pgcode formatter

Beautifies your PLpgSQL and SQL code

Document	pgcode_formatter_manual
Version	0.1 - 2021-01-26
Copyright	© Splendid Data Product Development B.V. 2020
License	https://www.gnu.org/licenses/gpl-3.0.en.html

Table of Contents

1. Introduction

The pgcode_formatter formats sql and plpgsql code according to a provided configuration xml file.

1.1 Command line

The pgcode_formatter can be used via the command line. The program directory contains a jar file containing the com.splendiddata.plpgsql.code.formatter.PlPgSqlCodeFormatterMain main class and a lib directory containing some libraries that are used by the program.

```
execute
java -jar pgcode_formatter-0.1-SNAPSHOT.jar -help
to see usage information.
```

Example:

```
java -jar pgcode_formatter-0.1-SNAPSHOT.jar --input source.sql --output
formattedSource.sql --config formatterConfig.xml
```

1.1.1 Command line options

? or --help

Shows the command line usage

-c or --config

One of the predefined profile names : profile1, profile2, profile3 and compact or a path to a configuration xml file in which the formatter options are specified.

If not provided, then the configuration saved in the user preferences/registry will be used.

If no configuration is found in the user preferences/registry then the default configuration will be used, see 3.1.13.

-i or --input

The input sql file that is to be formatted.

-o or --output

The file in which to put the formatted output.

```
If not provided, then stdout is assumed.
```

-S, --store-config

The provided configuration file, if any, will be stored in user preferences. This will be used in future calls when neither a profile name nor a configuration xml file is provided

1.2 Embedded

If you want to use the code formatter in an embedded way, then two classes are important: com.splendiddata.pgcode.formatter.FormatConfiguration and com.splendiddata.pgcode.formatter.CodeFormatter.

The FormatConfiguration can be instantiated wit a path to a config file or with an instance of com.splendiddata.pgcode.formatter.configuration.xml.v1_0.Configuration – or null if you want default formatting.

The CodeFomatter needs to be instantiated with an instance of FormatConfiguration. The CodeFormatter.format(String) method will do the trick.

2. Profiles

The pgcode_formatter provides different “standard” profiles that can be used for formatting the PL/pgSQL source code. Besides that, it offers the possibility to create your own format configuration and use it through the “-c” option.

3. Configuration

The XSD file for the configuration options can be found in the Jar file under META-INF/pgcode_formatter-v1_0.xsd.

A default configuration file exists in the jar file under com/splendiddata/pgcode/formatter/DefaultConfig.xml.

Many options have a "weight" clause. When two options collide, then the option with the highest "weight" value will be used.

3.1 Options

3.1.1 General settings

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Configuration
.emptyLine</td><td>Indicates what to do with empty lines.

Options:

<table><tr><th>Option</th><th>Description</th></tr><tr><td>preserveAll</td><td>All white lines are left as they are.</td></tr><tr><td>preserveOne</td><td>Blocks of white lines are reduced to a single white line</td></tr><tr><td>remove</td><td>All white lines are removed</td></tr></table>
Default: preserveOne</td></tr>
<tr><td>configuration
→lineWidth
.value</td><td>Maximum line width in characters. For tab characters, the amount of space characters that they would represent are counted.

Default: 140</td></tr>
<tr><td>configuration
→lineWidth
.weight</td><td>Defines the importance for the lineWidth setting.

Default: 10</td></tr>
<tr><td>configuration
→tabs
.tabWidth</td><td>The tab interval to be used, specified in number of characters.

Default: 4</td></tr>
<tr><td>configuration
→tabs
.tabsOrSpaces</td><td>Use tab characters or spaces.

