14.4. `xdrlib` — Encode and decode XDR data

Source code: Lib/xdrlib.py

The `xdrlib` module supports the External Data Representation Standard as described in **RFC 1014**, written by Sun Microsystems, Inc. June 1987. It supports most of the data types described in the RFC.

The `xdrlib` module defines two classes, one for packing variables into XDR representation, and another for unpacking from XDR representation. There are also two exception classes.

class `xdrlib.`**`Packer`**

`Packer` is the class for packing data into XDR representation. The `Packer` class is instantiated with no arguments.

class `xdrlib.`**`Unpacker`**(*data*)

`Unpacker` is the complementary class which unpacks XDR data values from a string buffer. The input buffer is given as *data*.

See also:

RFC 1014 - XDR: External Data Representation Standard

This RFC defined the encoding of data which was XDR at the time this module was originally written. It has apparently been obsoleted by **RFC 1832**.

RFC 1832 - XDR: External Data Representation Standard

Newer RFC that provides a revised definition of XDR.

14.4.1. Packer Objects

`Packer` instances have the following methods:

`Packer.`**`get_buffer`**()

Returns the current pack buffer as a string.

`Packer.`**`reset`**()

Resets the pack buffer to the empty string.

In general, you can pack any of the most common XDR data types by calling the appropriate `pack_type()` method. Each method takes a single argument, the value to pack. The following simple data type packing methods are supported: `pack_uint()`, `pack_int()`, `pack_enum()`, `pack_bool()`, `pack_uhyper()`, and `pack_hyper()`.

`Packer.`**`pack_float`**(*value*)

Packs the single-precision floating point number *value*.

`Packer.`**`pack_double`**(*value*)

Packs the double-precision floating point number *value*.

The following methods support packing strings, bytes, and opaque data:

`Packer.`**`pack_fstring`**(*n*, *s*)

Packs a fixed length string, *s*. *n* is the length of the string but it is *not* packed into the data buffer. The string is padded with null bytes if necessary to guaranteed 4 byte alignment.

`Packer.`**`pack_fopaque`**(*n*, *data*)

Packs a fixed length opaque data stream, similarly to `pack_fstring()`.

`Packer.`**`pack_string`**(*s*)

Packs a variable length string, *s*. The length of the string is first packed as an unsigned integer, then the string data is packed with `pack_fstring()`.

`Packer.`**`pack_opaque`**(*data*)

Packs a variable length opaque data string, similarly to `pack_string()`.

`Packer.`**`pack_bytes`**(*bytes*)

Packs a variable length byte stream, similarly to `pack_string()`.

The following methods support packing arrays and lists:

`Packer.`**`pack_list`**(*list*, *pack_item*)

Packs a *list* of homogeneous items. This method is useful for lists with an indeterminate size; i.e. the size is not available until the entire list has been walked. For each item in the list, an unsigned integer `1` is packed first, followed by the data value from the list. *pack_item* is the function that is called to pack the individual item. At the end of the list, an unsigned integer `0` is packed.

For example, to pack a list of integers, the code might appear like this:

```
import xdrlib
p = xdrlib.Packer()
p.pack_list([1, 2, 3], p.pack_int)
```

`Packer.`**`pack_farray`**(*n*, *array*, *pack_item*)

Packs a fixed length list (*array*) of homogeneous items. *n* is the length of the list; it is *not* packed into the buffer, but a `ValueError` exception is raised if `len(array)` is not equal to *n*. As above, *pack_item* is the function used to pack each element.

`Packer.`**`pack_array`**(*list*, *pack_item*)

Packs a variable length *list* of homogeneous items. First, the length of the list is packed as an unsigned integer, then each element is packed as in `pack_farray()` above.

14.4.2. Unpacker Objects

The `Unpacker` class offers the following methods:

`Unpacker.`**`reset`**(*data*)

Resets the string buffer with the given *data*.

`Unpacker.`**`get_position`**()

Returns the current unpack position in the data buffer.

`Unpacker.`**`set_position`**(*position*)

Sets the data buffer unpack position to *position*. You should be careful about using `get_position()` and `set_position()`.

`Unpacker.`**`get_buffer`**()

Returns the current unpack data buffer as a string.

`Unpacker.`**`done`**()

Indicates unpack completion. Raises an `Error` exception if all of the data has not been unpacked.

In addition, every data type that can be packed with a `Packer`, can be unpacked with an `Unpacker`. Unpacking methods are of the form `unpack_type()`, and take no arguments. They return the unpacked object.

`Unpacker.`**`unpack_float`**()

Unpacks a single-precision floating point number.

`Unpacker.`**`unpack_double`**()

Unpacks a double-precision floating point number, similarly to `unpack_float()`.

In addition, the following methods unpack strings, bytes, and opaque data:

`Unpacker.`**`unpack_fstring`**(*n*)

Unpacks and returns a fixed length string. *n* is the number of characters expected. Padding with null bytes to guaranteed 4 byte alignment is assumed.

`Unpacker.`**`unpack_fopaque`**(*n*)

Unpacks and returns a fixed length opaque data stream, similarly to `unpack_fstring()`.

`Unpacker.`**`unpack_string`**()

Unpacks and returns a variable length string. The length of the string is first unpacked as an unsigned integer, then the string data is unpacked with `unpack_fstring()`.

`Unpacker.`**`unpack_opaque`**()

Unpacks and returns a variable length opaque data string, similarly to `unpack_string()`.

`Unpacker.`**`unpack_bytes`**()

Unpacks and returns a variable length byte stream, similarly to `unpack_string()`.

The following methods support unpacking arrays and lists:

`Unpacker.`**`unpack_list`**(*unpack_item*)

Unpacks and returns a list of homogeneous items. The list is unpacked one element at a time by first unpacking an unsigned integer flag. If the flag is `1`, then the item is unpacked and appended to the list. A flag of `0` indicates the end of the list. *unpack_item* is the function that is called to unpack the items.

`Unpacker.`**`unpack_farray`**(*n*, *unpack_item*)

Unpacks and returns (as a list) a fixed length array of homogeneous items. *n* is number of list elements to expect in the buffer. As above, *unpack_item* is the function used to unpack each element.

`Unpacker.`**`unpack_array`**(*unpack_item*)

Unpacks and returns a variable length *list* of homogeneous items. First, the length of the list is unpacked as an unsigned integer, then each element is unpacked as in `unpack_farray()` above.

14.4.3. Exceptions

Exceptions in this module are coded as class instances:

exception `xdrlib.`**`Error`**

The base exception class. `Error` has a single public attribute `msg` containing the description of the error.

exception `xdrlib.`**`ConversionError`**

Class derived from `Error`. Contains no additional instance variables.

Here is an example of how you would catch one of these exceptions:

```
import xdrlib
p = xdrlib.Packer()
try:
    p.pack_double(8.01)
except xdrlib.ConversionError as instance:
    print('packing the double failed:', instance.msg)
```