

# Inhaltsverzeichnis

<b>Voraussetzungen</b> .....	1
Systemanforderungen .....	1
Erforderliche Plugins .....	1
<b>1. Schritt: Einrichten von CoDaBix</b> .....	2
1.1 Installation .....	2
1.2 Konfiguration .....	2
<b>2. Schritt: Einrichten der Prozessdatenerfassung</b> .....	3
2.1 Anlegen der Prozessdaten .....	3
2.2 Anlegen der RFC-1006 Device Channels .....	3
2.3 Deserialisierung der XML-Daten .....	6
<b>2.4 Alternative 1 - Daten in die Datenbank schreiben mithilfe vom DB Plugin</b> .....	10
<b>2.5 Alternative 2 - Daten in die Datenbank schreiben mithilfe vom SQL Exchange Plugin</b> .....	11



# RFC 1006 empfangene XML-Datei in SQL Datenbank speichern

Diese Anleitung beschreibt das „Schritt für Schritt“-Vorgehen, um eine per RFC 1006 empfangene XML-Datei nach dem Empfang in eine Datenbank zu schreiben.

Als Datenquelle wird hierbei beispielhaft der **RFC 1006 Device Plugin** und als Datenziel eine lokale **MySQL Datenbank** verwendet.

Durch das einheitliche Interface, das CoDaBix für den Zugriff auf angebundene Geräte und Datenbanken bietet, ist dieses Vorgehen jedoch auf **jede Art von Daten**, die in CoDaBix definiert sind, anwendbar.

## Voraussetzungen

### Systemanforderungen

Für die Ausführung der folgenden Schritte benötigen Sie einen Rechner (mit Windows oder Linux Betriebssystem), auf dem Sie die erforderlichen Rechte besitzen, um Anwendungen installieren zu können.

Desweiteren ist eine Internetverbindung notwendig, um CoDaBix und die bereitgestellte Standardkonfiguration zu laden.

Die genauen Systemanforderungen an Hardware und Betriebssystemversion finden Sie hier:

- [Systemanforderungen für Windows](#)
- [Systemanforderungen für Linux](#)

### Erforderliche Plugins

Folgende CoDaBix Plugins sind erforderlich

- RFC 1006 Device Plugin: Anbindung der RFC 1006 Verbindung
- [SQL Exchange Plugin](#): Verbindung zur SQLite Datenbank
- [Script Interface Plugin](#): Verarbeiten der empfangenen Daten und des Triggers und Transfer der Daten

# 1. Schritt: Einrichten von CoDaBix

In diesem Schritt installieren Sie CoDaBix auf Ihrem System.

Für diese Anleitung stellen wir Ihnen eine Grundkonfiguration für die Prozessdatenerfassung bereit.


## 1.1 Installation

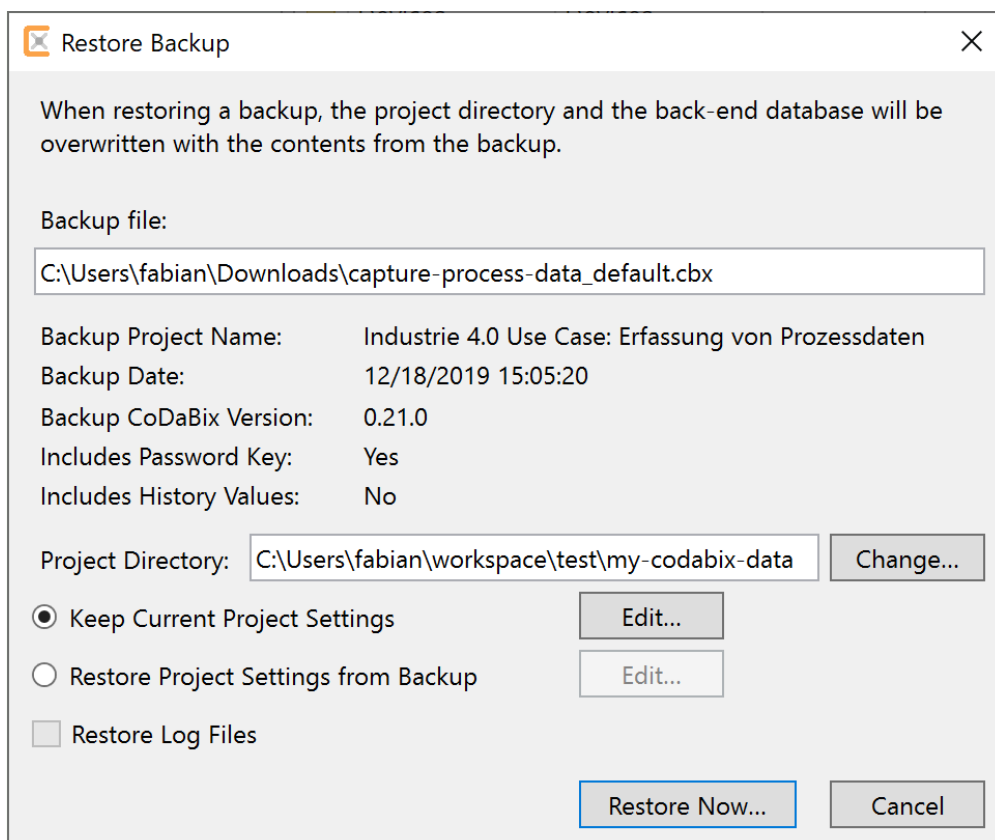
- Laden Sie im **Downloadbereich** die aktuelle Version von CoDaBix (diese Anleitung setzt **mindestens v0.21.0** voraus) für das System herunter,