Possible values: tabs, spaces

Default: spaces</td></tr>
</table>

Option	Description
Configuration →indent .indentWidth	Indentation size Default : 4
Configuration →indent .tabsOrSpaces	Use tab character or spaces of indenting. Default : spaces
Configuration →indent .indentInnerFunction	Indicates whether a nested function (create function enclosed in another create function or inside an anonymous block) should be indented or not. Default : false
configuration →letterCaseKeywords	Converts the postgres keywords to upper/lower case or keeps it unchanged. Possible values: lowercase, uppercase, unchanged Default: unchanged
configuration →letterCaseFunctions	Converts the postgres built-in functions to upper/lower case or keeps it unchanged. Possible values: lowercase, uppercase, unchanged Default: unchanged

3.1.2 query settings

Option	Description
configuration →queryConfig .indent	Are the major parts of a query indented relative to the start of the query? Default: false Example:

true	**false**
SELECT the, target, list FROM some_table WHERE where condition	SELECT the, target, list FROM some_table WHERE where condition

Option	Description

configuration →queryConfig .majorKeywordsOnSeparateLine	Do the major keywords in a query get a line of their own?
	The major keywords are “select”, “from”, “where”, “group by”, “order by” etc.
	Possible values: true, false
	Default: false
	Example:
	true
	SELECT
	  the, target, list
	FROM
	  some table
	WHERE
	  the where condition
	false
	SELECT the, target, list
	FROM some_table
	WHERE the where condition

configuration →queryConfig →maxSingleLineQuery .value	When a query does not exceed this number of characters, it will be written on a single line. Otherwise it will be split up by major keywords and spread over several lines. Default: 60
configuration →queryConfig →maxSingleLineQuery .weight	Determines the importance of the maxSingleLineQuery when for example competing with the configuration→lineWidth setting. Default: 5

3.1.3 Comma separated list settings

Settings are used for a comma separated list of no specific settings are set.

Option	Description
configuration →commaSeparatedListGrouping →maxSingleLineLength .value	When a comma separated list (for example is select statement) does not exceed this number of characters, it will be written on a single line. Otherwise it will be split-up over several lines. Default: 80
configuration →commaSeparatedListGrouping →maxSingleLineLength .weight	Importance of this setting over conflicting options (if any). Default: 5

Option	Description
configuration →commaSeparatedListGrouping .multilineOpeningParenBeforeArgument	When a comma separated list is split-up over several lines, will the opening parenthesis be written before the first argument (true) or will it be written after the function name (false). Possible values: true, false Default: true
configuration →commaSeparatedListGrouping .multilineClosingParenOnNewLine	When a comma separated list is split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last argument (false). Possible values: true, false Default: false
configuration →commaSeparatedListGrouping →maxArgumentsPerGroup .value	When a comma separated list is split-up over several lines, then how many arguments are to be written per line. Default: 10
configuration →commaSeparatedListGrouping →maxArgumentsPerGroup .weight	The importance of this setting for example compared with the maxLengthOfGroup setting. Default: 5
configuration →commaSeparatedListGrouping →maxLengthOfGroup .value	When a comma separated list is split-up over several lines, then how long can a group of arguments on a single line be (in characters). Default: 50
configuration →commaSeparatedListGrouping →maxLengthOfGroup .weight	The importance of this setting for example compared with the maxArgumentsPerGroup setting. Default: 10
configuration →commaSeparatedListGrouping →indent .value	When a comma separated list is split-up over several lines, then how will the arguments in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument
configuration →commaSeparatedListGrouping →indent.weight	The importance of this setting, for example when competing with the max line length. Default: 10

Option	Description
configuration →commaSeparatedListGrouping →commaBeforeOrAfter	If the comma separated list is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more "natural" than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before

3.1.4 Function call argument settings

Option	Description
configuration →functionCallArgumentGrouping →maxSingleLineLength .value	When a function call does not exceed this number of characters, it will be written on a single line. Otherwise it will be split-up over several lines. Default: 80
configuration →functionCallArgumentGrouping →maxSingleLineLength .weight	Importance of this setting over conflicting options (if any). Default: 5
configuration →functionCallArgumentGrouping .multilineOpeningParenBeforeArgument	When a function call is split-up over several lines, will the opening parenthesis be written before the first argument (true) or will it be written after the function name (false). Possible values: true, false Default: true
configuration →functionCallArgumentGrouping .multilineClosingParenOnNewLine	When a function call is split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last argument (false). Possible values: true, false Default: false

