## **Application programming interface**

An **application programming interface** (**API**) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as POSIX or vendor documentation such as the Microsoft Windows API, or the libraries of a programming language, e.g. Standard Template Library in C++ or Java API.

An API differs from an application binary interface (ABI) in that the former is source code based while the latter is a binary interface. For instance POSIX is an API, while the Linux Standard Base is an ABI. [1]

## Language used

An **API** can be:

- language-dependent, meaning it is only available by using the syntax and elements of a particular language, which
  makes the API more convenient to use.
- language-independent, written so that it can be called from several programming languages. This is a desirable
  feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote
  procedure calls or web services. For example, a website that allows users to review local restaurants is able to
  layer their reviews over maps taken from Google Maps, because Google Maps has an API that facilitates this
  functionality. Google Maps' API controls what information a third-party site can use and how they can use it.

The term API may be used to refer to a complete interface, a single function, or even a set of APIs provided by an organization. Thus, the scope of meaning is usually determined by the context of usage.

## **Detailed explanation**

An API may describe the ways in which a particular task is performed. In procedural languages like C language the *action* is usually *mediated* by a function call. Hence the API usually includes a description of all the functions/routines it provides. For instance: the math.h include file for the C language contains the definition of the function prototypes of the mathematical functions available in the C language library for mathematical processing (usually called libm). This file describes how to *use* the functions included in the given library: the function prototype is a signature that describes the number and types of the parameters to be passed to the functions and the type of the return value. The *behavior* of the functions is usually described in more details in a human readable format in printed books or in electronic formats like the man pages: e.g. on Unix systems the command man 3 sqrt will present the signature of the function sqrt in the form:

```
#include <math.h>
double sqrt(double X);
float sqrtf(float X);

DESCRIPTION
DESCRIPTION
sqrt computes the positive square root of the argument...

RETURNS
On success, the square root is returned. If X is real and positive...
```

That means that the function returns the square root of a positive floating point number (single or double precision) as another floating point number. Hence the API in this case can be interpreted as the collection of the

include files used by the C language and its human readable description provided by the man pages.

#### **Documentation**

Many program development environments provide the documentation associated with an API in some digital format, e.g. perl comes with the tool perldoc:

python comes with the tool pydoc:

```
$ pydoc math.sqrt
Help on built-in function sqrt in math:
math.sqrt = sqrt(...)
    sqrt(x)
    Return the square root of x.
```

ruby comes with the tool ri:

Java comes with the documentation organized in HTML pages (JavaDoc format), while Microsoft distributes the API documentation for its languages (Visual C++, C#, Visual Basic, F#, etc...) embedded in Visual Studio's help system.

#### **API** in object-oriented languages

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enums...).

The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class *interface*). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions.

More generally, one can see the API as the collection of all the *kinds* of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a *technical detail* of how the behavior is implemented.

For instance: a class representing a Stack can simply expose publicly two methods push() (to add a new item to the stack), and pop() (to extract the last item, ideally placed on top of the stack).

In this case the API can be interpreted as the two methods pop() and push(), or, more generally, as the *idea* that one can use an item of type Stack that implements the behavior of a stack: a pile *exposing* its top to add/remove elements. The second interpretation appears more appropriate in the spirit of object orientation.

This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language) API include the interface Serializable, which is a marker interface that requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be *saved* (serialized) at any time (this is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device).

Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation.<sup>[2]</sup>

In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods.

In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation.

The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

#### **API libraries and frameworks**

An API is usually related to a software library: the API describes and prescribes the *expected behavior* while the library is an *actual implementation* of this set of rules. A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface.

An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the *access* to the behavior *built into the framework* is mediated by extending its content with new classes plugged into the framework itself. Moreover the overall program flow of control can be out of the control of the caller, and in the hands of the framework via inversion of control or a similar mechanisms.<sup>[3]</sup>

#### **API** and protocols

An API can also be an implementation of a protocol.

In general the difference between an API and a protocol is that the protocol defines a standard way to exchange requests and responses based on a common transport and agreeing on a data/message exchange format, while an API (not implementing a protocol) is usually implemented as a library to be used directly: hence there can be no *transport* involved (no information physically transferred from/to some remote machine), but rather only simple information exchange via *function calls* (local to the machine where the elaboration takes place) and data is exchanged in formats expressed in a specific language. [4]

When an API implements a protocol it can be based on proxy methods for remote invocations that underneath rely on the communication protocol. The role of the API can be exactly to hide the detail of the transport protocol. E.g.: RMI is an API that implements the JRMP protocol or the IIOP as RMI-IIOP.