auf dem Sie CoDaBix verwenden wollen.

- Nach erfolgreichem Download installieren Sie CoDaBix auf Ihrem System und folgen den Schritten für den ersten Start:
  - **Installation und erster Start unter Windows**
  - **Installation und erster Start unter Linux und Raspberry Pi**

## 1.2 Konfiguration

- Laden Sie folgende Standardkonfiguration herunter: [capture-process-data\\_default.cbx](#)
- Importieren Sie die Konfiguration in CoDaBix
  - Windows:
    - Klicken Sie auf  rechts in der Taskleiste
    - Wählen Sie im Backup-Dialog die zuvor geladene Datei aus und starten den Import durch Klicken auf **Restore Now...**



- Linux/Raspberry Pi:
  - Wählen Sie in der Konsolenanwendung den Menüpunkt 4) Restore Backup
  - Geben Sie den Pfad zu der geladenen Datei an und starten den Import durch drücken der Enter-Taste

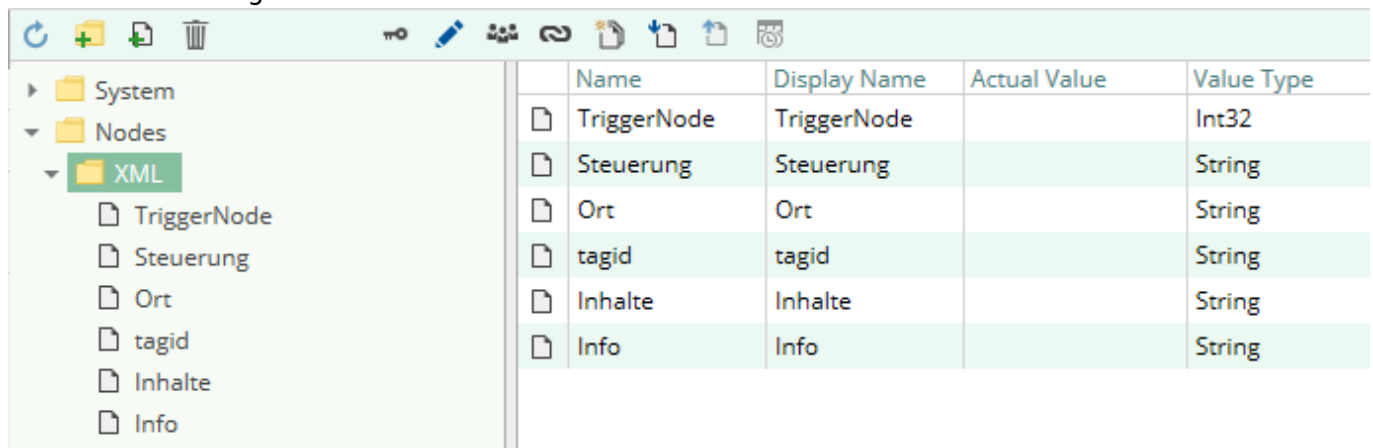
## 2. Schritt: Einrichten der Prozessdatenerfassung

