**B4P Automatic Document Generation Scheme**

**Automatic generation of HTML and online documentation**

**Georg A. zur Bonsen**

* **Feature**
* **Feature**
* **Feature**

All rights reserved, copyright © 2012…2020 by Georg zur Bonsen

[1 Introduction 5](#_Toc30828931)

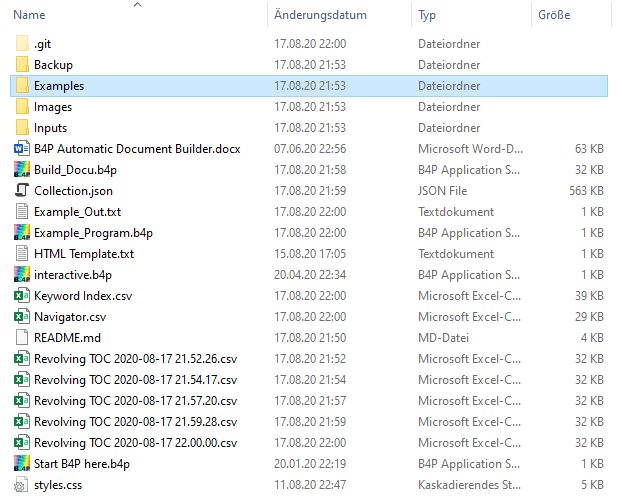
# Introduction

In the past, all B4P features have been described in a single document from which PDF's have been derived. In the following versions, the documents will be shifted to an online scheme using standardized data formats.

* Original documentation should be located in the same file where the implementation has been made, i.e. in the C/C++ or B4P library code in form of comment blocks. More generalized document sections may be located in separate text files
* The decentral nature of original documentation allows easy integration of additional contents, e.g. from persons who provided a library file with additional functions
* JSON is to be used as original structural framework to provide overseeable data stucture
* Text contents should be in HTML, allowing some text styles and including tables and illustrations where needed

**Docu\_Builder.b4p** is a B4P program which collects all input documentation, lines them up with the revolving table of contents and generates the HTML files.

# File Structure



|  |  |  |
| --- | --- | --- |
| **Style** | **Description** | **Usage** |
| Backup directory | Make backups of choice in order to recover older files if you spoiled more recent files. | You only |
| Examples directory | Data files used by code examples in B4P documentation | You need to provide them.  The Program will use them when running off code examples and the code is referring to these example files. |
| Images directory | Location of all images (and related files like PDF) which are referenced by HTML files | Updated by you  The Program will copy all contents to the destination directory along with the HTML files. |
| Inputs directory | (One of the) location(s) containing input documentation files to be included. | Updated by you  The Program will pick up all documentation inputs |
| B4P Automatic Document Builder.docx | This document | For your information |
| Build\_Docu.b4p | The document builder, referred to as "The Program". | You need to start it |
| Collection.json | Temporary JSON master file which contains all collected documentation which will then be used to generate the HTML output. | The Program  No harm if you delete this file. |
| Example\_Program.b4p Example\_Out.txt | Example programs in the documentation inputs will be run off automatically if "automatic" is specified in the "Ouput:" section. | Used by the Program.  No harm if you delete these files |
| HTML Template.txt | Framework template which is used to fill in the contents of the template in order to generate the HTML files. | Read by the Program |
| interactive.b4p start B4P here.b4p | Basic utilities | You may delete them |
| Navigator.csv Keyword\_Index.csv | Intermediate files for gathering keywords and constructing the navigator. | The Program |
| README.md | Intro file (for github) | You |
| styles.css | Stylesheet file | Updated by you  The Program will copy all contents to the destination directory along with the HTML files. |
| Revolving TCO YYYY.MM.DD hh.mm.ss.csv | Revolving Table of Contents. This file is used to manage the document structuring and location of the different documentation inputs. | Both you and The Program. |

# Adapting The Program for your environment

## Input Files

In the beginning part of Build\_Docu.b4p, you can specify all directories and file patterns where to look for B4P documentation contents.

// Specify all paths and file name patterns where to look for contents to include.

// The main variable must be "inputs[...]" where the member names are free of choice.