Protocols are usually shared between different technologies (system based on given computer programming languages in a given operating system) and usually allow the different technologies to exchange information, acting as an abstraction/mediation level between the two *worlds*. While APIs can be specific to a given technology: hence

the APIs of a given language cannot be used in other languages, unless the function calls are wrapped with specific adaptation libraries.

#### Object API and protocols

An object API can prescribe a specific object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message sent to a remote system.

When a message is exchanged via a protocol between two different platforms using objects on both sides, the object in a programming language can be transformed (marshalled and unmarshalled) in an object in a remote and different language: so, e.g., a program written in Java invokes a service via SOAP or IIOP written in C# both programs use APIs for remote invocation (each locally to the machine where they are working) to (remotely) exchange information that they both convert from/to an object in local memory.

Instead when a similar object is exchanged via an API local to a single machine the object is effectively exchanged (or a reference to it) in memory: e.g. via the memory allocated by a single process, or among multiple processes using shared memory or other sharing technologies like tuple spaces.

#### API sharing and reuse via virtual machine

Some languages like those running in a virtual machine (e.g. .NET CLI compliant languages in the Common Language Runtime and JVM compliant languages in the Java Virtual Machine) can share APIs.

In this case the virtual machine enables the language interoperation thanks to the common denominator of the virtual machine that abstracts from the specific language using an intermediate bytecode and its language binding.

Hence this approach maximizes the code reuse potential for all the existing libraries and related APIs.

#### Web APIs

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services towards more direct Representational State Transfer (REST) style communications. Web APIs allow the combination of multiple services into new applications known as mashups.

#### Web use to share content

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

- 1. Photos can be shared from sites like Flickr and Photobucket to social network sites like Facebook and MySpace.
- 2. Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.
- 3. Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs.
- 4. Video content can be embedded on sites which are served by another host.
- 5. User information can be shared from web communities to outside applications, delivering new functionality to the web community that shares its user data via an open API. One of the best examples of this is the Facebook Application platform. Another is the Open Social platform. [7]

## **Implementations**

The POSIX standard defines an API that allows a wide range of common computing functions to be written in a way such that they may operate on many different systems (Mac OS X, and various Berkeley Software Distributions (BSDs) implement this interface); however, making use of this requires re-compiling for each platform. A compatible API, on the other hand, allows compiled object code to function without any changes to the system implementing that API. This is beneficial to both software providers (where they may distribute existing software on new systems without producing and distributing upgrades) and users (where they may install older software on their new systems without purchasing upgrades), although this generally requires that various software libraries implement the necessary APIs as well.

Microsoft has shown a strong commitment to a backward compatible API, particularly within their Windows API (Win32) library, such that older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode". [8]

Apple Inc. has shown less concern, breaking compatibility or implementing an API in a slower "emulation mode"; this allows greater freedom in development, at the cost of making older software obsolete.

Among Unix-like operating systems, there are many related but incompatible operating systems running on a common hardware platform (particularly Intel 80386-compatible systems). There have been several attempts to standardize the API such that software vendors may distribute one binary application for all these systems; however, to date, none of these have met with much success. The Linux Standard Base is attempting to do this for the Linux platform, while many of the BSD Unixes, such as FreeBSD, NetBSD, and OpenBSD, implement various levels of API compatibility for both backward compatibility (allowing programs written for older versions to run on newer distributions of the system) and cross-platform compatibility (allowing execution of foreign code without recompiling).

## Release policies

The two options for releasing API are:

- Protecting information on APIs from the general public. For example, Sony used to make its official PlayStation 2 API available only to licensed PlayStation developers. This enabled Sony to control who wrote PlayStation 2 games. This gives companies quality control privileges and can provide them with potential licensing revenue streams.
- 2. Making APIs freely available. For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa, so that software can be written for their platforms.

A mix of the two behaviors can be used as well.

#### **APIs and Copyrights**

In 2010 Oracle sued Google for having distributed a new implementation of Java embedded in the Android operating system. <sup>[9]</sup> The litigation concerns also the fact that Google did not acquire any permission to reproduce the Java API. Instead a similar permission was given to the OpenJDK project. However, Hon. William Alsup ruled on May 31, 2012, that APIs cannot be copyrighted.

## **API** examples

- · ASPI for SCSI device interfacing
- · Carbon and Cocoa for the Macintosh
- DirectX for Microsoft Windows
- EHLLAPI

- · Java APIs
- ODBC for Microsoft Windows
- · OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran
  on many architectures, including Unix and Microsoft Windows platforms.
- Simple DirectMedia Layer (SDL)
- Talend integrates its data management with BPM from Bonita Open Solution

### Language bindings and interface generators

APIs that are intended to be used by more than one high-level programming language often provide, or are augmented with, facilities to automatically map the API to features (syntactic or semantic) that are more natural in those languages. This is known as language binding, and is itself an API. The aim is to encapsulate most of the required functionality of the API, leaving a "thin" layer appropriate to each language.

