

# AUTOMATIC CONFIGURATION OF MOTORS AT THE EUROPEAN XFEL

F. Sohn\*, W. Ehsan, G. Giovanetti, D. Goeries, I. Karpics, K. Sukharnikov  
European XFEL GmbH, Schenefeld, Germany

## Abstract

The European XFEL (EuXFEL) scientific facility relies heavily on the SCADA control system Karabo to configure and control a plethora of hardware devices. In this contribution a software solution for automatic configuration of collections of like Karabo devices is presented. Parameter presets for the automatic configuration are stored in a central database. In particular, the tool is used in the configuration of collections of single-axis motors, which is a recurring task at EuXFEL. To facilitate flexible experimental setup, motors are moved within the EuXFEL and reused at various locations in the operation of scientific instruments. A set of parameters has to be configured for each motor controller, depending on the controller and actuator model attached to a given programmable logic controller terminal, and the location of the motor. Since manual configurations are time-consuming and error-prone for large numbers of devices, a database-driven configuration of motor parameters is desirable. The software tool allows to assign and apply stored preset configurations to individual motors. Differences between the online configurations of the motors and the stored configurations are highlighted. Moreover, the software includes a “locking” feature to prevent motor usage after unintentional reconfigurations, which could lead to hardware damage.

## INTRODUCTION

Research at the EuXFEL relies heavily on the use of more than 3000 motors to move components within the various scientific installations. Experimental setups are frequently modified to provide the best suitable infrastructure for specific scientific measurements. For this reason and, furthermore, to save resources, motors are relocated within the EuXFEL and reinstalled in various locations within the experimental setups. Typically, each motor has more than 150 configurable parameters and needs to be configured at each new location.

The necessity to reconfigure the large number of motors and their parameters leads to an extensive time investment from staff and is error-prone, if done manually. Hence, the *Motor Configurator* software tool for the automatic configuration of motors has been developed, which aims to achieve the following main goals:

- minimize time spent by staff to configure motors,
- minimize mistakes due to manual configuration of motors,
- protect against hardware damage due to accidental misconfigurations of motors.

The software tool is based on the SCADA system Karabo [1,2], which is developed at the EuXFEL. Karabo provides a high-performance, reliable, and user-friendly environment to configure and control a plethora of hardware devices, and implement high-level procedures on top of these devices. One distinctive feature of Karabo are so-called scenes, easily configurable, multi-purpose graphical user interfaces that can be shipped with Karabo-based software tools. The Motor Configurator software includes a scene to provide a user-friendly graphical user interface.

## TECHNICAL BACKGROUND

For the integration of a hardware device in Karabo a software *device class* is written, which provides access to hardware features. Within Karabo, each individual hardware device is represented in terms of an *instance* of the respective device class. For each device instance, a *configuration*, i. e. a set of *operational parameters* for the hardware device, is held by the control system.

Figure 1 shows a schematic of the IT infrastructure used to control the motorized stages in the experimental instruments at EuXFEL. Most motors at EuXFEL are connected to *programmable logic controllers (PLCs)*, one motor to one PLC terminal. The PLCs are interfaced with the Karabo SCADA system. Information about the motor hardware is held by the PLCs and forwarded to Karabo. Operational parameters and motion commands are issued within Karabo and routed via the PLCs to the motor hardware.

Within Karabo each PLC terminal receives a unique identifier, the *terminal ID*. An instance of the correct device class for the connected hardware is automatically created after information about the detected hardware has been forwarded from the PLC to Karabo. Importantly, a configuration is assigned to a terminal ID, but cannot be assigned to the connected hardware directly.

## OPERATIONAL REQUIREMENTS FOR EXPERIMENTAL SETUPS

As motors are relocated between and within experimental setups, the assignment of motors to PLC terminals is subject to changes due to motors being

- reassigned to different PLC terminals,
- added to or removed from the setup.

