

This white paper will look at the early origins of APIs and how they have evolved. It will address integration issues associated with “plugging into” a telecom service provider’s operational support infrastructure and the role APIs play. It discusses industry efforts in the area of API standardization for essential processes that are internal and external to a telecommunication service provider’s operations. This paper also examines where third-party middleware fits into the maze of systems integration, and what OSS providers must do to help ease the integration woes of customers.

We would be pleased to receive any feedback about this paper. Please send your comments or questions to info@eftia.com.

Application Program Interfaces

The cornerstones of communications

The use of application program interfaces (API) is on the rise, with nearly every major corporation utilizing APIs in some way, shape or form, regardless of industry segmentation. But how does this movement toward common and reusable APIs impact operational support system (OSS) vendors and the telecommunications market as a whole? Before we can answer this, we need to take a look at the groundwork that has been laid.

Born of Efficiency

The early origins of APIs can be traced back to the need to decrease delivery time between trading partners and reduce costly errors related to the manual transfer of information. Trading partners used to (and many businesses still do) pass information back and forth via telephone conversations or written communications that were delivered by postal or other delivery services. Imagine, thousands to millions of physical pieces of paper flying about between businesses on a daily basis, creating a cottage industry for overnight courier services, increasing deforestation and making the overall management of business transactions slow and cumbersome.

The arrival of computers in the business world changed the entire approach to exchanging information. Computers enabled businesses to mechanize many of their internal processes surrounding order fulfillment, receipt and management of goods, and tracking and reporting on work efficiencies. It was this efficiency-reporting factor that highlighted the need to shorten delivery cycles between trading partners. If data was already entered and stored in a businesses’ system, then why couldn’t it be sent directly to a trading partner or, better yet, directly into a trading partner’s order management computer system. This would reduce the manual labor associated with re-writing existing information; eliminate transaction errors that occur when information is transposed from one source to another; and dramatically speed up order processing by transferring information in real-time.

Soon companies began developing electronic data interchange (EDI) interfaces with their trading partners. These interfaces electronically bound together dissimilar systems located hundreds or even thousands of miles apart; however, problems with this



approach soon became apparent. Proprietary interface formats were used between trading partners, so it was common for one trading partner to have numerous interfaces with varying formats from an array of customers and suppliers. These interfaces would frequently “break” causing a disruption in service as systems changed or additional information was added to one and not the other. The disruptions hurt many businesses, especially those that had become reliant on the EDI interfaces and no longer had the personnel in place to deal with business processes corresponding to the manual transfer of information. Plus, the cost of developing and maintaining different interfaces with multiple trading partners placed an increased burden on the information system groups that were handling emerging corporate computing infrastructure, all without the automated network management and administration tools available in today’s environment.

Managing EDI interfaces with trading partners was challenging. In addition to software and hardware component issues, these interfaces had an impact on aspects of the telecommunications network. The electronic exchange of information between trading partners is typically conducted over some form of private line network or what’s referred to in the EDI world as a value-added network (VAN). The VAN serves several key functions:

- Route and transfer data to one or more trading partners
- Validate information and apply business rules associated to each transaction
- Hold pending transactions until the partner can process the transactions (see Figure 1)

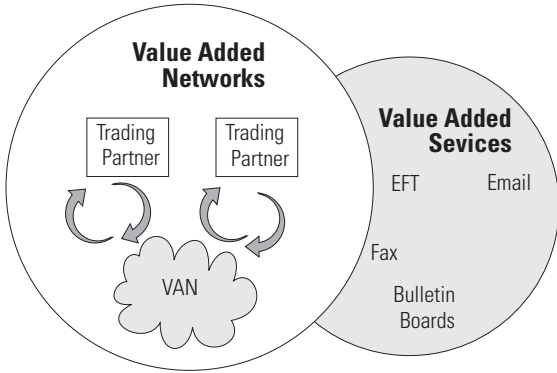


Figure 1. Value added networks (VAN)

Although EDI interfaces are the premier method of exchanging data in the electronic commerce world, the use of e-mail, faxes, electronic funds transfer (EFT) and bulletin boards play a role in business to business transactions.

