rexroth
A Bosch Company

Application Manual

Key Value Database App

Managing Data in the Data Layer 03VRS

R911425250 Edition 01

Disclaimer

The data specified above only serve to describe the product. As our products are constantly being further developed, no statements concerning a certain condition or suitability for a certain application can be derived from our information. The information given does not release the user from the obligation of own judgment and verification. It must be remembered that our products are subject to a natural process of wear and aging.

e56f864820ef09670a347e8851147945, 1, en_US

Table of contents

1 About this documentation

Editions of this documentation

Edition	Date	Note
01	2024 – 11	First edition for version KVD-V-0302

2 Important directions on use

2.1 Intended use

2.1.1 Introduction

Rexroth products are developed and manufactured to the state-of-the-art.

The products are tested prior to delivery to ensure operational safety and reliability.

⚠ WARNING	**Personal injury and damage to property due to incorrect use of products!** The products may only be used as intended. Failure to use the products as intended may cause situations resulting in property damage and personal injury.

NOTICE	**Damages resulting from unintended use** Rexroth As the manufacturer does not assume any warranty, liability or compensatory claims for damages resulting from unintended use of the products. The user alone shall bear the risks of an unintended use of the products. Before using Rexroth products, make sure that all the prerequisites for an intended use of the products are met: – Personnel that in any way, shape or form uses Rexroth products must first read and understand the relevant safety instructions and be familiar with their intended use – Leave hardware products in their original state, i.e., do not make any structural modifications. It its not permitted to decompile software products or alter source codes – Do not install damaged or defective products or commission them – It has to be ensured that the products have been installed as described in the relevant documentation

2.1.2 Areas of use and application

Products of the ctrlX series are suitable for Motion/Logic applications.

NOTICE	Products of the ctrlX series may only be used with the accessories, mounting parts, and other components specified in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same applies to cables and lines. Only to be operated with the hardware component configurations and combinations expressly specified and with the software and firmware specified in the corresponding documentations and functional descriptions.

Products of the ctrlX series are suitable for single-axis as well as for multi-axis drive and control tasks. Device types with different equipment and interfaces are available for using the system in specific applications.

Typical areas of application:

- Building automation
- IoT and Security Gateway or Device
- Handling & Robotic

Controls of the ctrlX CORE series may only be operated under the mounting and installation conditions, in the position of normal use and under the ambient conditions (temperature, degree of protection, humidity, EMC, etc.) specified in the related documentations.

2.2 Unintended use

"Unintended use" refers to using the ctrlX products outside of the above-mentioned areas of application or under operating conditions and technical data other than described and specified in the documentation.

ctrlX products must not be used if they are exposed to following conditions:

- Operating conditions that do not meet the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited
- Applications that have not been expressly authorized by Rexroth

3 Safety instructions

The Safety instructions contained in the available application documentation feature specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.

The Safety instructions in this documentation are designed as follows:

⚠ **DANGER**	In case of non-compliance with this safety instruction, death or serious injury **will** occur.
⚠ **WARNING**	In case of non-compliance with this safety instruction, death or serious injury **could** occur.
⚠ **CAUTION**	In case of non-compliance with this safety instruction, minor or moderate injury could occur.
NOTICE	In case of non-compliance with this safety instruction, property damage could occur.

4 Introduction and overview

4.1 Basics

The Key Value Database app facilitates the configuration of user-defined persistent nodes on the ctrlX Data Layer.

Functions:

- Configuring explicit nodes of any type.
- Programmatic creation of implicit dynamic variable nodes, by any Data Layer client (create-on-write).

The functional scope of the Key Value Database app also includes reliable and powerful data storage. Persistence is based on a key/value database (KVD).

Licensing

The following license is required to operate the Key Value Database:

Description	Type identifier	Part number
License to operate the Key Value Database app	SWL-XC*-KVD-KEYVALUE*DB**-NNNN	R911418610

Further information

Documentation

- ctrlX CORE - App basics
- ctrlX CORE - License overview

Reference source

- ctrlX App Store

Web links

- ctrlX CORE - Community
- ctrlX CORE - How to
- ctrlX CORE - Forum

4.2 Quick start

1. Install the Key Value Database app on your ctrlX device.
2. Open the ctrlX-CORE web interface by logging in to the ctrlX device via a web browser.
3. Open the Data Layer window via the page navigation: *“Settings → Data Layer”*
4. Open the "kvd" node.
   The node is created by installing the Key Value Database app in the Data Layer and contains some child nodes as an application example.

Application

Configure explicit nodes or root nodes for implicit (dynamic) child nodes (.../*, .../**) to be inserted into the Data Layer address space. The configuration can be done by editing the configuration file "nodes.json", also see the following web documentation:

Setup – Introduction

Structure of the setup file

Supported interaction with a Data Layer client

Interaction	Description
Read value of a node Method: **"read"**	Reading a node by calling the "read" method. The argument of the method has to be the address of the node to be read.
Write value of a node Method: **"write"**	Create the node of the nodeClass "Variable" by calling the "write" method. The argument of the method has to be a variant containing a value of any data type.

Interaction	Description
Create node by writing a value Method: **"create-on-write"**	Create a previously non-existent implicit node as a child node of an implicit root node, by initially writing a value. Creating the node can be done by writing any value that creates an implicit node of the nodeClass "Variable" with the specified data type. The initial data type of the node is set and is retained. Subsequent write access to the node value fails if the data type of the written data does not match the data type of the node.
Delete node Method: **"remove"**	Supported only for implicit nodes! Delete a node by calling the "remove" method. The argument of the method has ot be the node address.

Unsupported interactions

Interaction	Description
Create node Method: **"create"**	Creating a node by calling the "create" method is not supported! Use the "create-on-write" method to create a node, see the "Supported interactions" section above.

