# DocuVeeb Specification VBA Auto-Documentation Tool

This Specification describes how to document procedures in a VBA library so that the comments can be gathered into usable external documentation. The approach is designed to gather items of information that are relevant in the VBA context of running within Office applications.

Most (which?) of the documentation comments should be treated as HTML, and may include basic HTML tags. Remember that extensive use of HTML may adversely affect the readability of the comments within the code while enhancing the appearance of the output documentation. Users are encouraged to seek a balance between the two, since both internal and external documentation are essential.

## Procedure Comments

Documentation for a procedure is in a block of comments directly preceeding the procedure declaration. The comment block conforms to the following specification. Most of the attributes of the documentation are noted with @tags.

### Procedure Comment Format

The block begins with a blank line followed by a line with two comment marks only, with no space between them and no text following them.

A brief description (usually a single sentence) is on the first line(s) following the double comment mark, followed by a blank comment line. The start of the brief description is implied, with no @tag.

A full description follows the first blank comment line, and ends at the first @tag. The start is also implied, with no @tag.

Other attributes are denoted by a @tag at the start, and continue until either the following @tag or the procedure declaration.

Place @tags at start of the line, after the comment mark, with no following punctuation. A space between comment mark and @ sign is encouraged, to improve legibility, but the space is not required.

The text may include basic HTML tags, such as <br>, <p> or <code>…</code>, which will be used in the formatted output. Text to be formatted as code should be placed within HTML <code> tags to be treated literally.

The tags should not include any information that can be gleaned from the declaration (the DocuVeeb tool will include that information automatically), except to call attention as needed. If certain information is ambiguous in the declaration, such as a variant parameter type that can accept specific different variable types, the ambiguity should be clarified in the tags.

The description text should be considered a block of HTML, with HTML tags if appropriate. Many of the @tag fields can also contain HTML, except those tags that are interpreted.

## Module-Level Comments

### Group Note Format

The groupnote block must appear within the Declaration section of the class module. The note begins like the procedure description, with a blank line followed by a line with two comment marks only, with no space between them and no text following them. The following line begins with a comment mark, then the tag @groupnote. All comment lines following the tag are included in the group note, which ends with the first line starting with a pair of single quotes.

The only tag supported in a class module is @groupnote.

### Class Modules

In a class module, a group comment section can describe the class. The comments will be displayed on the class (group) page in the default HTML output. The comments are typically used to provide an overview of the class.

### Standard Modules

[Module-Level Comments for standard modules are still under development.]

In a standard module, a group note applies to a single code group, specified by the @group tag on the last line before the group comment closes. The argument to the @group tag must match a group specified in procedure comments, and must match the same format.

## Supported Tags

| **Tag** | **Purpose** | **Content** |
| --- | --- | --- |
|  | Brief description | Quick overview of the procedure, with few details |
|  | Full description | Includes the details that are left out of the brief description; should not duplicate information from the tags or the procedure declaration |
| @param | Description of parameters, with tag for each | Parameter description only, not including name, type, passing, optional, etc., which are determined from declaration. Must match the list of parameters in the declaration, in both count and order |
| @return | Possible return values | As with parameters, ignore that which is stated in the declaration; instead, describe potential return values and what they mean.  Currently ignored for subs; however, using @return to describe the expected outcome of a sub is under consideration |
| @example | Reference to external example(s) | An external reference specified one of two ways:  1. A statement with a link, either explicit or {@link} tag  2. Path (relative or absolute) to an HTML file of examples  Only a single @example tag may appear for a procedure; if more than one appears, all but the final statement will be discarded.  Short examples can appear in the description; examples longer than a few lines should be placed in a separate file. Note that in-line examples in the description should be enclosed in <code> tags, if appropriate.  Currently under development |
| @group\* | Related procedures | Name of the function group(s) that includes the proc; replace spaces with underscores; multiple group names are separated by semicolons. In a class module a group tag is ignored (the class itself is the group) |
| @see | Related procedures or groups, or other information | Individual references, external to the group or class. Each @see reference must be on its own line. Must not contain semicolons unless the text is enclosed in quotes.  May be specified several ways:  @see {@link …}  @see <a …>: fully specified HTML link  @see “text”: the text in quotes is treated literally |
| @author\* | Author’s name | Primary author’s name may be followed by revising authors (on one line, separated with semicolons, or on separate lines) |
| @date\* | Date the procedure was created | More reliable for comparison than string-based version info, which can be dependent on string formatting |
| @version\* | Version in which proc first appeared | For example:  ' @version 2.10.4 |
| @docdate\* | Date the doc was last updated | Refers to the documentation for a procedure, not for the overall code base |
| @deprecated | Indicates proc is deprecated | Note replaced by what, and in which version; the {@link...} tag should refer to the replacement documentation. In the default output the text is displayed after a “Deprecated” flag. Must appear on a single line |
| @worksheet\* | Useable in a worksheet | “Yes/No” or a note: whether the function can be used in a worksheet cell; either word by itself will generate a generic statement in the default output.  May also contain a note if explanation is needed |
| @groupnote\* | Describes a class  (or procedure group) | Description that applies at the module (class) level.  Support for HTML tags and for group notes in standard modules are currently under development |