**inputs**[hand written stuff] = "inputs\\*.txt";

**inputs**[from source files] = "C:\Users\Zur Bonsen Georg\source\repos\Beyond4P\_Code\F\*.cpp";

Every source location must be defined as a member variable of **inputs**[…]. Choose unique and meaningful member names for each directory.

## Destination Directory

All outputs relevant for the web server will be copied into a destination directory. Make sure you specify an existing destination directory. Missing directories will not be created automatically. This specification is also in the beginning part of Build\_Docu.b4p.

// Destination path for all HTML files, images, etc.

// I am putting the files directly into the directory where the web contents will be copied to. Include slash at the end.

destination[] = "C:\Users\Zur Bonsen Georg\Documents\Programme\georgzurbonsen.github.io\";

## Multiple Users Sharing the Same Program

If multiple users (on different machines) are using this program, then include an if-statement to check the user name and then assign their directories. See B4P system variable **system info [ user name ]**.

# Document Sources

## Documentation Zones

The Program looks for **documentation zones** in C/C++ files (obviously inside /\* … \*/ comments in order to avoid interfering with the compiler or interpreter), B4P library files and any other text files. These zones have a starting and ending identifier.

The documentation zone begins with the next row below **B4PDOCU.START** and ends with the row above **B4PDOCU.STOP** . The Program will capture all contents inside these documentation zones.

Example

...

...

// C/C++ or B4P statements or any other text (ignored)

/\* **B4PDOCU.START**

// Documentation contents in JSON

// Format according to rules

// described in the follwing sections

// Comments inside documentation zones are considered as comments and

// are not included in the documentation.

**B4PDOCU.STOP** \*/

// C/C++ or B4P statements or any other text follow next(ignored)

...

...

One file may contain more than one documentation zone. I advise to add documentation zones next to every B4P function implemented so the documentation contents are located nearby and can be updated along with updating the source code.

## Documentation Syntax

### Enhanced Formulations

The syntax is based on JSON plus an enhancement to allow for clean documentation of contents requiring multiple lines.

Instead of doing something awkward like

"Description":

[ "This is row 1" ],

[ "This is row 2" ],

[ "This is row 3 "],

[ "This is the \"last\" row" ],

The Program supports enhanced formulations which begins with 2 consecutive colons, contents below and concluding with 3 plus-signs +++ on a following row. The contents are much easier to read and edit.

"Description"::

This is row 1

This is row 2

This is row 3

This is the "last" row

+++

The Program will automatically convert these enhanced formulations into true JSON syntax for further processing.

### Embedding HTML tags

Some content types allow HTML (e.g. "Description": blocks) in order to change style (e.g. <b>…</b> for boldfaced text), embedding images or adding hyperlinks to external web pages. Inserting tables with HTML is not recommended. Please use the markdown features for tables.

## Documentation Schemes

Following **documentation schemes** exist:

1. **General Descriptions**  
   Generic contents, e.g. introduction to B4P.
2. **Function Descriptions**  
   Structural definition of procedures and functions including their parameters, return values and examples

### Text Formats

Unicode is fully supported (UTF-8).

Entities (e.g. &amp; &eaigu;) are supported. Use &lt;, &gt; &amp; to use these symbols directly.

Quotation marks as part of contents must precede with backslash, e.g. \" to avoid confusing with end of string

### Color Conventions in Examples

Different colors are used to better describe the document structure to use.

|  |  |
| --- | --- |
| **Style** | **Description** |
| "*Blue italic text*" | Text of choice, but without any HTML markups or other formatting features |
| "*Light blue italic text*" | Text of choice, HTML markups allowed for formatting purposes |
| "Green text" | Required identifiers |
| "Light green text" | Optional identifiers |
| *Light gray italic text* | Remarks (information only in this manual) |
|  |  |

For various descriptions arrays may be required or at least optional.

Arrays for descriptive contents facilitate describing long text. Text in different array elements do not imply new line or new paragraphs. For HTML contents, specify <br/> for new lines.

JSON standards require all text to be in quotation marks.

### General Description

All procedure and function must be used using the **documentation scheme** "Function Description". This scheme provides clear rules to structure the function description.

The descriptive items may be in any order (except parameters as they must be described in the given sequence).

