Antenna House Regression Testing System

User Guide

Antenna House Regression Testing System
User Guide

About the Antenna House Regression Testing System

The Antenna House Regression Testing System (AHRTS) is a fast and scalable solution for automating the visual comparison of formatted documents or pages. It is not a solution for tracking changes or locating variations in the content of multiple versions of the same document--its main function is to identify the visual dissimilarities that exist between a pair or set of documents. That being said, AHRTS does have some capability to locate content discrepancies within a limited set of parameters.

AHRTS was originally developed to meet Antenna House's requirement for regression testing new releases of our Formatter software. We are now making AHRTS available commercially to meet the needs of customers who must also perform regression testing of their output whenever any aspect of their production process has been altered. This includes not only the output tool, but also the editing, content management and graphic tools that are used. A slight change in any of the software- or hardware of a system could have unintentional consequences for the integrity of the final output--text, graphics and tables could be altered unexpectedly. Determining whether or not changes occurred during the production process can be a very time consuming and tedious task. If the changes are minute, they may go undetected and ultimately lead to errors in the information being distributed.

Originally designed to work with Antenna House Formatter, the software can now be used with any PDF output software. This includes formatting software such as Antenna House Formatter and desktop publishing systems, to business applications, report generators, graphic packages, word processing software and every other imaginable application that generates a PDF file. A change to any of your system components must be regression tested to ensure the integrity of the resulting product.

What makes this application special is its visually-oriented solution. AHRTS performs a precision pixel-by-pixel comparison of two PDF files and generates a report that allows for the quick visual identification of any discrepancies that exist between the compared documents. This system greatly increases the efficiency of regression testing, reducing the amount of time and effort required to manually check output discrepancies. AHRTS enables users to accurately perform regression tests by using a more comprehensive test set that can be processed in hours as opposed to days. It will enable users to find even the slightest change in the visual appearance of a document.

AHRTS can also be used to test that changes made to any of the style sheets and/or software produce the intended results. Has a change in margins, a shift of lines or a modification of fonts been applied as expected? Have they produced any unforeseen results? These are cases where the regression testing tool via its visual output can quickly help determine how the changes have been applied, and if any unintended consequences have occurred.

AHRTS is available in four versions. The table below displays the capabilities of each:

Feature	Standard	Plus	Expanded	Distributed
Easy to Use GUI	X	X	X	X
Compare 2 Versions of the Same PDF	X	X	X	X
Compare Multiple PDFs Contained in Directories		X	X	X
Compare Documents Using 2 Versions of Formatter			X	X
API for Integration into Workflow				X
Distributed Regression Testing across Multiple Systems to Speed Testing				X

The Standard, Plus and Expanded are offered both as Standalone and Server versions. The Server version includes an API to enable integration of AHRTS into the tool chain. A Standalone license includes a GUI and may be used on a laptop, desktop or workstation by an individual. The Server license is intended for use by multiple people or for integration with other applications via the command line API.

Automated Regression Testing and its Benefits

The traditional solution to the visual regression testing of a formatted paged output has been to visually compare two documents side-by-side and page-by-page to see if changes in the software produced the intended--or unintended--results, or disrupted some part of the production process that was previously functioning at normal. It involves testing either a subset of documents or a large collection of documents (this is the most desirable). You are comparing visual output to determine if the new file corresponds to, or diverges from, the reference file.

The challenge is that different underlying code can produce the same output. To the best of our knowledge, there has been no commercial product available to handle the regression testing of large numbers of documents.

Traditional regression testing is time consuming, costly, resource intensive and unreliable. It is a monotonous chore that nobody ever wants to do; nevertheless it is absolutely necessary to ensure an error-free final product. The fact that this approach is prone to missing minor--but often critical--differences can lead to delays in product release.

The Benefits of Automated Regression Testing

- 90%+ reduction in human effort
- Greatly reduces processing time
- Significantly more accurate
- Able to detect subtle differences
- Easy multiple re-testing
- Transforms what was a labor-intensive chore into a quick and easy process

Table of Contents

Installation

Windows

Prerequisites

Standard/Plus Versions	Expanded/Distributed Versions
Windows XP or later	Same as Standard/Plus
Oracle Java Virtual Machine 6 (recommended for optimum performance)	Plus the 2 or more versions of AH Formatter you will be regression testing
Antenna House XSL Formatter V4.3 or later * PDF Viewer	

Installation Steps

Standard/Plus Versions	Expanded/Distributed Versions
1. Download the latest version of AHRTS: ftp://support.antennahouse.com/ahrts	1. Same as Standard/Plus
2. Run the *ahrts-install.jar* installer by double-clicking the file	2. Manually setup location of the versions of Antenna House Formatter in the engine configuration file.*
3. Follow the steps indicated by the install wizard	
4. Enter the license key provided to you in the installation directory	

Linux

Prerequisites

Standard/Plus versions	Expanded/Distributed Versions
Oracle Java Virtual Machine 6 (recommended for optimum performance)	Same as Standard/Plus
Antenna House XSL Formatter V4.3 or later	Plus the 2 or more versions of AH Formatter you will be regression testing
PDF Viewer	
MuPDF V1.0 or higher. This can be downloaded from http://www.mupdf.com/	

Installation Steps

Standard/Plus versions	Expanded/Distributed Versions
1. Download the latest version of AHRTS: ftp://support.antennahouse.com/ahrts	1. Same as Standard/Plus
2. Run the *ahrts-install.jar* file by double-clicking it or running *java -jar ahrts-install.jar* from the command line	2. Manually setup location of the versions of Antenna House Formatter in the engine configuration file.*
3. Follow the steps indicated by the install wizard	
4. Copy the license key provided to you into the installation directory	

* If Antenna House Formatter is installed in the default directories, this step is not necessary.

