



STC Headquarters
9401 Lee Highway, Suite 300
Fairfax VA 22031
(703) 522-4114
Fax 703-522-2075
stc@stc.org

Chief Executive Director:
Kathryn Burton
kathryn.burton@stc.org

Membership Manager:
Julia O'Connor
julia.oconnor@stc.org

STC President: Dr. Hillary Hart
hart@mail.utexas.edu



Chapter Officers & Volunteers

President: Need volunteer!
president@stc-berkeley.org

VP Programs: Need volunteer!
programs@stc-berkeley.org

VP Membership: Patrick Lufkin
membership@stc-berkeley.org

Secretary: Susan Jaeger
secretary@stc-berkeley.org

Treasurer: Nicolette Davis
treasurer@stc.berkeley.org

Public Relations: Carole Baden
publicrelations@stc-berkeley.org

Employment: Need volunteer!
employment@stc-berkeley.org

Acting Webmaster: Linda Urban
internet@stc-berkeley.org

Member-at-large: Patrick Lufkin
memberatlarge@stc-berkeley.org

Past-President: Richard Mateosian
past-president@stc-berkeley.org

Arrangements: Tonie Flores

Newsletter Editor: Gwendolynne Barr
newsletter@stc-berkeley.org

Education: Need volunteer!
education@stc-berkeley.org

Elections: Need volunteer!
elections@stc-berkeley.org

Recognition: Need volunteer!
recognition@stc-berkeley.org

Volunteers: Need volunteer!
volunteers@stc-berkeley.org

Other contacts

Chapter Job List:
employment@stc-berkeley.org

Address, phone, or email changes:
membership@stc-berkeley.org

Technical communication is the bridge between those who create ideas and those who use them. Conveying scientific and technical information clearly, precisely, and accurately is an essential occupation in all sectors of business and government.

The Society for Technical Communication (STC) has members worldwide. Its members include writers and editors, artists and illustrators, photographers and audiovisual specialists, managers and supervisors, educators and students, employees and consultants.

STC strives to:

- Advance the theory and practice of technical communication
- Promote awareness of trends and technology in technical communication

Ragged Left

The newsletter for the [Berkeley Chapter of the Society for Technical Communication](#)

Winter 2012
Volume 25, Number 1

Editor's Notes

by Gwendolynne Barr

Dear readers,

Welcome to our first issue of 2012. This quarter, for our Employment and Training section, we present you with an [Introduction to Documenting APIs, SDKs, and Web Services](#). Thank you to Denise Green, Ed Marshall, Richard Smith, and Susan Gallagher for agreeing to be interviewed for this article.

For our program summaries, [Tracy Baxter](#) tells us about some nifty new features in the Adobe Tech Comm Suite 3.5. And [Greg Hubbard](#) reminds us why ~~us~~ ~~writers~~ we writers need editors in our lives.

In our [Student Corner](#), you can learn how to attend the STC Summit in Chicago for zero to \$500. If you know of even better ways to save money, [shoot me an email](#) and I'll add it as an update.

Finally, check out our [upcoming speakers](#), Bruce Poropat and Nicki Davis. In March, Bruce is speaking on [Plain Language](#), a skill that is important in government jobs (believe it or not). And Nicki is speaking on how we can better understand users with ethnographic research (so bring your binoculars).

As always, we welcome volunteers for the newsletter. If you have an idea, please [send us your proposal in an email](#). Writing an article is a great way to learn about a subject that is important to you and interview experts in that field.

[Top](#)

Election Results

They came, they ran, they conquered! The 2012 election results are in:

- **Patrick Lufkin** won as Vice President for Membership
- **Susan Jaeger** won as chapter Secretary
- **Nicki Davis** won as chapter Treasurer

Congratulations to the winners. But more importantly, we thank them for their hard work supporting our chapter. And thank you to our outgoing officers, **Richard Mateosian** and **Linda Urban**. They have put in long hours to ensure that our chapter keeps running smoothly.

This is a good place to inform members that **many positions in the Berkeley STC chapter are currently open, including that of President and VP of Programs**. To learn more about these positions, contact past-president@stc-berkeley.org.

Please consider volunteering, if not for a position, then for a single project or a limited period of time.

Finally, don't forget to vote in the STC 2012 [international election](#) which will be open from March 9-30.

[Top](#)

Program Notes

November 2011 Program Notes

Tools of the Trade: Adobe Technical Communication Suite 3.5

Presentation by Dustin Vaughn: Working with FrameMaker, RoboHelp, and Captivate in Adobe Technical Communication Suite 3.5

Article by Tracy Baxter

Writers like me who are making a transition to technical communication stand before a many-tined fork in the road. Where to start? Is professional satisfaction to be found in a medical writing setting? In financial services? In consumer electronics? In virtually every sector, there is an ongoing need to make sense of specialized information for other people to put to use. With so many opportunities and so broad a discipline, deciding on a point of entry is difficult.

Fortunately, surveying the tools of the trade and figuring out what they do can help narrow down the possibilities somewhat. Turns out there are a slew of tools, too, but Adobe is the go-to provider of software for technical writers, which makes exploring its technical communication suite of products a very valuable investigation.