The VAN is made up of software that brokers the transactions. All parties involved mutually subscribe to the software, and it is often owned and operated by a third party (such as AT&T, GE Information Systems, Sterling Commerce). It includes some form of communications fabric, usually consisting of a non-switched circuit that connects two or more parties via one or more incumbent local exchange carriers (ILEC) and optionally one or more inter-exchange carriers (IXC), depending on each trading partner’s location. Sometimes dialup connectivity is used across the public switched telephone network (PSTN) but this requires special equipment that performs callback. The use of modern communications technology and security applications/standards have all but eliminated the need for callback-based technology and have fostered increases in business-to-business transactions. As an example, the Internet has even enabled businesses to eliminate VANs, reducing costs associated with monthly third-party service fees (such as message size/length, time of day usage, monthly surcharge) and communications.



Defining Moments

In 1968, the United States transportation industry organized the Transportation Data Coordinating Committee (TDCC) to define standards for the industry. The TDCC defined the objectives of the EDI program with four key points:

- It must provide generalized data standards and formats for computer-to-computer interfaces.
- It must operate uniformly, regardless of the company's software or hardware.
- It should allow for any type of networking services, protocols or transmission speeds.
- It should be able to exchange documents with the corporate database.

The TDCC initially established 45 transactions (electronic documents or syntaxes). The EDI transactions were used to perform electronic communications between transportation carriers, including ocean, motor, air, and rail carriers, and their goods and services providers, such as brokers, customs, freight forwarders and bankers.

By the 1970s work had begun to commercialize EDI transaction standards for a wider audience in the business world. Leveraging the work done by the TDCC, the American National Standards Institute (ANSI) American National Standards Institute (ANSI) American National Standards Institute (ANSI) formed an Accredited Standards Committee (ASC) Accredited Standards Committee (ASC) Accredited Standards Committee (ASC) to develop what has become known as the X12 EDI standard. The X12 standard is currently managed by the Data Interchange Standards Association (DISA) Data Interchange Standards Association (DISA) Data Interchange Standards Association (DISA), which is a non-profit organization responsible for the organizational membership, standards development and maintenance, publications, and communications with ANSI.

“X12 standards facilitate order placement and processing, shipping and receiving information, invoicing, and payment and cash application data, and data to and from entities involved in finance, insurance, education, and state and federal governments by establishing a common, uniform business language for computers to communicate across town or around the world. With more than 275 transaction sets, X12 standards can be used to electronically conduct nearly every facet of business-to-business operations.” (Source: ASC Web site www.x12.org)

While the X12 EDI standard only pertains to businesses within the United States, a separate set of EDI standards called EDI for Administration, Commerce and Transport (EDIFACT) EDI For Administration, Commerce and Transport (EDIFACT) EDI for Administration, Commerce and Transport (EDIFACT) have been developed for other countries and sponsored by the United Nations. United Nations. These standards are in wide use across Europe and Asia. EDIFACT is so widespread in Europe that any com-



pany wishing to compete must be compatible. To this end, work began in 1997 to align the X12 standards with the EDIFACT standards to enable US-based companies to compete globally.

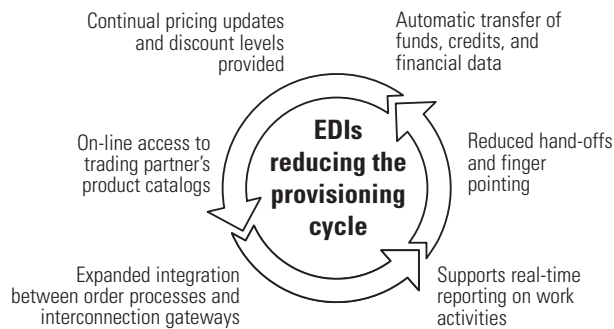


Figure 2. EDI reduces the provisioning cycle

There are a number of standards for EDI that have been created, both privately and publicly, but only X12 and EDIFACT are mandated for use within the US Federal Government. But what about the need for EDI standards in the telecommunications business sector? Where do the EDI standards come into play, and who is helping to establish them?

