Whitepaper

BayStateDocumentation

We Make your **Products Speak**

The ABCs of API Documentation

www.BayStateDocs.com

Table of Contents

What is API Documentation?

API stands for application programming interface. It is an interface used by programmers to work with (or interface with) an application. In today's world of ever specialized applications, standalone software is extremely rare. Developing complete solutions requires working with applications from other parties. To successfully do this, programmers require well written API documentation. API documentation is also required to develop applications on platforms provided by companies such as Google, Intuit, Sun, and Salesforce.com etc.

API documentation is writing about software code. Having a technical background helps. Today, however, companies don't always have the luxury of hiring writers with technical backgrounds, so "non-technical" writers are being asked to help out with API documentation. For writers without technical backgrounds, API documentation is often technical and mysterious – a cause for instant stress. This white paper will try to demystify API documentation to help writers get started with writing API documentation.

The Big Picture – SDK and Developer's Documentation

Before we dive down into describing API documentation, let's take a step back so that we can better understand the big picture. To enable developers to work with a company's application, the developers require more than just an API document. They may also need:

- Code libraries
- APIs
- Sample programs
- Tutorials
- Developer's documentation that describes some or all of these items.

Companies often include all these items in a software development kit or SDK.

Types of Developer's Documentation

Developer's documentation consists of one or more of the following:

- API Reference Documentation
- Programmer's Guide
- Installation and Configuration Guide

This white paper focuses on API reference documentation. API reference documentation is also called API documentation for short, and that's the term we will use for the rest of this white paper.

Overview of API Documentation

API documentation is the most widely used developer's documentation. This is primarily a reference document that describes:

- An application's code
- What information (or parameters) a developer sends to the application
- What the application will do
- Response back from the application

Intended for advanced users, API documentation includes precise information about the software code. It describes the APIs (also called functions or methods) that programmers use to interface with the application.

API documentation can be developed for virtually all software technologies such as Java, JavaScript, AJAX, C++/C#, TCP/IP, JSP tag libraries, XML schemas etc.

Some key areas to consider when writing API documentation include:

- Content source for the API document
- Delivery formats
- Document outline
- Information to document for each function
- Automatic document generation programs
- Best practices

These are being described here in more detail.

Content Source

Content for API documentation may come from one or more of the following sources:

- Source Code
- Specifications
- By using the software
- Raw APIs written by developers (writers may provide fill-in-the-blank templates). Raw APIs may also be provided by QA or other technical groups
- Documentation for previous releases
- Bug tracking software e.g. Bugzilla
- Standards groups

Given the technical nature of the material, writers often start out with the raw content written by developers.

Typical Outline

As API reference documentation is primarily a reference document, the bulk of the information is focused on the APIs themselves.

There is no such thing as a “standard outline” – but a typical API document will consist of the following topics:

- Introduction (a few pages)
    - Who needs this
    - What can I do with this API
    - How the guide is organized
- Quick Tutorial (a few pages)
    - “I want to see it working”
    - How to install or reference
- Concepts (a few pages, slightly more if needed)
- Reference (80% of the content, if not more)
    - List of all functions along with the detailed information for each one

Content to Document for Each Function

Although the exact content will vary for each document, you should typically include the following elements for each function:

- A brief description. This is no more than a sentence or two and describes the purpose of that function.
- The syntax. This describes the core components or parameters of the code, the order in which they occur, the required elements, the optional elements, etc.
- The actual parameters of the function. If there are many, use a table.
- Syntax for common parameters such as “request” and “response”
- Syntax for error messages. Error messages are a form of response, but may be documented separately as they are important to call out.
- Sample code. A snippet of the code can be pasted right in the document or can be located at some central site with the links provided in the document.
- Any examples relevant for the developer to see.
- Any relevant notes/cross-references/limitations.

As this is reference material, consistency is very important. Therefore you should document all elements for each function in exactly the same sequence. The following is an example of a function called “Get Exchange User”.

Example of an API "Get Exchange User"

Description

The Get Exchange User API fetches the metadata properties for the specified exchange user. This does not support the fetching of 5x exchange user.

Resource Location

/v1/services/workspaces/workspaceUsers?workspaceId=###&workspaceUserId=###

HTTP Method

This API uses HTTP GET Method.

Parameters

These parameters are specific to the Get Exchange User API.

Name	Description	Format	Required	Example
workspaceId	Exchange ID.	long	Y	?workspaceId=123
workspaceUserId	Exchange user ID.	long	Y	workspaceUserId=456

Request

There is no XML request

Response

```
<?xml version="1.0" encoding="UTF-8"?>
        <workspaceUser>
                <id>345</id>
                <version> 84b129f40a88475032</version>
                <userId>9999</userId>
        </workspaceUser>
```

The exchange user XML definition is shown in the following table:

Element	Description	Value Type	GET (always returned)	CREATE	UPDATE	Examples
userId	GUD user ID.	long	YES	W+4	NO	56789
id	The exchange user ID.	long	YES	NO	W+	456
version	The hashed version of the versionId.	string	YES	NO	W+	84bdb129f40a88475032

Automatic Document Generation Programs (DGP)

API documentation can be extremely lengthy and repetitive – but automated document generation programs (DGPs) can help. The way it works is that developers add comments to the software during the software development process. The DGPs then parse the source code for embedded comments to generate API documentation. This works even better if the writer and developer agree in advance about the type and depth of comments to include.