Adobe bills its tightly integrated technical communication suite as an end-to-end solution, and it's hard to argue with that since it is composed of FrameMaker, RoboHelp, Captivate,

In this issue

- [Editor's Notes](#)
- [Election Results](#)
- [Program Notes](#)
- [Nov 2011 Program Notes](#): Working with Adobe TCS 3.5
- [Feb 2012 Program Notes](#): Editing: You Can't Write Without It
- [Student Corner](#): Attend the STC Summit for \$500 or Less
- [Employment & Training](#): Introduction to Documenting APIs
- [Upcoming Programs](#)
- [Meeting Logistics](#)
- [Local STC Chapters](#)
- [Other Organizations](#)

- Aid the educational and professional development of its members

Membership

[STC Membership](#) is open to everyone. Classic membership is \$215 per year with an additional \$25 per chapter and \$10 per SIG. STC also offers Limited, E-Membership, and Student Membership options. To receive additional information and an application form, email membership@stc-berkeley.org.

Insurance

Members of STC can apply for health, disability, and other insurance at STC group rates. For more information, contact STC office at stc@stc.org or (703) 522-4114.

Worldwide activities

STC's annual conference brings together more than 2,000 technical communicators from around the world for educational programs, seminars, and workshops conducted by experts in the field. **The 2012 STC Summit will be held in Chicago, May 20-23, 2011.** In addition the STC sponsors many regional conferences, which feature the same sorts of programs, seminars, and workshops on a more intimate scale. STC sponsors international and regional competitions in all aspects of technical communication. STC [Special Interest Groups \(SIGs\)](#) bring together members with common experiences and interests to share their skills and knowledge.

STC sponsors research grants and scholarships in technical communication.

STC publishes the journal [Technical Communication](#), the newsletter [Intercom](#), and other periodicals, reference materials, manuals, anthologies, standards, and booklets.

Formed in 1953, STC has today become the largest professional society in the world dedicated to advancing the theory and practice of technical communication.

Local activities

The six northern California chapters of STC conduct a variety of individual and joint activities. These and a list of other local organizations in which STC members may be interested are included in *Ragged Left*.

Subscriptions

This newsletter is free to the public.

Advertising rates

Ragged Left is not accepting advertising at this time.

Submissions

Ragged Left publishes original articles and illustrations. We edit them to meet our needs. You retain copyright but grant every STC publication royalty-free permission to reproduce the article or illustration in print or any other medium. Please talk with the editor for details of how to submit articles and illustrations.

The deadline for unsolicited submissions is the 15th of the last month in each quarter (March, June, September, December).

Other STC publications are hereby granted permission to reprint articles from *Ragged Left*, provided such reprints credit the author and the specific *Ragged Left* issue, and a copy of any publication containing such a reprint is sent to the *Ragged Left* editor.

Contact *Ragged Left* at newsletter@stc-berkeley.org.

Photoshop and Acrobat, applications that, by and large, cover the production of most the essential pieces of technical documentation. And the applications are of course continually retooled as the technical communication discipline and user expectations evolve.

The very personable Dustin Vaughn, Adobe Business Development Manager (Central U.S. & Canada), took some time at the November 2011 meeting of the STC Berkeley Chapter to review a few key changes among the many in Adobe Technical Communication Suite 3.5.

Not surprisingly, he said that the social media experience drove a number of tweaks in the package. Over 35 hours of video are uploaded to YouTube *every minute*. More than 30 *billion* pieces of content (web links, news stories, blog posts, notes, photo albums, etc.) are shared on Facebook each month. With numbers like that, it's easy to see just how much social media consumers are accustomed to rich and interactive media experiences. Users of technical documentation have similar expectations. Help pages now need some oomph.

A number of features in TCS 3.5 help add the social media kick to documentation. Captivate, for example, offers quiz templates, one-click YouTube publishing, MP4-format publishing for video streaming, podcasts, and video tutorials in its mix of new functionalities, giving tech communicators plenty of dynamic options for engaging users.

Then there are enhanced features in RoboHelp that deliver content to popular gizmos, putting help pages on smartphones, e-Book readers, and tablets. You can publish in the format you want: to print, or as WebHelp, XML, HTML, or PDF files, whatever is appropriate, with just a click.

Before long in Dustin's presentation, I began to marvel at just how many tools a tech writer can use on the job and just how many people can end up working on a document. With so many cooks in the kitchen, the process could be chaotic and the results just as bad. Turns out that some of the most impressive updates to the suite keep loose ends to a minimum.

FrameMaker allows you to launch Captivate to pop in, say, a how-to video without leaving the FrameMaker interface. With FrameMaker you can also edit a graphic with Photoshop, again, without switching applications. And here's something really sweet in the suite to make it easier to collaborate: the ability to review and edit documents for free. All that's needed is for the document to be created in TCS, saved as a PDF, and then distributed by acrobat.com, Microsoft SharePoint, or an email link to colleagues, subject matter experts, etc. All the document recipient needs to mark up their document is the free Adobe Acrobat reader, so only one TCS license is required. Their edits can then be incorporated in the final document by the original author.