Even if a motor driver of the same type as the previous setup is connected to a PLC terminal, the previous configuration for the PLC terminal in Karabo might not be suitable due to a motion stage with different mechanical and electrical requirements being connected to that driver. Hence, a

\* florian.sohn@xfel.eu

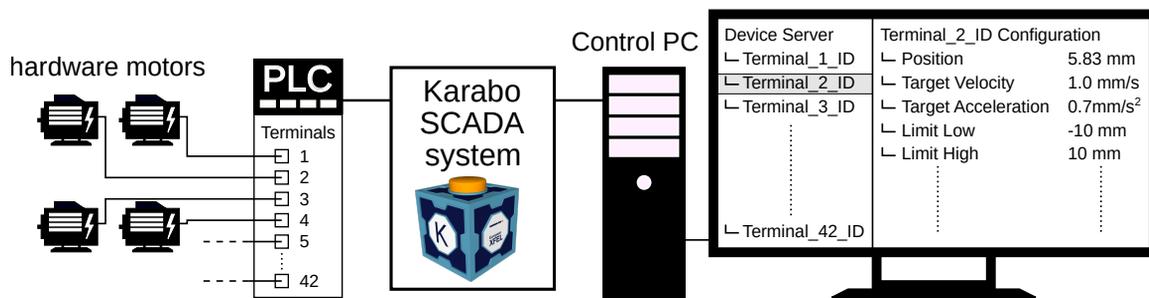


Figure 1: Schematic of the IT infrastructure used to control motorized stages in experimental components at EuXFEL. Details are given in the main text.

reconfiguration of the parameters of a terminal ID is necessary upon each exchange of the connected hardware device. Due to the number of motors in use at the endstations, and the number of parameters in a motor configuration, manual reconfigurations are time-consuming and error-prone.

## THE MOTOR CONFIGURATOR

The Motor Configurator software tool has been implemented to eliminate the necessity for manual reconfigurations of motor parameters. The software compares the configuration of each online motor device with a respective configuration preset stored in a database and highlights motor configurations that diverge from the respective presets. Motors with inconsistent configurations can be locked automatically to prevent further movement, and thus potential damage. If a configuration mismatch is detected, a stored preset configuration can be easily applied to the online motor. In the following, the features of the software tool are described in detail.

Configuration presets are created by reading a set of selected parameters from the configuration of an online motor. The configuration preset is then saved to an XML file. Parameters to be added to a configuration preset can be set in the *Motor Parameters* table, see Fig. 2, for each device class. In the same way, the Motor Parameters table lets the user specify whether a parameter shall be included in comparisons of preset and online configurations, and whether a parameter of a preset shall be written to the online configuration once the preset is applied to the online motor.

The *Stage Assignment* table is used to assign each terminal ID a configuration preset. This configuration preset is then used in comparisons with the online configuration of the motor connected to the respective PLC terminal. The *result table* indicates for each terminal ID whether differences between its current configuration and the stored preset are detected. If a configuration mismatch is found, a dialog with a detailed comparison of the configuration and the preset can be opened by double-clicking on the respective table entry. The preset is applied to the online motor by clicking the “Apply” button at the bottom of the dialog.

A check of the configurations of all online motors can be either triggered manually by clicking the “Compare” button or automatically in periodic time intervals, if the monitoring

feature of the software tool is activated. Moreover, the Motor Configurator has the ability to automatically “lock” a motor, i. e. prevent any movement commands in the control system, if the online configuration does not match the preset. The locking feature can be toggled for each terminal ID separately in the Stage Assignment table.

Picking and assigning the correct preset from all available presets becomes increasingly cumbersome if the number of stored presets grows large. To help the user track all presets that are typically assigned to a terminal ID over multiple modifications of the experimental setup, the software provides an advanced *Preset Assignment* table. In the table a set of multiple presets can be stored for each terminal ID and one preset is selected for comparisons with the respective online configuration. Entries in the stage assignment table on the main scene are overwritten accordingly on the click of the button “Apply Preset Assignment”.

The software tool provides the option to store the configuration preset XML files in a central database in the local network. This allows access to configuration presets from multiple control machines in different experimental setups. To protect against data loss and to allow to restore the previous version of a preset, the database uses version control. In the current implementation, git [3]/gitlab [4] is used as a backend.

## CONCLUSION

The Motor Configurator software tool has been implemented to facilitate the configuration of motors in frequently changing experimental setups at EuXFEL. It is based on the in-house developed SCADA system Karabo and makes use of Karabo’s features to retrieve and modify configurations of online motors. Karabo’s scene feature is employed to ship the tool with a user-friendly graphical user interface. The tool drastically reduces the need for manual reconfigurations of motors, hence leading to a significant reduction of the time needed to configure motor setups. Moreover, the likelihood of motor misconfigurations is lowered, which, in turn, leads to a lower incidence of damage to hardware components in the vicinity of the motors.

