Requirements for the Module com_lib.serialialization

Conventions

Requirements listed in this document are constructed according to the following structure:

Requirement ID: REQ-UVW-XYZ

Title: Title / name of the requirement

Description: Descriprion / definition of the requirement

Verification Method: I / A / T / D

The requirement ID starts with the fixed prefix 'REQ'. The prefix is followed by 3 letters abbreviation (in here 'UVW'), which defines the requiement type - e.g. 'FUN' for a functional and capability requirement, 'AWM' for an alarm, warnings and operator messages, etc. The last part of the ID is a 3-digits *hexadecimal* number (0..9|A..F), with the first digit identifing the module, the second digit identifing a class / function, and the last digit - the requirement ordering number for this object. E.g. 'REQ-FUN-112'. Each requirement type has its own counter, thus 'REQ-FUN-112' and 'REQ-AWN-112' requirements are different entities, but they refer to the same object (class or function) within the same module.

The verification method for a requirement is given by a single letter according to the table below:

Term	**Definition**
Inspection (I)	Control or visual verification
Analysis (A)	Verification based upon analytical evidences
Test (T)	Verification of quantitative characteristics with quantitative measurement
Demonstration (D)	Verification of operational characteristics without quantitative measurement

Functional and capability requirements

Requirement ID: REQ-FUN-300

Title: Module's content

Description: The module should implement a number of classes defining the specific data types with the built-in (de-) serialization functionality and representing

- An empty object - None / NULL
- Structure - a collection of heterogeneous elements (attributes / fields) accessible by name
- Array - a collection of homogeneous elements (same type) accessible by index

Verification Method: A

Requirement ID: REQ-FUN-301

Title: (De-) serialization functionality

Description: The defined classes (as data types) should provide functionality for serialization into and deserialization from:

- JSON format strings
- Packed bytestrings without padding

The both serialization (packing) and deserialization (unpacking) must prevent any data loss and be completely reversible and unambiguous.

Verification Method: A

Requirement ID: REQ-FUN-302

Title: Common API

Description: All defined classes (as data types) should provide the unified interface - class or instance methods:

- *getSize* - class method returning the number of bytes required to represent all stored data as a bytestring, None value is an indication of a not fixed size container
- *unpackBytes* - class method returning a new instance of the class, with the stored data extracted from the passed bytestring
- *unpackJSON* - class method returning a new instance of the class, with the stored data extracted from the passed JSON format string
- instantiation without parameters or with a single argument of a native Python type
- *packBytes* - instance method returning a bytestring representing all internally stored data
- *packJSON* - instance method returning a JSON format string representing all internally stored data
- *getNative* - instance method reurning native Python representation of the stored data

The both serialization (packing) and deserialization (unpacking) must prevent any data loss and be completely reversible and unambiguous.

Verification Method: T

Requirement ID: REQ-FUN-303

Title: Endianness

Description: By default the native platform endianness of the bytes representation should be used, which is little endian on the majority of the modern platforms (with the majority of the targeted users expected to use x86 / x86-64 architecture PCs); however, the specific big endian or little endian byte order can be enforced in the byte-packing and unpacking methods.

Verification Method: T

Requirement ID: REQ-FUN-310

Title: Empty object - definition and functionality

Description: This data type is designed to represent an empty data field or element of a container data type, and it should be represented as

- JSON format string "null"
- An empty bytestring
- **None** as the native Python data type

Verification Method: T

Requirement ID: REQ-FUN-311

Title: Empty object - instantiaton

Description: The optional argument passed into the instantiation method should be simply ignored.