Syntax:

"*Title of generic description*" :

{

"Documentation": "General Description", // Use the value "General Description"

"Feature Names": [ "*name 1*", "*name 2*", *etc.* ], // Specify min 1 feature name

"Keywords": [ "*keyword 1*", "*keyword 2*", *etc.* ], // Provide keywords for

// cross referencing

"Library": "*Library name*", // Optional info

"Version": "7.00" // Optional info

"Description 01 *text*": [ "*html contents*", " *html contents* ", *etc.* ], // Contents

"Examples 01": [ "*contents*", "*contents*", *etc.* ], // Code example

"Output 01": [ "*contents*", "*contents*", *etc.* ], // Specify "automatic" for automatic  
 // execution of the code and insertion of

// the output right here

"Description 02 *text*": [ "*html contents*", " *html contents* ", *etc.* ],

"Examples 02": [ "*contents*", "*contents*", *etc.* ]

"Output 02": [ "*contents*", "*contents*", *etc.* ]

… etc …

// Up to 10 descriptions examples and outputs are available.

// If you have program exaple for Description 05 ... , then you must use "Examples 05" and

// "Output 05" in order to avoid displacement to locations above.

"Restrictions": [ "*html e.g. No indirect parameter passing*", *etc.* ],

"OS differences": [ "*html contents.details about deviating behavior in different OS*", *etc.* ],

"Exceptions": [ "*html contents*", "*html contents*", *etc.* ]

"See also": [ "*contents*", "*contents*", *etc.* ]

// Items under "See also" will be cross-referenced automatically.

"Notes": [ "*html contents*", "*html contents*", *etc.* ]

},

"*Title of next generic description*" :

{

*( Additional contents like above )*

}

### Function Descriptions

All procedure and function must be used using the **documentation scheme** "Function Description". This scheme provides clear rules to structure the function description.

The descriptive items may be in any order (except parameters as they must be described in the given sequence).

Syntax:

"*Title of function description*" :

{

"Documentation": "Function Description", // Required value, must be "Function Desccription"

"Function Names": [ "*name 1*", "*name 2*", *etc.* ],

"Keywords": [ "*keyword 1*", "*keyword 2*", *etc.* ],

"Library": "*Library name*",

"Synopsis": [ "*html contents, e.g. way how function is called*", *etc.* ],

"Version": "7.00"

"Description": [ "*html contents*", " *html contents* ", *etc.* ],

"Call as": "*procedure or function*", // or "procedure" or "function"

"Restrictions": [ "*html e.g. No indirect parameter passing*", *etc.* ],

"OS differences": [ "*html contents.details about deviating behavior in different OS*", *etc.* ],

"Parameter count": "*1*",

"Parameters":

[ { "Number": "1.",

"Types": [ "*Boolean*", "*Numeral*", *etc.* ],

"Name": "*Name of input parameter*",

"Direction": "*input*",

"Description": [ "*html contents*", "*html contents*", *etc.* ],

"Default value": "*contents*" // Use this one if parameter to provide is optional

},

{ "Number": "2.",

*( Additional contents like above )*

}

]

"Return value":

[ { "Name": "*Name of input parameter*",

"Type": "*Boolean*",

"Description": [ "*html contents*", "*html contents*", *etc.* ]

},

{ "Name": "2.",

*( Additional contents like above )*

}

]

"Exceptions": [ "*html contents*", "*html contents*", *etc.* ]

"Examples": [ "*contents*", "*contents*", *etc.* ], // Example program

"Output": [ "*contents*", "*contents*", *etc.* ], // or "automatic" for automatic execution  
 // of this example program

"See also": [ "*contents*", "*contents*", *etc.* ],

// Items under "See also" will be cross-referenced automatically.