Some popular DGPs include:

- JavaDoc (for Java code)
- Doxygen (for .NET code)

Automation does not change what is being created, just how it is created. Automated documentation is ideal for high-volume, low-polish documentation e.g. documenting public APIs for groups that must use each other's code.

But automated documentation falls short in certain situations e.g.:

- Many developers don't like writing comments, so the code may not have adequate comments to generate useful API documentation.
- Developers may start out by documenting the code, but may skimp on it as delivery deadlines loom.
- Writers may not be good with written English, leading to consistency and quality issues.
- And, for the final document to be useful, it is important to include non-automated content such as overviews, tutorials, installation instructions etc.

Quite often, technical writers use the auto-generated content as a starting point, clean it up, format it, and include additional information to make it more useful.

Common Delivery Formats

These documents are delivered in printed format, as PDF, as online help or a combination thereof. The trend is more towards online formats, especially as more companies move towards an online product delivery or Software as a Service (SaaS) model.

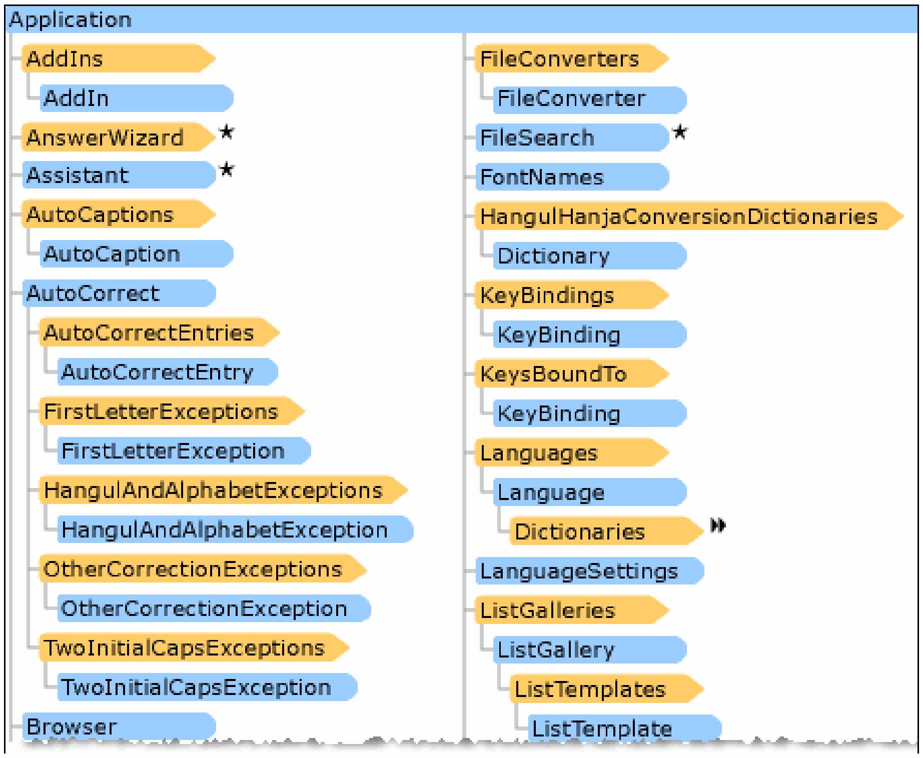
Depending on the software platform, the help can be generated in any format e.g. HTMLHelp, WebHelp, JavaHelp, etc. WebHelp seems be gaining traction as more companies move towards a web-based delivery model.

Best Practices

Organization

- Wherever possible, make things visual. This could be in the form of hierarchical diagrams, block diagrams, tables etc. Include workflows and illustrations to describe concepts.
- If possible, create a single view that displays all functions/objects, organized in some logical way. The following is an example from a Microsoft application.

Example of Single Page View



Navigation and Search

- API Reference Documents tend to be long and repetitive, and many functions look similar. Organizing functions in some logical fashion, e.g. alphabetically, makes information easier to find.
- Because of its length, it is important that the navigation and search capabilities be strong. Some developers like to start with the TOC, some with the index, while others prefer to search. It pays dividends to have all of these incorporated.

Samples

- In general, the more the better.
- Include samples for both trivial and complex use of the functions / APIs.

Editing

- When using an automatic document generation program, it is preferable to make edits to the comments in the code rather than manually editing the document. (Writers may have to provide guidance to the developers).

Summary

In a world of ever increasing software integration, more and more writers are being asked to help write API documentation. For writers not trained in API documentation, this can be intimidating. This white paper introduces writers to API documentation and discusses some best practices. It is intended to reduce writers' anxiety in getting started with API documentation. API documentation will continue to gain importance in the future, and technical writers will be well served in becoming a part of this business.

"We Make your **Products Speak"**

Bay State Documentation

45 Coventry Lane

North Andover, MA 01845

Phone: 978-852-7019

Send comments to comments@BayStateDocs.com

www.BayStateDocs.com

V1 081409