#	Reference	Comment/Suggestions	YANG “leafref” nodes	802.1Q conform-ance	Bad 802.1Q reference	YANG data consistency
1	ieee802-dot1q-bridge.yang:284: list component	a) The given reference to 12.3 in 802.1Q-2018 is wrong. 12.3 describes the data types, it should be 12.4.1.5, which contains Table 12-1. b) The list itself and all leafs in this list are configuration data, but most should be state [802.1Q, 12.4.1.5].			X	
2	ieee802-dot1q-bridge.yang:295: leaf name	This leaf is child of the Bridge component node, but there is no such thing like a (unique) component name in 12.3 or 12.4.1.5 of 802.1Q-2018. If there is, please add a proper reference node pointing to the location in 802.1Q-2018 (leaf name has no “reference” node set). If not, delete the name leaf from node component, and let “id” become the key of the enclosing list (line 285).		X	X	
3	ieee802-dot1q-bridge.yang:341: leaf ports	The number of ports is implicitly given by leaf-list bridge-port in line 352. Suggest to delete leaf ports, or put a constraint via a “must” for data consistency.				X
4	ieee802-dot1q-types.yang:61: typedef port-number-type	The referenced item [12.3, item I)] says range 1..4095, the YANG definition says 1..65535. Correct to 1..4095.		X		
5	ieee802-dot1q-bridge.yang::258: leaf ports	a) should be of type “port-number-type” b) implicitly given by components:ports, though this is a level 2 indirection which may be ok (to be discussed, a constraint via “must” may be possible here too)				X
6	ieee802-dot1q-bridge.yang:278: leaf components	The leaf provides the number of Bridge components in a Bridge. This information is already available due to the existence of a list (list component in line 284). Remove leaf components, or make a constraint via “must”.				X
7	ieee802-dot1q-bridge.yang:1163: leaf component-name	Refers to associated component the enclosing bridge-port belongs to. Issues: a) A reference by name (string) is inefficient, and there is nothing like a component name (see prev. related issue) – use the component ID instead. b) Writing a description that this leaf is a reference is nothing machines can understand. YANG (RFC7950) provides “leaf-ref”, which allows to create a reference a machine (and compilers) can understand and verify. “leaf-ref” should be used, examples are found in IETF files (e.g., interface-ref type) and IEEE files (ieee802-dot1q-stream-filters-gates).	X			

8	ieee802-dot1q-tsn-types.yang:122: leaf interface-name	Proposed change: Use IETFs interface-ref type, avoid reference by name	X			
9	ieee802-dot1q-types.yang:630:leaf port-ref	Should point to (ieee802-dot1q-bridge.yang:1487:leaf port-number) via leaf-ref. Note that this may require refactoring (i.e., moving contents from one file to another) to avoid circular dependencies.	X			
10	ieee802-dot1q-bridge.yang:435: container filtering-database	Issues: a) Leafs static-entries, dynamic-entries, static-vlan-registration-entries, dynamic-vlan-registration-entries, and mac-address-registration-entries provide the number of respective entries currently in the FDB. This information is redundant because the associated list-nodes already provide the number of entries. b) The given container-node is child of node “list component” (line 284), where each entry is fro one out of multiple Bridge components of a Bridge. The management interface, though, is specified to provide the FDB as a whole for a Bridge (12.7.1, first and second sentence). Proposed change: On a): We may want to remove the leafs, or use “must” to implement associated constraints On b): Move container filtering-database to become child of node “list bridge” (line 226).		X		X
11	ieee802-dot1q-bridge.yang:660: container permanent-database	Issues: a) Contains the number of static filtering entries via child leaf “filtering-entries”, though the number of filtering entries is implicitly provided via child list “filtering-entry” (redundant information). b) Contains the number of static vlan registration entries via child leaf static-vlan-registration-entries, but does not provide an associated child list for static VLAN registration entries. c) The management of the FDB is presented as a whole for a Bridge [12.7.1]. Though this is not a strict requirement for the permanent DB (at least I could not find a sufficient explicit statement), it is recommendable to let the permanent DB also be one entity per Bridge, not per Bridge component. Otherwise the operation on FDB initialization would have to be modeled very explicit. Proposed Change: On a): We may want to remove leaf “filtering-entries”, or use “must” to implement a constraint On b): Add a child list for static VLAN registration entries (cmp. [8.8.2]), and delete leaf “static-vlan-registration-entries” to eliminate the redundancy. On c): Move container permanent-database to become child of node “list bridge” (line 226).		X		X

