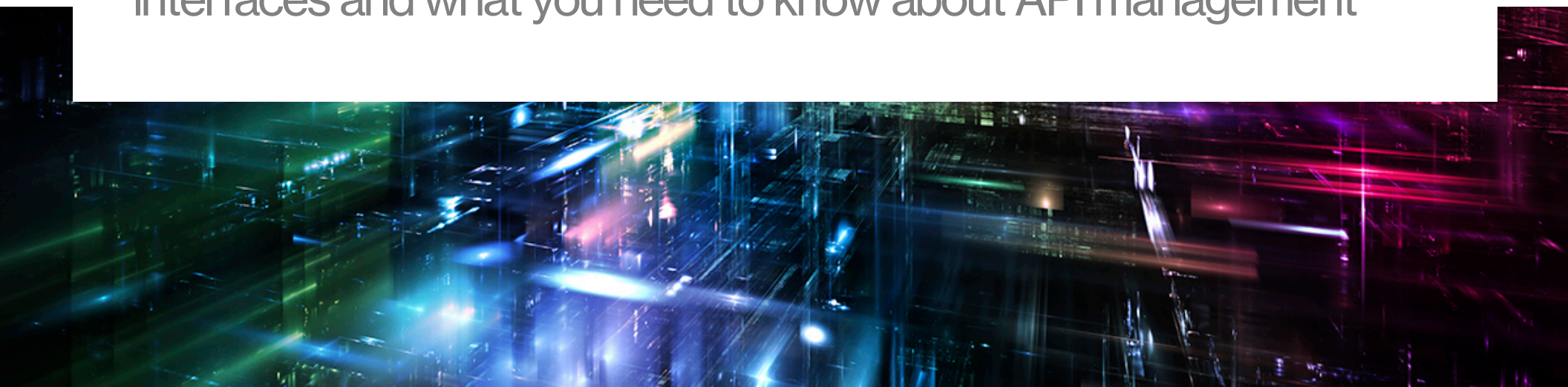
TechTarget | SearchSOA

E-guide

PRO+ Content

The Keys to Successful API Management

Your expert guide to the latest trends in application programming interfaces and what you need to know about API management



In this e-guide

In this e-guide:

This guide brings together a range of stories that highlight examples of successful API management, trends with the technology and key terms developers need to know.

Architects involved with integration have to be on top of the latest trends in API management and implementation. In our APIs 101 section, learn which tactics to employ from others' trials and tribulations. Find out how some organizations are making good use of their APIs.

Before you dive in to API management, it's important to know the basics. In our API trends in action section, receive tips around successful project planning and development.

In this e-guide

Articles within **APIs 101** starting on page 3:

- Effective portlet development means respecting the servlet API
- API development communities requires online and offline presence
- API project planning: Steps to add value in data, app integration
- API design: How to properly build an application program interface
- Digging into quality: API best practices, problems and advice
- Advice: API development calls for key principles, artistic thinking

Articles within **API trends in action** starting on page 26:

- Developers get into BaaS, mobile APIs for new opportunities
- What CIOs, developers should know about the 'API economy'
- Software developers demanding resource based, RESTful APIs
- Inside the world of Web APIs, the cloud and mobile technology
- NFL statistical data company scores touchdown with API service
- API management, servers enable mobile, Web gift services strategy

In this e-guide

APIs 101

API management: What you need to know

Before you dive in to API management, it's important to know the basics. Read on for tips around successful project planning and development.

Effective portlet development means respecting the servlet API

Cameron McKenzie, TheServerSide

For some bizarre reason, when experienced software developers jump into JSR-286 portlet development, their brains turn to mush, and they seem to throw away all of the basic rules of web based software development in Java. Portlet and portal based applications are more similar to standard servlet and JSP based applications than they are different, so when application developers embark upon building portlet based applications, it can't be stressed enough that all of the basic concepts they learned when developing web based Java applications apply equally when doing portlet based development.

In this e-guide

Servlet and JSP development rules still apply

Here are three of the rules of servlet and JSP development that no web developer would ever violate, but for some reason, when those web developers become portlet developers, these rules seem to go out the window:

1. You don't spawn new threads in a servlet, nor should you spawn them in a portlet

   The portal server is already threading client invocations. Spawning new threads will only wreak havoc on the portal server. Sure, Java 7 provides all sorts of great facilities for doing threading or implementing concurrent programs, but the body of a portlet is not the right place to be playing around with these new features.

2. You don't declare instance variables in a Servlet and you shouldn't declare them within a portlet either

   As far as instance variables go, there are times when they are appropriate, but those times usually involve extending the servlet or portal framework as opposed to creating a portlet or servlet that corresponds to a user story.

   When you are tempted to place an instance variable in a portlet, think instead about how the various scopes, such as the PortletSession or

In this e-guide

the PortletContext, might be a more appropriate place to maintain the data.

3. You don’t synchronize servlet methods, and you shouldn't synchronize portlet methods either

   Synchronizing methods is also a bad idea in a portlet. Synchronizing portlet methods can create a bottleneck in your applications. Move complex logic as far away from the portlet tier as possible, and if you really need to synchronize a method, synchronize it in that far away place, not in the portlet itself.

Portlet development isn't a revolutionary new way to develop applications. It's more of an evolutionary change, and that means building upon the patterns and best practices of servlet and JSP based development, and not reverting back to anti-patterns and poor web-based programming techniques.

Next section

In this e-guide

API development communities requires online and offline presence

Christine Parizo, SearchCIO

Learn how API development communities are expanding beyond the company setting.

Third-party developers aren't just uber-smart college kids coding in their dorm rooms, hopped up on Red Bull and Doritos. These days, a third-party developer could be an up-and-coming programmer or even a large company, and companies releasing APIs need to develop a third-party developer community to ensure the API's success. Companies can build API development communities in person, online and by offering incentives, according to API management companies and veterans.

Building the developer community is critical, particularly for cloud applications, according to experts.

Software as a Service (SaaS) by its nature needs to be accessible and provide benefits to users, whether or not the ideal solution comes from the API development company or a third party, said Kevin O'Brien, senior director of the AppConnect program at email marketing provider Constant

Content. "The benefits we've received [from the third-party developer community] have been immediate," he said.

The first step, however, is to have a compelling, easy-to-use application that results in a natural fit for third-party developers to work with the API, according to John Thomas, director of product management at San Francisco-based database software provider Embarcadero. "When you build a framework that is well-designed and architected and is extensible, it means you use languages and programming patterns that make it straightforward to take the base work and [create] something additional," he said.

For example, a framework may offer a lot of functions but not a specific one, like printing to a specific plotting printer. In that case, a third-party developer would see an opportunity to connect the framework to that type of printer, Thomas said.

Get personal – and offer prizes

Once the API is ready, one way to build a developer community in person is to host in-person contests like hackathons, said Alex Gaber, API evangelist at Washington, D.C.-based Layer 7 Technologies. Hackathons typically are weekend-long events dedicated to allowing third-party developers to create new ways of using a company's API. Often, companies will offer prizes based on specific challenges, he said.

In this e-guide

Another personal method to foster a third-party developer community is to run online contests for third-party developers, Gaber said. "We see big companies throw their hat into the ring," he said, citing Samsung's first place prize offering of $100,000 to the developer that could build the best integration with SDK tablets, or Netflix's movie recommendations algorithm contest that offered a $1 million purse to the winner. "Netflix ended up with a bunch of different solutions that were built, functioned and worked," he said.