Creation of APIs for Telecommunications

Unlike financial institutions, the transportation industry and many large manufacturing firms, telecommunications service providers have been somewhat slower to jump into the electronic interface game. In part, because many of the ILECs in North America had systems that were developed by large internal, or externally funded, research and development organizations. These systems were designed as complete end-to-end solutions that would interact between separate components and programs using database calls and other proprietary access methods. Also, suppliers of telecommunications equipment were mostly the same manufacturers that supplied the OSS, thus reducing the need to have EDI type interfaces. But the advent of divestiture ended all that, and a whole new breed of OSS vendors cropped up.

Suddenly, instead of one or more systems from a single vendor that were designed to interact seamlessly, there were a variety of OSSs from a slew of vendors that ranged across all aspects of order fulfillment, service assurance and billing. These new OSSs were a necessity since the monolithic systems of the past were difficult to maintain and even more difficult to enhance. Enhanced functionality was a crucial factor, as service providers were facing a whole new set of challenges in introducing new services and competing with a long list of new competitive local exchange carriers (CLEC). In addition, the new telecommunications service provider entrants were also in need of OSSs. They were looking to deploy the latest and greatest technology, and as a result, component OSSs became hot items.

However, it soon became apparent that this new and disparate breed of OSSs did not communicate with one another or with service providers' existing legacy systems, which were still supporting critical day-to-day business operations. In order to solve this problem, computer equipment manufacturers and third-party software vendors joined forces to create windowing terminals that allowed four or more OSSs to be con-

nected to a system controller. This arrangement enabled users to jump between OSSs on a single terminal. More advanced windowing terminals allowed users to copy and paste information between windows, program software keys with login/password information and other OSS-specific commands to shorten typing and repetitive data entry. These more advanced terminals even offered programming tools that facilitated the use of background applications that automatically transferred data between OSSs and invoked system commands using terminal emulation—a precursor to APIs.

Regardless of the approach, data overlap and inconsistencies began to permeate OSSs, creating discrepancies that often times required senior management or highly skilled employees to manually reconcile errors on the fly and filter reporting information to form accurate views of business activities. Also, the hand-off of information between internal groups increased dramatically, which elongated task completion time and delivery of customer services (service orders and trouble tickets), and created confusion in a once streamlined and efficient organization.

To even further complicate matters, the ILECs and CLECs needed to electronically trade information in order to meet service level commitments and policies legally agreed to between one another and those mandated by the Federal Communications Commission (FCC). Standards bodies that were already in place, such as the Ordering and Billing Forum (OBF) stepped up to the challenge and tackled the issue of interoperability between service providers, defining what is known as interconnection tasks and related details: local service requests (LSR), access service requests (ASR) and detailed line records (DLR). Similarly, the Canadian Radio-Television and Telecommunications Commission (CRTC) has established a similar set of transactions for interconnection between carriers. The works by the OBF and CRTC are in extensive use today throughout the US and Canada, and service providers continue to evolve, as does the telecom market.

Additional initiatives by the TeleManagement Forum and Sun's Java Community Process organization have more recently taken on the issue of interoperability between applications and the creation of frameworks that enable OSS vendors to host their applications. (see table 1)

Table 1. API industry initiatives

| Operational Support System/Java (OSS/J) | New Generation OSS (NGOSSsm) | Javatm 2 Platform, Enterprise Edition (J2EE) |
|---|---|---|
| <ul style="list-style-type: none"> • Service Activation API (ver. 0.7, 04/12/01) • Trouble Ticket API (ver. 0.1, 04/27/01) • Quality of Service (QoS) API (ver. 0.6.1, 04/23/01) • IP Billing (under development) | <ul style="list-style-type: none"> • NGOSS Technology Application Note - J2EE (05/01/01) • NGOSS Technology Application Note - XML (05/01/01) | <ul style="list-style-type: none"> • Supports an application-level interface used by the application components to access a database • Supports a service provider interface to attach a JDBC driver to the J2EE platform |