Option	Description
configuration →functionCallArgumentGrouping →maxArgumentsPerGroup .value	When a function call is split-up over several lines, then how many arguments are to be written per line Default: 10
configuration →functionCallArgumentGrouping →maxArgumentsPerGroup .weight	The importance of this setting for example compared with the maxLengthOfGroup setting. Default: 5
configuration →functionCallArgumentGrouping →maxLengthOfGroup .value	When a function call is split-up over several lines, then how long can a group of arguments on a single line be (in characters). Default: 50
configuration →functionCallArgumentGrouping →maxLengthOfGroup .weight	The importance of this setting for example compared with the maxArgumentsPerGroup setting. Default: 10
configuration →functionCallArgumentGrouping →indent .value	When a function call is split-up over several lines, then how will the arguments in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument
configuration →functionCallArgumentGrouping →indent .weight	The importance of this setting, for example when competing with the max line length. Default: 10
configuration →functionCallArgumentGrouping →commaBeforeOrAfter	If the comma separated list of function call arguments is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more "natural" than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before

3.1.5 Function definition argument settings

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>configuration
→functionDefinitionArgumentGrouping
.defaultIndicator</td><td>In a function definition, arguments can have default values. A default value can be defined using an equals sign (=) or the word “default”.

<table><tr><th>Option</th><th>Description</th></tr><tr><td>alterToDefault</td><td>The equals sign will be altered to the word “default”.</td></tr><tr><td>alterToEqualsSign</td><td>The word “default” will be altered to “=”.</td></tr><tr><td>asIs</td><td>Nothing will change.</td></tr></table>
Default: asIs</td></tr>
<tr><td>configuration
→functionDefinitionArgumentGrouping
→argumentGrouping
→maxSingleLineLength
.value</td><td>When function definition arguments do not exceed this number of characters, they will be written on a single line. Otherwise they will be split-up over several lines.

Default: 80</td></tr>
<tr><td>configuration
→ functionDefinitionArgumentGrouping
→argumentGrouping
→maxSingleLineLength
.weight</td><td>Importance of this setting over conflicting options (if any).

Default: 5</td></tr>
<tr><td>configuration
→ functionDefinitionArgumentGrouping
→argumentGrouping
.multilineOpeningParenBeforeArgument</td><td>When function definition arguments are split-up over several lines, will the opening parenthesis be written before the first argument (true) or will it be written after the function name (false).

Possible values: true, false

Default: true</td></tr>
<tr><td>configuration
→ functionDefinitionArgumentGrouping
→argumentGrouping
.multilineClosingParenOnNewLine</td><td>When function definition arguments are split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last argument (false).

Possible values: true, false

Default: false</td></tr>
</table>

Option	Description
configuration → functionDefinitionArgumentGrouping →argumentGrouping →maxArgumentsPerGroup .value	When function definition arguments are split-up over several lines, then how many arguments are to be written per line. Default: 10
configuration → functionDefinitionArgumentGrouping →argumentGrouping →maxArgumentsPerGroup .weight	The importance of this setting for example compared with the maxLengthOfGroup setting. Default: 5
configuration → functionDefinitionArgumentGrouping →argumentGrouping →maxLengthOfGroup .value	When function definition arguments are split-up over several lines, then how long can a group of arguments on a single line be (in characters). Default: 50
configuration → functionDefinitionArgumentGrouping →argumentGrouping →maxLengthOfGroup .weight	The importance of this setting for example compared with the maxArgumentsPerGroup setting. Default: 10
configuration → functionDefinitionArgumentGrouping →indent .value	When function definition arguments are split-up over several lines, then how will the arguments in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument
configuration → functionDefinitionArgumentGrouping →argumentGrouping →indent .weight	The importance of this setting, for example when competing with the max line length. Default: 10

Option	Description
configuration → functionDefinitionArgumentGrouping →argumentGrouping →commaBeforeOrAfter	If the comma separated list of function definition arguments is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more "natural" than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before
configuration → functionDefinitionArgumentGrouping →argumentName .alignment	How should the argument name be aligned? Possible values : verticallyAligned, subsequent, atHorizontalPosition Default : verticallyAligned
configuration → functionDefinitionArgumentGrouping →argumentName .minPosition	The minimum position where the argument name can be placed. Default : 0
configuration → functionDefinitionArgumentGrouping →argumentName .maxPosition	The maximum position where the argument name can be placed. Default : 9
configuration → functionDefinitionArgumentGrouping →dataType .alignment	How should the data type be aligned? Possible values : verticallyAligned, subsequent, atHorizontalPosition Default : verticallyAligned
configuration → functionDefinitionArgumentGrouping →dataType .minPosition	The minimum position where the data type can be placed. Default : 5
configuration → functionDefinitionArgumentGrouping →dataType .maxPosition	The maximum position where the data type can be placed Default : 40

