Table of Contents

Plugins

Different Plugins are available for CoDaBix® which are grouped by their classification. Depending on its classification the plugin provides a specialized set of services, entities and Nodes. Some plugins can provide configuration files and additionally, as the circumstances require, a configuration application.

Available Plugins

Device Plugins

- General
- AKLAN Device Plugin
- Allen-Bradley Device Plugin
- EUROMAP Device Plugin
- H1 Device Plugin
- Melsec Device Plugin
- Modbus Device Plugin
- Omron Device Plugin
- OPC UA Client Device Plugin
- S7 Device Plugin
- MQTT Device Plugin
- UniPi Device Plugin

Exchange Plugins

- General
- CSV Exchange Plugin
- Database Plugin
- SQL Exchange Plugin
- XML Exchange Plugin

Interface Plugins

- General
- OPC UA Server Interface Plugin
- REST Interface Plugin
- Script Interface Plugin

Activation

During the startup progress of CoDaBix® all plugins located in the plugins directory (<CodabixPluginsDir>) are recursively run through, loaded, instantiated and started in alphabetical order. Each plugin which is not in the plugin directory or in any sub directory is neither loaded nor started by the CoDaBix® Plugin Manager.

While the CoDaBix® Engine is started, it is possible to restart a plugin. This is done by sending a request to the CoDaBix® Plugin Manager using the plugin's proxy instance (note that this function makes modifications without preannouncement).

State Machine

During the startup progress and during the runtime of CoDaBix® the plugin can run through different states. Each state does restrict the possible transition from one state to the other while some of them occur consecutively. The plugin instance itself stays in the state Created after the CoDaBix® Plugin Manager has loaded and instantiated the plugin.

After the Plugin Manager has loaded and instantiated all plugins the start sequence calls the startup sequence of the plugins. During the startup the state is changed to Starting. After a successful start of the plugin the state is changed to Started. If the plugin cannot be strted,the state is set back to the value it held before Starting. Only plugins with the state Started can be stopped.

A plugin is stopped during the restart sequence or during the shutdown process of the CoDaBix® Plugin Manager. The plugin then processes its stop sequence (only if its state is Started). During this sequence the plugin state changes to Stopping. If the shutdown of the plugin fails, the state is set to Stopped. If the plugin cannot be stopped, it goes back to the state it had before being changed to Stopping.

Diagnostics

The provided diagnostic information of a plugin depends on the classification of the plugin and whether the plugin provides the specific diagnostic information itself. The S7 Device Plugin e.g. is a Device Plugin and does provide status information and other diagnostic information through its CoDaBix® Device, CoDaBix® Device Channel and S7 Device Variables.

Location

All plugin specific files are stored in a plugin directory in the `plugins` directory, which is located in the CoDaBix® installation directory (selected during the installation). The following list illustrates the hierarchy:

- `<CodabixInstallDir>`
  - `plugins\`
    - `<PluginName>`
      - `<PluginAssembly>`
      - …

This can look like as follows:

- `C:\Program Files\Traeger\CoDaBix\`
  - `plugins\`
    - `S7Device\`
      - `CoDaBix.S7DevicePlugin.dll`
      - …

Classification

Device Plugins

All plugins classified as Device Plugin integrate physical or logical devices in CoDaBix®.

A device itself can be any kind of resource made accessible for CoDaBix® by the channel model defined by the CoDaBix® Device Model.

In general a device plugin defines a specified type of CoDaBix® Device using the CoDaBix® Device Model (for more information see Device Plugins) and therefore provides the necessary accessibility layer for such a device.

Exchange Plugins

All plugins classified as Exchange Plugin link storage engines with CoDaBix®.

A storage engine itself can be any kind of database management system (DBMS) of which the instances are accessible by the relational model using the CoDaBix Storage Model.

The storage engine itself has to provide at least one primitive relational model like e.g. a table (= entity in DBMS).

