Inhaltsverzeichnis

Database Plugin

Allgemein

Das Database Plugin ermöglicht es Ihnen, CoDaBix® Datenpunktnodewerte in externe Datenbanken wie MySQL oder Microsoft SQL Server zu schreiben.

Was tut das Plugin?

Das Plugin lässt Sie:

- Datenbankverbindungen definieren
- Trigger definieren (z.B. Intervalltrigger)
- DataSets definieren, die bestehen aus
  - Datenpunktnodes
  - Systemwerten

Wenn ein Trigger auslöst, sammelt das Plugin Werte für einen „Datensatz“, indem es einen synchronen Lesevorgang der spezifizierten CoDaBix® Datenpunktnodes durchführt (CoDaBix® fordert das zugrundeliegende Gerät auf, tatsächlich seine Variablen zu lesen) und schreibt dann die Werte des Datensatzes in eine Datenbanktabelle.

Um das Plugin zu benutzen, müssen Sie eine Konfigurationsdatei erstellen.

Funktionen

- Nodewerte sammeln und in eine Datenbank schreiben, wenn ein Trigger auslöst
- Unterstützt verschiedene Triggertypen:
  - Intervalltrigger
  - Flankentrigger (löst aus, wenn der Wert eines Nodes sich in einen festgelegten Wert ändert)
  - Wertänderungstrigger (löst aus, wenn sich der Wert einer Node ändert)

Unterstützte Datenbankmotoren

- MySQL 5.5 oder höher
- Microsoft SQL Server 2008 oder höher

Zweck & Anwendung

- Daten von CoDaBix® auf eine externe Datenbank spiegeln
- Daten von Geräten auf Auslöen eines Triggers einsammeln

Installation

Dieses Plugin ist Bestandteil des CoDaBix® Setups. Bitte konsultieren Sie CoDaBix® Setup und erster Start für weitere Informationen darüber, wie dieses Plugin installiert und deinstalliert werden kann.

Anforderungen

- Die Maschine, die CoDaBix® betreibt, muss Zugriff auf eine der unterstützten Datenbank-Engines haben.

Konfiguration

Dieses Plugin kann nur über die **XML-Konfigurationsdatei** wie unten beschrieben konfiguriert werden. Sie müssen die XML-Konfigurationsdatei („CoDaBix.DatabasePlugin.Settings.xml“) im Projektverzeichnis (siehe Ordner & Dateien) anlegen. Wird die Datei geändert während Codabix läuft, startet das Database Plugin automatisch neu und benutzt die neue Konfigurationsdatei.
Struktur

PluginSettings-Element

Jedes CoDaBix® Plugin definiert die Wurzel seines Elementenbaums über das „PluginSettings“-Element wie folgt:

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

Channels-Element

Das `Channels`-Element dient als Container für ein oder mehrere `Channel`-Elemente. Das `Channel`-Element definiert einer Gruppe von Datenbankverbindungen, Triggern und Datensätzen. Für jeden Kanal gibt es einen Worker-Thread, der einen Trigger durch das Sammeln von Nodewerten und anschließendes Schreiben dieser Werte in spezifizierte Datenbanken verarbeitet.

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

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

Jedes `Channel`-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
`id`	**ja**	String	Identifiziert diesen Kanal eindeutig. Auf diesen Wert kann z.B. in einem der `SystemValue`-Elemente verwiesen werden.
`active`	nein	Boolean	Bestimmt, ob dieser Kanal tatsächlich betrieben wird. Wenn `"false"`, ist der Kanal nicht gestartet, andernfalls schon.

DbConnections-Element

Das `DbConnections`-Element dient als Container für ein oder mehrere `DbConnection`-Elemente.

