



**Idris Samawi Hamid**  
**Luigi Scarso**

# **Notepad++**

**FOR CONTEXT MKIV**

# Table of Contents

1	Background .....	2
1.1	Motivation .....	2
1.2	History .....	2
2	Introduction to NOTEPAD++ .....	3
2.1	Features .....	3
2.2	NOTEPAD++ and SCITE .....	3
2.3	Lexers and Plugins .....	4
2.4	Installing NOTEPAD++ .....	5
3	The NOTEPAD++ for CONTEXT Package .....	5
3.1	Components .....	5
3.2	Installation .....	8
4	Highlighting and Themes .....	8
4.1	Solarized++: Screen Contrast and Color Scheme .....	8
4.2	On Syntax and Semantic Highlighting .....	8
4.3	Silver Twilight Hi and Silver Twilight Lo .....	11
4.4	ALM Fixed .....	13
5	The NPPEXEC Console .....	13
5.1	NPPEXEC and Console Scripts .....	13
5.2	NPPEXEC and CONTEXT .....	14
5.3	NPPEXEC and the Macro Submenu .....	17
5.4	Configuring Shortcut Mapper .....	18
6	The CONTEXT Lexer .....	18
6.1	Components of the Lexer .....	18
6.2	Configuring Keyword Classes: Style Configurator and <code>context.xml</code> .....	18
6.3	Configuring Autocompletion and Tooltips: <code>ConTeXt.xml</code> .....	18
6.4	Configuring Macros, Tags, and Shortcuts: <code>ConTeXt.ini</code> .....	18
6.5	Configuring the Right-Click Menu .....	18
6.6	Usage .....	18
6.7	Note on bidirectional editing .....	18
7	References .....	18

# 1 Background

## 1.1 Motivation

A continuing desideratum for `CONTEXT` is a user-friendly writing and editing environment, where the range of application of the category “user-friendly” especially includes non-experts in programming or software development. The lack of such an environment is one factor that inhibits the wider use of `CONTEXT`. Despite its incredible power and precision, at present it is not generally feasible for instructors and researchers in, e.g., the humanities to assign the use of `CONTEXT` to students, or to use it to collaborate on projects.

The first author of this manual, Idris Samawi Hamid, is a professor who has felt the acuteness of this lacuna. In the course of an ongoing effort to address it, in 2017 a project to develop a set of utilities for the Windows editor Notepad++, including a dedicated `CONTEXT` lexer plugin, was launched. The software development plan was developed and supervised by Hamid, as well as the color-scheme and themes. The initial C++ code and Python scripts were written by Jason Wu (a research assistant at Colorado State University); currently the code and scripts are written by and maintained with coauthor Luigi Scarso. This manual documents a major release of that project: For the moment we call the project, simply, **Notepad++ for `CONTEXT` MkIV**.

## 1.2 History

Prior to his move to `CONTEXT`, Hamid was using the shareware editor WinEdt. At that time WinEdt was (and probably still is) a very polished environment for writing and processing documents written in `TEX`. However, configuring WinEdt for `CONTEXT` was critically impeded, due in major part to the fact that much of its graphical user interface was hardcoded for a certain famous document preparation system. Around the same time, lexers and tools were being developed for SciTE, which eventually became the standard text-editor for `CONTEXT`. Despite its `CONTEXT`-friendly tools, Hamid continued to miss many of the configuration and interface options of WinEdt that made editing and processing `TEX` documents so efficient and user-friendly for non-programmers. After trying virtually every available option – explicitly `TEX`-friendly or other – he finally settled upon Notepad++. Its look, feel, and extensive configuration options allowed Hamid to quickly achieve a setup analogous to WinEdt. A few characteristics of WinEdt were still missed; on the other hand, Notepad++ brought to the table other features missing in WinEdt; these made the transition worth it. For example, Notepad++ supports global bidirectional text editing essential for the Arabic script – WinEdt had no such support).

Eventually, over a decade ago, a basic package for Notepad++ was released to the `CONTEXT` community by Hamid. It consisted of a number of configuration files, including, among other things,

- a UDL (User-Defined Language) file for code highlighting of different classes of `TEX`-commands and other keywords;
- an autocompletion “API”; and
- some console scripts, many of which appear under the menu item “Macros”. These provided, among other things, a functionality largely identical to that provided by the corresponding SciTE scripts for `CONTEXT` – found under the menu item “Tools”.<sup>1</sup>

Although remarkably versatile, the UDL system was still too restrictive. Other Notepad++ mechanisms, such as auto-completion of control sequences, were not designed with `TEX`-type languages

---

<sup>1</sup> That package, now obsolete, is remains available here: [http://wiki.contextgarden.net/File:Npp\\_ConTeXt-Uni.zip](http://wiki.contextgarden.net/File:Npp_ConTeXt-Uni.zip)

in mind, resulting in certain limitations or annoyances. Among other issues: As `CONTEXT MkIV` has continued to develop in the direction of a pure markup language, its syntax has

- become considerably more verbose; and
- demanded a mechanism for easy tagging of text with, e.g., braces or a set of `\start|stop` commands.

Mere auto-completion of commands was no longer sufficient for efficient content writing and editing. Fortunately `WebEdit`, a `Notepad++` plugin designed for XML-type tagging and related function completion, came to the rescue. Unfortunately it also had certain limitations which inhibited a fully satisfactory solution.

In the wake of these and other limitations: What we needed was a dedicated `CONTEXT` lexer and plugin to assist content writing and editing. In combination with other mechanisms and plugins, the result would be a complete `Notepad++` system for writing, editing, and processing `CONTEXT` documents. Hence **Notepad++ for `CONTEXT MkIV`**.