\* Tag does not support HTML

### Basic Header Block

''  
' Brief description, usually a single line/sentence, but may extend to following lines.  
'   
' Longer description should fully explain the procedure.   
' Use as many lines as needed.  
' <p>Longer description, cont’d., as new paragraph.  
' @param Describe the first parameter  
' @param Describe the next parameter  
' @return Describe return values  
' @group Name of the function group  
' @worksheet Yes/No  
' @see {@link procedure}  
' @example Include\ProcExample.html  
' @deprecated Note explaining the deprecation, preferably with {@link} to preferred code  
' @author Primary author’s name  
' @date First created, or revised  
' @version Text or number; first software release with this proc  
' @docdate Most recent update for this proc’s documentation  
Function Declaration()…

### Examples

Some tags do not require an argument, but are more useful with further explanation. For example, the @deprecated tag by itself (no argument) will flag a procedure as deprecated, but leaves the user in the dark about its replacement:

' @deprecated

The following statement notes that a procedure has been deprecated, and provides a reference to the preferred proc.

' @deprecated As of Release 2.3, replaced by {@link NewImprovedProc}.

### Sample Procedure Header

The comment block below demonstrates extended syntax. Note that each parameter has its own tag; the description of the first parameter is assumed to continue until the next tag is encountered. Note also the use of the HTML line-break tag “<br>” within the first parameter description to force new lines in the formatted output. The first parameter also illustrates an exception to the rule of not stating the variable type. In this case, because multiple types are specifically allowed for the variant, each acceptable type is described.

''  
' Finds the pay period the specified number of PPs in either direction.  
'  
' Offsets the PP, rolling over at the start of calendar years.  
' @param The starting PP, from which the offset is to be calculated.  
' Accepts the following input types:<br>  
' Date<br>  
' Date & PP as long integer (NOT STRING): YYYYPP<br>  
' String denoting a PP: "PP/Year"  
' @param The desired number of PPs to travel; negative to move backwards. Next PP if no value specified.  
' @return Returns in the format "PP/Year" on success; returns any error from PpStart() or PpOfDate().  
' @group Pay\_Periods  
' @worksheet Yes  
' @author Dr. Richard Cook  
' @date 5/4/2015  
Public Function PpOffset(ByVal varInput As Variant, Optional nOffset As Integer = 1) As Variant

In the above example, note the parameter descriptions. The first parameter is a variant that excepts three data types, so the three types are listed and explained. The second is optional. DocuVeeb will automatically note that it is optional, and that the default value is 1. The description explains that the default value of 1 indicates the following pay period, just to clarify its meaning (1 PP in the future).

### Sample Module (Class) Comment

The comment block below demonstrates a description of a group note at the class level. Note that the group comment must both begin and end with a pair of single quotes.

Support for HTML tags within group notes is currently under development.

''  
' @groupnote The ClsGroup class handles the grouping of declarations within a project.  
' <P>Groups are essential to DocuVeeb, because the collection of groups  
' includes all information gleaned from the code and the comments.  
' Passing the collection of groups to a subroutine passes all information available  
' for processing the set of declarations.  
''

## In-Line Tags

In-line tags are used within other statements or tags, such as a description. Currently the only in-line tag supported is the {@link} tag, which is used to link to documentation for the specified procedure, class or group. The {@link} tag is used for internal references only; external links must be spelled out literally, using standard HTML tags.

All {@link} tags in a project are resolved after all declarations have been read. The text containing the tag is converted to HTML with an embedded link. Inline tags that are within HTML <code> tags are treated literally, without interpretation.

