Eloquence Webinar

What’s new in Eloquence B.08.20

- Part 2 -

New functionality

- Eloquence B.08.20 comes with a number of substantial enhancements in various product components

- Database full text search functionality

- Major language enhancements
- PCL to PDF conversion
- Improved WebDLG
- Improved JDLG

Language enhancements

- Syntax enhancements
- Support longer identifier names (up to 26 characters)
- Support PATH to replace MSI
- NLN (no line numbers) mode removes dependency on unique line numbers at runtime
- On demand loading of functions and subroutines
- Introducing support for class methods
- Support external class definitions in separate files
- On demand loading of classes and methods

Syntax enhancements

- Support longer identifier names (up to 26 characters)

```
Long_variable_name = 1
CALL Subroutine_with_long_name
```

Note: Using identifier names longer than 15 characters will make programs incompatible with previous Eloquence versions

Syntax enhancements

- Support member variables in IF .. THEN implied LET

```
IF .. THEN Obj.Member=1
```

- Support STRUCT assignment in IF .. THEN implied LET

```
IF .. THEN STRUCT A=B
```

Syntax enhancements

- Support member variables with MAT statement

```
MAT Obj.Member=ZER
MAT Obj.Member=(1)
MAT Obj.A=Obj.B+(4711)
```

  Note: Using a member variable with MAT will make the program incompatible with previous Eloquence versions.

- Support member variables with SUM, ROW and COL array functions

```
PRINT SUM(V.R)
```

Syntax enhancements

- LIST SUB/FN [TO END]

```
LIST FNDate$
LIST SUB Example TO END
LIST SUB Class:Method
```

Syntax enhancements

- Improved SPACE DEPENDENT mode
  - SPACE INDEPENDENT mode is tried first
  - Improved handling of conflicting syntax
- Support inserting program segments

Syntax enhancements

- The SETENV statement allows to set, change or undefine environment variables from an application.

```
SETENV "Name","Value"
SETENV "Name‟
GETENV$("Name")
```

- Environment variable names are case sensitive and limited to a max. of 255 characters.
- When a value argument is present, the environment variable is added or changed to the specified value. When a value a argument is not present the environment variable is deleted.
- Note: It is currently undefined behavior (platform specific) if any changed environment variables are visible to sub processes (eg. executed with COMMAND).

PATH

- If a PATH is defined it is used instead of the default volume label (MSI) to locate files

```
LOAD "Prog"
GET FORM "Form"
```

- A PATH consists of one or more elements
- Elements are separated by a colon (on HP-UX and Linux) or a semicolon (on Windows)

```
.::/opt/eloquence/8.2/share/prog
```

PATH consisting of the local directory, the current MSI volume and an absolute path

Defining a PATH

- Config file

```
PATH ".:/opt/eloquence/8.2/share/prog"
```

- EQPATH environment variable

```
export EQPATH=.:/opt/eloquence/8.2/share/prog
```

- PATH statement

```
PATH ".:/opt/eloquence/8.2/share/prog“
PATH GETENV$("HOME")&":"&PATH$
```

PATH elements

- An absolute directory starts with a slash (or with a backslash, a drive letter a colon and a backslash for Windows)

```
/opt/eloquence/8.2/share/prog
C:\program files\eloquence\8.2\share\prog
```

- A relative directory (relative to the current directory) is specified with a leading dot

```
./test
```

PATH elements

- A volume label may be used to reference the path specified in the volume definition

  ```
  LABEL
  ```

- A relative path and a volume label (separated by comma)

  ```
  test,LABEL
  ```

- Anything else is considered a directory relative to the (current) MSI volume label. An empty element refers to the current MSI volume

  ```
  test
  ```

