XDOP Device Class Description

Document Version 1.0

Includes Descriptions for following devices:

- urn:schemas-coscom-org:device:MCU5Ladder:1

creation date: Thursday, September 24, 2009

Table of Contents

1 Device class MCU5Ladder

Namespace: XDOP.schemas_coscom_org.device
Assembly: MCU5Ladder.dll

Syntax

```
public class MCU5Ladder : IDeviceObject
```

Constructors

MCU5Ladder()	Initializes a new instance of the class.

Device Properties

Property name	Value
DeviceType	urn:schemas-coscom-org:device:MCU5Ladder:1
EPC	
FriendlyName	Ladder ergometer
Manufacturer	h/p/cosmos
ManufacturerURL	www.h-p-cosmos.com
ModelDescription	
ModelName	
ModelNumber	
ModelURL	
SerialNumber	

Common Properties

Property name	Description
AcceptNewerVersions	Indicates if the device object accepts newer versions of the device type.
LazyLoading	If set, the device loads its services not until they are really needed.
XDOPDevice	Gets the XDOPLib.IDevice Interface of the device object if already connected
Services	Get's a read-only collection of the device services. For a list of services see 'Services'
ConnectionStatus	Gets the actual XDOPLib.ConnectionStatus object for this device.

Device Services

Following list enumerates the service (type) objects of the 'Services' Property.

- urn:schemas-coscom-org:service:SunTechTangoStressBPBridge:1
- urn:schemas-coscom-org:service:TreadmillProfileEditor:1
- urn:schemas-coscom-org:service:TreadmillGeneralConfig:1
- urn:schemas-coscom-org:service:TreadmillUserControl:1
- urn:schemas-coscom-org:service:TreadmillOptions:1
- urn:schemas-coscom-org:service:TreadmillDirectControl:1
- urn:schemas-coscom-org:service:TerminalLCD6x4Panel:1
- urn:schemas-coscom-org:service:TreadmillMeasures:1
- urn:schemas-coscom-org:service:MCU5FlashUpdate:1
- urn:schemas-coscom-org:service:DataLogger:1

Common Methods

Name	Description
void Connect(ITransportObject transportObject);	Connects to a device.
void Disconnect();	Closes a established the connection.
ProtocolParameter GetProtocolParameter();	Gets the protocol parameter of the current connection.
ITransportObject GetTransportObject();	Gets the ITransportObject object of the current connection.
void DisposeDevice();	Disposes the current device object.

1.1 Service class SunTechTangoStressBPBridge

This service manages a communication bridge to the SunTech Tango Blood Pressure device. If the Tango device is connected with one of the other serial ports, it can be controlled with the actions of this service.
See SunTech Tango manual for more details or contact h/p/cosmos.

Namespace: XDOP.schemas_coscom_org.service
Assembly: SunTechTangoStressBPBridge.dll

Syntax

```
public class SunTechTangoStressBPBridge : IServiceObject
```

Constructors

SunTechTangoStressBPBridge(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:TangoBP
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:SunTechTangoStressBPBridge:1

Service Properties and Events

Type	Name	Flags	Description
System.String	Message	E G	Message string from the Tango blood pressure device

Service Methods

Name	Description
BeginBPStudy	Begin blood pressure study.
ClearBPReadings	Clear blood pressure readings.
EndBPStudy	End blood pressure study.
RecallBPValues	Recall blood pressure values.
StartBPSample	Start blood pressure sample.
StopBPSample	Stop blood pressure sample.

Service error codes

Error code	Error description
823	No SunTech BP monitor

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.1.1 Actions

1.1.1.1 Action BeginBPStudy

Begin blood pressure study.

Syntax

```
public void BeginBPStudy()
```

Action Errors

823	No SunTech BP monitor

1.1.1.2 Action ClearBPReadings

Clear blood pressure readings.

Syntax

```
public void ClearBPReadings()
```

Action Errors

823	No SunTech BP monitor

1.1.1.3 Action EndBPStudy

End blood pressure study.

Syntax

```
public void EndBPStudy()
```

Action Errors

823	No SunTech BP monitor

1.1.1.4 Action RecallBPValues

Recall blood pressure values.

Syntax

```
public void RecallBPValues()
```

Action Errors

823	No SunTech BP monitor

1.1.1.5 Action StartBPSample

Start blood pressure sample.

Syntax

```
public void StartBPSample()
```

Action Errors

823	No SunTech BP monitor

1.1.1.6 Action StopBPSample

Stop blood pressure sample.

Syntax

```
public void StopBPSample()
```

Action Errors

823	No SunTech BP monitor

1.2 Service class TreadmillProfileEditor

With the profile editor all stored profiles can be read and the user profiles can be edited and saved. There are three types of profiles in the treadmill:

- Scalable profiles, which can be started in the mode "profile". This profiles are read-only.

Profile numbers:
1 to 6

- Test profiles, which provide specific tests. This profiles are read-only.
Profile numbers:
4 = Bruce
5 = Naughton
6 = Balke
8 = Ellestad A
9 = Ellestad B
21 = User Test Profile 1
22 = User Test Profile 1
23 = User Test Profile 1
24 = User Test Profile 1
70 = MBAFC Interval 600m / 14.5 - 17.5 km/h
71 = MBAFC Interval 500m / 15 - 19 km/h
72 = MBAFC Interval 400m / 16 - 21 km/h
73 = MBAFC Interval 300-600m / 15 - 21 km/h
74 = MBAFC Interval 300-600m / 15 - 21 km/h
75 = MBAFC Interval 300m / 20 km/h
76 = walk protocol 13 steps
80 = VO2 / 10K
81 = VO2 / 11K
82 = VO2 / 12K
84 = VO2 / 14K
85 = SUPER BALKE
90 = Test 90
91 = Test 91
92 = Test 92
93 = Test 93
94 = Test 94
99 = Burn-in

- User profiles, which can be edited and saved.
Profile numbers:
21 to 24 MCU4 (MCU5 28)

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillProfileEditor.dll

Syntax

```
public class TreadmillProfileEditor : IServiceObject
```

Constructors

TreadmillProfileEditor(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:Profiles
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillProfileEditor:1

Service Properties and Events

No Variables

Service Methods

Name	Description
GetProfileHeader	Gets profile header information.
ReadProfileStep	Reads profile step information.
WriteUserProfileStep	Writes user test profile step information.

Service error codes

Error code	Error description
800	Not in select mode
804	Invalid profile
813	Invalid test number
821	Invalid step
822	Invalid profile type

Details

On the following pages you will find a detailed description
for every action and variable of this service.

1.2.1 Actions

1.2.1.1 Action GetProfileHeader

Gets profile header information.

Syntax

```
public void GetProfileHeader(
   System.Byte profileType
   System.Byte number
   ref System.Byte steps
   ref System.Byte intervalType
   ref System.UInt16 intervalSum
   ref System.Single maxSpeed
   ref System.Single maxGrade
)
```

Parameters
profileType
Profile type:
0 = Factory profile (scalable)
1 = Test profile
2 = User profile

number
Profile identification number

steps
Number of steps (max. step index)

intervalType
Interval type:
0 = mixed
1 = Time
2 = Distance

intervalSum
Sum of all step intervals (interval type is time or distance)

maxSpeed
Maximum speed of all steps

maxGrade
Maximum grade of all steps

Remarks

The returned header informations is type-dependent:

Profile type: 0 = Factory profile (scalable)
steps = number of stored steps
intervalType = Time (1) or Distance (2)

maxSpeed = maximum speed of all steps
maxGrade = maximum grade of all steps

Profile type: 1 = Test profile
steps = number of stored steps
intervalType = Mixed (0)
maxSpeed = 0 (no header information available)
maxGrade = 0 (no header information available)

Profile type: 2 = User profile
steps = 10 (maximum possible number, not the actual number!)
intervalType = Mixed (0)
maxSpeed = 0 (no header information available)
maxGrade = 0 (no header information available)

Action Errors

822	Invalid profile type
804	Invalid profile
800	Not in select mode
813	Invalid test number

1.2.1.2 Action ReadProfileStep

Reads profile step information.

