NAME

xo_open_list, **xo_open_list_h**, **xo_open_list_hd**, **xo_open_list_d xo_open_instance**, **xo_open_instance_h**, **xo_open_instance_hd**, **xo_open_instance_d xo_close_instance**, **xo_close_instance_h**, **xo_close_instance_hd**, **xo_close_instance_d xo_close_list**, **xo_close_list_h**, **xo_close_list_hd**, **xo_close_list_d** - open and close lists and instances

LIBRARY

Text, XML, JSON, and HTML Output Emission Library (libxo, -lxo)

SYNOPSIS

#include <libxo/xo.h>

xo_ssize_t
xo_open_list_h(*xo_handle_t *xop, const char *name*);

xo_ssize_t
xo_open_list(*const char *name*);

xo_ssize_t
xo_open_list_hd(*xo_handle_t *xop, const char *name*);

xo_ssize_t
xo_open_list_d(*const char *name*);

xo_ssize_t
xo_open_instance_h(*xo_handle_t *xop, const char *name*);

xo_ssize_t
xo_open_instance(*const char *name*);

xo_ssize_t
xo_open_instance_hd(*xo_handle_t *xop, const char *name*);

xo_ssize_t
xo_open_instance_d(*const char *name*);

xo_ssize_t
xo_close_instance_h(*xo_handle_t *xop, const char *name*);

xo_ssize_t

xo_close_instance(*const char *name*);

xo_ssize_t
xo_close_instance_hd(*xo_handle_t *xop*);

xo_ssize_t
xo_close_instance_d(*void*);

xo_ssize_t
xo_close_list_h(*xo_handle_t *xop*, *const char *name*);

xo_ssize_t
xo_close_list(*const char *name*);

xo_ssize_t
xo_close_list_hd(*xo_handle_t *xop*);

xo_ssize_t
xo_close_list_d(*void*);

DESCRIPTION

Lists are sequences of instances of homogeneous data objects. Two distinct levels of calls are needed to represent them in our output styles. Calls must be made to open and close a list, and for each instance of data in that list, calls must be make to open and close that instance.

The name given to all calls must be identical, and it is strongly suggested that the name be singular, not plural, as a matter of style and usage expectations.

A list is a set of one or more instances that appear under the same parent. The instances contain details about a specific object. One can think of instances as objects or records. A call is needed to open and close the list, while a distinct call is needed to open and close each instance of the list:

```
xo_open_list("item");

for (ip = list; ip->i_title; ip++) {
    xo_open_instance("item");
    xo_emit("{L:Item} '{:name/%s}':\n", ip->i_title);
    xo_close_instance("item");
}

xo_close_list("item");
```

Getting the list and instance calls correct is critical to the proper generation of XML and JSON data.

EXAMPLE:

```
xo_open_list("user");
for (i = 0; i < num_users; i++) {
    xo_open_instance("user");
    xo_emit("{k:name}:{:uid/%u}:{:gid/%u}:{:home}\n",
            pw[i].pw_name, pw[i].pw_uid,
            pw[i].pw_gid, pw[i].pw_dir);
    xo_close_instance("user");
}
xo_close_list("user");
```

TEXT:

```
phil:1001:1001:/home/phil
pallavi:1002:1002:/home/pallavi
```

XML:

```
<user>
  <name>phil</name>
  <uid>1001</uid>
  <gid>1001</gid>
  <home>/home/phil</home>
</user>
<user>
  <name>pallavi</name>
  <uid>1002</uid>
  <gid>1002</gid>
  <home>/home/pallavi</home>
</user>
```

JSON:

```
user: [
  {
    "name": "phil",
    "uid": 1001,
    "gid": 1001,
    "home": "/home/phil",
  },
  {
    "name": "pallavi",
    "uid": 1002,
    "gid": 1002,
```

```
            "home": "/home/pallavi",
        }
    ]
```

LEAF LISTS

In contrast to a list of instances, a "leaf list" is list of simple values. To emit a leaf list, call the **xo_emit**() function using the ""l"" modifier:

```
for (ip = list; ip->i_title; ip++) {
        xo_emit("{Lwc:Item}{l:item}\n", ip->i_title);
}
```

The name of the field must match the name of the leaf list.

In JSON, leaf lists are rendered as arrays of values. In XML, they are rendered as multiple leaf elements.

```
JSON:
  "item": "hammer", "nail"
XML:
  <item>hammer</item>
  <item>nail</item>
```

SEE ALSO

xo_emit(3), libxo(3)

HISTORY

The **libxo** library first appeared in FreeBSD 11.0.

AUTHORS

libxo was written by Phil Shafer <*phil@freebsd.org*>.