Of course, Dustin's presentation was the tip of the information iceberg. But for me, his presentation, combined with a few well-spent hours completing software tutorials at Lynda.com, has given me some much-needed perspective. Technical communication isn't only about writing—although the ability to craft clear, simple, direct content is a big part of it. It's also about using the right tools to deliver information in the right format. It's both a right- and left-brain occupation.

Good to know. Now as I scan job postings at craigslist, Kforce, and Dice, I have a clearer picture of what I'd like to do as a technical writer. Captivate is high on my list of interesting tools; same with RoboHelp. Fortunately, the interfaces for both are fairly intuitive (as are those for all of the components of the suite, really) so building up a proficiency in using them should come quickly. After that, it'll be easier to identify companies and sectors where those skills are in demand.

Dustin Vaughn is a Business Development Manager with Adobe Systems Inc. in the Technical Communications group.

Tracy Baxter is a technical communicator and marketing copywriter in the San Francisco Bay Area.

[Top](#)

February 2012 Program Notes

Editing: You Can't Write Without It

Presentation: Bringing the Editor's Perspective to Our Work

Linda Urban moderated a panel of 4 editors: Louise Galindo, Jeff Gardiner, Deirdre Greene, and Daniel Milne.

Article by Greg Hubbard

Editing is a crucial part of creating effective technical communication. And in the past, many technical communicators developed their writing skills by working with outstanding editors. But in an environment where having access to an editor is becoming more rare, what can writers do to make up for this shift?

The Editors Panel Discussion at the Berkeley Chapter meeting on February 8th included a wide ranging discussion about editors, their various roles in the current environment, and what strategies you can employ when you don't have access to an editor.

The Editorial Landscape

The writers in the audience confirmed what many have suspected: more and more writers are working without the benefit of editors. When a writer *does* work with an editor, the writer/editor ratio can range from 20:1 to 1:1, and the functions the editors perform can vary from developmental editing to copy editing and proofreading for punctuation and grammar errors.

Each editor on the panel had a valuable perspective to share from his or her unique work situation:

- **Daniel Milne** edits for Oracle University where he proofreads and copy edits the work of writers creating course content (often as much as 300 proofread or 125 copy edited

pages a day!).

- **Louise Galindo** leads a team of 20 writers at VMWare and edits 10-12 topics a day in a DITA/Xmetal framework while trying to stay on top of documentation structure for upcoming features.
- **Deirdre Greene** is president of Roaring Forties Press. She also edits in a variety of contract situations, but most often works one-on-one with the writer of the project.
- **Jeff Gardiner** edits the work of SMEs at VMWare who create highly technical training content, often on topics he's not familiar with and with little means of getting feedback on his edits before he finalizes them.

There are many different tasks that the panelists perform in their role as editors:

- Creating style sheets and taking the lead in choosing tools.
- Getting involved in the product chain and becoming a SME.
- Copy editing and proofreading.
- Helping non-writers with basic writing skills, such as structuring steps in tasks.
- Keeping translation costs down.
- Staying in the loop on upcoming features to prepare topics in advance.
- Teaching writers to write and catching writing problems.

Working Without an Editor

Great editors have the focus and discipline to see the patterns and the inconsistencies that many writers miss. The panel was asked if they had advice for writers who don't have access to an editor. The panel had two main suggestions.

#1: Use Resources and Style Guides

Take full advantage of reference materials, both those that already exist and ones that you create. Stay familiar with style manuals, use available style guides, and keep a style sheet where you track document-specific choices.

#2: Remember the Basics

Stay focused on the basic principles of good technical communication. Use topic-based, minimalist writing. In topic-based writing, each topic has only one task, concept and reference. Minimalist writing emphasizes giving the user only what they need, nothing more. Use simple, plain language that is unambiguous and communicates clearly to the user.

Other common sense ideas were shared:

- After writing, let some time go by before you attempt to edit your own work so that you can look at it with a fresh eye.
- Think about whether you proofread more effectively on the screen or do better with a hard copy.
- Don't forget to take advantage of your writer peers when you need the benefit of another perspective on your writing.

Most importantly, the panelists stressed that we all need to continue to advocate for the importance of effective technical communication in keeping costs low and giving the customers a great experience with the product. Help management see that the skills writers and editors possess are crucial to customer satisfaction and that involving them early in the development process can help create more effective products.

Louise Galindo is a senior technical editor in the VMware Technical Publications department.

Jeff Gardiner is a senior editor in the Content Development group at VMware.

Deirdre Greene is a founder and president of Roaring Forties Press. She also freelances as a copyeditor, developmental editor, and project manager.

Daniel Milne has worked as a technical editor since 2000, first at Siebel University and now at Oracle University.

Greg Hubbard is a technical communicator in the San Francisco Bay Area.

[Top](#)

Student Corner

Attend the STC Summit in Chicago for \$500 or Less

Article by Gwendolynne Barr

Students, get ready for the STC conference in May! In this article, I explain how you can attend for roughly \$500 or less. If you live in Chicago, you can attend for free.

Conference

The summit is advertised as being in Chicago, but technically, it is in Rosemont at the [Donald Stephens Convention Center](#) by the [Hyatt Regency O'Hare](#) and near Chicago O'Hare airport. The convention center and hotel are [within walking distance](#) of each other.