Das `DbConnection`-Element definiert eine physische Verbindung zu einer Datenbank, inklusive des Tabellennamens. Dies bestimmt, in welche Datenbanktabelle geschrieben wird.

```
<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>
```

Jedes `DbConnection`-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Type	Purpose
`id`	**yes**	String	Identifiziert diese Datenbankverbindung eindeutig. Dies wird in den `DataSet`-Elementen benutzt, um auf diese Verbindung zu verweisen.
`type`	**yes**	Enumeration	Spezifiziert den Typ der Datenbank-Engine. `"MSSQL"`: Microsoft SQL Server `"MySQL"`: MySQL Server
`hostname`	**yes**	String	Der Hostname, der DNS-Name oder die IP-Adresse des Datenbankservers.
`port`	**yes**	Integer	Der TCP-Port des Datenbankservers. Bei der Verwendung von `MSSQL` können Sie `0` als Port angeben, in welchem Fall eine Verbindung zur standardmäßigen SQL-Server-instanz hergestellt werden soll.
`username`	**yes**	String	Der Benutzername, der zum Herstellen der Datenbankverbindung benutzt wird.
`password`	**yes**	String	Das **verschlüsselte Passwort** für den Benutzernamen. Um das verschlüsselte Passwort zu erhalten, öffnen Sie bitte CoDaBix und klicken in der Webkonfiguration auf **Password Security**. Dort können Sie das Originalpasswort eingeben und dieses verschlüsseln, sodass es in dieses Attribut eingefügt werden kann.
`database`	**yes**	String	Der Datenbankname.
`table`	**yes**	String	Der Name der Datenbanktabelle, in die die Werte geschrieben werden sollen.
`connectTimeout`	no	Integer	Der Timeout in Sekunden, der beim Verbinden mit der Datenbank verwendet werden soll. Falls nicht spezifiziert, ist der benutzte Wert 30.
`commandTimeout`	no	Integer	Der Timeout in Sekunden, der beim Ausführen eines Datenbankbefehls verwendet werden soll. Falls nicht spezifiziert, ist der benutzte Wert 60.
`maxPoolSize`	no	Integer	Die maximale Anzahl der gebündelten aktiven Datenbankverbindungen. Falls nicht spezifiziert, ist der benutzte Wert 40.

Beachten Sie: Im Moment schreibt das Database Plugin immer Werte in eine Datenbanktabelle, indem es eine `INSERT`-Anweisung ausführt (bedeutet, dass neue Werte bei jedem Schreibvorgang an die Tabelle angehängt werden).

Triggers-Element

Das `Triggers`-Element dient als Container für ein oder mehrere `Trigger`-Elemente.

Das `Trigger`-Element definiert ein Objekt, das ein Ereignis auslöst. Auf ein Trigger kann in einem DataSet Element verwiesen werden, um festzulegen, wann das Database Plugin Nodewerte einsammeln und in die Datenbank schreiben soll.

```
<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>
```

Jedes `Trigger`-Element stellt folgende Attribute zur Verfügung:

Attribut	**Verpflichtend**	**Typ**	**Zweck**
`id`	**ja**	String	Identifiziert eindeutig den Trigger. Dies wird in den `DataSet` Elementen benutzt, um auf diesen Trigger zu verweisen.
`type`	**ja**	Enumeration	Spezifiziert den Typ des Triggers. `"interval"`: Der Trigger wird im eingestellten Intervall regelmäßig ausgelöst. `"edge"`: Der Trigger löst aus, wenn ein bestimmter Wert in den angegebenen Node geschrieben wurde (während der vorherige Node-Wert ein anderer Wert war). `"valueChange"`: Der Trigger löst aus, wenn sich der Wert des angegebenen Nodes ändert.
`interval` (falls `type="interval"`)	**ja**	Integer	Spezifiziert das Triggerintervall in Millisekunden. Beispielsweise bedeutet ein Wert von `2000`, dass der Trigger alle zwei Sekunden auslöst.
`node` (falls `type="edge"` or `type="valueChange"`)	**ja**	String	Spezifiziert den Nodepfad des Nodes, den der Trigger benutzen soll, um Wertänderungen festzustellen.
`edgeValue` (falls `type="edge"`)	**ja**	String	Der Wert, auf den der Trigger überprüft. Der Trigger löst aus, wenn dieser Wert auf einen Node geschrieben wird und der vorherige Nodewert ein anderer war.

Attribut	Verpflichtend	Typ	Zweck
changeBackValue (falls type="edge")	nein	String	Falls spezifiziert, schreibt das Database Plugin diesen Wert in den Node, nachdem der Trigger ausgelöst hat und der Datensatz über einen synchronen Lesevorgang eingesammelt wurde.
instantChangeBackValue (falls type="edge")	nein	String	Falls spezifiziert, schreibt das Database Plugin diesen Wert in den Node, sofort nachdem der Trigger ausgelöst hat (bevor der Datensatz eingesammelt wurde).