## 2 Introduction to **NOTEPAD++**

### 2.1 Features

Developed by Don Ho, `Notepad++` is a very popular text editor for the Windows platform. Although geared towards programmers and web designers, it has a number of features that make it exceptionally appropriate for non-programmers. `Notepad++` features, among other things

- A user-friendly configuration system, via graphical dialogs and settings saved to editable XML files;
- both multiple and single-document splitting;
- translation of its display interface into multiple languages;
- the toolset `TextFX`, which provides a plethora of functions that would normally involve writing scripts on the part of the user;
- a plugin system and a vast catalogue of over 100 available plugins which immensely extend the capabilities of `Notepad++` in a user-friendly manner.
- the User-Defined Language system, which allows the user to easily define folding rules and syntax highlighting for a coding language that does not already come with `Notepad++`. It is especially useful for simple scripting languages or text-file formats.<sup>2</sup>

### 2.2 **NOTEPAD++** and **SCITE**

As mentioned earlier, `CONTEXT` already comes with `SciTE`. Both `SciTE` and `Notepad++` are based on the text-editing component `Scintilla`. Thus a user switching between the two editors can expect a similar typing experience. A fundamental difference between the is that `Notepad++`'s preferences, thematic styles, and shortcuts are all extensively configurable via a system of menus and dialogs, whose style is mostly common to mainstream programs that use a GUI. For non-programmers and the like, this is more comfortable than, e.g., editing the `.properties` files used by `SciTE`.

One of the most important features of `Notepad++` is its support for global bidirectional editing. Some background: Unfortunately `Scintilla` never implemented bidirectional editing, and the developer of

---

<sup>2</sup> For example, one may edit tables in an OpenType font editor, then save those tables to a text file with an associated syntax. One may then choose to work with the text file instead of the Graphical User Interface (GUI).

Scintilla apparently has little interest in pursuing it. Visually, basic mixed right-to-left (RTL) and left-to-right (LTR) text *may* look normal, but selection of text whose direction is opposite to that of the global direction of the editor will *generally* not copy and paste correctly. For SciTE the global direction is, naturally, LTR; hence RTL will *generally* not copy and paste correctly.<sup>3</sup> Notepad++ provides a mechanism that mirrors Scintilla behavior so that it can be used for RTL editing, except that LTR will now generally not copy and paste correctly. So for proper RTL or LTR editing one must switch the global direction to match the immediately desired editing direction.

SciTE in `CONTEXT` features a set of commonly used scripts that may be found under the Tools menu. In Notepad++ for `CONTEXT` a similar set of tools – with identical shortcuts as much as possible – may be found under the Macros menu.

The core of Notepad++ is explicitly designed for speed. On Windows Notepad++ generally starts up fast, even faster than SciTE. A few plugins will slow Notepad++ down, however.

## 2.3 Lexers and Plugins

Notepad++ ships with highlighting and theme support (*internal lexers*) for over 50 code languages, and the UDL system allows the user to easily configure and add more. For maximum flexibility and control, Notepad++ also supports *external lexers*, development of which requires some C++ programming skill: This will appear under the Language menu and in the associated dialogs. An external lexer can add support for a previously unsupported language, or it can be used to provide an alternative to a currently supported language. For example, one can use the Lua highlighting that comes with Notepad++, or one can download the external lexer Gmod Lua, then configure that to be the default lexer for the Lua language. An external lexer can also be augmented by other features, which will then appear under the Plugins Menu.

For use as a complete environment for writing and editing documents, a number of plugins complement the Notepad++ for `CONTEXT` system. The following are highly recommended:

- **NppExec**  
This is the console, and is an integral component of Notepad++ for `CONTEXT`. Although one can have Notepad++ launch the command prompt or other console of one's choosing, NppExec is also needed to show a set of select scripts under the Macros menu. A standard installation gives the option of installing the console.
- **Explorer**  
Notepad++ can launch the normal Windows Explorer. But there is also the Explorer plugin which can be docked inside of the editor or detached; it has some useful features such as a filter which allows one to view only files of a selected type.
- **DSpellCheck**  
This spell checker works well, although it could be improved. Currently it doesn't make exceptions for words that begin with a backslash; this means that most `TEX` are treated as misspelled. We hope to have this fixed in the short term.
- **Compare**  
This is a plugin for comparing files; it launches a double-pane view and a dockable applet.
- **XBrackets Lite**  
This plugin provides automatic completion of different types of brackets and is configurable. Notepad++ comes with some facility for bracket control, but XBrackets Lite is more useful.

---

<sup>3</sup> The use of 'may' and 'generally' are meant to indicate that there are some important subtleties: See Section 6.7.

- **Plugin Manager**

This plugin maintains a list of i) all registered plugins, ii) installed plugins, iii) installed plugins for which updates are available. One can choose to install, update, or delete any given plugin as desired.

In addition to the recommended set above, there are many other plugins available, e.g., NppDocShare for collaborative editing, MarkdownViewer++ for previewing markdown output, and XMLTools. With a little research and some tweaking, it is not hard to turn Notepad++ into a development environment to suit most of one's needs.

