Table of Contents

Database Plugin

General

The Database Plugin allows you to write CoDaBix® Datapoint Node values to external databases, like MySQL or Microsoft SQL Server.

What Does the Plugin Do?

The plugin allows you to:

- define database connections
- define triggers (e.g. interval trigger)
- define DataSets, consisting of
  - datapoint Nodes
  - system values

When a trigger fires, the plugin collects values for a “record set” by doing a Synchronous Read of specified CoDaBix® datapoint Nodes (which means CoDaBix® requests the underlying device to actually read its variables) and then writes the values of the record set to a database table.

To use the plugin, you need to create an XML configuration file.

Features

- Collect node values and write them into a database when a trigger fires
- Supports multiple trigger types:
  - Interval trigger
  - Edge value trigger (fires when the value of a node changes to a specified value)
  - Value change trigger (fires when the value of a node changes)

Supported Database Engines

- MySQL 5.5 or higher
- Microsoft SQL Server 2008 or higher

Purpose & Use Cases

- mirror data from CoDaBix® to an external database
- collect data from devices once a trigger fires

Installation

This plugin is part of the CoDaBix® Setup. Please consult CoDaBix® Setup and First Start for more information on how to install and uninstall this plugin.

Requirements

- The machine which runs CoDaBix® must have access to one of the supported database engines.

Configuration

This plugin can be only configured by using the **XML Configuration File** as described below. You will need to create the XML Configuration File (“CoDaBix.DatabasePlugin.Settings.xml”) in the project directory (see Folders & Files). When the file is changed while Codabix is running, the Database Plugin will automatically restart and use the new configuration file.
Structure

PluginSettings Element

Each CoDaBix® plugin defines the root of its element tree by the “PluginSettings” element as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<PluginSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- Channels element -->
</PluginSettings>
```

Channels Element

The `Channels` element serves as a container for one or more `Channel` elements. The `Channel` element defines a group of database connections, triggers and datasets. For each channel there is a worker thread that handles a trigger by collecting Node values and then writing them to the specified database connections.

```
<Channels>
  <Channel id="ch1" active="true">
    <!-- DbConnections element -->
    <!-- Triggers element -->
    <!-- DataSets element -->
  </Channel>

  <!-- More <Channel> elements... -->
</Channels>
```

Each `Channel` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`id`	**yes**	String	Uniquely identifies this channel. This value can be referenced e.g. in one of the `SystemValue` elements.
`active`	no	Boolean	Determines if this channel is actually run. If "`false`", the channel is not started, otherwise it is.

DbConnections Element

The `DbConnections` element serves as a container for one or more `DbConnection` elements.

The `DbConnection` element defines a physical connection to a database, including the table name. This determines to which database table is written.

```
<DbConnections>
  <DbConnection id="con1" type="MSSQL"
                hostname="192.168.0.1" port="1234"
                username="myuser" password="pw"
                database="MyDB" table="MyLogTable" />

  <!-- More <DbConnection> elements... -->
</DbConnections>
```

Each `DbConnection` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
id	**yes**	String	Uniquely identifies this database connection. This is used in the `DataSet` elements to refer to this connection.
type	**yes**	Enumeration	Specifies the type of the database engine. "MSSQL": Microsoft SQL Server "MySQL": MySQL Server
hostname	**yes**	String	The hostname, DNS name or IP address of the database Server.
port	**yes**	Integer	The TCP port of the database server. When using `MSSQL` as Server Type, you can specify `0` as port, which means a connection to the default SQL Server instance shall be established.
username	**yes**	String	The username which is used to establish the database connection.
password	**yes**	String	The **encrypted password** for the username. To get the encrypted password, open CoDaBix, and in the Web Configuration click on **Password Security**. Here, you can enter your original password and encrypt it, so that it can be specified for this attribute.
database	**yes**	String	The database / scheme name.
table	**yes**	String	The name of the database table to which the values shall be written.
connectTimeout	no	Integer	The timeout in seconds to use when connecting to the database. If not specified, the value 30 is used.
commandTimeout	no	Integer	The timeout in seconds to use when running a database command. If not specified, the value 60 is used.
maxPoolSize	no	Integer	The maximum number of active database connections that are pooled. If not specified, the value 40 is used.

Note: Currently, the Database Plugin always writes values to a database table by using an `INSERT` statement (meaning new values are appended to the table at each write).

Triggers Element

The `Triggers` element serves as a container for one or more `Trigger` elements.

The `Trigger` element defines an object which fires an event. A trigger can be referenced in a DataSet element to determine when the Database Plugin should collect and write Node values to the database.