Verification Method: T

Requirement ID: REQ-FUN-320

Title: Fixed length array - definition

Description: A fixed length array is a semi-mutable ordered container type storing a limited number of elements of the same type. Both the type of the stored elements and their number are defined at the class level, and all instances have exactly the same number of the same type of elements. Neither the number of elements nor their type cannot be changed at the run-time. The allowed types of the elements are:

- Any scalar primitive C compatible data type defined in the standard library **ctypes**
- A fixed-size structure (not containing a dynamic length array)
- A fixed length array

Verification Method: T

Requirement ID: REQ-FUN-321

Title: Fixed length array - element access

Description: An element can be accessed by an integer index (standard Python indexing approach), but not using slicing notation. The following rules are applied:

- Read access
  - The values returned are of the native Python types **int**, **float**, **bool**, **bytes** (1 byte length) or **str** (1 char lentgth, ASCII only as **char** but not **wchar**) are returned for the scalar types of elements
  - The stored instance of the respective class is returned if the array stores structures or arrays
- Wrire access
  - The same native scalar Python types values can be assigned to any element if the value is compatible with the **ctypes** type declared as the elements type
  - Write access is not allowed if the elements type is a structure or array

Verification Method: T

Requirement ID: REQ-FUN-322

Title: Fixed length array - instantiation

Description:

- Without passed argument all elements are created with their default values
- If the optional argument passed into the instantiation method
    - It should be of any Python sequence type or an instance of a fixed or dynamic length array
    - If the length of passed argument is less than the length of the array the remaining tail elements are created with their default values
    - If the length of the passed argument is greater than the length of the array only the respective number of the head elements are used for the instantiation of the elements of the array
    - The value of each of the element of the passed argument must be compatible with the declared type of the elements of the array

Verification Method: T

Requirement ID: REQ-FUN-323

Title: Fixed length array - serialization to bytes

Description: All elements of the array must be placed after each other without any padding, i.e. using exactly the size of the declared type of the elements in bytes per element, and exactly in the same order, as they are stored. The declared for the class endianness should be used for the representation of a scalar values. If a scalar type is declared as the type of the elements of the array, the current isntance is responsible for the convertion. If a nested container is declared as the type of the elements of the array, the packing of an element should be delegated to itself. The resulted bytestring's length must be equal to the declared number of elements times the size in bytes of the element type.

Verification Method: T

Requirement ID: REQ-FUN-324

Title: Fixed length array - deserialization from bytes

Description: The length of the bytestring must be exactly equal to the size of the declared element type in bytes times the declared number of elements. A new instance of the class is created, and its elements are instantiated using the bytes representation stored in the respective slices (sub-strings) of the bytestring. If a scalar type is declared as the elements type, the class itself is responsible for the conversion of the bytes representation into a Python object, and the declared for the class endianness should be preserved. Otherwise (nested container), the respective slice of the bytestring should be passed into the respective unpacking method of the class declared as the elements type.

Verification Method: T

Requirement ID: REQ-FUN-325

Title: Fixed length array - serialization to JSON string

Description: The returned by the method value should be a string (type **str**) representing a JSON array with exactly the same number of elements as stored in the array being serialized and exactly in the same order, as they are stored. All elements of the JSON array must be of the same type, corresponding to a native Python representation of an element of the array being serialized.

Verification Method: T

Requirement ID: REQ-FUN-326

Title: Fixed length array - deserialization from JSON string

Description: The JSON string must be a representation of a an array of the same type elements, which is compatible with the declared elements type of the array class, and the same number of elements as declared for the array class. A new instance of the array class should be created, and its elements must be instantiated with the values of the respective elements of the JSON array.

Verification Method: T

Requirement ID: REQ-FUN-327

Title: Fixed length array - native Python representation

Description: The stored data in the native Python format must be a **list** of the length equal to the declared length of the array, with all elements of the Python list must be of the same type (scalar, **dict** or **list**) defined by the declared type of the array's elements. In case of the nested container declared type (structure or array class) the native Python representation of the value of an element should be delegated to the respective method of the element's type class. It is the responsibility of an instance of the array class to provide the native Python representation of the stored values if a scalar type of the elements is declared.