## 2.4 Installing NOTEPAD++

# 3 The NOTEPAD++ for CONTeXt Package

## 3.1 Components

Notepad++ for ConT<sub>E</sub>Xt is organized as follows:

```

/Npp-for-ConTeXt/Program Files
/Npp-for-ConTeXt/Roaming
/Npp-for-ConTeXt/scripts

/Npp-for-ConTeXt/Program Files/Notepad++

/Npp-for-ConTeXt/Program Files/Notepad++/plugins
/Npp-for-ConTeXt/Program Files/Notepad++/plugins/ConTeXt.dll

/Npp-for-ConTeXt/Program Files/Notepad++/plugins/APIs
/Npp-for-ConTeXt/Program Files/Notepad++/plugins/APIs/context.xml

/Npp-for-ConTeXt/Program Files/Notepad++/plugins/Config
/Npp-for-ConTeXt/Program Files/Notepad++/plugins/Config/ConTeXt.xml

/Npp-for-ConTeXt/Roaming/Notepad++
/Npp-for-ConTeXt/Roaming/Notepad++/contextMenu.xml
/Npp-for-ConTeXt/Roaming/Notepad++/shortcuts.xml
/Npp-for-ConTeXt/Roaming/Notepad++/userDefineLang.xml
/Npp-for-ConTeXt/Roaming/Notepad++/stylers.xml

/Npp-for-ConTeXt/Roaming/Notepad++/plugins

/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/ConTeXt.ini
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/NppExec.ini
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/npes_saved.txt

/Npp-for-ConTeXt/Roaming/Notepad++/plugins/doc
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/doc/context/npp-context-manual.pdf
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/doc/context/npp-context-manual.tex

```

```

/Npp-for-ConTeXt/Roaming/Notepad++/plugins/themes
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/themes/Silver Twilight Hi.xml
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/themes/Silver Twilight Lo.xml

/Npp-for-ConTeXt/scripts/command_primitives_api_new.py
/Npp-for-ConTeXt/scripts/update-ConTeXt.py

```

Following is a brief description of each component of this system:

### 1. **CONTEX<sub>T</sub> Lexer and Plugin**

`ConTeXt.dll` is the heart of the system. It manages the classes specified for content highlighting, autocompletion, and tooltips, as well as the content-markup and templates system.

```

/Npp-for-ConTeXt/Program Files/Notepad++/plugins/ConTeXt.dll

```

### 2. **Initialize Plugin**

`ConTeXt.ini` allows the user to add, remove, configure, and organize commands for content markup into menus and submenus, as well as to specify shortcuts that can be replaced by templates in running text.

```

/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/ConTeXt.ini

```

### 3. **Right-Click Menu**

Notepad++ features a right-click menu mechanism, whose settings are managed via the configuration file `contextMenu.xml`. The full set of markup menus in the plugin can be added to this file, then edited manually as desired. Note that, despite appearances, the name `contextMenu.xml` has nothing to do with CONTEX<sub>T</sub>; it is native to Notepad++.

```

/Npp-for-ConTeXt/Roaming/Notepad++/contextMenu.xml

```

### 4. **Autocompletion API**

The so-called “API” `context.xml` features (what aims to be) a complete list of official CONTEX<sub>T</sub> commands, organized alphabetically for autocompletion purposes.<sup>4</sup> For a subset of this list, each is also tagged with information about usage; when typed and followed by a left bracket ‘[’, this information will appear as a *tooltip* (also called a *calltip*).

```

/Npp-for-ConTeXt/Program Files/Notepad++/plugins/APIs/context.xml

```

### 5. **Content-Highlighting Classes**

`ConTeXt.xml` includes the same list of official CONTEX<sub>T</sub> commands, this time organized into semantic *classes*. These and other classes are configured for content highlighting through Notepad++’s Style Configurator.

```

/Npp-for-ConTeXt/Program Files/Notepad++/plugins/Config/ConTeXt.xml

```

### 6. **Highlighting: Silver Twilight High and Silver Twilight Lo**

Two general themes for content highlighting have been developed especially for this project: the first and default theme is light, the second dark. Each may be accessed and tweaked via Style Configurator, or copied to a new name and modified to make a new theme. See Section 4.3.

<sup>4</sup> The list of CONTEX<sub>T</sub> commands is currently generated from the CONTEX<sub>T</sub> sources by a Python script; see below. There is still a small residue of commands that are missed in the sources for the list, and thus by the script as well. We hope to see that gap closed in the near future.

Silver Twilight themes apply to one degree or other throughout the default languages that come with Notepad++ (there remains some work to do in that respect).

```
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/themes/Silver Twilight Hi.xml
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/themes/Silver Twilight Lo.xml
```

The file `stylers.xml` is optional: It is identical to Silver Twilight Hi, and is a starting point for the user to make one's own changes to the theme. This file will appear in Style Configurator labeled `Default (stylers.xml)`.

```
/Npp-for-ConTeXt/Roaming/Notepad++/stylers.xml
```

## 7. NppExec Scripts

A number of scripts commonly used for CON<sub>TEXT</sub> productivity are saved in `npes_saved.txt`. Normally one configures these through the dialog that appears when the console is executed (by typing F6).

```
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/npes_saved.txt
```

## 8. Initialize NppExec and Configure Macro Menu

Default settings for the appearance of NppExec, consistent with the Silver Twilight themes, are saved in `NppExec.ini`. This file also maintains a list of console scripts that are to appear under the Macro menu; this is normally edited via the NppExec Advanced Options dialog.

```
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/config/NppExec.ini
```

## 9. Users Manual

The user's manual (this document) and its source are named, respectively, `npp-context-manual.pdf` and `npp-context-manual.tex`.

```
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/doc/context/npp-context-manual.pdf
/Npp-for-ConTeXt/Roaming/Notepad++/plugins/doc/context/npp-context-manual.tex
```

## 10. Shortcuts

Most menu commands can be assigned a keyboard shortcut, and each shortcut is configurable. A basic system of shortcuts, consistent across a number of recommended or useful plugins, is provided by `shortcuts.xml`.

