30-Nov-2020

Piotr P. Nikiel

# On original_files.txt and files.txt and finding of new format for their storage

## Introduction

quasar is version-control-system agnostic - i.e. it should do equally well no matter which VCS you use, if any at all.

To support convenient work with any VCS, since about the beginning (~2014, see OPCUA-131, OPCUA-160 or related tickets) it has a built-in tool that handles:

- creating new quasar projects,
- upgrading existing quasar projects to newer quasar versions,
- checking consistency of existing projects (including co-operating with any supported[1] VCS if they are used),
- managing "quasar releases" (this is for quasar release managers only).

This part of quasar mostly lies within manage_files.py, inside FrameworkInternals and is backed by two files in the same directory:

- original_files.txt - a human-managed data source on which files quasar is composed of, and how they should be used on project creation / project upgrade / file depreciation events.

[1] Being SVN or Git at the moment of writing.

- files.txt - a direct derivative of original_files.txt, supplemented by checksums for ensuring integrity of files that should not be modified by users.

# Current format of (original_)files.txt as of quasar 1.4.2 / nebula.B1

The idea is quite simple:

- Every non-empty, non-comment line is a line that describes a directory within root of quasar project or a file within the preceding directory
- For directories, lines read as:
  Directory <NAME> comma-separated-options…
- For files, lines read as:
  File <NAME> comma-separated-options…
- There might be no options in any case, i.e. there can be just Directory or File literal and the name.
- Options are keyword only, or keyword=value
- Recognized options for directories:
  - install, what to do when installer runs, value in [create]
- Recognized options for files:
  - install, what to do when installer runs, value in [overwrite, ask_to_merge, copy_if_not_existing]
  - must_be_versioned - the file should be under user's VCS for the project to be well-maintained
  - must_be_md5_checked - the file should be controlled for (accidental) user changes
  - must_exist - the file is crucial to quasar project, its disappearance must be reported.
  - md5 - file's md5, in format equal to produced by common "md5sum" tool.
  - deprecated - file should be wiped because it is no longer needed by quasar (good example: former XSLT transforms).

Example:

```
Directory AddressSpace/templates install=create
File designToGeneratedCmakeAddressSpace.jinja  must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassHeader.jinja              must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassBody.jinja                must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelHeader.jinja   must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelBody.jinja     must_exist,must_be_versioned,md5=check,install=overwrite
File designToAddressSpaceDocHtml.jinja      must_exist,must_be_versioned,md5=check,install=overwrite
File designToSourceVariablesHeader.jinja    must_exist,must_be_versioned,md5=check,install=overwrite
File designToSourceVariablesBody.jinja      must_exist,must_be_versioned,md5=check,install=overwrite

Directory bin install=create
```

```
File ServerConfig.xml                       must_exist,must_be_versioned,install=copy_if_not_existing
File config.xml                             install=copy_if_not_existing

Directory CalculatedVariables install=create
File CMakeLists.txt                         must_exist,must_be_versioned,md5=check,install=overwrite

Directory CalculatedVariables/ext_components install=create

Directory CalculatedVariables/ext_components/muparser-amalgamated install=create

Directory CalculatedVariables/ext_components/muparser-amalgamated/include install=create
File muParser.h                             must_exist,must_be_versioned,md5=check,install=overwrite
```

# Inconveniences of the current format

- the parser is home-brew (-> maintenance burden for people taking over)
- the parser does not support e.g. defaults per directory (they would significantly simplify/reduce both files). This feature was done but the parser become more complex to maintain. (-> maintenance burden for people taking over)
- in 2020 maybe it is not necessary to invent custom file formats anymore (we have XML, JSON, YAML, TOM, …)

# Proposal 1: go to XML

The comparison between the current format and XML, and with support for default per-directory options, would be as below:

**Current format**

```
Directory AddressSpace/templates install=create
File designToGeneratedCmakeAddressSpace.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassHeader.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassBody.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelHeader.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelBody.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToAddressSpaceDocHtml.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToSourceVariablesHeader.jinja
```

**XML proposal**

```
<Directory name="AddressSpace/templates" install="create">
  <FileDefaults must_exist="true" must_be_versioned="true" md5="check"
install="overwrite"/>
  <File name="designToGeneratedCmakeAddressSpace.jinja" apply_defaults="true" />
  <File name="designToClassHeader.jinja"              apply_defaults="true" />
  <File name="designToClassBody.jinja"                apply_defaults="true" />
  <File name="designToInformationModelHeader.jinja"   apply_defaults="true" />
  <File name="designToInformationModelBody.jinja"     apply_defaults="true" />
  <File name="designToAddressSpaceDocHtml.jinja"      apply_defaults="true" />
  <File name="designToSourceVariablesHeader.jinja"    apply_defaults="true" />
  <File name="designToSourceVariablesBody.jinja"      apply_defaults="true" />
</Directory>

<Directory name="bin" install="create">
```