DataSets-Element

Das DataSets-Element dient als Container für ein oder mehrere DataSet Elemente.

Das DataSet-Element definiert eine Gruppe von Datenpunktnodes und SystemValues, die zu einem Datensatz eingesammelt werden und dann auf eine oder mehrere Datenbankverbindungen geschrieben werden. In einem DataSet können Sie auf ein oder mehrere DbConnection-Elemente (bedeutet, dass ein DataSet auf all diese Datenbankverbindungen geschrieben wird) und einen oder mehrere Trigger (bedeutet, dass die Nodewerte gesamment und geschrieben werden, wenn einer dieser Trigger auslöst) verweisen.

```
<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>
```

Jedes DataSet-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
id	**ja**	String	Identifiziert dieses DataSet eindeutig.
writeDelay	nein	Integer	Spezifiziert die maximale Verzögerung (in ms), nachdem die gepufferten Datensätze auf die Datenbank geschrieben werden.

Attribut	Verpflichtend	Typ	Zweck
writeBufSize	nein	Integer	Spezifiziert die maximale Anzahl an gepufferten Datensätzen, bis sie auf die Datenbank geschrieben werden.

Für jedes DataSet, nachdem eine Reihe an Werten eingesammelt wurde (ein Datensatz), wird der Datensatz gepuffert, damit effizient gleich mehrere Datensätze in einem geschrieben werden können. Falls entweder das `writeDelay`- oder das `writeBufSize`-Attribut spezifiziert ist (oder beide), werden die gepufferten Datensätze erst in die Datenbank geschrieben, wenn die `writeDelay` Zeit seit dem letzten Schreibvorgang abgelaufen ist oder die Anzahl der gepufferten Datensätze, die durch `writeBufSize`-Attribut bestimmt wird, überschritten wurde. Wenn keines der Attribute spezifiziert ist, werden die Datensätze sofort geschrieben.

Jedes`DbConnection`-Element und `Trigger`-Element stellt folgende Attribute zur Verfügung (im `DataSet`-Element):

Attribut	Verpflichtend	Typ	Zweck
id	**ja**	String	Verweist auf eine vorher definierte DbConnection oder einen Trigger.

Nodes-Element

Das `Nodes`-Element dient als Containter for ein oder mehrere `Node`-Elemente.

Das `Node`-Element verweist auf einen CoDaBix® Datenpunknode, entweder einen absoluten oder relativen Nodepfad benutzend. Wenn das `Nodes`-Element einen Pfad in seinem `root`-Attribut spezifiziert, ist der `path` der `Node`-Elemente relativ zum `root`-Pfad der `Nodes`.

Beispiel mit root-Pfad:

```
<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>
```

Beispiel ohne root-Pfad:

```
<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>
```

Jedes `Nodes`-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
root	nein	String	Spezifiziert den Pfad zum Parent Node. Falls spezifiziert, ist der Pfad der Node-Elemente relativ zu diesem root-Pfad. Andernfalls muss der Pfad der Node-Elemente der komplette Pfad sein.

Jedes Node-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
path	**ja**	String	Der relative Pfad von `root` zum Node oder der absolute Pfad, wenn der `root` des `Nodes` Elements nicht spezifiziert ist.
column	**ja**	String	Der Name der Datenbanktabellenspalte, in welche der Wert geschrieben werden soll.
property	nein (Standardwert: `value`)	String	Die Eigenschaft des zu schreibenden Nodes. `"value"` (default): Der Nodewert. `"timestamp"`: Der Zeitstempel des Nodewerts (UTC). `"timestampLocal"`: Der Zeitstempel des Nodewerts (local time). `"nodeID"`: Der Identifier (lokaler Identifier) des Nodes. `"nodeName"`: Der Name des Nodes. `"nodeDisplayName"`: Der Anzeigename des Nodes. `"nodeUnit"`: Die Einheit des Nodes.

SystemValues

Das `SystemValues`-Element dient als Container für ein oder mehrere `SystemValue`-Elemente.