Syntax

```
public void ReadProfileStep(
   System.Byte profileType
   System.Byte number
   System.Byte step
   ref System.Byte intervalType
   ref System.UInt16 interval
   ref System.Single speed
   ref System.Byte accelIndex
   ref System.Single grade
)
```

Parameters
profileType
Profile type:
0 = Factory profile (scalable)
1 = Test profile
2 = User profile

number
Profile identification number

step
Step index (1 ... max. steps)

intervalType
Interval type:

1 = Time
2 = Distance

interval
Time [s] or distance [m]

speed
Speed [m/s]

accelIndex
Acceleration index (1 to 4)

grade
Grade [%]

Action Errors

822	Invalid profile type
804	Invalid profile
821	Invalid step
800	Not in select mode
813	Invalid test number

1.2.1.3 Action WriteUserProfileStep

Writes user test profile step information.

Syntax

```
public void WriteUserProfileStep(
   System.Byte number
   System.Byte step
   System.Byte intervalType
   System.UInt16 interval
   System.Single speed
   System.Byte accelIndex
   System.Single grade
)
```

Parameters
number
Profile identification number

step
Step index (1 ... max. steps)

intervalType
Interval type:
1 = Time
2 = Distance

interval
Time [s] or distance [m]

speed

Speed [m/s]

accelIndex
Acceleration index (1 to 4)

grade
Grade [%]

Action Errors

804	Invalid profile
821	Invalid step
800	Not in select mode

1.3 Service class TreadmillGeneralConfig

This service can be used to get general information about the treadmill and to make general configurations.

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillGeneralConfig.dll

Syntax

```
public class TreadmillGeneralConfig : IServiceObject
```

Constructors

TreadmillGeneralConfig(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:GeneralConfig
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillGeneralConfig:1

Service Properties and Events

Type	Name	Flags	Description
System.Byte	Error	E G	General control error. See action GetErrors for more details.

Service Methods

Name	Description
GetDefaultType	Gets the default type and sub type, which has been set by the factory. This values are important, if the configuration parameters (options) of the treadmill are lost (e.g. the battery of the Real-Time-Clock has been replaced or the signature of the configuration memory is false).
GetErrors	Gets all three types of error codes: user control error, drive error and elevator error.
GetFirmwareVersion	Gets the firmware version.
GetLockMask	Gets the lock bit mask. Each bit presents a lock for a control mode. If a control mode is locked, the user cannot select and start this mode.
GetOEMCode	Gets the OEM code of this treadmill. Please contact h/p/cosmos for further details.
GetRTCTime	Gets the time fields of the battery-buffered real time clock.
GetSerialNumber	Get the serial number of the treadmill. Contact h/p/cosmos for details.

GetType	Gets the type and sub type (treadmill type and variant). Contact h/p/cosmos for a list of all available types and sub types.
SetFailSafeTimeout	This is a very important action, which should be used to set a failsafe timeout before the treadmill is remote controlled.
SetProtocolParams	Sets optional protocol parameters.
SetRTCTime	Gets the time fields of the battery-buffered real time clock.
SetSerialNumber	Get the serial number of the treadmill. Contact h/p/cosmos for details. This action needs two access keys for a successful change.

Service error codes

Error code	Error description
801	Wrong access keys

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.3.1 Actions

1.3.1.1 Action GetDefaultType

Gets the default type and sub type, which has been set by the factory. This values are important, if the configuration parameters (options) of the treadmill are lost (e.g. the battery of the Real-Time-Clock has been replaced or the signature of the configuration memory is false).

Syntax

```
public void GetDefaultType(
   ref System.Byte type
   ref System.Byte subType
)
```

Parameters
type
Major type (major treadmill type)

subType
Sub type (treadmill type variant)

1.3.1.2 Action GetErrors

Gets all three types of error codes: user control error, drive error and elevator error.

Syntax

```
public void GetErrors(
   ref System.Byte controlError
   ref System.Byte driveError
   ref System.Byte elevatorError
)
```

Parameters
controlError
General control error

driveError
Drive error

elevatorError
Elevator error

Remarks

Control errors:

0 = No error
1 = Oil interval passed
2 = Service interval passed

20 = Elevator top crash
21 = Elevator increment error
30 = Drive increment error
41 = RTC initialized (backup battery weak)
50 = FU error

Drive errors:

0 = No error
1 = Drive increment error (=> control error 30)
2 = FU error (=> control error 50)

Elevator errors:

0 = No error
1 = Elevator increment error (=> control error 21)
2 = Elevator top crash (=> control error 20)

1.3.1.3 Action GetFirmwareVersion

Gets the firmware version.

Syntax

```
public void GetFirmwareVersion(
   ref System.String version
)
```

Parameters
version
A string which is structured like this:
Major version.subversion 1.subversion 2.Buildnumber.optional appendix

Examples: 4.04.1.0011, 4.04.1.0012.Test

1.3.1.4 Action GetLockMask

Gets the lock bit mask. Each bit presents a lock for a control mode. If a control mode is locked, the user cannot select and start this mode.

Syntax

```
public System.Byte[] GetLockMask(
   ref System.Byte[] mask
)
```

Parameters
mask
Lock mask:
0x0F = General
0x10 = Manual
0x20 = Profile
0x40 = Cardio
0x80 = Test

1.3.1.5 Action GetOEMCode

Gets the OEM code of this treadmill. Please contact h/p/cosmos for further details.

Syntax

```
public System.Byte GetOEMCode(
   ref System.Byte code
)
```

Parameters
code
OEM code number

1.3.1.6 Action GetRTCTime

Gets the time fields of the battery-buffered real time clock.

Syntax

```
public void GetRTCTime(
   ref System.Byte sec
   ref System.Byte min
   ref System.Byte hrs
   ref System.Byte day
   ref System.Byte month
   ref System.UInt16 year
)
```

Parameters
sec
RTC seconds: 0 - 59

min
RTC minutes: 0 - 59

hrs
RTC hours: 0 - 23

day
RTC day: 1 - 31

month

RTC month: 1 - 12

year
RTC year: 2000 - 2099

1.3.1.7 Action GetSerialNumber

Get the serial number of the treadmill. Contact h/p/cosmos for details.

Syntax

```
public void GetSerialNumber(
  ref System.Char code1
  ref System.Char code2
  ref System.Char code3
  ref System.UInt32 num8
)
```

Parameters
code1
Code character 1

code2
Code character 2

code3
Code character 2

num8
Serial number between 0 and 99999999

1.3.1.8 Action GetType

Gets the type and sub type (treadmill type and variant). Contact h/p/cosmos for a list of all available types and sub types.

Syntax

```
public void GetType(
  ref System.Byte type
  ref System.Byte subType
)
```

Parameters
type
Major type (major treadmill type)

subType
Sub type (treadmill type variant)

1.3.1.9 Action SetFailSafeTimeout

This is a very important action, which should be used to set a failsafe timeout before the treadmill is remote controlled.

Syntax

```
public void SetFailSafeTimeout(
   System.Byte newTimeout
   ref System.Byte oldTimeout
)
```

Parameters
newTimeout
New timeout value in 1/10 seconds (0 to 25.5 seconds)

oldTimeout
Old timeout value in 1/10 seconds (0 to 25.5 seconds)

Remarks

A timeout value of 0 disables the timeout timer and a timeout value between 1 and 255 (0.1 to 25.5 seconds) enables the timer.

The timeout timer increments every 100 milliseconds.
If the set timeout value is reached, the treadmill is stopped (drive and elevator are stopped and the current mode is leaved).

