The Object Rexx Serializing framework

Version 1.9

The Serializable class offers a complete infrastructure for storing arbitrary data structures persistently. It offers mechanisms for converting data structures into a special file format and to read this format and convert it back to the original data structure.

The Serializable class

This class is a mix in class to indicate that an object can be serialized. It defines some default behaviour which can be overwritten if desired.

By default the class expects a subclass to implement an attribute called `PersistentData`. This attribute is intended to store all data that needs to survive the serializing/deserializing process.

Methods the Serializable class defines:

INIT (overwrites Object class and Class class method)

WriteObject

ReadObject (Class Method)

Methods inherited from the Object class:

...

INIT

```
>>-INIT(data)-------------------------------------------------------------><
```

Create an instance of the class. By default the *data* is stored in the attribute `PersistentData`, which has to be implemented by the class inheriting from the Serializable class.

INIT (Class)

The Serializable class generates a SerializeSerializable instance that is able to serialize subclasses of the Serializable class. As the Serializable class has to get notified about the creation of sub classes they must forward the init message.

If a sub class overwrites this method it has to forward the signal to the super class.

WriteObject

A method which has to return the data an object wants to save. This data will be serialized.

ReadObject (Class Method)

```
>>-ReadObject(data)-------------------------------------------------------><
```

This method has to return an instance of the receiver class. The instance has to be initialized with the data given. The default implementation passes the data to `INIT`, which initializes the instance and fills the attribute `PersistentData`.

Example:

```
object = .TotalRunningTime~new(0)
object~start
call SysSleep 3
object~stop
serialobject = .SerializeFunctions~Serialize(object)
call SysSleep 1
object = .SerializeFunctions~DeSerialize(serialobject)
object~start
call SysSleep 3
object~stop
exit
::REQUIRES “Serializable.cls”
::CLASS TotalRunningTime MIXINCLASS Serializable
::METHOD PersistentData ATTRIBUTE
::METHOD Start
      expose running
      running = .true
      Say “Starting at” self~PersistentData
      reply
      do while running
            call SysSleep 1
            call charout ,self~PersistentData” “
            self~PersistentData = self~PersistentData + 1
      end
::METHOD Stop UNGUARDED
      expose running
      running = .false
/*
Possible output:
Stating at 0
0 1 2 Starting at 2
2 3 4
*/
```

This class implements the PersistentData attribute. It will be used to store persistent data. The variable `serialobject` is a mutablebuffer object. When an application starts this data could be passed to the deserializer, returning the object that was serialized.

The SerializeFunctions Class

This class implements the logic necessary to serialize data and to convert it to the equivalent of the serialized data at a different time. There are is an interface to serialize and derserialize data for normal applications and another interface which helper modules use to make other classes serializable.

Methods the SerializeFunctions class implements:

SetHandler (Class Method)

Serialize (Class Method)

DeSerialize (Class Method)

Internal methods: Init, FromSerializedDataCtrl, FromSerializedData, ToSerializableDataCtrl, ToSerializableData (Not supposed to be used from outside; their specification can be read in the sources.)

SetHandler (Class Method)

Registers handlers for serializing and deserializing data.

Serialize (Class Method)

```
>>-Serialize(data)-------------------------------------------------------------><
```

This method returns the serialized form of the data object passed. The return value is a mutable buffer storing the serial objects.

DeSerialize (Class Method)

```
                           +-,-0------+
>>-DeSerialize(Buffer-----+----------+-)-------------------------------------><
                           +-,-Offset-+
```

This method deserializes data from a mutable buffer or string. The optional argument offset can be used to start at a different object.