### 2.1 Anlegen der Prozessdaten

Im „Nodes“ Verzeichnis wird eine Folder-Node XML erstellt.  
Dort erstellen wir folgende Nodestruktur:

- TriggerNode
  - Datentyp: Int32
  - Auslöser für das Schreiben der Daten in die Datenbank
- Steuerung
  - Datentyp: String
- Ort
  - Datentyp: String
- tagid
  - Datentyp: String
- Inhalte
  - Datentyp: String
- Info
  - Datentyp: String

Das sieht dann folgendermaßen aus:

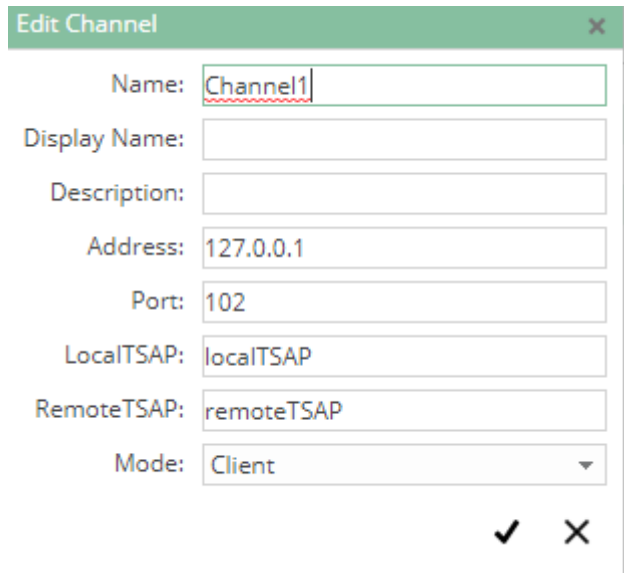


Name	Display Name	Actual Value	Value Type
TriggerNode	TriggerNode		Int32
Steuerung	Steuerung		String
Ort	Ort		String
tagid	tagid		String
Inhalte	Inhalte		String
Info	Info		String

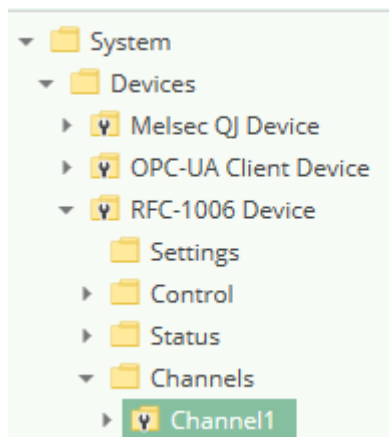
### 2.2 Anlegen der RFC-1006 Device Channels

#### Client Channel anlegen

Unter System → Devices → RFC-1006 Device legen Sie einen neuen Channel mit dem Namen Channel1 und folgenden Einstellungen an.



Hier verwenden wir die localhost IP-Adresse 127.0.0.1, da ein weiterer Channel angelegt wird, welcher als Testserver dient.



Dann können Sie wie folgt vorgehen, um aus dem Rfc1006-Channel1 ein XML-Telegramm zu empfangen und dieses in die Nodes zu schreiben, welche dann wiederum in eine Datenbank geschrieben werden können:

Stellen Sie bitte im Rfc1006-Channel im Ordner „Nodes“ den Value Type der „ReceiveBuffer“-Node von Blob auf „String“ um

Name	Display Name	Actual Value	Value Type
ReceiveBuffer	ReceiveBuffer		Blob
TransmitBuffer	TransmitBuffer		Blob

**Edit Variable**

Name:

Display Name:

Value Type: **Blob** (dropdown menu open showing: String, Blob, Boolean, Byte, SByte, Int16, UInt16, Int32)

Description:

Location:

Path:

Min Value:

Max Value:

Max:

### Testserver Channel anlegen

Nun legen wir noch den Channel für den Testserver mit dem Namen TestServer an. Hier werden die LocalTSAP und RemoteTSAP vertauscht um mit dem Client kommunizieren zu können.