The timeout timer resets and doesn't count up, if the drive is stopped and the elevator doesn't go up or down.

For safety reasons the timeout value should always be set, if the speed or the elevation of the treadmill is changed by an action message.
A reasonable value for the timeout is 1 second (timeout value = 10).

1.3.1.10 Action SetProtocolParams

Sets optional protocol parameters.

Syntax

```
public void SetProtocolParams(
   System.Boolean processChecksum
   System.Boolean appendCRLF
   System.Boolean outputDescNames
)
```

Parameters

processChecksum
Enables the processing of an 2 hex digit (modulo 256) checksum with element Y0.

appendCRLF
Appends the characters CR and LF at the end of a message (after *Z).

outputDescNames
Enables the output of the optional index names in the description lines.

1.3.1.11 Action SetRTCTime

Gets the time fields of the battery-buffered real time clock.

Syntax

```
public void SetRTCTime(
   System.Byte sec
   System.Byte min
   System.Byte hrs
   System.Byte day
   System.Byte month
   System.UInt16 year
)
```

Parameters
sec
RTC seconds: 0 - 59

min
RTC minutes: 0 - 59

hrs
RTC hours: 0 - 23

day
RTC day: 1 - 31

month
RTC month: 1 - 12

year
RTC year: 2000 - 2099

1.3.1.12 Action SetSerialNumber

Get the serial number of the treadmill. Contact h/p/cosmos for details. This action needs two access keys for a successful change.

Syntax

```
public void SetSerialNumber(
   System.Char code1
   System.Char code2
   System.Char code3
   System.UInt32 num8
```

```
  System.Byte key1
  System.Byte key2
)
```

Parameters

code1
Code character 1

code2
Code character 2

code3
Code character 3

num8
Serial number between 0 and 99999999

key1
Access key 1

key2
Access key 2

Action Errors

801	Wrong access keys

1.4 Service class TreadmillUserControl

This service instantiates a higher level control interface in contrast to the terminal or direct control interface. It models mainly the user control concept of the user terminal. For example one can start a specific test, which is implemented in the treadmill computer.

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillUserControl.dll

Syntax

```
public class TreadmillUserControl : IServiceObject
```

Constructors

TreadmillUserControl(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:UserControl
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillUserControl:1

Service Properties and Events

Type	Name	Flags	Description
System.Byte	ControlStatus	E G	Control status: 0 = Stop 1 = Run 2 = Pause 8 = Graded test pause freezed 9 = Emergency stop
System.Byte	HrcStatus	E G	Actual heart rate control status: 0 = idle 1 = heart rate too low - load is increased 2 = heart rate too high - load is decreased 3 = hearte rate is okay 4 = failure
System.Byte	Mode	E G	Actual user control mode: 1 = Select 2 = Manual 3 = Profile 4 = Cardio 5 = Test 6 = User Options 7 = Administrator Options
System.Byte	NextStepCountdown	E G	Countdown time value [1/10 s] until next step starts
System.Byte	Step	E G	Current step (1 ... 255), if a profile or test is processed

Service Methods

Name	Description
Cooldown	Starts the cooldown phase. After the cooldown time (duration) the treadmill stops.
Countdown	Switches the time display mode from incrementing to decrementing (countdown). The time arguments set the time until the countdown expires. Then the treadmill optionally stops or the time increments again (starts with the the total time).
DecGrade	Decreases the treadmill grade.
DecSpeed	Decreases the treadmill speed (deceleration).
FreezeStepperTestPause	Freezes the pause of the stepper test (sets the pause time to infinite).
GetHRCStatus	Gets the actual status of the heart rate control.
GetNextStepStatus	Gets the status values of the next step. See action ShowNextStep.
GetPersonData	Gets all person data.
GetProfileStatus	Gets the actual profile status values.
GetProfListEntry	Gets a profile entry.
GetProfListSize	Gets number of stored profiles (profile list size). To get an entry use action GetProfileListEntry.
GetTestListEntry	Gets a test entry.
GetTestListSize	Gets number of implemented test (test list size). To get an entry use action GetTestListEntry.
GetTestStatus	Gets the actual test status, if a test is running.
GetUKKTestResult	Gets the UKK test result at the end of the test.
HoldGrade	Holds the actual grade (switches the elevator off).
HoldSpeed	Holds the actual speed (no acceleration or deceleration).
IncGrade	Increases the treadmill grade.
IncSpeed	Increases the treadmill speed (acceleration).
Pause	Same as DecSpeed. If the treadmill reaches the speed zero, the speed display shows "PAUS" and the incrementing of the run time is stopped until the treadmill accelerates again.
ProfileStepBack	Steps one profile step back, if a profile is currently running (profile mode or test mode with a test profile).
ProfileStepForward	Steps one profile step forward, if a profile is currently running (profile mode or test mode with a test profile).
QuickStop	Same as action "Stop", but with maximum deceleration.
SetPersonData	Sets all person data.
SetProfileStepInterval	Sets the interval end time or distance of the current profile step.

ShowNextStep	Shows next speed and/or grade values as long as the countdown runs or infinite. At the end of the countdown the actual speed and/or grade are not set. This have to be done with another action.
StartHeartRateControl	Starts heart rate control, which increases or decreases the speed and elevation to get the heart rate between the allowed range (minimum and maximum heart rate).
StartManual	Start manual mode.
StartNextStepperTestInterval	Starts next stepper test intervall (Finishes the pause interval).
StartProfile	Start a profile with scale parameters.
StartTestConconi	Starts the conconi test.
StartTestCooper	Starts the cooper test.
StartTestGardner	Starts the gardner test.
StartTestProfile	Starts a profile-based test.
StartTestRamp	Starts the ramp test.
StartTestStepper	Starts stepper test.
StartTestUKK	Starts the UKK Test.
Stop	Stops the treadmill and the actual mode. It's the same as pressing the key "Stop" at the user terminal.
TestCooperStepBack	Jumps one cooper test step back. Only valid if a cooper test is running.
TestCooperStepForward	Jumps one cooper test step forward. Only valid if a cooper test is running.

Service error codes

Error code	Error description
802	No cooldown possible
803	No manual mode possible
804	Invalid profile
805	No OEM profile
806	No profile mode possible
807	No cardio mode possible
808	No test mode possible
809	Invalid UKK test parameters
810	No UKK test active
811	No stepper test active
812	No cooper test active
813	Invalid test number
814	Invalid person parameter
815	Invalid nextstep mode
816	No profile started
817	HRC not started
818	Test not started
831	Speed out of range
832	Grade out of range

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.4.1 Actions

1.4.1.1 Action Cooldown

Starts the cooldown phase. After the cooldown time (duration) the treadmill stops.

Syntax

```
public void Cooldown(
   System.Single speed
   System.Single grade
   System.UInt16 duration
)
```

Parameters
speed
Cooldown speed [m/s].

grade
Cooldown grade [%].

duration
Cooldown duration [s].

Action Errors

832	Grade out of range
802	No cooldown possible
831	Speed out of range

1.4.1.2 Action Countdown

Switches the time display mode from incrementing to decrementing (countdown). The time arguments set the time until the countdown expires. Then the treadmill optionally stops or the time increments again (starts with the the total time).

Syntax

```
public void Countdown(
   System.Byte hrs
   System.Byte min
   System.Byte sec
   System.Boolean stopFlag
)
```

Parameters
hrs
Countdown hours

min
Countdown minutes

sec
Countdown seconds

stopFlag
Flag to stop the treadmill, when the countdown is expired.

1.4.1.3 Action DecGrade

Decreases the treadmill grade.

Syntax

```
public void DecGrade()
```

1.4.1.4 Action DecSpeed

Decreases the treadmill speed (deceleration).

