



ICM Open Data Import / Export Centre Ruby Scripts



Ruby scripting can be used to make the import / export of data via the open data import centre more streamlined for users of the software, by using Ruby scripts from the UI in conjunction with pre-prepared configuration files, and the Ruby scripting's UI elements.

At its simplest, if you can hard-code the paths of all files, then this can be done with 2 lines of code.

For import:

```
net=WSApplication.current_network
net.odic_import_ex('CSV','d:\temp\odic.cfg',nil,'Node','d:\temp\goat.csv','Pipe','d:\temp\stoat.csv')
```

For export:

```
net=WSApplication.current_network
net.odex_export_ex('CSV','d:\temp\odxc.cfg',nil,'Node','d:\temp\goat2.csv','Pipe','d:\temp\stoat2.csv')
```

As described above, both methods take a variable number of parameters. If you are importing a large number of files you may find it less unwieldy to call the method multiple times importing one file at a time e.g.

For import:

```
net=WSApplication.current_network
import=[[ 'Node', 'goat'], ['Pipe', 'stoat']]
import.each do |f|
  net.odic_import_ex('CSV','d:\temp\odic.cfg',nil,f[0],'d:\temp\\'+f[1]+' .csv')
end
```

For export:

```
net=WSApplication.current_network
export=[[ 'Node', 'goat'], ['Pipe', 'stoat']]
export.each do |f|
  net.odex_export_ex('CSV','d:\temp\odxc.cfg',nil,f[0],'d:\temp\\'+f[1]+'2.csv')
end
```

It should be noted that you will not see any of the error messages on import that would appear in the text box. Exceptions are not thrown for that sort of error, only for more serious errors in the processing. Also by using nil in the 3rd parameter of each method, default behaviour will be used for the options set on the dialog, this may not be what you want.

The first of these issues can be solved by specifying an error text file e.g.



```
net=WSApplication.current_network
import=[['Node','goat'],['Pipe','stoat']]
import.each do |f|
  params=Hash.new
  params['Error File']='d:\\temp\\errs'+f[0]+'.txt'
  net.odic_import_ex('CSV','d:\\temp\\odic.cfg',params,f[0],'d:\\temp\\'+f[1]+'.csv')
end
```


The aim is to produce one file per table. The files will be created but of zero bytes long if there are no errors for that table.

You will probably want to communicate the errors to the user. In its simplest form this could be done by checking the size of the files and displaying a message box at the end of the process e.g.

```
require 'FileUtils'
net=WSApplication.current_network
import=[['Node','goatwitherrs'],['Pipe','stoat']]
errFiles=Array.new
import.each do |f|
  params=Hash.new
  errFile='d:\\temp\\errs'+f[0]+'.txt'
  params['Error File']=errFile
  net.odic_import_ex('CSV','d:\\temp\\odic.cfg',params,f[0],'d:\\temp\\'+f[1]+'.csv')
  if File.size(errFile)>0
    errFiles << errFile
  else
    FileUtils.rm errFile
  end
end
if errFiles.size>0
  msg="Errors occurred - please consult the following files:"
  errFiles.each do |f|
    msg+="\r\n"
    msg+=f
  end
  WSApplication.message_box msg,nil,nil,nil
end
```

Note the inclusion of FileUtils and the use of the FileUtils.rm method to delete files of zero length. This will display a message reporting to the user the error files which should be consulted.

If you wish to show the user the actual messages then this can be achieved either by reading the files and outputting them to the standard output e.g.




```
require 'FileUtils'
net=WSApplication.current_network
import=[['Node','goatwitherrs','nodes'],['Pipe','stoat','pipes']]
errInfo=Array.new
import.each do |f|
  params=Hash.new
  errFile='d:\\temp\\errs'+f[0]+'.txt'
  if File.exists? errFile
    FileUtils.rm errFile
  end
  params['Error File']=errFile
  net.odic_import_ex('CSV','d:\\temp\\odic.cfg',params,f[0],'d:\\temp\\'+f[1]+'.csv')
  if File.size(errFile)>0
    temp=Array.new
    temp << errFile
    temp << f[2]
    errInfo << temp
  else
    FileUtils.rm errFile
  end
end
if errInfo.size>0
  puts "Errors importing data:"
  errInfo.each do |ei|
    puts "Errors for #{ei[1]}:"
    outputString=''
    File.open ei[0] do |f|
      f.each_line do |l|
        l.chomp!
        outputString+=l
        outputString+="\r"
      end
    end
    puts outputString
  end
end
```