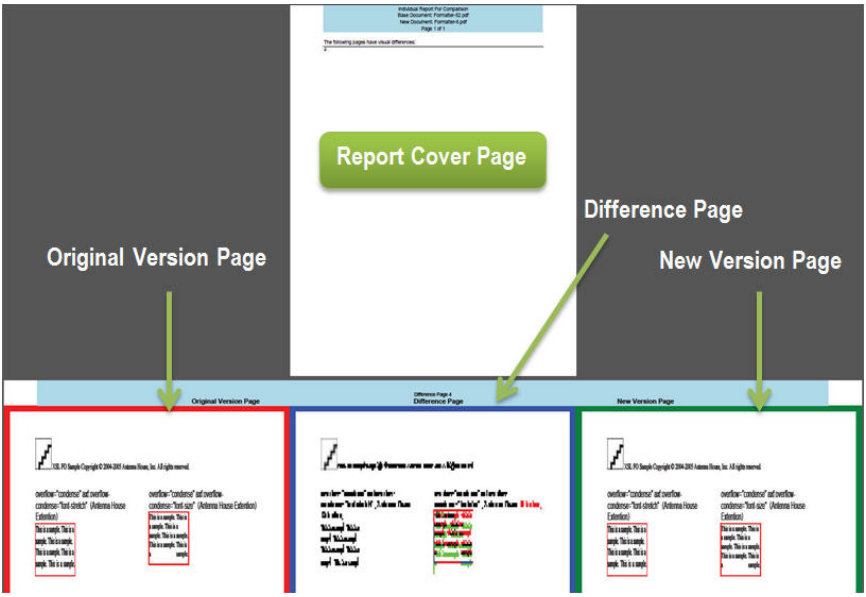
Understanding the Reports

Reports

The compare step generates an Overview Report and an Individual Document Report. The Overview Report is created only when multiple documents are compared. It shows which documents have do and don't have differences. The Individual Document Report is a report for each set of compared documents that have been identified as having differences. If you want to send the report PDFs to someone or move them you can, as long as you keep it in the same relative position as the rest of the output. An easy way to do this is to create an archive (both zip and tar work well) of the output directory and move that to the desired location.

Individual Document Reports

An individual document report is generated for each document containing a difference. The report has a cover page indicating which pages of the document contain discrepancies, and then an additional report page for every page of the tested document with a difference. (In cases where no discrepancies have been identified, only a cover page will be produced.)



Reading the Color-Coded Reports

Both the left and right panes are the actual pages from the original PDFs and thus have all the color attributes that were in the original documents. In the center is a composite/ difference image which is composed of a grey scale version of the baseline image. Areas that were white in the baseline image but not white in the new image are colored green. Areas that were not white in the baseline image but white in the new image are colored red. The areas that were not white in either image but not the same color in the new image are colored blue. The following summarizes the color scheme:

Black/Grayscale	No Differences. This is a composite of those portions of the left and right pages that are identical.
Red	Is on the original PDF, but not on the new PDF.
Green	Is on the new PDF, but not on the original PDF.
Blue	Content in the same location on both PDFs, but there is a difference.

The following are examples of differences that may appear between PDFs:

- Color change
- Change in the type or weight of line
- Font change
- Slight pixel shift

(**Note**: The reports are generated using Antenna House XSL Formatter. If you have Formatter on your machine, then that copy will be used for the reports. If you do not already own a copy, then a restricted version of Formatter is provided with AHRTS to enable you to generate reports. What makes the report possible is Antenna House Formatter's ability to merge selected individual pages from a PDF into a single PDF file it creates. Formatter creates the individual pages of the report by using a page from each of the PDFs being tested and a bitmap image that was created during the regression testing process.)

In the example below, we can see a clear difference in supporting floating objects between the original and the new version of the same document.