Notepad++ has a `Run...` command that allows the user to execute a script that will call an external programs; that script can be saved. Saved scripts appear under the `Run` menu; these are also saved in `shortcuts.xml`. The user will almost certainly want to edit the `Run` menu at some point.

```
/Npp-for-ConTeXt/Roaming/Notepad++/shortcuts.xml
```

## 11. Python Scripts

New versions of CON<sub>TEXT</sub> are released often, and the addition of new commands is not uncommon. For those who update often: The lists of official commands in `ConTeXt.xml` and `context.xml` are generated from the sources via the Python script `command_primitives_api_new.py`; `update-ConTeXt.py` makes sure that local changes to the `ConTeXt.xml` configuration are saved and not overridden.

```
/Npp-for-ConTeXt/scripts/command_primitives_api_new.py
/Npp-for-ConTeXt/scripts/update-ConTeXt.py
```

## 12. Bib<sub>T</sub><sub>E</sub><sub>X</sub>

Finally, there is a UDL (user-defined language) file configured for content highlighting of `.bib` files; it is consistent with the Silver Twilight themes. This file may be considered optional. Any additional UDL's defined or imported by the user will also be saved to the file `userDefineLang.xml`.



/Npp-for-ConTeXt/Roaming/Notepad++/userDefineLang.xml

## 3.2 Installation

# 4 Highlighting and Themes

## 4.1 Solarized++: Screen Contrast and Color Scheme

Writing and editing content via a digital display for many hours on end can cause severe strain on the eye. One way to ameliorate this is to use a comfortable color scheme for one's editor. The individual colors provide the building blocks for themes and for distinguishing the various types of written content involved in one's editing.

Color-scheme preferences will naturally differ from person to person to one degree or other. However, a couple of general rules appear to stand out:

- Maintain a *medium-to-high* balance of contrast between text and background color; i.e., strong contrast, but not too high.
- Choose *soft* colors for text; not too bright, not too dim.

One of the most thought out and successful color schemes is Solarized, by Ethan Schoonover.<sup>5</sup> It features two series: a series of eight *background tones* and another series of eight *accent colors*. As excellent as it is, the first author found the background tones to exude something of a murky and “swampy” aesthetic. The light theme is too bright for continuous full-screen use (see Section 4.3). The content colors are more successful: They are both soft and distinct, although Solarized green contains perhaps too much yellow.

In Notepad++ for CONTEX<sub>T</sub> the first author has developed a modification of the Solarized; our resultant color scheme is called, perhaps appropriately, **Solarized++**. There are nine background tones and ten accent colors. The background colors are entirely different from the original Solarized. The accent colors are largely the same. However, Solarized green has been replaced with Solarized++ green, Solarized green has become Solarized++ yellowgreen, and an additional color, Solarized++ maroon, has been added. See Figure 1.




















In addition, Solarized++ currently features a series of five supplementary *anti-base* tones for purposes of contrast when needed. As the name suggests, these five are meant to complement the base tones; see Figure 2.

## 4.2 On Syntax and Semantic Highlighting






Syntax highlighting has been shown to have a positive impact on the comprehension of computer programs.<sup>6</sup> In the experience of the authors, the same is true for highlighting of structural and stylistic markup in CONTEX<sub>T</sub>. There is a (perhaps pedantic) difference: Although the *basic* CONTEX<sub>T</sub> interface is expressed in terms of control sequences that take the form of T<sub>E</sub>X commands, T<sub>E</sub>X per se closely exemplifies the paradigm of a *programming* language in a strict sense; whereas CONTEX<sub>T</sub> has developed

<sup>5</sup> See <http://ethanschoonover.com/solarized>.

<sup>6</sup> See, e.g., Sarkar (2015).

Name	Hex	Sample	Name	Hex	Sample
Base Tones			Accent Colors		
Yellow	#B58900		Base04	#1E2D2E	
Orange	#CB4B16		Base03	#324140	
Red	#DC322F		Base02	#475652	
Magenta	#D33682		Base01	#5C6B64	
Violet	#6C71C4		Base0	#718076	
Blue	#268BD2		Base1	#899589	
Cyan	#2AA198		Base2	#A2AA9D	
Green	#399900		Base3	#BABFB1	
Maroon	#A12A33		Base4	#D3D5C5	
Yellowgreen	#859900			#	

**Figure 1** Solarized++: Base Tones and Accent Colors

Anti-Base Tones		
Antibase0	#73606D	
Antibase1	#897781	
Antibase2	#9F8E96	
Antibase3	#B5A5AA	
Antibase4	#CCBDBF	

