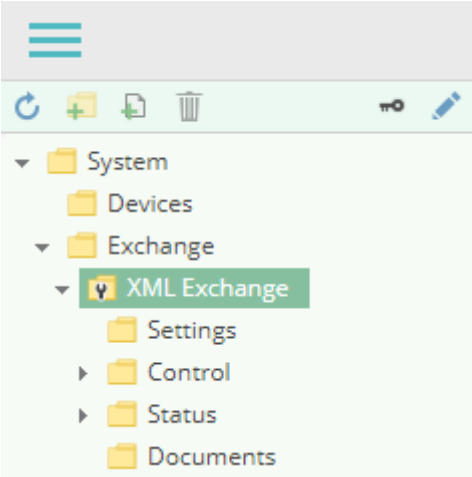
Table of Contents

XML Exchange Plugin

The XML Exchange Plugin allows reading and writing values in XML files (with a fixed structure).

Configuration

The whole XML Exchange Plugin configuration is located in the node path `/System/Exchange/XML Exchange`.



Document

An XML Exchange Document (similar to a Channel for device plugins) represents an XML file.

Settings

File Path

The path to the XML file in the file system.

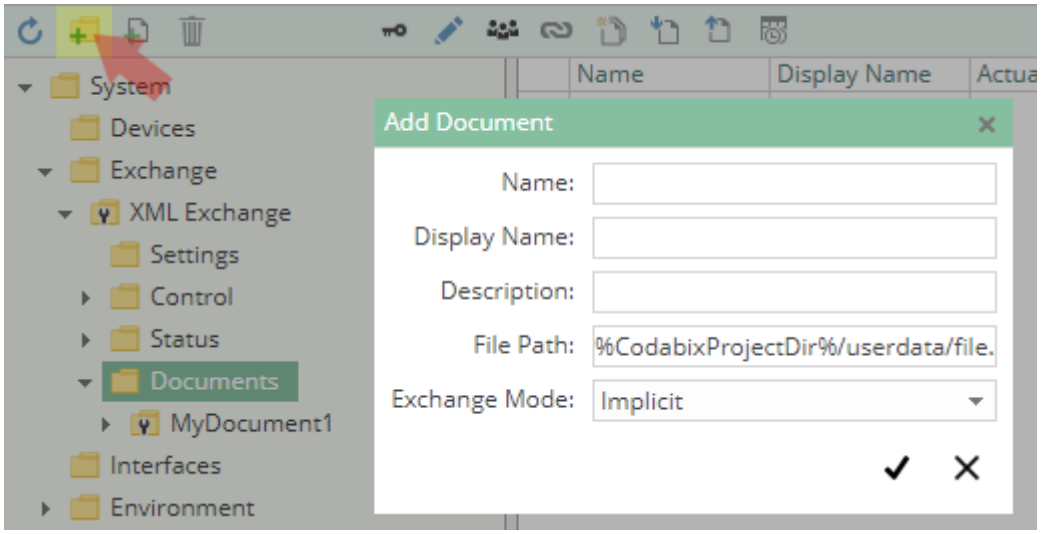
Exchange Mode

Specifies how the XML file is to be accessed:

- **Implicit**: The XML file will be read once one or more variables are read; and the XML file will be written once one or more variables are written.
- **Explicit**: No file access occurs when reading or writing variables. The file needs to be read or written explicitely, by calling the `Load` or `Save` method in order to transfer

the file content to the variables and vice versa.

Adding a Document (Channel)



To add a new **XML Document** (which corresponds to a **Channel** for device plugins), please follow these steps:

1. Add a Folder Node within the node XML `Exchange/Documents`, or right-click on the XML `Exchange/Documents` node and select `Add Document`.
2. In the `Add Document` dialog, specify the settings for the XML file.
3. After clicking on “Save”, the document node is created.
4. You can start the channel by selecting the document node and clicking the start button.