As I [posted in a previous issue](#), if you are an STC student member, you can attend the conference for free by working as a volunteer. Look for an email from Lloyd Tucker about this in the upcoming weeks. I had a great experience volunteering at the 2011 conference, despite having to work during some of the sessions in which I was interested.

If you would prefer not to volunteer, the [student rate is \\$175](#). You can attend at this rate until 11 May 2012.

Hotel

To save money on a hotel, consider staying in a [youth hostel](#). The sponsored hotel, The Hyatt Regency, costs \$200 per night. If you stay for the entire conference, 3 days and nights, that's a pricey \$600 plus tax.

One way to stay at the Hyatt for less is to share a room. Four can share a room for \$249.00, or roughly \$68.00 a night. Two people can share a room for \$209, or 104.50 a night. Of course, "sharing" means sharing a bed as well as sharing a room.

The cheapest option is the youth hostel located in downtown. Rates on the weekend are \$33 and during the week, \$29. Non-members have to add \$3 a night and the hotel tax in downtown Chicago is 15.4%. That adds up to about \$115 for the entire conference.

Another idea is to stay at a [cheap hotel near the airport](#). You should be able to book a room for 3 days for about \$160 total. Make sure it provides shuttle service to the airport. You can ride your hotel shuttle to the airport, then ride the Hyatt shuttle to the Hyatt, then walk around the corner to the convention center.

You could also try [Airbnb](#). Every rental differs, but the gist is that you rent a space from some person just like you. Some rentals are in that person's house. Others are in an unoccupied investment of theirs. If you decide on Airbnb, just make sure that you can get to the convention center easily.

Public Transportation

Public transportation from the youth hostel in downtown Chicago to the convention center takes about an hour each way. You have to [walk 5 minutes to the Jackson station](#) on the [Blue line](#). Get off at the Rosemont station and [walk 10 minutes to the convention center](#). Addresses are:

- [Youth hostel](#): 24 East Congress Parkway, Chicago, IL 60605
- [Jackson-Blue CTA station](#): 328 S. Dearborn Street, Chicago, IL 60604
- [Rosemont-Blue CTA station](#): 5801 North River Road, Rosemont, IL 60018
- [Donald Stephens Convention Center](#): 5555 N. River Road Rosemont, IL 60018

Airfare

Airfare to Chicago O'Hare airport (ORD) differs per your location, but the basic range is from \$150 (LGA-ORD) to \$400 (SFO-ORD).

A good site for finding flights is the [Matrix Airfare Search](#) by Ita Software. You cannot buy tickets through this site, but it can provide a bird's-eye view on existing fares. When you select "See calendar of lowest fares," you can see an entire date range of prices.

For tips on purchasing tickets, check out these articles:

- [Hacker Fares Sound Sketchy?](#)
- [How to Beat High Airfares](#)
- [Booking a Flight the Frugal Way](#)
- [Cheapest Days to Fly and Best Time to Buy Airline Tickets](#)

Total Costs

In sum, if you watch your wallet you can attend the full conference from \$0 to \$515:

- Conference: Free (as a volunteer)
- Airfare: \$150-400
- Room: \$115 (youth hostel) or \$160 (cheap hotel)

Of course, you will have to spend some money on food. But if you are on a budget, you can bring enough to supplement the meals and snacks included at the conference.

[Top](#)

Employment & Training

Introduction to Documenting APIs, SDKs, and Web Services

by Gwendolynne Barr

What is an API? What does it mean to document an API? And how can you get into this niche if you don't have a computer science degree? To investigate these questions, I interviewed four technical writers who specialize in documenting APIs and API-related products such as web services and software development kits (SDKs).

- Denise Green documents SDKs for Gracenote, a division of Sony.
- Ed Marshall runs his own consulting firm, [Marshall Documentation Consulting](#).
- Richard Smith works at Netflix and documents an SDK for TV video streaming.
- Susan Gallagher is the author of the whitepaper, "[Yesterday API was just another acronym: today I have to document one!](#)".

I also spoke with Charles Hoffman, Sunil Vemulapalli, and Alan Michaels—all software developers at Thomson Reuters, who reviewed my article and gave me advice. I am grateful to all of them for patiently explaining the world of APIs and what their documentation is all about.

APIs on the Rise

An API is the means by which two software systems talk to each other. A web service is a type of API often called a web API. And an SDK is a set of tools that include the API to assist the application developer in using the API.

These days, offering an API to the public is all the rage. [Twitter](#) does it, [Facebook](#) does it, even [BART](#) does it. The question is, what are APIs, *exactly*, and why are so many companies creating them? The Programmable Web recently counted [5,000 APIs](#) and noted that "like the Wild West, there's an "anything goes" approach" to API development.

Back in 1999, Susan Gallagher wrote that encapsulation and modularity in object-oriented programming (OOP) "are the forces that drive the development of APIs and SDKs." Encapsulation allows for code to be hidden behind an interface; modularity allows for it to be arranged in classes hierarchically and easily packaged.

In other words, APIs let a company share the power of its software implementation without revealing the code itself. Enterprising developers use these APIs to create applications or "apps" that leverage the functionality of those implementations.