```
<Triggers>
  <Trigger id="t1" type="interval" interval="5000" />

  <Trigger id="t2" type="edge" node="/Path/to/TriggerNode"
edgeValue="1" changeBackValue="0" />
```

```
  <Trigger id="t3" type="valueChange" node="/Path/to/TriggerNode" />

  <!-- More Trigger elements... -->
</Triggers>
```

Each `Trigger` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
id	**yes**	String	Uniquely identifies this trigger. This is used in the `DataSet` elements to refer to this trigger.
type	**yes**	Enumeration	Specifies the type of the trigger. `"interval"`: Fires at a specific regular interval, specified by the `interval` attribute. `"edge"`: Fires when a specific value is written into the specified Node (and the previous Node value was a different value). `"valueChange"`: Fires when a value of the specified Node changes.
interval (when `type="interval"`)	**yes**	Integer	Specifies the interval of the trigger in milliseconds. For example, a value of `2000` means the trigger fires every two seconds.
node (when `type="edge"` or `type="valueChange"`)	**yes**	String	Specifies the Node path of the Node which the trigger should use to check for value changes.
edgeValue (when `type="edge"`)	**yes**	String	The value for which the trigger checks. The trigger fires when this value is written to a Node and the previous Node value was a different one.
changeBackValue (when `type="edge"`)	no	String	If specified, the Database Plugin will write this value into the Node after the trigger has fired and the record set has been collected using an synchronous read.
instantChangeBackValue (when `type="edge"`)	no	String	If specified, the Database Plugin will write this value into the Node immediately after the trigger has fired (before the record set has been collected).

DataSets Element

The `DataSets` element serves as a container for one or more `DataSet` elements.

The `DataSet` element defines a group of datapoint Nodes and SystemValues which will be collected into a record set and then written to one or more database connections. In a DataSet, you can reference one or more `DbConnection` elements (meaning that a DataSet is written to all these database connections) and one or more triggers (meaning the Node values are

collected and written when one of these triggers fires).

```
<DataSets>
  <DataSet id="ds1" writeDelay="2000" writeBufSize="10">

    <!-- One or more DbConnection elements referencing a database
connection... -->
    <DbConnection id="con1" />

    <!-- One or more Trigger elements referencing a trigger... -->
    <Trigger id="t1" />

    <!-- Nodes element -->
    <!-- SystemValues element--->
    <!-- AfterSyncActions element -->
  </DataSet>

  <!-- More <DataSet> elements... -->
</DataSets>
```

Each `DataSet` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`id`	**yes**	String	Uniquely identifies this DataSet.
`writeDelay`	no	Integer	Specifies the maximum delay (in ms) after the cached record sets are written to the database.
`writeBufSize`	no	Integer	Specifies the maximum number of cached record sets until they are written to the database.

For each DataSet, after a set of values has been collected (a record set), the record set is buffered to efficiently handle a number of record sets to write them in one go. If either the `writeDelay` or `writeBufSize` is specified (or both), the buffered record sets are not written to the database until the `writeDelay` time has passed since the last write, or the number of cached record sets specified by `writeBufSize` has been exceeded. If none of these attributes are specified, the record sets are written immediately.

Each`DbConnection` element and `Trigger` element provides the following list of attributes (in the `DataSet` element):

Attribute	Mandatory	Type	Purpose
`id`	**yes**	String	References a previously defined DbConnection or Trigger.

Nodes Element

The `Nodes` element serves as a container for one or more `Node` elements.

The `Node` element references a CoDaBix® datapoint Node using either an absolute or relative Node path. If the `Nodes` element specifies a path in its `root` attribute, the `path` of the `Node` elements is relative to the `Nodes`'s `root` path.

Example with root path:

```
<Nodes root="/Nodes/Demo-Nodes/">
  <Node path="Temperature" column="ColTemp" />
  <Node path="Pressure" column="ColPressure" />
  <Node path="Pressure" property="timestamp"
column="ColPressureTimestamp" />

  <!-- More Node elements... -->
</Nodes>
```

Example without root path:

```
<Nodes>
  <Node path="/Nodes/Demo-Nodes/Temperature" column="ColTemp" />
  <Node path="/Nodes/Demo-Nodes/Pressure" column="ColPressure" />
  <Node path="/Nodes/Demo-Nodes/Pressure" property="timestamp"
column="ColPressureTimestamp" />

  <!-- More Node elements... -->
</Nodes>
```

Each `Nodes` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`root`	no	String	Specifies the path to the parent Node. If specified, the path of the `Node` elements is relative to this root path. Otherwise, the path of the `Node` elements must be the complete path.

Each `Node` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`path`	**yes**	String	The relative path from `root` to the Node, or the absolute path if the `root` of the `Nodes` element is not specified.
`column`	**yes**	String	The name of the database table column in which the value shall be written.
`property`	no (default: `value`)	String	The property of the Node which shall be written. `"value"` (default): The Node's value. `"timestamp"`: The timestamp of the Node's value (UTC). `"timestampLocal"`: The timestamp of the Node's value (local time). `"nodeID"`: The identifier (local identifier) of the Node. `"nodeName"`: The name of the Node. `"nodeDisplayName"`: The display name of the Node. `"nodeUnit"`: The unit of the Node.

SystemValues

The `SystemValues` element serves as a container for one or more `SystemValue` elements.

The `SystemValue` element either contains a literal value or references a predefined system value, which should be written to the database.