"Notes": [ "*html contents*", "*html contents*", *etc.* ]

},

"*Title of next function description*" :

{

*( Additional contents like above )*

}

|  |  |  |
| --- | --- | --- |
| **Descriptor** | **Description** | **Arrays** |
| "*Title of function description*" | Use a unique title which has not been used elsewhere. | n/a |
| "Documentation" :  "Function Description" | Must specify "**Function Description**" to identify that this structure is describing a procedure and/or a function and applies the rules listed below: | No |
| "Function Names" | Provide 1 or more function names.  You are allowed to describe multiple similar functions together. | Optional |
| "Keywords" | List of keywords. Will be used to facilitate searching | Required |
| "Library" | Specify library if needed, e.g. "Office Library". | No |
| "Synopsis" | Describe how the function is called. | Optional |
| "Version" | Earliest B4P version supporting this function | No |
| "Description" | Describe the behavior of the function. Use arrays for multiple lines. | Optional |
| "Call as" | Rules if function is to be called as procedure, function (inside an expression), or both | No |
| "Restrictions" | Specify any applicable restrictions | Optional |
| "OS differences" | Specify any operating-system specific deviating behavior | Optional |
| "Parameter count" | Specify number of parameters. Value must be in text form. You are allowed to specify "1 or 2", etc. | No |
| "Parameters" | Describe the parameters (if applicable) | Required |
| "Number" | Parameter number. Use text form to specify "1 or 2" where needed | No |
| "Name" | Name of parameter | No |
| "Types" | List the supported parameter types  See separate table | Optional |
| "Direction" | Parameter passing direction:  "input" "output" "io" "reference" Must specify a variable being referenced "comparison " Code piece "RH expression" " "LH expression" " "statements" " | No |
| "Description" | Describe the parameter. | Optional |
| "Default value" | Describe the default value which applies if the parameter is not specified. | No |
| "Return value" | Specify the return value (if appliable). Use multiple entries if function may return values of different types. | Required |
| "Name" | Name of return value | No |
| "Types" | Variable type returned (e.g. numeral) | Optional |
| "Description" | Describe the return value | Optional |
| "Exceptions" | Exceptions | Optional |
| "Examples" | Code examples | Optional |
| "Output" | Output of code examples  To run the code example listed above automatically and insert the text output in here, then specify  **"Output" : "automatic".**  Otherwise, it's hand-written output. | Optional |
| "See also" | Specify further function names.  B4P will automatically generate links to their function descriptions. | Optional |
| "Notes" | Optional: Further notes | Optional |

## Types

|  |  |  |
| --- | --- | --- |
| **Code** | **Type Designation** | **Remarks** |
| A | "all types" | All types |
| B | "boolean" |  |
| C | "table columns" |  |
| D | "date" |  |
| E | "all" | Forces conversion to da te |
| F | "numeral or date" |  |
| G | "numeral or date or blank literal" |  |
| H | "numeral or literal or date or boolean" | valid types except parameter set |
| I | "numeral or boolean" |  |
| J | "date or literal" |  |
| K | "numeral or parameter set of numerals" |  |
| L | "literal" |  |
| M | "all types" | The type is converted to literal |
| N | "numeral" |  |
| O | "all types" | Converted to numeral |
| P | "parameter set" |  |
| Q | "parameter set or literal or numeral" | Softquoted literal with commas inside translate into multiple parameters  Softquoted '' translates to empty parameter set |
| R | "parameter set " | Like P, Members are converted to literal |
| S | "parameter set or literal or numeral" |  |
| T | "parameter set or literal" | Take both types, but no conversion made |
| U | "numeral or literal" |  |
| V | "valid types" |  |
| W | "numeral or literal or date" |  |
| X | "valid except boolean" | Like W, but includes parameter set |
| Y | "numeral or blank literal" |  |
| Z | "numeral or blank literal" | Blank literals are converted to 0 |
| 1 | "comparison expression" |  |
| 2 | "expression" |  |
| 3 | "variable", ":literal" |  |
| 4 | "statements" | 1 or more statemetns |

# Markdowns

## Itemizations

2-level itemizations using filled and unfilled round bullet symbols are supported.

First level items must begin with one \* followed by space. 2nd level items must begin with \*\* followed by space. White spaces before the asterisk aer allowed.

\* Countries

\*\* France

\*\* Spain

\* States

\*\* Ohio

\*\* Quebec

## Tables

Text blocks may contain tables. A dedicated markdown syntax allow easy definition of tables. Limit tables of total 1000 pixel width. Start the table on a new row with === symbol followed by column widths and optionally table classes and style instructions. 'noheader' suppresses creating a header row with different format (e.g. bold text) with the data in the 1st row.

