# Python wrapper for the MCSControl.dll C/C++ library

(unofficial pre-release version Sep. 2017)

The MCSControl_PyhtonWrapper.py library is utilizing the '*ctype*' package from the Python standard library to integrate all functions of the MCSControl.dll C/C++ library. This allows the user to access the full functionality of the MCSControl.dll C/C++ library described in the '*MCS Programmers Guide*' to control SmarAct positioners via the MCS controller using Python code.

## Import MCSControl_PyhtonWrapper.py

To be able to access all functionalities of the MCS control system described in the '*MCS Programmers Guide*' you have to import the 'MCSControl_PyhtonWrapper.py' package via: `from MCSControl_PythonWrapper import *` at the beginning of your Python code.

The '*ctype*' package import is included in the 'MCS_Control_PythonWrapper' and can be accessed using the prefix `ct.<cmd>`.

## Dealing with data types

The MCSControl.dll C/C++ library expects C/C++ data types. Therefore, every variable which interacts with a function from the MCSControl.dll C/C++ library has to be implemented as a '*ctype*' data type, e.g. C/C++ `int` corresponds to a '*ctype*' `ct.c_int()` or a C/C++ `unsigned int` corresponds to a '*ctype*' `ct.c_ulong()`. The special SmarAct `SA_PACKET` type is defined and initialized by the command `packet = SA_packet()`. It is necessary to initialize an inheritance of the `SA_packet class` before setting a `POINTER` to it. All C/C++ types expected by the functions are referenced in the comments in the MCS_Control_PythonWrapper.py file (see reference to the original C/C++ function call). To access the value of a 'ctype' data type you can use the `.value` method, e.g.: `number = ct.c_ulong(20)` with `print(number.value) = 20`, while `print(number) = c_ulong(20)`.

## Python Str vs. C/C++ char byte arrays

The functions `SA_OpenSystem` and `SA_FindSystems` expecting `const char pointer(s)` in form of locator string as a `byteArray`(s) to work properly. To convert Python `strings` to `byteArray(s)` you can use the command: `bytes("YOUR_LOCATOR_STRING",'utf-8')`.
The output of `SA_FindSystems` is already a `byteArray` which can be directly pass along to the `SA_OpenSystem` function. To decode a `byteArray` to a Python `string` use e.g.: `outBuffer[:].decode("utf-8")`

More information can be found in the programming examples.

SmarAct GmbH ● Schuette-Lanz-Str. 9 ● D-26135 Oldenburg ● www.smaract.com
phone.: +49 (0) 441-800879-0 ● fax: +49 (0) 441-800879-21 e-mail: info@smaract.com