Document No: WG21 N3906
Date: 2014-02-03
Project: Programming Language C++
References: SC22 N4901, SC22 N
Reply to: Barry Hedquist
INCITS/PL22.16 IR
beh@peren.com

ISO/IEC PDTS 18822, File System, National Body Comments

Attached is a revision of SC22 N4901, the complete set of National Body Comments submitted to JTC1 SC22 in response to the SC22 Letter Ballot for ISO/IEC PDTS 18822, File System.

Comments that were submitted without numbering were numbered manually in the exact order of the NB's official ballot response. Those with numbering were left as submitted. A key was added at the end of the ballot comments to indicate the designations used for those National Bodies submitting comments:

CH -	Switzerland
FI -	Finland
GB -	Great Britain
US -	United States

Responses to each of the National Body comments is required prior to the next ballot round, DTS.

Also attached is the complete list of ballot responses from all National Bodies. One National Body, Switzerland, voted to Disapprove the PDTS. Switzerland also indicated that their vote will be changed to Approve if their comments are adopted.

Document numbers referenced in the ballot comments are WG21 documents unless otherwise stated.

Template for comments and secretariat observations

Date:2014-02-03	Document: WG21 N3906	Project: PDTS 18822

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CH 1	all			ge	Please coordinate with the networking study group to make sure that the pathname grammar works together with the URI grammar (pathnames should be a subset of URIs).		
FI 5				ge/ed	The namespace for the filesystem should be std::experimental, not std::tbd	Change the namespace to experimental everywhere tbd occurs.	
US 1		all		ed	Many cross-references contain the symbolic section name rather than a link to that section. Examples include occurrences in 1 ([fs.norm.ref]), 2.1 ([fs.def.race]), 2.2 ([fs.def.osdep]), etc.	Replace cross-references using symbolic section names with actual links to the referenced section.	
US 2		all		ed	References to the C++ Standard are given in various forms. For example, the first paragraph of 13 [class.directory_iterator] says, "`directory_iterator` satisfies the requirements of an input iterator ([input.iterators])," with no indication that the referenced section is part of the C++ Standard., while the *effects* clause of `operator++` in 13.1 [directory_iterator.members] uses the full form "the C++ Standard, 24.1.1 Input iterators [input.iterators]." Sometimes the full word "Standard" is used and sometimes "Std".	Use a canonical form for all references to the C++ Standard.	
US 3		all		ed	The pattern of cross-references from synopses to the description of an entity is inconsistent. For example, in the synopsis of header `<filesystem>` in 6 [fs.filesystem.synopsis], the link for `file_type` is found on the symbolic section name in a comment, while for `file_status` it is on the class name. Two of the `copy` overloads have links, while two do not. In the synopsis for class `path` in 8 [class.path], some of the member functions have links from the function name, while others have links from preceding comment text	Use a consistent pattern of cross-references. A link on each entity name would probably be most useful.	
US 4		all		ed	Paragraphs are not numbered.	Add paragraph numbers.	
US 5		various		te	Namespace `tbd` is used but not described. It is not clear whether `tbd` is intended to be replaced by a different name at some point in the future	Add a note clarifying the intent of using that name.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					(i.e., is an abbreviation for "To Be Determined") or if it is intended to be an actual name.		
US 6		2.1		ed	"#error" is in body font.	Set "#error" in program font.	
FI 1		2.1 POSIX conformance [fs.conform.9945]		te	It is unfortunate that error reporting for inability to provide reasonable behaviour is completely implementation-defined. This hurts portability in the sense that programmers have no idea how errors will be reported and cannot anticipate anything.	Change “If an implementation cannot provide any reasonable behavior, the implementation shall report an error in an implementation-defined manner.” to “If an implementation cannot provide any reasonable behavior, the code using the facilities for which reasonable behaviour cannot be provided shall be ill-formed.” and strike the note. .	
US 7		3		ed	"C++" is split across lines.	Eliminate the line break between the "+"s.	
GB 2	Page 2	4		Ge	‘extension’ should be added to the terms and conditions as the term 'extension' is not defined in 4 [fs.definitions] but is used in the Remarks for extension() in 8.4.9	The term 'extension' should be added to 4 [fs.definitions] and the term then used in the effects for replace_extension (8.4.5)	
