APIs and SDKs: Breaking into a Specialty Market

Ed Marshall

Copyright 2012
APIs and SDKs

API = Application Programming Interface
SDK = Software Development Kit

• Typical users and why they use them
• Typical producers of these products
• Examples
Typical Documentation Deliverables

• Programmer’s Reference Guides (often delivered online in some format)
• Programmer’s Guides
• Performance & Tuning Guides
• API and SDK Installation Manuals
Ideal Information for SDKs

• Provide an overview of the SDK
• Describe the tools and components in the SDK and how they relate to the APIs
• Describe each tool in detail
• Describe any sample programs included in the SDK
Ideal Information for APIs

• Break each component into the various families
• Describe each API completely, including cross-references to any types used in the definition
• Provide and explain examples that show both trivial and complex use of the class/API
Reference Information for APIs

• Brief description
• Syntax
• Examples, examples, examples!
• Error messages
• Cross-references
Key Programming Concepts

- Data types and variables
- Program control – loops, conditions, etc.
- Logical operators
- Data structures, such as arrays
- Functions and methods
Benefits to the Writer

• Do more advanced technical writing = Higher pay, higher status
• Good if you like to play with software at the code level, create and test examples, talk and write in gibberish
• Work more closely with developers
• Willingness/confidence to work closely with senior developers
• Ability to develop context-sensitive level help at a lower level than the typical end-user (window-level) help
Drawbacks to the Writer

• Possibly restrictive/repetitive writing
• Possibly less contact with users as they are developers/programmers themselves
• Possibly, more technically challenging development/build environments
Knowledge/Personality Traits that Work Well

• Some knowledge of programming languages BUT you don’t have to be a programmer!
• Willingness to work with advanced/programmer types of tools – Use software instead of specs
• Desire to work at the code level and write for developers who work at the code level
   Willingness/confidence to work closely with senior developers
   Ability to develop context-sensitive level help at a lower level than the typical end-user (window-level) help
Typical Skills Needed

• Good writer, but not great
• Strong organizational skills
• Proactive in getting information
• Have a flair for tools
Ways to Add Value

• Organizing information – change from timeline to alphabetical, grouping by categories of function, etc.
• Adding missing information
• Removing unnecessary information
• Correcting discrepancies/non-parallel formats/descriptions
• Using terms consistently
• Providing better definitions – meaningful, not just repeating the function
• Providing usage notes
• Recommending sequence of use – flowcharts
• Alphabetizing lists of error messages (removing duplicates)
• Correcting spelling via custom spell-check dictionaries
Ways to Get Information

• Read the code / run automated tools against the code to get reference information: “Single Source of Truth”

• Use traditional technical communication interviewing skills to get why and when to use a specific function/method (flowcharts are excellent for showing common tasks).

• Provide fill-in-the-blank templates for developers.
Information You Can Get from the Code?

Much of your reference information:
- Function/methods: signatures, parameters and their data types, and return values
- Error codes
Information You Can’t Get from Reading the Code

• Why someone would use a certain function
• The recommended logic flow – the order in which functions need to be called
• Background information
• Usage notes
• Relationships between functions (get/set, open/close, etc.), but can deduce these and verify with developers
Possible Strategies/Timeline for Accessing Code

1. Edit comments in code but don’t make changes in the code – shows you know the difference between comments and code.

2. Learn how to generate output from code and demonstrate “safe computer practices.” Do test builds and verify against known good builds.

3. Commit to only changing comments – and stick to it.

4. Best practice: Build outputs locally using the company’s system before checking in changes – builds trust in your work.
Typical Daily Activities

1. Check nightly build logs for failures/check-ins.
2. Update Perforce – look for changes to doc.
3. Research any changes to APIs, error messages, etc.
4. Check bug system for my assigned bugs, changes to bugs I’m following, new API-related bugs.
5. Download and install new kits (if it’s safe to do so).
6. Document new features/changes to APIs.
Commonly Used Automated Tools

• Doxygen
• Javadoc
• Sandcastle – Tool for HTML Help 1.x / .NET help (MSDN style)
Doxygen

- Very powerful code generation tool
- Free
- Reads specially formatted comments in code
- Supports C/C++, Java, Python, IDL, and C#
- Outputs RTF, compiled HTML Help, browser-based help, and LaTeX (PDF)
- Compatible with Javadoc 1.1
- Active development/support
Javadoc

- Powerful code generation tool for Java
- Free
- Reads specially formatted comments in code
- Outputs browser-based help
- Active development
- [www.doclet.com](http://www.doclet.com) – source for Java Doclets and Javadoc information
Integrated Development Environments (IDEs)

  Free from www.microsoft.com/express/download/

• NetBeans – free, very good for Java
  netbeans.org/downloads/index.html

• Eclipse – www.eclipse.org (a free open source IDE) – also good for Java
Determining Which Help Format to Use

Issues to consider:

• Platforms
• Browsers
• Minimum versions required by your product

Commonly used help formats:

• HTML Help 1.x (chm) – Windows only
• WebHelp/Web Help (or Javadoc) – good for all platforms
Writing Style Differences

• Use active voice.
• Sentence fragments are acceptable.
• Use verbs/gerunds to start phrases.
• Eliminate “useless” words that are noise to developers.
Helpful Low-cost Tools

Advanced Text Editors:
• NoteTab Pro - www.notetab.com
• EditPad Pro - www.jgsoft.com

File/Folder Level Comparison (Differencing Tools)
• Beyond Compare – www.scootersoftware.com/
• Araxis Merge - www.araxis.com/merge/index.html

Search and Replace Tools
• Funduc – http://www.funduc.com
• FAR (Find and Replace) – http://www.helpware.net/FAR/
Sample APIs

• Google Earth API – earth.google.com/comapi/
• Google Maps API – code.google.com/apis/maps/
• BackPack – www.backpackit.com/
• Oracle JMS (Java Messaging Service) API – www.oracle.com/technetwork/java/overview-137943.html
Breaking into this Market

• Get training to develop the skills:
  - Courses – College-level programming courses, workshops (STC)
  - Self-paced training
  - On-the-job training

• Make your own sample help systems, with context-sensitive help coded

• Write some sample programs
Self-Paced Training

- Manuel Gordon’s API materials
  (www.gordonandgordon.com)
- *Documenting APIs: Writing Developer Documentation for Java APIs/SDKs* – James Bisso/Victoria Maki
  (www.bitzone.com/book.html)
- Deitel & Deitel “(C/C++/C#/Java) How to Program”
- Sams “*Teach Yourself…*”
- “Fundamentals of Programming” – University of Toronto
  (https://www.coursera.org/)
- Sample projects, such as the Microsoft HTML Help API
Listservers and Other Groups

• STC API – groups.yahoo.com/group/svcstcapi/
• API writers – groups.yahoo.com/group/APIWriters/
• HATT – groups.yahoo.com/group/HATT/
• MS Help Viewer (formerly MSHelp 3) – http://tech.groups.yahoo.com/group/MSHelpViewer/
• Eclipse – groups.yahoo.com/group/eclipse_tw/
• API Documentation – a LinkedIn Group
Summary

• Description of APIs and SDKs
• Benefits to writers
• Drawbacks to writers
• Training
• Writing considerations (tools, formats, issues for context-sensitive help)
Closing

• Thank you.
• Questions?

Ed Marshall
Marshall Documentation Consulting
ed.marshall@verizon.net
LinkedIn: Ed Marshall
Twitter: EdMarshall
+1 978-339-3095