While these initiatives by the TM Forum and Sun are warranted, none have been seriously adopted and used by the OSS vendor community. The real-world definition and creation of OSS APIs has to date been left up to the individual vendors and middleware vendors that support enterprise infrastructure management. Nevertheless, forward-looking OSS vendors recognize the importance of these industry forums and initiatives and are participating in the work being conducted by the TM Forum and monitoring Java-based activities.

API Enabling of an OSS

In today's economic environment, every telecommunications service provider is focused on reducing operating costs. As such, the ability to integrate OSSs for sharing common information and business processes is absolutely essential. The use of APIs is a key step towards this goal. OSS vendors, must create a set of reusable APIs that relate to major business functions that the OSS product conducts with external systems. However, in order to create these APIs, a reliable framework must be established to access data.

The best OSS vendors will have a number of pre-defined, tested and released APIs that are available for implementation. Most of these APIs will be related to extraction of data associated with customer and contact based information, such as name, address and location. They will also offer additional and more complex APIs that address service activation, and full order and asset management. However, APIs are not the best approach for every integration effort. Sometimes very simple interfaces that employ the transfer of flat files can suffice when only small and infrequent amounts of data are passed, thus reducing time of integration and cost. Other times, the OSS and supported APIs by themselves may not provide enough communications infrastructure for complex transaction sets and must be augmented with the use of third-party middleware.

Roles of APIs and Middleware

Although APIs perform all the dirty work of mining data and exposing business processes within an application by accessing low-level commands sets or system calls, APIs play only one part in the integration process. Third party middleware augments APIs by managing many of the communications, handling processes, such as connectivity, publish and subscribe processes, queuing, message brokering, and error handling—functional aspects typically beyond the scope of an API. (see figure 5)

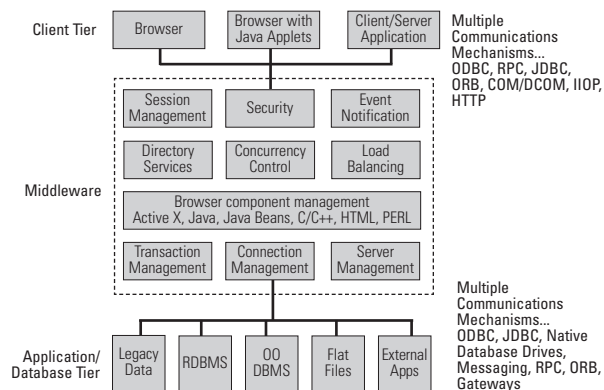


Figure 5. Middleware architecture

Middleware plays a critical role in the all-around performance of a service provider's operational infrastructure. It supports monitoring of connections, transaction processing queues, and network resources to ensure optimal performance and guaranteed flow and receipt of data.



Road to Integration

An aggressive stance on pursuing the development of APIs is just one step in the overall process needed to ease integration difficulties. There is an entire methodology surrounding integration that begins before an API is even created or an existing one is implemented or modified. (see figure 6)

The first step begins with analysis and understanding the problem to be solved, but many times with APIs, the analysis is multifaceted. Instead of trying to solve one problem, the OSS vendor may be tackling a series of problems depending on the number of applications that need to be integrated and the complexity of transactions conducted between them. For instance, an API between a service management and a service activation engine may only require a simple file transfer of information from the service management system and some form of an acknowledgement back from the engine.

A more complex example would be an API between a service management system and a billing system where the systems share common data attributes. In this scenario, numerous transactions or events may be continuously firing between the two applications. Customer profile and detailed service information is passed from the service management system to the billing system as service orders are completed, involving one or more transactions that can be large in volume, depending on the size of the order, with each transaction requiring some form of an acknowledgement and corresponding error handling capability.