Or by using the open_text_view method, in which case the block beginning with if ErrInfo.size>0 would be replaced with the following:

```

if errInfo.size>0
  consolidatedErrFileName='d:\\temp\\allerrs.txt'
  if File.exists? consolidatedErrFileName
    FileUtils.rm consolidatedErrFileName
  end
  consolidatedFile=File.open consolidatedErrFileName,'w'
  errInfo.each do |ei|
    consolidatedFile.puts "Errors for #{ei[1]}:"
    File.open ei[0] do |f|
      f.each_line do |l|
        l.chomp!
        consolidatedFile.puts l
      end
    end
  end
  consolidatedFile.close
  WSAApplication.open_text_view 'Open Data Import Centre Errors',consolidatedErrFileName,false
end

```



You may wish to not hard code the path of the config file but to store it with the Ruby script. This may be done by obtaining the path of the folder containing the script then adding the configuration file name onto the name e.g.

```
configfile=File.dirname(WSApplication.script_file)+'\\odicwithsource.cfg'
```

This works via the following 3 steps:

- Get the file name of the script file e.g. d:\temp\myscript.rb
- Use the `File.dirname` method to obtain the folder name e.g. d:\temp
- Add the configuration file name e.g. d:\temp\odicwitsource.cfg

Alternatively you may wish to allow the user to choose a config file using the `WSApplication.file_dialog` method e.g. by beginning the script with

```

net=WSApplication.current_network
configfile=WSApplication.file_dialog(true,'cfg','Open Data Import Centre Config File',nil,false,
false)
if configfile.nil?
  WSAApplication.message_box 'No config file selected - no import will be performed',nil,nil,false
  exit
end

```

Similarly you may wish to allow the user to choose the location of the data files or database tables etc. This may be done in numerous ways depending on the data type and/or how things are structured.

Allowing the user to select a folder and then using hard-coded names based on that folder:

```

require 'FileUtils'
net=WSApplication.current_network
configfile=WSApplication.file_dialog(true,'cfg','Open Data Import Centre Config File',nil,false,
false)
if configfile.nil?
  WSApplication.message_box 'No config file selected - no import will be performed',nil,nil,false
else
  folder=WSApplication.folder_dialog 'Select a folder containing the files to import',false
  if folder.nil?
    WSApplication.message_box 'No folder selected - no import will be performed'
  else
    import=[['Node','goatwitherrs','nodes'],['Pipe','stoat','pipes']]
    errInfo=Array.new
    import.each do |f|
      params=Hash.new
      errFile=folder+'\\errs'+f[0]+' .txt'
      if File.exists? errFile
        FileUtils.rm errFile
      end
      params['Error File']=errFile
      net.odic_import_ex('CSV',configfile,params,f[0],folder+'\\'+f[1]+' .csv')
      if File.size(errFile)>0
        temp=Array.new
        temp << errFile
        temp << f[2]
        errInfo << temp
      else
        FileUtils.rm errFile
      end
    end
    if errInfo.size>0
      puts "Errors importing data:"
      errInfo.each do |ei|
        puts "Errors for #{ei[1]}:"
        outputString=''
        File.open ei[0] do |f|
          f.each_line do |l|
            l.chomp!
            outputString+=l
            outputString+="\r"
          end
        end
        puts outputString
      end
    end
  end
end
end
end