These contests can attract more than just individual developers, according to Gaber. For example, a 20-person developer shop could submit an entry, or a large company that wants a better relationship with the contest sponsor may also throw its hat into the ring with a solution built around that company's API, he said.

Dedicated portals provide a community feeling

Meanwhile, companies looking to foster community online need dedicated portals for their third-party API developers, according to experts. "With API providers, every company in the world is going to have an API portal. These portals are where you really go to get access to these APIs," Gaber said.

Broadsoft has a typical portal since it launched its first API four years ago, complete with forums and documentation, according to Leslie Ferry, vice president of marketing at the Gaithersburg, Md.-based VoIP company. That community has grown to 5,000 members, and those members have come

In this e-guide

up with new uses for Broadsoft's API. For example, one company in New Zealand tied the API into their billing system and can send emails to remind customers that their accounts are past due, which not only is less invasive for the customer but also reduces the length of time the bill is outstanding by 50%. "Our community enables third-party developers and our own customers to create new processes," Ferry said.

Meanwhile, Constant Contact shares their insights from working with small businesses to third-party developers, alerting them to trends, according to Constant Contact's Kevin O'Brien. Utilizing newsletters, forums and webinars, Constant Contact provides developers with what they know to help the developers know where to start.

Offer incentives as well

To encourage third-party participation in the company's API development, offering incentives helps. Broadsoft offers an incubator program that provides seed funding for developers. By providing developers with money up front, Broadsoft is able to speed the developers' time to create a solution for a common request, according to Ferry.

Meanwhile, Constant Contact has a revenue-sharing program in place with its third-party developers, according to O'Brien. "We have a marketplace that we promote to customers," he said. This marketplace includes the solutions third-party developers build.

In this e-guide

No matter what, though, the developer experience should be as smooth as possible. According to O'Brien, a lesson learned is that, because of the volume of applications that a third-part developer can integrate, the developer's experience with the company's API should be straightforward and uncomplicated. "If they're making choices [about] which API to build on, it needs to reflect the functionality and be simple for them to use," he said.

API project planning: Steps to add value in data, app integration

Maxine Giza, SearchSoftwareQuality

API projects can be less than successful if proper planning isn't performed, according to experts.

While many organizations still keep data on-premises, many are also answering the siren's call to move data and applications to the cloud. In this scenario, data needs to move through various systems, and that is where application programming interfaces (APIs) can step in.

APIs aid in integration and interoperability because they act as a uniform space, allowing for communication between new and legacy systems. Poorly planned data and application integration and interoperability projects, however, stymie data flow and don't meet business needs.

In this e-guide

"Clearly defined APIs and clearly defined services help reduce 'spaghetti' code and allow for loose coupling across different components, improving interoperability," said James Governor, industry analyst at RedMonk.

Rushing into development without cross-team planning is the cause of most integration and API project failures, according to Governor and experts, as well as the recent MuleSoft 2013 Global Mobile, API and SaaS Enterprise Adoption Survey. In the latter, 300 IT workers also said they have either started or "felt strongly" that their organization should start to include an API strategy as part of its IT strategy. That plan should include collaboration between business, IT, development and users.

An organization's goals must come first in designing API implementations, said MuleSoft Founder and Vice President of Product Strategy Ross Mason. Failure to plan ahead has resulted or can result in wasted time and money.

"Even just a few years ago, developers were being asked to build APIs without having a good context of the business problem they were trying to solve," Mason said. "Before you write a line of code and get developers thinking about how to actually build the API, there is a planning phase. Many organizations still miss this planning phase of understanding [that] the longer-term strategy is for the API."

The business plan has to precede the technology project, Mason said. With the plan as a baseline, technology decisions -- such as building versus

In this e-guide

sourcing an API -- can be honed to deliver the right product. "The API is really a product: a product you are giving to third parties to access your information," he said.

Business and technology leaders should work together to determine the product goals for API-based projects. Sometimes organizations fail to keep everyone, from developers to partners, engaged in the project, said Evgeny Popov, CEO and co-founder of Apiphany, an API management and delivery platform company in Washington, D.C. If internal teams don't collaborate, the result is a product no one will use, largely because they don't understand it or there is a lack of support from one disenfranchised part of the business. "I've seen a lot of companies spend years thinking about and building an API, and then it doesn't work [properly]," he said.

Popov recommends a step-by-step, iterative approach to implementing an API strategy, so modifications and direction changes can be made before a final product is delivered.

There are several key features to keep in mind when making API decisions, such as the following:

- Design
- Security
- Availability of analytics data
- Scalability

In this e-guide

Scalability is particularly important, Popov said, because "when you pick the tool, you want to make sure it will not be a bottleneck in your API infrastructure."

While there isn't a 100% surefire way to avoid snags that may arise during the implementation of an API strategy, keeping these considerations in mind will help prevent an API project from becoming completely derailed.

API design: How to properly build an application program interface

Tom Nolle, CIMI Corporation

Don’t let your entire API project go awry. Knowing the applications well is just one way to nail the design and foster developer loyalty.

In a world of interactive componentized software, there's nothing more important than the application program interfaces (APIs) used to link components with each other, mobile devices and browsers. Build an API the right way and it helps assure functional integration and developer loyalty; do it the wrong way and compromise an entire project.

In this e-guide

There are three ways to help get on the right side of API building:

- Know the applications and constraints of use;
- Address component architecture and binding framework; and
- Ensure changes are handled gracefully.

APIs expose features and services to developers. The way an API is used and the spectrum of services represented is the primary design driver. One of the most significant mistakes developers and architects make in API building is ignoring a constituency. It's vital an API design be a good fit in the ecosystem of developers, languages and other APIs.

Common API design issues

The REST versus SOAP debate is an example of a constraint set for APIs. Where applications already depend on one or the other, new APIs should obviously conform. Less obvious is that most APIs are part of a trend toward componentization and exposure of features. The movement may take a set of APIs more in a REST or SOAP direction over time, so be sure to anticipate that migration.

Architects can easily be bitten by conforming to object architecture and binding framework. Picking the right API design is important because it's difficult for developers to use an interface that doesn't match the

architecture of the applications they're building. It should be noted that RESTful APIs typically represent resources whereas SOAP APIs represent remote processes or procedures.

Some protocol is likely used to bind APIs to API users and with Web applications, which are typically Hypertext Transfer Protocol (HTTP)/HTTPS. Using HTTP with a Hypertext Markup Language or Extensible Markup Language (XML) data format, or JavaScript Object Notation (JSON) and JavaScript on client devices, makes it easy to create graphical user interfaces from an API but may not be appropriate where browser access isn't the application's intention. Some applications and APIs may use a specific Transmission Control Protocol or User Datagram Protocol port rather than Web port 80. While this can help to keep API traffic separate from Web activity, it may have firewall/security implications that will demand special system configuration that either exposes the APIs or uses them remotely.

General API design rules

Most APIs can be viewed as syntax of verbs and nouns. For example, a sentence with a verb representing a requested action (get, put, delete) and the nouns signify arguments appropriate to the action. It's good practice to always generate a status/result variable that communicates error conditions or successful execution. The error conditions should be comprehensive enough to communicate problems unambiguously.

In this e-guide

The semantics of the API, meaning the syntax of the functions provided, are important because the ability of the API to convey its services and parameters clearly reduces developer errors. One critical point is that if the API represents a stateful service, the function semantics should be session-oriented (find-record, update-record, delete-record) where the stateful nature of the service is made clear.

It follows that if, as in this example, the update and delete functions operate on the previously located data element, then the update and delete functions do not supply their own data element keys; they'd be redundant and risk generating developer confusion. On the other hand, a stateless service must always provide all of the data since no session context could be inferred.

