PAR5E

User Guide

Contents

Introduction

PAR5E is a software Parser tool for creating modules and bulk loading data for FantasyGrounds v3.0 or higher rulesets.

The tool (whilst not sharing any source code) has been heavily inspired by the original 4EParser tool by Tenian. It works on a similar basis. You copy and paste text from roleplaying game system rulebooks and modules into text files, mark up the text where necessary and then point PAR5E at them to parse and produce a shiny new module. However, whilst the 4EParser remains forever a valued tool for the D&D 4E FG community, it offers no value to any other FG ruleset user, in addition, it has been some while since it has been actively developed and supported.

In fairness 2013 introduced several other similar 'parser' tools for other rulesets however each addresses the approach differently and again only support specific FG rulesets.

Enter PAR5E.

As data content and FG rulesets are so very tightly coupled, PAR5E has been written modularly in an attempt to enable multiple FG rulesets to be supported.

The PAR5E engine itself does not implement any ruleset specific parsing awareness or logic; instead its ruleset extensibility is implemented through the use of independent ruleset libraries (Java Package and Classes) which are dynamically linked to (at run time) providing the specific ruleset and content parsing functions.

The PAR5E tool itself implements a GUI front-end (Java Swing) to allow the configuration and control of the module output as well as provide the main program control logic for the abstracted parsing process. It provides a library of support functions which can be used by the ruleset libraries for common tasks as well as a main program loop. The main program loop takes care of the sequencing and high level functions such as opening, reading, tokenizing, writing and optimising data files and makes use of registered callback handlers (defined in the ruleset libraries) to perform the actual parsing of ruleset specific content and formatting of output data.

As such, the ruleset libraries define and implement the raw input content parsing methods specific to a role playing game system as well as the output format of the XML specific to a Fantasy Grounds v3 ruleset. When coupled with a suitable ruleset library, PAR5E provides an end to end tool for processing bulk data payloads and producing modules to a high quality standard.

PAR5E is bundled with an example ruleset library for the D&D Next/5E role playing game system which produces modules for the D&D Next/5E Fantasy Grounds ruleset. In addition, PAR5E is bundled with a base ruleset Library for CoreRPG. This base library can be extended to support any CoreRPG dervied FG v3 ruleset. The libraries are freely available to the community for re-use and as templates for the development of other ruleset libraries. Further information on developing a ruleset library for PAR5E can be found in the Developing Ruleset Libraries section towards the end of this document.

PAR5E is written in Java and should run on any platform that supports Java and the Swing library. Currently both Microsoft Windows 7 and Apple OS X Mountain Lion have been tested and confirmed as working. Linux and other UNIX based platforms should work albeit have not been specifically tested.

Users who are willing and able to test PAR5E on other platforms including Linux or ruleset developers that wish to exploit PAR5E to enable support for alternative rulesets should contact me for access to the latest development build.

You can contact me via:

- FantasyGrounds Forum PM
- admin@drzeuss.co.uk

Zeus

Installing PAR5E

Prerequisites

PAR5E itself is a very lightweight application (<6Mb in size) consisting of several bundled Java Libraries, a primary .jar archive and a small number of resource for the GUI assets, however PAR5E depends upon and requires several other Java packages including the standard Java Swing library as well as one or more custom Ruleset Libraries. The dependent Java packages as well as a suitable Java environment are detailed in the following sections.

Java

PAR5E is written in Java. As such PAR5E requires a suitable Java virtual machine (32 or 64bit) environment to be installed and available to execute the PAR5E engine.

Most modern day platforms will ship with Java environments already installed or with base packages that install Java the very first time a program tries to execute in a Java virtual machine. Please note that PAR5E requires **Java Run Time Environment 1.7 or higher** and therefore the pre-installed or stock Java versions that are bundled may not suffice, in which case please read on.

The easiest method of obtaining and installing Java is to install the latest run time environment from Oracle's java site.

https://www.java.com/en/download/manual.jsp

Visit the link above and download the latest Oracle Java Package (latest as of writing this document is Java Version 7 Update 51 which will install java-rte-1.7.0_51). Double click or mount the downloaded file to install.