Overview Reports

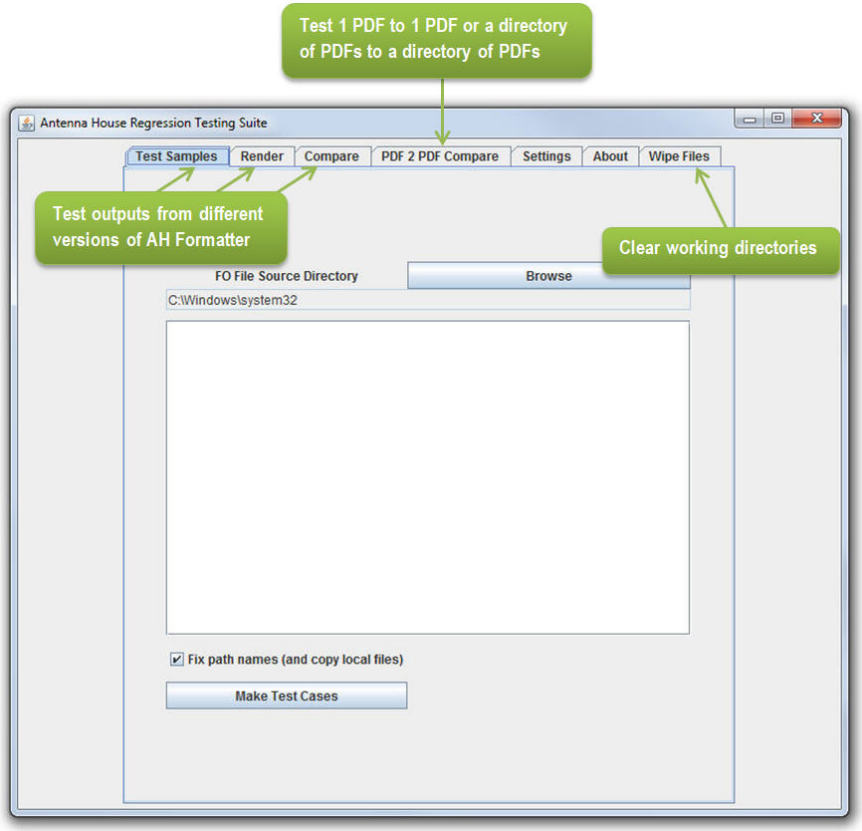
For the overview report, the documents containing no differences are displayed in green, while those with differences in red. The documents with differences (red) are hyperlinked to their respective individual document reports. Simply click the link to view any of the reports.

GUI Operation

Understanding the GUI

When using AHRTS for the first time, follow these steps to get up and running. (The term "ahrts-gui" will be used in place of *ahrts-gui.bat* [Windows] and *ahrts-gui.sh* [Linux]).

The GUI consists of 7 tabs across the top:



1) The first three tabs (*Test Samples*, *Render* and *Compare*) are used when testing different versions of Formatter.
2) The *PDF 2 PDF Compare* tab is for comparing individual PDFs or directories of PDFs against one another.
3) The *Settings* tab is for customizing the directory settings used by AHRTS.
4) The *About* tab shows the version of AHRTS and the Open Source software used in its development.
5) The *Wipe Files* tab is used to clear the working directories before starting a new test.

DPI Settings

AHRTS works by converting the PDFs into bitmap images and then doing a pixel-by-pixel comparison of the generated bitmaps. This enables you to compare anything on a page (including color) regardless of the document's content—language has no bearing on the process whatsoever.

The Dots Per Square Inch (DPI) of a comparison is the resolution at which a PDF document from a rendering engine is rasterized. This limits the precision of the comparison, which is useful if you need to process numerous or large-sized documents. (The more pixels to compare per page, the longer the comparison will take.) At 30 DPI, you have 900 dots per square inch. At 60 DPI, you have 3,600 dots per square inch. At 100 DPI, you have 10,000 dots per square inch.

In both the "Compare" and "PDF2PDF Compare" tabs, there is an option for setting the DPI. The purpose of setting the DPI is to allow the user to control the level of comparison, and how sharp the composite page looks in the final report.

The table below gives a description of your ability to detect differences at various DPI settings:

DPI Setting	Difference Detection Ability
30 DPI	Provides a composite image that enables locating differences, but the type is not readable.
60 DPI	The type becomes very readable.
90 DPI	The type is displayed at close to screen resolution. Minute changes to small characters (e.g., the accent over a letter) may actually be difficult to identify.
>90 DPI	Possible to generate differences from changes as minute as a very slight shift in a character serif.

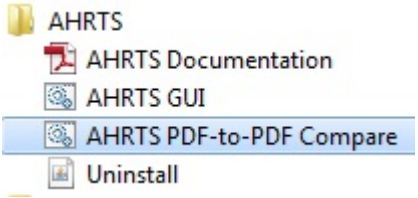
In our experience of testing and using AHRTS, we have found 30 DPI to be more than enough resolution to find differences in font types (normal vs. italic or bold, serif vs. sans-serif), minor alterations in spacing and line width, as well as larger ones such as differences in the breaking of pages.

300 DPI is enough resolution to view a document under magnification while maintaining image clarity, so it is far beyond the necessary resolution to locate even half-point changes in font size (72 points equals 1 inch). Because each pixel in a rasterized page is an average of the area it covers on the vector graphic, the average color value of the underlying vector parts are reflected in the resulting pixel—any ratio alterations will have an impact on the resulting pixel. Even a subpixel-sized difference can be located without identifying the exact vector position of the change.

