**mvdXML Unit Test Cases**

**Overview**

The test case contains the following files:

1. IFC4 test files
   * mvdXML\_ifc4\_unit-test.ifc
   * mvdXML\_ifc4\_unit-test \_type.ifc
   * mvdXML\_ifc4\_unit-test\_not-empty-fireRating.ifc
2. Testdescription (describes the content of the IFC files)
   * Testdescription\_mvdXML\_ifc4\_unit-test.xlsx
   * Testdescription\_mvdXML\_ifc4\_unit-test\_type.xlsx
   * Testdescription\_mvdXML\_ifc4\_unit-test\_not-empty-fireRating.xlsx
3. mvdXML for testing various aspects (applicability, properties, attributes, classification, ..)
   * mvdXMLUnitTestsforIFC4.mvdxml  
     This file contains the following ExchangeRequirements (each ExchangeRequirement should be tested separately to check if a concept fails or passes)
     + Applicability Unit Tests : mvdXML Unit Tests
     + Classification Unit Tests : mvdXML Unit Tests
     + Direct Attributes Unit Tests : mvdXML Unit Tests
     + Property Unit Tests : mvdXML Unit Tests
     + Quantity Unit Tests : mvdXML Unit Tests
     + User Defined Property Unit Tests : mvdXML Unit Tests
     + Empty String test : mvdXML Unit Tests  
       *to be tested against mvdXML\_ifc4\_unit-test\_not-empty-fireRating.ifc*
4. Documentation of the usage settings (if a concept is MANDATORY, OPTIONAL or NOT ALLOWED)
   * UsageSettingsOf\_mvdXMLUnitTestsforIFC4.xlsx  
     (documentation for mvdXMLUnitTestsforIFC4.mvdxml)

**Test Principle**

For unit testing a single ExchangeRequirement should be tested agains a single IFC file. Each concept with a usage setting should then be tested against the IFC file. The following example should clarify the test.

***Example:***

|  |  |
| --- | --- |
| Selected IFC test file: | mvdXML\_ifc4\_unit-test.ifc |
| Selected ExchangeRequirement: | Applicability Unit Tests  Code: TS1-APL |

The screenshot in Figure 1 shows the definition of usage settings of concepts for the different ExchangeRequirements. For the given example, ExchangeRequirement TS1-APL, column B is relevant. It shows for instance that a Concept called “External flag” (row 8) is required (MAN). This concept is tested for “External loadbearing walls” (row 5).

The screenhost in Figure 2 shows how to decide whether a usage setting should pass or fail. Columns C to I describe the content of the IFC file. There are 7 instances (“Wand-001”, “Wand-002”, …). Each row indicates if a concept is available or not. For instance row 10 describes the selected concept “External flag” for External loadbearing walls. All cells in that row except for “Wand-002” are marked with red, which means that only this element fulfills the criteria defined for an external load bearing wall. Accordingly, the test should fire for “Wand-002” only. Cell 10-D shows that the property “External flag” is available in the IFC file (indicated by Yes). As this property is required for the selected ExchangeRequirement the test should pass.

