Enduro/X Core - Support #251

Consider moving to heap-allocation of message buffers

11/20/2017 10:19 AM - Madars

Status:	Closed	**Start date:**	11/20/2017
Priority:	Normal (Code 4)	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Seems like for example Golang default cgo stack size is 2MB. This could cause problems for runtimes, where big messages are used. For example 1M test with tpenqueue/dequeue in Golang fails.

History

#1 - 11/21/2017 01:25 PM - Madars

```
(gdb) where
#0  0x0000000000444d09 in ?? ()
#1  0x0000000800e8fd00 in ndrx_tpviewtojson (cstruct=0x0, view=<value optimized out>, buffer=0x0, bufsize=0, flags=0) at view2exjson.c:458
#2  0x00000000004453b3 in ?? ()
#3  0x000000c4201b3af8 in ?? ()
#4  0x000000000000000b in ?? ()
#5  0x000000c420334180 in ?? ()
#6  0x0000000000000000 in ?? ()
(gdb)
#0  0x0000000000444d09 in ?? ()
#1  0x0000000800e8fd00 in ndrx_tpviewtojson (cstruct=0x0, view=<value optimized out>, buffer=0x0, bufsize=0, flags=0) at view2exjson.c:458
#2  0x00000000004453b3 in ?? ()
#3  0x000000c4201b3af8 in ?? ()
#4  0x000000000000000b in ?? ()
#5  0x000000c420334180 in ?? ()
#6  0x0000000000000000 in ?? ()
(gdb)
#0  0x0000000000444d09 in ?? ()
#1  0x0000000800e8fd00 in ndrx_tpviewtojson (cstruct=0x0, view=<value optimized out>, buffer=0x0, bufsize=0, flags=0) at view2exjson.c:458
#2  0x00000000004453b3 in ?? ()
#3  0x000000c4201b3af8 in ?? ()
#4  0x000000000000000b in ?? ()
#5  0x000000c420334180 in ?? ()
#6  0x0000000000000000 in ?? ()
(gdb) quit
```

Code:

```
/**
 * Build json text from UBF buffer
 * @param p_ub  JSON buffer
 * @param buffer output json buffer
 * @param bufsize       output buffer size
 * @param flags BVACCESS_NOTNULL -> return only non NULL values, if not set,
 * return all
 * @return SUCCEED/FAIL
 */
expublic int ndrx_tpviewtojson(char *cstruct, char *view, char *buffer,
        int bufsize, long flags)
{
    int ret = EXSUCCEED;
    int occs;
    int is_array;
    double d_val;
    char strval[CARR_BUFFSIZE+1];
```

```
char b64_buf[CARR_BUFFSIZE_B64+1]; <<< 458
```

So basically it got core on stack entry...

#2 - 09/23/2019 06:22 PM - Madars

we could write some allocator which would keep the list of free blocks.
If new block is required, it would check and return from free list, if no block available, then do malloc.
Program could keep a count of blocks allocate in free list, if overreaches a global limit, then perform the free. This could be done when blocks is returned to list.

the spin lock could be used only when fetching, adding to free list.

#3 - 07/05/2020 01:17 PM - Madars

- Status changed from New to Rejected

Duplicate of Feature #549

#4 - 07/05/2020 01:18 PM - Madars

- Status changed from Rejected to Closed