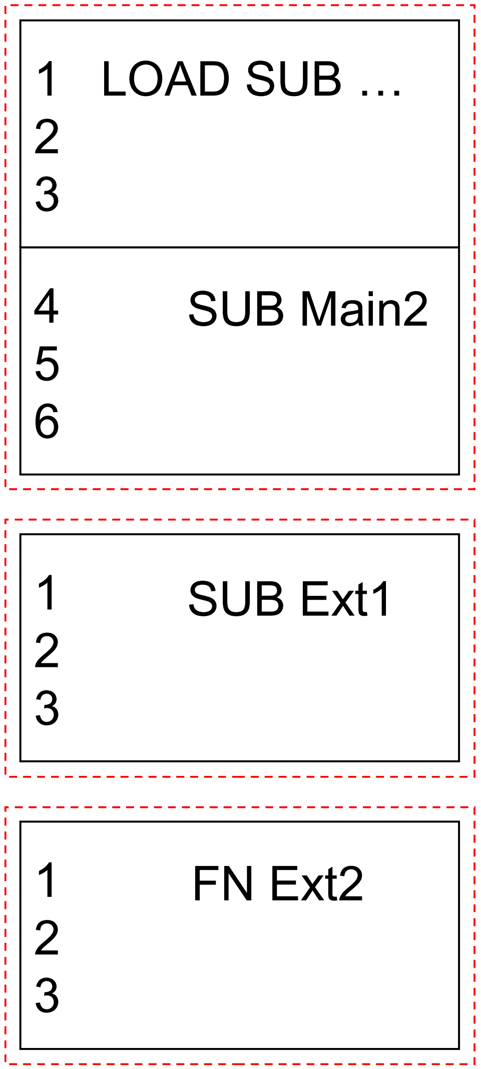
NLN mode

- If NLN (no line numbers) mode is enabled a LOAD SUB will no longer renumber program code, retaining its original line numbers
- As a consequence line numbers are no longer unique across program segments when executing an application
- Line numbers continue to be unique per program segment and in program files on disk
- With few exceptions Eloquence does not depend on unique line numbers at runtime

NLN mode

- NLN mode mostly affects debugging using the text mode development environment
- When the program is not running line numbers are unique and behavior is unchanged

NLN mode



- When the program is running line numbers are context specific
- NLN segments are hidden unless in use (and removed once the program stops)
- When executing in a non NLN segment lines in a NLN segment can't be modified
- When executing in a NLN segment lines outside this segment can't be modified, any line numbers refer to this program segment

Configuring NLN mode

- The Eloquence NLN mode may be configured in the config file

```
OPTION NLN 1
```

- NLN mode is disabled by default to ensure full backwards compatibility

Automatic code management

- Program code may be loaded automatically on first use and is removed when no longer referenced
- Autoloaded program code uses NLN mode (regardless if its configured)

- Autoloading relies on program file naming conventions
  - SUB Test -> Test.PROG
  - FNTest -> FNTest.PROG
  - FNTest$ -> FNTest$.PROG

- Unused program code is cached temporarily (for performance reasons) and removed from memory when the cache size reaches an internal limit

Automatic code management

- A missing function or subroutine is automatically loaded, so a

  ```
  CALL Test
  ```

  will load a file Test.PROG

- A custom handler subroutine may be specified to override the default behavior

  ```
  ON UNDEFINED CALL Handler
  SUB Handler(Name$)
      LOAD SUB Name$
  END SUB
  ```

Configuring autoload

- The Eloquence autoload may be configured in the config file

```
OPTION AUTOLOAD 1
```

- Autoload is enabled by default. It may be disabled to ensure full backwards compatibility

DEMO

- Autoload code
- ON UNDEFINED

Classes and methods

- Functions or subroutines specific to a class are called methods
- A method is called with an implied reference to the object
- Member variables and methods are accessible using the THIS keyword
- Base class methods are accessible using the SUPER keyword

```
SUB Class:Method [ (arguments) ]
DEF FN Class:Method [ (arguments) ]
DEF FN Class:Method$ [ (arguments) ]

CALL A.Print
CALL SUPER.Print
```

External Class definitions

- A Classes (and its methods) may be defined in a separate program file
- Class definitions may be loaded explicitly with the LOAD CLASS statement
- When enabled missing class definitions may be loaded automatically

```
CLSPATH “/path/to/classes1:/path/to/classes2”
DIM V:Tname

LOAD CLASS “…”
DEL CLASS Tname
```

Configuring class autoload

- The Eloquence class autoload may be configured in the config file

```
OPTION CLSLOAD 1
```

- Class autoload is enabled by default. It may be disabled to ensure full backwards compatibility

DEMO

- Autoload class with methods

Relative access to class members

- The WITH / END WITH keywords define a context
- A leading dot in a variable (or method) name indicates access relative to this context
- The default context in a class method is the current object

```
PRINT Obj.Name$
CALL Obj.Print


WITH Obj
   PRINT .Name$
   CALL .Print
END WITH
```

Return object from function

- A function may return an object

```
DIM STRUCT A

STRUCT A=FNX
PRINT TYPEOF(A)
END

DEF FNX

   TYPE T
      INTEGER I
   END TYPE

   NEW V:T
   V.I=1234
   RETURN STRUCT V

FN END
```

