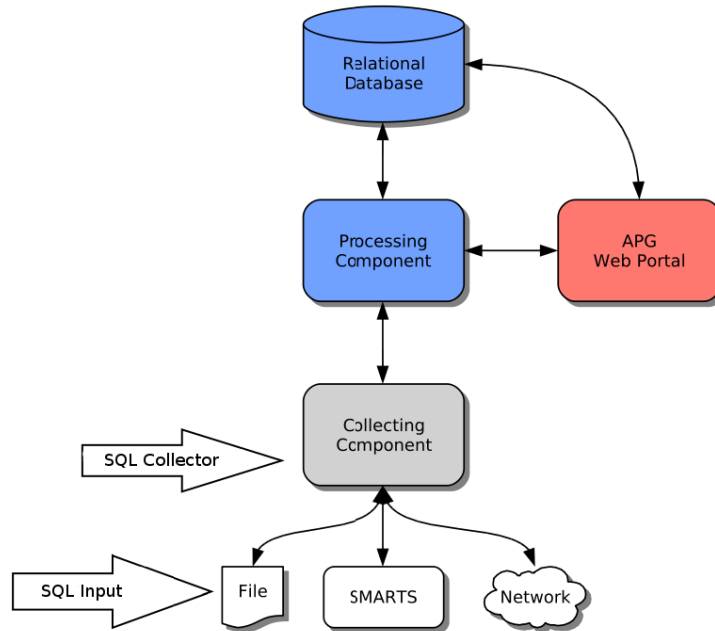
Moreover, it relies on provided JDBC drivers to connect to the SQL data sources:

- Microsoft SQL Server JDBC Driver version 2.0 or higher (`sqljdbc4.jar`)
- jTDS JDBC Driver (for Microsoft SQL Server) version 1.2.4 or higher (`jtds-*.jar`)
- Oracle JDBC Driver version 1.4 or higher (`ojdbc*.jar`)
- MySQL Connector/J version 5.0.7 or higher (`mysql-connector-java-*-bin.jar`)

2

Architectural Overview

The SQL Collector integrates seamlessly into the APG structure. The following diagram illustrates APG's global architecture and how collectors can be attached to it.



The SQL Collector is composed of one main module which is responsible for these tasks:

- Loads its configuration.
- Creates a thread pool to allow multi-threading and sets up the number of simultaneous threads.
- Launches the main tasks* at the specified polling interval.
- Clears the properties cache at the specified refresh properties interval.
- Retrieves data from SQL data sources and transforms it into raw values (metrics).
- Nest child tasks* to retrieve properties if necessary.
- Forwards complete raw values to APG (any registered next element).

* Each task may contain multiple SQL queries if necessary.

3

SQL Data Sources

SQL data can be acquired from any JDBC compliant data sources including Microsoft SQL Server, Oracle and MySQL. The data source configuration properties can be specified in the SQL Collector configuration file (see section [Collector configuration](#) page 16).

1 Microsoft SQL Server

1.1 Required user privileges

The user must be granted the SELECT and CONNECT privileges on the database or the schema. The collector supports SQL Server user and Windows user (local and domain).

1.2 Microsoft SQL Server JDBC Driver

Description The official JDBC Driver provided by Microsoft.

Usage

- the URL:

jdbc:sqlserver://IP:1433;instanceName=INSTANCE;databaseName=DATABASE;selectMethod=cursor

- the driver class name: com.microsoft.sqlserver.jdbc.SQLServerDriver
- the library: sqljdbc4.jar

Single-Sign-On (Windows authentication) This JDBC driver supports the use of Type 2 integrated authentication on Windows only. To use SSO, you need to download the [Microsoft SQL Server JDBC Driver 2.0](#).

- add ;integratedSecurity=true at the end of the URL.
- leave the username and password attributes empty.
- place the sqljdbc_auth.dll file into a directory on the Windows system PATH on the host where the JDBC driver is installed.

Use this table to know which DLL version to use:

Java Virtual Machine	Processor	DLL version to use
32-bit	x86	auth\x86\sqljdbc_auth.dll
32-bit	x64	auth\x86\sqljdbc_auth.dll
64-bit	x64	auth\x64\sqljdbc_auth.dll
64-bit	IA-64	auth\IA64\sqljdbc_auth.dll

1.3 jTDS JDBC Driver (for Microsoft SQL Server)

Description An open source (GNU LGPL) 100% pure Java (type 4) JDBC 3.0 driver for Microsoft SQL Server (6.5, 7, 2000, 2005 and 2008) based on FreeTDS. This JDBC driver may be a good solution for specific needs like Windows authentication on non-Windows hosts.