But OOP has been around for a long time. Why the sudden increase? Richard Smith thinks one reason is "the sharing of services between organizations." Everyone benefits when services are shared: application developers gain access to a commercial service, end users have a tightly integrated app, and both companies—the one offering the API and the one using it—make money.

Definitions

Before we look at what the job of API documentation entails, let us clarify the terms.

What is an API?

An [Application Programming Interface](#), or API, defines how one software component can communicate with another. "It's a hook," said Richard Smith, "that gives you access to someone's software library." That hook—that interface for programming applications—can be a set of methods that you call to access the functionality of a program.

More simply, when you physically click a button in Google Maps, you are "calling" some method in their code. Methods in computer code are like verbs in a human language. They define the actions of a program. The code that comprises Google Maps is loaded with such methods for the various actions you can take: search, get directions, zoom in and out, print, link, etc.

When another program wants to use Google Maps, it cannot physically press a button. It needs to call those methods within the code itself; that is, it needs to call them *programmatically*. And it can only call them if Google opens them to the public. Those shared methods are APIs—you insert them in your own application code in order to call the implementation code of Google Maps.

Say you want to create an app that displays, in Google Maps, all the coffee houses within a certain radius. You *could* attempt to write the entire program from scratch, recreating Google Maps code and mashing it with your coffee shop feed. But that would be a waste of time because [Google Maps provides its API](#) for free, and hence, programmatic access to the functionality encapsulated underneath.

In fact, someone has already created such an app called [CoffeeSeeker](#). The website, [Google Maps Mania](#), lists tons of other Google Maps mashups. The video, [What is a Mashup?](#), by David Berind at ZDNet, is a good introduction to the concept of APIs and mashups.

What is a Web Service?

A [web service](#) is a specific kind of API and is often referred to as a web API. The big difference is that rather than implement the methods of another system in your own code, you send requests to that system. Put another way, a web service is an API that uses a request and response architecture and is usually, though not always, executed over a network such as the internet or an intranet.

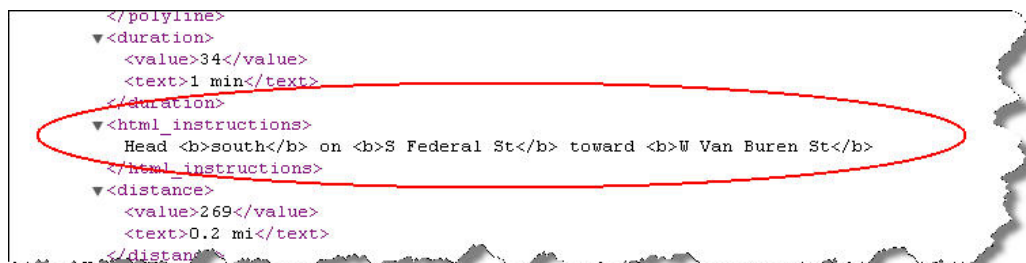
But how do you know what to request? And how to request it? With a traditional API, you get an actual package of methods. With a web service, you generally refer to a template from the web service provider in order to make a valid request.

This template is written in a language called [Web Services Description Language](#) (WSDL), a variant of XML, and is usually referred to as "the WSDL" [pronounced WIZ-dull]. Using this template, you create requests, perhaps in XML, that you send to the provider. The provider, in turn, sends a response, in the same data format, with the data you requested. [Amazon](#) is a web service provider and you can see one of their WSDLs here: <http://soap.amazon.com/schemas2/AmazonWebServices.wsdl>.

Not all web services use a WSDL, however. For example, to make requests to the [Directions \(web\) API](#) in Google Maps, you simply pass your request in a URL such as this one for directions from Chicago to LA via [Route 66](#):

<http://maps.googleapis.com/maps/api/directions/xml?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&sensor=false>

When you paste the URL above in a browser, Google sends back an XML response with directions.

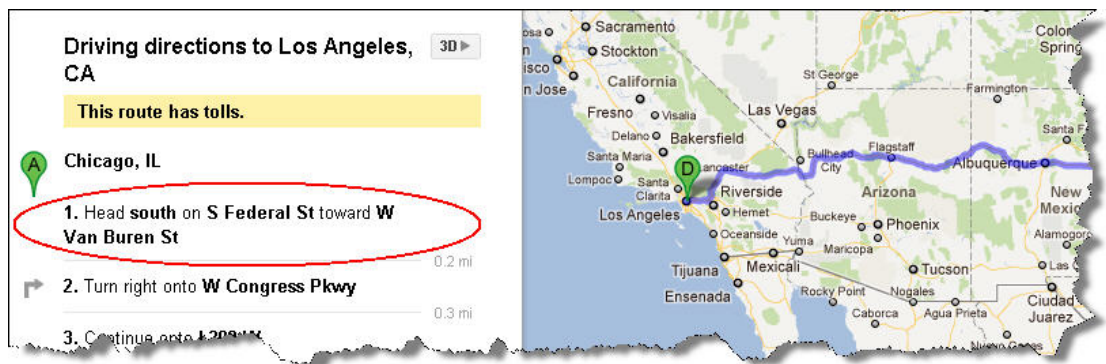


```

</polyline>
  <duration>
    <value>34</value>
    <text>1 min</text>
  </duration>
  <html_instructions>
    Head <b>south</b> on <b>S Federal St</b> toward <b>W Van Buren St</b>
  </html_instructions>
  <distance>
    <value>2.69</value>
    <text>0.2 mi</text>
  </distance>

```

Search the XML response for "html_instructions" and you will see the [same route as on the Google Maps GUI](#).