Verification Method: T

Requirement ID: REQ-FUN-328

Title: Fixed length array - additional API

Description: The class should provide support for the standard Python function *len*()

Verification Method: T

Requirement ID: REQ-FUN-330

Title: Dynamic length array - definition

Description: A dynamic length array is a semi-mutable ordered container type storing a limited number of elements of the same type. The type of the stored elements is defined at the class level, and all instances have exactly the same type of elements, which cannot be changed at the run-time. The allowed types of the elements are:

- Any scalar primitive C compatible data type defined in the standard library **ctypes**
- A fixed-size structure (not containing a dynamic length array)
- A fixed length array

The number of elements is defined during the instantiation of the class based on the size of the input data, and it cannot be changed afterwards.

Verification Method: T

Requirement ID: REQ-FUN-331

Title: Dynamic length array - element access

Description: An element can be accessed by an integer index (standard Python indexing approach), but not using slicing notation. The following rules are applied:

- Read access
  - The values returned are of the native Python types **int**, **float**, **bool**, **bytes** (1 byte length) or **str** (1 char lentgth, ASCII only as **char** but not **wchar**) are returned for the scalar types of elements
  - The stored instance of the respective class is returned if the array stores structures or arrays
- Wrire access
  - The same native scalar Python types values can be assigned to any element if the value is compatible with the **ctypes** type declared as the elements type
  - Write access is not allowed if the elements type is a structure or array

Verification Method: T

Requirement ID: REQ-FUN-332

Title: Dynamic length array - instantiation

Description:

- Without passed argument an empty array (no elements) is created
- If the optional argument passed into the instantiation method
  - It should be of any Python sequence type or an instance of a fixed or dynamic length array
  - The length of the passed argument defines the length of the created array
  - The created elements are instantiated with the values of the respective elements values of the passed argument
  - The value of each of the element of the passed argument must be compatible with the declared type of the elements of the array

Verification Method: T

Requirement ID: REQ-FUN-333

Title: Dynamic length array - serialization to bytes

Description: All elements of the array must be placed after each other without any padding, i.e. using exactly the size of the declared type of the elements in bytes per element, and exactly in the same order, as they are stored. The declared for the class endianness should be used for the representation of a scalar values. If a scalar type is declared as the type of the elements of the array, the current isntance is responsible for the convertion. If a nested container is declared as the type of the elements of the array, the packing of an element should be delegated to itself. The resulted bytestring's length must be equal to the number of elements in the array times the size in bytes of the element type.

Verification Method: T

Requirement ID: REQ-FUN-334

Title: Dynamic length array - deserialization from bytes

Description: The length of the bytestring must be either 0 (results in an empty array) or the size of the declared element type in bytes times a positive integer number, which defines the length of the array to be created. A new instance of the class is created, and its elements are instantiated using the bytes representation stored in the respective slices (sub-strings) of the bytestring. If a scalar type is declared as the elements type, the class itself is responsible for the conversion of the bytes representation into a Python object, and the declared for the class endianness should be preserved. Otherwise (nested container), the respective slice of the bytestring should be passed into the respective unpacking method of the class declared as the elements type.

Verification Method: T

Requirement ID: REQ-FUN-335

Title: Dynamic length array - serialization to JSON string

Description: The returned by the method value should be a string (type **str**) representing a JSON array with exactly the same number of elements as stored in the array being serialized and exactly in the same order, as they are stored. All elements of the JSON array must be of the same type, corresponding to a native Python representation of an element of the array being serialized.

Verification Method: T

Requirement ID: REQ-FUN-336

Title: Dynamic length array - deserialization from JSON string

Description: The JSON string must be a representation of a an array of the same type elements, which is compatible with the declared elements type of the array class. A new instance of the array class should be created with the same number of elements as the JSON array, and its elements must be instantiated with the values of the respective elements of the JSON array.

Verification Method: T

Requirement ID: REQ-FUN-337

Title: Dynamic length array - native Python representation

Description: The stored data in the native Python format must be a **list** of the length equal to the current length of the array, with all elements of the Python list must be of the same type (scalar, **dict** or **list**) defined by the declared type of the array's elements. In case of the nested container declared type (structure or array class) the native Python representation of the value of an element should be delegated to the respective method of the element's type class. It is the responsibility of an instance of the array class to provide the native Python representation of the stored values if a scalar type of the elements is declared.