Common questions and issues

Syntax issues created by updates or changes to the API are often ignored. There are two sides to each API and the change process can pull them out of synchrony. Some architects will express a version variable in an API to ensure that both sides are expecting the same formats. At minimum, both the server and client side of an API should perform basic validation to protect against changes that create a mismatch of syntax so they don't contaminate information or crash applications.

In this e-guide

Another common question has to do with the data format. XML is the most general way of expressing parameters and exchanging information, and it is applicable to REST and SOAP interfaces. But XML processing is a heavyweight and most valuable for expressing structureless data. For REST, JSON has gained favor as being easier to use while providing some specific variable typing widely used and expected in API building. Where APIs exchange rigidly defined data elements, JSON is likely a better choice for RESTful exchanges.

API testing is often lumped into application lifecycle management processes. Some of it rightfully belongs there, but there should also be specific unit-test processes designed to validate APIs and establish that they'll perform gracefully even when data includes errors. The looser the data binding and typing of an API, the more risk there is of passing information that will result in a later error or crash. This is why it's important to adopt tight constraints on variables and to test each API with a range of data.

API build problems will destroy an application faster and more thoroughly than almost any other type of architecting error. Spending more time designing APIs to anticipate current error conditions and future changes will be time well spent.

In this e-guide

Digging into quality: API best practices, problems and advice

Alan R. Earls, Contributor - SearchSoftwareQuality

Planning and testing are crucial parts of API development, because a bad API can mean a longer development cycle and higher defect rate.

For many large services, an application program interface (API) has become the primary channel for reaching users. Services like SalesForce, NetFlix, Facebook and Twitter typically serve more requests through API calls than through their own front-ends for users. Considering many APIs carry such heavy burdens, it's surprising how many are rushed through development without adequate testing, according to IT architect Peter Hendriks and other API development experts.

API quality matters a lot, both for API consumers and implementers. "For consumers, a bad API means a longer development cycle and a higher defect rate; and in some cases, even a skills problem in the team -- a dependency on the one person that mastered the black art of calling the API correctly," said Hendriks, IT architect/competence center Java lead at Netherlands-based Infosupport.

In this e-guide

For API implementers, he said, API quality shows in the popularity of the API, as well as the volume of support calls or the amount of supporting documentation or training needed.

Fortunately, there are many approaches that yield better and more consistent results in API development. Those approaches include leveraging best practices at leading organizations, considering usability and deploying appropriate tools.

In the early days of the technology, APIs to manipulate Extensible Markup Language "were terrible," Hendriks said. Javascript Object Notation (JSON), an alternative data format, became popular very quickly, because APIs were very simple from day one. A similar situation exists with Spring versus Java Enterprise standards in the Java space, he noted. "Spring competed on API quality for the most part, causing radical API changes in the newer Java Enterprise standards in order to stay relevant," he said.

Les Hazlewood

A common mistake is publishing an API and then deciding it needs changes -- changes that might impact customers, according Les Hazlewood, CTO of San Mateo, California-based Stormpath Inc., a user management and authentication service for developers.

In this e-guide

"You want to be frugal when you release an API," he said. Make sure to learn about and follow all the best practices, even though an API is based just on an architectural style rather than a standard.

Peter Hendriks

For guidance in constructing REST APIs at Stormpath, Hazlewood looked at what was being done by Twitter, as well as less well-known but robust business-oriented APIs. He found the work of Roy Fielding very valuable. Fielding's doctoral dissertation describes Representational State Transfer (REST) as a key architectural principle of the World Wide Web.

Digging into API quality

Hendriks explained that in modern application development, most application code involves calling an API. The code uses APIs provided by the platform, such as the Java APIs, frameworks (Java Enterprise Edition APIs, for example), third party libraries (like log4j for logging), or external systems and services (Web Services, Twitter, Facebook, SalesForce, Google). "API quality determines how easy it is to learn, how productively can be used and how likely it is that mistakes will be made," Hendriks said. And, he added, "For a developer, API quality can make a night-and-day difference in their effectiveness."

In this e-guide

On top of that, the code that developers write is shaped by the APIs they use. "It is very difficult to write clean code once you need to consume a poorly designed API, but a beautiful API will almost automatically lead to nice API consumer code," Hendriks said.

According to Hendriks, libraries and frameworks dominate current application development. "Nobody builds anything from scratch anymore. It is much easier and more productive to use a good framework. This makes API design very important for application developers; virtually every code they write calls an API," Hendriks explained.

Since many capable frameworks are available nowadays, there is comparatively little competition for features. However, he added, the newest frameworks place heavy emphasis on API design. Their main competitive advantage is the ability to start fresh and fix design mistakes in established frameworks, making them easier and more productive to use, he said.

Hazlewood relies on three general principles that should be applied to creating a REST or JSON API.

1. Be sure you thoroughly understand the HTTP protocol and how it handles posts and puts requests, because "HTTP is the foundation that REST is built upon."

In this e-guide

2. Understand how to link between resources and code. "The Internet is at its massive, global scale because of linking … so linking and data types matter."

3. Determine how you are going to represent a single resource as compared to a group of resources. Although there are no formal standards, developers should look for best practices.

What's on the horizon for API design? For the Java world, at least, support for lambda expressions in Java 8 is a "big thing" for API design, according to Hendriks, because it makes it much easier to pass behavior to an API. "Until Java 8, this required a lot of boilerplate code with confusing semantics. Many API scenarios, such as user interfaces, message handling or parallel programming will now be much easier and less error prone," he explained.

Additionally, in Java 8, many other small improvements are made to static code checking and annotation support. That means tools like FindBugs and JSR305 can be used in more areas of the API than was previously possible, Hendriks said.

Next section

In this e-guide

Advice: API development calls for key principles, artistic thinking

Alan R. Earls, Contributor - SearchSoftwareQuality

During API development, it's important to keep in mind that the technology is used by programmers.

There are some general principles that should be used during API development. The first thing to consider is that application programming interfaces are used by programmers. "That means that API design is not just about structured software design, but also about human understanding," said Peter Hendriks, IT architect and Competence Center Java lead at Netherlands-based Info Support.

A good API supports a limited set of well-defined use cases or scenarios. Start out with creating example code that uses the API for all these scenarios, Hendriks said.

He explained that example code is the best way to get feedback on usability, and the code can later be used for tests and documentation. If example code is "really good," users will copy-paste or emulate examples closely. That's the "art" of API development, he said.

In this e-guide

Hendriks uses a usability model by professor Ben Shneiderman for assessing API usability. Shneiderman is the author of the book, *Leonardo's Laptop: Human Needs and the New Computing Technologies*. For Hendriks, the key elements of the usability model include:

- Learnability: How easy is it to accomplish basic tasks the first time?
- Efficiency: Once users learned the design, how quickly can they perform tasks?
- Memorability: When users return after a while, how easily can they re-establish proficiency?
- Errors: How many errors do programmers make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the design?

On the science side of API development, a variety of tools are available to help. For example, static analysis tools, such as FindBugs for Java code, can do a great job at detecting problematic API design and bad API usage.

Hendriks frequently uses JSR 305: Annotations for Software Defect Detection, a standardized library that provides simple annotations to declare basic API hints, such as optional nullable values or support for multithreading. "FindBugs can detect many problems based on these

In this e-guide

annotations; not just in API design or implementation, but also in API consumer code," Hendriks said. He finds FindBugs easy to use and helpful for preventing a lot of common mistakes.

Many code-quality tools, such as SonarQube, also provide statistics that apply to API quality. "The LCOM4 statistic, for instance, implies that a class may be divided into multiple classes," Hendriks said. Furthermore, he noted, basic limits on lines of code per class and limits on the number of parameters per method also keep API complexity down.

Next article

In this e-guide

API trends in action

The latest with APIs: Trends and effective use cases

Architects involved with integration have to be on top of the latest trends in API management and implementation. One of the best ways to know which tactics to employ is to learn from others' trials and tribulations. Read more to find out how some organizations are making good use of their APIs.

Developers get into BaaS, mobile APIs for new opportunities

Tom Nolle, CIMI Corporation

Backend as a Service and mobile APIs offer developers a whole new dimension of opportunity: cloud services to build the mobile apps of the future.

In this e-guide

Tom Nolle, president of CIMI Corp.

A new development in the application programming interface (API) market, Backend as a Service (BaaS), offers developers a whole new dimension of opportunity: cloud services to build the mobile applications of the future.

Mobility, mobile broadband and mobile apps are transforming the Internet and the way people communicate and use information. For developers, the mobile app represents an enormous, lucrative new market. From a business perspective, each of the mobile platforms has its own benefits and limitations.

Each also has its technical framework -- its own APIs. Mobile development platform APIs generally fall into two categories: platform APIs related to the mobile device's own operating system and middleware, and service APIs related to the access of Web-hosted material.

The new opportunity in APIs is Backend as a Service, an extension of the service API model. The goal of BaaS is to convert common and useful elements of mobile application logic -- storage, identity management, social network integration, photo enhancing -- into representational state transfer (REST) Web services that the application invokes as needed, making these services "back ends" to mobile apps.

In this e-guide

As a concept, BaaS is similar to Software as a Service (SaaS) and Platform as a Service (PaaS): It offers functionality as a Web service. With SaaS, what's offered is an application or application component. Salesforce.com's CRM is a good example. PaaS offers a collection of services designed to present a complete virtual operating system; Microsoft's Windows Azure is the best-known PaaS example. BaaS is somewhere in between; it offers PaaS-like features, but not as a complete programming platform.

Like SaaS, those features might perform business functions oriented to vertical markets or functionalities that can be used across industries. In all cases, though, BaaS aims to enhance mobile development.

Implementing BaaS

Success with BaaS requires a targeted business case and careful implementation to support mobile development practices. Most BaaS deployments today are horizontal -- they cover a variety of markets -- and deal with well-known issues such as social network integration. The problem with these types of BaaS implementations is that they are so obvious that competition is inevitable, and so basic that differentiation is difficult. For example, it's likely that major cloud providers will move into BaaS services related to storage and identity, and social network companies will offer social network BaaS integration. Vertical markets, where a developer already has skills, contacts and software products, are useful places to look

In this e-guide

into for BaaS opportunity. In horizontal markets, data analysis and reduction tools, or forecasting and projection tools, are good candidates for BaaS.

Technically, the key issue for BaaS is preserving the dominant RESTful Web service model of interface. These interfaces appear as a simple PUT/GET transactional model that doesn't remain in a processing state between calls.

Most of the major PUT/GET giants have exposed their own service APIs as RESTful interfaces that receive and respond using XML data structures. XML handling can be a chore, however, and it's helpful to either accept a simpler data structure or provide programming logic to use the interface as a reusable function in each of the target mobile devices.

Creating a BaaS service with the correct level of functional granularity is also important. Mobile users typically expect a quick response to a request, and there are interface timing issues that make it unadvisable to turn a complex function with a long execution time into a single service. Since RESTful interfaces are expected to be stateless, care is required in turning a complex function into a series of elements that can be used independently, but can function together when needed. Each request must stand on its own, and data from the response to a given request, if needed later, must be stored in the device and returned or maintained and updated by the service.

In this e-guide

Passing as PaaS

Perhaps the most significant aspect of BaaS, from the developer's perspective, is the relationship with PaaS. Any developer who has reviewed the APIs for search or social network giants, or for cloud management, knows that multiple APIs typically form an ecosystem that, if it grows large enough, can approach the functionality of a platform, as in PaaS.

A few disconnected BaaS APIs are less compelling than, for example, an entire BaaS ecosystem aimed at supporting mobile-social network collaboration for enterprises. Popular open source blogging or chat room tools, for instance, could be built on unified communications and collaboration, augmented with APIs to create voice and video calls.

BaaS will lead to a systematized view of background services for mobile development, which is for all practical purposes a mobile-driven PaaS platform. Developers, operators and enterprises need to think about this evolution -- because a cooperative and symbiotic collection of BaaS tools are more valuable to developers and consumers than individual APIs could ever be. The BaaS movement could be a catalyst for moving developer focus from devices to the cloud -- and that means faster service and increased competition among handset vendors. But BaaS, like any cloud trend, is in the early stages, and there's a whole mobile world out there.

In this e-guide

What CIOs, developers should know about the 'API economy'

Jan Stafford, SearchSoftwareQuality

'API economy' is a catchphrase fueled by evidence that APIs are a rapidly expanding economic force.

Chris McNabb

Use of application programming interfaces (APIs) is expanding rapidly because legacy application integration strategies fit business needs in the client-server world, but fail to meet today's need for real-time, distributed cloud and mobile applications, according to Dell Boomi general manager Chris McNabb. For CIOs, using APIs enables efficient sharing of information and data across applications, something that's been hard to do until now. The challenge is that API-centricity calls for big changes in traditional middleware stacks.

Simply put, an API is a toolset of protocols and routines for specifying how one application can interact with another application. "Think of your products being sold on Amazon.com, or collaborating with SharePoint files via a platform like Yammer," said McNabb in a recent SearchSOA interview.

In this e-guide

"When properly implemented, APIs make it far easier for software developers to bring together data from various sources to create today's modern Web applications."

Evidence that APIs are an economic force that merits the catchphrase API economy comes from many sources, said McNabb. Consider, he said, that Netflix has used its API to extend its service to over 200 different devices -- television, gaming devices, remote controls -- on all major mobile platforms in less than two years. Then look at predictions for APIs' future. For example, 50% of business-to-business collaboration will take place through Web APIs by 2017, and by next year 75% of Fortune 1000 firms will offer public Web APIs, according to IT research firm Gartner Inc.

McNabb elaborates on APIs' uses, API strategies and changes wrought by cloud integration in this Q&A.

What technology and business changes contribute to the big increase in API use?

Chris McNabb: In the traditional hub-and-spoke model a central server, or hub, performs the data transformations and communicates independently with each application, or the spokes. Today, however, enterprise IT is highly decentralized and includes data assets and huge volumes of real-time data that often travel great distances over the Internet.

In this e-guide

In this scenario, the central server creates a choke point and a single point of failure. In addition, with the centralized hub-and-spoke model, enterprises typically need to purchase a separate instance of a vendor's integration product for each integration process, and each product needs to be administered separately, creating an expensive and highly complex environment.

A cloud-based application integration platform turns the integration model on its head. To reduce latency, integration processes take place in close proximity to *any number of places* where the integration of the applications is required. And while the execution of the integration is distributed at multiple points throughout the enterprise, the development and administrative functions are centralized, providing the ability to develop, deploy and manage all the integration processes anywhere across the WAN from a single location. In short, we are witnessing the end of middleware as we know it.

What are the most common uses of Web APIs today?

McNabb: Web APIs are required for mobile and social applications, as well as for innovative forms of e-commerce that combine Web services and contextual information to deliver a compelling user experience.

A Web-based API can expose an on-premises data asset, such as a product catalog or a SharePoint file, to a Web-based application, such as a

marketplace or a social media platform. It can also connect on-premises to on-premises, and cloud to cloud.

What's the role of cloud computing in API strategies?

McNabb: A cloud-first [strategy] is appealing because it promises to lower costs and increase agility. However, CIOs should view their API strategy not just as the integration methodology that enables cloud-first, but also as a way to syndicate assets to new internal and external audiences. When considered this way, it becomes clear that a well-conceived and properly implemented API strategy also has the potential to drive innovation, increase productivity, and create new channels and markets.

What software development and integration challenges do moving to cloud as an integration platform address?

McNabb: Adopting an API strategy requires effective management and governance in order to achieve smooth and cost-effective integration. Even enterprises that recognize the value of APIs often treat them as an afterthought on a project list rather than a core feature. As a result, the APIs are not well designed or properly built and can wind up costing tens of thousands of dollars in ongoing maintenance due to infrastructure costs and the drain on engineering resources.

In this e-guide

A cloud-based application integration platform can help solve these challenges as well by providing a consistent development environment that provides a centralized place to create, manage and monitor APIs and their performance.

Where will API and services dominance impact existing systems most?

McNabb: You'll see a decomposition of ERP systems, as an example, that have been on-premises and in the cloud, or hybrid for a long time. You'll see a constant migration from 'I own and can control everything,' to constant changes from on-premises to cloud; some may come from some areas outside of IT, and that's the architecture of the future.

For example, Gartner is saying that by 2017, CMOs will spend more on IT than CIOs. What will they be doing with that? They will be providing marketing services and answers to the rest of the business -- a sentiment analysis to sales, for example -- to offer up those solutions. And they may have bought it from Salesforce or Radian6, but they'll be using APIs to make those services and data available.

Could you offer advice to CIOs about what assessment and planning stages in enterprise architecture, as that relates to the API economy?

McNabb: Honestly, my advice to CIOs is to really listen to and understand the market landscape and varying business needs. As this services economy

In this e-guide

comes, they'll need to understand the meaning and significance of all the things they've traditionally been doing.

Talking to industry analysts is a good starting point, allowing CIOs to think through what fits for their business. There is no one-size-fits-all -- the CIO of a Fortune 50 manufacturing firm will have a different take on what type of enterprise architecture works best than the CIO of a midmarket company. So my counsel would be for them to work with analyst firms and industry experts to determine what's best for the organization. There is no one single thing that I would state to that question, I would advise CIOs to read what analysts have to say about platforms, about integration and about mobile and big data, and work with them to tailor their planning and strategy for all that stuff to their organization at their pace.

Software developers demanding resource based, RESTful APIs

Jason Tee, TheServerSide

Well-seasoned enterprise architects know that there is no silver bullet when it comes to application integration, and no single middleware solution will ever meet every integration requirement criteria to the tee, but when it comes to modularization and web service development, a REST based approach is becoming more and more popular. REST may not be the best

In this e-guide

approach for every situation, but it does have enough virtues to put it on the short list for any organizations that is comparing the various different approaches to developing web service, creating OSGi components and deploying micro-services . And the fact is, a REST based approach can be effective in even the most challenging corner-cases, where other options like Java messaging or SOAP based web services run out of steam.

The effective application of RESTful techniques

One of the key strengths of REST is its simplicity, but that simplicity sometimes raises objections from developers who question whether such a simple approach to application integration can deal with the complex challenges enterprises must deal with on a daily basis. For example, software architects often question the veracity of REST when it comes to handling complex, real world, transactional systems that involve coordinating multiple resources whose interactions need to be synchronized. Jerome Louvel (@jlouvel), coauthor of Restlet in Action and the creator of the Restlet framework, addresses the topic of transactional, e-commerce systems by framing it as a simple use case with a straightforward solution. For example, to make a traditionally developed online shopping application into a RESTful experience, an architect simply needs to break down the associated business transactions into all of their underlying components.

In a transactional, online e-commerce type of system, people need to be able to browse products, select products, view and edit their cart, enter

In this e-guide

payment and shipping information and then make a final decision to pay. At any point up to the point where the user completes their checkout, users need to be able to navigate back to a previous step.

From a pure API standpoint, this means mapping all the important parts of the transaction into resources via URIs. At the end, the transaction is completed by sending a final HTTP request that coordinates the state of each of the resources involved in the transaction. “It’s just viewing your transaction as a set of one or multiple resources depending on the complexity of your business processes. Technically, nothing prevents you from doing that. It’s just not trying to map what you’re used to doing with RPC or putting too much context in one request,” says Louvel.

Of course, a RESTful approach to software design is only one piece of the puzzle when it comes to creating integration middleware that will tie together back-end services with mobile applications, browser based web sites and even embedded systems. Creating application integration middleware also means creating modular and reusable components, with the most proven approach for achieving these goals being the creation of micro-services using OSGi. Unfortunately, the manner in which JAX-RS, REST based applications connect with OSGi based components hasn't always been clear. "People always struggle to connect OSGi concepts with JAX-RS concepts," said Holger Staudacher, a software developer and consultant at EclipseSource. However, the integration path between RESTful web

services and modular, OSGi components has become increasingly more clear through projects such the OSGi JSX-RS connector.

These software tools help fill in the gaps, simplify the development of REST based micro-services, and make modularity easier to achieve by integrating OSGi with REST based, JAX-RS frameworks such as Jersey and Wink. "We provide the glue" said Staudacher about the various software projects that help simplify the development of application that use REST and OSGi. Needless to say, using a REST based approach for developing a distributed architecture effectively becomes a whole lot easier when there is a proven and effective structure within which applications can be built.

Easing into a resource based approach to APIs

A RESTful approach to software development can greatly simplify the task of application integration, but this new approach can often cause troubles for those with more of a legacy background in distributed computing. That's when developers finds themselves creating more problems than they fix. Louvel's advice is not to take the usual, traditional approach when developing RESTful APIs. Forget what you learned from the service-oriented, RPC based world. Then, you'll discover that you are working without limitations and you can really start to solve your business problems. Furthermore, the modularity involved in developing a RESTful architecture means developers have smaller, more agile parts that provide greater flexibility when designing solutions that really work.

In this e-guide

The main takeaway Louvel has for Java developers is this: apply as many principle from the tenants of RESTful development and design as possible, but at the same time, use only as much as is needed. See everything as a resource. Use calls from the HTTP protocol, especially GETs, PUTs, POSTs and DELETEs to effectively create and update your resources. Eventually, approaching the problem domain in a RESTful manner becomes second hand. By applying these foundational concepts, developers get a lot of built-in features that are common to the world-wide-web such as caching and automatic compression for free. And once the development team has got the basics down, kick it up a notch and try to see every problem as having a RESTful solution

Inside the world of Web APIs, the cloud and mobile technology

Maxine Giza, SearchSoftwareQuality

Rob Daigneau offers his expert insights on the convergence of the cloud, mobile technology and Web APIs.

Download this podcast:

In this e-guide

Good afternoon. This is Maxine Giza, for SearchSOA. Joining me today is Rob Daigneau and he's going to talk to us a little about convergence of Web APIs, the Cloud, and mobile. Thanks for joining us, Rob.

Daigneau: Hi. Thanks for having me.

All right. Let's dive right into things here. SOA has been around for some time, but we still have a lot of people arguing about what it is and whether or not it helps. Some have argued that regardless of what you think about SOA, services are what we should be focusing on. What's your perspective on this suggestion?

Daigneau: SOA certainly has a number of definitions. For example, think about OASIS [Organization for the Advancement of Structured Information Standards]; they define it as a paradigm for organizing and utilizing distributive capabilities that are under ownership from different domains. Eric Newcomer, I think, has a great definition: He thinks of it more as being a methodology to manage applications over their lifetime. Others say that SOA is more about integration and middleware, and the list goes on and on.

Throughout it all, regardless of agreement on what the definitions might be, there have been a lot of big promises. We've heard people say that we

In this e-guide

should get greater agility through this coupling and reuse, and because of that reuse more cost savings.

Certainly some of this has come true, but in many ways it's fallen short because we can't figure out, or come to, a common agreement, on what SOA really is. That being said, SOA can be very difficult to achieve, so it hasn't fully delivered on all the promises. Meanwhile, Web services have been used by many enterprises for so many years and have been put to some good use achieving great value out there, namely for integration.

Another thing along with SOA, Cloud platforms have also been debated hotly. Many have argued that similar to SOA, they would yield greater agility, lower cost, fault tolerance and several other things. Would you say this is true? What's the future for cloud enterprises?

Daigneau: The cloud certainly provides value, but the hype surrounding it these days has been a bit over the top. Fortunately, we seem to be coming out of that stage of peak inflated expectations, as Gartner might call it. There is evidence that the cloud can reduce Opex and to Capex, and that also facilitates agility by making it easier to provision and deploy system images. However, the cloud seems to fall short on providing unbounded scalability, and high availability is still a bit of a challenge out there.

Enterprises are also not yet comfortable with how secure their assets might

In this e-guide

be in the cloud. They realize that they'll have to give up some degree of control over their systems.

Add to this the fact that enterprises need to understand things like Infrastructures as a Service [IaaS], Platform as a Service [PaaS] and Software as a Service [SaaS]. It's easy to see why many businesses are moving slowly to the cloud. However, I do think that many businesses will use the cloud in the future, and certainly many are right now.

I'm really referring to public cloud platforms here, not internal virtual machines, which many vendors are now calling the private cloud; a neat marketing trick. Anyway, to this end, I think that many enterprises will continue to leverage Infrastructure as a Service before Platform as a Service because it's simply easier to understand and doesn't lock you into a particular cloud provider. IaaS provides some portability when we need it.

We're also already seeing businesses adopt a hybrid approach, where some of their business services are pushed to the cloud while some other portion remains on-premises in their data center. For example, we might use a public cloud as an edge computing platform to process some subset of business functionality and data.

This provides a really nice and practical partitioning approach that promotes overall availability, performance and security across the enterprise, because the enterprise can now manage the services independently, that is those in

In this e-guide

the cloud, separately from those that are on-premises. All in all, I think the future is bright for the cloud.

I think that we would be remiss not to delve into mobile platforms a bit. They are without a doubt an example of a disruptive innovation, and it's clear the consumer market has embraced these technologies, and quite honestly is changing the way we live. Where do you see things going in respect to mobility in the enterprise?

Daigneau: Enterprises seem to be jumping onto this movement fairly quickly. For one, they recognize that their customers are using the mobile platforms, and they also see it as an effective way to provide capabilities to field personnel and employees on the go.

Business executives also see mobile as a great way to provide business intelligence, dashboards, things of that nature.

Enterprises are facing a number of challenges in the mobile space. Mobile apps cannot be designed like traditional applications, and software developers and architects need to become more familiar with security issues. They must also think about how cellular networks affect application performance. They must also design these apps to work even when they can't connect to some back-end Web service.

Enterprises are awful at trying to figure out which mobile platforms to target.

In this e-guide

Should they look at iOS, Android, Windows? Some are even still using Blackberry.

Some of these decisions will be driven by their customer preferences, others by company policies, or in some cases by the preferences of employees, especially in environments where they adopt a bring your own device [BYOD] philosophy. In any event, it's clear that enterprises will be using mobile devices and platforms more and more as the future comes into our vision here.

Could you just delve a little into, if you could, mobile apps and how Web APIs and services work together?

Daigneau: Sure. Web APIs and Web services are used to receive data from mobile apps, and are also used to send information to these apps, such as alerts and notifications. Of course, this data might also be multimedia, as well. Mobile apps and Web services really are quite complementary.

Web services are also used to encapsulate business functionality. You can certainly distribute some of these business functions and position some of this within the mobile app themselves, but when the business capability is complex, hard to distribute, perhaps lengthy in terms of execution, things of these nature, it can really make sense to consolidate that type of logic, that type of capability in a Web service. Other functions like graphical functions and data visualization, of course, those make sense to perform down on the

mobile device.

Web services are also used to update the mobile apps themselves. This is essentially what's going on, for example, when your smartphone tells you that you have a number of apps that need to be updated. The device is communicating with the Web service to find out if there are any updates, and then it uses another service to download these updates. Most mobile apps, especially for the enterprise, I think, they are going to be relying on the Web services.

We've seen that Web APIs as compared to more traditional Web services that use SOAP [Simple Object Access Protocol] and WSDL [Web Services Description Language] are the most natural way to provide these capabilities to mobile apps. They're just so much easier to work with, and for some other reasons. Probably we don't have enough time to get into all those details on this podcast.

Could you just tell us a little bit about how the cloud plays into all of this?

Daigneau: I hinted at some of this a little earlier. Cloud platforms provide an excellent way to host your Web services. Some businesses will just go out and build everything on the cloud in an attempt to avoid having to maintain their own data center. Others will attempt to adopt that hybrid approach that I mentioned earlier, where some of the business services are pushed to the

In this e-guide

cloud and some remains on-premises in their data center.

We get in this scenario that the business might use the public cloud platform to handle some subset of their overall business computing needs. In these scenarios, we can see how Web services, Web APIs, mobile apps and the cloud, all of this stuff, come together; they converge and they help to create some really exciting opportunities for business.

In terms of convergence, what about native applications; HTML5, CSS, JavaScript? How do these decisions factor into that?

Daigneau: First, we've got to remember there's always going to be situations where it's going to make more sense to build an application using native mobile technologies. This approach will probably always provide the richest and most responsive UX [user experience]. Also at this time, it seems that the developer tools, debugging tools and diagnostics are a little bit ahead and probably will remain so for a time. For example, we've seen companies like LinkedIn giving up on their recent HTML5 work in favor of going more towards a rich native platform environment.

This all being said, the problem is that when you have to recreate your application for multiple mobile platforms, you got a lot of development to do and, really, a lot more work. For situations like these, some businesses select

In this e-guide

mobile SDKs [software developer's kits] that enable developers to write once and deploy to multiple mobile platforms.

Where have we heard that before? Sounds kind of familiar, right? Like Java, for example. The issue here is that these SDKs often choose a lowest common denominator approach, and they're not really able to harness the unique capabilities of the given mobile platform.

For many of these reasons, many of us are leveraging technologies like HTML5, CSS3, JavaScript and a host of open source frameworks that let us create rich mobile applications using industry standard and nonproprietary technologies. In a way, all of these things define a new type of platform that has really taken on a lot of popularity. For example, Netflix has decided, on the other hand, to move away from proprietary technologies -- in this case it was Silverlight -- to use HTML5; that's was neat. Again, these technologies work really, really well with Web APIs, and it's almost like a marriage made in heaven. There you go. That's my view on the convergence of Web APIs, the cloud and mobile.

Wonderful. Thank you so much for sharing your insights, Rob. Thank you, again.

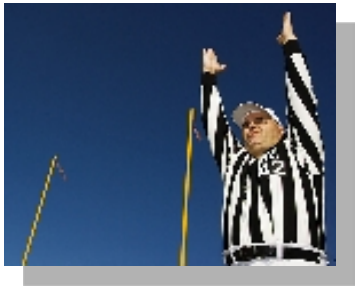
Daigneau: Thanks again for having me.

In this e-guide

NFL statistical data company scores touchdown with API service

Maxine Giza, SearchSofwareQuality

NFL data statistics company discusses how it evaluated several API service vendors.



For die-hard sports fans, having the most up-to-date statistics in an easy-to-read fashion isn't something that is just nice to have, it's a way of life. When Philadelphia -based NFLdata.com decided it was time to update its application programming interface (API), the company tackled the challenge head-on to find a cost-effective, reliable solution to meet clients' expectations.

NFLdata.com is still a relatively young company, with ambitions of expanding to cover more sports down the road. With that in mind, CEO Scott Gimpel needed a platform that could be built upon and would allow data to quickly be presented in an aesthetically pleasing way.

In this e-guide

Specific requirements for an API service

The IT team had already developed an API, but Gimpel noted many of the desired features necessary to push his company forward would require a lot of time. In an effort to save time on labor, Gimpel looked to outside API service vendors. Like many other businesses, in the early stages or not, affordability was a must.

Some of the wish-list components included:

- Caching functionality
- Developer portal
- Rights management

Scott Gimpel, CEO, NFLdata.com

Going through the API evaluation process

It's one thing to reach the point where a business knows it's time to upgrade its API, it's another to know just how to go about it. Gimpel said he waded through several other vendors, including Mashery and Apigee, but ultimately went with Apiphany. The vendor was spot-on in a couple of areas on Gimpel's requirement list.

In this e-guide

"We looked at some of the bigger, more established companies that do API services," said Gimpel. "One of the problems was that since we are a smaller business, they weren't affordable for us. It was too much of an up-front commitment, too much up-front capital."

Expensive, multiyear commitments were a big turnoff for Gimpel, who said Apiphany was much more flexible with its payment options. The vendor allows NFLdata.com to pay it month to month, which is key for the company, since revenue varies depending on whether the NFL is in season.

While cost is certainly an important factor, it doesn't trump everything. The type of technology used was also something the CEO took into account. "All of our stuff is Microsoft-based, all their stuff is Microsoft-based, so that helped us feel that our systems would interact well together," said Gimpel.

Offering a system that would make things easy for the NFLdata.com team wasn't the only thing Gimpel kept in mind; he also wanted to streamline procedures for people who use his service. Apiphany was able to offer some important customizations, such as making it possible for users who sign up on the NFLdata.com site to automatically be registered in the developer portal.

In this e-guide

"That was a huge thing for us because then people wouldn't have to have two accounts," said Gimpel. "We had all of these users in our database, and that was sort of a technical hurdle; we didn't want to have people create two accounts with different passwords they'd have to remember."

Scoring an API service touchdown

Like any company who takes on a new technology task, Gimpel's team learned a bit about what they will do differently when it does expand its sports coverage in the future. For example, user migration wasn't without snags.

"It wasn't that smooth of a transition only because we had a lot of people on our old platform, so that was a curve to get through," said Gimpel. "We are still getting people migrated to a new environment, but it's a lengthy process. You send people emails about it and they just ignore them, but you also can't pull the rug out from under them because they are depending on you."

Since the company so far is pleased with its new API, when NFLdata.com expands it's anticipated to go smoothly and simply build off the existing platform. Unlike what was initially done, the company won't waste time doing some things on its own.

"Before we got integrated into the platform, what we did was we built some things ourselves, like user management, into our website. That took time and it had bugs and we had to test it, fix those bugs and get it working," said

In this e-guide

Gimpel. "Now being able to piggyback on the new platform, we won't even bother with any of that stuff. We'd focus on data and structuring that part of it."

API management, servers enable mobile, Web gift services strategy

Jan Stafford, SearchSoftwareQuality

By exposing a network through application programming interfaces, Blackhawk Network is making the change from physical to digital cards.

Blackhawk Network, a global provider of prepaid gift and reloadable cards, is moving from physical to digital card offerings, particularly in the mobile computing space. A key part of this move is exposing the network through a set of application program interfaces (APIs) to digital distribution partners. Leading this project is API development and management expert, Mike Gionfriddo, chief technology officer (CTO) at the Pleasanton, Calif., company. Gionfriddo is the original author of the Jam API, now known as Java Management Extensions, or JMX.

In this e-guide

"Taking Blackhawk Network's dominance in the physical card space and then moving into the virtual space was an exciting opportunity for me, due to my background in APIs," said Gionfriddo, a former Sun Microsystems Divisional CTO and Juniper Networks Distinguished Engineer.

He saw it as a natural follow-up to his work on Java FX and client-side Java at Sun. "Due to my hard-core engineering experience, I know a lot about mobile devices, codecs, audio transforms, and system and network management," he said. In this interview, Gionfriddo shares his API management best practices and technology choices for the physical-to-digital services migration.

What is Blackhawk Network's foundation approach for API use?

Mike Gionfriddo: We've built our technology and API set to have a robust underlying system for physical business, and now we're tapping into that to open the digital business.

Our pattern of API usage for digital services closely mirrors what we do in the physical world. API management is about setting limits. I'm not a huge believer in creating big catalogs. Part of the dilemma with those is as soon as you create them, they're out of date. We have a couple standard ones that we use: Apache, mostly at the server tier and then, because we have some .NET, we use IAS [Internet Authentication Service]. If you pick just a few vendors that you want really want to work with, and they have a limited set of APIs, management is easier.

In this e-guide

What everyday practices does Blackhawk Network's team in API management use?

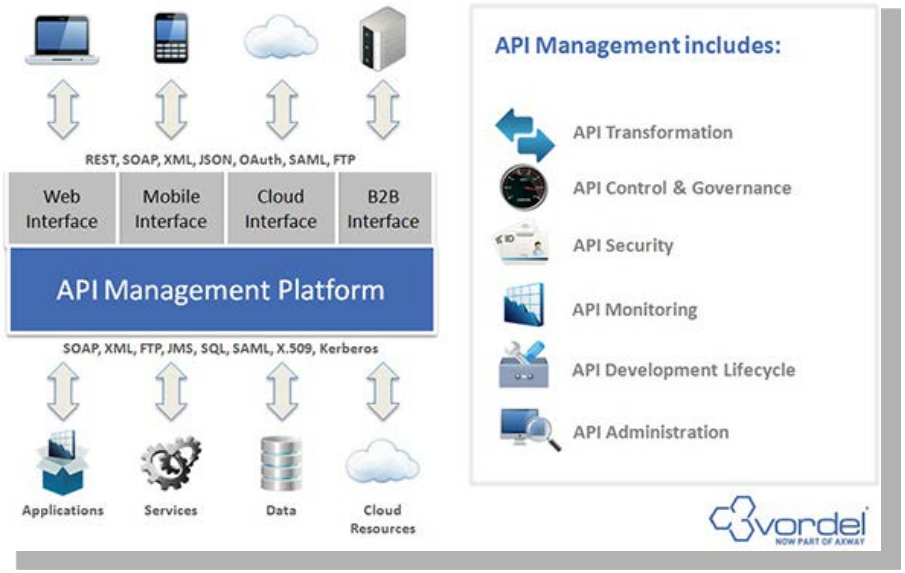
Gionfriddo: We base our services off an underlying framework, so the services share some common code. Also, we have a simple checklist that developer managers have to go through. We have two classifications of APIs, internal and external; those shared inside of Blackhawk and those outside of Blackhawk. We do some self-assessment and checklists before publishing APIs.

The important job is separating out that [internal and external] policy. Enforcement usually points away from the actual code itself. We use Vordel API Server for security access control in both a physical and logical way to separate out API policy from the business logic. The Vordel server's security access control is helpful there, because control is set in one place and we know how to modify it. We don't have the app teams do that, and that provides us that separation. But it also provides us with auditability. Auditing is critical for our business, because we need to be PCI [Payment Card Industry] compliant and SSAE 16 compliant. Another need auditability fills is informing us on how we want to price for these services. This takes services development out of having to implement pricing.

In this e-guide

What led you and Blackhawk Network to choosing an API server?

Gionfriddo: I got there in an interesting way. I was the chief architect at Sun's Middleware Division, and we were working closely with a competitor of Vordel. Also, I saw what IBM had done with DataPower, a nice architectural pattern for services. When I came to Blackhawk, a [team member] here had seen the [DataPower] patterns.



In this e-guide

So, we were on the same page in deciding to make API management layered, using approaches like rate limiting and even to route credentials potentially to two different instances. We have a lower SLA [service-level agreement] for a certain class of users, for example. Using an API server as a control point gives us the flexibility to allow our business teams to say how they want to use digital card services, charge for them and create interesting models. An API server -- in our case, Vordel API Server -- gives us that flexibility. Again, for compliance, it gives us a nice way to have that control in a single point and not spread throughout the application tier.

Overall, I don't see any solution on the horizon that's going to manage all the APIs. Vordel API Server, however, delivers the controls we need, and integrates well with our development environment, so we can test those controls.

Next article

In this e-guide

Definitions

Must-know: Key API terms

The glossary provides common terms related to API management.

Application program interface (API)

An application program interface (API) is code that allows two software programs to communicate with each other. Typically, APIs are released for third-party development as part of a software development kit (SDK) or as an open API published on the Internet.

The API defines the correct way for a developer to write a program that requests services from an operating system (OS) or other application. APIs are implemented by function calls composed of verbs and nouns. The required syntax is described in the documentation of the application being called.

Typically, APIs are released for third-party development as part of a software development kit (SDK) or as an open API published on the Internet.

In this e-guide

If the applications are written in different languages or have been written for different platforms, middleware can provide messaging services so the two applications can communicate with each other.

Business interest in APIs grew with Web 2.0 mashups and executive dashboards that pull data from two or more sources. Cloud computing has fueled even more interest in APIs, as companies experiment with ways to integrate a cloud provider's service with on-premises systems or other cloud services.

Open API

An open API, sometimes referred to as a public API, is an application program interface that provides a developer with programmatic access to a proprietary software application.

An API is a software intermediary that makes it possible for application programs to interact with each other and share data. It's often an implementation of REST that exposes a specific software functionality while protecting the rest of the application.

In this e-guide

Open APIs are published on the Internet and shared freely. A startup software company, for example might publish a series of APIs to encourage third-party developers in vertical industries to be innovative and figure out new ways to use the startup's software product. In theory, it's a win-win business arrangement.

The third-party developer can make money by licensing his new program, a mashup with advanced functionalities that would be almost impossible to create from scratch. The startup gets to expand their company's user base without having to spend any money to develop niche industry software -- and they still get to keep their source code proprietary.

Open APIs can be problematic for developers, however, because the company publishing the API has all the power. If the startup ever decides to change the terms of use for its API, for example, or decides to charge a fee for licensing the API, the third-party developer has no choice but to accept it and deal with it.

API management

API management is the process of publishing, promoting and overseeing application programming interfaces (APIs) in a secure, scalable environment. It also includes the creation of end user support resources that define and document the API.

In this e-guide

The goal of API management is to allow an organization that publishes an API to monitor the interface's lifecycle and make sure the needs of developers and applications using the API are being met.

API management software can be built in-house or purchased as a service through a third-party provider like Mashery Inc., Apigee Corp. or SOA Software Inc. API management software tools typically provide the following functions:

1. Automate and control connections between an API and the applications that use it.
2. Ensure consistency between multiple API implementations and versions.
3. Monitor traffic from individual apps.
4. Provide memory management and caching mechanisms to improve application performance.
5. Protect the API from misuse by wrapping it in security procedures and policies.

In this e-guide

A growing open API movement, spearheaded by big name companies like Facebook, Google and Twitter, has led to reduced API dependency upon conventional service-oriented architecture (SOA) in favor of more lightweight JSON and REST services. Some API management tools are capable of converting existing SOAP, JMS or MQ interfaces into RESTful APIs or JSON content.

REST (representational state transfer)

REST (REpresentational State Transfer) is a simple stateless architecture that generally runs over HTTP.

REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of Web services. The use of REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet. The SOAP approach requires writing or using a provided server program (to serve data) and a client program (to request data).

In this e-guide

REST'S decoupled architecture, and lighter weight communications between producer and consumer, make REST a popular building style for cloud-based APIs, such as those provided by Amazon, Microsoft, and Google. When Web services use REST architecture, they are called RESTful APIs(Application Programming Interfaces) or REST APIs.

REST architecture involves reading a designated Web page that contains an XML file. The XML file describes and includes the desired content. Once dynamically defined, consumers may access the interface.

REST, which typically runs over HTTP (Hypertext Transfer Protocol), has several architectural constraints:

1. Decouples consumers from producers

2. Stateless existence

3. Able to leverage a cache

4. Leverages a layered system

5. Leverages a uniform interface

REST is often used in mobile applications, social networking Web sites, mashup tools, and automated business processes.

In this e-guide

The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations (verbs). Flexibility is provided by assigning resources (nouns) their own unique Universal Resource Identifiers(URIs). Because each verb has a specific meaning (GET, POST, PUT and DELETE), REST avoids ambiguity.

There are some downsides. In the world of REST, there is no direct support for generating a client from server-side-generated metadata. SOAP is able to support this with Web Service Description Language (WSDL).

REST provides the following advantages, specifically advantages over leveraging SOAP:

- RESTful Web services are easy to leverage by most tools, including those that are free and inexpensive. REST is becoming the dial tone for systems interaction, including the use of RESTful Web services, which are, for the most part, the way cloud providers externalize their cloud services.
- SOAP services are much harder to scale than RESTful services. Thus, REST is often chosen as the architecture for services that are exposed via the Internet (like Facebook, MySpace, Twitter, and most public cloud providers).

In this e-guide

- The learning curve seems to be reduced. Developers are able to make use of REST from within applications faster than they can with SOAP. This saves time, which saves money.
- REST uses a smaller message format than SOAP. SOAP uses XML for all messages, which makes the message size much larger, and thus less efficient. This means REST provides better performance, as well as lowers costs over time. Moreover, there is no intensive processing required, thus it's much faster than traditional SOAP.
- REST is designed for use over the Open Internet/Web. This is a better choice for Web scale applications, and certainly for cloud-based platforms.

Moving forward, REST is likely to continue its growth as enterprises seek to provide open and well-defined interfaces for application and infrastructure services. The growth of public and private cloud computing is driving much of this demand, and will continue to drive growth into the future.

Next article

In this e-guide

Getting more PRO+ exclusive content

This e-guide is made available to you, our member, through PRO+ Offers – a collection of free publications, training and special opportunities specifically gathered from our partners and across our network of sites.

PRO+ Offers is a free benefit only available to members of the TechTarget network of sites.

Take full advantage of your membership by visiting http://pro.techtarget.com/ProLP/

Images; Fotalia