Payroll Engine Console Application

👉 This application is part of the Payroll Engine.

The Payroll Console application provides API-like commands. See the Payroll Engine samples and tests for examples of how to use this tool. For a better understanding of the working concepts, it is recommended to read the Payroll Engine Whitepaper.

The application is controlled by two types of command line arguments:

- Command: a single command
- Command file: a file containing several commands.

Commands

The Payroll Console provides the following commands:

Command	Group	Description
`ActionReport`	Action	Report custom actions from an assembly
`CaseChangeExcelImport`	Case	Import payroll case change from Excel file
`CaseTest`	Case	Test case availability, build data and user input validation
`LogTrail`	Diagnostics	Trace the tenant log [1)]
`Stopwatch`	Diagnostics	Stopwatch based on environment user variable
`Write`	Diagnostics	Write to the console and/or log file

`HttpGet`
`HttpPost`
`HttpPut`
`HttpDelete`
	Http	Execute HTTP GET, POST, PUT or DELETE request

`PayrollExport`	Payroll	Export any payroll data to JSON file
`PayrollImport`	Payroll	Import any payroll data from JSON/zip file
`PayrollResults`	Payroll	Report payroll data to screen and/or file
`PayrunEmployeeTest`	Payrun	Execute employee payrun and test the results
`PayrunJobDelete`	Payrun	Delete payrun job with payroll results
`PayrunRebuild`	Payrun	Rebuild payrun
`PayrunStatistics`	Payrun	Display payrun statistics
`PayrunTest`	Payrun	Execute payrun and test the results
`RegulationExcelImport`	Regulation	Transfer Excel regulation to the file or backend

Command	Group	Description
`RegulationRebuild`	Regulation	Rebuild the regulation objects
`RegulationShare`	Regulation	Manage regulation shares
`DataReport`	Report	Report data to JSON file
`Report`	Report	Report to file [2]
`ReportTest`	Report	Test report output data
`ScriptExport`	Script	Export regulation scripts to folder
`ScriptPublish`	Script	Publish scripts from C# file
`TenantDelete`	Tenant	Delete tenant
`ChangePassword`	User	Change a user password
`UserVariable`	User	View and change the environment user variable
`Help`	App	Show the command reference

[1] Tenant logs are generated by the regulations and should not be confused with the application log.

[2] Based on FastReports.

The required command parameters and options/toggles are passed after the command name. An example of how to bulk import a payroll from a JSON file:

```
PayrollConsole PayrollImport MyPayroll.json /bulk
```

The `Help` command describes the commands and their parameters:

```
PayrollConsole Help PayrollImport
```

Command Files

The command file is a file with the extension `.pecmd` that contains a series of instructions to the Payroll Engine Console. The lines of the file are executed sequentially and contain the following line types

- Comment starting with `#`
- Command instruction
- Start another command file

Example command file `Delete.pecmd` with a comment and a command statement:

```
# delete my tenant
TenantDelete MyTenant /trydelete
```

The following example uses the `Delete.pecmd` command file and then runs the command to test the payrun:

```
# clranup and execute test
Delete.pecmd
PayrunTest *.pt.json
```

Command files access path

If the called command file is in a different folder, it will be activated before execution:

```
Test1/Test.pecmd
Test2/Test.pecmd
```

This is necessary if the called command file is to be defined with local references. The `keeppath` option prevents directories from being changed:

```
Tools/Import1.pecmd /keeppath
Tools/Import2.pecmd /keeppath
```

In the example above, both import command files are run in the startup folder.