5 Configuration editors

5.1 General

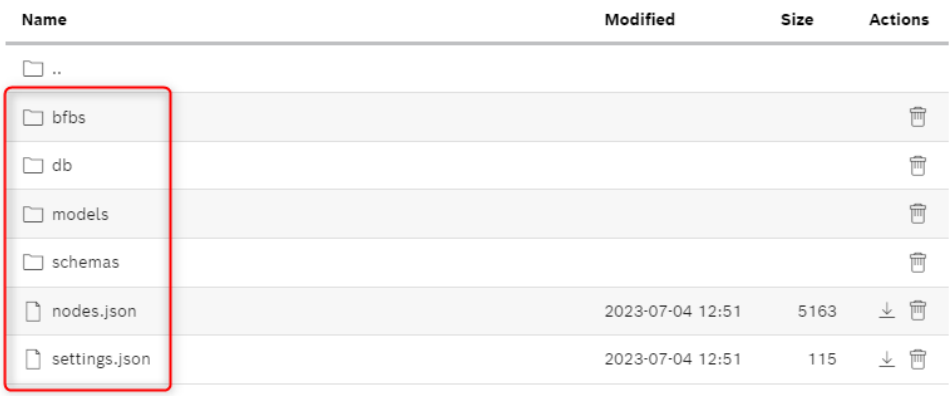
The Key Value Database app is either configured via access using WebDAV or in the ctrlX web interface in the app settings in the “Manage app data” window, see:

➦ Configuration via the ctrlX web interface
➦ Configuration via WebDAV

NOTICE

Warning

Please do not edit, delete or rename files within the folder structure where the "db" directory is located.

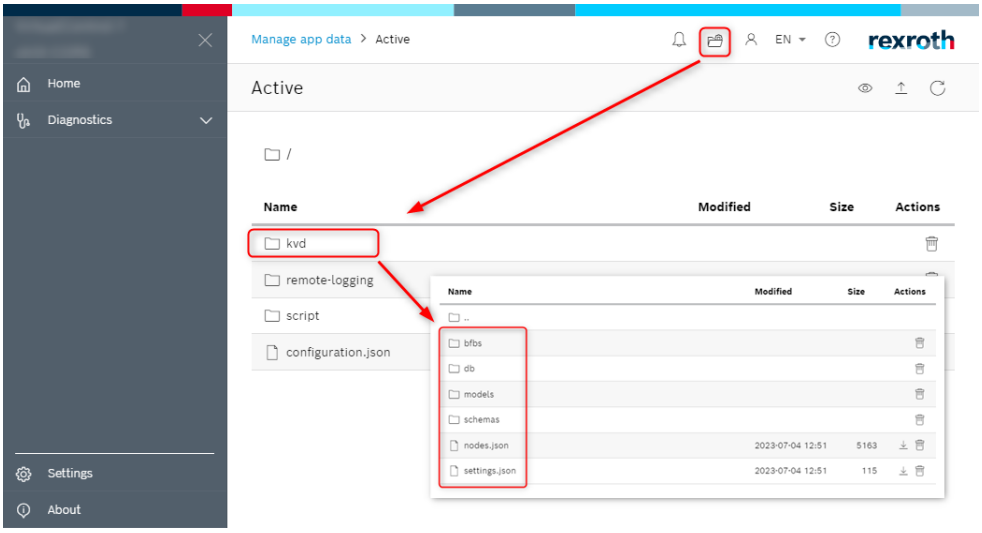


NOTICE	**Resetting the KVD persistence** Reset the Key Value Database persistence by deleting the "db" directory from the app configuration. In this case, the app is restarted and all data is deleted and reset to default values!

5.2 Configuration via the ctrlX web interface

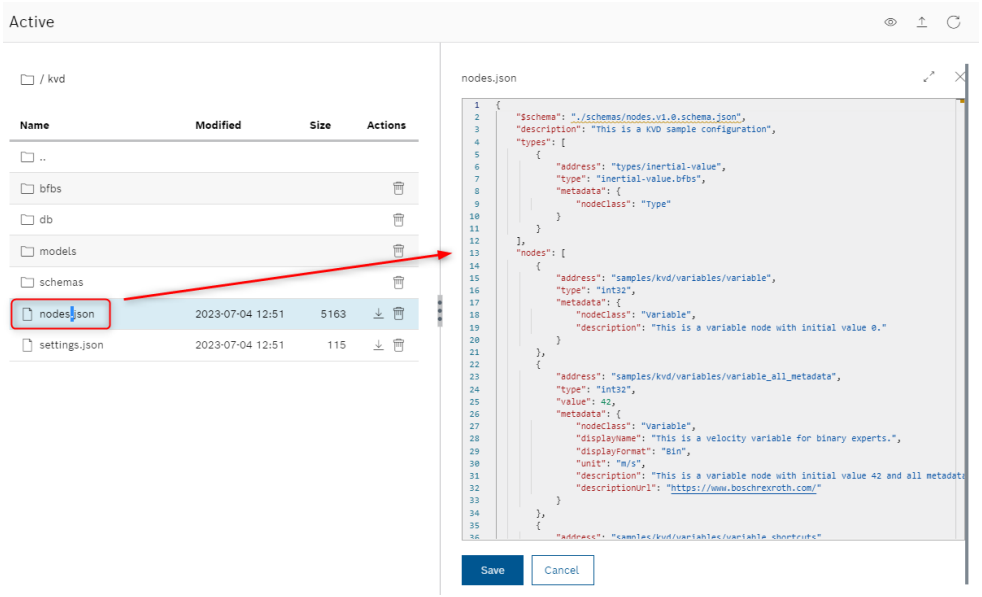
The Key Value Database app is directly configured in the web interface in the “Manage app data” window, see also:

➥ Window – Manage app data



Double-click on one of the configuration files to open the app data editor and select and save the configuration settings, as shown in the following example in the “nodes.json” file.

Saved changes are applied immediately.