**Figure 2** Solarized++:  
Anti-Base Tones

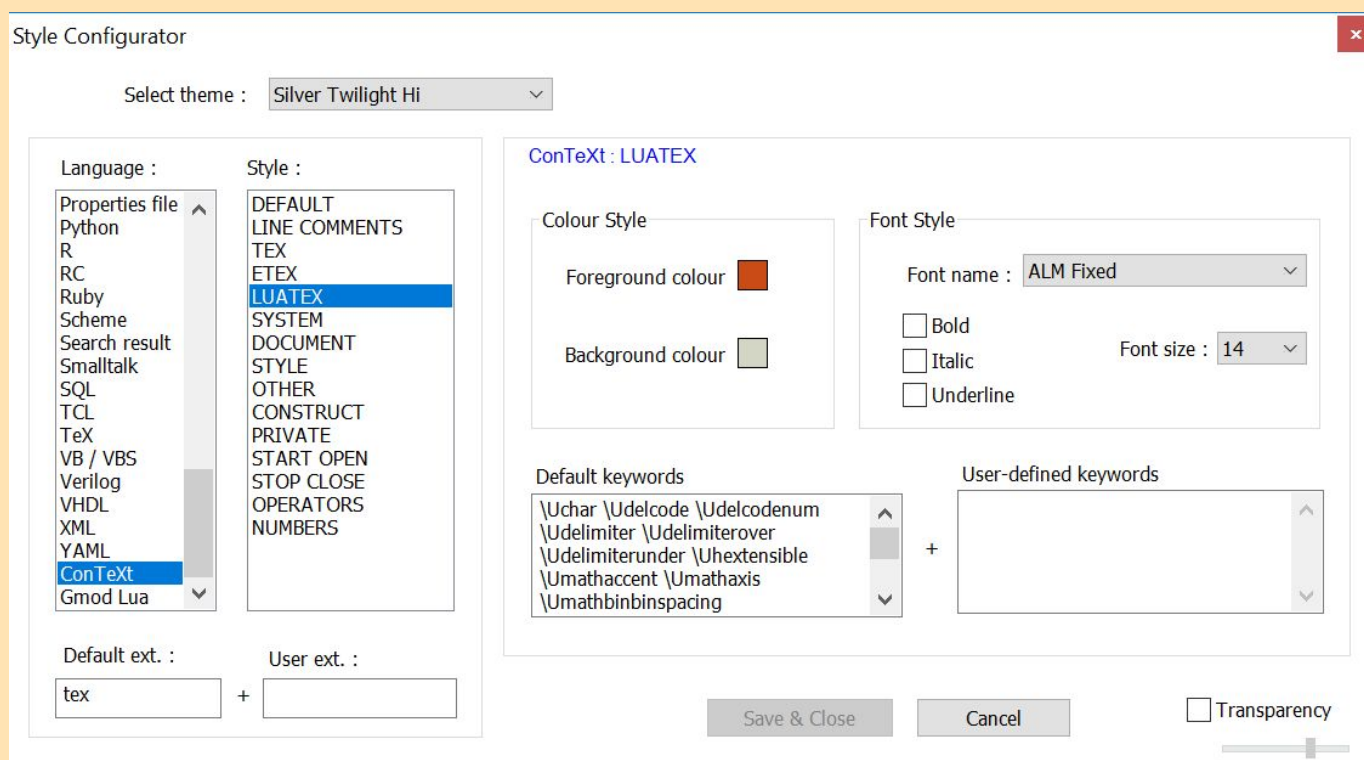
towards exemplifying the paradigm of a *markup* language. Technically speaking, even if one writes a basic `CONTEXT` document with pure markup and no deeper commands, one still has to run that document through a compiler which will interpret the input and convert it to some output, normally a PDF document. We might describe the basic `CONTEXT` interface as a hybrid: markup language in appearance and programming language in reality.

Markup is focused more on meaning, i.e., *semantics*, and less on grammar, i.e., *syntax*. Programming involves syntax to a high degree, and also semantics. Because syntax is often subtle and slippery to the programmer, code highlighting for programming languages generally takes the form of *syntax highlighting*, so much so that ‘code highlighting’ and ‘syntax highlighting’ are often treated as synonymous. In recent years, some coders have begun to emphasize a distinction between syntax highlighting

and *semantic highlighting*.<sup>7</sup> Because the interpreting of structural and stylistic markup pertains much more to matters of meaning than to grammar, highlighting of CON<sub>T</sub>EXT code is best contextualized in terms of semantic highlighting. Of course, there is syntax to CON<sub>T</sub>EXT as well: The different mechanisms between the earlier Table and the now standard TABLE environments (for typesetting of tabular data) exhibit stark differences in syntax. Considering possible models and implementations of code highlighting specific to the clarification of CON<sub>T</sub>EXT syntax is a matter for future research.

Settings for semantic highlighting of CON<sub>T</sub>EXT keywords in Notepad++ are saved in the configuration file ConTeXt.xml, mentioned earlier. In particular, there are 14 classes of keywords; members of each class are given a specific color; these may be viewed (and edited) in Style Configurator, under Language: ConTeXt.<sup>8</sup>

Following is a brief description of each keyword class supported in the CON<sub>T</sub>EXT lexer. See also Figure 3.



**Figure 3** CON<sub>T</sub>EXT Lexer and Notepad++ Style Configurator

### 1. **DEFAULT**

This is the default keyword class, applied to strings which involve no other semantics. Normal text will generally belong to the default class. As default, there are no keywords specified for this class.

### 2. **LINE COMMENTS**

This class includes the percent sign and all text on the same line that comes after it. Of general scope, there are no keywords specified for this class.

### 3. **TEX ETEX**

Primitive commands of T<sub>E</sub>X and  $\epsilon$ -T<sub>E</sub>X are treated as one class.

*Allows user-defined keywords:* **No**

<sup>7</sup> For detailed discussion of the distinction between syntax and semantic highlighting, see

<https://visualstudiomagazine.com/Articles/2014/08/01/Semantic-Code-Highlighting.aspx>; and

<https://zwabel.wordpress.com/2009/01/08/c-ide-evolution-from-syntax-highlighting-to-semantic-highlighting/>.

<sup>8</sup> Because the CON<sub>T</sub>EXT language comes in the form of a lexer plugin, it will generally appear near the bottom of the language list on the left side of the dialog, after the natively supported languages, and along with other lexer plugins, if installed. Some classes allow the user to add one's own keywords to the class as well

4. **LUATEX**

LuaTeX has its own class. Although not often, new primitives can appear, and LuaTeXperts can define their own.

*Allows user-defined keywords: Yes*

5. **SYSTEM**

This is an official CONTEXT keyword class. It includes system-level commands, those which are not meant for general typesetting and which the average user will never see.

*Allows user-defined keywords: Yes*

6. **DOCUMENT**

This is an official CONTEXT keyword class. It includes commands that are generally meant to *produce* a stream of text within a document.

*Allows user-defined keywords: Yes*

7. **STYLE**

This is an official CONTEXT keyword class. It includes commands that are generally meant to *style* a stream of text within a document.