CH 2		4.7 [fs.def.filename]		te	Filename lengths are also implementation dependent. This is not the same as FILENAME_MAX that specifies the maximum length of pathnames.	Add a bullet: “Length of filenames.”	
US 8		4.14		ed	"dot-dot" is referred to as a "pathname." Everywhere else it is referred to as a "filename."	Change "pathname" to "filename".	
CH 3		4.14 [fs.def.parent]		te	The concept of a parent directory for dot or dotdot exists, but the definition doesn’t apply.	Remove the paragraph “This concept does not apply to dot and dot-dot.” Add a definition for dot and dot-dot.	
CH 4		4.14 [fs.def.parent]		te	8.1 [path.generic] says: “The filename dot-dot is treated as a reference to the parent directory.” So it must be specified what “/..” and “/../..” refer to.	Add a statement what the parent directory of the root directory is.	
CH 5		4.15 [fs.def.path]		te	Path depth is implementation dependent.	Add a paragraph: "The maximum length of the sequence (i.e. the maximum depth) is implementation dependent.	
US 9		4.18		ed	"." and ".." are referred to as "paths". The distinction between "path" and "pathname" is not	Change "path" to "pathname" in the reference to "." and "..".	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	**Line number**	**Clause/ Subclause**	**Paragraph/ Figure/Table**	**Type of comment**[2]	**Comments**	**Proposed change**	**Observations of the secretariat**
					completely clear, but those would appear to fit the definition of "pathname" better than "path," i.e., "A character string that represents the name of a path," since "." and ".." are character strings. (It's also not clear when "." and ".." are used and when "dot" and "dot-dot" are used.)		
US 10		4.19		ed	The definition of a symbolic link is given as, "A link with the property that when the file is encountered..." This assumes that links are files, which is not how they are described in 4.9 [fs.def.link]: "A directory entry object that associates a filename with a file."	Assuming that the intent is that symbolic links are files and hard links are not, the definition should be reworded to something like, "A link that is a file with the property..."	
GB 3	Page 5	6	P1	Ge	The namespace 'tbd' needs clarification	The synposes all belong in namespace std::tbd::filesystem with no indication of how the eventual namespace to replace 'tbd' will be arrived at.	
GB 4	Page 6	6	synopsis	Te	Use of the term a 'non-privileged' process The comment for available in the `struct space_info` refers to: free space available to a non-privileged process This seems quite specific to a POSIX implementation (on Windows, for instance, the equivalent data would be user-specific but not directly related to privilege)	Remove the comment and add a note to 15.32 [fs.op.space]: [Note: the precise meaning of available space is implementation dependent. --end note]	
CH 6		6 [fs.filesystem. synopsis]		te	The name for the namespace is currently *tbd*.	Specify an actual name for the namespace.	
CH 7		6 [fs.filesystem. synopsis]		te	Must the file_time_type hold times before 1960 and after 2050?	Specify the requirements to unspecified-trivial-clock for file_time_type.	
FI 2		6 Header <filesystem> synopsis [fs.filesystem. synopsis]		te	It is unclear why the range-for support functions (begin()/end()) for directory_iterator and recursive_directory_iterator return different types for begin() and end(), namely that begin() returns a reference to const and end() returns a value.		
FI 4		6 Header <filesystem>		te	It is unclear why copy, copy_file and copy_symlink have different return types, and why the attribute-	Make the status/error reporting consistent or add a note explaining why it's different.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
		synopsis [fs.filesystem. synopsis]			version of create_directory has a different return type than the create_directory that takes no attributes. The status/error reporting in these functions seems inconsistent.		
FI 3		6 Header <filesystem> synopsis [fs.filesystem. synopsis], 8 Class path [class.path]		ed	The specification uses InputIterator begin, InputIterator end in various places for the names of iterator parameters. This makes searching for the functions begin()/end() hard and is inconsistent with the style used in the C++ standard.	Change “InputIterator begin, InputIterator end” to “InputIterator first, InputIterator last”.	

GB 1		6 & 15		Te	There is no relative() operation, to complement both absolute() and canonical()
					
					The TS introduces relative paths.
					