5.3 Configuration via WebDAV

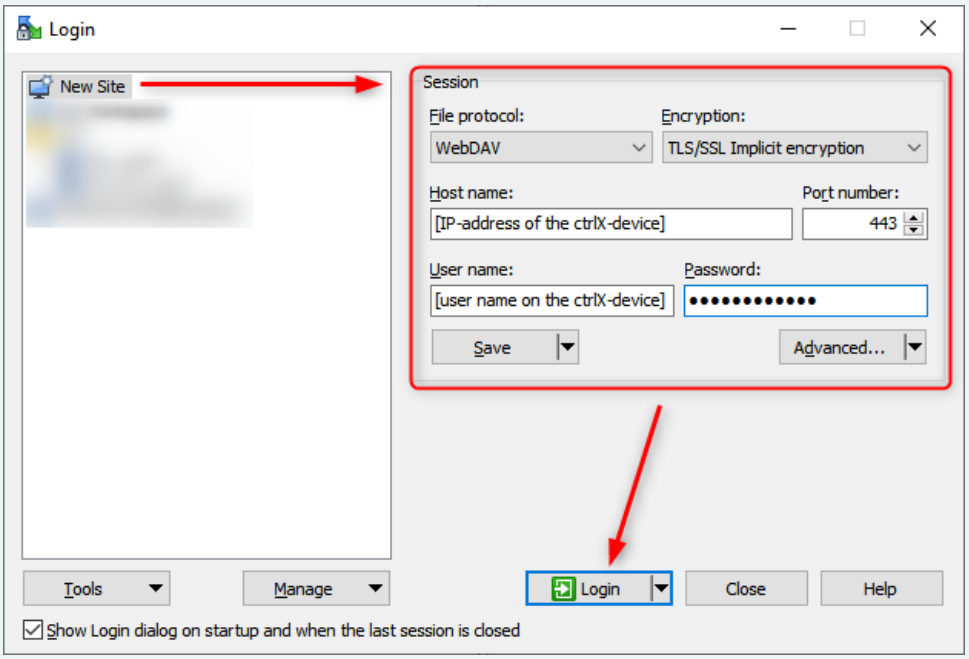
As an alternative to configuration via the ctrlX web interface, the app can also be configured remotely via a WebDAV client Key Value Database app.

The use of WinSCP is recommended by Rexroth, see: ➦ https://winscp.net/

Example: Change the configuration via WinSCP

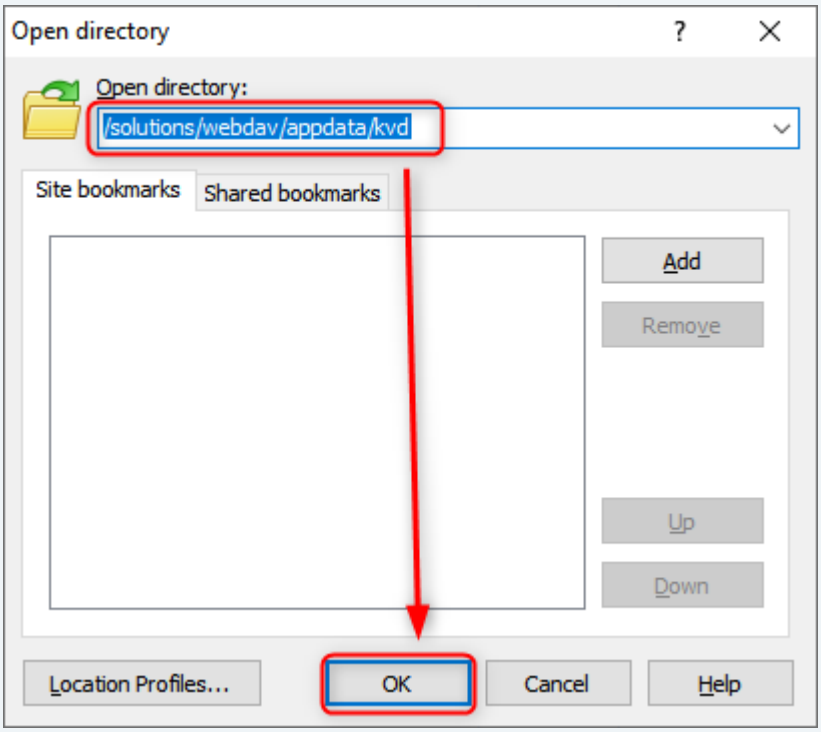
1. Start WinSCP and create a new session via “"New Site"”.
2. Select following settings:
   - File protocol: “WebDAV”
   - Encryption: “TLS/SSL Implicit encryption”
   - Host name: [IP address of the ctrlX device]
   - Username: [your username on the ctrlX device]
   - Password: [your password on the ctrlX device]

   ➡

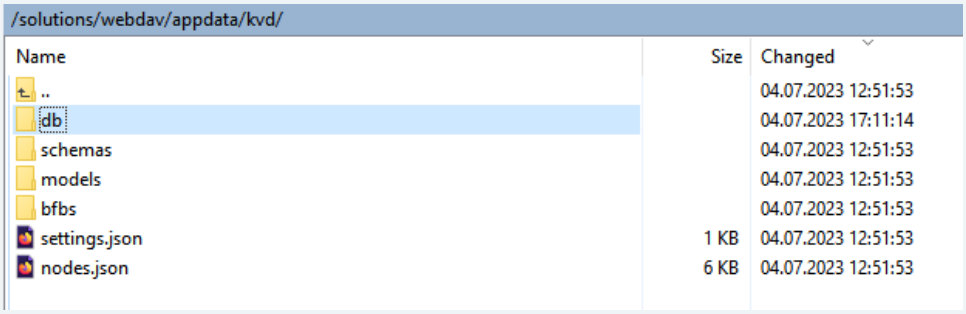


3. Click on “Login” to connect to the ctrlX device
4. If required, confirm the dialogs for selecting certificates
   ➡ WinSCP now has access to the file system of the ctrlX device

5. Select the target directory: `/solutions/webdav/appdata/kvd`
   Click in the remote session field and subsequently on [Ctrl + o] to specify the destination path.
   ➡



6. The available KVD configuration files are displayed.
   ➡



7. Change the desired configuration and save the change.
   ➡ The changes take effect immediately after saving.

6 General configuration

6.1 General

The general configuration settings of the Key Value Database app are stored in the “"settings.json"” configuration file. The file can be edited either via the ctrlX web interface or via WebDAV access, see: ➦ Configuration editors

The file contains the following entries that can be edited:

- **"db"**

Specifies the path of the underlying database. The app data directory `kvd/db` is the default value.

Also refer to: ➦KVD database configuration (location)

- **"readOnly"**
  Sets the configured node values to read-only, which means that write operations are forbidden and fail. Default value is: `false`.
- **"verbose"**
  Activates the detailed trace. Default value is: `false`.

6.2 KVD database configuration (location)

The location of the Key Value Database database can be configured in the ""settings.json"" configuration file and is set to the ctrlX device internal memory in the“kvd/db” directory by default.

Depending on the frequency of the database write cycles, it is recommended to transfer the database to an external memory in order to avoid afflicting the device-internal memory Key Value Database.

For this purpose, mounted external storage media can be used, which are configured either for data exchange or as memory extension on the ctrlX device.