*Allows user-defined keywords: Yes*

8. **CHARDEF (formerly OTHER)**

This is an official CONTEXT keyword class. It consists of commands that translate to certain Unicode characters that are needed but normally inconvenient to typeset directly.

*Allows user-defined keywords: Yes*

9. **CONSTRUCT**

This class includes keywords used to constitute prefixes to other keywords, such as `\place` and `\set`. The prefix and any immediately following string connected to that prefix is treated as a keyword. Words in other classes that already contain one of these prefixes are not effected.

*Allows user-defined keywords: Yes*

10. **PRIVATE**

These are for keywords defined by the user. A few highlight commands are given for illustration, and the user can add more.

*Allows user-defined keywords: Yes*

11. **START OPEN**

These are opening commands that begin a folding environment; each must have an associated closing keyword in the STOP CLOSE class. A small symbol will appear in the margin next to the opening keyword, with a bright line leading to the closing symbol.

*Allows user-defined keywords: Yes*

12. **STOP CLOSE**

These are closing commands that end a folding environment; each must have an associated opening keyword in the START OPEN class.

*Allows user-defined keywords: Yes*

13. **OPERATORS**

This class includes punctuation and related symbols.

*Allows user-defined keywords: No*

14. **NUMBERS**













This includes numerals and related symbols.

*Allows user-defined keywords: No*

## 4.3 Silver Twilight Hi and Silver Twilight Lo

The Solarized++ color scheme and lexer keyword classes for semantic highlighting together constitute the components which go into Silver Twilight. Silver Twilight consists of two closely related themes

which are designed for writing and editing for long hours, usually on a monitor in portrait mode. Portrait mode is generally more efficient than landscape mode for writing and editing productivity: It allows for the editor to comfortably fill most or all of the width of the screen, depending on the monitor resolution. The maximum width of the editor window should correspond to a maximum of between 77 to 105 characters per line within the typing area of the editor (average 91), depending on the zoom level and the choice of fixed-width font.<sup>9</sup> This leaves a generous full length of the rest of the screen available for writing or editing with a minimum need for scrolling.

Hi					
Style	Color	Sample	Style	Color	Sample
Global Override Background (B)	Base4		Global Override Foreground (F)	Base04	
Line Number Margin B	Base3		Line Number Margin F	Antibase0	
Current Line Background	Base3		Comment	Base0	
Inactive Tabs	Base2		Smart Highlighting	Cyan	
Selected Text Color	Base1		Fold Active	Cyan (NPP)	
Fold Margin B	Antibase4		Fold Margin F	Base0	













**Figure 4** Global Style: Silver Twilight Hi

On the other hand, staring at such a large area of writing space for long periods needs to be ameliorated, as discussed earlier. The Silver Twilight themes are designed to address and meet that need. Silver Twilight Hi is a light theme, perhaps best for daylight hours, but works for nighttime as well. Silver Twilight Lo is a dark theme, perhaps best for nighttime, but works for daylight as well. At the time of writing this manual, the first author is somewhat more satisfied with Silver Twilight Hi than with Silver Twilight Lo; your mileage may vary. Both could benefit from improvement in future versions; suggestions from the `CONTEXT` community are welcome!

In Notepad++ Style Configurator, a *global style* may be configured to set the general appearance of the editor. See (Language: Global Styles): Individual elements for configuration are listed to the right under Language: Style: <element>. A *lexer style* involves setting the code highlighting rules for each keyword class of a given lexer. See Language:<language>: Individual keyword classes for each lexer are also listed under Language: Style: <keyword class>. See also Figure 3.

Each Silver Twilight theme consists of a global and a lexer style. See Figures 4 and 5 for the global style of Silver Twilight High and of Silver Twilight Lo respectively.

<sup>9</sup> Typographers recommend a length of 45 to 75 characters per line (average 60); see Bringhurst (2008). However, writing and editing in a fixed-width font is not the same as reading the final output in a book or on a web page. Restricting the typing area of an editor to 45 to 75 characters per line feels forced (and is probably bad for anyone who has or is at risk for myopia). That said, Notepad++ can display a vertical edge and the user can choose a value for "number of columns", i.e., number of characters per line (we set it to 91). It would be nice if Notepad++ could automatically soft wrap (i.e., wrap without line breaks) the text at the vertical edge instead of at the border of the edge of the typing area.

Hi					
Style	Color	Sample	Style	Color	Sample
Global Override Background (B)	Base04		Global Override Foreground (F)	Base4	
Line Number Margin B	Base03		Line Number Margin F	Antibase0	
Current Line Background	Base03		Comment	Base0	
Inactive Tabs	Base02		Smart Highlighting	Cyan	
Selected Text Color	Base01		Fold Active	Cyan (NPP)	
Fold Margin B	Antibase4		Fold Margin F	Base0	

**Figure 5** Global Style: Silver Twilight Lo

Note that the lexer styles for Silver Twilight Hi and Lo for `CONTEXT` are almost identical: The only difference is that the foreground and background colors for the `DEFAULT` keyword class are reversed; see Figure 6. This is intentional: the two themes are intended to form a single system. In order for a common lexer style to work well between themes, the color scheme has to be well thought out.<sup>10</sup> Again, there is always room for improvement.

## 4.4 ALM Fixed













The default font for Silver Twilight is Arabic-Latin Modern Fixed, a derivation from Latin-Modern Mono developed by Idris Samawi Hamid. Designed for extensive use of Arabic script and its diacritics, it has a larger than usual interline spacing. For those who desire tighter interline spacing or just another default tpeface: Instead of tediously replacing the font in every dialog of Style Configurator, one can open `ConTeXt.xml` and `stylers.xml` and make a global substitution of the name ‘`ALM Fixed`’ with that of another font (preferably fixed-width) of one’s choosing, e.g., ‘`Dejavu Sans Mono`’.