| | |
|---|---|
| must_exist,must_be_versioned,md5=check,install=overwrite<br>File designToSourceVariablesBody.jinja<br>must_exist,must_be_versioned,md5=check,install=overwrite<br><br>Directory bin install=create<br>File ServerConfig.xml<br>must_exist,must_be_versioned,install=copy_if_not_existing<br>File config.xml<br>install=copy_if_not_existing<br><br>Directory CalculatedVariables install=create<br>File CMakeLists.txt<br>must_exist,must_be_versioned,md5=check,install=overwrite<br><br>Directory CalculatedVariables/ext_components install=create<br><br>Directory CalculatedVariables/ext_components/muparser-amalgamated<br>install=create<br><br>Directory<br>CalculatedVariables/ext_components/muparser-amalgamated/include<br>install=create<br>File muParser.h<br>must_exist,must_be_versioned,md5=check,install=overwrite | <File name="ServerConfig.xml" must_exist="true" must_be_versioned="true"<br>install="copy_if_not_existing"/><br><File name="config.xml" install="copy_if_not_existing" /><br></Directory><br><br><Directory name="CalculatedVariables" install="create"><br><File name="CMakeLists.txt" must_exist="true" must_be_versioned="true" md5="check"<br>install="overwrite"/><br></Directory><br><br><Directory name="CalculatedVariables/ext_components" install="create"/><br><br><Directory name="CalculatedVariables/ext_components/muparser-amalgamated"<br>install="create" /><br><br><Directory name="CalculatedVariables/ext_components/muparser-amalgamated/include"<br>install="create"><br><File name="muParser.h" must_exist="true" must_be_versioned="true" md5="check"<br>install="overwrite"/><br></Directory> |

The features of XML option are:

- reuse of existing tools and libs (no need for custom parser anymore) - lxml *is not* in Python's stdlib, but has been required for ages for other important features of quasar, so it's like standard.
- extensibility for future,
- it's schema-aware

The look&feel of XML option is a matter of taste.

# Proposal 2: go to JSON

JSON is another format which is well represented in Python's standard library, thus makes a good candidate for the move.

| **Current format** | **JSON proposal** |
|---|---|

```
Directory AddressSpace/templates install=create
File designToGeneratedCmakeAddressSpace.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassHeader.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToClassBody.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelHeader.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToInformationModelBody.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToAddressSpaceDocHtml.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToSourceVariablesHeader.jinja
must_exist,must_be_versioned,md5=check,install=overwrite
File designToSourceVariablesBody.jinja
must_exist,must_be_versioned,md5=check,install=overwrite

Directory bin install=create
File ServerConfig.xml
must_exist,must_be_versioned,install=copy_if_not_existing
File config.xml
install=copy_if_not_existing

Directory CalculatedVariables install=create
File CMakeLists.txt
must_exist,must_be_versioned,md5=check,install=overwrite

Directory CalculatedVariables/ext_components install=create

Directory CalculatedVariables/ext_components/muparser-amalgamated
install=create

Directory
CalculatedVariables/ext_components/muparser-amalgamated/include
install=create
File muParser.h
must_exist,must_be_versioned,md5=check,install=overwrite
```

```
{
  "AddressSpace/templates": {
         "install": "create",
         "file_defaults": {
         "must_exist": true,
         "must_be_versioned": true,
         "md5": "check",
         "install": "overwrite"
         },
         "files": {
         "designToGeneratedCmakeAddressSpace.jinja": {
         "apply_defaults": "true"
         },
         "designToClassHeader.jinja": {
         "apply_defaults": "true"
         },
         "designToClassBody.jinja": {
         "apply_defaults": "true"
         },
         "designToInformationModelHeader.jinja": {
         "apply_defaults": "true"
         },
         "designToInformationModelBody.jinja": {
         "apply_defaults": "true"
         },
         "designToAddressSpaceDocHtml.jinja": {
         "apply_defaults": "true"
         },
         "designToSourceVariablesHeader.jinja": {
         "apply_defaults": "true"
         },
         "designToSourceVariablesBody.jinja": {
         "apply_defaults": "true"
         }
         }
  },
  "bin": {
         "install": "create",
         "files": {
         "ServerConfig.xml": {
         "must_exist": true,
         "must_be_versioned": true,
         "install": "copy_if_not_existing"
         }
         }
  }
}
(... example incomplete! )
```

The features of JSON option are:

- reuse of existing tools and libs (no need for custom parser anymore) - json is in Python's stdlib,
- it's not naturally schema-aware.

The look&feel of JSON option is a matter of taste.

## Proposal 3: don't change the format and keep the home-brew parser

The look&feel of this option is a matter of taste.