Verification Method: T

Requirement ID: REQ-FUN-338

Title: Dynamic length array - additional API

Description: The following additional API is required:

- Class method *getElementSize* - size in bytes of the declared type of the elements
- Support for the standard Python function *len*()

Verification Method: T

Requirement ID: REQ-FUN-340

Title: Structure - definition

Description: A structure is a semi-mutable stuctured container type storing a limited number of fields of any of the allowed types, which can be accessed using unique identifiers. The types of the stored fields and their ientifiers are defined at the class level; and they cannot be changed at the run-time on the instances. Each instance is quaranteed to have the declared fields, for which the allowed types are:

- Any scalar primitive C compatible data type defined in the standard library **ctypes**
- A fixed-size structure (not containing a dynamic length array)
- A fixed length array
- Only as the last (bottom-most) field, in which case the size of the structure is not fixed:
    - A dynamic length array
    - A nested structured containint a dynamic length array

New attributes cannot be added to an instance of the structure at the run-time.

Verification Method: T

Requirement ID: REQ-FUN-341

Title: Structure - field (attribute) access

Description: An attribute can be accessed by its attribute. The following rules are applied:

- Read access of the declared fields

    - The values returned are of the native Python types **int**, **float**, **bool**, **bytes** (1 byte length) or **str** (1 char lentgth, ASCII only as **char** but not **wchar**) are returned for the scalar type attributes
    - The stored instance of the respective class is returned for the nested containers
- Write access to the declared fields
    - The same native scalar Python types values can be assigned to any element if the value is compatible with the **ctypes** type declared as the elements type
    - Write access is not allowed if the elements type is a structure or array
- Any access to an undeclared identificator is an error, except for:
    - (Call) access to the 'public' class and instance methods
    - Read access to some 'magic' attributes - part of the Python data model; but only to those that are absolutely neccessary for the maintenance of the attribute resolution scheme and proper functionality of the Python built-in functions and used dependencies

Verification Method: T

Requirement ID: REQ-FUN-342

Title: Structure - instantiation

Description:

- Without passed argument all declared fields are instantiated with their default values
- If the optional argument passed into the instantiation method
    - It should be of any Python mapping type or an instance of a structure
    - If the declared field's identifier is present as a key / field in the passed argument, the corresponding value must be compatible with the declared type of the field of the structure, and it will be used for the instantiation of that field
    - If the declared field's identifier is not present as a key / field in the passed argument, the corresponding field is created with the default value

Verification Method: T

Requirement ID: REQ-FUN-343

Title: Structure - serialization to bytes

Description: All fields of the structure must be placed after each other without any padding, i.e. using exactly the size of the declared type of the field in bytes per element, and exactly in the same order, as they are declared. The declared for the class endianness should be used for the representation of a scalar values. If a scalar type is declared as the type of the field, the current isntance is responsible for the convertion. If a nested container is declared as the type of the field, the packing of that field should be delegated to itself. The resulted bytestring's length must be equal to the sum of the current byte sizes of all declared fields.

Verification Method: T

Requirement ID: REQ-FUN-344

Title: Structure - deserialization from bytes

Description: For the fixed length structure the length of the bytestring must be exactly equal to the sum of the sizes of the declared fields types in bytes. If the structure includes a dynamic array (as the last field), the length of the bytestring must be not less than the sum of the sizes in bytes of all other fields, and the difference between those values must be equal to the size in bytes of the declared elements type of the dynamic array times a non-negative integer number. A new instance of the class is created, and its fields are instantiated using the bytes representation stored in the respective slices (sub-strings) of the bytestring exactly in the order of their declaration. If a scalar type is declared for the field, the class itself is responsible for the conversion of the bytes representation into a Python object, and the declared for the class endianness should be preserved. Otherwise (nested container), the respective slice of the bytestring should be passed into the respective unpacking method of the class declared as the field type.

