# Greenfox – outline

## Getting started with greenfox

Consider **system S** – an imaginary system which is a collection of web services. We are going to validate a *file system representation* which is essentially a set of test results, accompanied by resources supporting validation (XSDs, codelists and data about expected response messages). The following listing shows a file system tree which is an example representation of system S, as observed at a certain point in time:

**system-s**

. resources

. . **codelists**

. . . *codelist-foo-article.xml*

. . **xsd**

. . . *schema-foo-article.xsd*

. testcases

. . **test-t1**

. . . config

. . . . *msg-config.xml*

. . . input

. . . . *getFooRQ\*.xml*

. . . output

. . . . *getFooRS\*.xml*

. . **+test-t2 (contents: see test-t1)**

. . usecases

. . . usecase-u1

. . . . usecase-u1a

. . . . . **+test-t3 (contents: see test-t1)**

The concrete file system tree must be distinguished from the *expected file system tree*, which is described by the following rules.

|  |  |  |
| --- | --- | --- |
| **File or folder** | **Name or pattern** | **Expectation** |
| folder | codelists | Contains one or more codelist files |
| folder | codelists/\* | A codelist file; may have any name; must be an XML document containing <codelist> elements with a |name attribute and <entry> children |
| folder | xsd | Contains one or more XSDs describing services messages |
| file | xsd/\* | An XSD schema file; may have any name |
| folder | test-\* | A testcase folder, containing input, output and config folders |
| folder | config | A test case config folder, containing file msg.config.csv |
| file | msg.config.csv | A CSV file (separator: comma; with headline; three columns with request file name, response file name, expected return code |
| folder | input | A test case input folder, containg files representing request messages |
| file | input/\* | A file representing a request message; may have any name, but must have name extension .xml or .json; mediatype must correspond to name extension |
| file | output/\* | A file representing a response message; may have any name, but must have name extension .xml or .json; mediatype must correspond to name extension |

It should be noted that the number and location of testcase folders (test-\*) are unconstrained. This means that the testcase folders may be grouped and wrapped in any way, althoug they must not be nested, as a testcase folder has exactly three members: folders input, output and config with contents as described by the table. So the use of a testcases folder wrapping all testcase folders - and the use of usecase-\* folders adding additional substructure - is possible, but must not be expected. The placing of XSDs in folder resources/xsd, on the other hand, is obligatory, and likewise the placing of codelist documents in folder resources/codelists. The names of XSD and codelist file, on the other hand, are not constrained.

Besides the structural expectations, there are also content-related expectations:

* For every response message in XML format, there must be exactly one XSD against which the message can be validated successfully
* In response messages (XML or JSON format) with root element <getFooRS>, the values in elements //fooValue must be found in the codelist with name foo-article.
* In response messages (XML or JSON format), the return code must be as configured by the corresponding row in msg-config.csv.
* In response messages, a <returnCode> with value “OK” implies the occurrence of at least one <fooValue> elements, which must have non-empty text content.