Below are listed some interface generator tools which bind languages to APIs at compile time.

- SWIG open-source interfaces bindings generator from many languages to many languages (Typically Compiled->Scripted)
- F2PY:<sup>[10]</sup> Fortran to Python interface generator.

#### References

- [1] Stoughton, Nick (April 2005). "Update on Standards" (https://db.usenix.org/publications/login/2005-04/openpdfs/standards2004.pdf) (PDF). USENIX. . Retrieved 2009-06-04.
- [2] Bloch, Joshua (2008). "Effective Java (2nd edition)" (http://java.sun.com/docs/books/effective/). Addison-Wesley. pp. 259–312. ISBN 978-0-321-35668-0.
- [3] Fowler, Martin. "Inversion Of Control" (http://martinfowler.com/bliki/InversionOfControl.html). .
- $\label{eq:comcgi} \begin{tabular}{ll} \begin$
- [5] Benslimane, Djamal; Schahram Dustdar, and Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications" (http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso\_level1&path=dsonline/2008/09&file=w5gei.xml&xsl=article.xsl). *IEEE Internet Computing*, vol. 12, no. 5. Institute of Electrical and Electronics Engineers. pp. 13–15.
- [6] Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?" (http://www.pcworld.com/businesscenter/article/145039/so\_what\_is\_an\_enterprise\_mashup\_anyway.html), PC World,
- [7] "Dynamic Community content via APIs". October 26, 2009.
- [8] Microsoft (October 2001). "Run Older Programs On Windows XP" (http://www.microsoft.com/windowsxp/using/helpandsupport/learnmore/appcompat.mspx). Microsoft. p. 4. .
- [9] "Oracle and the End of Programming As We Know It" (http://www.drdobbs.com/jvm/232901227). DrDobbs. 2012-05-01. Retrieved 2012-05-09.
- $[10] \ \ "F2PY.org" \ (http://www.f2py.org/). \ F2PY.org. \ . \ Retrieved \ 2011-12-18.$

## **External links**

- What is an API? Your Guide to the Internet (R)evolution (http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf)
- How to design a good API and why it matters (http://lcsd05.cs.tamu.edu/slides/keynote.pdf)
- How to Write an API (http://www.lior.ca/publications/api\_design.pdf)
- LMAX Application Programming Interface (API) technology (http://www.lmax.com/trading-tech/api-trading)