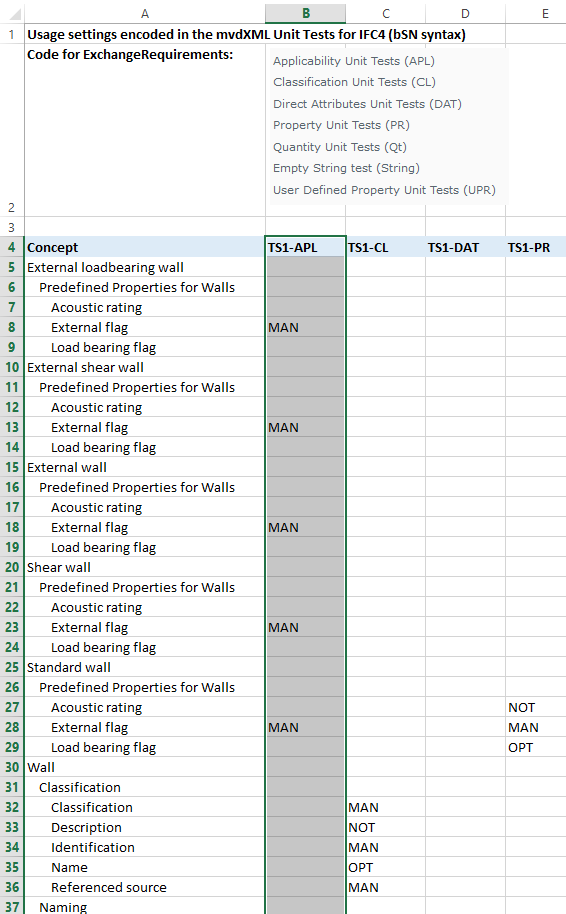


Figure 1: Matrix showing the usage definitions in UsageSettingsOf\_mvdXMLUnitTestsforIFC4.xlsx

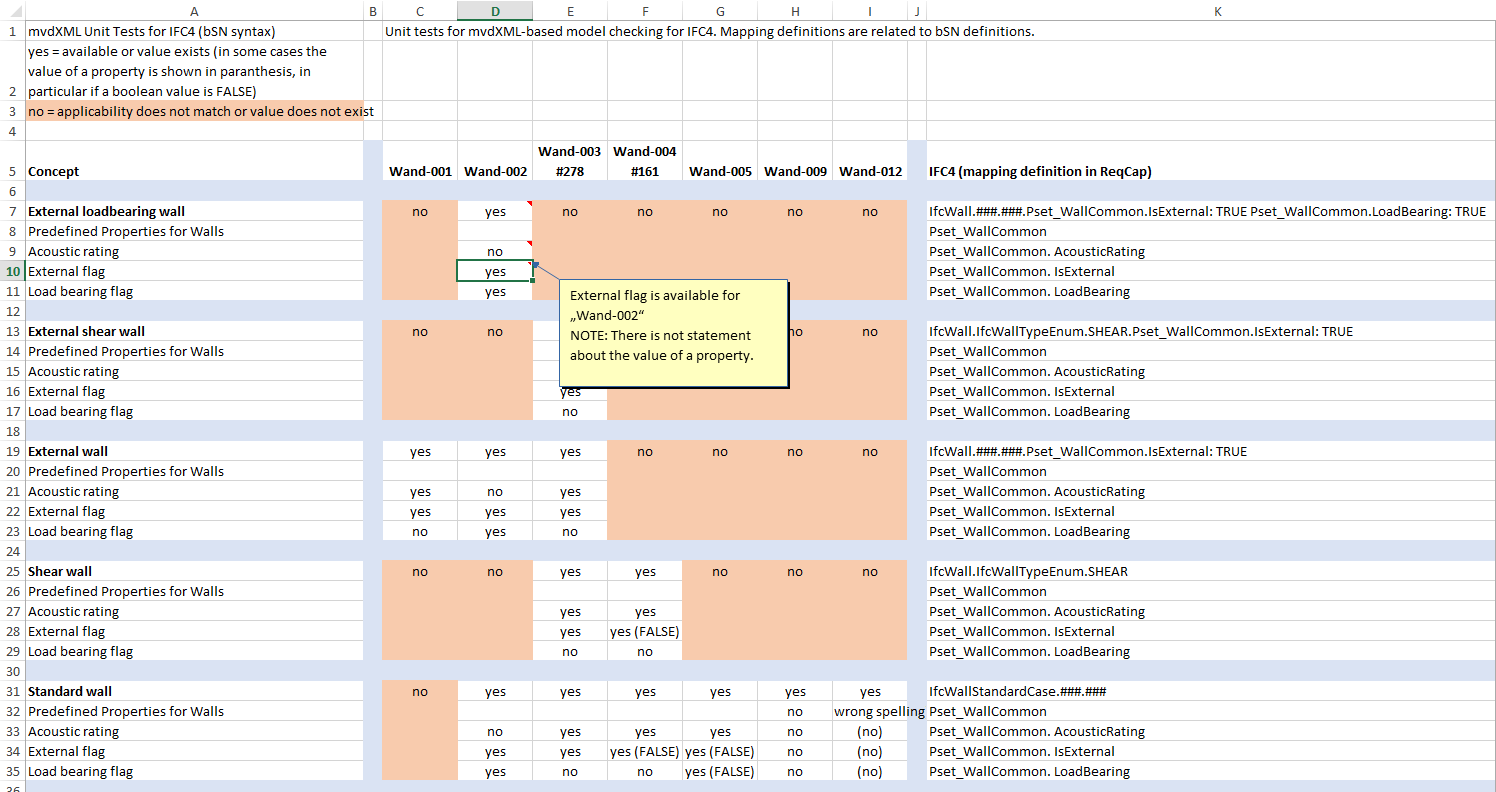


Figure 2: Content of the IFC test file to check if a test should pass or fail (shows Testdescription\_mvdXML\_ifc4\_unit-test.xlsx).