Once installed, you can check to ensure your Java environment is installed correctly by entering the following:

Windows (32bit)

```
C:\Program Files (x86)\Java\jre7\bin>java.exe -version
```

This should result in a similar message to the one below being displayed.

```
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Client VM (build 24.51-b03, mixed mode, sharing)
```

Windows (64bit)

```
C:\Program Files\Java\jre7\bin>java.exe -version
```

Mac OSX

```
java -version
```

These commands should both result in a similar message to the one below being displayed.

```
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

Java Package Dependencies

PAR5E is dependent upon a number of Java Class Libraries all of which are bundled with the main application or available in the Java runtime environment. There should be no need to install additional libraries.

Finally, PAR5E is dependent upon one or more Ruleset Library Package which must be bundled and built at compile time. The initial release of PAR5E supports CoreRPG and 5E rulesets. Other rulesets including D&D 3.5E, Pathfinder and Savage Worlds are planned for a future release.

PAR5E Ruleset Library Package	Function
par5e.CoreRPG	PAR5E Ruleset Library for CoreRPG ruleset
par5e.DD5E	PAR5E Ruleset Library for D&D 5E ruleset

Starting PAR5E

Starting PAR5E couldn't be simpler.

Windows

1. Open Windows Explorer from the Start Menu and navigate to the location of the downloaded PAR5E folder
2. Double click the PAR5E.exe file.

From the command line :

```
cd <path to PAR5E folder>
PAR5E.exe
```

Mac OSX

1. Open Finder and navigate to the location of the downloaded PAR5E folder
2. Double click the PAR5E.app file.

From the command line :

```
cd <path to PAR5E folder>
open PAR5E.app
```

Linux

1. Open Finder and navigate to the location of the downloaded PAR5E folder
2. Double click the PAR5E.jar file.

From the command line :

