Code Documentation Project check list

Goals for this project:

- Describe what the function does. Do not describe what the function is used for.
- The following forms should be used:
  - method_table::method_name
  - function_name()
  - array_name[]
- Add a description at the start of the file that gives the reader an overall idea of what the functions in this file are for, and a bit of background on the code itself. This is expected to be at least a paragraph or two, but can be considerably longer if the file is important for the code Lustre operation, or implements a complex algorithm that may not be easily understood just from reading the functions
- Update the Intel copyright line at the top of the file to 2014. If there is an Oracle or Sun copyright line, or a notice about the Lustre trademark, it should be left alone.
- Replace the Sun URL for the GPLv2 with http://www.gnu.org/licenses/gpl-2.0.html
- Add a comment block for each function in the file that is more than two or three lines long. Include all important details for users of this function:
  - external locking requirements, or if function changes the locking state itself
  - requirements on buffer sizes, memory that is allocated or freed, etc.

See Coding Guidelines for details on the formatting of the function comments themselves. The function comment should contain most or all of the comments for the function, so that the reader can understand what the function is doing, instead of having comments spread throughout the function. Comments inline with the code should be short and be used to point out some tricky bit of implementation detail that needs to be explained. For example:

```
/**
 * Implements method_table::method_name() method for subsystem.
 *
 * Provide a description of the function here, including any details
 * about the function that may be important.  This includes information
 * about locks held by the caller or other requirements that the function
 * has of the calling function and how \a var1 and \a var2 might be used.
 *
 * \param[in]     var1    description of this variable
 * \param[in]     var2    another description of a variable,
 *                        with more detail than fits on one line
 * \param[in,out] var3    this one is for both input and output
 *
 * \retval        0       success
 * \retval        -EAGAIN need to call this function again
 * \retval        -ENOMEM error allocating some structure
 * \retval        negative error number for other errors
 */
```

- As you work through the functions in a file, you may see other technical debts that should be fixed. While it is OK to fix a bit of whitespace, do NOT fix everything in the documentation patch, since this will make it more difficult to inspect the patch and ensure that it does not introduce some problem itself. Make separate patches to fix problems found in the code, and/or keep a separate list of issues that need to be fixed later.
- If there is some part of the code in the file that you are working on that you don't completely understand, look who else worked on the function recently using "`git blame path/to/file.c`"

then chat or call someone in Skype to discuss the code and get a good understanding of it to write a good comment. Failing that, write what you *think* the function is doing, and then add the original authors as inspectors for the patch.

- Don't worry if you don't understand everything 100% - it is better to have *something* written down that can be improved, than having nothing at all.
- Don't worry about proper English grammar. That can be fixed up by others during patch inspection, once the technical content is written.
- Once you have a patch, push it to Gerrit and select inspectors that also are familiar with the code in this file (also using "`git blame path/to/file.c`" to see other developers who worked on the file if needed).

file	LOC	assigned engineer	patch complete	patch landed
dt_object	1614	AZ	in review	
LOD				
lod_dev.c	942	AZ	in review	
lod_lov.c	1164	AZ	in review	
lod_object.c	2428	AZ	in review	
lod_pool.c	667	AD	in review	
lod_qos.c	1465	AZ	in review	
lproc_lod.c	549	AZ	in review	
OSP				
lproc_osp.c	537	LS (AZ)	in review	
lwp_dev.c	437	NY, WD	in review	
osp_dev.c	1305	WD	in review	
osp_md_object.c	715	WD	in review	
osp_object.c	1627	FY (AZ, MP)	in review	
osp_precreate.c	1328	AZ, MP, WD	in review	
osp_sync.c	1325	AZ, MP	in review	
osp_trans.c	402	WD, FY	LANDED	a0da0ad14
OFD				
lproc_ofd.c	661	MP	in review	
ofd_capa.c	231	LS	LANDED	886472a7
ofd_dev.c	2417	AZ, MP, JL	in review	
ofd_dlm.c	258	MP	in review	
ofd_fdm.c	245	MP	in review	
ofd_fs.c	515	AZ, MP	LANDED	94a6bc2a
ofd_grant.c	1081	JL	LANDED	b808da75
ofd_io.c	1013	MP	in review	
ofd_lvb.c	315	AZ, MP	in review	
ofd_obd.c	1187	AD, AZ, MP	in review	

ofd_objects.c	682	AZ, MP	in review	
ofd_trans.c	79	AZ, MP	in review	

Quick links in Gerrit:

http://review.whamcloud.com/#/q/message:LU-4974,n,z

http://review.whamcloud.com/#/q/message:LU-4975,n,z

http://review.whamcloud.com/#/q/message:LU-4976,n,z