Usage

- the URL:

```
jdbc:jtds:sqlserver://IP:1433;instanceName=INSTANCE;databaseName=DATABASE;selectMethod=cursor
```

- the driver class name: `net.sourceforge.jtds.jdbc.Driver`
- the library: `jtds-*.jar`

Windows authentication

- add `;domain=DOMAIN_NAME` at the end of the URL.
- specify the `username` and `password` attributes.

Single-Sign-On with the native SPPI library (Windows only) You need to download the [jTDS - SQL Server JDBC driver](#).

- add `;domain=DOMAIN_NAME` at the end of the URL.
- leave the `username` and `password` attributes empty.
- place the `ntlmauth.dll` file to a directory on the Windows system PATH on the computer where the JDBC driver is installed.

Use this table to know which DLL version to use:

Java Virtual Machine	Processor	DLL version to use
32-bit	x86	x86\SS0\ntlmauth.dll
32-bit	x64	x86\SS0\ntlmauth.dll
64-bit	x64	x64\SS0\ntlmauth.dll
64-bit	IA-64	IA64\SS0\ntlmauth.dll

2 Oracle

2.1 Required user privileges

The user must be granted the `SELECT` privilege on the tables, views...

2.2 Oracle JDBC Driver

Description The official JDBC driver provide by Oracle.

Usage

- the URL:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=IP)(PORT=1526))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=NAME)))
```

- the driver class name: `oracle.jdbc.OracleDriver`
- the library: `ojdbc*.jar`

Note: You have to use double quotes " around each label in the SQL queries to make sure that Oracle returns the labels in lowercase.

Example:

```
SELECT field1 AS "cid_mycid", field2 as "value", field3 as "timestamp" FROM table1
```

3 MySQL

3.1 Required user privileges

The user must be granted the `SELECT` privilege on the schema.

3.2 MySQL Connector/J

Description MySQL Connector/J is a native Java driver that converts JDBC calls into the network protocol used by the MySQL database.

Usage

- the URL:

```
jdbc:mysql://IP:3306/DATABASE
```

- the driver class name: `com.mysql.jdbc.Driver`
- the library: `mysql-connector-java-*-bin.jar`

4 Other DBMS

4.1 Any JDBC Drivers with API version 4.0 or higher

Description The JDBC API is an industry standard so most DBMS have their own JDBC drivers.

Usage The SQL collector should work with any DBMS but it requires:

- the URL
- the driver class name
- the library: *.jar

5 Logging with log4jdbc

Description A Java JDBC driver that can log SQL and/or JDBC calls (and optionally SQL timing information) for other JDBC drivers using the Simple Logging Facade For Java (SLF4J) logging system.

Usage To use this logger, you need to add `jdbc:log4` at the beginning of the URL string.

- the URL:

```
jdbc:log4jdbc:DBMS\_STRING://IP:3306/DATABASE
```

- the driver class name: `net.sf.log4jdbc.DriverSpy`
- the libraries: `log4jdbc*.jar` + the usual library (JDBC driver) matching the `DBMS\_STRING`

4

Installation

Prerequisites The following operations must be performed before any installation or configuration steps are taken.

1. Check if the SQL Collector is available:
 - Use the Module Manager to list the available packages.
 - Look for the *Filename* `sql-collector-*` and the *Module* `SQL-Collector`.
 - If you can't find the SQL Collector, you need to get the package `sql-collector-*.pkg` and place it in the APG Module Repository directory.
 - Then, use the Module Manager again to check if the SQL Collector module is available.
2. Shut down the Collector Manager service with the Module Manager.
3. Backup this current Collector Manager main configuration file:
`<APG Root>/Collecting/Collector-Manager/<Instance>/conf/collecting.xml`

Module installation

- Use the Module Manager to install the SQL Collector module.
You should use an instance name matching the SQL Collector usage.

5

Configuration

This section is about the configuration file for the SQL Collector. In APG 5.x and higher, the file is located in the `<APG Root>/Collecting/SQL-Collector/<Instance>/conf` directory. In case this file does not exist, it must be created in this folder.

An SQL collector instance can collect multiple data sources but, if needed, you can use one instance per collected source.

1 Collecting service configuration

The SQL Collector must be configured in the Collector Manager main configuration file:

- <APG Root>/Collecting/Collector-Manager/<Instance>/conf/collecting.xml.

Add the following section if it is not present:

```
<collectors>
...
<collector enabled="true" name="name" next="Backend" conf="SQL-Collector/<Instance>/conf/sql-
collector.xml" />
...
</collectors>
```

You should choose a collector name that reflects the source from which the collected data comes from. Please, refer to the Collecting Manager documentation for more details about the collecting service configuration.

2 Collector configuration

The file SQL-Collector/<Instance>/conf/sql-collector.xml is the configuration file for the adapter. It contains information about how data will be collected, as well as how the data will be interpreted. If you already have the SQL Collector configuration file, you can skip this section and just use this file. If you have to create or modify the SQL Collector configuration file, the administrator has to decide how to name this file and to make sure that this name is used to configure the collector in the collecting.xml file (see [Collecting service configuration](#) page 16).

2.1 Global structure

The following is the global structure of the configuration file:

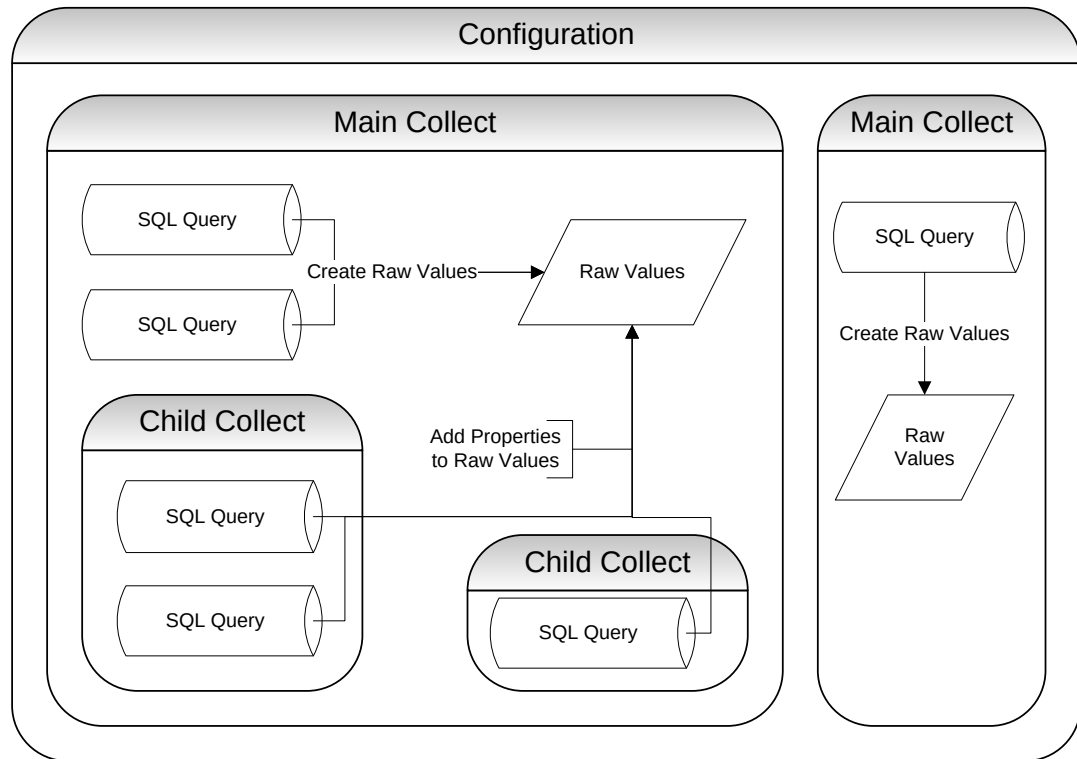


Figure 1: Configuration file structure

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

- `<simultaneous-collecting>`: The number of simultaneous collects for this SQL collector. The value should be 1 because you should collect only 1 SQL source.
- `<collecting-group>`: The group in which the variables will be collected. This group *must* be defined in `<APG Root>/Backends/APG-Backend/<Instance>/conf/config.xml`, or values will be ignored.
- `<source>`: The value of the *source* property, as it will appear in the web portal. The *source* property is used to identify the collector that pushed the given metric in the web portal.
- `<collecting-configuration id="id">`: This is a container for the configuration options that relate to the same data source.

The mandatory parameter *id* must be unique and is used to identify all variables that were obtained from this data source. The *id* is also used to save the state of the collect in a *id.save* file. If multiple instances of the SQL Collector are running on the same APG install, they must have a different *id*. The *id* must match the file system rules of the host operating system.

- `<sql-collector>`: the Java implementation of the SQL Collector.
There is only 1 available right now:
`com.watch4net.apg.v2.collector.plugins.sqlcollector.DefaultSQLCollector.`
- `<polling-interval>`: the time (in seconds) between 2 pollings.
- `<max-threads>`: the number of threads used by this collect. Should match to the number of DB connections available.
- `<refresh-properties>`: the duration (in seconds) during which the raw value properties are kept in cache and when the properties are refreshed.
If set to 0, no properties cache will be used and properties will be refreshed when different or every 24 hours.

- `<main-collecting-configuration id="id">`: this is a container for the configuration options that relate to the main queries that collect the raw values (metrics). If the collected source requires multiple distinct queries to collect distinct raw values (metrics), you should use multiple `main-collecting-configuration`. A main collect can contain multiple queries and sub queries if needed.

The mandatory parameter *id* must be unique and is used to identify the collect.

- * `<db-connection-id>`: the DB connection ID used to run the SQL query (see `<db-connection>` page 21).
- * `<query id="name">`: the SQL query that will be executed each time. The collected raw values (metrics) don't need to contain all properties, *child-collecting-configuration* can be used to collect other properties.
You can use multiple `<query>` tags in the same *main-collecting-configuration* only if they collect similar metrics (same columns in the SQL query).
The mandatory parameter *name* must be unique and is used to link the `<query>` and `<initial-query>`.
The [The SQL queries configuration](#) section page 22 explains how to write the query.
- * `<initial-query id="name">`: (optional) the SQL query that will be executed only at the first execution to initiate the collector. You can use multiple `<initial-query>` tags in the same *main-collecting-configuration* to match multiple `<query>`.
The mandatory parameter *name* must be unique and is used to link the `<query>` and `<initial-query>` tags.
The [The SQL queries configuration](#) section page 22 explains how to write the query.
- * `<fetch-by>`: the number of SQL result lines fetched by the collector before it processes them as raw values (metrics). This is useful when the query result size is large as it allows the collector to process result lines as they are retrieved.

- * `<child-collecting-configuration id="name">`:
this is a container for the configuration options that relate to the SQL queries that collect additional properties to complete the data collected by the main queries.
The mandatory parameter *name* must be unique and is used to identify the collect.
 - `<db-connection-id>`: the DB connection ID used to run the SQL query (see `<db-connection>` page 21).
 - `<query id="name">`: the SQL query that will be executed each time. It collects raw values propertie to complete the raw values collected by the *query* in the parent *main-collecting-configuration*.
You can use multiple `<query>` into the same *child-collecting-configuration* only if they collect similar properties (same columns in the SQL query).
The mandatory parameter *name* must be unique.
The [The SQL queries configuration](#) section page 22 explains how to write the query.
 - * `<raw-value-variable separator="separator" properties="properties">`:
this element contains the configuration of the raw values variable property that identify each metrics.
The mandatory parameter *separator* is used to put a character between each values composing the variable property.
The mandatory parameter *properties* is used to list all the values used to create the variable property. Each value is separated by a ','. Moreover, each value is between '`{`' and '`}`'.
The available values are:
 - the raw value properties: "`{property=name}`",
 - the `<collecting-configuration>` id attribute: "`{collecting-configuration-id}`",
 - the internal IDs: "`{cid=name}`".
- The following example uses the *source* property, the `<collecting-configuration>` *id* attribute and the cid *cc1*.