					• They are defined in section **4.18 relative path [fs.def.relative-path]**
					• A decomposition method relative_path() is described in section **8.4.9 path decomposition [path.decompose]**
					• Two query methods to determine if a path either has_relative_path() or is_relative() described in **8.4.10 path query [path.query]**
					
					However there is no way to create a relative path as a path relative to another. Methods are provided to create absolute and canonical paths.
					In section **15.1 Absolute [fs.op.absolute]**:
					path absolute(const path& p, const path& base=current_path());
					and in section **15.2 Canonical [fs.op.canonical]**
					path canonical(const path& p, const path& base = current_path());
					path canonical(const path& p, error_code& ec);
					path canonical(const path& p, const path& base, error_code& ec);
					By providing a operations to achieve absolute and canonical paths there is no impediment to
						Modify section:
						
						6 Header <filesystem> synopsis [fs.filesystem.synopsis]
						
						by adding the operational functions after canonical:
						
						path relative(const path& p, const path& to = current_path());
						path relative(const path& p, error_code& ec);
						path relative(const path& p, const path& to, error_code& ec);
						
						Insert the section:
						
						15.3 Relative [fs.op.relative]
						path relative(const path& p, const path& to = current_path());
						path relative(const path& p, error_code& ec);
						path relative(const path& p, const path& to, error_code& ec);
						
						Overview: Return a relative path of p to the current directory or from an optional to path.
						
						Returns: A relative path such that	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat

					providing a similar operation relative() that attempts to return a new path relative to some base path.
					For example:
					path relative(const path& p, const path& to = current_path());
					path relative(const path& p, error_code& ec);
					path relative(const path& p, const path& to, error_code& ec);
					This would return a path, if possible, that is relative to to. The implementation can make use of absolute() and canonical() to determine the relative path, if it exists.
					The File System TS is based on the boost::filesystem library and it too suffers from this anomaly. There are open tickets for this in Boost Trac:
					• #5897 Make path relative function
					• #1976 Inverse function for complete
					and it is the subject of several posts on StackOverflow for example:
					• http://stackoverflow.com/questions/10167382/boostfilesystem-get-relative-path
					• http://stackoverflow.com/questions/5772992/get-relative-path-from-two-absolute-paths
					Other languages typically provide a similar function. For example python provides: os.path.relpath(path[, start])
					Return a relative filepath to path either from the current directory or from an optional start directory. This is a path computation: the filesystem is
						canonical(to)/relative(p,to) == canonical(p), otherwise path(). If canonical(to) == canonical(p) the path path(".") is returned. For the overload without a to argument, to is current_path(). Signatures with argument ec return path() if an error occurs.
						Throws: As specified in Error reporting.
						Remarks: !exists(p) or !exists(to) or !is_directory(to) is an error.
						and bump all following sections up by 0.1. Update the contents and any cross-references accordingly.
						Question: Should *Returns* be specified in terms of equivalence? For example: equivalent(canonical(to)/relative(p,to), canonical(p))
						Question: Should canonical(to) == canonical(p) return path(".") or path()? Why?
						Question: Should to be spelt start?	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					not accessed to confirm the existence or nature of path or start. start defaults to os.curdir.		