6.2.1 Swap the database to an external storage medium configured for data exchange

Important notes for use

- The Key Value Database database is unencrypted and not protected from unauthorized access.
- The Key Value Database database can be transferred to another ctrlX OS device on which a different Key Value Database app instance is running.
- The Key Value Database database can be backed up or it can be replaced.

Example: Removable disk as memory extension / entry in “settings.json”

```
{
"db": "/media/mmcblk1p1/db",
...
}
```

6.2.2 Swap database to external storage medium configured for memory extension

Important notes for use

- The Key Value Database database is fully encrypted and protected from unauthorized access.
- The Key Value Database database cannot be transferred to another ctrlX OS device and is bound to the device.
- A backup or exchange of the Key Value Database database is currently not possible.
- After unmounting (removing) the removable disk, the Key Value Database database on this ctrlX device is lost.

Example: Removable disk as memory extension / entry in “settings.json”

```
{
"db": "$SNAP_COMMON/storage-extension/$SNAP_NAME/db",
...
}
```

7 Configuring the nodes

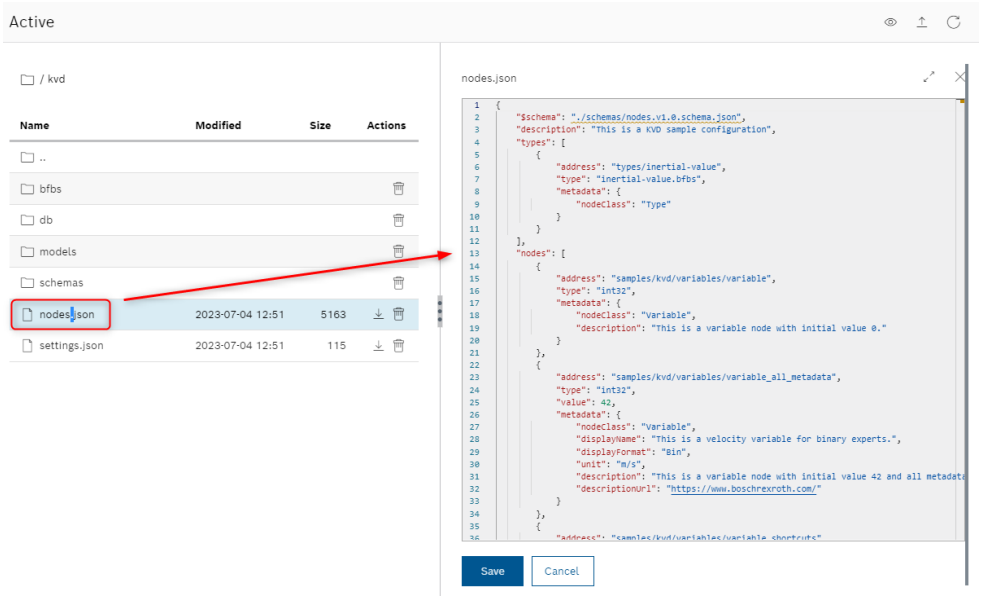
7.1 General

Nodes that are to be persisted in the Data Layer have to be configured in the “nodes.json” configuration file. The configuration can be done either via the ctrlX web interface, or via remote connection (WebDAV), see:

➦ Configuration via the ctrlX web interface
➦ Configuration via WebDAV

The configuration file is stored in the app data of the Key Value Database app and can be viewed and edited in the “Manage app data” window and can be edited, see also:

➦ Window – Manage app data



The configuration supports explicit and implicit (dynamic) root nodes.

7.2 Configure nodes [nodes.json]

This section describes the elements of the configuration file “nodes.json”.

Example: Configuration file “nodes.json” **in delivery state after installing the Key Value Database app**

nodes.json

```
{
    "$schema": "./schemas/nodes.v1.0.schema.json",
    "description": "This is a KVD sample configuration",
    "types": [
        {
            "address": "types/inertial-value",
            "type": "inertial-value.bfbs",
            "metadata": {
                "nodeClass": "Type"
            }
        }
    ],
    "nodes": [
        {
            "address": "samples/kvd/variables/variable",
            "type": "int32",
            "metadata": {
                "nodeClass": "Variable",
                "description": "This is a variable node with initial value 0."
            }
        },
        {
            "address": "samples/kvd/variables/variable_all_metadata",
            "type": "int32",
            "value": 42,
            "metadata": {
                "nodeClass": "Variable",
                "displayName": "This is a velocity variable for binary experts.",
                "displayFormat": "Bin",
                "unit": "m/s",
                "description": "This is a variable node with initial value 42 and all metadata.",
                "descriptionUrl": "https://www.boschrexroth.com/"
            }
        },
```

Node element description

Node element	Description
"types"	The list of types to be registered (nodeClass Type).
"nodes"	List of nodes to be filled. Hierarchical structuring nodes and variable nodes are supported, see **"nodeClass"** below.
"address"	Contains the node address. The following address types can be used to specify an implicit node: • Absolute path (leaf) • Simple wildcard end address (/*) • Duplicate wildcard end address (/**)
"type"	For the data type of the node, see: ➦ Node data type
"value"	The initially selected node value, see: ➦ Node element – **"value"**
"metadata"	Metadata of the node. **Access restrictions for users**: To configure a read-only node branch or to restrict access to the node, please use the ➦ Security functions of the ctrlX Data Layer
"nodeClass"	Configuring the node class, see: ➦ Node class

Node element	Description
"volatile"	Node function for configuring volatile nodes, see: ➦ Volatile nodes
"write"	Meta data for configuring the write access, see also: ➦ Read-only nodes
"mode"	Mode of the node, for example "read-only" to configure read-only nodes, see: ➦ Read-only nodes
"unit"	Unit of measurement of the node value, see: ➦ Node element – "unit"
"displayName"	For the display name of the node, see: ➦ Node element – "displayName"
"displayFormat"	For the display format of the node, see: ➦ Node element – "displayFormat"
"description"	For the node description, see: ➦ Node element – "description"
"descriptionUrl"	For the description URL of the node, see: ➦ Node element – "descriptionUrl"

7.3 Nodes – configuration examples

Nodes with minimal meta data

Numeric variable node (int32)