Syntax

```
public void DecSpeed(
   System.Byte accelIndex
)
```

Parameters
accelIndex
Index (1 - 7) for a deceleration table with 7 values.
1 = lowest deceleration
...
7 = highest decelartion

1.4.1.5 Action FreezeStepperTestPause

Freezes the pause of the stepper test (sets the pause time to infinite).

Syntax

```
public void FreezeStepperTestPause(
   ref System.Boolean freezed
)
```

Parameters
freezed

Action Errors

808	No test mode possible

1.4.1.6 Action GetHRCStatus

Gets the actual status of the heart rate control.

Syntax

```
public void GetHRCStatus(
   ref System.Byte heartRate
   ref System.Byte controlState
   ref System.Byte minHeartRate
   ref System.Byte maxHeartRate
)
```

Parameters
heartRate
Actual heart rate

controlState
Actual heart rate control status:
0 = idle
1 = heart rate too low - load is increased
2 = heart rate too high - load is decreased
3 = hearte rate is okay
4 = failure

minHeartRate
Actual minimum heart rate

maxHeartRate
Actual maximum heart rate

Action Errors

817	HRC not started

1.4.1.7 Action GetNextStepStatus

Gets the status values of the next step. See action ShowNextStep.

Syntax

```
public void GetNextStepStatus(
   ref System.Byte mode
   ref System.Single speed
   ref System.Single grade
   ref System.Byte countdown
)
```

Parameters
mode
0 = off
1 = display speed value
2 = display grade value
3 = display both values

speed
Speed value [m/s]

grade
Grade value [%]

countdown
Display time in seconds.
0 = show next values infinite
1 - 25 = with countdown peeps and automatic clearing of the shown values

1.4.1.8 Action GetPersonData

Gets all person data.

Syntax

```
public void GetPersonData(
   ref System.Char sex
   ref System.Byte age
   ref System.Byte weight
   ref System.Byte size
)
```

Parameters
sex
Person sex:
'F' = female
'M' = make

age
Person age [years]

weight
Person weight [kg]

size
Person size [cm]

1.4.1.9 Action GetProfileStatus

Gets the actual profile status values.

Syntax

```
public void GetProfileStatus(
   ref System.Byte number
   ref System.Byte step
   ref System.Single speed
   ref System.Single grade
   ref System.Byte intervalType
   ref System.UInt16 intervalStart
   ref System.UInt16 intervalEnd
)
```

Parameters
number
Identification number of the profile

step
Current step number (1 ... 255)

speed
Actual speed [m/s]

grade
Actual grade [%]

intervalType
Interval type:
1 = time
2 = distance

intervalStart
Interval start time [s] or distance [m]

intervalEnd
Interval end time [s] or distance [m]

Action Errors

816	No profile started

1.4.1.10 Action GetProfListEntry

Gets a profile entry.

Syntax

```
public void GetProfListEntry(
   System.Byte index
   ref System.Byte number
   ref System.Byte intervalType
   ref System.Boolean allowed
)
```

Parameters
index
Profile list index (0 to GetProfileListSize() - 1)

number
Displayed (identification) number

intervalType
Profile interval type:
0 = time and distance steps
1 = time steps
2 = distance steps

allowed
True, if this profile can be started
False, if this profile is not allowed by the actual treadmill configuration

1.4.1.11 Action GetProfListSize

Gets number of stored profiles (profile list size). To get an entry use action GetProfileListEntry.

Syntax

```
public System.Byte GetProfListSize(
   ref System.Byte size
)
```

Parameters
size
Number of stored profiles

1.4.1.12 Action GetTestListEntry

Gets a test entry.

Syntax

```
public void GetTestListEntry(
   System.Byte index
   ref System.String name
   ref System.Byte number
   ref System.Byte testType
   ref System.Boolean allowed
)
```

Parameters
index
Test list index (0 to GetTestListSize() - 1)

name
Test name

number
Displayed (identification) number

testType
Test type:
1 = complex
2 = factory profile
3 = user profile

allowed
True, if this test can be started
False, if this test is not allowed by the actual treadmill configuration

1.4.1.13 Action GetTestListSize

Gets number of implemented test (test list size). To get an entry use action GetTestListEntry.

Syntax

```
public System.Byte GetTestListSize(
   ref System.Byte size
)
```

Parameters
size
Number of implemented tests

1.4.1.14 Action GetTestStatus

Gets the actual test status, if a test is running.

Syntax

```
public void GetTestStatus(
   ref System.String name
   ref System.Byte number
   ref System.Byte testType
   ref System.Byte step
)
```

Parameters
name
Test name

number
Test identification number

testType
Test type:
1 = Complex
2 = Profile
3 = User Profile

step
Current step number (1 ... 255)

Action Errors

818	Test not started

1.4.1.15 Action GetUKKTestResult

Gets the UKK test result at the end of the test.

Syntax

```
public void GetUKKTestResult(
   ref System.Byte heartRate500
   ref System.Byte heartRate1000
   ref System.Byte heartRate1500
   ref System.Byte heartRate2000
   ref System.Byte heartRateAverage
   ref System.UInt16 time2000
   ref System.Byte index
)
```

Parameters
heartRate500
Heart Rate at 500 meters

heartRate1000
Heart Rate at 1000 meters

heartRate1500
Heart Rate at 1500 meters

heartRate2000
Heart Rate at 2000 meters

heartRateAverage
Heart rate average

time2000
Time [s] for test length of 2000 meters

index
UKK test index
Interpretation of the index:
< 70: Considerably below average fitness
70 - 89: Slightly below average fitness
90 - 110: Average fitness
111 - 130: Slightly above average fitness
> 130: Considerably above average fitness

Action Errors

810	No UKK test active

1.4.1.16 Action HoldGrade

Holds the actual grade (switches the elevator off).

Syntax

```
public void HoldGrade()
```

1.4.1.17 Action HoldSpeed

Holds the actual speed (no acceleration or deceleration).

Syntax

```
public void HoldSpeed()
```

1.4.1.18 Action IncGrade

Increases the treadmill grade.

Syntax

```
public void IncGrade()
```

1.4.1.19 Action IncSpeed

Increases the treadmill speed (acceleration).

Syntax

```
public void IncSpeed(
  System.Byte accelIndex
)
```

Parameters
accelIndex
Index (1 - 7) for an acceleration table with 7 values.
1 = lowest acceleration
...
7 = highest accelartion

1.4.1.20 Action Pause

Same as DecSpeed. If the treadmill reaches the speed zero, the speed display shows "PAUS" and the incrementing of the run time is stopped until the treadmill accelerates again.

Syntax

```
public void Pause()
```

1.4.1.21 Action ProfileStepBack

Steps one profile step back, if a profile is currently running (profile mode or test mode with a test profile).

Syntax

```
public void ProfileStepBack()
```

Action Errors

806	No profile mode possible

1.4.1.22 Action ProfileStepForward

Steps one profile step forward, if a profile is currently running (profile mode or test mode with a test profile).

Syntax

```
public void ProfileStepForward()
```

Action Errors

806	No profile mode possible

1.4.1.23 Action QuickStop

Same as action "Stop", but with maximum deceleration.

Syntax

```
public void QuickStop()
```

1.4.1.24 Action SetPersonData

Sets all person data.

Syntax

```
public void SetPersonData(
   System.Char sex
   System.Byte age
   System.Byte weight
   System.Byte size
)
```

Parameters
sex
Person sex:
'F' = female
'M' = male

age
Person age [years]

weight
Person weight [kg]

size
Person size [cm]

Action Errors

814	Invalid person parameter

1.4.1.25 Action SetProfileStepInterval

Sets the interval end time or distance of the current profile step.