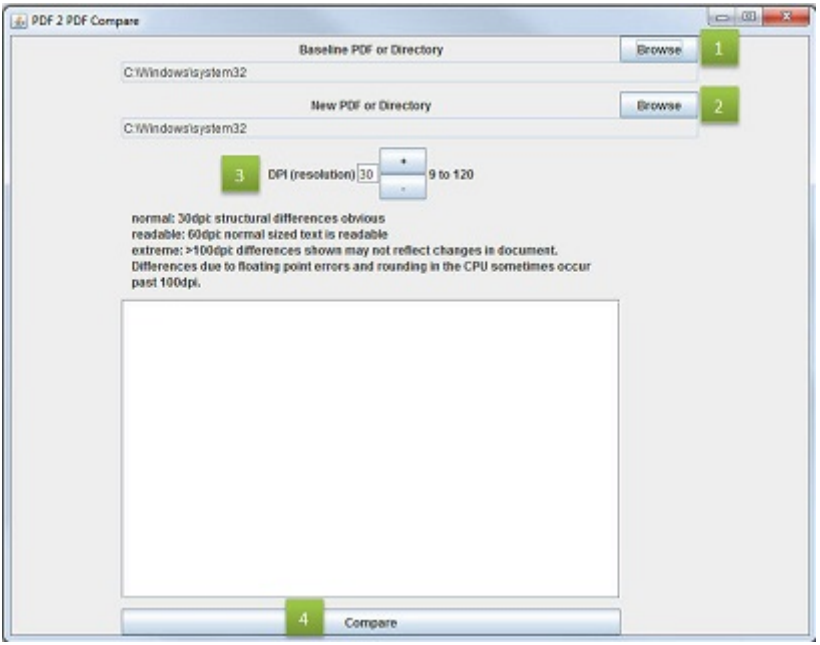
Quick Start Guide

PDF2PDF (Testing Individual PDFs)

Open the PDF2PDF Compare GUI either via the shortcut in the start menu or the *ahrts-pdf2pdf-gui* script in the install directory.



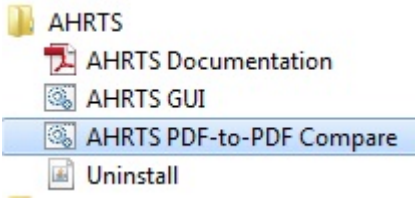
- Select the first (1) and second (2) PDFs you wish to compare.
- Select your DPI setting (3).
- Click the *Compare* button (4) to begin.



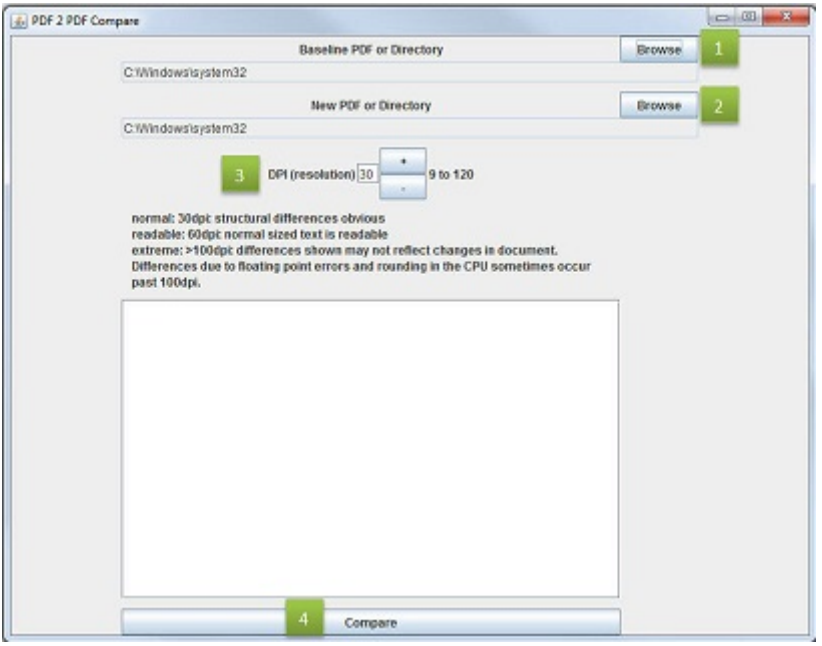
Once the process has finished, a report will automatically open.

PDF2PDF (Testing PDF Directories)

Open the PDF2PDF Compare GUI either via the shortcut in the start menu or the *ahrts-pdf2pdf-gui* script in the install directory.



- Select the first (1) and second (2) directories you wish to compare.
- Select your DPI setting (3).
- Click the *Compare* button (4) to begin.

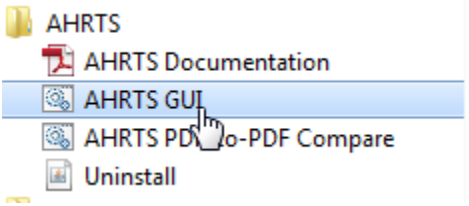


A multi-document report will automatically open once the process has finished.

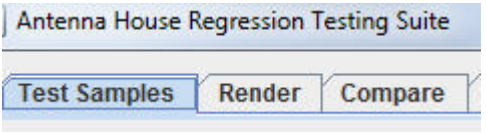
Testing Multiple XSL-FO Files with 2 Versions of Formatter

Create a Test Case

Open the GUI either via the shortcut in the start menu or the *ahrts-gui* script in the install directory.