In turn, products and services information stored and managed by the billing system within its product catalog are sent to the service management system for use in the order entry and provisioning process. Transactions associated with this functionality may be automated for portions of the product catalog, such as periodic updates or deltas to existing services, while entirely new products and services and/or downloads of the entire catalog may require an administrator of the service management application to manually initiate an API transaction to request the information from the billing system. Additionally, application logic outside the API may be required to handle synchronization and reconciliation of information along with rollbacks, if necessary.

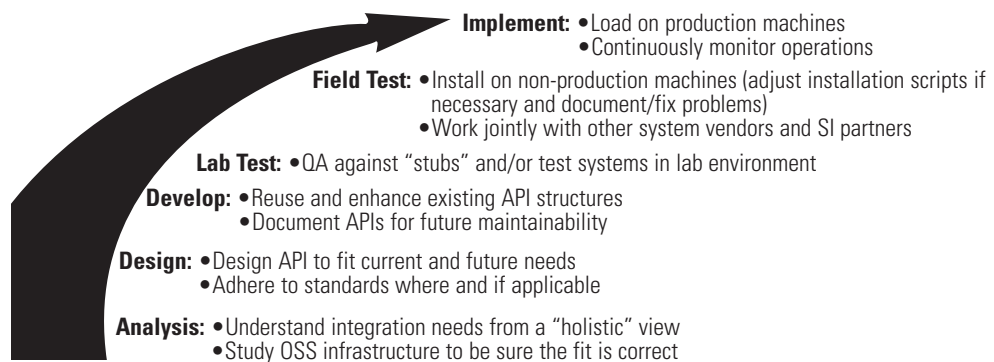


Figure 6. API integration steps

Even though an OSS vendor may have a number of APIs ready and in development, the vendor must still rely on its SI partners to assist in the integration process. This includes adherence to a defined integration methodology, defined by either the vendor or the SI partner. The APIs must be continually maintained to

handle expended transactions and data sets that will be required as a service provider's business evolves.

What Does It All Mean

APIs give OSS vendors a sound technical and business approach to integration that decreases the overall implementation cycle for their customers. As such, customers realize a return on their OSS investment in a shorter time frame. APIs also give SI partners the tools they need to accurately estimate work efforts based on a known set of costing parameters that start with available APIs and infrastructure frameworks that enable them to modify or create new APIs.

An OSS vendor's approach to APIs says a lot about their commitment to building an OSS that is truly in tune with customer needs. An OSS vendor should be involved in the various standards bodies and demonstrate a strong commitment to continuing efforts to increase customer satisfaction by applying state of the art solutions to difficult problems. An OSS vendor augments its value proposition to the customer when it stays abreast of the evolving telecom industry. It should be positioned to lead the charge when it comes to API definitions and knowledgeable enough to increase functionality within and between OSSs. Most importantly, APIs allow OSS vendors to offer components of their product. These days, service providers aren't looking for a complete end-to-end solution; generally, they are looking for best of breed OSSs for discrete functional levels. APIs are the glue that binds these disparate applications together. They connect trading partners, and help establish a seamless operational infrastructure for telecommunications service providers looking to compete in an ever-changing marketplace.

Published: October 2001

Eftia is a leader in developing, deploying and managing OSS products designed to meet the service management and delivery needs of tier one, wireless and next-generation service providers.

The Eftia Master.Scribe® Suite of integrated OSS products provides comprehensive order provisioning and fulfillment; problem management; telecom circuit and asset inventory management; and Internet protocol (IP) address and telephone number management. Eftia also offers Master.Xchange, a configurable OSS interconnection gateway.

If you would like to learn more about Eftia's OSS solutions, please visit www.eftia.com or contact us at 1-888-423-3842.

Eftia (design) is a registered trademark of Eftia OSS Solutions Inc.
© Copyright 2001, Eftia OSS Solutions Inc. All rights reserved. Printed in Canada.
01-10-16

www.eftia.com