In general an Exchange Plugin defines a specified type of CoDaBix® Storage using the CodaBix Storage Model (for more information see Exchange Plugins) and therefore provides the necessary accessibility layer for such a storage engine.

Interface Plugins

All plugins classified as Interface Plugin link CoDaBix® with other platforms or technologies. The interface itself can be any kind of language, service, protocol, etc. The API model only has to depict the special environment onto the API defined by CoDaBix®.

In general an Interface Plugin defines the a speciialized type of platform or technology API (for more information see Interface Plugins) and therefore provides the necessary accessibility layer for such a platform or technology to the CoDaBix® API.

Configuration

Using an Application

Location:

In case the plugin provides an application for configuration it can be located in the `<CodabixInstallDir>` directory). The following list illustrates the hierarchy:

- `<CodabixInstallDir>`
  - `<PluginConfigurationAppName>`
  - **…**

For example:

- `C:\Program Files\Traeger\CoDaBix\`
  - `CoDaBix.S7DevicePlugin.Configurator.exe`
  - **…**

Using a Configuration File

Location

In case the plugin uses a configuration file it can be located in the plugin directory within the CoDaBix project directory (configured in CoDaBix). The following list illustrates the hierarchy:

- `<CodabixProjectDir>`
  - `plugins\`
    - `<PluginConfigurationFileName>`

        - ...

For example:

- D:\Data\Traeger\CoDaBix.Data\
  - CoDaBix.S7DevicePlugin.Settings.xml
  - [CoDaBix.S7DevicePlugin.Settings.xsd]
  - **...**

Structure

An XML based plugin configuration file defines at least the `PluginSettings` element, while the further defined elements depend on the classification of the plugin and their own custom elements and attributes.

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

Entity Attributes

Some elements in the configuration file create or represent an entity in CoDaBix®. For such elements some attributes are eserved in order to service those entities easier.

The Entitiy `Identifier` Attribute
This attribute is of the type GUID. This ID is produced while producing the entity of CoDaBix®. Through this identifier (= in CoDaBix® the Global Node Identifier) it is possible to uniquely identify the entity and assign it uniquely to the configuration element.

In case a configuration element is missing or the element / attribute to be identified cannot be found, it is assumed that a new entity shall be created. If a new configuration element shall be created, it is not necessary to define the `Identifier` attribute. If an existing element shall be modified ,the `Identifier` attribute must be defined.

The Entity ChangeType Attribute
The following values are valid for attributes of that type:

Value	Description
“None”	The entity configuration element hasn't been changed recursively.
“Created”	The entity configuration element is new and requires a new entity to be created.
“Updated”	The entity configuration element was changed and requires the entity depicted to be updated.
“Deleted”	The entity configuration element is obsolete and requires the entity represented to be deleted.

In case there the attribute has the value:

- “None”: Nothing special is to be done and the value itself is ignored. Then the plugin evaluates the attribute.
- “Created”: The `Identifier` attribute is used (if set) to find the entity. If the entity is

not found, a new entity is created. The Global Node Identifier of the entity is added to the `Identifier` attribute. Then the plugin evaluates the attribute.
- “`Updated`”: The `Identifier` attribute is used to find the entity. If the entity is not found, a new entity is created. The transferred value is updated. Then the plugin evaluates the attribute.
- “`Deleted`”: The `Identifier` attribute is used to find the entity. If the entity exists it is deleted. Then the pluginchecks if the attribute was removed.

Synchronization

As soon as a plugin is loaded and started by the CoDaBix® Plugin Manager its configuration file (if available) is read and synced by the plugin with the appropriate CoDaBix® entities.

In case the configuration file changes the plugin manager notifies the according plugin. The plugin decides, if a restart or a synchronisation of the according configuration entities happens.

f at least one of configuration entitiy changes, the plugin has to handle the entity change event and synchronize the entity / entities with the configuration file.