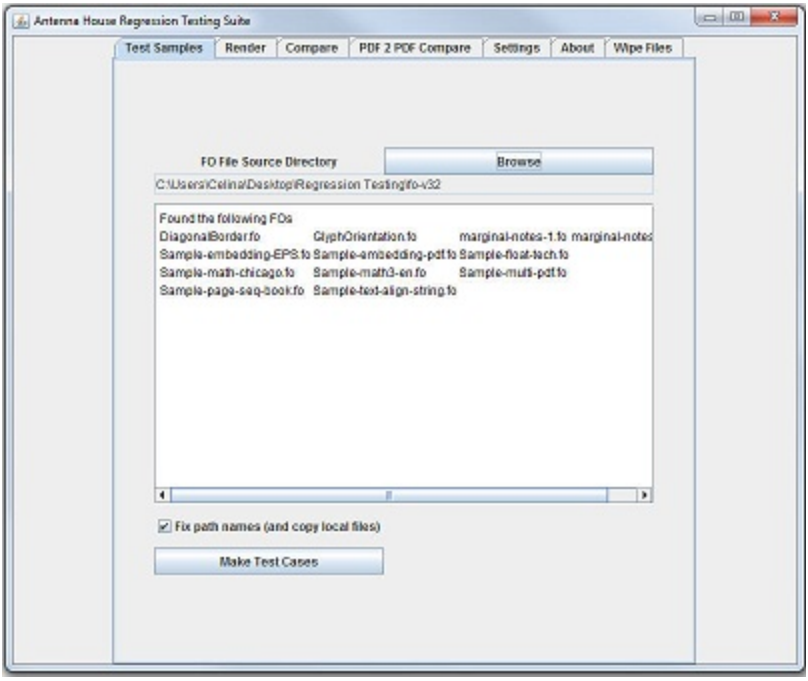


Navigate to the *Test Samples* tab.



Click *Browse* and select a directory that contains your FO files. Uncheck *Fix Path Names* if you do not wish to include local files referenced by the FOs, and adjust the paths to those files to match the new location.

(**Note**: The FO files must not be in sub-directories if you intend to include them. This restriction is in place to help prevent accidentally loading very large sets of FOs.)



Click *Make Test Cases* to make the test cases in the Test Cases Directory specified on the Settings page. (Demo/test-cases in the install directory by default)

Render Test Cases

This step may need to be repeated to produce two sets of rendered documents the first time the system is run. One set can be compared to itself, but no differences will be found.

- Select a rendering engine from the Select *Rendering Engine* drop-down list.
- Click the *Render* button.
- Repeat for 2nd engine.

Compare the two sets of rendered documents:

- Select a baseline rendering engine from the *Baseline Rendering Engine* drop-down list.
- Select a new rendering engine from the *New Rendering Engine* drop-down list.
- Adjust the DPI as needed; d) click the *Compare* button.

A multi-document report will automatically open once the process has finished.

Edit Engine Files

The Engine Files tell AHRTS where to find the different versions of Antenna House Formatter that will be used with the software. AHRTS comes with many premade engine files which should work for default installations of Formatter. For demonstration purposes, using a full license to render once and then an evaluation license to render a second time under a different engine version gives you a visual example of the comparison, as well as an idea of how long the process may take if each document contained discrepancies. If you already have two versions of Formatter installed in the default locations, you should not need to edit anything.

The following is an example of an engine file:

```
<?xml version="1.0"?>
<engine-config>
    <type>FO</type>
    <name>Formatter</name>
    <version>53</version>
    <config-version>1</config-version>
    <cmd-path>C:\Program Files\AntennaHouse\AHFormat-
terV53\AHFCmd.exe</cmd-path>
    <arguments>
        <arg sub="false" name="-x">4</arg>
        <arg sub="true" name="-d"><path name="input-
file"/></arg>
        <arg sub="true" name="-o"><path name="output-
file"/></arg> <!-- The 'output-file' path is generated au-
tomatically, it is not specified in the Manifest.xml -->
        <arg sub="true" name="-i" omit-if-emp-
ty="true"><path name="config-file"/></arg>
        <arg sub="true" name="-s" omit-if-emp-
ty="true"><path name="stylesheet"/></arg>
    </arguments>
    <environment>
        <var name="AHFS10_FONT_CONFIGFILE"><path
name="font-config"/></var>
    </environment>
</engine-config>
```

Settings

The settings tab has options that correspond to the contents of the *ahrts.properties* file. The default settings will work out of the box for all the examples cited in this manual.

- **Test Case Directory -** The directory from which AHRTS reads the test cases.
- **Render Output Directory -** The directory where AHRTS stores the output of the rendering step and reads the same as input for the comparison step.
- **Comparison Output Directory -** The directory where AHRTS stores the XML output of the comparison step.
- **Difference Image Output Directory -** The directory where AHRTS stores the difference images created during the comparison step.
- **Report Output Directory -** The directory where AHRTS stores the reports generated during the comparison step.
- **Engine File Directory -** The directory from which AHRTS reads the rendering engine

files.

- **Temporary File Directory -** The location AHRTS uses when writing temporary files to disk.
- **MuDraw Executable -** The absolute path to the mudraw (mudraw.exe on Windows) executable.