### Specification of the @link Tag

The full specification of an in-line @link tag includes the project, module and procedure names, as below, with a dot between project and module, and a pound-sign between module and procedure:

{@link vbproject.module#procedure label}

In the context of documentation in VBA, the difference between module and class can be safely ignored (a class is a type of module), so the full specifications of each are identical:

{@link vbproject.module#procedure label}

{@link vbproject.class#member label}

The “label” in the above specs is the text to be displayed in the HTML link. The label is optional, as explained below. In most contexts, an abbreviated specification is used; only enough information to resolve ambiguity is required.

In most cases, a single vbproject is involved, the “vbproject” is optional and can be left off.

### Abbreviated Specifications

Abbreviated specifications for @links are commonly used, according to the following rules.

Procedure names may be specified as single-word references, with or without the leading #:

{@link procedure}  
{@link #procedure}

Procedure and label need not be repeated; if no label is specified, the procedure name itself is used for the link text. The following three statements are equivalent; each yields a link to the named procedure, and the procedure name is displayed in the link:

{@link ProcedureName}  
{@link #ProcedureName}  
{@link #ProcedureName ProcedureName}

If a procedure name is duplicated in multiple modules, the module name must be included in this statement, or the resulting link may lead to the wrong procedure. (The parsing tool has no way to determine which is intended.) This is just one of many reasons why duplicate procedure names within a project, outside of classes, should be forbidden as standard practice.

#### Class/Group/Module Links

A reference to a class or group name may be specified as a single word, but the leading dot is required, as in “.ClsDeclaration” to link to the ClsDeclaration class page:

{@link .ClsDeclaration}

The leading dot for a module or class/group name is not optional; without it, DocuVeeb would assume it refers to a procedure and would be unable to resolve the link.

## Front End Usage

### Basic Parameters

The following parameters must be provided on the Parameters page in the DocuVeeb workbook. Most should be self-explanatory. To add a new project to the list, enter the set of parameters in the next available column to the right. The Project Name will automatically appear in the “Select Project to Document” drop-down.

|  |  |
| --- | --- |
| Project Name |  |
| Display Name | Name displayed in output |
| Project Path (output) | Fully qualified path to the documentation output |
| VBA Project (wkbk; start of name) | Start of VBA project, typically the workbook filename |
| Topic Files Path | Subfolder for individual proc files, e.g., Topics\ |
| Group Files Path | Subfolder for group files, e.g., Groups\ |
| Contents Filename | Name of the main HTML output file |
| HTML file extension (.htm/.html) | Extension to be used on all HTML files |
| Create Web Help Contents | Y/N, |
| Include Non-Public declarations | Y/N, |
| Include Enums (under dev) | Y/N, |
| Include Constants (under dev) | Y/N, |
| Generate Group pages | Y/N, |

### HTML Help Parameters

The following parameters apply when creating a Windows HTML Help file as part of the output.

|  |  |
| --- | --- |
| **Help (.hhp) File** |  |
| Create HTML Help Project | Y/N, whether to create the HTML project file |
| Create HTML Help Contents | Y/N, |
| [OPTIONS] |  |
| Binary Index | Y/N, whether to create a binary index |
| Index Filename | If yes, its filename |
| Compatibility | Version of Windows HH Workshop |
| Compiled file | Name of the .chm file |
| Contents file | Name of the .hhp file |
| Default topic | Page to display when Help is first opened |
| Display compile progress | Y/N, |
| Error log file | Filename |
| Full-text search | Y/N, to allow full-text search |
| Language | Windows value, e.g., “0x409 English (United States)” |
| Title | Text to display in the title bar |

The optional HTML Help output creates input files for the Windows HTML Help Workshop.

### Execution

The source workbook to be documented must be open before running DocuVeeb. The name of the workbook matches the VBA Project parameter, which must include enough of the start of the workbook filename to be unambiguous among open workbooks. DocuVeeb acts on the first open workbook it finds that matches the parameter.

For example, if the filename includes version info, such as FooBar\_v3.1.xlsm, the VBA Project parameter “FooBar” will match as long as no other open workbook also begins with “FooBar”. In other words, if FooBar\_v3.0.xlsm is also open, DocuVeeb will act on the first of those that it finds, which is not necessarily the desired version.