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/temperature1",
    "type": "int32",
    "metadata": {
      "nodeClass": "Variable"
    }
  },
  ...
  {
    "address": "mycompany/europe/myplants/plant1/temperature2",
    "type": "int32",
    "nodeClass": "Variable"
  },
  ...
]
```

Nodes with maximum meta data

Numeric variable node (int32)

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/temperature",
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "displayName": "This is a velocity variable node for binary experts.",
      "displayFormat": "Bin",
      "unit": "m/s",
      "description": "This is a variable node with all metadata.",
      "descriptionUrl": "https://www.boschrexroth.com/"
    }
  },
  ...
]
```

Flat buffer variable node (built-in type)

To configure a node of a "built-in" flat buffer type, set the type of the node to the address of the relevant target type: `"types/..."`

Example

```
nodes: [
  {
    ...
    "type": "types/datalayer/server-settings",
    "metadata": {
      "nodeClass": "Variable",
    }
  },
  ...
]
```

Flat buffer variable node (custom type)

To create a customized flat buffer, configure a node of the `"nodeClass": "Type"` and add it to the `"types"` section.

Set the address of the type to the address you want the instance nodes to point to, in this example: `"types/inertial-value"`

Set the type to the name of the corresponding compiled flat buffers file (*.bfbs).

Upload the *.bfbs file to the app data directory kvd/bfbs.

Example

```
nodes: [
  {
    "address": "types/inertial-value",
    "type": "inertial-value.bfbs",
    "metadata": {
      "nodeClass": "Type",
     }
  },
  ...
]
```

A variable node of this type can be specified (value optional):

Example

```
nodes: [
  {
    "address": "kvd/samples/variables/flatbuffers/inertial-value",
    "type": "types/inertial-value",
    "value": {
      "x": 1,
      "y": 2,
      "z": 3
    },
    "metadata": {
      "nodeClass": "Variable",
      "description": "custom flatbuffers type: 'types/inertial-value'"
    }
  },
  ...
]
```

7.4 Explicit nodes

An explicit node has an absolute address and cannot be deleted (regardless of allowed operations by the nodeClass).

Example: Explicit node

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/temperature",
    "type": "int32",
    "metadata": {
      "nodeClass": "Variable",
    }
  },
  ...
]
```

7.5 Implicit nodes

An implicit node can be created (create-on-write) and removed.

Implicit root nodes

An implicit root node defines the entry of an implicit branch that allows the creation and deletion of child nodes only at the next hierarchy level (/*) or at all levels (/**).

Example: Implicit root node/creation and deletion only possible on the next hierarchy level

```
{
  "address": "mycompany/europe/myplants/plant1/*",
},
```

Example: Implicit root node/creation and deletion only possible on all subordinate hierarchy levels

```
{
  "address": "mycompany/europe/myplants/plant1/**",
},
```

7.6 Volatile nodes

The configuration of volatile, non-permanent nodes can be realized by using the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By setting the extension key "autosafe" in the meta data to "false".
- **Method 2** (flat configuration scheme/only possible for explicit nodes)
  By setting the node function "volatile" to "true".

1. Volatile node with: "metadata > extensions > autoSave > false"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/volatile_temperature",
    "type": "int32",
    "metadata": {
      "nodeClass": "Variable",
      "extensions": [
        {
         "key": "autoSave",
         "value": "false"
        }
      ]
    }
  },
  ...
]
```

2. Volatile node with: "volatile > true"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant2/volatile_temperature",
    "type": "int32",
    "volatile": true,
  },
  ...
]
```

7.7 Read-only nodes

The configuration of read-only nodes can be realized by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By setting the node meta datum "operations": "write" to "false".
- **Method 2** (flat configuration scheme)
  By setting the node mode "mode" to "read-only".

1. Write protection: "metadata > operations > write > false"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "operations": {
         "write": false
      },
      "description": "This is a read-only variable node"
    }
  }
  ...
]
```

2. Write protection: "mode > read-only"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "mode": "read-only",
    "metadata": {
      "nodeClass": "Variable",
      "description": "This is a read-only variable node"
    }
  }
  ...
]
```

7.8 Node class

In the node element **"nodeClass"**, the class of the node is configured (default value: `[Variable]`).

Supported node classes

Hierarchical structuring nodes:

- nodeClass: **Folder**
- nodeClass: **Collection**
- nodeClass: **Resource**

Variable nodes:

- nodeClass: **Variable**

Configuring the node class

The node class can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By labeling the node class in the node meta data, in the meta data "node-Class".
- **Method 2** (flat configuration scheme)
  By labeling the node class in the first structure level via the node entry "nodeClass".

1. Definition of the node class via: "metadata > nodeClass"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "description": "This is a variable node"
    }
  }
  ...
]
```

2. Definition of the node class in the first level with: "nodeClass"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "nodeClass": "Variable",
    "metadata": {
      "description": "This is a variable node"
    }
  }
  ...
]
```

Supported operations

Executable by any connected client, depending on the node class:

Operation	nodeClass: Variable	nodeClass: Resource	nodeClass: Collection	nodeClass: Folder	nodeClass: Type
Write	X	X	X		X
Read	X	X			
Delete		X			
Browse	X	X	X	X	X

7.9 Node data type

In the node element **"type"**, the data type of the node is configured (default value:`unknown`).

Once the variable node is created with its configured data type, the written data type has to match the configured data type, otherwise an error code is returned to the client performing the write operation.

Scalar

Type	Description
`unknown`	Unknown data type (not initialized, empty, null)
`bool8`	bool (8 bit)
`int8`	signed int (8 bit)
`uint8`	unsigned int (8 bit)
`int16`	signed int (16 bit)
`uint16`	unsigned int (16 bit)
`int32`	signed int (32 bit)
`uint32`	unsigned int (32 bit)
`int64`	signed int (64 bit)
`uint64`	unsigned int (64 bit)
`float, float32`	float (32 bit)
`double, float64`	double (64 bit)
`string`	string (UTF-8)
`timestamp`	timestamp (FILETIME) 64 Bit 100ns since 1.1.1601 (UTC)
`flatbuffers`	bytes as complex data type coded as flat buffer
`raw`	raw bytes

Arrays

Type	Description
`arbool8`	array of bool (8 bit)
`arint8`	array of signed int (8 bit)
`aruint8`	array of unsigned int (8 bit)
`arint16`	array of signed int (16 bit)
`aruint16`	array of unsigned int (16 bit)
`arint32`	array of signed int (32 bit)
`aruint32`	array of unsigned int (32 bit)
`arint64`	array of signed int (64 bit)
`aruint64`	array of unsigned int (64 bit)
`arfloat, arfloat32`	array of float (32 bit)
`ardouble, arfloat64`	array of float (64 bit)
`arstring`	array of string (UTF-8)
`artimestamp`	array of timestamp (FILETIME) 64 bits 100 ns since 1.1.1601 (UTC)

7.10 Node element – "value"

Specifies the preset value of the node.

Default value: `(0, false, "")`

Supported value types:

- `string`
- `number`
- `integer`
- `boolean`
- `array`
- `object`

Examples

Variable node of type numerical value