Command-Line Interface

Configuration

Configuring AHRTS involves a three-part process:

- Edit/create a properties file.
- Edit/create engine files for the software used to render the PDFs.
- Edit/create manifest files for the test cases to be used.

Not all of these steps are required. For example, the creation of manifests is done by the script that changes your FOs into test cases, and AHRTS comes with example engine files that may work without editing.

Properties File

The properties file is the file that holds user-configurable settings. The default location of the properties file is labeled ahrts.properties in the installation directory, but it can be specified via the -p option in all CLI programs.

Properties File Format - The properties file is a Java properties file and shares its specific formatting rules. The following is an example of this format:

```
# comment
key=value
```

For our system there are a few refinements:

- Spaces are allowed in paths, but not quotes
- Only forward slashes are allowed in path (in Windows replace any backslashes "\" with forward slashes "/")
- No trailing slashes on directories

Property File Values -The system that reads the properties file will ignore keys it does not recognize (e.g., “Key” is not the same as “key”).

The following are valid key and value descriptions:

- **dataDir** - The absolute path to the location of the test cases.
- **configDir** - The absolute path to the location of the directory that contains configuration files and directories (only change this default setting if the configuration files are located somewhere else).
- **tmpDir** - The absolute path to the location of the directory you wish the rendered pages to be written.
- **engineDir** - The absolute path to the location of the engine directory.
- **testOutputDir** - The absolute path to the location of the directory where the test results will be placed after the render step and analyzed during the compare step.
- **compareOutputDir** - The absolute path to the location of the directory where the compare results will be placed after the compare step and analyzed during the report generation step.
- **diffImgDir** - The absolute path to the location of the directory where the difference images will be placed after the compare step and analyzed during the report genera-

tion step.
- **reportDir -** The absolute path to the location of the directory where the reports will be placed after the report generation step.
- **pdfDraw -** The absolute path to the location of MuPDF's PDFDRAW executable.
- **axfExt** - Antenna House XSLFO Namespace Extention generally uses axf, but this depends on its definition in the FO.

Engine File Format

Engine files are used by the system to recognize and label the particular rendering software. The following is a common format to all engine file types:

```
<?xml version="1.0"?>
<engine-config>
<type>{input type}</type>
<name>
<!-- for now the file name must match the name specified
here -->
{Software Name}
</name>
<version>
<!-- for now the file name must match the version speci-
fied here -->
{Software Version}
</version>
<config-version>
{Unsigned  integer matching a config version in the test
case definition}
</config-version>
<!-- engine specific configurations  here -->
</engine-config>
```

The root element is *engine-config* and contains the following child elements:

- **type -** The input type (FO/*XSL/CSS/etc).
- **name -** The name of the software as it will appear in the output file names.
- **version -** The version of the software as it will appear in the output file names.
- **config-version** - The version of the configuration specified in the test-cases.

Engine Command-Line Notes

When an engine is invoked, the command-line from the engine file will run in the system console. This console does not inherit any environmental values. When necessary, those must be set by using a wrapper script or the environmental variable config. Below is the file format for the command-line engine type:

```
<?xml version="1.0"?>
<engine-config>
<type>{input type}</type>
<name>{Software Name}
<version>{Software Version}</version>
<config-version>{Unsigned integer}</config-version>
<cmd-path>{ Absolute path to the software, for example C:
\Program Files\Antenna
House\AHFormatterV6\AHFCmd.exe }</cmd-path>
<arguments>
<arg sub="false" name="-x">
<!-- This says to use this value ("4") directly for option
"-x" –4 -->
</arg>
<arg sub="true" name="-d">
<!-- This says to substitute the value found when evaluat-
ing the child "path" element as the value of option "-d" --
>
<path name="input-file"/>
<!-- the path element references a correspondingly named
element in the test case manifest file -->
</arg>
<arg sub="true" name="-o"><path name="output-file"/></arg>
<arg sub="true" name="-i" omit-if-empty="true">
<!-- This says to substitute the value of the child "path"
element but it that is empty, then it should not be passed
on the command line -->
<path name="config-file"/>
</arg>
</arguments>
<environment>
<var name="AHFS10_FONT_CONFIGFILE">
<!-- This will (when this feature is implemented) create
an environmental variable of the var@name with the value
of the evaluation of the child path element, or the string
value just as is done above. -->
<path name="font-config"/>
</var>
</environment>
</engine-config>
```

The root element is engine-config, and the engine specific child elements are as follows:

- **cmd-path** - The absolute path to the script or executable for the program to be run
- **arguments** - The element that contains the arguments to be passed to the executable specified in cmd-path as child arg elements
- **arguments/arg** - The individual argument with attributes determining the way it will be passed, and a value that will be processed and/or passed for the option. It has the

following attributes:

- *arguments/arg@name*: (string) The command-line option as it will be passed to the executable (e.g., -x to set the exit level for Formatter).
- *arguments/arg@sub*: (boolean) If true, the system will try to substitute the value of this element for the item it represents (default is true).
- *arguments/arg@omit-if-empty*: (boolean) If the value or the item substituted for it is empty, then the system will not pass it on the command line (default is false).
- *arguments/arg.value*: (string or XML) Either the string representation of the value to be passed to the argument, or an XML element that can be used to reference elements in the test case manifest files
- *arguments/arg/path*: An element whose name attribute is a pointer to its counterpart in the test case manifest files. The value of the name is an arbitrary string that has an identical counterpart in one or more test case manifest files.
- *environment*: (When implemented) The element that contains the environmental variables to be set before running the executable.
- *environment/var*: The individual element that contains the name of the environmental variable (name) and the value it's to be set to.