Das `SystemValue`-Element enthält entweder einen tatsächlichen Wert order verweist auf einen vordefinierten Systemwert, der in die Datenbank geschrieben werden soll.

```
<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>
```

Jedes `SystemValue`-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
column	**ja**	String	Der Name der Datenbank-Tabellenspalte, in die der Wert geschrieben werden soll.

Attribut	Verpflichtend	Typ	Zweck
type	**ja** (falls value nicht spezifiziert ist)	Aufzählung	Legt fest, welcher Systemwert benutzt werden soll. "triggerTimestamp": Der Zeitstempel, an dem der Trigger ausgelöst hat (UTC). "triggerTimestampLocal": Der Zeitstempel, an dem der Trigger ausgelöst hat (lokale Zeit). "channelID": Der Identifier des enthaltenden Kanals.
value	**ja** (falls type nicht spezifiziert ist)	String	Legt einen direkten (tatsächlichen) Wert fest, der geschrieben werden soll.

Beachten Sie: Entweder das type- oder das value-Attribut muss festgelegt sein, aber nicht beides.

AfterSyncActions-Element

Das AfterSyncActions-Element dient als Container für ein oder mehrere AfterSyncAction-Elemente.

Das AfterSyncAction-Element ermöglicht es, einen Wert auf einen Node zu schreiben, nachdem der Datensatz eingesammelt wurde.

```
<AfterSyncActions>

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

</AfterSyncActions>
```

Jedes AfterSyncAction-Element stellt folgende Attribute zur Verfügung:

Attribut	Verpflichtend	Typ	Zweck
type	**ja**	Aufzählung	Legt den Typ der AfterSyncAction fest. "writeNodeValue": Nachdem der Datensatz eingesammelt wurde, wird der spezifizierte Wert auf den Node geschrieben.
node	**ja**	String	Spezifiziert den Nodepfad des Nodes des zu schreibenden Werts fest.
value	**ja**	String	Der zu schreibende Wert.

Beispiel Konfigurationsdatei

Nachfolgend finden Sie eine beispielhafte Konfigurationsdatei, die die oben beschriebenen Elemente benutzt:

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>
```

Fehlerdiagnose

Das Database Plugin schreibt die folgenden Ereignisse in die Logdatei `[LoggingFolder]\logfile.txt` (siehe Pfad der Logdatei):

- Kanäle werden gestartet oder gestoppt (weil CoDaBix® startet oder stoppt oder die Konfigurationsdatei sich geändert hat und das Plugin neustartet).
- Ein Wertesatz wurde erfolgreich in die Datenbank geschrieben (nachdem ein synchroner Lesevorgang durchgeführt wurde).
- Ein Fehler ist aufgetreten, als versucht wurde, einen Wertesatz in die Datenbank zu

schreiben.

Entities

Das Database Exchange Plugin verwendet das CoDaBix® Entity Modell nicht, da es über eine XML-Konfigurationsdatei (`CoDaBix.DatabasePlugin.Settings.xml`) konfiguriert wird und daher keine Entities zur Verfügung stellt.

Ordner & Dateien

Ordner

Name	Pfad	Zweck / Anwendung
AssemblyFolder	`<CodabixInstallDir>/plugins/DatabasePlugin/`	Beinhaltet die Plugin Assemblydatei.
ConfigFolder	`<CodabixDataDir>/plugins/DatabasePlugin/`	Beinhaltet die Plugin Konfigurationsdatei.
LoggingFolder	`<CodabixDataDir>/plugins/DatabasePlugin/`	Beinhaltet die Plugin Logdatei.

Files

Typ	Pfad	Zweck / Anwendung
Assembly	`[AssemblyFolder]/CoDaBix.DatabasePlugin.dll`	Die Plugin Assemblydatei.
Config File	`[ConfigFolder]/CoDaBix.DatabasePlugin.Settings.xml`	Die Konfigurationsdatei.
Logging	`[LoggingFolder]/logfile.txt`	Die Logdatei.

Versionsinformation

Dieses Dokument

Datum	`2018-06-14`
Version	`1.5`

Plugin

Name	Database Plugin
Version	1.0.10

Assembly

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