Variables

Within the `DocumentElement` node you can create variables (nodes) which correspond to the structure of the XML document (which means the tree structure of the CoDaBix variables corresponds to the structure of the XML nodes):

- A Folder Node variable maps to an **XML element**, where the `Path` property contains the name of the element; or, if it is empty, the name of the variable will be used as name for the XML element.
- A Datapoint Node variable maps to an **XML attribute** or **XML text node** which can be read or written as `String`. The `Path` property can contain the following expressions:
  - `text()`: The variable maps to a **XML text node**.
  - Does **not** start with `/` (e.g. `abc`): The variable maps to an **XML attribute** of the parent XML element with the specified name (in the example,`"abc"`).
  - Starts with `/` (e.g. `/*/A/B/text()[2]`): The variable represents an **XPath query**, which either returns a text as result (which means it cannot be written to), or an XML node set where then the first node is used (must be a text node, attribute node or comment node). **Note:** When using an XPath query, the position of the variable within the node tree will not be considered.

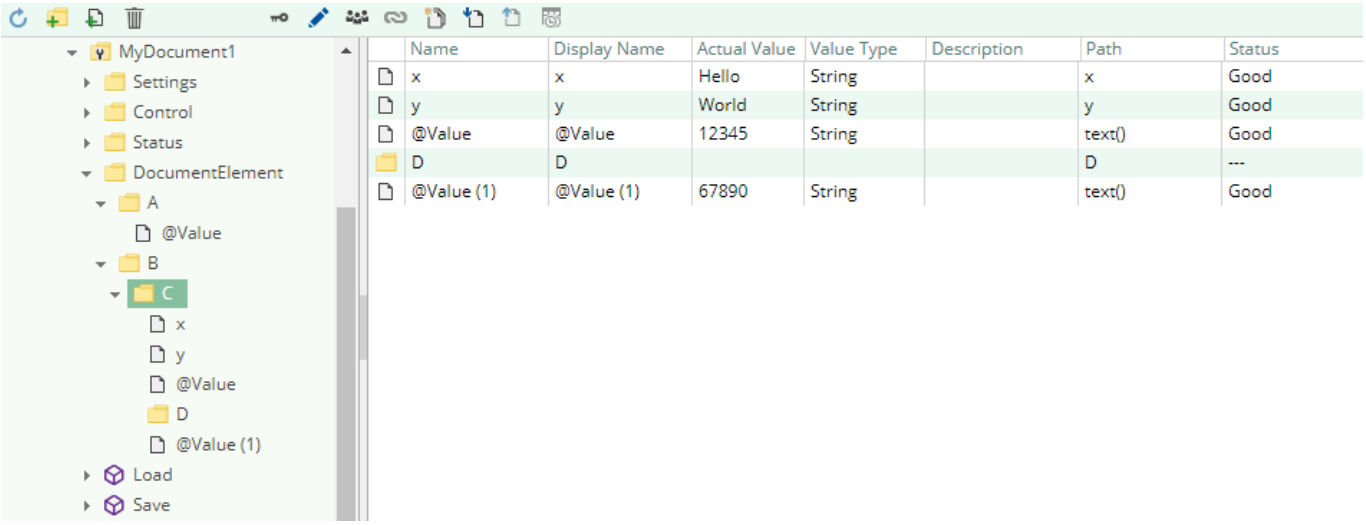
When the XML file already exists, you can browse the Document (Channel) to automatically create the CoDaBix variables for the XML document structure.

Example

example.xml

```
<Root>
  <A>Test</A>
  <B>
    <C x="Hello" y="World">
      12345
      <D />
      67890
    </C>
  </B>
</Root>
```

After browsing the document, the following node structure is created in CoDaBix:



Read/Write

For read operations as well as for write operations, the CoDaBix variables will be mapped to XML nodes by using their position within the node tree. When you write values and there are variables in CoDaBix which cannot be found in the XML file (which also happens e.g. if the XML file doesn't exist yet), the corresponding nodes will be created in the XML file.

Read Behavior (depending on the set **Exchange Mode** in the settings).