The next figure show the mvdXML snippet that defines the describe concept. The name attribute of ConceptRoot elements can be used to search for the name of a concept. Three parts are relevant in that definition:

* applicableRootEntity and the <Applicability> element (see Figure 4)  
  defines when a test should fire (all IfcWall instances that fullfil the applicability constraints)
* <TemplateRules> element   
  defines what should be tested (availability of the property Pset\_WallCommon. IsExternal)
* <Requirements> element (see Figure 3)  
  defines the usages for a set of ExchangeRequirements (MAN for the ExchangeRequirement with the ID 00000023-0000-0000-0000-000000000352)[[1]](#footnote-1)

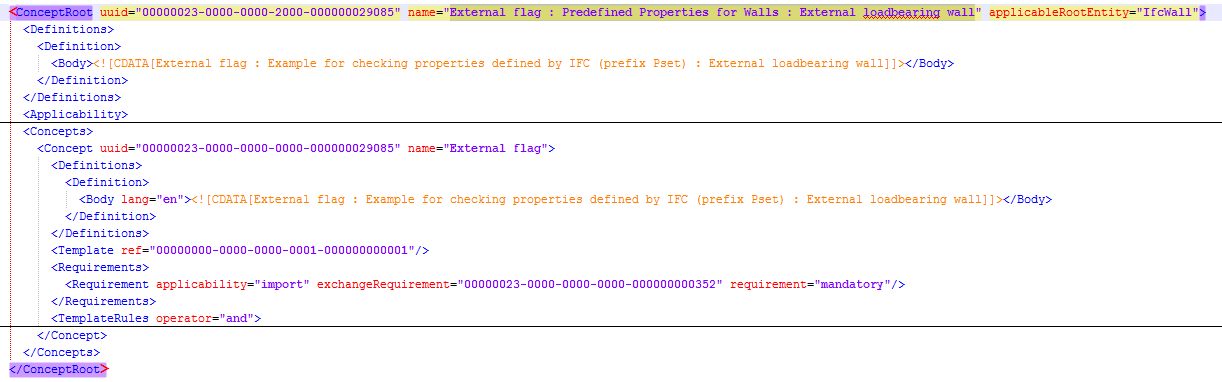


Figure 3: mvdXML snippet showing the definition of the “External flag” concept of “External loadbearing wall”.

**Applicability**

This part of ConceptRoot defines the condition under which the requirement test has to be applied to an IFC instance. The example shown in Figure 4 defines the applicableRootEntity to be IfcWall, which includes all subtypes of IfcWall. Additionally, the <Applicability> element specifies a constraint on properties, in this case to Pset\_WallCommon.IsExternal and Pset\_WallCommon.LoadBearing. Both properties must be available (either defined on occurrence or type instances) and must be TRUE.