```
cd <path to PAR5E folder>
java -jar PAR5E.jar
```

Once started, PAR5E loads the default configuration (CoreRPG) and starts the GUI interface.

Using PAR5E

User Interface General Overview

- ✦ The Configuration tab enables users to enter required information to parse a new module. It is split in two.
  - ✦ The top half (left side) is dedicated to essential information; module name, id, type, category name and basic path information.
  - ✦ The paths can be set manually by entering text in the appropriate fields or by clicking on the folder icon at the end of each field. The latter opens a file dialog allowing you to navigate to the desired file/folder location.

    Note: Setting the Output Module Path is required if you want PAR5E to automatically move the compiled module at the end of the compilation to a target folder.
  - ✦ The top half (right side) is dedicated to PAR5E engine options including setting the debug level (effects verboseness of output messages) and the PAR5E ruleset library - which defines and provides the functions for parsing specific ruleset content. By default PAR5E will be set for CoreRPG ruleset and content.
  - ✦ The bottom half of the configuration tab is dedicated to the ruleset content type input paths and inclusion controls. The input path controls are defined at initialisation time by the PAR5E ruleset library (CoreRPG by default) - therefore the contents of the lower half will change based upon the ruleset library selected (top right). Regardless of which ruleset library is in use, all controls offer a text field for defining the path to the input files or folders as well as a folder icon for point n click navigation.
  - ✦ All data input paths are **relative** to the Module Path defined at the top of the configuration tab.
  - ✦ In addition to path controls for the data input files, the lower half also provides up to 3[1] tick boxes per content type. Selecting these tick boxes (labelled as **Cmp, Ref** and **Arch**, **C**ampaign, **R**eference and **A**rchive) determines where PAR5E will write the data to.

TickBox	Output
Campaign	the data is written to the campaign data nodes - data appears in the campaign Story, Encounters, Personalities, Items etc. etc. under individual Tabs
Reference	the data is written to the reference library data nodes - data appears in the reference Library when the module is selected
Archive	the data is written to an Object database for persistent storage and data retrieval. The Object data can be referenced in subsequent parsing of new content. Default location is the PAR5E program folder ./archive.db4o

- ✦ The Archive tab provides an interface to a persistent Object data store that can be used as a source for a parse process.
  - ✦ The top of the tab provides access to path information for the archive database file. The default location is the PAR5E program folder ./archive.db4o
  - ✦ The left side of the tab is dedicated to listing the contents of the archive. Users can select the data type to be listed using a pull down menu, select archive entries from the table and either copy them for inclusion in the module to be parsed or delete them from the archive.
  - ✦ The right side of the tab is dedicated to listing the archive content that will be included in the module output. Again, users can select the data type to display from a pull down menu, select archive entries that are to be included and delete them from inclusion in the output.
- ✦ The Console tab provides a visual cue of the parsing process. The console will display messages as the parsing process is executed. The debug level set (1-3) determines the verbosity of debug messages (sent to STDOUT) where 1 is high level information only and 3 is detailed subroutine messaging. The messages are displayed with a status/information field (right side) which are color coded:

[1] Depends on content type and is defined and controlled by the ruleset library

Colour	Description	Meaning
Blue	Information	all is well
Green	Data	the data was parsed as expected
Red	Warning	the data was not parsed as expected or an error occurred

By default, PAR5E will only display high level information per content type (debug level 1); at debug level 1 individual items are displayed as they are parsed (this is akin to the level of output from the 4EParser). Levels 2 and up are really for development debugging and should be avoided by general users.

Menus, Buttons & Status Bar

PAR5E has a limited set of Menus and Buttons for users to interact with at initial release. I anticipate adding further options moving forward.

File Menu

The File Menu contains configuration file operations including Load and Save. The File Menu also provides a Quit option.

Menu Item	Description
New	Resets and clears the current configuration.
Load	Loads configuration settings from a file.
Save	Saves the current configurations settings to a file.
Quit	Ends the program

Note: the File Menu and its options will be disabled during a parse operation.

Action Menu

The Module Menu contains module operations including Validate Input and Parse.

Menu Item	Description
Validate Input	Verifies all the paths exist/all required information has been supplied
Parse	Verifies all the paths exist/all required information has been supplied, switches to the Console tab and then converts the raw data into FGII format using the specified PAR5E ruleset library.

Note: the Action Menu and its options will be disabled during a parse operation.

Button Tool Bar

PARS5E implements a single tool bar at the top of the main window. It contains Exit, Load, Save and Parse buttons.

Button	Description
New	Resets and clears the current configuration.
Load	Loads configuration settings from a file.
Save	Saves the current configurations settings to a file.

Button	Description
Parse	Verifies all the paths exist/all required information has been supplied, switches to the Console tab and then converts the raw data into FGII format using the specified PAR5E ruleset library.
Exit	Ends the program

Note: the buttons on the Tool bar will be disabled during a parse operation.

Status Bar

PAR5E includes a status bar at the bottom of the window. It provides additional status and progress messages during a parse operation.

Parsing a Module

To parse a module in PAR5E, follow these steps:

1. Start PAR5E
2. Enter the **Name** of the module in the **Module Name** field (avoid characters such as &) - this name is used for the both the .mod file as well as the internal FGII module name
3. Enter the **ID** of the module in the **Module ID** field (avoid characters such as &) - This prevents the library/adventure content objects for this module from conflicting with other modules.
4. Enter the **Category Name** of the module in the **Category Name** field (avoid characters such as &) - this names the group this module will appear under when opened in within the library e.g. Core Rules or Rule Supplements.
5. Enter the Module **Path** of the module in the **Module Path** field; to do this you can either enter the path in the text field or click the folder button at the end of the field to navigate to the folder. This path will be used as a relative base for all the input files
6. If you want your module to have a thumbnail when viewed in the Library, enter the Thumbnail file details in the Thumbnail Path field. Again you can use the text field or the folder button at the end of the field.

   Note: the path is relative to the Module Path and must be a valid .png file.
7. Enter the **Temp Path** of the module in the **Temp Path** field; to do this you can either enter the path in the text field or click the folder button at the end of the field to navigate to the folder. This path will temporarily hold the XML and module output files before its is archived into a .mod (ZIP) file.

   Note: the path is relative to the Module Path
8. PAR5E will automatically dump the output .mod file to the Module Path specified, if you want PAR5E to automatically copy the resulting module into the FG app data modules folder, enter the path in the **Output Path** field.
9. Select the **debug** level (0 = default (recommended), 1+ additional developer messages (not required unless your developing for PAR5E)
10. Select the **PAR5E Library** - by default **CoreRPG** will be selected. Selecting a different Library will update the lower half of the GUI to reflect the data items supported by the selected Library

Depending upon which PAR5E Library is selected will determine which **Module Content Options** are presented in the lower half of the Configration window. In any case all options include:

- ✦ a Path text field - enter a path (relative to the Module Path provided earlier) to the module content file
- ✦ a Path folder button - click to point and select a module content file or folder
- ✦ Module Output Options (**Cmp Ref Db**)
  - ✦ Cmp - select to output the content to the campaign/adventure class windows in FantasyGrounds
  - ✦ Ref - select to output the content to the reference library class windows in FantasyGrounds

- ✦ Db - select to output the content to a database (not supported in initial release)

Note: output options will vary for module content types and specific options may be disabled if not relevant. The output options are defined and controlled by the selected PAR5E Library.

Once everything is entered, select the Validate option from the Modules menu. This will check to ensure all paths are valid.

If the validation is not successful, recheck the paths you have entered are correct.

If validation is successful, you can now either **Save** the configuration file (useful if you want to ever re-parse a module as it saves you having to re-enter all the above information again) or **Parse** the module by clicking the appropriate buttons on the Tool bar or selecting the menu items under **File** and **Action**.

Once you selected/clicked Parse, the view will be changed to the Console tab and the parse process will begin. Watch for output messages in the console. When completed, if successful the module will be saved to the module folder or moved to FGII's module folder (depending upon options selected).

Starting PAR5E with a configuration file pre-loaded

You can start PAR5E with a configuration file from the command line