# 5 The NPPEXEC Console

## 5.1 NPPEXEC and Console Scripts

The `NppExec` console is an integral part of the Notepad++ for `CONTEXT` system. When executed, it open a window which features a typing area for you to write a script, an option to save it, as well as a drop-down list of saved scripts. See Figure 7. There are 24 scripts that come with Notepad++ for `CONTEXT`; you can add and remove these or your own private scripts as well.

<sup>10</sup> The developer of Solarized had this ideal in mind: A single color scheme should work across nearly all keyword classes for each of a pair of light and dark themes. Note that a pair of Solarized themes is available for Notepad++ (the user will have to change any background tones used by the `CONTEXT` lexer style, as they are not compatible).

	<b>Hi</b>		<b>Lo</b>	
<b>Keyword Class</b>	<b>Color</b>	<b>Sample</b>	<b>Color</b>	<b>Sample</b>
DEFAULT (F)	Base03		Base03	
LINE COMMENTS	Base0		Base0	
TEX/ETEX	Maroon		Maroon	
LUATEX	Orange		Orange	
SYSTEM	Yellowgreen		Yellowgreen	
DOCUMENT	Green		Green	
STYLE	Yellow		Yellow	
CHARDEF	Magenta		Magenta	
CONSTRUCT	Violet		Violet	
PRIVATE	Blue		Blue	
STOP OPEN	Cyan		Cyan	
STOP CLOSE	Cyan		Cyan	
OPERATORS	Maroon		Maroon	
NUMBERS	Cyan		Cyan	

**Figure 6** CONT<sub>E</sub>XT Lexer Style: Silver Twilight

Let's take a look at **Plugins->NppExec->Advanced Options**; see Figure 8. At the top right you will notice that a script can be executed when Notepad++ starts or exits. By default, the **Scratch TeX File** script is executed when Notepad++ starts: The user will have to edit that script to the correct directory for one's scratch file. Of course one can disable the execution of any script. The **Purge Files** script is executed when Notepad++ exits; again, one can disable this.

## 5.2 NPPEXEC and CONT<sub>E</sub>XT

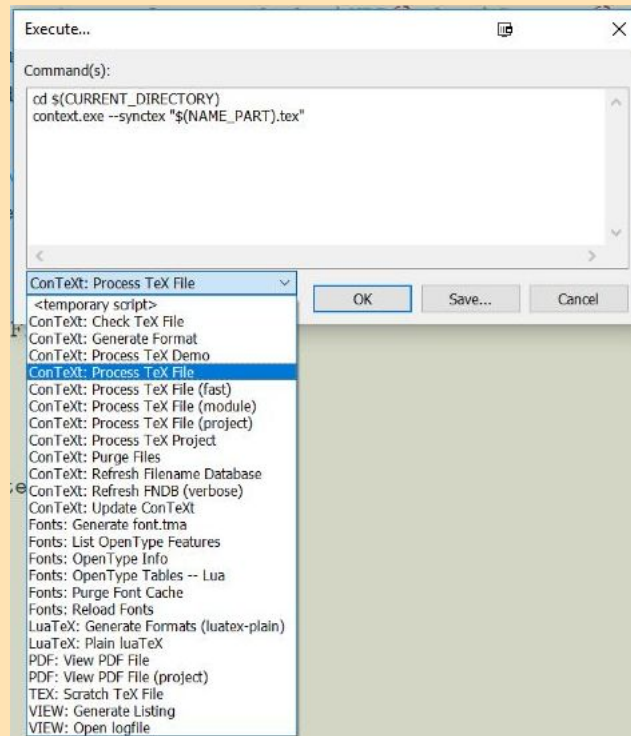
A brief description of each console script that ships with Notepad++ for CONT<sub>E</sub>XT follows; the names should be self-explanatory.

### 1. Check T<sub>E</sub>X File (Ctrl-0)

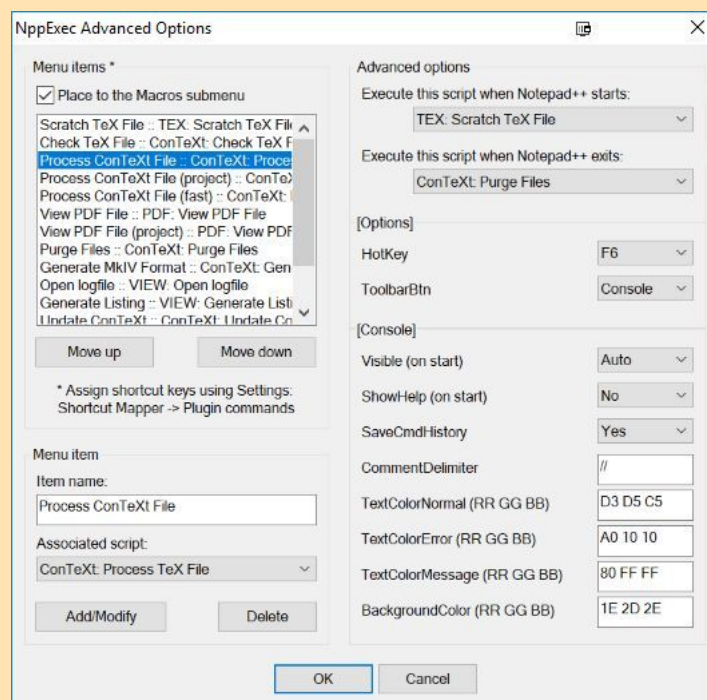
Note

```
mtxrun --autogenerate --script check
```