Verification Method: T

Requirement ID: REQ-FUN-345

Title: Structure - serialization to JSON string

Description: The returned by the method value should be a string (type **str**) representing a JSON dictionary / struct with exactly the same number of key : value pairs as declared in the class definition. The names of the keys must be the declared field names, whereas the bound values must be JSON representation of the native Python representation of the value of the restective field of the instance being serialized.

Verification Method: T

Requirement ID: REQ-FUN-346

Title: Structure - deserialization from JSON string

Description: The JSON string must be a representation of a dictionary / struct with exactly the same number of key : value pairs as the declared fields of the class, and for each declared field name the JSON dict must contain the respective (same name) key, and the value bound to that key must be compatible with the declared type of the respective field. A new instance of the structure class should be created, and its fields must be instantiated with the values of the corresponding keys of the JSON dict.

Verification Method: T

Requirement ID: REQ-FUN-347

Title: Structure - native Python representation

Description: The stored data in the native Python format must be a **dict** containing only keys with the names equal to the declared fields of the class, and the respective values being the native Python representation of the values of the fields. In case of the nested container declared type (structure or array class) the native Python representation of the value of an element should be delegated to the respective method of the field's type class. It is the responsibility of an instance of the strucutre class to provide the native Python representation of the stored value if a scalar type of a field is declared.

Verification Method: T

Requirement ID: REQ-FUN-348

Title: Structure - additional API

Description: Due to ability to include a dynamic length array as the last field into the class definition, additional API is required:

* Class method *getMinSize* - size in bytes of all fields excluding the dynamic array (if declared)
* Instance method *getCurrentSize* - size in bytes of all data stored in all fields including the dynamic array (if declared) at the moment

Verification Method: T

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-300

Title: Improper definition of the class

Description: **TypeError** or its subclass should be raised if an inacceptable type is declared for the elements of an array class or field(s) of a structure.

Verification Method: T

Requirement ID: REQ-AWM-301

Title: Improper type of the argument of initialization method

Description: **TypeError** or its subclass should be raised if an inacceptable type argument is passed into the initialization method of an array or structure class.

Verification Method: T

Requirement ID: REQ-AWM-302

Title: Improper data structure of the argument of initialization method

Description: **ValueError** or its subclass should be raised if the type of an element of a sequence or value of the key of a mapping type passed as the argument into the initializer method of an array or structure class is not compatible with the declared type of an element / field.

Verification Method: T

Requirement ID: REQ-AWM-303

Title: Improper type of the de-serialization method argument

Description: **TypeError** or its subclass should be raised if

- Not a bytestring is passed into the class method *unpackBytes*
- Not a JSON string is passed into the class method *unpackJSON*, or the JSON encoded data type is not compatible with the array / structure

Verification Method: T

Requirement ID: REQ-AWM-304

Title: Improper content of the de-serialization method argument

Description: ValueError or its subclass should be raised if

- The bytestring passed into the class method *unpackBytes* is too short or too long
- The structure of a JSON string is passed into the class method *unpackJSON* does not meet the requirements:
  - too litle or too much elements in the array
  - missing or undeclared keys (as field names)
  - incompatible data type of element of an array or value of a key

Verification Method: T

Requirement ID: REQ-AWM-305

Title: Improper attribute access

Description: AttributeError or its subclass should be raised if

- Any attribute except for the methods or declared fields (of a structure) is read-accessed
  - With exception for the 'magic' data attributes (fields) *__class__*, *__name__*, which are required for the proper operation of the custom, enhanced exceptions with the traceback processing
- Any non-method attribute except for the declared fields (of a structure) is write-accessed

Verification Method: T

Requirement ID: REQ-AWM-306

Title: Improper index access

Description: IndexError or its subclass should be raised if the passed index is not an integer number within the bounds of (- length of an array) to (length of array - 1)

Verification Method: T

Requirement ID: REQ-AWM-307

Title: Improper type field / element assignment

Description: TypeError or its subclass should be raised if an attempt is made to assign a value to an element of an array or a field of a structure in the following cases:

- The accessed element or field is a nested container class
- The value to be assigned is not compatible with the declared type of the element / field

Verification Method: T