Option	Description
configuration → functionDefinitionArgumentGrouping →defaultValue .alignment	How should the default expression be aligned? Possible values : verticallyAligned, subsequent, atHorizontalPosition Default : verticallyAligned
configuration → functionDefinitionArgumentGrouping →defaultValue .minPosition	The minimum position where the default value/expression can be placed. Default : 10
configuration → functionDefinitionArgumentGrouping →defaultValue .maxPosition	The maximum position where the default value/expression can be placed. Default : 60

3.1.6 Table definition settings

Option	Description
configuration →tableDefinition →argumentGrouping →maxSingleLineLength.value	When a table definition does not exceed this number of characters, it will be written on a single line. Otherwise it will be split-up over several lines. Default: 80
configuration →tableDefinition →argumentGrouping →maxSingleLineLength .weight	Importance of this setting over conflicting options (if any). Default: 1
configuration → tableDefinition →argumentGrouping .multilineOpeningParenBeforeArgument	When table definition is split-up over several lines, will the opening parenthesis be written before the first column definition (true) or will it be written after the function name (false). Possible values: true, false Default: true
configuration →tableDefinition →argumentGrouping .multilineClosingParenOnNewLine	When a table definition is split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last column definition (false). Possible values: true, false Default: false

Option	Description
configuration →tableDefinition →argumentGrouping →maxArgumentsPerGroup .value	When a table definition is split-up over several lines, then how many column definitions are to be written per line. Default: 1
configuration →tableDefinition →argumentGrouping →maxArgumentsPerGroup .weight	The importance of this setting for example compared with the maxLengthOfGroup setting. Default: 10
configuration →tableDefinition →argumentGrouping →maxLengthOfGroup .value	When a function definition is split-up over several lines, then how long can a group of column definitions on a single line be (in characters). Default: 50
configuration →tableDefinition →argumentGrouping →maxLengthOfGroup .weight	The importance of this setting for example compared with the maxArgumentsPerGroup setting. Default: 10
configuration →tableDefinition →indent.value	When a table definition is split-up over several lines, then how will the column definitions/constraints in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument
configuration →tableDefinition →argumentGrouping →indent .weight	The importance of this setting, for example when competing with the max line length. Default: 1

Option	Description
configuration →tableDefinition →argumentGrouping →commaBeforeOrAfter	If the comma separated list of table column definitions/constraints is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more "natural" than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before
configuration →tableDefinition →dataType .alignment	How should the data type be aligned? Relative position of the data type horizontally to its neighbouring items and vertically to data types Possible values : verticallyAligned, subsequent, atHorizontalPosition Default : verticallyAligned
configuration →tableDefinition →dataType .minPosition	The minimum position where the data type can be placed. Default : 5
configuration →tableDefinition →dataType .maxPosition	The maximum position where the data type can be placed. Default : 40
configuration →tableDefinition →columnConstraint .alignment	How should the column constraint be aligned? Relative position of the column constraint horizontally to its neighbouring items and vertically to column constraint Possible values : verticallyAligned, subsequent, atHorizontalPosition Default : verticallyAligned
configuration →tableDefinition →columnConstraint .minPosition	The minimum position where the column constraint can be placed? Default : 10

Option	Description
configuration →tableDefinition →columnConstraint .maxPosition	The maximum position where the data type can be placed. Default : 60

3.1.7 FROM clause grouping

Option	Description
configuration →fromItemGrouping →maxSingleLineLength .value	When a FROM clause does not exceed this number of characters, it will be written on a single line. Otherwise it will be split-up over several lines. Default: 100
configuration →fromItemGrouping →maxSingleLineLength .weight	Importance of this setting over conflicting options (if any). Default: 10
configuration →fromItemGrouping .multilineOpeningParenBeforeArgument	When a FROM clause is split-up over several lines, will the opening parenthesis be written before the first argument (true) or will it be written after the function name (false). Possible values: true, false Default: true
configuration →fromItemGrouping .multilineClosingParenOnNewLine	When a FROM clause is split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last argument (false). Possible values: true, false Default: false
configuration →fromItemGrouping →aliasAlignment.alignment	How aliases in a FROM clause should be aligned? Possible values : verticallyAligned, atHorizontalPosition, subsequent Default: verticallyAligned
configuration →fromItemGrouping →aliasAlignment .minPosition	The minimum position where the alias in a FROM clause can be placed. Default : 10

