Palmtree – User documentation  
2-feb-2021

Max van den Boom  
Benny van der Vijgh

Index

[Palmtree startup options 3](#_Toc63168626)

[Command-line arguments 3](#_Toc63168627)

[<App or Taskname>.config file 3](#_Toc63168628)

[Default or User-prompt 4](#_Toc63168629)

[Modules 5](#_Toc63168630)

[Logging and data-output 6](#_Toc63168631)

Palmtree startup options  
  
All Palmtree Windows-Application projects allow configuration using command-line arguments or using the .config file that accompanies the application (the file is called App.config in the source or <taskname>.config in the build). Command-line arguments will take precedence over .config files

## Command-line arguments

|  |  |
| --- | --- |
| **Argument** | **Function** |
|  | |
| -nogui | Do not load or show a graphical user interface (GUI) |
| -startupconfigandinit | Automatically start the configuration and initialization at startup |
| -startupstartrun | Automatically start a run at startup |
| -parametersfile | Automatically load a parameter file at startup |
| -language | Overwrite the windows-environment culture setting with the given argument. This will cause the respective language resource file to be used. If a language does not exists, then the language will revert to English, |
| -source | Specify the source-module that should be used  The current options are:   GenerateSignal  KeypressSignal  NexusSignal  PlaybackSignal |
|  |  |
|  |  |
|  |  |

## <App or Taskname>.config file

In the source code, an app.config file is part of each project. This file, upon building will be placed next to the executable and renamed to <appname>.config. The pipeline source-module and filter-moduls are configured in a configuration section that looks something like this:  
  
 <Pipeline>

<Filters>

<add name="FeatureSelector" type="RedistributionFilter" />

<add name="TimeSmoothing" type="TimeSmoothingFilter"/>

<add name="Adaptation" type="AdaptationFilter"/>

<add name="LinearClassifier" type="RedistributionFilter"/>

<add name="KeySequence" type="KeySequenceFilter"/>

<add name="ThresholdClassifier" type="ThresholdClassifierFilter"/>

<add name="ClickTranslator" type="ClickTranslatorFilter"/>

<add name="Normalizer" type="NormalizerFilter"/>

<add name="Wasup" type="FlexKeySequenceFilter"/>

</Filters>

</Pipeline>  
  
The filter are ordered in the way they are defined in this section. Additionally you can configure a .config file to automatically use one of the source modules by adding:  
 <Pipeline>

**<Source name="Source" value="KeypressSignal" />**

<Filters>

Default or User-prompt

When no source-module is specified in by a command-line argument nor .config file, then user will be prompted upon startup on which source module to use

# Modules

Source

Filters

# Logging and data-output

|  |  |  |  |
| --- | --- | --- | --- |
| **What** | **File** | **Usage** | **Remarks** |
| *From the start of the program till the end of program* | | | |
| System and application related output | /log/<file>.log | - Feedback on application  - Debugging | Always on, contains information about the working of the application. This information is also displayed in the GUI console (if enabled) for the expert. |
| *For each ‘Start’ till ‘Stop’ (each run)* | | | |
| Initial parameters settings of all modules | <file>.prm | - Replaying data  - Signal analysis (research)  - Debugging | Same file as when you would save all the parameters, thus this file could be used to load the exact same parameter settings. |
| Raw sample data  - direct input from source | <file>.src | - Replaying data  - Signal analysis (research)  - Debugging | Can have a different frequency than the output of the source (e.g. 200Hz, while the output of the source might be 5Hz). |
| Raw sample data  - Output from the source  - Output from the filters  - Intermediate values (e.g. adaptation means) | <file>.dat | - Replaying data  - Signal analysis (research)  - Optimization of  processing parameters  for the user  - Debugging | This data is logged with the same frequency. |
|  | | | |
| Events | <file>.evt |  |  |
| Events level 1  - Start and end of program modules  - Settings of program modules at start  - Source and filter runtime reconfigurations  (including logging new settings)  - Source and filter events (e.g. adaptation  started or re-adaptation; or keysequence  trigger in filter; or signal lost trigger in filter) |  | - Signal analysis (research)  - Debugging | Events have a timestamp and can optionally be connected to a sample(id) |
| Events level 2  - Start and end of tasks  - Clicks/events in tasks  - User actions in tasks or menu (e.g. next mole, or menu being shown) |  | - Research statistics on  usage  - Debugging |  |
| Events level 3  - Typing summary statistics |  | - Research statistics on  home typing (e.g. #  characters per minute) |  |
| Events level 4  - Tobii events (mainly when a enter is being  sent to Tobii) |  | - Research statistics on  home typing (e.g. #  characters per minute) |  |

\*\* file is in the format of <identifier>\_yyyy\_mm\_dd\_\_hh\_mm\_ss.<ext>

General remarks

* Initially the expert configures what is logged using the configuration parameters
* Some logging settings (enable/disable) should be controlled from the UNPMenu by the user on the fly. Since these changes happen on the fly, the filter is responsible for passing 0’s when a certain stream is disabled.
* versions should be stored (in parameter file)

To discuss

* Events in one file or separate files. Would propose separate because of possible privacy issues. Downside is that you might have to combine the event files for debugging.