**Edit Channel**

Name:

Display Name:

Description:

Address:

Port:

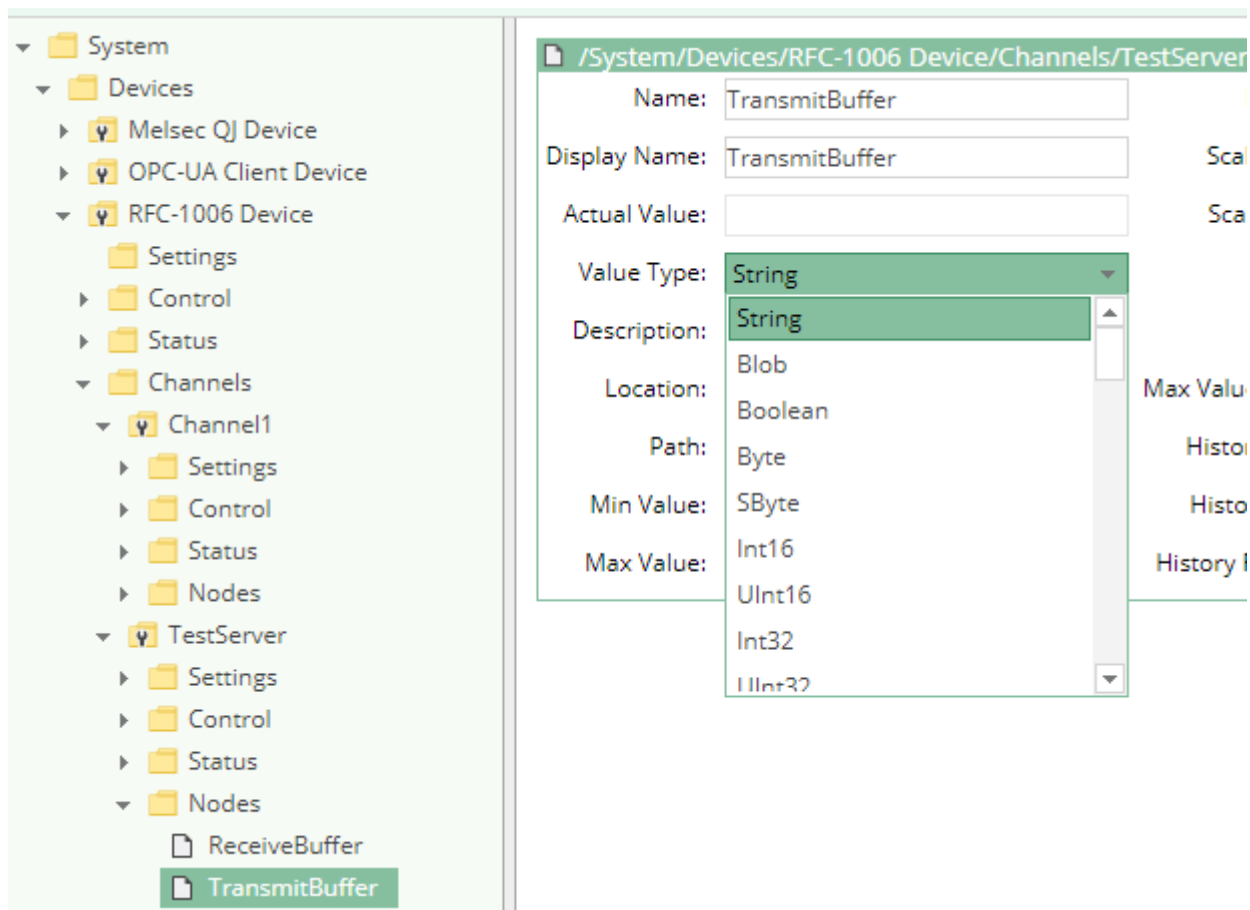
LocalTSAP:

RemoteTSAP:

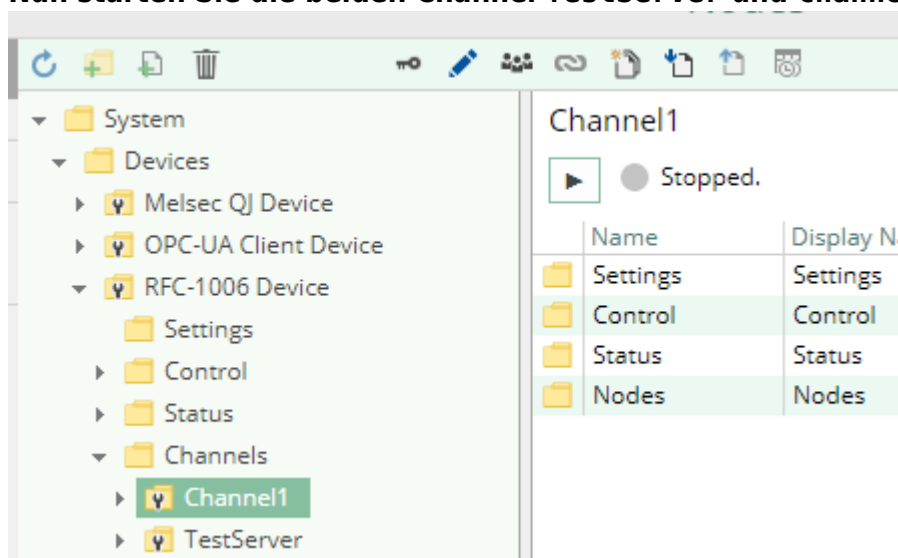
Mode: **Server** (dropdown menu)

✓ ✕

Stellen Sie bitte im Rfc1006-Channel im Ordner „Nodes“ den Value Type der „TransmitBuffer“-Node von Blob auf „String“ um



**Nun starten Sie die beiden Channel TestServer und Channel1**



## 2.3 Deserialisierung der XML-Daten

- Nun wechseln Sie in das Script Interface und erstellen ein neues Script. im Beispiel ist das RFC1006 Deserialisation
- Nach dem Erstellen öffnen Sie den Script-Editor durch Klick auf das Icon „</>“ (oder Rechtsklick auf die Zeile und dann „Script Editor“ wählen).
- Im Script-Editor ersetzen Sie bitte den vorhandenen Code durch den Code mit folgendem Script:

Bitte beachten Sie, dass das Script als Beispiel noch nicht alle String-Werte korrekt dekodiert (z.b. wenn mit „&“ kodierte Zeichen vorkommen). Wir arbeiten jedoch aktuell an der Implementierung



eines separaten XML-Plugins für CoDaBix, dass das Parsen von XML-Dateien übernimmt, sodass alle Stringwerte korrekt ausgelesen werden können.

### RFC1006-samplescript.ts

```
runtime.handleAsync(async function () {

    const rfc1006ChannelReceiveBufferPath = "/System/Devices/RFC-1006  
Device/Channels/Channel1/Nodes/ReceiveBuffer";
    const rfc1006ChannelReceiveBufferNode =  
codabix.findNode(rfc1006ChannelReceiveBufferPath);
    if (!rfc1006ChannelReceiveBufferNode)
        throw new Error(`Could not find node  
'${rfc1006ChannelReceiveBufferPath}'.`);

    const tagBaseNodePath = "/Nodes/XML";
    const triggerNodePath = tagBaseNodePath + "/TriggerNode";
    const triggerNode = codabix.findNode(triggerNodePath);
    if (!triggerNode)
        throw new Error(`Could not find node '${triggerNodePath}'.`);

    const tagNames = ["Steuerung", "Ort", "tagid", "Inhalte", "Info"];

    // Find the nodes for each tag name.
    const tagNodes = new Map<string, codabix.Node>();
    for (let tagName of tagNames) {
        const nodePath = tagBaseNodePath + "/" + tagName;
        const node = codabix.findNode(nodePath);
        if (!node)
            throw new Error(`Could not find node '${nodePath}'.`);

        tagNodes.set(tagName, node);
    }

    // Reset the trigger node.
    void codabix.writeNodeValueAsync(triggerNode, 0);

    // Add a value changed event listener to the Receive Buffer.
    rfc1006ChannelReceiveBufferNode.addValueChangedEventListener(e => {
        if (typeof e.newValue.value !== "string") {
            logger.logError(`Value Type of node  
'${rfc1006ChannelReceiveBufferPath}' is not 'String'!`);
        }
        else {
            let stringValue = e.newValue.value;

            // Log the received XML.
            logger.log("Received: " + stringValue);

            // Find the tags.
```

```

    let tags = extractTagsFromXmlString(stringValue);

    // Now write the tags into their corresponding nodes.
    codabix.scheduleCallback(() => {
        let valuesToWrite: {
            node: codabix.Node,
            value: string | number
        }[] = [];
        tags.forEach((value, key) => valuesToWrite.push({
            node: tagNodes.get(key)!,
            value
        }));

        // Also, write 1 to the Trigger Node.
        valuesToWrite.push({
            node: triggerNode,
            value: 1
        });

        // Write the values.
        codabix.writeNodeValuesAsync(valuesToWrite);

        // After writing the values, switch the trigger node
back to 0.
        codabix.writeNodeValueAsync(triggerNode, 0);
    });
}

});

function extractTagsFromXmlString(xmlString: string): Map<string,
string> {
    let map = new Map<string, string>();

    for (let tagName of tagNames) {
        // Search for the start tag.
        let startTag = `<${tagName}>`;
        let endTag = `</${tagName}>`;

        let startTagIndex = xmlString.indexOf(startTag);
        if (startTagIndex >= 0) {
            // OK, we found the start tag. Now search for the end
tag.
            let endTagIndex = xmlString.indexOf(endTag,
startTagIndex + startTag.length);
            if (endTagIndex >= 0) {
                // OK, we now have the value.
                // TODO: To get the correct string value, we would
need to XML-decode the string.
                let content = xmlString.substring(startTagIndex +
startTag.length, endTagIndex);
                logger.log(`Found Tag '${tagName}' with value