- Exchange Mode **“Implicit”**:
  - Synchronous read of one or more variables:
    - The XML file is read completely.
    - The XML nodes are mapped to the existing CoDaBix variables using the node tree

structure.
        - Values are written only in those CoDaBix variables which were part of the synchronous read operation.
            - If reading the XML file fails, a value with status Bad is written to the variables.
    - Calling the Load method:
        - An error is raised because the method can only be called in **Explicit** mode.
- Exchange Mode **“Explicit”**:
    - Synchronous read of one or more variables:
        - No file access occurs; the current values of the read variables are returned as result of the synchronous read operation.
    - Calling the Load method:
        - The XML file is read completely.
        - The XML nodes are mapped to the existing CoDaBix variables using the node tree structure.
        - Values that were read from the corresponding XML nodes are written into all CoDaBix variables.
            - If reading the XML file fails, nothing is written into the variables, but the method raises an error.

Write Behavior:

- Exchange Mode **“Implicit”**:
    - Writing values in one or more variables:
        - The XML file is read completely.
        - The XML nodes are mapped to the existing CoDaBix variables using the node tree structure.
          If a XML node cannot be found for an existing CoDaBix variable, a new node is created in the XML file (even when that variable isn't part of the write operation), but initially without a value (text).
        - Values are inserted only in those XML nodes whose corresponding CoDaBix variables were part of the write operation.
        - The XML file is written.
    - Calling the Save method:
        - An error is raised because the method can only be called in **Explicit** mode.
- Exchange Mode **“Explicit”**:
    - Writing values in one or more variables:
        - No file access occurs; the status Good is used as result of the write operation.
    - Calling the Save method:
        - The XML file is read completely.
        - The XML nodes are mapped to the existing CoDaBix variables using the node tree structure.
          If a XML node cannot be found for an existing CoDaBix variable, a new node is created in the XML file, but initially without a value (text).
        - The current values of the CoDaBix variables are inserted into all XML nodes.
        - The XML file is written.

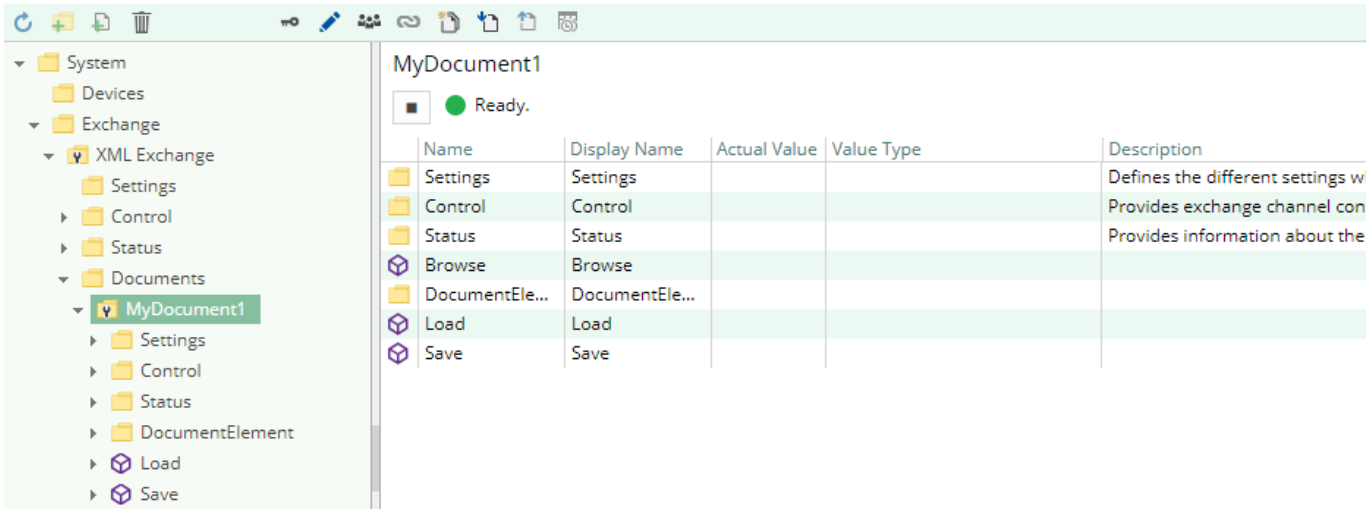
Diagnostics

The XML Exchange Plugin provides different status information depending on the layer to inspect. In general the document-based diagnostic information is produced by the access status of the XML file.