12	ieee802-dot1q-bridge.yang:524: leaf database-id	The leaf is child of “container filtering-database:435/list filtering-entry:519”. There is no such thing like an (uint32) database-id in 12.7.7 of IEEE 802.1Q-2018. The only thing I could find would be an enumeration with literals {“Permanent Database”, “FDB”} used as database identifier for General Database operations (cmp. 12.7.7.{1,2,3,4}.1), which would make the input identifier 12.7.7.{1,2,3,4}.2 a choice out of the two aforementioned literals. However, this choice is superfluous because the permanent database and the FDB are separate containers (lines 435 and 660). Proposed change: Remove leaf database-id in lines 524, 620, and 697.		X		
13	ieee802-dot1q-bridge.yang:721: leaf status	This node is inside of container permanent-database:660. The permanent database only contains static filtering entries [8.8.1 → MAC Addr. Spec.+VID Spec+Port Map] and static VLAN registration entries [8.8.2 → VID + Port Map], as stated in [8.8.11]. It is a) where the leaf comes from at all (no reference into 802.1Q-2018 given), and b) unclear why literal “learned” and potentially others are needed in the inline type definition of this leaf node. Proposed Change: Delete the leaf node, or add missing normative reference and remove enumeration literals for which no normative reference can be provided.		X	X	
14	ieee802-dot1q-bridge.yang:704: leaf address	This node is part of a filtering entry (line 692) in the permanent database. The permanent database only contains static filtering entries [8.8.1 → MAC Addr. Spec.+VID Spec. + Port Map].Type mac-address (pattern “[0-9a-fA-F]{2}(-[0-9a-fA-F]{2}){5}”) of the given node cannot capture the wildcard cases of static filtering entires [8.8.1, items a3) through a5)]. Proposed change: Make it a choice out of type mac-address and the wildcard cases.		X		
15	ieee802-dot1q-bridge.yang:540: leaf vids	The leaf is child a filtering entry (line 519) in the FDB (line 435). Due to leaf entry-type (line 548), the filtering entry node (line 519) models both: static and dynamic filtering entries. Though dynamic filtering entries [8.8.3] can be associated with multiple VIDs via the FID [8.8.3, item b)], the management interface is specified to be one out of these VIDs [12.7.7, 1st and 2nd sentence]. Contrary, the leaf is modeled as a set of VIDs (implies multiple) due to its type (vid-range-type) and its description. This type defines a string format to specify multiple ranges, which makes semantic interpretation nearly impossible for machines. Proposed changes: a) Change the type of leaf vids to the leaf name to “vid”, and make it an integer within 1..4095 (i.e., allow the wildcard value 4095 for static filtering entries [Table 9-2, 8.8.1]). b) Do similar for the “vids” leafs in lines 627 (VLAN registration entries, i.e. rename to “vid” and make it range 1..4094 without wildcard) and 713 (filtering entries in the permanent DB, i.e. rename to “vid” and make it range 1..4095 with wildcard).		X		

16	ieee802-dot1q-bridge.yang:982: leaf vids	Part of VID-FID allocations [12.10.3]. Upfront: The VID-FID YANG model reflects the information according 12.10.3. However, due to the type of leaf vids, the encoding is more compressed than the table described in 12.10.3 (one row per VID). Thus, the YANG encoding is considerable rougher for individual VID access operations [12.10.3, items c), e), f)].			X	
17	ieee802-dot1q-bridge.yang:982: l	There are multiple incarnations of FID-VID mappings in the file: - ieee802-dot1q-bridge.yang:982: list vid-to-fid-allocation - ieee802-dot1q-bridge.yang:1019: list fid-to-vid-allocation (backwards?) - ieee802-dot1q-bridge.yang:1061: list vid-to-fid I do not think all of these are needed, especially given that there are no data consistency constraints in place (unsure whether these are possible at all). IEEE Std 802.1Q-2018 models FID to VID mappings as a table according to [12.10.3, item a)]. It appears that the multiple (redundant) representations attempt to model the different access operations, as specified in [12.10.3, items b) through f), plus subclauses]. The way to model such operations in YANG would be to implement RPC operations in YANG. However, I believe we don't want to go there because YANGs RPC capabilities are not used at several other places in existing YANG files where RPC would be the proper way. I think we are rather modeling the underlying data model in a bridge, which would be a plain table. This said, we should discuss whether we want to simplify the YANG model to a single table, which would at least resolve the data integrity issues (i.e., there would be less redundant representations of the underlying table). In either case, the redundant representations should be addressed in some reasonable way.				X