CH 8		6 [fs.filesystem. synopsis], 15.14 [fs.op.file_size]		te	uintmax_t is specified to hold at least 64 bit. This is not enough for sizes beyond 4 exabytes.	Specify whether an implementation must provide a uintmax_t that can hold the maximum possible space and file size values.	
GB 5	Page 10	7		Ed	Duplicate word: "appropriate appropriate" appears in the last bullet in 7 [fs.err.report]	Delete the duplicate	
US 11		7		ed	The next-to-last sentence has a duplicated word: "...argument is set as appropriate appropriate for the specific error."	Delete one instance of "appropriate."	
CH 9		7 [fs.err.report], all		te	The specification of the actual error conditions for the functions that specify Throws: As specified in Error reporting. is missing.	Add those specifications.	
FI 6		8 Class path [class.path]		ed	It doesn't seem to be necessary to explicitly =default destructors.	Remove superfluous declarations.	
GB 6	Page 14	8.1		Ed	Spurious whitespace in the grammar for directory-separator: "slash directory-separator" is indented more than the other three items.	Align the four items	
US 12		8.1		ed	The second production in the grammar for *directory-separator* is indented too far.	Align "*slash directory-separator*" with the other productions.	
US 13		8.3	final bullet	ed	The third option for the `Source` template parameter is, "A character array that after array-to-pointer decay results in a pointer to a NTCTS." NTCTS is defined in 4.12 [fs.def.ntcts] as "a sequence of values of a given encoded character type terminated by that type's null character." Given that definition and the fact that array-to-pointer decay results in a pointer to the first element of the array, there does not appear to be a way for a character string to satisfy this	Reword to say, "A character array that is a NTCTS."	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					requirement, since a single character cannot be a NTCTS.		
GB 7	Page 16	8.4.1		Te	Incorrect postcondition for copy/move constructor The postconditions for the copy/move constructor for path are shown as "empty()". This appears to have been incorrectly copied from the default ctor.	Remove the 'postconditions' clause from the copy/move ctor.	
GB 8	Page 17	8.4.1		Ge	No specification for characters with no representation. The example at the end of 8.4.1 refers to "for other native narrow encodings some characters may have no representation" — what happens in such cases?	Add some definition of the behaviour for characters with no representation.	
US 14		8.4.1	example	ed	The last line in the code of the example is missing a right parenthesis.	Add a right parenthesis before the semicolon.	
CH 10		8.4.1 [path.construct]		te	Postcondition for copy and move constructors is wrong.	Remove wrong postcondition.	
CH 11		8.4.3 [path.append]		te	Is the added separator redundant in p1 /= p2, where p1 is empty? I.e. does the result start with a separator?	Specify what behaviour is required.	
GB 10	Page 21	8.4.5		Ed	Use of “concatined”	In the Effects for replace_extension() change to concatenated in the phrase: "replacement is concatinated to the stored path".	
GB 9	Page 20	8.4.5		Ed	Unusual form of post-condition. The phrasing of the postcondition for clear() "this->empty() is true" differs from other similar postconditions throughout the ISO C++ standard and this TS.	Postcondition: empty()	
CH 12		8.4.5 [path.modifiers]		te	As we have move semantics, member swap functions shouldn’t be necessary anymore.	Remove swap().	
US 15		8.4.9		ed	The *returns* descriptions for `stem()` and `extension()` both contain the phrase, "...does not consist solely of one or to two periods." The word "to" seems extraneous.	Omit "to" in both descriptions.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)