```
{
...
"Wert": "42",
}
```

A variable node of the type string value

```
{
...
"Wert": "42",
}
```

7.11 Node element – "unit"

The node element defines the unit of measurement of the node value, for example m/s (meters per second).

By default, the unit of measurement of a node value is not preconfigured.

The unit of measurement can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By configuring the desired unit of measurement in the meta data of the node via the "unit" entry.
- **Method 2** (flat configuration scheme)
  By configuring the unit of measurement not in the meta data, but in the first structure level with the "unit" entry.

1. Configure the unit of measurement in the meta data: "metadata > unit"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "unit": "m/s",
    }
  }
  ...
]
```

2. Configure the unit of measurement text in the first level with “UNIT”

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "unit": "m/s",
    "metadata": {
      "nodeClass": "Variable",
    }
  }
  ...
]
```

7.12 Node element – "displayName"

The node element defines the displayed node name.

By default, the node address is preconfigured as the display name.

The display name can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By configuring the display name in the meta data of the node, via the "displayName" entry.
- **Method 2** (flat configuration scheme)
  By configuring the display name not in the meta data, but in the first structure level with the "displayName" entry.

1. Configure the display name in the meta data: "metadata > displayName"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "unit": "m/s",
      "displayName": "velocity axis1",
    }
  }
  ...
]
```

2. Configure display name in first level with "displayName

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "unit": "m/s",
    "displayName": "velocity axis1",
    "metadata": {
      "nodeClass": "Variable",
    }
  }
  ...
]
```

7.13 Node element – "displayFormat"

The node element defines the display format of the node.

By default, the setting `Auto` is preconfigured.

Supported display formats:

- **Auto** (automatic selection of the display format)
- **Bin** (binary)
- **Oct** (Octal)
- **Dec** (Decimal)
- **Hex** (Hexadecimal)

The display format can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By configuring the display format in the metadata of the node, via the entry "displayFormat".
- **Method 2** (flat configuration scheme)
  By configuring the display format not in the meta data, but in the first structure level with the entry "displayFormat".

1. Configure the display format in the meta data: "metadata > displayFormat"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "displayFormat": "Dez",
    }
  }
  ...
]
```

2. Configure the display name in the first level with "displayFormat"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "displayFormat": "Dez",
    "metadata": {
      "nodeClass": "Variable",
    }
  }
  ...
]
```

7.14 Node element – "description"

The node element defines a description text for the node.

By default, no description text is preconfigured.

The description text can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By configuring the description text in the meta data of the node, via the "description" entry.
- **Method 2** (flat configuration scheme)
  By configuring the description text not in the meta data, but in the first structure level with the "description" entry.

1. Configure the description text in the meta data: "metadata > description"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "unit": "m/s",
      "displayName": "velocity axis1",
      "description": "Velocity of axis1 in m/s",
    }
  }
  ...
]
```

2. Configure the description text in first level with “description”

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "unit": "m/s",
    "displayName": "velocity axis1",
    "description": "Velocity of axis1 in m/s",
    "metadata": {
      "nodeClass": "Variable",
    }
  }
  ...
]
```

7.15 Node element – "descriptionUrl"

The node element defines a description URL for the node.

By default, no description URL is preconfigured.

The description URL can be configured by the following entries in the nodes.json configuration file:

If you want to have the flattest possible node structure, use method 2.

- **Method 1** (standard configuration scheme)
  By configuring the description URL in the meta data of the node, via the "desctriptionUrl" entry.
- **Method 2** (flat configuration scheme)
  By configuring the description URL not in the meta data, but in the first structure level with the "descriptionUrl" entry.

1. Configure the description text in the meta data: "metadata > descriptionUrl"

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant2,
    "type": "int32",
    "value": 42,
    "metadata": {
      "nodeClass": "Variable",
      "unit": "m/s",
      "displayName": "velocity axis1",
      "descriptionUrl": "https://www.boschrexroth.com/",
    }
  }
  ...
]
```

2. Configure the description text in first level with “descriptionUrl”

```
nodes: [
  {
    "address": "mycompany/europe/myplants/plant1/myconstant1,
    "type": "int32",
    "value": 42,
    "unit": "m/s",
    "displayName": "velocity axis1",
    "descriptionUrl": "https://www.boschrexroth.com/",
    "metadata": {
      "nodeClass": "Variable",
    }
  }
  ...
]
```

8 Configuring Data Layer model nodes

8.1 General

The Key Value Database app supports ctrlX Data Layer models in YAML format (*.yaml), from which model node instances can be populated. The models have to conform to the ctrlX Data Layer schema definition.

For further details please refer to the example models in the Data Layer directory: kvd/models

Model example

```
$schema: datalayer.v1.0.schema.json
name: snake
version: 0.0.1
elements:
    weight:
        description: The weight of the snake.
        nodeClass: Variable
        mode: read-write
        value: 42
        typeSchema:
            type: int32
    venomous:
        description: The snake bite is venomous.
        nodeClass: Variable
        mode: read-write
        value: true
        typeSchema:
           type: bool8
```

8.2 Model interpretation and limitations

- The Key Value Database app interprets the models without checking their syntax against the schema definition for correctness.
- Unknown or not useful configurations are ignored (e.g. `"itemsModel"` of a `"collection"`, ...).

8.3 Uploading models

Upload your ctrlX Data Layer models (*.yaml) to the app data directory: **kvd/models**

8.4 Configuring the model instance

1. Add the node instances to the node configuration file nodes.json.
2. Set the model of the node instance to the file name of your model, e.g. animals.v1.0.yaml
3. Set the `"nodeClass"` of the meta data to `"Resource"`.