Option	Description
configuration →fromItemGrouping →aliasAlignment .maxPosition	The maximum position where the alias in a FROM clause can be placed. Default : 60
configuration →fromItemGrouping →comma	If the FROM clause is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more “natural” than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before

3.1.8 Target list grouping

Option	Description
configuration →targetListGrouping →maxSingleLineLength .value	When a target list (for example is select statement) does not exceed this number of characters, it will be written on a single line. Otherwise it will be split-up over several lines. Default: 80
configuration →targetListGrouping →maxSingleLineLength .weight	Importance of this setting over conflicting options (if any). Default: 5
configuration →targetListGrouping .multilineOpeningParenBeforeArgument	When a target list is split-up over several lines, will the opening parenthesis be written before the first argument (true) or will it be written after the function name (false). Possible values: true, false Default: true
configuration →targetListGrouping .multilineClosingParenOnNewLine	When a target list is split-up over several lines, will the closing parenthesis be written on a new line (true) or will it be written directly after the last argument (false). Possible values: true, false Default: false
configuration →targetListGrouping →maxArgumentsPerGroup .value	When a target list is split-up over several lines, then how many arguments are to be written per line. Default: 10
configuration →targetListGrouping →maxArgumentsPerGroup .weight	The importance of this setting for example compared with the maxLengthOfGroup setting. Default: 5
configuration →targetListGrouping →maxLengthOfGroup .value	When a target list is split-up over several lines, then how long can a group of arguments on a single line be (in characters). Default: 50

Option	Description
configuration →targetListGrouping →maxLengthOfGroup.weight	The importance of this setting for example compared with the maxArgumentsPerGroup setting. Default: 10
Configuration →targetListGrouping →indent .value	When a target list is split-up over several lines, then how will the arguments in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument
configuration →targetListGrouping →indent .weight	The importance of this setting, for example when competing with the max line length. Default: 10
configuration →targetListGrouping →commaBeforeOrAfter	If the target list is to be split up over several lines, is the comma to be placed at the end of the previous line or at the beginning of the next line. The comma at the end of the line looks more "natural" than a comma at the begin of a line as that is the normal way of writing in texts. But especially in complex constructs, the comma at the beginning of the line appears to be a lot less error-prone. Possible values: before, after Default: before

3.1.9 Logical operators (WHERE clause)

configuration →logicalOperatorsIndent.indent	When a WHERE clause is split-up over several lines and this clause contains logical operantors AND/OR, then how will the operands in following lines be indented. Possible values: underFirstArgument, indented, doubleIndented Default: underFirstArgument

3.1.10 Case operand clause

The caseOperand setting are for case clauses whereby the case keyword is immediately followed by an operand, so "CASE <*some_operand*> WHEN <*another_operand*> THEN ...".

Option	Description
configuration →caseOperand →maxSingleLineClause .value	When the total length of a case clause does not exceed this number of characters, it will be written on a single line. Default: 60
configuration →caseOperand →maxSingleLineClause .weight	Importance of this setting compared with for example the max line width or the maxSingleLineQuery setting. Default: 5
configuration →caseOperand →whenPosition .value	In a "case operand" clause that is split-up over several lines, where is the first when clause to be positioned (following when clauses will be positioned under the first). Possible values: whenAfterCase, whenUnderCase, whenIndented Default: whenUnderCase
configuration →caseOperand →whenPosition .weight	Importance of this setting over for example the max line length. Default: 10
configuration →caseOperand →thenPosition .value	In a "case operand" clause that is split-up over several lines, where is the then clause to be positioned. Possible values: thenAfterWhenDirectly, thenAfterWhenAligned, thenUnderWhen, thenIndented Default: thenAfterWhenAligned
configuration →caseOperand →thenPosition .weight	Importance of this setting over for example the max line length. Default: 10
configuration →caseOperand .elsePosition	In a "case operand" clause that is split-up over several lines, where is the else clause to be positioned. Possible values: elseUnderWhen, elseUnderThen Default: elseUnderThen

Option	Description
configuration →caseOperand .endPosition	In a "case operand" clause that is split-up over several lines, where is the end keyword to be positioned. Possible values: endUnderCase, endUnderWhen, endAtSameLine Default: endUnderCase

3.1.11 Case when clause

The caseWhen setting are for case clauses whereby the case keyword is immediately followed by an operand, so "CASE WHEN <*some_condition*> THEN ...".