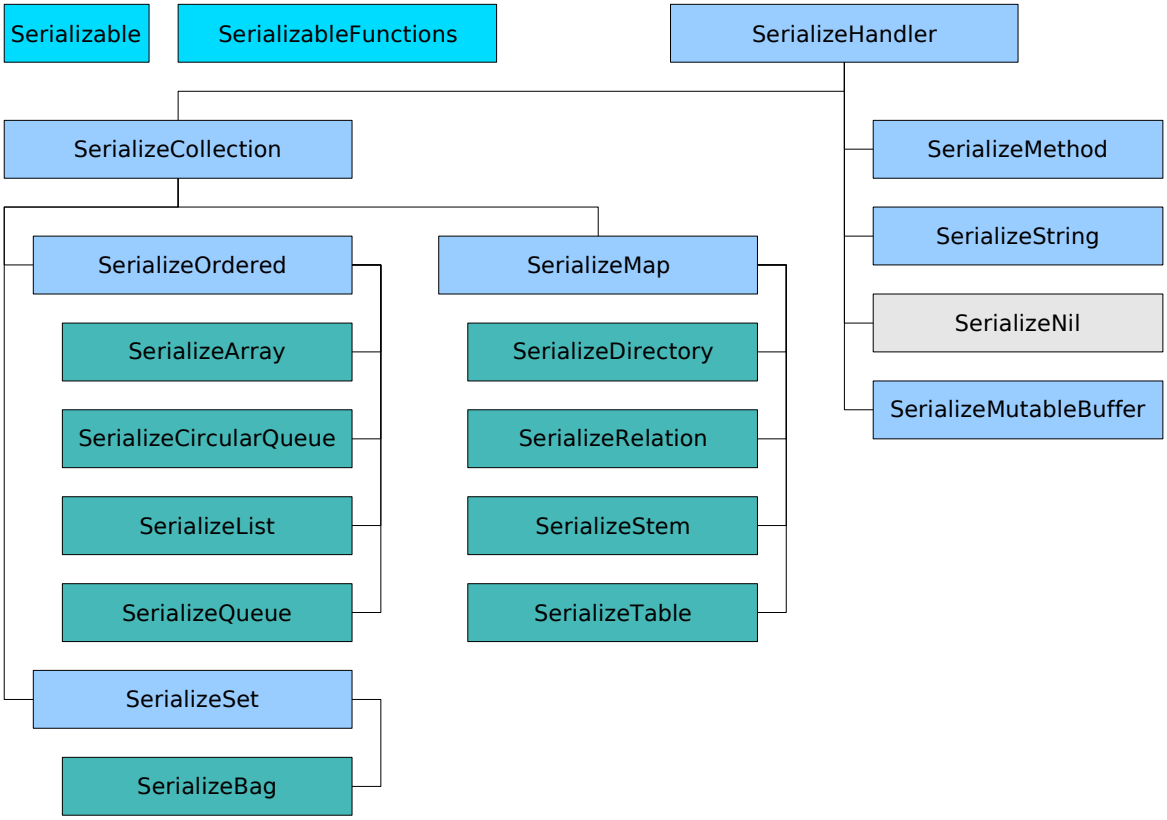
The SerializeHandler class

This class is a super class for all handlers. Handlers are classes for serializing and deserializing a specific data type. There are handlers for most built in classes that can be serialized, but this can be enhanced by custom handlers to support other objects which do not support serialization themselves.

A data handler has to define several methods. This leads to the constrain that all handlers must inherit from the SerializeHandler.

- Name: return the identifier used to store the data
- Serialize(Caller,Buffer,Object) – appends data to buffer
- DeSerialize(Caller,Arguments): object -- returns an instance of the object it is responsible for

Calls to the INIT class method must be forwarded. All custom handlers need to inherit from CheckType.



The data language

The format to store the data is a string format based on lines. Each line represent one object. Objects can refer to other lines by their line number (the array index).

index: Integer number.
reference: Position relative to the start of the root of the tree.
string: Base 64 encoded string.
object: “ARRAY” ({**index**} “:” {**reference**} “ “)*
| (“BAG” | “CIRCULARQUEUE” | “LIST” | “QUEUE”) ({**reference**} “ “)*
| “RELATION” ({**reference**} ”:” ({**reference**}”,”)*? {**reference**}?)*
| (“DIRECTORY” | “STEM” ({**string**} | ”-”) | “TABLE”) ({**reference**} “:” {**reference**})*
| (“METHOD” {0-7}| “MUTABLEBUFFER” | “OBJECT”) {**reference**}
| “S” {**string**}
| “NIL”
| “OBJECT” {**reference**} {**reference**}
line: ({**object**} “.”)+ “;” {**line**}?

The entry for serializable object first references a encoded class ID, the next reference points to the instance data.

Changes

Version 1.9 080501

- SerializeSerializable handler removed, arbitrary objects are now handled directly during the (de-)serialization process. This required a change to the data language.
- Minor changes to the test application, formatting of objects and relations improved.

Version 1.3

- Several updates: general clean ups.
- Corrected indentation
- Changed deserialization to make use of .String~makeArray, increases speed by about 30%
- Updated ObjectStream class to use new stream classes.
- Several updates to the test.cmd.

Version 0.7

- test.cmd updated
- Strings are marked with “S”, was “STRING” before
- added serializable.testUnit