By parsing the data in a web service response, you can extract the data you need to build an application.

Web services come in different flavors. For example, to send information, they can use one of several protocols such as REST, SOAP, JavaScript, and XML-RPC. They can also have one of several data formats such as XML, JSON, RSS, and HTML.

For more context on web services, see: 1) [Introduction to Web Services](#), by Hasan Mir at ZeroToProTraining, 2) Janet Wagner's article, [The Increasing Importance of APIs in Web Development](#), and 3) Brian Suda's thesis, "[SOAP Web Services](#)". Of course, don't forget [Programmable Web](#).

What is an SDK?

A [Software Development Kit](#) (SDK) is a tool, or set of tools, that help an application developer use an API. Some SDKs are as simple as the API itself--a library of API calls and classes. Others come with fancy tools to help you build programs in a particular environment. A typical SDK might include one or more APIs, an API reference manual, and sample code.

An SDK is sometimes confused with an Integrated Development Environment (IDE) such as Visual Studio or Eclipse. An IDE is an environment that makes programming easier, much like Word and FrameMaker make technical writing easier (than in Notepad). An SDK is the content with which you develop a particular application, whether in an IDE or with a text editor such as Vim. In sum, an IDE is a programming environment; an SDK is the API plus materials to help you understand and use that API.

Some SDKs come with an IDE plug-in so that you can use that particular SDK within your favorite IDE. The Android SDK, for example, comes in a [vanilla version](#) (that you can use in Vim, etc.) or as an [Eclipse plug-in](#) that lets you develop programs for the Android platform within Eclipse.

On the Job

Let us take a look at APIs and SDKs in the context of two real companies. I talked to Denise Green at [Gracenote](#) and Richard Smith at [Netflix](#).

Working at a B2B

Gracenote, formerly known as CDDDB, is an independent subsidiary of Sony that sells access to its large databases of metadata on CDs (and also DVDs) via APIs. Their music metadata, for example, is used in products such as Apple iTunes and CarStars which recognize and recommend songs based on your preferences.

For example, Gracenote has some code—that code has an implementation (hidden, for private use only) and an interface (open, for public use by authorized developers). The developers at Gracenote *implement* the Gracenote API. The developers at Apple and Ford *use* the Gracenote API.

Suppose Gracenote has an API call named `track.getArtist()` that returns the artist for a given track. The Ford code uses `track.getArtist()` to get an artist; the Gracenote code contains the logic to query some database and return the artist associated with that track.

Ford doesn't know, and doesn't care how Gracenote goes about getting the artist; it just works. Gracenote keeps the implementation hidden, but exposes the interface (API) to Ford.

It is Denise's job to document Gracenote SDKs so that the software developers at Apple and Ford know how to use the appropriate API. This means she has to describe methods and write sample code using those methods. And she has to do this for several languages such as C++, Java, and Objective C, as Gracenote provides SDKs for multiple platforms.

Delivering the API and the App

We all know that Netflix rents DVDs and streaming video. What you might not know is that they also create APIs. TV manufacturers, for example, embed the Netflix streaming software into their own software so that you can access the Netflix movie catalog from your TV. Richard documents the SDK for that streaming software.

Netflix takes their SDKs a step further, and for some of their TV partners, they write the application software too. That is, they ship both the SDK, which their partners port, plus the "app" that uses the API in that SDK. They often do this because TV manufacturers work on extremely tight schedules. In retail, it's all about seasonal shopping cycles, and you can't reschedule Christmas, said Richard. For those who want to create their own differentiated app, Netflix provides design rules and guidelines.

Like Denise, Richard has to document SDKs for different platforms; but in his case, platform means the hardware architecture and components the TV makers use. Each TV platform is different so that what a developer needs to do to make the Netflix software work on that TV is also different. Richard has to understand each one.

Nuts and Bolts of Documenting APIs

The heart of documenting an API is commenting the code base by explaining what each method does, then generating documentation from those comments in the code. That is, you enter markup language in the code base, then run tools like Javadoc or Doxygen against it to generate documentation.

Of course, getting permission to touch the code requires that the developers trust you. As Ed told me, you might need to prove that you are capable of entering doc comments without breaking the build. If you are competent, you may find yourself helping out in other ways such as recognizing and reporting bugs in the code itself.

API Documentation Deliverables

APIs need to be documented so that application developers know how to use them. A developer needs to know the *structure* of the API, the *functionality* of the API, and *how to use* individual software components, say Jim Bisso and Viki Maki in their book, [Documenting APIs](#).

Joshua Bloch at Google thinks that a *well-written* API can be used without documentation; and yet, he says, "documentation matters." From the [PDF](#) of his talk, [How to Design a Good API and Why it Matters](#): "Document every class, interface, method, constructor, parameter, and exception."

Traditionally, the two most important deliverables include a reference guide and a programmer's guide. Again, from [Documenting APIs](#): the reference guide describes "the constituent parts of the API, such as the classes and methods." The programmer's guide is more conceptual and "provides a description of how to use the API" and "often includes tutorials and practical examples."