```
<raw-value-variable separator="_"
  properties="{property=source},{collecting-configuration-id},{cid=cc1}" />
```

- `<db-connection id="id" driverClassName="driver.class.Name" url="url" username="username" password="password">`:

This element contains the database connection's configuration.

- The parameter *id* must be unique and is used to identify the database connection.
- The parameter *driverClassName* is the fully qualified Java class name of the JDBC driver to be used.
- The parameter *url* is the JDBC connection URL.
- The parameter *username* is the user name used to connect to the database.
- The parameter *password* is the password used to connect to the database.
You may use the encryption library to crypt this password. The script (shell and UI) is usually located in the APG bin directory.
- The parameter *connectionProperties* can be used to set other JDBC properties.
- Other configuration properties can also be added here.

- `<connection-group id="id">`:

This element is used to run the same configuration on multiple databases.

Each `<connection-group>` needs to contain all `<db-connection-id>` used in the `<collecting-configuration/>`.

- The parameter *id* must be unique and is used to identify the database connection group.
- `<db-connection>`: the database connection used to run the SQL query (same as `<db-connection>` [page 21](#)).

2.2 The SQL queries configuration

The SQL queries used in the configuration file are regular SQL queries (matching the targeted DBMS rules) with some other rules to identify the data.

You need to use aliases to identify each field:

- **id_***: a field(s) used to filter collected raw values in the main query.
This example collects all raw values with the *field1* value greater than the max already collected *field1* values.