=== 100, 200, 500, noheader, noframe\_1stcolbold

Column 1 in bold | column 2 | column 3

Column 1 in bold | column 2 | column 3

Column 1 in bold | column 2 | column 3

Column 1 in bold | column 2 | column 3

===

Multiple table classes can be inherited. In this case, list the classes (as defined in style.css file ) with space as separator, not comma.

Available class names:

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | **Remarks** |
| **noheader** | 1st row is considered part of contents | This is symbol is treated by B4P and is not a CSS class to format HTML contents |
| **noframe** | No frames |  |
| **bold\_1st\_column** |  |  |
|  |  |  |

## Markdown for Hyperlinks

All keywords in "see also" will be checked for references to other pages. If yes, then the first available cross-reference will be initiated with a hyperlink.

Inside any text (except code), specific keywords may be checked for available cross-referencing. If found, then they will be associated with hyperlinks. It will check for headers, feature names (includes function and procedure names) and all keywords.

Use 2 consecutive underscore symbols to encapsulate a term which should be hyperlinked. the underscore symbols will vanish in the output document.

The B4P function \_\_table consolidate\_\_ provides simple means to reduce a lot of data in many rows to condensed information you want to have and is easy to understand.

Special case: Hyperlinks to function names

This markdown \_\_literal\_\_ refers to the definition of the literal data type.

This markdown \_\_literal(\_\_) refers to the definition of the function name called literal

## Template

/\* B4PDOCU.START

"Function Title" :

{

"Function Names": [ "...", "..." ],

"Documentation": "Function Description",

"Keywords": [ "optional", "..." ],

"Library": "optional",

"Version": "optional",

"Synopsis": [ "fuction name (...);" ],

"Description": [

"... ",

"... " ],

"Call as": "procedure or function",

"Restrictions": "No indirect parameter passing",

"OS differences": "None",

"Parameter count": "1",

"Parameters": [

{

"Number": "..",

"Name": "...",

"Direction": "input,output,io,reference,comparison,RH expression,LH expression,LH variable, statements",

"Types": [ "literal,..." ],

"Description": [

"...",

"… " ],

"Default value": "optional"

}

],

"Exceptions": "optional",

"Examples": [

"optional",

"..." ]

"Output": "optional or automatic"

"See also": [ "optional", "..." ],

"Notes": [ "optional", "..." ],

}

B4PDOCU.STOP \*/

# Revolving Table of Contents

The document generation tool will use a revolving table of content scheme to manage all the contents and put them into one sequential and structured order like in a user manual. The docu generator reads the most recent file for existing structure, makes necessary updates and saves the file again.

Input File name: Revolving TOC\*.csv (most recent file available)

Output File name: Revolving TOC YYYY-MM-DD hh.mm.ss.csv

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Status** | **Category** | **Level** | **Section Nr** | **HTML File Name** | **Section Name** (used as unique identifier) | **Feature Names** | **Keywords** | **Remarks** |
| OK | Body | 1 | Body 1 | functions\_trig.html |  | sin,cos,tan | trigonometric functions, sine, cosine, tangent | This is a remark |
| Check | Appendix | 2 | Appendix 1.1 |  |  |  |  |  |

**Status** is updated by the tool and can take following values:

New Has been added automatically (Level and section number still blank)

Missing Specified Section Name cannot be matched with an existing document Section

OK Confirmed entry and used for docu generation.

Check E.g. Level missing

**Category**: Maintained by user, used by tool

Intro Beginning Part

Body Document Body

Appendix Appendix Part

**Level**: Maintained by user, used by tool

x

**Section Nr**.: Maintained by the tool

Section numbers will be derived from the level and sequence.

Will only be generated if a level is specified (0 or blank: None, 1, 2, 3 = 1, 1.1, 1.1.1) etc.

**HTML File Name**: Maintained by the tool

**Section Name:** Maintained by both user and the tool (both can update)

**Feature Names:** Maintained by the tool

**Keywords**: Maintained by the tool

**Remarks:** Maintained by the user, and not processed by the tool

# Further References

<https://stackoverflow.com/questions/9725675/is-there-a-standard-format-for-command-line-shell-help-text>

Syntax rules