Syntax

```
public void SetProfileStepInterval(
   System.UInt16 intervalEnd
)
```

Parameters
intervalEnd
Interval end time [s] or distance [m]

Action Errors

816	No profile started

1.4.1.26 Action ShowNextStep

Shows next speed and/or grade values as long as the countdown runs or infinite. At the end of the countdown the actual speed and/or grade are not set. This have to be done with another action.

Syntax

```
public void ShowNextStep(
   System.Byte mode
   System.Single speed
   System.Single grade
   System.Byte time
)
```

Parameters
mode
The mode determines which values are shown:
0 = off (no argument values are shown and a running countdown is canceled)
1 = display speed value
2 = display grade value
3 = display both values

speed
Speed value [m/s]

grade
Grade value [%]

time
Display time in seconds.
0 = show next values infinite

1 - 25 = with countdown peeps and automatic clearing of the shown values

Action Errors

815	Invalid nextstep mode

1.4.1.27 Action StartHeartRateControl

Starts heart rate control, which increases or decreases the speed and elevation to get the heart rate between the allowed range (minimum and maximum heart rate).

Syntax

```
public void StartHeartRateControl(
  System.Single maxSpeed
  System.Byte maxHeartrate
  System.Byte minHeartrate
)
```

Parameters
maxSpeed
Maximum speed to get a higher heart rate. If this speed value isn't sufficient for the heart rate control, then the grade is increased.

maxHeartrate
Maximum heart rate

minHeartrate
Minimum heart rate

Action Errors

831	Speed out of range
807	No cardio mode possible

1.4.1.28 Action StartManual

Start manual mode.

Syntax

```
public void StartManual()
```

Action Errors

803	No manual mode possible

1.4.1.29 Action StartNextStepperTestInterval

Starts next stepper test intervall (Finishes the pause interval).

Syntax

```
public void StartNextStepperTestInterval()
```

Action Errors

811	No stepper test active

1.4.1.30 Action StartProfile

Start a profile with scale parameters.

Syntax

```
public void StartProfile(
   System.Byte number
   System.Byte scaleSpeed
   System.Byte scaleGrade
   System.Byte scaleInterval
)
```

Parameters

number
Profile identification (displayed) number

scaleSpeed
Speed scale parameter. Allowd values:
60 = 60 %
80 = 80 %
100 = 100 %
120 = 120 %
140 = 140 %
160 = 160 %

scaleGrade
Grade scale parameter. Allowd values:
60 = 60 %
80 = 80 %
100 = 100 %
120 = 120 %
140 = 140 %
160 = 160 %

scaleInterval
Interval scale parameter. Allowd values:
60 = 60 %
80 = 80 %
100 = 100 %
120 = 120 %
140 = 140 %
160 = 160 %

Action Errors

804	Invalid profile
806	No profile mode possible
805	No OEM profile

1.4.1.31 Action StartTestConconi

Starts the conconi test.

Syntax

```
public void StartTestConconi(
  System.Single startSpeed
  System.UInt16 interval
  System.Single speedIncrement
)
```

Parameters
startSpeed
Start speed [m/s]

interval
Interval distance [m] of test steps

speedIncrement
Speed increment [m/s] between the test steps

Action Errors

808	No test mode possible
831	Speed out of range

1.4.1.32 Action StartTestCooper

Starts the cooper test.

Syntax

```
public void StartTestCooper()
```

Action Errors

808	No test mode possible

1.4.1.33 Action StartTestGardner

Starts the gardner test.

Syntax

```
public void StartTestGardner()
```

Action Errors

808	No test mode possible

1.4.1.34 Action StartTestProfile

Starts a profile-based test.

Syntax

```
public void StartTestProfile(
   System.Byte number
)
```

Parameters
number
   Test identification number

Action Errors

808	No test mode possible
813	Invalid test number

1.4.1.35 Action StartTestRamp

Starts the ramp test.

Syntax

```
public void StartTestRamp(
   System.Single endSpeed
   System.Byte interval
)
```

Parameters
endSpeed
   End speed [m/s]

interval
   Interval time [s]

Action Errors

808	No test mode possible
831	Speed out of range

1.4.1.36 Action StartTestStepper

Starts stepper test.

Syntax

```
public void StartTestStepper(
   System.Single startSpeed
   System.UInt16 intervalTime
   System.Single speedIncrement
   System.Byte accelIndex
   System.UInt16 pauseTime
)
```

Parameters
startSpeed
Start speed [m/s]

intervalTime
Interval time [s]

speedIncrement
Speed increment [m/s]

accelIndex
Acceleration/Deceleration index (1 to 7)

pauseTime
Pause time [s]

Action Errors

808	No test mode possible
831	Speed out of range

1.4.1.37 Action StartTestUKK

Starts the UKK Test.

Syntax

```
public void StartTestUKK(
  System.Char sex
  System.Byte age
  System.Byte weight
  System.Byte size
)
```

Parameters
sex
Sex of the running person:
'F' = female
'M' = male

age
Person age [years]

weight
Person weight [kg]

size
Person size [cm]

Action Errors

808	No test mode possible
809	Invalid UKK test parameters

1.4.1.38 Action Stop

Stops the treadmill and the actual mode. It's the same as pressing the key "Stop" at the user terminal.

Syntax

```
public void Stop()
```

1.4.1.39 Action TestCooperStepBack

Jumps one cooper test step back. Only valid if a cooper test is running.

Syntax

```
public void TestCooperStepBack()
```

Action Errors

812	No cooper test active

1.4.1.40 Action TestCooperStepForward

Jumps one cooper test step forward. Only valid if a cooper test is running.

Syntax

```
public void TestCooperStepForward()
```

Action Errors

812	No cooper test active

1.5 Service class TreadmillOptions

This is a management service for all optional configuration settings and values.

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillOptions.dll

Syntax

```
public class TreadmillOptions : IServiceObject
```

Constructors

TreadmillOptions(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:Options
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillOptions:1

Service Properties and Events

No Variables

Service Methods

Name	Description
GetListEntry	Gets an entry from a list.
GetListSize	Gets the number of entries in a list.
GetOption	Gets an option value and its range.
GetVersion	Gets the version of the options.
SetOption	Sets an option value.

Service error codes

Error code	Error description
800	Not in select mode
801	Wrong access keys
824	Unknown option number
825	Unknown option error
826	Option value out of range
827	Invalid option list index
828	Invalid option level
829	No option list available
833	Current communication protocol cannot be changed

Details

On the following pages you will find a detailed description
for every action and variable of this service.

1.5.1 Actions

1.5.1.1 Action GetListEntry

Gets an entry from a list.

Syntax

```
public void GetListEntry(
   System.Char level
   System.Byte number
   System.Byte index
   ref System.Int32 _value
)
```

Parameters
level
Option level:
'A' = Administrator
'U' = User

number
Option identification number or 0 for a list with all options

index
List index (0 to GetListSize() - 1)

_value
List entry value

Action Errors

828	Invalid option level
824	Unknown option number
829	No option list available
800	Not in select mode
825	Unknown option error
827	Invalid option list index

1.5.1.2 Action GetListSize

Gets the number of entries in a list.

Syntax

```
public void GetListSize(
   System.Char level
   System.Byte number
   ref System.Byte size
)
```

Parameters
level
Option level:
'A' = Administrator
'U' = User

number
Option identification number or 0 for options list

size
Number of entries in the list

Action Errors

828	Invalid option level
824	Unknown option number
829	No option list available
800	Not in select mode
825	Unknown option error

1.5.1.3 Action GetOption

Gets an option value and its range.

Syntax

```
public void GetOption(
   System.Char level
   System.Byte number
   ref System.Int32 _value
   ref System.Int32 min
   ref System.Int32 max
)
```

Parameters
level
Option level:
'A' = Administrator
'U' = User

number
Option identification number

_value
Actual option value

min
Minimum option value

max
Maximum option value

Action Errors

828	Invalid option level
824	Unknown option number
800	Not in select mode
825	Unknown option error

1.5.1.4 Action GetVersion

Gets the version of the options.