Manifest File

The manifest file is in the top level directory of the individual test case and contains the test case instructions for AHRTS.

The bare minimum contents of a manifest file are as follows:

```
<test-manifest>
<test-name>test</test-name>
<path name="input-file">test.fo</path>
<conversion type="FO"></conversion>
</test-manifest>
```

An example of a manifest for use with Formatter with different configuration requirements between the versions to be compared:

```
<test-manifest>
<test-name>test</test-name>
<path name="input-file">test.fo</path>
<conversion type="FO"></conversion>
<engine-configuration name="Formatter" config-version="0">
<path name="config-file" >ahfsettings.xml</path>
<path name="font-config">font-config-0.xml</path>
</engine-configuration>
<engine-configuration name="Formatter" config-version="1">
<path name="config-file" >ahfsettings.xml</path>
<path name="font-config">font-config-1.xml</path>
</engine-configuration>
</test-manifest>
```

Test Manifest Element Descriptions

- **test-name** - The name of the test (also the directory inside the test cases directory that contains it).
- **conversion** - Information about what is being converted, and to what it is being converted.
- **conversion@type** - The type of file(s) being converted.
- **engine-configuration@name** - The name of the engine targeted by these settings.
- **engine-configuration@config-version** - The version of these settings. (This refers not to the version of the engine, rather the version of the configuration. (See configuration version [p. 19] in the glossary.)
- **path.value** - The value used to replace the path element matching the @name of this path element in any engine file.
- **path@name** - The name used to reference the correspondingly named path element in any engine file.

More Specifically

- **test-manifest/path@name="input-file"** - The value of this path element will be used as the input file by any engine (-d equivalent for Formatter).
- **engine-configuration/path@name="config-file"** - The value of this path element will be used as the configuration file by any engine (-i equivalent for Formatter).
- **engine-configuration/path@name="font-config"** - The value of this path element will be used as the font configuration file by any engine.

Test Case (Directory Structure)

Directory is the name of the test case, which contains a manifest file with the name manifest.xml (case sensitive). The manifest file can point to anything as a test file, but for the sake of simplicity it is recommended that all files directly related to a test case be placed in the same directory with the manifest file.

Creating Test Cases

(The term "ahrts-make_test_cases" will be used in this documentation in place of *ahrts-make_test_cases.bat* [Windows] and *ahrts-make_test_cases.sh* [Linux].)

Requirements:

- An FO file in a directory or directory full of FO files
- A (preferably empty) directory for the test cases in which to place the test cases
- A properties file with the dataDir value set (optional)

To Run the Script:

```
ahrts-make_test_cases -s <your directory of FOs>  -t <the
directory where your test-cases will be placed>
```

Check the test cases to be sure they are set the way you need:

- If you have a properties file already setup, you can omit -t and use -p to specify the properties file.
- If you want locally stored images to be copied to the test case directories, use -d to have this script grab anything it can locate.
- If you want the paths to be adjusted to fit the new location of the FO files, use –f (implicitly sets -d).

Generating Test Data

Render Test Cases (The term "ahrts-render" is used in place of *ahrts-render.bat* [Windows] and *ahrts-render.sh* [Linux].)

To render one test case:

```
ahrts-render -t <test name> -e <engine name> -v <engine
version>
```

To render all test cases:

```
ahrts-render -all -e <engine name> -v <engine version> Run-
ning a Visual Comparison
```

To compare one test case:

```
ahrts-compare -t <test name> -be <baseline  engine name> -
bv <baseline engine version> -ne <other/new engine name> -
nv <other/new engine version>
```

To compare all test cases:

```
ahrts-compare  -all -be <baseline  engine name> -bv <base-
line engine version> -ne <other/new engine name> -nv <oth-
er/new engine version>
```

Visual Comparison across Distributed Systems

The distributed comparison is comprised of two components: a server and a client. The server will collect all the tasks in a target directory and then listen for connections from clients that are ready to process those tasks. The client component requests tasks, one at a time, from the server. When it is done processing a task, it requests a new one. When the server reports that there are no more tasks, the client ceases to request new tasks.

Only one server should be run, but there may be multiple clients. If a machine has several processors or cores, it can run multiple clients.

To run server:

```
prompt> ahrts-compare-service.bat <public IP> <engine-A>
<engine-B> [test-name ...]
```

The engine-A and engine-B arguments are the version strings that identify which engine was used to generate the PDFs you wish to compare.

You can specify an arbitrary number of tests (by name) to be compared. If you do not specify any test names, then all the tests found in the result directory will be tested.