```

Allowing the user to choose one file and then selecting similarly named files in the same folder (e.g. if we are expecting a file with the suffix 'stoat' and we find a file called 'northwest_stoat' we will also look for files called 'northwest_goat' etc.):



```

require 'FileUtils'
net=WSApplication.current_network
configfile=configfile=File.dirname(WSApplication.script_file)+'\\odicwithsource.cfg'
import=[['Node','goat','nodes'],['Pipe','stoat','pipes']]
file=WSApplication.file_dialog(true,'csv','CSV File',nil,false,false)
if file.nil?
  WSApplication.message_box 'No file selected - no import will be performed','OK',nil,false
elsif file[-4..-1].downcase!='.csv'
  WSApplication.message_box 'Not a csv file - no import will be performed','OK',nil,false
else
  folder=File.dirname(file)
  name=File.basename(file)[0..-5]
  prefix=''
  found=false
  import.each do |i|
    if name.downcase[-i[1].length..-1]==i[1].downcase
      prefixlen=name.length-i[1].length
      if prefixlen>0
        prefix=name[0..prefixlen-1]
      end
      found=true
      break
    end
  end
  if !found
    WSApplication.message_box 'File name does not have an expected suffix - no import will be performed','OK',nil,false
  else
    # errInfo is an array of arrays, with one entry added for each imported CSV file with some sort of issue
    # it will either contain the error file name and a name to be used for the table in error messages
    # or nil and a filename for any expected files which are missing
    errInfo=Array.new
    import.each do |f|
      csvfilename=folder+'\\'+prefix+f[1]+' .csv'
      if !File.exists? csvfilename
        temp=Array.new
        temp << nil
        temp << csvfilename
        errInfo << temp
      else
        params=Hash.new
        errFile=folder+'\\errs'+f[0]+' .txt'
        if File.exists? errFile
          FileUtils.rm errFile
        end
        params['Error File']=errFile
      end
    end
  end
end

```



```
net.odic_import_ex('CSV',configfile,params,f[0],csvfilename)
if File.size(errFile)>0
  temp=Array.new
  temp << errFile
  temp << f[2]
  errInfo << temp
else
  FileUtils.rm errFile
end
end
end
if errInfo.size>0
  puts "Errors importing data:"
  errInfo.each do |ei|
    if ei[0].nil?
      puts "Expected file #{ei[1]} not found"
    else
      puts "Errors for #{ei[1]}:"
      outputString=''
      File.open ei[0] do |f|
        f.each_line do |l|
          l.chomp!
          outputString+=l
          outputString+="\r"
        end
      end
      puts outputString
    end
  end
end
end
end
```



Allowing the user to select multiple files and choosing the data type to import based on the file names:


```

require 'FileUtils'
net=WSApplication.current_network
configfile=configfile=File.dirname(WSApplication.script_file)+'\\odicwithsource.cfg'
import=[['Node','goat','nodes'],['Pipe','stoat','pipes']]
files=WSApplication.file_dialog(true,'csv','CSV File',nil,true,false)
if files.nil? || files.length==0
  WSApplication.message_box 'No file selected - no import will be performed','OK',nil,false
else
  nErrs=0
  errInfo=Array.new
  files.each do |file|
    folder=File.dirname(file)
    name=File.basename(file)
    if name[-4..-1].downcase=='.csv'
      name=name[0..-5]
      import.each do |i|
        if i[1].downcase==name.downcase[-i[1].length..-1]
          params=Hash.new
          nErrs+=1
          errFile=folder+'\\errs'+nErrs.to_s+'.txt'
          if File.exists? errFile
            FileUtils.rm errFile
          end
          params['Error File']=errFile
          net.odic_import_ex('CSV',configfile,params,i[0],file)
          if File.size(errFile)>0
            temp=Array.new
            temp << errFile
            temp << i[2]
            errInfo << temp
          else
            FileUtils.rm errFile
          end
          break
        end
      end
    end
  end
end
if errInfo.size>0
  puts "Errors importing data:"
  errInfo.each do |ei|
    if ei[0].nil?
      puts "Expected file #{ei[1]} not found"
    else
      puts "Errors for #{ei[1]}:"
      outputString=''
      File.open ei[0] do |f|
        f.each_line do |l|
          l.chomp!

```



```
        outputString+=1
        outputString+="\r"
      end
    end
    puts outputString
  end
end
end
end
```



Except where otherwise noted, this work is licensed under a Creative Commons Attribution-nc-sa/3.0/ NonCommercial-ShareAlike 3.0 Unported License (<https://creativecommons.org/licenses/by-nc-sa/3.0/>). Please see the Autodesk Creative Commons FAQ (<https://autodesk.com/creativecommons>) for more information.

© 2025 Autodesk Inc. All rights reserved