Command file parameters

Command files can also be controlled by parameters. Such parameters are used with the placeholder `$Name$` in the commands. In the following example, the command file `OwnerTest.pecmd` has the parameter `Owner`:

```
# payrun employee test
# argument owner: job owner
PayrunEmployeeTest fileMask:*.et.json owner:$owner$ /wait
```

Calling this command file with the `Owner` parameter looks like this:

```
OwnerTest.pecmd owner:Test01
```

Command files in the operating system

Command file association in Windows:

1. Open File Explorer (right click Start -> File Explorer)
2. Find the file you want to associate

3. Right click the file and select `Properties`
4. In this window click `Opens With: Change...`
5. Select the payroll console program

Command file association in MacOS:

1. On your Mac, click the Finder icon in the Dock to open a Finder window.
2. Select a file, then choose `File` > `Get Info`.
3. You can also Control-click the file, then choose `Get Info`.
4. In the Info window, click the arrow next to `Open with`.
5. Click the pop-up menu, choose an app, then click `Change All`.

> Note: Don't click `Change All` if you only want the specific file you selected to open with that app.

Command file association in Linux:

1. TODO

Global Toggles

The following toggles apply to all Payroll Console commands:

Name	Description	Values	Command file
Display level	Command information level	`full` [1], `compact`, `silent`	Preset for executing commands
Error mode	Show failed tests and errors	`errors`[1], `noerrors`	Preset for executing commands
Wait mode	Wait at the program end	`waiterror`[1], `wait`, `nowait`	Final application wait mode
Path mode	Path change mode	`changepath`[1], `keeppath`	Only for command files

[1] Default value.

Application Extensions

The Payroll Console provides a plug-in mechanism for integrating custom commands:

- Command parameters with names and options
- Access to the Payroll REST API via the Payroll HTTP client
- Control output to console and logger
- Program exit code definition

To develop a Payroll Console extension, do the following

1. Develop the extension library in C# (example PayrollEngine.Client.Tutorials).
2. Copy the output DLL of the library into the `extensions` subfolder of the application.
3. Display the command help `PayrollEngine Help MyCommandName`.
4. Run the command with `PayrollEngine MyCommandName`.

Application Configuration

The Payroll Console configuration `appsetings.json` contains the following settings:

Setting	Description	Type	Default
`StartupCulture`	The payroll console process culture	string	System culture
`ApiSettings`	The backend api configuration	Backend API	
`Serilog`	Logger configuration	Serilog	

> It is recommended that you save the application settings within your local User Secrets.

Backend API Configuration

Setting	Description	Type	Default
`BaseUrl`	The backend base url	string	
`Port`	The backend url port	string	
`Timeout`	The backend request timeout	TimeSpan	100 seconds
`ApiKey`	The backend API key	string	

The backend API configuration can be declared in the following locations.

Priority	Source	Content
1.	Environment variable `PayrollApiConnection`	Backend API configuration connection string.
2.	Environment variable `PayrollApiConfiguration`	Path to the backend API configuration JSON file.
3.	File `apisettings.json`	Backend API configuration JSON file located in the program folder.
4.	File `appsettings.json`	`ApiSettings` from the application configuration JSON file.

API Configuration from connection string

The following is an example of a backend API configuration within a connection string:

```
BaseUrl=https://localhost; Port=44354; Timeout=02:46:40; ApiKey=MyApiKey;
```

API Configuration from JSON

The following is an example of a JSON-based backend API configuration:

```
{
    "BaseUrl": "https://localhost",
    "Port": 44354,
    "Timeout": "02:46:40",
    "ApiKey": "MyApiKey"
}
```

Api Key

If a key is required to access the backend API, it must be obtained from one of the following sources (in order of priority):

1. Environment variable `PayrollApiKey`
2. Payroll HTTP configuration (see `PayrollEngine.Client.Core`)

Application Logs

The payroll console stores its logs in the application folder `logs`.

Docker Support

Build the Docker image:

```
docker build -t payroll-console .
```

Run the Docker Payroll Console:

```
docker run -it --rm payroll-console Setup.pecmd
```

Delete the Docker image:

```
docker stop payroll-console
```

Solution projects

The.NET Core application consists of the following projects:

Name	Type	Description
`PayrollEngine.PayrollConsole.Commands`	Library	Console commands
`PayrollEngine.PayrollConsole`	Exe	Console application

Third party components

- Excel conversion with NPOI - license `Apache 2.0`
- Logging with Serilog - license `Apache 2.0`