Syntax

```
public System.String GetVersion(
   ref System.String version
)
```

Parameters
version
   Version string e.g. "101"

1.5.1.5 Action SetOption

Sets an option value.

Syntax

```
public void SetOption(
   System.Char level
   System.Byte number
   System.Int32 _value
   System.Byte key1
   System.Byte key2
)
```

Parameters
level
   Option level:
'A' = Administrator
'U' = User

number
   Option identification number

_value
   New option value

key1
   Access key 1

key2
   Access key 2

Action Errors

828	Invalid option level
824	Unknown option number
800	Not in select mode
826	Option value out of range
825	Unknown option error
833	Current communication protocol cannot be changed

801	Wrong access keys

1.6 Service class TreadmillDirectControl

The direct control service has actions to control the speed of the drive and the grade of the elevator directly.

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillDirectControl.dll

Syntax

```
public class TreadmillDirectControl : IServiceObject
```

Constructors

TreadmillDirectControl(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:DirectControl
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillDirectControl:1

Service Properties and Events

Type	Name	Flags	Description
System.Single	Acceleration	E G	Actual acceleration (>0) deceleration (<0) value [m/s²]
System.Single	ActualGrade	E G	Actual grade value [%]
System.Single	ActualSpeed	E G	Actual speed value [m/s]
System.Byte	DriveStatus	E G	Actual drive status: 0 = Stopped (RFR off) 1 = Constant speed 2 = Acceleration 3 = Deceleration
System.Byte	ElevatorStatus	E G	Actual elevator status: 0 = Off 1 = Up 2 = Down
System.Single	FinalGrade	E G	Final grade value [%]
System.Single	FinalSpeed	E G	Final speed value [m/s]

Service Methods

Name	Description
GetDriveAccelDecelRange	Gets the configured acceleration/deceleration range.
GetDriveSpeedRange	Gets the configured speed range.
GetDriveStatus	Gets drive status values.
GetElevatorGradeRange	Gets the configured grade range.
GetElevatorStatus	Gets elevator status values.
SetDriveSpeed	Sets a new drive speed and an acceleration/deceleration value.
SetElevatorGrade	Sets a new grade value.

Service error codes

Error code	Error description
831	Speed out of range
832	Grade out of range

Details

On the following pages you will find a detailed description
for every action and variable of this service.

1.6.1 Actions

1.6.1.1 Action GetDriveAccelDecelRange

Gets the configured acceleration/deceleration range.

Syntax

```
public void GetDriveAccelDecelRange(
   ref System.Byte minIndex
   ref System.Byte maxIndex
   ref System.Single minValue
   ref System.Single maxValue
)
```

Parameters
minIndex
Minimum acceleration/deceleration index value

maxIndex
Maximum acceleration/deceleration index value

minValue
Minimum acceleration/deceleration value [m/s²]

maxValue
Maximum acceleration/deceleration value [m/s²]

1.6.1.2 Action GetDriveSpeedRange

Gets the configured speed range.

Syntax

```
public void GetDriveSpeedRange(
   ref System.Single minSpeed
   ref System.Single maxSpeed
)
```

Parameters
minSpeed
Minimum speed [m/s]

maxSpeed
Maximum speed [m/s]

1.6.1.3 Action GetDriveStatus

Gets drive status values.

Syntax

```
public void GetDriveStatus(
   ref System.Byte driveStatus
   ref System.Single actualSpeed
```

```
   ref System.Single finalSpeed
   ref System.Single acceleration
)
```

Parameters
driveStatus
   Drive status:
0 = Stopped (RFR off)
1 = Constant speed
2 = Acceleration
3 = Deceleration

actualSpeed
   Actual Speed [m/s]

finalSpeed
   Final speed [m/s]

acceleration
   Acceleration [m/s²]

1.6.1.4 Action GetElevatorGradeRange

Gets the configured grade range.

Syntax

```
public void GetElevatorGradeRange(
   ref System.Single minGrade
   ref System.Single maxGrade
)
```

Parameters
minGrade
   Minimum grade [%]

maxGrade
   Maximum grade [%]

1.6.1.5 Action GetElevatorStatus

Gets elevator status values.

Syntax

```
public void GetElevatorStatus(
   ref System.Byte elevatorStatus
   ref System.Single actualGrade
   ref System.Single finalGrade
)
```

Parameters
elevatorStatus
   Elevator status:
0 = Off

1 = Up
2 = Down

actualGrade
Actual Grade [%]

finalGrade
Final Grade [%]

1.6.1.6 Action SetDriveSpeed

Sets a new drive speed and an acceleration/deceleration value.

Syntax

```
public void SetDriveSpeed(
  System.Single speed
  System.Byte accelIndex
  System.Single acceleration
)
```

Parameters
speed
Final Speed [m/s]

accelIndex
Acceleration index:
0 = Use "acceleration" argument
1 = (Max. FU speed) / 131 s
2 = (Max. FU speed) / 65.5 s
3 = (Max. FU speed) / 32.8 s
4 = (Max. FU speed) / 16.0 s
5 = (Max. FU speed) / 8.0 s
6 = (Max. FU speed) / 5.0 s
7 = (Max. FU speed) / 3.0 s

acceleration
Acceleration [m/s²]

Action Errors

831	Speed out of range

1.6.1.7 Action SetElevatorGrade

Sets a new grade value.

Syntax

```
public void SetElevatorGrade(
  System.Single grade
)
```

Parameters
grade
Final Grade [%]

Action Errors

832	Grade out of range

1.7 Service class TerminalLCD6x4Panel

This is a service to build a remote terminal with the same display and keyboard layout as the built-in treadmill terminal.

Namespace: XDOP.schemas_coscom_org.service
Assembly: TerminalLCD6x4Panel.dll

Syntax

```
public class TerminalLCD6x4Panel : IServiceObject
```

Constructors

TerminalLCD6x4Panel(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:Terminal
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TerminalLCD6x4Panel:1

Service Properties and Events

Type	Name	Flags	Description
System.Boolean	BlinkState	E G	Blink state of all LCD displays and LEDs. Every change sends an event. 0 = Off, 1 = On
System.Byte	BuzzerTime	E G	Current remaining buzzer on time in 1/100 seconds. The change events are sent as often as possible.
System.Byte[]	KeyState	E G	Bit field with MCU 4 key states (key on = bit set, key off = bit cleared). Key bit masks: 0x01 = STOP 0x02 = START 0x04 = UP 0x08 = DOWN 0x10 = PLUS 0x20 = MINUS 0x80 = EMERGENCY STOP Each state change sends an event, if the change rate isn't too high.

System.Byte[]	LEDGroups	E G	9 hex digits with the LED index of a LED group. Indexes of LED groups: first digit: 5 = manual 4 = profile 3 = cardio 2 = test 2. digit 5 = max 3. digit 1 = mph 2 = m/s 3 = km/h 4 = m/min 4. digit 3 = miles 4 = km 5 = m 5. digit 4 = <° 5 = % 6. digit 1 = No. 2 = Step 7. digit 1 = power 2 = energy 5 = met 8. digit 4 = v min. 5 = ^ max. 9. digit 1 = weight 2 = age 3 = sex index + A = blinking 0 = all off Each change sends an event.