```
<SystemValues>
```

```
  <!-- Example 1: Log the time when the trigger fired to DB column
"colTriggerTime" -->
  <SystemValue type="triggerTimestamp" column="colTriggerTime" />

  <!-- Example 2: Log the value "12345" to the column
"colLiteralValue": -->
  <SystemValue value="12345" column="colLiteralValue" />
</SystemValues>
```

Each `SystemValue` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`column`	**yes**	String	The name of the database table column in which the value shall be written.
`type`	**yes** (if `value` is not specified)	Enumeration	Specifies which system value to use. `"triggerTimestamp"`: The timestamp when the trigger has fired (UTC). `"triggerTimestampLocal"`: The timestamp when the trigger has fired (local time). `"channelID"`: The identifier of the containing channel.
`value`	**yes** (if `type` is not specified)	String	Specifies a direct (literal) value which shall be written.

Note: Either the `type` or the `value` attribute must be specified, but not both.

AfterSyncActions Element

The `AfterSyncActions` element serves as a container for one or more `AfterSyncAction` elements.

The `AfterSyncAction` element allows to write a value to a Node after the record set has been collected.

```
<AfterSyncActions>

  <AfterSyncAction type="writeNodeValue" node="/Path/to/Node"
value="MyValue" />

</AfterSyncActions>
```

Each `AfterSyncAction` element provides the following list of attributes:

Attribute	Mandatory	Type	Purpose
`type`	**yes**	Enumeration	Specifies the type of the AfterSyncAction. `"writeNodeValue"`: After collecting the record set, the specified value is written to the Node.
`node`	**yes**	String	Specifies the Node path of the Node to write the value.
`value`	**yes**	String	The value to write.

Example Configuration File

The following is an example configuration file, using the elements as shown above:

CoDaBix.DatabasePlugin.Settings.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<PluginSettings
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Channels>
    <Channel id="ch1" active="true">
      <DbConnections>
        <DbConnection id="con1" type="MSSQL"
                      hostname="192.168.0.1" port="1234"
                      username="myuser" password="pw"
                      database="MyDB" table="MyLogTable" />

      </DbConnections>

      <Triggers>
        <Trigger id="t1" type="interval" interval="5000" />
      </Triggers>

      <DataSets>
        <DataSet id="ds1" writeDelay="2000" writeBufSize="10">

          <DbConnection id="con1" />
          <Trigger id="t1" />

          <Nodes root="/Nodes/Demo-Nodes/">
            <Node path="Temperature" column="ColTemp" />
            <Node path="Pressure" column="ColPressure" />
          </Nodes>

          <AfterSyncActions>
            <AfterSyncAction type="writeNodeValue"
node="/Path/to/Node" value="MyValue" />
          </AfterSyncActions>

        </DataSet>
      </DataSets>
    </Channel>
  </Channels>
</PluginSettings>
```

Diagnostics

The Database Plugin writes the following events into the log file `[LoggingFolder]\logfile.txt` (see Folders & Files path of the log file):

- Channels are started or stopped (because CoDaBix® is starting or stopping, or the configuration file has changed and the plugin restarts).
- A set of values has been successfully written to the database (after doing a synchronous read).
- An error occured when trying to write a set of values to the database.

Entities

The Database Plugin does not use the CoDaBix® Entity Model because it is configured by an XML Configuration File (`CoDaBix.DatabasePlugin.Settings.xml`) and therefore does not provide entities.

Folders & Files

Folders

Content	Path	Usage
AssemblyFolder	`<CodabixInstallDir>/plugins/DatabasePlugin/`	Contains the plugin's assembly file.
ConfigFolder	`<CodabixDataDir>/plugins/DatabasePlugin/`	Contains the plugin's configuration file.
LoggingFolder	`<CodabixDataDir>/plugins/DatabasePlugin/`	Contains the plugin's log file.

Files

Type	Path	Usage
Assembly	`[AssemblyFolder]/CoDaBix.DatabasePlugin.dll`	The plugin's assembly file.
Config File	`[ConfigFolder]/CoDaBix.DatabasePlugin.Settings.xml`	The config file.
Logging	`[LoggingFolder]/logfile.txt`	The log file.

About Versions

This Document

Date	2018-06-14
Version	1.5

Plugin

Name	Database Plugin
Version	1.0.10

Assembly

Name	CoDaBix.DatabasePlugin.dll
Date	2018-06-14
Version	1.0.10.0