2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
US 16		8.4.9		ed	The *returns* description for `stem()` begins, "if `filename()` contains a character but..." Comparison with the corresponding text in the `extension()` entry suggests that this should apply when the name contains a period, not when it contains any character.	Replace the phrase with, "if `filename()` contains a period but..."	
FI 7		8.4.10 path query [path.query]		te	is_absolute says: "Returns: true if the elements of root_path() uniquely identify a file system location, else false." The "uniquely identify a location" seems confusing in presence of symlinks.	Clarify the returns clause so that there's no confusion about symlinks and 'location'.	
FI 8		8.6.1 path inserter and extractor [path.io]		te	"[Note: Pathnames containing spaces require special handling by the user to avoid truncation when read by the extractor. --end note]" sounds like a quoted manipulator as specified in the C++14 draft in [quoted.manip] would be useful.	Consider using quoted manipulators for stream insertion and extraction.	
GB 11	Page 32	10.2		Ed	Confusing section titles for copy_options - use of "copy_file effects for", "copy effect for" and "copy effects"	Use "copy effects for" consistently	
GB 13	Page 38	12.2		Te	Incorrect variable name in postcondition - the postcondition on path() for replace_filename() is defined as `path().parent_path() / s`	Define the postcondition on path() as `path().parent_path() / p`	
GB 12	Page 38	12.3		Ge	Since operator== for directory_entry does not check status, it would be worth highlighting that operator== only checks that the paths match.	Add [Note: does not include status values - end note]	
CH 13		14.1 [class.rec.dir.itr.members]		te	The behaviour of increment is underspecified: What happens if the implementation detects an endless loop (e.g. caused by links)? What happens with automounting and possible race conditions? More information on this can be found at <http://man7.org/linux/manpages/man3/fts.3.html>.	Specify the required behaviour in these cases.	
CH 14		15		te	Since create_symlink(), create_hardlink(), and create_directory() exist, there's no reason not to have a create_regular_file() function.	Consider adding a function create_regular_file() with the behaviour of the POSIX touch command.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
GB 14	Page 46	15.3		Te	Incorrect effects clause for path copy - the effect clause for copy [fs.op.copy] includes "equivalent(f, t)" — there is no equivalent() function defined for variables of this type (file_status)	Replace with "equivalent(from, to)"	
CH 15		15.4 [fs.op.copy_file]		te	Even if to and from are different paths, they may be equivalent.	Specify what happens if (options & copy_options::overwrite_existing) but from and to resolve to the same file.	
CH 16		15.13 [fs.op.equivalent]		te	Equivalence is a volatile property.	Consider adding a note that equivalence cannot be determined race-free.	
FI 9		15.15 Hard link count [fs.op.hard_lk_ct]		te	“The signature with argument ec returns static_cast<uintmax_t>(-1) if an error occurs.”, one would expect that BOTH signatures return that if an error occurs?	Clarify the Returns clause, and apply the same for every function that returns an uintmax_t where applicable.	
GB 15	Page 56	15.25		Te	The constraint on last_write_time is too weak: It is noted that the postcondition of last_write_time(p) == new_time is not specified since it might not hold for file systems with coarse time granularity. However, might it be possible to have a postcondition that last_write_time(p) <= new_time ?	Add postcondition: last_write_time(p) <= new_time	
GB 16	Page 57	15.27		Te	Unclear semantics of read_symlink on error: 15.27 [fs.op.read_symlink] has: Returns: If p resolves to a symbolic link, a path object containing the contents of that symbolic link. Otherwise path(). and also [Note: It is an error if p does not resolve to a symbolic link. -- end note] I do not believe path() can be a valid return for the overload not taking error_code.	Strike "Otherwise path(). "	
CH 17		15.28 [fs.op.remove]		te	The specification can be read to require the existence test. As this introduces a race, the existence test must not happen.	Change to: “Effects: p is removed as if by POSIX remove().”	
CH 18		15.30 [fs.op.rename]		te	POSIX guarantees some kind of atomicity for rename().	Clarify that POSIX’ rename() guarantee “If the rename() function fails for any reason other than [EIO], any file named by new shall be unaffected.” holds for C++ as well.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
FI 10		15.36 System complete [fs.op.system_ complete]			"[Example: For POSIX based operating systems, system_complete(p) has the same semantics as complete(p, current_path())." What is this 'complete' that is referred here?	Clarify the example.	
CH 19		15.38 [fs.op.unique_ path]		te	unique_path() is a security vulnerability. As the Linux manual page for the similar function tmpnam() writes in the "BUGS" section: "Never use this function. Use mkstemp(3) or tmpfile(3) instead." mkstemp() and tmpfile() avoid the inherent race condition of unique_path() by returning an open file descriptor or FILE *.	Remove this function. Consider providing a function create_unique_directory(). If it fits the scope of the proposed TS, consider providing functions create_unique_file() that returns ifstream, ofstream and iofstream.	

```
H:\cc51\Public\SC 22 Project\4900\CommentFiles\ISO_IEC PDTS 18822_ANSI.doc: Collation successful
H:\cc51\Public\SC 22 Project\4900\CommentFiles\ISO_IEC PDTS 18822_BSI.doc: Collation successful
H:\cc51\Public\SC 22 Project\4900\CommentFiles\ISO_IEC PDTS 18822_SFS.doc: Collation successful
H:\cc51\Public\SC 22 Project\4900\CommentFiles\ISO_IEC PDTS 18822_SNV.doc: Collation successful
Collation of  files was successful. Number of collated files : 4
SELECTED          (number of files):  4 .
FILES IN THIS GROUP(number of files):  4.
PASSED TEST       (number of files):  4.
FAILED TEST       (number of files):  0.
KEY:
        US - United States
        GB - Great Britain
        FI - Finland
        CH - Switzerland
```