System.String	LLDisplay	E G	Lower left LCD display string and attributes: - the first 4 characters build the display string - the fifth character (hexdecimal) describes the bit field for blinking digits - the sixth character (hexdecimal) describes the bit field for points between the digits - the seventh character (hexdecimal) describes the bit field for blinking points Bit pattern for blinking display digits (character 5 in the message string) 0x0 No blinking digits 0x1 Left digit is blinking 0x2 Middle left digit is blinking 0x4 Middle right digit is blinking 0x8 Right digit is blinking Bit pattern for display points (character 6 in the message string) 0x0 No points 0x1 Left point 0x2 Middle point 0x4 Right point 0x8 Colon Bit pattern for blinking display points (character 7 in the message string) 0x0 No blinking points 0x1 Blinking left point 0x2 Blinking middle point 0x4 Blinking right point 0x8 Blinking colon Example: " 00040" shows the string " 00" with a right point and looks like " 0.0" Each state change sends an event, if the change rate isn't too high.
System.String	LMDisplay	E G	Lower middle LCD display string and attributes. Same format as LLDisplay.
System.String	LRDisplay	E G	Lower right LCD display string and attributes. Same format as LLDisplay.
System.String	ULDisplay	E G	Upper left LCD display string and attributes. Same format as LLDisplay.
System.String	UMDisplay	E G	Upper middle LCD display string and attributes. Same format as LLDisplay.
System.String	URDisplay	E G	Upper right LCD display string and attributes. Same format as LLDisplay.

Service Methods

Name	Description
Beep	Switches the buzzer of the treadmill terminal on. The beep lasts as long as the given duration argument.

GetDistanceUnit	Gets the configured distance unit.
GetGradeUnit	Gets the configured grade unit.
GetKeyLock	Gets the keyboard lock state. If locked, only the key "-" and "Stop" can be used. Both stop the treadmill.
GetSpeedUnit	Gets the configured speed unit.
KeyDown	Changes the key state to "key down" (key pressed).
KeyUp	Changes the key state to "key up" (key released).
SetDistanceUnit	Sets the distance unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.
SetGradeUnit	Sets the grade unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.
SetKeyLock	Sets the keyboard lock state. If locked, only the key "-" and "Stop" can be used. Both stop the treadmill.
SetModeLED	Sets one of the mode LEDs or switches all off. The current control mode isn't be influenced.
SetSpeedUnit	Sets the speed unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.

Service error codes

Error code	Error description
819	Invalid LED mode
820	Invalid unit

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.7.1 Actions

1.7.1.1 Action Beep

Switches the buzzer of the treadmill terminal on. The beep lasts as long as the given duration argument.

Syntax

```
public void Beep(
   System.Byte duration
)
```

Parameters
duration
   Beep time in 1/100 seconds (0 to 2.55 s).

1.7.1.2 Action GetDistanceUnit

Gets the configured distance unit.

Syntax

```
public System.Byte GetDistanceUnit(
   ref System.Byte unit
)
```

Parameters
unit
   0 = km, 1 = miles, 2 = m

1.7.1.3 Action GetGradeUnit

Gets the configured grade unit.

Syntax

```
public System.Byte GetGradeUnit(
   ref System.Byte unit
)
```

Parameters
unit
   0 = %, 1 = ° (degree)

1.7.1.4 Action GetKeyLock

Gets the keyboard lock state. If locked, only the key "-" and "Stop" can be used. Both stop the treadmill.

Syntax

```
public void GetKeyLock(
   ref System.Boolean _lock
)
```

Parameters
_lock
   0 = unlocked, 1 = locked

1.7.1.5 Action GetSpeedUnit

Gets the configured speed unit.

Syntax

```
public System.Byte GetSpeedUnit(
   ref System.Byte unit
)
```

Parameters
unit
   0 = km/h, 1 = m/s, 2 = mph, 3 = m/min

1.7.1.6 Action KeyDown

Changes the key state to "key down" (key pressed).

Syntax

```
public void KeyDown(
   System.Char key
)
```

Parameters
key
   + = Minus, - = Plus, U = Up, D = Down, E = Start, S = Stop, F = Emergency Stop

1.7.1.7 Action KeyUp

Changes the key state to "key up" (key released).

Syntax

```
public void KeyUp(
   System.Char key
)
```

Parameters
key
   + = Minus, - = Plus, U = Up, D = Down, E = Start, S = Stop, F = Emergency Stop

1.7.1.8 Action SetDistanceUnit

Sets the distance unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.

Syntax

```
public void SetDistanceUnit(
  System.Byte unit
)
```

Parameters
unit
0 = km, 1 = miles, 2 = m

Action Errors

820	Invalid unit

1.7.1.9 Action SetGradeUnit

Sets the grade unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.

Syntax

```
public void SetGradeUnit(
  System.Byte unit
)
```

Parameters
unit
0 = %, 1 = ° (degree)

Action Errors

820	Invalid unit

1.7.1.10 Action SetKeyLock

Sets the keyboard lock state. If locked, only the key "-" and "Stop" can be used. Both stop the treadmill.

Syntax

```
public void SetKeyLock(
  System.Boolean _lock
)
```

Parameters
_lock
0 = unlocked, 1 = locked

1.7.1.11 Action SetModeLED

Sets one of the mode LEDs or switches all off. The current control mode isn't be influenced.

Syntax

```
public void SetModeLED(
  System.Byte mode
  System.Boolean blink
)
```

Parameters
mode
0 = All Off, 1 = Manual, 2 = Profile, 3 = Cardio, 4 = Test

blink
0 = permanent on, 1 = blinking

Action Errors

819	Invalid LED mode

1.7.1.12 Action SetSpeedUnit

Sets the speed unit temporarily. If the treadmill is switched off or if the configuration mode is entered, then this setting is lost.

Syntax

```
public void SetSpeedUnit(
  System.Byte unit
)
```

Parameters
unit
0 = km/h, 1 = m/s, 2 = mph, 3 = m/min

Action Errors

820	Invalid unit

1.8 Service class TreadmillMeasures

Treadmill measures like speed, grade, time, distance, mets, heart rate, ...

Namespace: XDOP.schemas_coscom_org.service
Assembly: TreadmillMeasures.dll

Syntax

```
public class TreadmillMeasures : IServiceObject
```

Constructors

TreadmillMeasures(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:Measures
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:TreadmillMeasures:1

Service Properties and Events

Type	Name	Flags	Description
System.Single	Distance	E G	Actual distance value [m].
System.Single	Energy	E G	Actual energy value [kJ].
System.Single	Grade	E G	Actual grade value [%].
System.Byte	HeartRate	E G	Actual heart rate [1/min].
System.Single	MET	E G	Actuale MET (metabolic equivalent) value.
System.Single	Power	E G	Actual power value [W].
System.Single	Speed	E G	Actual speed value [m/s].
System.UInt32	Time	E G	Actual run time value [s].

Service Methods

Name	Description
ResetDistance	Resets the distance value.
ResetEnergy	Resets the energy value.
ResetTime	Resets the run time.
SetDistance	Sets the distance value.
SetEventMask	Sets the event mask to control the eventing of each measure.
SetTime	Sets the time values.

Service error codes

Error code	Error description
830	Distance value too big

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.8.1 Actions

1.8.1.1 Action ResetDistance

Resets the distance value.

Syntax

```
public void ResetDistance()
```

1.8.1.2 Action ResetEnergy

Resets the energy value.

Syntax

```
public void ResetEnergy()
```

1.8.1.3 Action ResetTime

Resets the run time.

Syntax

```
public void ResetTime()
```

1.8.1.4 Action SetDistance

Sets the distance value.

Syntax

```
public void SetDistance(
  System.Single distance
)
```

Parameters
distance
The resolution of the distance value is 0.1 m and the maximum value depends on the treadmill type.
e.g. type 1.3 (167400 increments / m): 0 - 256569 m

Action Errors

830	Distance value too big

1.8.1.5 Action SetEventMask

Sets the event mask to control the eventing of each measure.

Syntax

```
public void SetEventMask(
   System.Byte[] eventMask
)
```

Parameters
eventMask
Event Mask: time 0x01, distance 0x02, speed: 0x04, grade 0x08, heart rate 0x10, power 0x20, energy 0x40, MET 0x80

1.8.1.6 Action SetTime

Sets the time values.

Syntax

```
public void SetTime(
   System.Byte hrs
   System.Byte min
   System.Byte sec
)
```

Parameters
hrs
Hours value from 0 to 255

min
Minutes value:
hrs = 0: 0 - 59 minutes or 0 - 99 minutes
hrs > 0: 0 - 59 minutes

sec
Seconds value from 0 to 59

1.9 Service class MCU5FlashUpdate

This service can be used to update the firmware in the flash of the microcontroller. The update is separated in two phases. First a data buffer (data flash on the electronic board) is written with the firmware and constant data (WriteBuffer). In the second phase the microcontoller reads from the data flash part and writes to its on-chip flash memory (StartUpdate) without any intervention from the client. After a successful update of the on-chip flash, the microcontroller resets automatically.

Namespace: XDOP.schemas_coscom_org.service
Assembly: MCU5FlashUpdate.dll

Syntax

```
public class MCU5FlashUpdate : IServiceObject
```

Constructors

MCU5FlashUpdate(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:MCU5FlashUpdate
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:MCU5FlashUpdate:1

Service Properties and Events

Type	Name	Flags	Description
System.UInt16	UpdatePageCountDown	EvG	

Service Methods

Name	Description
GetDefaultType	Gets the default type and sub type, which has been set by the factory. This values are important, if the configuration parameters (options) of the treadmill are lost (e.g. the battery of the Real-Time-Clock has been replaced or the signature of the configuration memory is false).
GetFirmwareVersion	Gets the firmware version.
ReadBuffer	Reads 128 bytes of the data flash buffer. The data flash buffer has a size of 256 bytes. An even page index reads the lower 128 bytes of data flash buffer and an odd page index reads the upper 128 bytes of the data flash buffer. Any call to ReadBuffer loads the data flash buffer from the data flash first. The data flash has a size of 256 kByte (= 2048 * 128).
ShowUpdateScreen	Shows flash update strings on the displays. This action should be used before the action "StartUpdate" is called.

SoftReset	After the response of this action is sent, the microcontroller jumps to the reset address (soft reset).
StartUpdate	Starts the flash update of the microcontroller built-in flash.
WriteBuffer	Writes 128 bytes to the data flash buffer. The data flash buffer has a size of 256 bytes. If the second 128 Bytes (odd page index) of the buffer are written, then the whole buffer is written to the data flash. The data flash has a size of 256 kByte (= 2048 * 128).

Service error codes

No error codes defined

Details

On the following pages you will find a detailed description for every action and variable of this service.

1.9.1 Actions

1.9.1.1 Action GetDefaultType

Gets the default type and sub type, which has been set by the factory. This values are important, if the configuration parameters (options) of the treadmill are lost (e.g. the battery of the Real-Time-Clock has been replaced or the signature of the configuration memory is false).

Syntax

```
public void GetDefaultType(
   ref System.Byte type
   ref System.Byte subType
)
```

Parameters
type
Major type (major treadmill type)

subType
Sub type (treadmill type variant)

1.9.1.2 Action GetFirmwareVersion

Gets the firmware version.

Syntax

```
public System.String GetFirmwareVersion(
   ref System.String version
)
```

Parameters
version
String with 4 digits (e.g. 4011)

1.9.1.3 Action ReadBuffer

Reads 128 bytes of the data flash buffer. The data flash buffer has a size of 256 bytes. An even page index reads the lower 128 bytes of data flash buffer and an odd page index reads the upper 128 bytes of the data flash buffer. Any call to ReadBuffer loads the data flash buffer from the data flash first. The data flash has a size of 256 kByte (= 2048 * 128).

Syntax

```
public void ReadBuffer(
   System.UInt16 page128Idx
   System.Byte key1
   System.Byte key2
   ref System.Byte[] packet
)
```

Parameters

page128Idx
Index of the 128 byte long page:
0 <= page128Idx < 2048

key1
Access key 1

key2
Access key 2

packet
Flash memory packet:
128 flash buffer bytes + 1 checksum byte = 129 bytes
129 bytes base64 coded = 172 characters
(129 / 3) * 4 = 172

1.9.1.4 Action ShowUpdateScreen

Shows flash update strings on the displays. This action should be used before the action "StartUpdate" is called.

Syntax

```
public void ShowUpdateScreen()
```

1.9.1.5 Action SoftReset

After the response of this action is sent, the microcontroller jumps to the reset address (soft reset).

Syntax

```
public void SoftReset()
```

1.9.1.6 Action StartUpdate

Starts the flash update of the microcontroller built-in flash.

Syntax

```
public void StartUpdate(
  System.Byte key1
  System.Byte key2
)
```

Parameters
key1
Access key 1

key2
Access key 2

1.9.1.7 Action WriteBuffer

Writes 128 bytes to the data flash buffer. The data flash buffer has a size of 256 bytes. If the second 128 Bytes (odd page index) of the buffer are written, then the whole buffer is written to the data flash. The data flash has a size of 256 kByte (= 2048 * 128).

Syntax

```
public void WriteBuffer(
  System.UInt16 page128Idx
  System.Byte[] packet
  System.Byte key1
  System.Byte key2
  ref System.Byte error
)
```

Parameters
page128Idx
Index of the 128 byte long page:
0 <= page128Idx < 2048

packet
Flash update packet:
128 flash buffer bytes + 1 checksum byte = 129 bytes
129 bytes base64 coded = 172 characters
(129 / 3) * 4 = 172

key1
Access key 1

key2
Access key 2

error
Packet checksum error:
0 = okay
1 = wrong

1.10 Service class DataLogger

This service provides a configurable data logger. The output of the logger is evented.

Namespace: XDOP.schemas_coscom_org.service
Assembly: DataLogger.dll

Syntax

```
public class DataLogger : IServiceObject
```

Constructors

DataLogger(XDOPLib.IService service)	Initializes a new instance of the class.

Common Service Properties

Property name	Value
ServiceId	urn:schemas-coscom-org:serviceId:Logger
MajorVersion	1
MinorVersion	1
ServiceType	urn:schemas-coscom-org:service:DataLogger:1

Service Properties and Events

Type	Name	Flags	Description
System.String	Record	E G	Record string with all configured column values

Service Methods

Name	Description
SetColumnDelimiter	Sets the column delimiter character.
SetRecordColumns	Sets the columns for the log record.
SetRecordEventPeriod	Sets the event period.
StartLogging	Starts the logging.
StopLogging	Stops the logging.

Service error codes

No error codes defined

Details

On the following pages you will find a detailed description
for every action and variable of this service.

1.10.1 Actions

1.10.1.1 Action SetColumnDelimiter

Sets the column delimiter character.

Syntax

```
public void SetColumnDelimiter(
   System.Char delimiter
)
```

Parameters
delimiter
   Delimiter character for delimiting the column values in the log record

Remarks

Comment for Action SetColumnDelimiter

1.10.1.2 Action SetRecordColumns

Sets the columns for the log record.

Syntax

```
public void SetRecordColumns(
   System.String columnList
)
```

Parameters
columnList
   String with semi-colon separated indexes of columns, which should be logged:
1 = Time
2 = Distance
3 = Speed
4 = Grade
5 = Heart Rate
6 = Power
7 = Energy
8 = MET

Example: 1;3;4 for time, speed and grade

1.10.1.3 Action SetRecordEventPeriod

Sets the event period.

Syntax

```
public void SetRecordEventPeriod(
   System.Byte period
)
```

Parameters
period
Event period (time between two log record outputs) [s]

1.10.1.4 Action StartLogging

Starts the logging.

Syntax

```
public void StartLogging()
```

1.10.1.5 Action StopLogging

Stops the logging.

Syntax

```
public void StopLogging()
```