```
PAR5E.exe -c <name of config properties filename>
```

```
PAR5E.app --args -c <name of config properties filename>
```

```
java -jar PAR5E.jar -c <name of config properties filename>
```

When passed the -c command line parameter and a valid .properties file, PAR5E will start, auto-load the configuration file and begin parsing the module. Once completed it will pause for 20 seconds before gracefully exiting. This allows PAR5E to be called form a command line as part of a script of toolchain. Handy for when re-parsing large module collections.

Troubleshooting Problems

PAR5E is built to catch as many exceptions and errors as possible, where possible PAR5E will warn you of problems in the console output and if debug is enabled in additional debug messages delivered to STDOUT.

Console Messages

When parsing content PAR5E will attempt to determine if all fields of expected data for each object were parsed completely, if so output is Green, if not output changes to Red. If output for particular items are Red, check your source content to ensure completeness of data and also check the syntax is correct and as documented in the various Library guides. Items reported in Red will still be included in the output XML albeit some of the fields of the object when opened in FG will be empty.

Malformed XML

If PAR5E produces malformed XML, this is detected and the parse action is halted. PAR5E will open an XML browser and will report the line and column number of the offending tag. The cursor position is also automatically moved accordingly.

Using this information its possible to identify the problem (usually a missing formattedtext end tag to an element), correcting the source data appropriately should fix the problem.

Debug

If for any reason PAR5E stalls, crashes or panics or its necessary to gather a little more information on whats potentially causing a problem for the engine, you can use the debug feature to output further debug messages from the engine to STDOUT.

To enable debug messages, the Debug value of the engine must be set to greater than 0 **AND** PAR5E must be launched from the command line:

Windows From the command line :

Open a Command Shell Window (cmd.exe)

```
cd <path to PAR5E folder>
PAR5E.exe 1
```

Mac OSX From the command line :

Open the Terminal.app

```
cd <path to PAR5E folder>
open PAR5E.app
```

Linux From the command line :

Open a Terminal window or Shell

```
cd <path to PAR5E folder>
java -jar PAR5E.jar
```

You can capture the debug messages to a file by redirecting STDOUT.

Windows

```
PAR5E.exe 1 >capturefile.txt
```

Max OSX

```
open PAR5E.app >capturefile.txt
```

Linux

```
java -jar PAR5E..jar  >capturefile.txt
```

Developing Ruleset Libraries

To be completed.