Example

```
{
  "address": "samples/kvd/models/zoo/animals",
  "model": "animals.v1.0.yaml",
  "metadata": {
    "nodeClass": "Resource",
    "description": "This is a animals model instance node"
  }
},
{
  "address": "samples/kvd/models/wildlife/animals",
  "model": "animals.v1.0.yaml",
  "metadata": {
    "nodeClass": "Resource",
    "description": "This is a animals model instance node"
  }
},
...
```

8.5 Configuring the read-only model node

Configure the value of a node to be read-only (e.g., for non-writable constant values) by setting the node mode "`mode`" to `read-only` instead of `read-write`.

Example

```
...
elements:
    my_constant:
        description: A constant value.
        nodeClass: Variable
        mode: read-only
        value: 42
        typeSchema:
            type: int32
...
```

8.6 Referencing the model node

Recursive model hierarchies are supported by the Key Value Database app and allow objects to be assembled using a class-oriented approach to configure a more complex node tree.

1. Ensure that all applications are in the **kvd/models** directory.
2. Use the keyword `$ref` to insert nodes of a child model into a parent node, see the following example.

Configuration example (main model and referenced child models)

Main model "animals.v1.0.yaml"

```
$schema: datalayer.v1.0.schema.json
name: animals
version: 0.0.1
elements:
    count:
        description: The number of animals.
        nodeClass: Variable
        mode: read-only
        value: 42
        typeSchema:
            type: string
    reptiles:
        description: The reptiles.
        $ref: reptiles.v1.0.yaml
    ...
```

Child model "reptiles.v1.0.yaml" referenced from "animals.v1.0.yaml"

```
$schema: datalayer.v1.0.schema.json
name: reptiles
version: 0.0.1
elements:
    snake:
        description: The snake.
        $ref: snake.v1.0.yaml
    ...
```

Child model "snake.v1.0.yaml" referenced from "reptiles.v1.0.yaml"

```
$schema: datalayer.v1.0.schema.json
name: snake
version: 0.0.1
elements:
    weight:
        description: The weight of the snake.
        nodeClass: Variable
        mode: read-write
        value: 42
        typeSchema:
            type: int32
    venomous:
        description: The snake bite is venomous.
        nodeClass: Variable
        mode: read-write
        value: true
        typeSchema:
            type: bool8
    ...
```

9 Interaction with the Node-RED app

The Key Value Database app enables the following interactions in interaction with the Node-RED app:

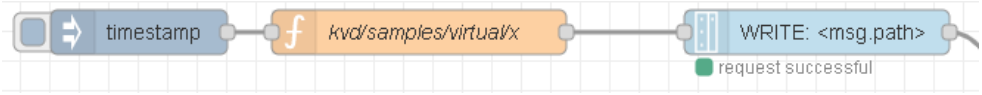
- **Read value of a node**
  Via a Data Layer request with the "READ" method.
  For this purpose, the call path and the address of the node to be read have to be transferred as arguments to the method.
- **Write the value of a node** (Create-on-Write)
  Via a Data Layer request with the "WRITE" method.
  For this purpose, the argument has to contain an object with the properties "type" and "value".
- **Delete a node**
  Via a Data Layer request with the "DELETE" method.
  The path has to transfer the address of the node to be removed as an argument to the method.

Unsupported methods

The creation of new nodes with the method "CREATE" is not supported.

Arguments to execute the methods

Use a function node as input to specify the required arguments.



Example: Arguments for type/value/path

```
var payload = {
    type: 'int32',
    value: 42
}

msg.path = 'mycompany/europe/myplants/plant1/temperature';
msg.payload = payload;
return msg;
```

10 Troubleshooting

If a configuration error occurs, the Key Value Database app does not start and no KVD node is displayed in the ctrlX Data Layer structure. In this case, please check the diagnostic system in the ctrlX web interface for error details (e.g. schema validation failed, etc.).

Search the ctrlX diagnostic system for Key Value Database errors

Step 1: Enabling the system messages in the diagnostics

1. Open the ctrlX web interface and switch to the window: *“Diagnostics → Logbook”*
2. Click on the ⚙ button to open the dialog “Settings”.
3. Enable the “Show system messages” option and save the setting.

Step 2: Filtering diagnostics according toKey Value Database errors

1. Click on Filter to open the filter settings.
2. Activate the filter category:“Units”
3. In the right drop-down list, scroll down to entry “snap.rexroth-kvd.kvd.service” and enable the filter option.
   ➡ Due to the filter setting, only entries of the Key Value Database app are displayed, which simplifies the error analysis.

Diagnostics during normal operation

With error-free operation of the Key Value Database app, the following diagnostic entry is displayed: “[kvd] running”

Diagnostics in case of errors

If an error occurs, update the diagnostic entries via the ↻ button to obtain detailed error information.