```


```

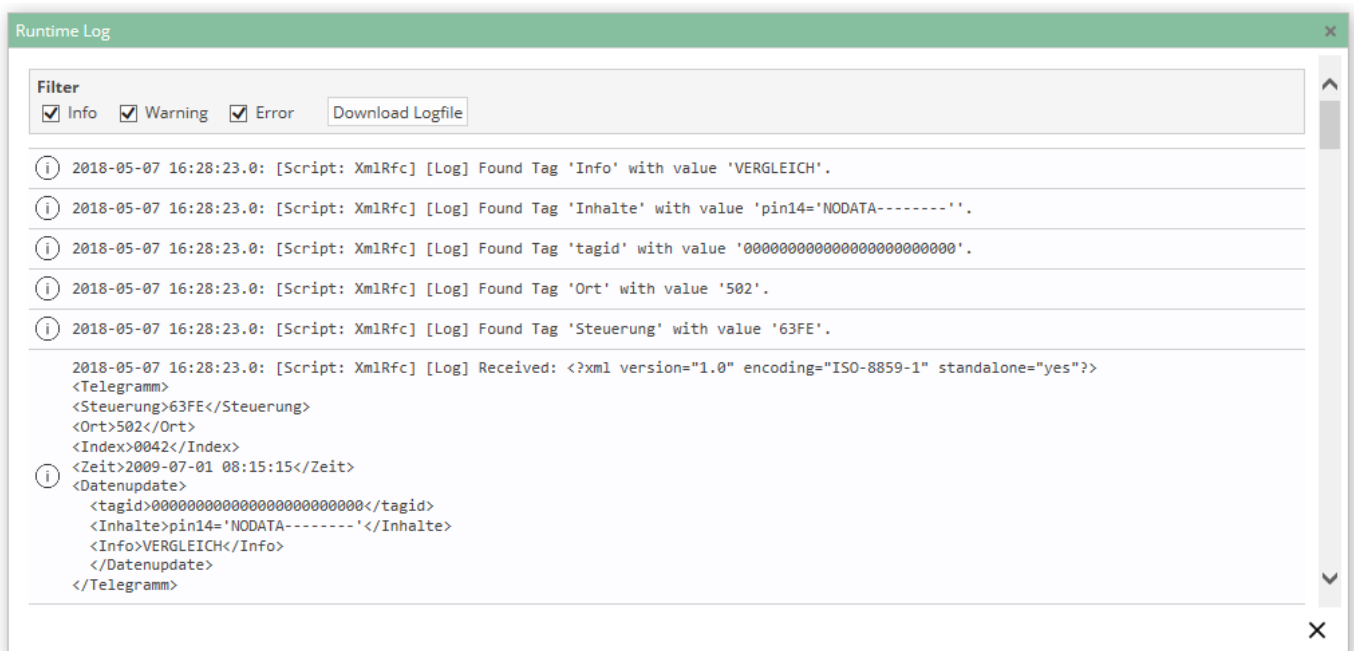
'${content}'.`);
        map.set(tagName, content);
    }
    else {
        logger.logWarning(`Could not find start tag
'${startTag}'.`);
    }
}
else {
    logger.logWarning(`Could not find start tag
'${startTag}'.`);
}
}

return map;
}

}());

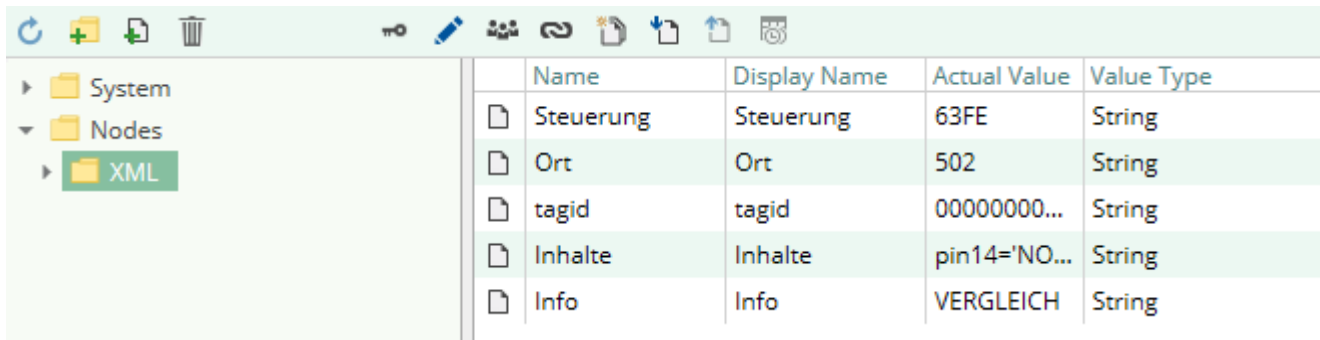
```