The variable-based diagnostic information is produced during the read/write access of the different variables.

Channel

To monitor and diagnose the status of a XML Document (Channel), take a look at the following image:



The image above depicts the XML Document's Control Panel which displays all status relevant information. The control panel will automatically update its status information when a new status is available.

Status Circle

Color	Meaning
	The channel is stopped. Click the ► button to start it.
	The channel is currently in the progress of starting or stopping or is waiting to establish a connection.
	The channel is ready for doing read/write operations. You can stop it by clicking the ■ button.
	The channel is running, but the connection is currently in an error state. Please check the status text for more information.

Variables

To monitor and diagnose the status of the different XML variables, take a look at the variable's `Status` property displayed in CoDaBix. If the Exchange Mode is set to **“Implicit”**, use the button “Read actual Value” to read the values from the PLC and store the result into the variables. Otherwise, you can call the `Load` method to verify if the file can be read correctly.

Name	Display Name	Actual Value	Value Type	Description	Path	Status
x	x		String		x	Bad: Name cannot begin with the '<' character, hexadecimal value 0x3C. Line 2, position 1.
y	y		String		y	Bad: Name cannot begin with the '<' character, hexadecimal value 0x3C. Line 2, position 1.
D	D				ns2:D	---

Log file

All channel related status information is also logged into the channel-specific log file stored in the `[LoggingFolder]`. Each log file is named in the naming scheme `XML Exchange.<ChannelName>.log`.

The content of such a log file can look as follows:

```
...
2018-04-11 11:32:37.0 +2: [Error] Error (Severity=High): Code=[-1],
Text=[The operation has timed-out.], Details=[]
...
```

Entities

Like every exchange plugin the XML Exchange Plugin extends the basic CoDaBix Exchange Model.

Exchange

The plugin's exchange type `XmlExchange` also defines the `XmlExchangeDocument` and therefore extends the basic `CodabixExchange` and `CodabixExchangeChannel` entities. While the `XmlExchange` just represents a concretization of the `CodabixExchange`, the `XmlExchangeDocument` extends the `CodabixExchangeChannel` with the XML Variable Entities.

Channel

Each channel is handled by a channel worker which access an XML file using file system operations.

By default, the worker does not read any values. When the Exchange Mode is set to `Implicit` and a client or plugin requests a synchronous read of the channel's variables in CoDaBix (e.g. using the CoDaBix Web Configuration's function “Read actual value”), the channel worker reads them from the underlying PLC and then writes them into the corresponding CoDaBix Nodes.
If the Exchange Mode is set to `Explicit`, the worker reads values as soon as the `Load` method of the channel is called.

Similarly, the channel worker will write values to file when a client or plugin writes values into the channel's variables (“Implicit”) or when the channel's `Save` method is called (“Explicit”).

To have an XML Exchange Variable being read steadily (when using mode “Implicit”), you can edit the Node in the CoDaBix Web Configuration and set “History Options” to `Yes` (which will create an internal subscription), or you can use e.g. a OPC UA Client connected to the OPC UA Server plugin and create a subscription for the XML variable nodes. In these cases, the channel worker reads the variables from the XML file at a regular interval and, if the value of one of the variables has changed, writes the new value into the corresponding CoDaBix nodes.

Folders & Files

Folders

Content	Path	Usage
AssemblyFolder	<CodabixInstallDir>/plugins/XmlExchangePlugin/	Contains the plugin's assembly file.
ConfigFolder	<CodabixProjectDir>/plugins/XmlExchangePlugin/	Contains the plugin's configuration file.
LoggingFolder	<CodabixProjectDir>/log/	Contains the plugin's log files.

Files

Type	Path	Usage
Assembly	[AssemblyFolder]/CoDaBix.XmlExchangePlugin.dll	The plugin's assembly file.
Logging	[LoggingFolder]/XML Exchange.<ChannelName>.log	The log file.

About Versions

This Document

Date	2020-02-06
Version	1.0

Plugin

Name	XML Exchange Plugin
Node	/System/Exchange/XML Exchange
Version	1.0.0

Assembly

Name	CoDaBix.XmlExchangePlugin.dll
Date	2020-02-06
Version	1.0.0.0