11 Related documentation

11.1 Overview

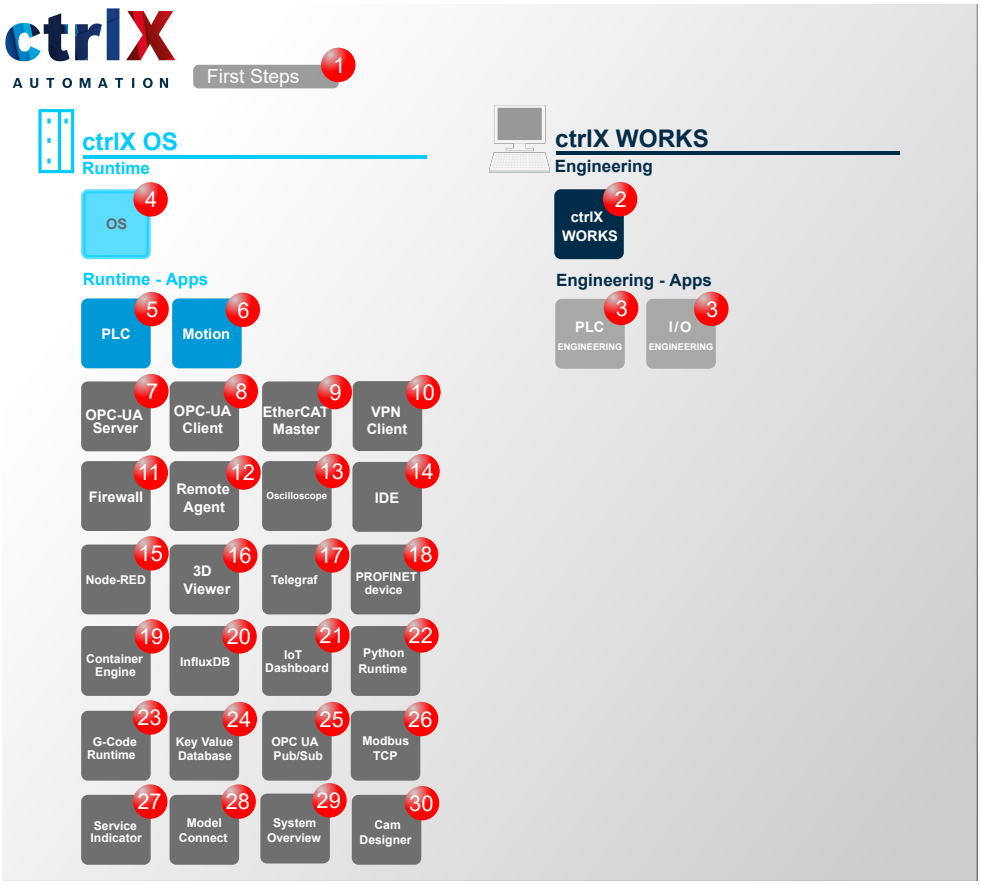


Fig. 1: Overview on further documentations

11.2 ctrlX AUTOMATION

No.	Documentation
1	**ctrlX WORKS First Steps** Quick Start Guide ➥ Web documentation link Ordering information: • DOK-XWORKS-F*STEP*****-QU01-EN-P • R911403760

11.3 ctrlX WORKS

No.	Documentation
2	**ctrlX WORKS - Basic System 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XWORKS-WRK***V03**-APRS-EN-P • R911423376
3	**ctrlX PLC Engineering - PLC Programming System 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XPLC**-PLE***V03**-APRS-EN-P • R911423378
3	**ctrlX PLC Engineering - PLC Libraries 03VRS** Reference Book ➦ Web documentation link Ordering information: • DOK-XPLC**-LIB***V03**-RERS-EN-P • R911423456

11.4 ctrlX OS

No.	Documentation
4	**ctrlX OS - Operating System for ctrlX CORE Control Devices 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-XCR***V03**-APRS-EN-P • R911423382
	ctrlX OS - Data Layer Nodes 03VRS Reference Book ➦ Web documentation link Ordering information: • DOK-XCORE*-DL****V03**-RERS-EN-P • R911423384
	ctrlX OS - Diagnostics 03VRS Reference Book ➦ Web documentation link Ordering information: • DOK-XCORE*-DIAG**V03**-RERS-EN-P • R911423386

11.5 ctrlX OS apps

No.	Documentation
5	**PLC App - PLC Runtime Environment for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-PLC***V03**-APRS-EN-P ● R911423401
6	**Motion App - Motion Runtime Environment for ctrlX CORE 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-MOT***V03**-APRS-EN-P ● R911423405
7	**OPC UA Server App - OPC UA Server for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-UAS***V03**-APRS-EN-P ● R911423392
8	**OPC UA Client App - OPC UA Client for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-UAC***V03**-APRS-EN-P ● R911423390
9	**EtherCAT Master App - EtherCAT Master for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-ECM***V03**-APRS-EN-P ● R911423394
10	**VPN Client App - Remote Support Software for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-VPN***V03**-APRS-EN-P ● R911423388
11	**Firewall App - Security Functions for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-FRW***V03**-APRS-EN-P ● R911423397

No.	Documentation
12	**Remote Agent App - ctrlX Device Portal Connection for ctrlX OS Devices 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-RMA***V03**-APRS-EN-P • R911423399
13	**Oscilloscope App - Oscilloscope Function for ctrlX OS Devices 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-OSC***V03**-APRS-EN-P • R911423407
14	**IDE App - Integrated Development Environment 02VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-IEN***V02**-APRS-EN-P • R911421612
15	**Node RED App - Graphic Programming for ctrlX OS 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-RED***V03**-APRS-EN-P • R911423403
16	**3D Viewer App - Browser-based 3D Kinematic Simulation for ctrlX OS 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-3DV***V03**-APRS-EN-P • R911423411
17	**Telegraf App - Server Agent for Collecting Data in the Data Layer 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-TSA***V03**-APRS-EN-P • R911425238
18	**PROFINET Device App - PROFINET Device for ctrlX OS 03VRS** Application Manual ➦Web documentation link Ordering information: • DOK-XCORE*-PND***V03**-APRS-EN-P • R911425232

No.	Documentation
19	**Container Engine App - Using Docker® Images on the ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-DOE***V03**-APRS-EN-P • R911425234
20	**InfluxDB App - Inflox Database Connection for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-IDB***V03**-APRS-EN-P • R911425240
21	**IoT Dashboard App - Data Visualization in Dynamic, Interactive Dashboards 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-GDB***V03**-APRS-EN-P • R911425248
22	**Python Runtime App - Python Runtime Environment for ctrlX CORE 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-PYR***V03**-APRS-EN-P • R911425244
23	**G-Code Runtime App - G-Code Interpreter for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-GCO***V03**-APRS-EN-P • R911425246
24	**Key Value Database App - Data Layer Management 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-KVD***V03**-APRS-EN-P • R911425250
25	**OPC UA Pub/Sub App - OPC UA Pub/Sub for ctrlX OS 03VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-UAP***V03**-APRS-EN-P • R911423409

No.	Documentation
26	**Modbus TCP App - Modbus TCP Communication over ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-MBT***V03**-APRS-EN-P • R911425236
27	**Service Indicator - App Service Indicator for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-SIN***V03**-APRS-EN-P • R911425242
28	**Model Connect App - Target for Model-based Development and Simulation for ctrlX OS 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-MOC***V03**-APRS-EN-P • R911425252
29	**System Overview App - System Topology and System Information 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-SOV***V03**-APRS-EN-P • R911425254
30	**Cam Designer - Configuring ctrlX MOTION Cams 03VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XWORKS-CAM***V03**-APRS-EN-P • R911427217

12 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts provide you with advice and assistance. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
Email:	➦ service.svc@boschrexroth.de
Internet:	➦ http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

13 Index

rexroth
A Bosch Company

Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main
Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911425250 01