With the present implementation of the software, existing configuration presets can be modified only by editing the respective XML file manually. Therefore, in future work,

# main scene

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

**Stage Assignment**

Terminal Id	Stage Preset ID	Lock Device on Mismatch
33 FXE_ILH_AUX/MOTOR/AUX06	ilh_um06_ta_incoupling	<input type="checkbox"/>
34 FXE_AUXT_LAS/MOTOR/LENS_X	lens_X	<input type="checkbox"/>
35 FXE_AUXT_LAS/MOTOR/LENS_Y	lens_Y	<input type="checkbox"/>
36 FXE_AUXT_LAS/MOTOR/ATTEN	topas_atten	<input type="checkbox"/>
37 FXE_AUXT_LAS/MOTOR/PERISCOPE_X	periscope_bottom	<input type="checkbox"/>
38 FXE_AUXT_LAS/MOTOR/PERISCOPE_Y	periscope_top	<input type="checkbox"/>
39 FXE_AUXT_LAS/MOTOR/GP1	mirror_X	<input type="checkbox"/>
40 FXE_AUXT_LAS/MOTOR/GP2	mirror_Y	<input type="checkbox"/>
41 FXE_AUXT_LAS/MOTOR/FOCUS	focus	<input type="checkbox"/>
42 FXE_AUXT_LAS/MOTOR/HWP800	hwp800	<input type="checkbox"/>

**Motor Parameters**

Parameter Key	Check	Save	Write	Enabled Classes
0 targetVelocity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
1 enableSWLimitLow	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
2 encoder.maxPositionLagValue	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
3 enableSWLimitHigh	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
4 swLimitLow	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
5 swLimitHigh	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
6 mc2.referenceVelocity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
7 activeController	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
8 mc2.moveDirection	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all
9 mc2.maxVelocity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	all

**result table**

Terminal Id	Changes
0 FXE_SMS_USR/MOTOR/UM16	No changes
1 FXE_AUXT_LAS/MOTOR/LENS_X	Changes detected
2 FXE_SMS_USR/MOTOR/UM07	Changes detected
3 FXE_AUXT_LAS/MOTOR/LENS_Y	Changes detected
4 FXE_SMS_USR/MOTOR/UM02	Changes detected
5 FXE_ILH_AUX/MOTOR/AUX06	No changes
6 FXE_AUXT_LAS/MOTOR/ATTEN	Changes detected
7 FXE_SMS_USR/MOTOR/UM19	Changes detected
8 FXE_ILH_AUX/MOTOR/AUX01	Changes detected
9 FXE_AUXT_LAS/MOTOR/PERISCOPE_X	Changes detected
10 FXE_SMS_USR/MOTOR/UM14	No changes
11 FXE_XTD9_MONO-2/MOTOR/TY	No changes
12 FXE_SMS_USR/MOTOR/UM03	Changes detected
13 FXE_AUXT_LAS/MOTOR/GP2	No changes
14 FXE_XTD9_MONO-1/MOTOR/TY	No changes
15 FXE_SMS_USR/MOTOR/UM18	Changes detected
16 FXE_XTD9_MONO-2/MOTOR/TX	No changes
17 FXE_SMS_USR/MOTOR/UM01	Changes detected
18 FXE_OGT2_PSLIT/MOTOR/BLADE1_IN_OUT	No changes
19 FXE_SMS_USR/MOTOR/UM15	No changes
20 FXE_XTD9_MONO-1/MOTOR/TX	No changes
21 FXE_OGT2_PSLIT/MOTOR/BLADE4_IN_OUT	No changes
22 FXE_AUXT_LAS/MOTOR/PERISCOPE_Y	Changes detected

**Gitlab**

Username:

Get From Git Push To Git Logout git user

**Preset Assignments**

Terminal ID	Stage Preset Options	Selected Preset ID
0 FXE_SMS_USR/MOTOR/UM01	um01_configA,um01_configB,svetina_sa...	um01_configA
1 FXE_SMS_USR/MOTOR/UM02	um02,svetina_det_1,kappa_Z	um02
2 FXE_SMS_USR/MOTOR/UM03	svetina_gonio_aroundX,kappa_X	svetina_gonio_aroundX
3 FXE_SMS_USR/MOTOR/UM04	liquidjet_kohzu_ZA04A,svetina_gonio_Y...	liquidjet_kohzu_ZA04A
4 FXE_SMS_USR/MOTOR/UM06	svetina_gonio_Z,kappa_phi	svetina_gonio_Z
5 FXE_SMS_USR/MOTOR/UM07	svetina_gonio_X,kappa_kappa	svetina_gonio_X
6 FXE_SMS_USR/MOTOR/UM08	svetina_magnet_Y	svetina_magnet_Y
7 FXE_SMS_USR/MOTOR/UM09	svetina_target_Z	svetina_target_Z
8 FXE_SMS_USR/MOTOR/UM10	svetina_target_X	svetina_target_X
9 FXE_SMS_USR/MOTOR/UM11	svetina_Dmirror_rot,kappa_chi	svetina_Dmirror_rot
10 FXE_SMS_USR/MOTOR/UM12	svetina_JF_Y	svetina_JF_Y
11 FXE_SMS_USR/MOTOR/UM14	svetina_gonio_aroundY	svetina_gonio_aroundY
12 FXE_SMS_USR/MOTOR/UM15	svetina_gonio_aroundZ	svetina_gonio_aroundZ
13 FXE_SMS_USR/MOTOR/UM16	svetina_det_2	svetina_det_2
14 FXE_SMS_USR/MOTOR/UM17	liquidjet_vertical,svetina_Dmirror_X	liquidjet_vertical
15 FXE_SMS_USR/MOTOR/UM18	liquidjet_alongbeam,svetina_sample_Z	liquidjet_alongbeam
16 FXE_SMS_USR/MOTOR/UM19	liquidjet_horizontal,svetina_sample_Y	liquidjet_horizontal
17 FXE_SMS_USR/MOTOR/UM20	svetina_sample_X	svetina_sample_X
18 FXE_OGT2_PSLIT/MOTOR/BLADE1_IN_OUT	pslit2_1	pslit2_1
19 FXE_OGT2_PSLIT/MOTOR/BLADE2_IN_OUT	pslit2_2	pslit2_2
20 FXE_OGT2_PSLIT/MOTOR/BLADE3_IN_OUT	pslit2_3	pslit2_3
21 FXE_OGT2_PSLIT/MOTOR/BLADE4_IN_OUT	pslit2_4	pslit2_4
22 FXE_AUXT_CSLIT/MOTOR/BLADE1_IN_OUT	cslit_1	cslit_1
23 FXE_AUXT_CSLIT/MOTOR/BLADE2_IN_OUT	cslit_2	cslit_2

Current Preset Assignment table applied  Apply Preset Assignments

**Device Configuration Preview**

Showing the configuration and changes for FXE\_SMS\_USR/MOTOR/UM01

**Changes**

Existing Configuration (Wed 27 Sep 2023, 15:25:07)	Requested Configuration (Thu 13 Jul 2023, 13:14:29)
<b>commissioning</b>	<b>commissioning</b>
gearRatio 1.0	gearRatio 28.444444
spindlePitch 1.0	spindlePitch 0.4
stepsPerRevolution 200	stepsPerRevolution 24
enableSWLimitHigh False	enableSWLimitHigh True
enableSWLimitLow False	enableSWLimitLow True
<b>homing</b>	<b>homing</b>
homingVelocityOffPlcCam 0.13636363636363635	homingVelocityOffPlcCam 0.015980113886052916
homingVelocityTowardsPlcCam 0.13636363636363635	homingVelocityTowardsPlcCam 0.015980113886052916
<b>mc2</b>	<b>mc2</b>
jerk 10.227272727272727	jerk 1.1985065414539685
maxVelocity 9.090909090909092	maxVelocity 1.0653409257368611
referenceVelocity 10.0	referenceVelocity 1.171875018310547
nominalCurrent 1500	nominalCurrent 200
reducedCurrent 300	reducedCurrent 50
swLimitHigh 50.0	swLimitHigh 100.01
swLimitLow -55.0	swLimitLow -0.01
targetVelocity 1.0	targetVelocity 0.1

configuration comparison dialog

Note: Press Apply to request to reconfigure the displayed configuration changes.

## advanced preset assignment scene

Figure 2: Motor Configurator graphical user interface/scene. Top: main scene with *Stage Assignment* table (top left), *Motor Parameters* table (left), *result table* (right) and controls for the gitlab integration (bottom left). Bottom right: Dialog window, showing the differences between the online configuration and the assigned configuration preset for a selected motor. Left: *Advanced Preset Assignment* table. More details are given in the main text.

the software should be extended by a configuration editor to allow for user-friendly and error-free modifications of existing configuration presets.

doi:10.1107/S1600577519006696

## REFERENCES

[1] S. Hauf *et al.*, “The Karabo distributed control system”, *J. Synchrotron Radiat.*, vol. 26, no. 5, pp. 1448–1461, Aug. 2019.

- [2] The Karabo SCADA Framework,  
<https://github.com/European-XFEL/Karabo>
- [3] git, <https://git-scm.com/>
- [4] Gitlab, <https://gitlab.com/>