```
SELECT field1 AS id_myid ...  
FROM table1  
WHERE field1 > ?max_id_myid?  
ORDER BY field1
```

- **timestamp**: the timestamp of raw value (in milliseconds).
The collector timestamp can be used instead of the database timestamp with `+LOCAL_TIMESTAMP+`:

```
SELECT +LOCAL_TIMESTAMP+ AS timestamp ...  
FROM table1
```

- **value**: the value of the raw value. Use only if one result line generates one raw value.
- **value_***: the value of the raw value. Use only if one result line generates multiple raw values.
Use *value_name1*, *value_name2*... to generate multiple raw values with the *name* property set to *name1*, *name2*...
- **cid_***: the field(s) use to link the raw values collected by the main query and the properties collected by child queries. The *cid* should be in the main query and in one or more child queries.
This example shows a main query generating raw values with *field1* as their *cid*:

```
SELECT field1 AS cid_mycid, field2 as value, field3 as timestamp  
FROM table1
```

This child query will complete the collected raw values by adding *field2* as *property1* for the raw values matching *field2* as *cid*:

```
SELECT field1 AS p_property1, field2 AS cid_mycid,  
FROM table2  
WHERE field2 IN (?cid_mycid[*]?)
```

- **p_***: the raw value properties (Example: *p_device* identify the *device* property).

To identify the ID that filters the queries in the main collect, it must use this pattern:

- `?KEY_id_X?` where `KEY` is the key for the collected ID. The possible keys are:
 - `last`: the last collected value
 - `min`: the minimum collected value
 - `max`: the maximum collected value

The following example uses *field1* as *id_myid*. The raw values are filtered by *last* collected *field1* value:

```
SELECT field1 AS id_myid
FROM table1
WHERE field1 > ?last_id_myid?
```

To link collected raw values and child queries, the child queries in the child collects need to use this pattern:

- `?cid_X[*]?` where `cid_X` is a column label of the main query.
The following example uses *field1* to filter the relevant properties to collect:

```
SELECT field1 AS cid_mycid, field2 AS p_property1
FROM table1
WHERE field1 IN (?cid_mycid[*]?)
```


6

Start Up

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

1. Use the Module Manager to check the *status* of the *mysql* and the *backend* services.
If they are not *running*, use the Module Manager to *start* the *mysql* and the *backend* services.
2. Use the Module Manager to *start* the *collector-manager* service.
3. Look at the collecting logs to make sure that everything is starting up as it should:
`<APG Root>/Collecting/Collector-Manager/<Instance>/logs`

Troubleshooting

1 The collector won't start

- Make sure that the contents of the <APG Root> folder is accessible for the `apg` user, especially the files that were copied for the installation of the collector (see the [Installation](#) section).
- Look at the collecting logs that are located in:
`<APG Root>/Collecting/Collector-Manager/<Instance>/logs`
These logs contain helpful messages that might help solve the problem.

2 The collector started successfully, but no data is showing up in APG's frontend

If the collector was just started, wait after a few collecting cycles have passed.
Data usually takes some time to show up in APG's frontend for the first time.

If there is still no data, check the following:

- In <APG Root>/Collecting/Collector-Manager/<Instance>/conf/collecting.xml, make sure that the SQL Collector is configured and that it is enabled (see [Collecting service configuration page 16](#)).
- In <APG Root>/Collecting/Collector-Manager/<Instance>/conf/collecting.xml, check the *next* attribute for the SQL 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 SQL 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 (<APG Root>/Collecting/Collector-Manager/<Instance>/logs/).

3 Enable SQL queries logging

To enable SQL queries logging in the Collector Manager logs:

- Use log4jdbc as described in the [Logging with log4jdbc](#) section page 11.
- Add the following lines to the Collector Manager configuration file:
<APG Root>/Collecting/Collector-Manager/<Instance>/conf/logging.properties