## **Article Sources and Contributors**

 $\textbf{Application programming interface} \ \textit{Source}: \ \texttt{http://en.wikipedia.org/w/index.php?oldid=498937988} \ \textit{Contributors}: 213.121.101.xxx, 24.108.233.xxx, 24.93.53.xxx, 4483APK, 64.105.112.xxx, 24.108.233.xxx, 24.93.53.xxx, 24.93.xxx, 24.93.xx, 24.93.xx, 24.93.xx, 24.93.xx, 24.93.xx, 24.93.xx, 24.93.xx, 24.$ 90. Aa1bb2cc3dd4ee5, AaronL, Aarsalankhalid, AbdulKhaalid2, Adah, Addshore, Ae-a, Aeons, Aeternus, Ahunt, Abzahraee, Airolaneman, Alan d, Alex43223, Altaf.attari86, Altenmann, Amanuse, Ancheta Wis, Andre Engels, Andremi, Andres, Andrey86, Andy16666, Anteru, Apoltix, Arindra r, Arjayay, Arthur Davies Sikopo, Aruton, Ashamerie, Asydwaters, Atheken, Atreys, Aude, Auntof6, Avk15gt, Awg1010, Bamyers99, Bdesham, Beano, Bearcat, Betterusername, Bevo, Bhat sudha, Bikingviking, Blackcats, Bobo192, Boing! said Zebedee, Bookiewookie, Borgx, Boyprose, Brianski, Bryan Derksen, BryanG, Bryanmonroe, CYD, Calton, CanisRufus, Capricorn42, Chadsmith729, Chameleon, Chealer, Chicago god, Chiefcoolbreeze, Chituokol1, ClaudiaHetman, CloudNine, Colonoh, Consult.kirthi, Conversion script, Coolbloke94, Courcelles, Cybercobra, Cynical, DShantz, Damian Yerrick, Daniduc, Danja, Darklight, Davejohnsan Davemck, David Gerard, Davron, Dawidl, DeeKay64, Deepugn, Dennislees, Derek farn, Detnos, Diego Moya, Diomidis Spinellis, Dipskinny, Discospinster, Dmarquard, Download, Dqpeck, Dr Marcus Hill, Dreadstar, Drewmeyers, Dschach, Dsnelling, Dylan620, EVula, Ebessman, Ed Poor, Edcolins, Edwardkerlin, Efa, Egmontaz, Ehn, Elf, Ellmist, Eloquence, Enochlau, Enric Naval, Epbr 123, Eric Aebozo, Espoo, Excirial, Farrwill, Fieldday-sunday, Fitch, Frao, Freakimus, Frecklefoot, Frummer Than Thou, Funyill, Fæ, GRAHAMUK, Gak, Geoff Purchase, Giftlite, Graham 87, Greensburger, Griznant, HUB, Hadal, Harryboyles, Hashar, HenkeB, Heraclius, Hfastedge, Hmains, Humu, Husond, InShaneee, Infinitycomeo, Itai, Ivan007, Ixfd64, Izno, JHunterJ, JLaTondre, JWSchmidt, Jakuzem, JamesMLane, Jbolden 1517, Jengod, Jerryobject, Jesse V., Jidanni, Jitse Niesen, Jleedev, Jmclaury, JoeB34, John of Reading, John.n-irl, JohnBlackburne, JulesH, Julien, Julienj, K12u, Kbolino, Kernel Saunters, Kichik, Kickin' Da Speaker, Kim Bruning, King of Hearts (old account 2), KnowledgeOfSelf, Kocio, Kurdo777, Landon1980, Lavers, Lee Daniel Crocker, Lekshmann, Leoholbel, Liempt, Limbo socrates, Liorma, LizardJr8, Lockeownzj00, Looc4s, Lord Chamberlain, the Renowned, Loren.wilton, Loshsu, Lotje, Lupin, M5, Mac, Mange01, Manop, Martyn Lovell, Marudubshinki, Materialscientist, Mattknox, Mav, Mbeychok, Meldraft, Mflore4d, Michael Hardy, Michael Hodgson, Michaelas10, Miguel, Mihai cartoaje, Miketwardos, Minghong, Minirogue, Miohtama, MoreThanMike, Morg, Mountain, Mpbaumg, MrOllie, Mrh30, Mshivaram.ie22, Mwarf, Myc2001, Nanshu, Neelix, NewEnglandYankee, Nopetro, Notinasnaid, Nsda, Obradovic Goran, Ocean Shores, Ohspite, Old Guard, OlegMarchuk, Oulrij, Oxymoron83, Pascal.Tesson, PaymentVision, Peak, Pearll's sun, Pengo, Pepe.agell, Percede, Pgimeno, Philip Trueman, Phuzion, Piano non troppo, Plamka, Poweroid, Prari, Pwforaker, Queenmomcat, Quinet, Quux, Qwertyus, Qx2020, Qxz, R'n'B, RJHall, Rackspacecloud, Raise exception, Randomalious, RedWolf, Redlentil, Reemrevnivek, Renatko, Rich Farmbrough, RichMorin, Riluve, Rktur, Robert K S, Robneild, Rocastelo, Ronocdh, RoyBoy, Rs rams, Rudyray, Rwww, SERIEZ, SQGibbon, ST47, Sakhal, Salvatore Ingala, Sam Korn, Scohil, Scott Ritchie, Seth Nimbosa, Sfmontyo, Shlomif, SimonTrew, SirSandGoblin, Sj, Skeejay, Skysmith, Slady, Slurrymaster, SocialRadiusOly, Sodium, Soumyasch, Spencer, Spikey, SqueakBox, Stephenchou0722, SteveBaker, Steven J. Anderson, Suruena, Syvanen, Szajd, T-borg, Ta bu shi da yu, Tantrumizer, TastyPoutine, Tedickey, Teryx, Teutonic Tamer, The Anome, The Thing That Should Not Be, TheSoundAndTheFury, TheresaWilson, Thiotimoline, Thisisborin9, Tide rolls, Tijuana Brass, Tonyl, Torchiest, Transpar3nt, TrentonLipscomb, Troymccluresf, Tseayl1, Uriyan, Utype, Uzume, Varworld, Vikramtheone, Visvadinu, Vkorpor, Voidxor, WTRiker, WaltBusterkeys, Wapcaplet, Wavelength, Whatsnxt, Whitehorse 212, WikHead, Willy-os, Wjejskenewr, Wysprgr 2005, Xqt, Yaris 678, Yordan Georgiev, Yurik, Zachlipton, Zeno Gantner, Zhinz, ZimZalaBim, Zodon, 계정명뭘로하지, 771 anonymous edits

# License

Creative Commons Attribution-Share Alike 3.0 Unported //creativecommons.org/licenses/by-sa/3.0/