- Setzen Sie einen Haken bei „Go Live“ und klicken dann auf den Speichern-Button.
- Wenn nun das Rfc1006-Plugin ein XML-Telegramm empfängt, sollte im CoDaBix-Log (dies kann rechts oben über den -Button geöffnet werden, setzen Sie dann dort noch einen Haken bei „Info“) das XML-Telegramm mit den Tags angezeigt werden, z.B.:



(Falls ein Fehler geloggt wird, dass eine Node nicht gefunden wurde, müssen Sie ggf. den Pfad zur Node im Script-Code anpassen.)

Nun sollten auch die Werte in den zuvor erstellen Nodes stehen:



	Name	Display Name	Actual Value	Value Type
Steuerung	Steuerung	Steuerung	63FE	String
Ort	Ort	Ort	502	String
tagid	tagid	tagid	00000000...	String
Inhalte	Inhalte	Inhalte	pin14='NO...	String
Info	Info	Info	VERGLEICH	String

## 2.4 Alternative 1 - Daten in die Datenbank schreiben mithilfe vom DB Plugin

Um nun jedes Mal wenn neue Werte geschrieben wurden, die Tags in eine Datenbank zu schreiben, können Sie z.B. unser Database-Plugin verwenden. Dazu können Sie folgende Konfiguration in der XML-Konfigurationsdatei (CoDaBix.DatabasePlugin.Settings.xml) angeben:

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="MySQL"
          hostname="localhost" port="3306"
          username="root" password="xxxxx"
          database="mydatabase" table="mytable" />
      </DbConnections>

      <Triggers>
        <Trigger id="t1" type="edge" node="/Nodes/XML/TriggerNode"
          edgeValue="1" />
      </Triggers>

      <DataSets>
        <DataSet id="ds1" writeDelay="1000" writeBufSize="10">

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

          <Nodes root="/Nodes/XML/">
            <Node path="Steuerung" column="Steuerung" />
            <Node path="Ort" column="Ort" />
            <Node path="tagid" column="tagid" />
            <Node path="Inhalte" column="Inhalte" />
            <Node path="Info" column="Info" />
          </Nodes>
        </DataSet>
      </DataSets>
    </Channel>
  </Channels>
</PluginSettings>
```

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

Dadurch würde nun immer wenn ein neues XML-Telegramm eintrifft, dieses in die Datenbank geschrieben werden.

## 2.5 Alternative 2 - Daten in die Datenbank schreiben mithilfe vom SQL Exchange Plugin

From:

<https://codabix.de/> - CoDaBix®

Permanent link:

<https://codabix.de/de/tutorials/rfc1006-to-sql>

Last update: **2021/04/30 12:59**