Charles Hoffman, a software developer at Thomson Reuters where I work, told me what he looks for in API documentation:

Reference Guide

- What the method does
- What data should be provided to the method and what data will be returned
- When the method should be used
- When the method should not be used
- Internal states being maintained, if any
- Side effects, if any
- Possible errors and exceptions and how should they should be handled

Programmer's Guide

- Concepts of use
- Overview of the API and service being manipulated
- Examples of initialization, use, and shutdown
- Examples of common tasks

Richard Smith told me that today, more and more companies "are producing the same materials in topic-based documentation sets" rather than as a traditional "guide." However you deliver your documentation, Ed emphasized that developers expect it to be complete, accurate, and well-organized.

The API Writing Process

While the approach to each job depends upon several factors, such as whether or not you receive a requirements specification, who the developers are, etc., the general process is something like this:

1. Gather source documents (code, functional specs, WSDL, bug reports)
2. Interview Subject Matter Experts (SMEs).
3. Document use of each class, method, etc.
4. Write code samples and tutorials.
5. Meet with developer to refine understanding.
6. Test and edit documentation.
7. Generate and deploy documentation set.

API writing is an iterative process of reading code, talking to SMEs, and writing stuff down.

Richard emphasized that he likes to interview multiple stakeholders before writing. He doesn't look at the developer as the only SME, and if possible, starts with the API consumer.

Ed added that he likes to check the code versioning system for code changes and the bug tracking system for new API-related issues. If possible, he also likes to install and use the product.

Susan mentioned running the doc tool itself—when you build the code from a tool such as Doxygen, you can see if the developer added or removed comments.

How Can You Become an API Writer?

I asked my interviewees if they thought that a computer science degree is necessary for becoming an API writer. Of the four, only Denise has a computer science degree. So the answer is no. What is required, of course, is the capacity and enthusiasm to learn about computers and their associated fields of knowledge such as hardware, software, programming, networking, etc.

If you have the desire to get into API writing, here are some of the things you can do to get started:

Read Literature on Documenting APIs. A terrific resource is the online, [Developer Support Handbook](#), by Pamela Fox at Google. She dedicates the first chapter to [API documentation](#).

The only book I know of that specifically addresses API writing is "[Documenting APIs: Writing Developer Documentation for Java APIs and SDKs](#)" by James Bisso and Victoria Maki. To get it, [contact Jim himself](#). This book was published in 2006 and my understanding is that a new version is going to be published sometime this year (2012).

[Programmable Web](#) is a website that lists tons of APIs and publishes an informative [blog](#). Another blog, directed at developers, is [The Amiable API](#) which has a post specifically on [Writing helpful API documentation](#).

The LinkedIn group, [API Documentation](#), can be useful. Also, don't forget to scan other blogs. You can find useful posts such as this one: [Writing an API Reference guide for iOS](#).

Read API documentation. Read the developer websites mentioned throughout this article. It is the very stuff you are going to be writing yourself. Google's API and web service pages might be overwhelming if you are just starting out. Twitter presents their APIs more simply: <https://dev.twitter.com/docs/api>.

Play with API documentation tools. The following tools allow you to document the code and generate help documentation from it.

- [Doxygen](#)
- [Javadoc](#)
- [Doc-To-Help](#)
- [Doc-O-matic](#)

Take courses. If you don't know which courses to take first, refer to your chosen school's certificate programs for guidance. For example, City College of San Francisco offers a range of certificates in both computer science and computer networking. And many of the courses are online.

- [CCSF Computer Science Certificates](#)
- [CCSF Computer Networking Certificates](#)

Take advantage of open courseware. One of the more famous open courseware (OCW) sites is that by MIT: <http://ocw.mit.edu/index.htm>. A great place to start is with their entry-level course on [structured programming using Python](#). This course is not about Python per se; rather, Python is used as a tool to teach the principles of computer science and the art of computing. For more OCW sites, refer to the open courseware consortium: <http://www.ocwconsortium.org/en/courses/ocwsites>

Teach yourself to program. A neat startup called Codecademy helps you learn to program in JavaScript for free at <http://www.codecademy.com>. They have plans to add other languages such as Python and Java. In addition to providing courses, they also have developing environments for Ruby, Python, and JavaScript in which you can practice without installing those platforms yourself: <http://labs.codecademy.com/>.

Use an API and develop an application. Developing an app shows a potential employer that you understand code, how APIs work, and the overall process of implementing an application. Richard told me that the [Twitter API](#) is one of the easiest to use. To get started, you generally have to register at that site in order to get a key for access to the API. If something goes wrong, they need to know who you are. There is no reason to be afraid, even if you are just learning how to code. There are no "software developer" police so go ahead and register.

Gwendolynne Barr is a technical writer at Thomson Reuters in San Francisco.

[Top](#)

Upcoming Programs

March: Plain Language in Government

Speaker: Bruce Poropat

Date: Wednesday, March 14, 2012, 7:00-8:30pm

Location: Highlands Country Club, 110 Hiller Drive, Oakland, California

Program

Plain language conversion (PLC) is a growing field as government agencies, companies, and other institutions respond to legal and policy requirements to convert dense jargon into clear, straight-forward language. Plain language simply means language that conforms to a set of principles that conveys information in the clearest, most efficient way. It stands in contrast to legalese or jargon-laden institutional prose.