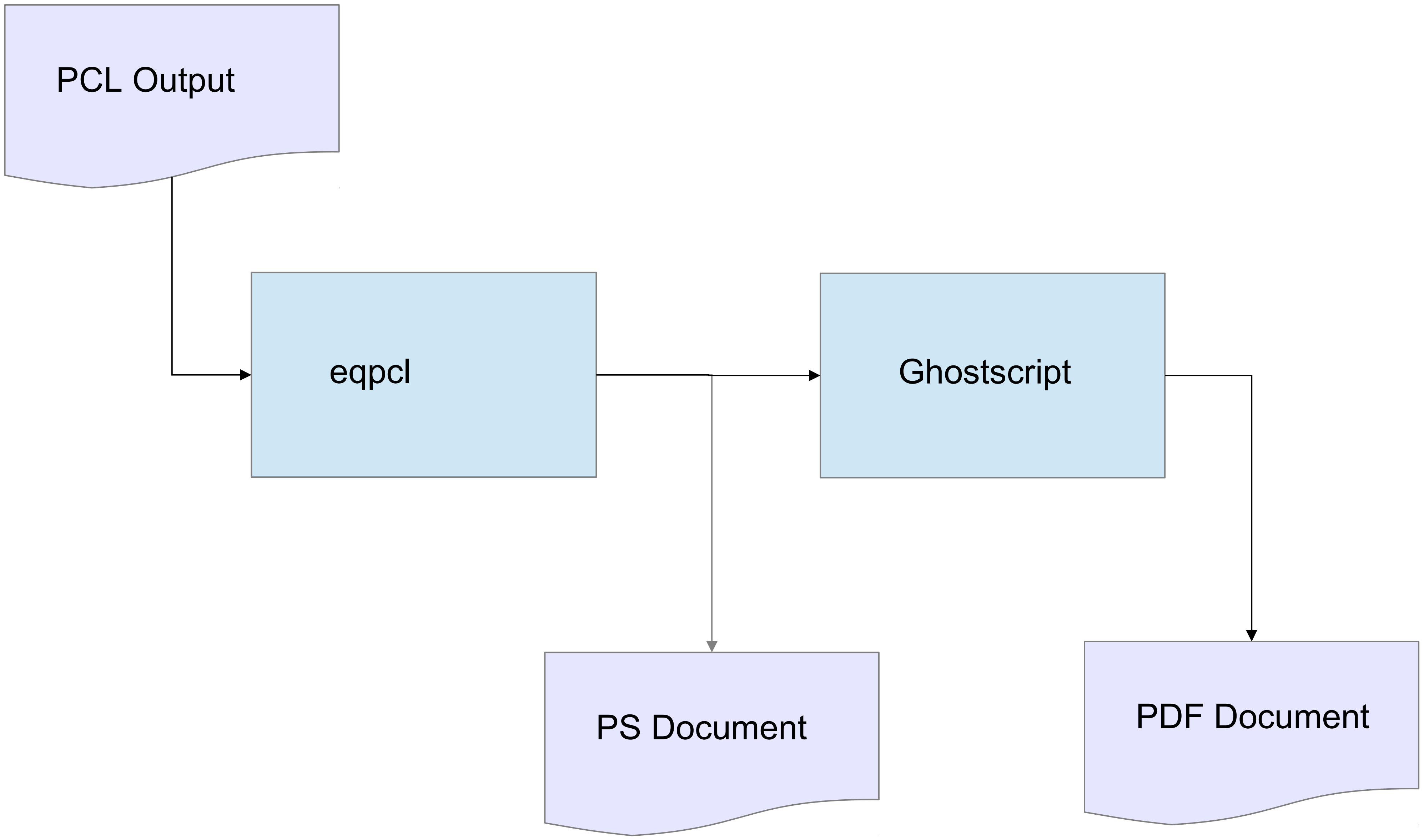
eqpcl utility

PDF Support

eqpcl utility

- New eqpcl utility adds PDF and Postscript support
- Converts PCL output to PDF or Postscript
- Implements a commonly used subset of PCL functionality and is expected to work with a majority of existing applications
- Unsupported PCL functions are silently ignored
- PCL command sequences and text are translated into Postscript and uses GNU Ghostscript to create the PDF output

eqpcl utility



eqpcl utility (limitations)

- PCL functions are <u>not</u> supported:
  - user defined fonts
  - user defined symbol sets
  - user defined patterns
  - area fill with patterns
  - HP GL/2
  - raster graphics (other than area fill)
- Limited support
  - limited fonts supported
  - limited symbol sets supported

eqpcl utility (enhancements)

- virtual trays to define paper format and forms
  - Define forms and letter heads in the configuration
  - No need to use PCL macros
  - Easily enhance any document

- various barcode types supported

DEMO

- Creating a PDF document from PCL application
- Using a page overlay
- Using barcode

More information

Detailed information is available on the

Eloquence web site

http://eloquence.marxmeier.com

Get in contact

info@marxmeier.com