Option	Description
configuration →caseWhen →maxSingleLineClause .value	When the total length of a case clause does not exceed this number of characters, it will be written on a single line. Default: 60
configuration →caseWhen →maxSingleLineClause .weight	Importance of this setting compared with for example the max line width or the maxSingleLineQuery setting. Default: 5
configuration →caseWhen →whenPosition .value	In a "case operand" clause that is split-up over several lines, where is the first when clause to be positioned (following when clauses will be positioned under the first). Possible values: whenAfterCase, whenUnderCase, whenIndented Default: whenAfterCase
configuration →caseWhen →whenPosition .weight	Importance of this setting over for example the max line length. Default: 10
configuration →caseWhen →thenPosition .value	In a "case operand" clause that is split-up over several lines, where is the then clause to be positioned. Possible values: thenAfterWhenDirectly, thenAfterWhenAligned, thenUnderWhen, thenIndented Default: thenAfterWhenAligned
configuration →caseWhen →thenPosition .weight	Importance of this setting over for example the max line length Default: 10

Option	Description
configuration →caseWhen .elsePosition	In a “case operand” clause that is split-up over several lines, where is the else clause to be positioned. Possible values: elseUnderWhen, elseUnderThen Default: elseUnderThen
configuration →caseWhen .endPosition	In a “case operand” clause that is split-up over several lines, where is the end keyword to be positioned. Possible values: endUnderCase, endUnderWhen, endAtSameLine Default: endUnderCase

3.1.12 Declare section and code section

Option	Description
Configuration →languagePlpgsql →declareSection →dataTypePosition .alignment	How should the data type in a declare section be aligned? Relative position of the data type horizontally to its neighbouring items and vertically to data types Possible values : subsequent, atHorizontalPosition, verticallyAligned Default : subsequent
Configuration →languagePlpgsql →declareSection →dataTypePosition .minPosition	The minimum position where the data type in a declare section can be placed. Default : 3
Configuration →languagePlpgsql →declareSection →dataTypePosition .maxPosition	The maximum position where the data type in a declare section can be placed. Default : 3
Configuration →languagePlpgsql →declareSection →dataTypePosition .constantPosition	The position of the keyword "constant" in a declare section. Possible values : subsequent, alignedBeforeDataType, alignedWithDataType Default : subsequent
Configuration →languagePlpgsql →codeSection →ifStatement.then	The position of the keyword "then" in an if statement. Possible values : afterCondition, onNewLine, singleLineAfterMultiLineUnder Default : afterCondition
Configuration →languagePlpgsql →codeSection →ifStatement .conditionIndent	The position of the condition in an if statement. Possible values : underFirstArgument, indented, doubleIndented Default : indented

Option	Description
Configuration →languagePlpgsql →codeSection →forStatement.loop	The position of the code in a for loop. Possible values : afterCondition, onNewLine, singleLineAfterMultiLineUnder Default : singleLineAfterMultiLineUnder