To run client:

```
prompt> ahrts-compare-client.bat <public IP> <server IP>
```

The server should be started first, then the clients can be started and will begin processing the tasks issued by the server. Once all the clients have completed their task and exited, the server can be stopped with CTRL-C.

Quick Document Compare (PDF to PDF Comparison)

Use the following:

```
ahrts-pdf2pdf -b <baseline file or folder> -bt <tag for
baseline> -n <new version file or folder> -c <path to
pdfdraw> -o <diff image directory> -nt <tag for new ver-
sion>
```

Image to Image Comparison

(This feature has yet to be implemented as a stand alone tool.)

Manual Test Report Generation

The Antenna House Regression Testing System generates a visual comparison output as an XML file. To make these files easier to read, AHRTS is configured by default to use Antenna House Formatter to generate a PDF report from the compare result XML.

The overview report lists each of the input documents, and whether or not visual differences were located between the results of the chosen rendering engines.

When discrepancies have been identified, an individual report is generated for each document containing a difference. The individual reports list the pages containing the problems, as well as a triple-wide page displaying each individual difference. For each page containing a discrepancy, three color-coded documents will be displayed: a page from the base engine (left side coded red), one from the new engine (right side coded green) and a bit composite image displaying the differences between the two (in the center coded blue).

Antenna House Formatter is the only FO rendering engine that has the necessary functionality to generate the various reports.

The XSL-FO files are located in the reports output directory configured through the AHRTS.properties file.

The style sheets to generate the FO files are included in the report directory of the AHRTS install directory.

(**Note**: these style sheets rely on Antenna House Formatter proprietary extensions for styling and linking functions.)

Glossary

Baseline
: The version of the rendering engine or document(s) produced by it that are used as the reference for comparison.

Configuration Version
: The version of the configuration that goes with one or more versions of the rendering engine. This is not necessarily the same as the version of the rendering engine.

DPI
: The Dots Per Inch (DPI) of a comparison is the resolution at which the output document from a rendering engine is rasterized. Adjusting the DPI settings may limit the precision of your comparison.

New
: The version of the rendering engine or document(s) produced by it that is used as the object to be measured in a comparison.

Engine Types
: Currently there is only one engine type (command-line). There will be at least one more that is specifically for Antenna House Formatter.

Rendering Engine
: Software used to process the test cases into output documents for comparison.

Temporary File Directory
: The temporary file directory can be set to a ram disk to improve performance, or a spinning platter hard drive to protect a solid state drive. At any given time, no more than one document (2x the page count) worth of data (rasterized pages, etc.) per comparison thread should be contained in this directory.

Notes on Usage

DPI Settings

In our experience of testing and using AHRTS, we have found 30 DPI to be more than enough resolution to find differences in font types (normal vs. italic or bold, serif vs. sans-serif), minor alterations in spacing and line widths, as well as larger ones such as differences in the breaking of pages.

300 DPI is enough resolution to view a document under magnification while maintaining image clarity, so it is far beyond the necessary resolution to locate even half-point changes in font size (72 points to 1 inch). Because each pixel in a rasterized page is an average of the area it covers on the vector graphic, the average color value of the underlying vector parts are reflected in the resulting pixel—any alteration of average color value will have an impact on the resulting pixel. Even a subpixel-sized difference can be located without identifying the exact vector position of the change.

Suggested Precautions

1) Wipe files before starting a new regression test to keep the size of the working directories manageable and to remove test cases that you do not want included in the new reports.
2) Close PDF files before starting the compare process. AHRTS needs to be able to read and write PDFs in order to process them and generate reports. If a PDF is open in Acrobat, then AHRTS will not be able to access that PDF or write over an old version when generating a new report. (Note: If the report PDF is left open, Formatter will not be able to write a new one. An FO file will still be created that can be manually processed through Formatter once the previous PDF is closed. This saves the need to rerun the entire regression test over because a PDF file was not closed.)
3) Use short directory paths. When you browse to a document or directory you have to navigate via the folder structure. Navigating through multiple folders can be very cumbersome.
4) Periodically clear the temporary file directory. AHRTS places the bitmaps in this directory during the testing step.

Open Source Components

The following Open Source components were used to build the Antenna House Regression Testing System:

MuPDF

http://mupdf.com/

MuPDF's mudraw.exe is included when AHRTS is installed on Windows systems. It is utilized by AHRTS to convert PDF pages to rasterized images.

MuPDF is Copyright 2006-2012 Artifex Software, Inc.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Saxon

http://saxon.sourceforge.net/

Saxon-HE is included as a Java Library (JAR) for the purpose of executing XSLT 2.0 transformations.

Saxon-HE is Copyright (C) Saxonica Limited and distributed under the Mozilla Public License 1.0. The full text of the license can be accessed at <http://www.mozilla.org/MPL/1.0/>.

Akka

http://akka.io/

The Akka library is included as a Java Library (JAR) and used by AHRTS for distributed threading.

The Akka library is licensed under the Apache 2 license (quoted below). Copyright 2009-2011 Typesafe Inc.

<http://www.typesafe.com>

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDI-

TIONS OF ANY KIND, either express or implied. See the License for the specific language governing the permissions and limitations under the License.

Index