**Figure 7** NppExec and ConTeXt-related Scripts



**Figure 8** NppExec Advanced Options

## 2. ConTeXt: Process TeX File

```
context.exe --synctex "$(NAME_PART).tex"
```

## 3. ConTeXt: Process TeX Project

```
cd "C:\<path to your directory>"
context.exe --synctex "<your project>.tex"
```



**4. ConT<sub>E</sub>Xt: Process TeX File (fast)**

```
context.exe -mode=*nofonts "$(NAME_PART).tex"
```

**5. ConT<sub>E</sub>Xt: Process TeX Demo**

```
context.exe --mode=demo "$(NAME_PART).tex"
```

**6. ConT<sub>E</sub>Xt: Process TeX File (module)**

```
context --ctx=s-mod typo-mar.mkiv
```

**7. ConT<sub>E</sub>Xt: Purge Files**

```
context --purge
```

**8. ConT<sub>E</sub>Xt: Generate Format**

```
luatools.exe --generate  
context.exe --make
```

**9. ConT<sub>E</sub>Xt: Update ConTeXt**

```
first-setup.bat --engine=luatex
```

**10. ConT<sub>E</sub>Xt: Refresh Filename Database**

```
mktexlsr  
luatools.exe --generate
```

**11. ConT<sub>E</sub>Xt: Refresh FNDB (verbose)**

```
luatools.exe --verbose --generate
```

**12. LuaTeX: Generate Formats (luatex-plain)**

```
luatex --ini luatex-plain.tex
```

**13. LuaTeX: Plain luaTeX**

```
luatex --ini luatex-plain.tex
```

**14. T<sub>E</sub>X: Scratch TeX File**

```
notepad++.exe "C:\<path to your directory>\scratch.tex"
```

**15. PDF: View PDF File**

```
sumatra.bat "$(CURRENT_DIRECTORY)\$(NAME_PART).pdf"
```

**16. PDF: View PDF File (project)**

```
cd "C:\<path to your directory>  
sumatra.bat "<your project>.pdf"
```

**17. VIEW: Open logfile**

```
noepad++.exe "$(CURRENT_DIRECTORY)\$(NAME_PART).log"
```

### 18. VIEW: Generate Listing

```
mtxrun --autogenerate --script context --extra=listing --pretty --result="$(NAME_PART)"
"$(NAME_PART).tex"
sumatra.bat "$(CURRENT_DIRECTORY)\$(NAME_PART).pdf"
```

### 19. Fonts: Purge Font Cache

```
mtxrun.exe --script cache --erase
mtxrun --generate
```

### 20. Fonts: List OpenType Features

```
mtxrun.exe --script font --list --info lmroman12-regular.otf
```

### 21. Fonts: OpenType Tables - Lua

```
mtxrun.exe --script font --save lmroman12-regular.otf
```

### 22. Fonts: Generate font.tma

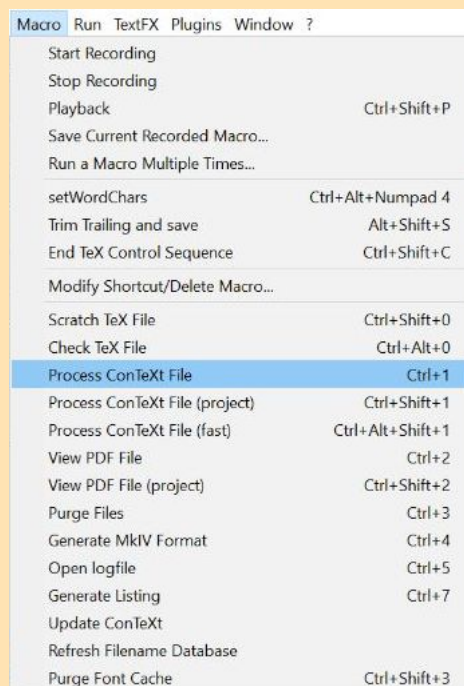
```
mtxrun.exe -- convert lmroman12-regular.otf
```

### 23. Fonts: Reload Fonts

```
mtxrun --script fonts --reload --force
```

## 5.3 NPPEXEC and the Macro Submenu

From the NppExec Advanced Options dialog, any console script can be made to appear at the bottom of the **Macro** submenu. We have configured 14 NppExec scripts appear there; see Figure 9. Note the submenu name need not be the same as the source console script name!



**Figure 9** NppExec Scripts and the Macro Submenu

## 5.4 Configuring Shortcut Mapper

# 6 The CON<sub>T</sub><sub>E</sub>X<sub>T</sub> Lexer

## 6.1 Components of the Lexer

## 6.2 Configuring Keyword Classes: Style Configurator and `context.xml`

## 6.3 Configuring Autocompletion and Tooltips: `ConTeXt.xml`

## 6.4 Configuring Macros, Tags, and Shortcuts: `ConTeXt.ini`

## 6.5 Configuring the Right-Click Menu

## 6.6 Usage

## 6.7 Note on bidirectional editing

In some cases, Scintilla will

# 7 References

Bringhurst, R. (2008). *The Elements of Typographic Style, Version 3.2*. Hartley & Marks, Publishers. (p. 11)

Sarkar, A. (2015, 7). The impact of syntax colouring on program comprehension. In The impact of syntax colouring on program comprehension. *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)*. Author. (p. 8)

## The Authors

author Idris Samawi Hamid, Professor  
 email [ishamid@colostate.edu](mailto:ishamid@colostate.edu)  
 affiliation Department of Philosophy  
 Colorado State University  
 The Oriental T<sub>E</sub>X Project

author Luigi Scarso  
 email [luigi.scarso@gmail.com](mailto:luigi.scarso@gmail.com)  
 affiliation The ConT<sub>E</sub>Xt Development Team  
 The LuaT<sub>E</sub>X Team

version January 31, 2018