```
jdbc.sqlonly.level=INFO
jdbc.sqltiming.level=OFF
jdbc.audit.level=OFF
jdbc.resultset.level=OFF
jdbc.connection.level=OFF
```

- Look at the Collector Manager logs that are located in:
<APG Root>/Collecting/Collector-Manager/<Instance>/logs
These logs now contain all SQL queries executed against the DBMS.

8

Configuration example

The configuration provided with the collector is located is:

```
<APG Root>/Collecting/SQL-Collector/<Instance>/conf/sql-collector.xml.
```

This example does not work but it is here to help you understand the SQL collector configuration.

If the data is collected in 2 different databases or if you want to reduce the amount of data collected, you will need to use child collect. In this example, the data is collected from 2 tables on 2 different DBMS.

Collecting configuration

- `<polling-interval>1800</polling-interval>` means that the main collect query will be run every 30 minutes.
- `<max-threads>5</max-threads>` means that the collector will use 5 threads maximum.
- `<refresh-properties>3600</refresh-properties>` means that the properties collect in the child query will stay in cache during 1 hour and all the raw values properties will be refreshed every hour.

Main collect configuration

- The main collect queries will be executed on the database connection 1:
`<db-connection-id>1</db-connection-id>`.
- The main collect has an initial query to collect the oldest data ID in the table which is the data with the lower ID in this case:
`SELECT MIN(id) AS id_one FROM table1.`
- The main collect query use this ID to filter each new query and to be sure that the same data will not be collected twice:
`SELECT id AS id_one ... WHERE id > ?max_id_one? ORDER BY id ASC.`

Depending on the collected system, it may not be necessary to use this ID.

- The main collect query generate raw values with a value from the `v` column, a timestamp from the `date` column and a property "property1" from the `p1` column:
`SELECT v AS value, date AS timestamp, p1 AS p_property1 ...`
- The raw values are also linked with a child ID `fkid` that will be use to complete the raw value in the child collect:
 - main collect query:
`SELECT id_2 AS cid_fkid FROM table1`
 - child collect query:
`SELECT id AS "cid_fkid" ... FROM table2 WHERE id IN (?cid_fkid[*]?)`
- If the main collect query have more that 1000 result lines, the collector will start to process the result lines as soon as the first 1000 result lines are sent by the database:
`<fetch-by>1000</fetch-by>`

Child collect configuration

- The child collect query will be executed on the database connection 2:
`<db-connection-id>2</db-connection-id>`.
 In this example, the database connection 2 is assumed to be an Oracle database so the labels are surrounded by double quotes ".
- The purpose of this child collect is to complete the raw values generated by the main collect.
- The new property sets will contain 2 properties *property2* and *property3*:
`SELECT p1 AS "p_property2", p2 AS "p_property3" ... FROM table2 ...`
- To link the collected raw values with the new collected property sets, the child ID *fkid* from the main query is used:
`SELECT id AS "cid_fkid" ... FROM table2 WHERE id IN (?cid_fkid[*]?)`

Raw value variable generation

This part sets the collector to generate the raw value variable (the raw value id).

- The different parts of the variable will be separated by an underscore '_':
`separator="_"`
- The variable will contain:
 - the raw value property source: `${property=source}`
 - the collecting configuration id: `${collecting-configuration-id}`
 - the child ID fkid: `${cid=fkid}`
 - the raw value property name: `${property=source}`

A raw value variable will look like: *W4N-SQL-Collector_SQL-Source-Collector_12345_name1*

Database connections

This example contains 2 database connections configurations because there are multiple queries that are executed in 2 different databases.

The first configuration may be a MySQL or SQL Server database and the second, an Oracle database.