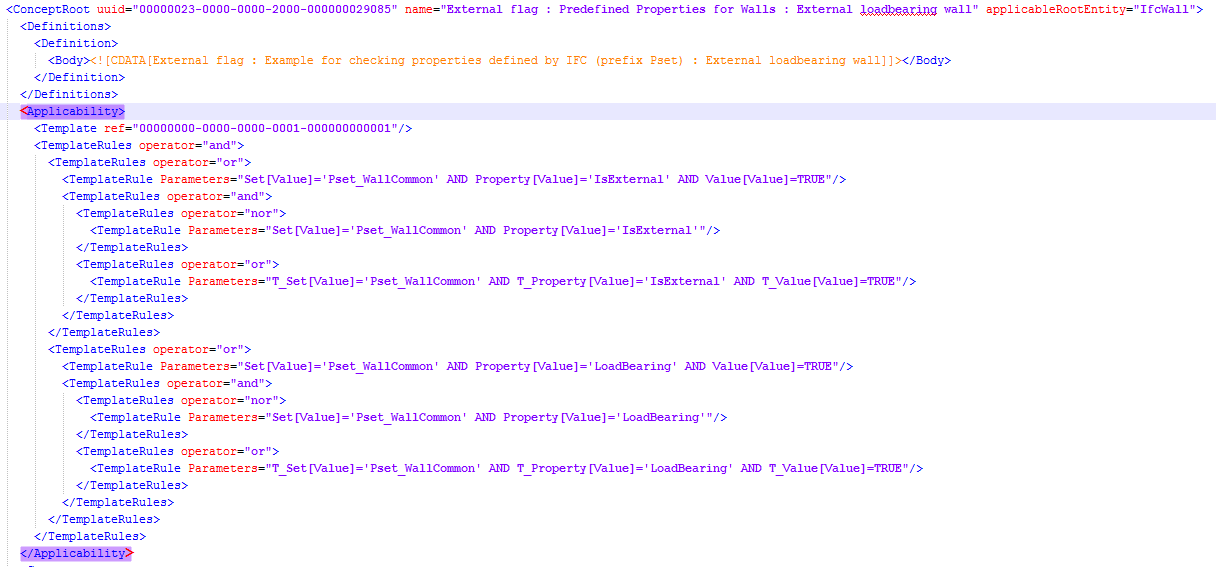


Figure 4: Applicability definition of a requirement.

**Requirements**

In a similar way as <Applicability> the requirements are defined in the <TemplatesRules> element. In this example, the existence of the property Pset\_WallCommon.IsExternal is checked. This example is very specific as it tests a property that is already part of the applicability test. This is not a typical situation. If the ExchangeRequirement is mandatory, then this test should always pass as it was already required in the applicability test (external walls are identified by this property). If the ExchangeRequirement is “excluded”, then this test should always fail.



Figure 5: Requirement definition of a requirement.

**Possible filter definitions for checking IFC files**

Input to xBIM:

* IFC file
* mvdXML file (an mvdXML file can have one or more ExchangeRequirements)

Test:

* each ConceptRoot must be checked for each ExchangeRequirement

Output:

* all passes/fails   
  for each IFC element and each ExchangeRequirement  
  exported as BCF

Usefull filter definitions:

* show only passes or fails   
  available (traffic light solution), may extend to support the five scales of mvdXML)
* show checking results for selected ExchangeRequirements

available, may support selection of 1 or more ExchangeRequirements

* show checking results for selected IFC entity types (but end user might not be familiar with IFC entities names)   
  available
* show checking results for selected requirements  
  ideally supported by providing a tree structure according to the requirement definitions  
  object -> property group -> property

Filter definitions should be combinable to enable more flexible selections.

Example for selecting requirements:

|  |  |  |
| --- | --- | --- |
| **Row in usage table** | **Concept tree** | **Filter for checking** |
| 5 | External loadbearing wall | If this element is selected, show checking results for all ConceptRoot that fulfill applicability test of “External loadbearing wall”  -> Test “External flag” as it is the only ConceptRoot in that example |
| 6 | Predefined Properties for Walls | Test all child node requirements. -> Test “External flag” |
| 8 | External flag | Test this requirement only -> Test “External flag” |
| 25 | Standard wall | If selected test for all “Standard walls”:   * “Acoustic rating”  (for ExchangeReq. TS1-PR) * “External flag”  (for TS1-APL and TS1-PR) * “Load bearing flag”  (for TS1-PR) |
| 26 | Predefined Properties for Walls |  |
| 27 | Acoustic rating |  |
| 28 | External flag |  |
| 29 | Load bearing flag |  |

The tree structure can/should show only elements that are defined in the mvdXML document.

ExchangeRequirement tree

* ExchangeRequirement
  + [Grouped by roots]
    - Concepts that point to the ER.

1. MAN maps to requirement=“mandatory“, OPT maps to requirement=“not-relevant“ and NOT maps to requirement=”excluded”  
   Not tested in the unit test cases are the mvdXML requirement settings “recommended” and “not-recommended”. [↑](#footnote-ref-1)