In this presentation, you will discover that plain language and good technical writing essentially share the same goal: inform the reader as succinctly and clearly as possible.

What you'll learn:

- What plain language really means
- What plain language means to today's governments, industries, and institutions
- How to write clear, concise prose that meets plain language objectives
- About career opportunities in plain language

Speaker

Bruce Poropat, a Bay Area-based contract technical writer, has authored online and print documentation for the University of California, Wells Fargo, Charles Schwab, Williams-Sonoma, ERG, ZipRealty.com, and many others. He has worked on plain language conversion projects for the California Department of Transportation (Caltrans) and the Port of Oakland.

April: Ethnographic Field Research: Find out what users really need

Speaker: Nicki Davis

Date: Wednesday, April 11, 2012, 7:00-8:30pm

Location: Highlands Country Club, 110 Hiller Drive, Oakland, California

Program

Whether you call it ethnographic research, contextual inquiry, or field studies, there's no better way to find out who your users are and what they need. By observing users in action in their normal working environment, you'll learn about needs that can't be expressed verbally. Best of all, this kind of information doesn't just benefit technical communicators; it can help software companies provide better products.

Nicki Davis will present a case study in which ethnographic research helped to reduce the scope of a new product by 50%, while providing functionality that was missing in the existing legacy product.

Speaker

Nicki Davis is a user interface writer with a strong technical background and experience in user experience design. She works at OSIssoft, LLC in San Leandro and is the treasurer of the Berkeley Chapter of STC.

[Top](#)

Meeting Logistics

The Berkeley STC meets the second Wednesday of every month. Each meeting consists of an optional dinner, short business discussion, and an hour-long program on a topic of interest to technical communicators. All are welcome.

Pricing

We use three-tiered pricing:

- **Lowest prices:** Anyone for whom the higher prices are a hardship--students, unemployed, underemployed. This appears on our website as "student/low income." Anyone can choose to pay this price. We trust that nobody will abuse the privilege.
- **Discounted prices:** STC members.
- **Regular prices:** Non-members (except those who choose the student/low income price).

Former STC members who intend to renew when their finances improve can choose whichever of the three rates they feel most comfortable paying. We do not want anyone to feel shut out of the STC Berkeley community.

[Reservations](#) must be made at least one day prior to the meeting. Walk-ins also welcome, however, dinner may or may not be available. We order dinner for the number of reservations plus a few for walk-ins.

Program and Dinner (Reservation / Walk-in)

- Students/Low Income: \$6/\$10
- Members: \$12/\$16
- Non-members: \$20/\$24

Program Only (Reservation / Walk-in)

- Students/Low Income: \$3/\$10
- Members: \$6/\$10
- Non-STC-Members: \$10/\$14

Special cost notes

Non-members are always welcome to STC meetings at non-member rates.

Non-member students are welcome at the *member* student rates.

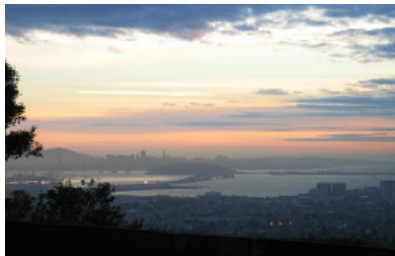
All members of the San Francisco Chapter of the IABC are welcome to register for Berkeley STC General Meetings at the member price by midnight on the day before the meeting.

Location and Directions

Highlands Country Club 110 Hiller Drive Oakland, California. [\[Google Maps\]](#)

Information at <http://www.stc-berkeley.org/MonthlyMeeting/directions.shtml>

If you need a ride from BART, email rides@stc-berkeley.org at least one day prior to the meeting.



View from the Highlands Country Club
Photo courtesy of Rhonda Bracey

[Top](#)

Local STC Chapters

Berkeley: www.stc-berkeley.org

East Bay: www.ebstc.org

North Bay: www.stc-northbay.org

Sacramento: www.stcsacramento.org

San Francisco: www.stc-sf.org

Silicon Valley: www.stc-siliconvalley.org

[Top](#)

Other Organizations

American Medical Writers Association (AMWA) of Northern California. Meets periodically at various Bay Area locations. www.amwancal.org

American Society for Training and Development, Mount Diablo Chapter. Meets monthly in Danville. <http://mtdiabloastd.org>

American Society of Indexers, Golden Gate Chapter. <http://www.asindexing.org/i4a/pages/index.cfm?pageid=3616#golden>

Association for Computing Machinery. <http://www.acm.org>

Association for Women in Computing. <http://www.awc-hq.org>

International Association of Business Communicators, San Francisco chapter. <http://sf.iabc.com>

National Writers Union (UAW). A labor union for freelance writers of all genres. www.nwu.org

Northern California Science Writers' Association. Quarterly meetings & other events. www.ncswa.org

Writer's UA. <http://www.writersua.com>

[Top](#)

Ragged Left is published four times a year at (roughly) the beginning of every quarter.

[Top](#)