3.1.13 Default xml file

Below you can find the default config xml file. This is the default configuration used for formatting when no profile or config xml file is provided and no configuration file could be found in the preferences/registry.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:configuration
xmlns:ns2="http://www.splendiddata.com/pgcode_formatter/1.0/"
emptyLine="preserveOne">
    <lineWidth value="140" weight="10.0"/>
    <tabs tabWidth="4" tabsOrSpaces="spaces"/>
    <indent indentWidth="4" tabsOrSpaces="spaces"/>
    <queryConfig majorKeywordsOnSeparateLine="false" indent="false">
        <maxSingleLineQuery value="60" weight="5.0"/>
    </queryConfig>
    <commaSeparatedListGrouping multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
        <maxSingleLineLength value="80" weight="5.0"/>
        <maxArgumentsPerGroup value="10" weight="5.0"/>
        <maxLengthOfGroup value="50" weight="10.0"/>
        <indent value="underFirstArgument" weight="10.0"/>
        <commaBeforeOrAfter>before</commaBeforeOrAfter>
    </commaSeparatedListGrouping>
    <functionCallArgumentGrouping multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
        <maxSingleLineLength value="80" weight="5.0"/>
        <maxArgumentsPerGroup value="10" weight="5.0"/>
        <maxLengthOfGroup value="50" weight="10.0"/>
        <indent value="underFirstArgument" weight="10.0"/>
        <commaBeforeOrAfter>before</commaBeforeOrAfter>
    </functionCallArgumentGrouping>
    <functionDefinitionArgumentGrouping defaultIndicator="asIs">
        <argumentGrouping multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
            <maxSingleLineLength value="80" weight="5.0"/>
            <maxArgumentsPerGroup value="10" weight="5.0"/>
            <maxLengthOfGroup value="50" weight="10.0"/>
            <indent value="underFirstArgument" weight="10.0"/>
            <commaBeforeOrAfter>before</commaBeforeOrAfter>
        </argumentGrouping>
        <argumentName alignment="verticallyAligned" minPosition="0"
maxPosition="9"/>
        <dataType alignment="verticallyAligned" minPosition="5"
maxPosition="40"/>
        <defaultValue alignment="verticallyAligned" minPosition="10"
maxPosition="60"/>
    </functionDefinitionArgumentGrouping>
    <tableDefinition>
        <argumentGrouping multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
            <maxSingleLineLength value="80" weight="1.0"/>
            <maxArgumentsPerGroup value="1" weight="10.0"/>
```

```
            <maxLengthOfGroup value="50" weight="1.0"/>
            <indent value="doubleIndented" weight="10.0"/>
            <commaBeforeOrAfter>before</commaBeforeOrAfter>
        </argumentGrouping>
        <dataType alignment="verticallyAligned" minPosition="5"
maxPosition="40"/>
        <columnContraint alignment="verticallyAligned" minPosition="10"
maxPosition="60"/>
    </tableDefinition>
    <fromItemGrouping comma="before"
multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
        <maxSingleLineLength value="100" weight="10.0"/>
        <aliasAlignment alignment="verticallyAligned" minPosition="10"
maxPosition="60"/>
    </fromItemGrouping>
    <targetListGrouping multilineOpeningParenBeforeArgument="true"
multilineClosingParenOnNewLine="false">
        <maxSingleLineLength value="80" weight="5.0"/>
        <maxArgumentsPerGroup value="10" weight="5.0"/>
        <maxLengthOfGroup value="50" weight="10.0"/>
        <indent value="underFirstArgument" weight="10.0"/>
        <commaBeforeOrAfter>before</commaBeforeOrAfter>
    </targetListGrouping>
    <caseOperand elsePosition="elseUnderThen" endPosition="endUnderCase">
        <maxSingleLineClause value="60" weight="5.0"/>
        <whenPosition value="whenUnderCase" weight="10.0"/>
        <thenPosition value="thenAfterWhenAligned" weight="10.0"/>
    </caseOperand>
    <caseWhen elsePosition="elseUnderThen" endPosition="endUnderCase">
        <maxSingleLineClause value="60" weight="5.0"/>
        <whenPosition value="whenAfterCase" weight="10.0"/>
        <thenPosition value="thenAfterWhenAligned" weight="10.0"/>
    </caseWhen>
    <letterCaseKeywords>unchanged</letterCaseKeywords>
    <letterCaseFunctions>unchanged</letterCaseFunctions>
    <languagePlpgsql>
        <declareSection>
            <dataTypePosition alignment="verticallyAligned" minPosition="30"
maxPosition="60" constantPosition="alignedBeforeDataType"/>
        </declareSection>
        <codeSection>
            <ifStatement then="singleLineAfterMultiLineUnder"
conditionIndent="indented"/>
            <forStatement loop="singleLineAfterMultiLineUnder"/>
        </codeSection>
    </languagePlpgsql>
</ns2:configuration>
```