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Result of voting

Ballot Information

Ballot reference	ISO/IEC PDTS 18822
Ballot type	DTS
Ballot title	Technical Specification -- File System
Opening date	2013-10-18
Closing date	2014-01-20
Note	

Member responses:

Votes cast (20)	Austria (ASI) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) Ireland (NSAI) Italy (UNI) Japan (JISC) Korea, Republic of (KATS) Netherlands (NEN) Portugal (IPQ) Romania (ASRO) Russian Federation (GOST R) Spain (AENOR) Switzerland (SNV) Ukraine (DTR) United Kingdom (BSI) United States (ANSI)
Comments submitted (0)	
Votes not cast (1)	Kazakhstan (KAZMEMST)

Questions:

Q.1	"Does your National Body approve the attached DTS to go forward to publication?"
Q.2	"If you approve the DTS Text with comments, would you please indicate which type ? (General, Technical or Editorial)"
Q.3	"If you Disappove the Draft, would you please indicate if you accept to change your vote to Approval if the reasons and appropriate changes will be accepted?"

Votes by members	Q.1	Q.2	Q.3

Austria (ASI)	Abstention	Ignore	Ignore
Canada (SCC)	Approval as presented	Ignore	Ignore
China (SAC)	Approval as presented	Ignore	Ignore
Denmark (DS)	Approval as presented	Ignore	Ignore
Finland (SFS)	Approval with comments	All	Ignore
France (AFNOR)	Abstention	Ignore	Ignore
Germany (DIN)	Abstention	Ignore	Ignore
Ireland (NSAI)	Approval as presented	Ignore	Ignore
Italy (UNI)	Approval as presented	Ignore	Ignore
Japan (JISC)	Approval as presented	Ignore	Ignore
Korea, Republic of (KATS)	Approval as presented	Ignore	Ignore
Netherlands (NEN)	Approval as presented	Ignore	Ignore
Portugal (IPQ)	Abstention	Ignore	Ignore
Romania (ASRO)	Abstention	Ignore	Ignore
Russian Federation (GOST R)	Approval as presented	Ignore	Ignore
Spain (AENOR)	Approval as presented	Ignore	Ignore
Switzerland (SNV)	Disapproval of the draft	Technical	Yes
Ukraine (DTR)	Approval as presented	Ignore	Ignore
United Kingdom (BSI)	Approval with comments	All	Ignore
United States (ANSI)	Approval with comments	All	Ignore

Answers to Q.1: "Does your National Body approve the attached DTS to go forward to publication?"

11 x	**Approval as presented**	**Canada (SCC)** **China (SAC)** **Denmark (DS)** **Ireland (NSAI)** **Italy (UNI)** **Japan (JISC)** **Korea, Republic of (KATS)** **Netherlands (NEN)** **Russian Federation (GOST R)** **Spain (AENOR)** **Ukraine (DTR)**

3 x	Approval with comments	Finland (SFS) United Kingdom (BSI) United States (ANSI)
1 x	Disapproval of the draft	Switzerland (SNV)
5 x	Abstention	Austria (ASI) France (AFNOR) Germany (DIN) Portugal (IPQ) Romania (ASRO)

Answers to Q.2: "If you approve the DTS Text with comments, would you please indicate which type ? (General, Technical or Editorial)"

0 x	General	
1 x	Technical	Switzerland (SNV)
0 x	Editorial	
3 x	All	Finland (SFS) United Kingdom (BSI) United States (ANSI)
16 x	Ignore	Austria (ASI) Canada (SCC) China (SAC) Denmark (DS) France (AFNOR) Germany (DIN) Ireland (NSAI) Italy (UNI) Japan (JISC) Korea, Republic of (KATS) Netherlands (NEN) Portugal (IPQ) Romania (ASRO) Russian Federation (GOST R) Spain (AENOR) Ukraine (DTR)

Answers to Q.3: "If you Disappove the Draft, would you please indicate if you accept to change your vote to Approval if the reasons and appropriate changes will be accepted?"

1 x	Yes	Switzerland (SNV)
0 x	No	
19 x	Ignore	Austria (ASI) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) Ireland (NSAI) Italy (UNI) Japan (JISC) Korea, Republic of (KATS)

	Netherlands (NEN) **Portugal (IPQ)** **Romania (ASRO)** **Russian Federation (GOST R)** **Spain (AENOR)** **Ukraine (DTR)** **United Kingdom (BSI)** **United States (ANSI)**

Comments from Voters

Member:	Comment:	Date:
Finland (SFS)	***Comment File***	2014-01-15 07:54:49
CommentFiles/ISO_IEC PDTS 18822_SFS.doc		
Switzerland (SNV)	***Comment File***	2014-01-15 09:31:14
CommentFiles/ISO_IEC PDTS 18822_SNV.doc		
United Kingdom (BSI)	***Comment File***	2014-01-16 11:45:43
CommentFiles/ISO_IEC PDTS 18822_BSI.doc		
United States (ANSI)	***Comment File***	2014-01-16 21:04:54
CommentFiles/ISO_IEC PDTS 18822_ANSI.doc		

Comments from Commenters

Member:	Comment:	Date: