Table of contents

Automation Core

Automation Core

Feature 395940: Data Layer extensions

Description

Additional functionalities:

- Trace added if the memory user lost the connection to the RT memory
- “timestamp” data type added
  - Time stamp of the type tree
  - Type Array of time stamp
  - Time stamp type in variant, also convertible to json and back.
- Maximum message size added; this can be configured in the datalayer/server/settings node
- Added to the Data Layer RT: nTelBuffer - n buffer systems to support n-2 RT users
  - can be configured in the *datalayer/realtime/broker/settings* node

Bugfixes:

- The provider could not reconnect during the snap installation
- A message is always send if the node is managed by the provider
- The provider disconnects the connection if the provider manages the subscription and also responds like the provider
- System crashes if the type in the MemoryMap in the Data Layer RT is not selected
- The first TCP token check after a deviceadmin restart fails as the socket is closed
- Subscription does not report an error when the INPROC connection cannot be established
- Reading of memuser of the MemoryMap can result in a crash
- Subscription of an empty node set should not fail
- createSubscription(A)sync() did not fail if no connection was established
- Subscription with identical ID does not result in an error (Bug 459232)
- Browsing returns OK but the node does not exist (Bug 451808)
- Add alias for FLOAT (FLOAT32) and DOUBLE (FLOAT64) (bug 452239)
- Deadlock in the DL Client during unsubscribe (Bug 452335)

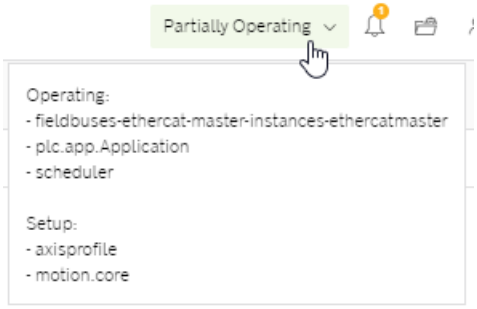
Feature 433774: System state of the ctrlX CORE Runtime

Description

System state

The state of all ctrlX CORE Runtime components (PLC, EtherCAT, Motion, Axis Profile, Profinet) are summarized to a system state. This new system state replaces the Service Mode.

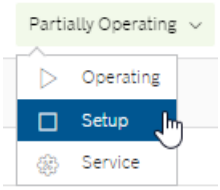
The SERVICE, SETUP, PARTIAL_OPERATING and OPERATING states are possible.



The system state can be read out via the Data Layer node *system/state*. The ctrlX CORE Runtime component states are represented below this node.

Switching the state

The ctrlX CORE Runtime component state can be changed via the ctrlX Scheduler.



By changing the scheduler state (*scheduler/admin/state*), the tasks as well as their callables are switched.

By switching the callables, the subordinate component of the callable is also switched to the corresponding state.

In the SERVICE state, the tasks are not executed and the hardware watchdog as well as the task watchdogs are disabled. Actions having an impact on the real-time can be executed in this state. Amongst others, these actions can include the installation and uninstallation of apps.

The SETUP state is used to configure the components. This state can be required to load a configuration. The system is should be in state OPERATING during operation.

LED state

state	color	
System state „SERVICE/SETUP“	blue	BU
System state „ PARTIAL_OPERATING“	flashing green	GN|GN|GN|GN|GN -|-|-|-|-
System state „ OPERATING“	green	GN

Necessity for switching

Switching of the state is not always required. It is required if apps were installed having special requirements on the real-time. For example, these apps can include a Motion or PLC app. Switching is also required if a task was configured with a task watchdog in the scheduler.

Via the *scheduler/admin/info/strict-mode* Data Layer node it can be read out if switching is necessary. The widget to display and switch the state is only displayed if required.

Feature 435362: Handling large data volumes in the Data Layer interface

Description

Nodes in the Data Layer with a large number of subnodes (several thousand nodes) are identified by a filter symbol. These node are handled individually during the operation and display on the interface.

Reason: the representation and operation of large data volumes is not user-friendly and the interface components are not intended to display large data volumes.

Feature 448724: Diagnostic extensions for Release 1.16

Description

From release 1.16 it is now possible to use the bundle interface of the common.log.diagnosis bundles outside the app.automationcore snap context with an individual celix framework. The following entry has to be added to the celix configuration file "config.properties”:

diagnosis.broker.enable=false

All available C++/C interfaces are mapped to app.automationcore via Data Layer Client calls.

Supported interfaces:

- i_registration3.h → registering and unregistering diagnostics
- i_log2.h → reporting diagnostic logs
- pending_diagnostics_itf.h → methods and information about the pending diagnostics

Unsupported interfaces:

- distributor/i_registration.h → messages about certain diagnostic events are contained
- i_registration.h → outdated, instead use i_registration3.h
- i_log.h → outdated, instead use i_log2.h

If app.automationcore is not available, the diagnostic bundle runs into fallback mode. The fallback mode also allows registering/unregistering as well as reporting of diagnostics so that messages do not get lost.