The Object Rexx Serializing framework

Version 0.6

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:

Classes (Class Method)

INIT (overwrites Object class and Class class method)

WriteObject

ReadObject (Class Method)

Methods inherited from the Object class:

...

Classes (Class Method)

This attribute stores all sub classes currently known. It is managed by INIT (Class) and read by helper classes.

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 Serilizable class generates a SerializeSerializable instance that is able to serilaize 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.)
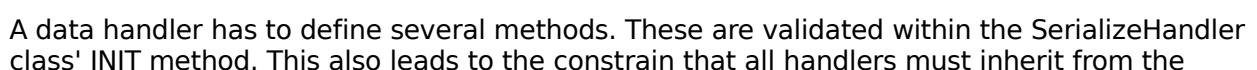
SetHandler (Class Method)

Registers handlers for serializing and deserializing data. See chapter on handlers for more information. Should only be called by the SerializeHandler INIT class method.

Serialize (Class Method)

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

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

DeSerialize (Class Method)

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

This method deserializes data from a mutable buffer. 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.

Serializable   SerializableFunctions   SerializeHandler

SerializeSerializable:
Instances are used to serialize data.

SerializeNil:
Used as default handler for unknown objects.

SerializeArray	SerializeBag
SerializeCircularQueue	SerializeDirectory
SerializeList	SerializeMethod
SerializeMutableBuffer	SerializeQueue
SerializeRelation	SerializeSerializable
SerializeStem	SerializeString
SerializeTable	SerializeNil

A data handler has to define several methods. These are validated within the SerializeHandler class' INIT method. This also leads to the constrain that all handlers must inherit from the

SerializeHandler.
- Name: return the identifier used to store the data
- CheckType: class -- The class this handler can serialize
- 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.

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 object.
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" | "MUTABLEBUFFER" | "OBJECT") {reference}
        | "STRING" {string}
        | "NIL"
line:     ({object} ".")+ ";" {line}?
```