QDL Initialization (ini) files

QDL support a plain text file format. This allows an application to read a configuration information in a structured way. In essence, this is just an ASCII/text way to write a single stem. Why have them? So that there is a simple, consistent, text-based way to send along configuration and initialization information to an application. No special knowledge of QDL or programming is needed to write an ini file. Also, ini files solve the “bootstrapping problem” of how to start a complex system. The easiest way is to have external information which has to exist out side of the system and should be in an easily editable format.

File syntax

```
[section(.x)*]
line: name =|:= entry (,entry)*

entry: boolean | integer | decimal | scientific number | ‘string’;

comment: // .* | /* .* */
```

the section and name are identifiers – standard syntax for variable name in QDL applies. If the section identifiers have embedded periods, they will be turned into stem entries as well. A line may have multiple entries separated by commas which will be turned into a list. You may have blank lines.

Unlike QDL, each entry is restricted to a single line and does not end with a ; (semi-colon). If you end a line with a semi-colon, it is ignored.

Extra commas are ignored.

Multiple entries on a line are turned in to a list.

A comment is either from a // (double slash) to the end of the line or anything between /* . . . */, which may span multiple lines. You may have these anywhere and in any number. They will be discarded in parsing.

Reading an ini file

You access these in the workspace by issuing a **`file_read`** with the type of 2. The result is a stem of the form

```
stem.section.name0 := line entry
stem.section.name1 := line entry
```

Example

Let’s say we have an ini file **`/tmp/sample.ini`**:

```
// last modified on 1 April 2021 by Robespierre Braithwate.
[owner]
name='Fiona Smythe'
```

```
organization := 'Big State University/Physics','Big State University/Astronomy'

[database]
# use IP address in case network name resolution is not working
server= '192.168.1.42'
port=1029
```

The resulting stem then

```
  ini. := file_read('/tmp/sample.ini',2);
  ini.

{
 owner: {
  organization:  ['Big State University/Physics','Big State University/Astronomy'],
  name:'Fiona Smythe'
 },
 database: {
  server:'192.168.1.42',
  port:1029
 }
}
```

So if you want to access the information for the owner name, you issue

```
  ini.owner.name
Fiona Smythe
```

Writing ini files from stems

You can simply save a stem as an ini file by using

```
file_write(file_name, contents., 2)
```

where the final argument, 2, tells the system to convert `contents.` Do note that ini files are substantially simpler than a general stem. Values may be scalars or simple lists of scalars and attempts to write something more complex will throw an exception (because there is no good way for a user to enter such a thing without effectively having it all be just QDL).

Example

If you have the following stem (line breaks added)

```
    my_ini. :=
   {'owner':
     {'organization':
        ['Big State University/Physics','Big State University/Astronomy'],
      'name':'Fiona Smythe'
     },
    'database':{'server':'192.168.1.42', 'port':1029}
   };
```

Then you can save it to an ini file, /tmp/my.ini by issuing

```
    file_write('/tmp/my.ini', my_ini., 2);
true
```

When saving stems as ini files, there is no way to put in comments.

Example with section stem names

In this example, a more complex topology of the stem is made. The section names have embedded periods, which means the section is turned into its own stem.

The ini file

```
// test file for QDL ini format.
 [qwe ]
 q:='test'

a := .456, -47,,, true
// foo
b = 'p',,,'q', 345.66, -3.13E17

[blarf]

z := -234,65.34,'bar'

[woof.foo]
 q := 42
```

```
becomes the stem
{blarf : {z : [-234,65.34,bar]},
   qwe : {a : [0.456,-47,true],
          b : [p,q,345.66,-3.13E17],
          q : test},
  woof : {foo : {q:42}}}
```

This has its uses, but should not be overused since it reduces readability.