WHITE PAPER

# AN APPLICATION SERVER ARCHITECTURE FOR COMMUNICATIONS SERVICES


**JONATHAN ROSENBERG**
Chief Scientist

**ED GOKHMAN**
Software Architect

**PETER MATAGA**
Solutions Architect

**KELVIN PORTER**
Software Architect



dynamicsoft®

# TABLE OF CONTENTS


| | |
|---|---|
| Abstract | 2 |
| Introduction | 3 |
| The Answer | 4 |
| Auto Conference: An Example | 4 |
| Service Exponentiation Effect: A Service Explosion | 5 |
| Choosing a Platform | 6 |
| Development Cycles | 7 |
| Models of Abstraction | 8 |
| Vertical Markets | 9 |
| Summary | 9 |
| Creating Platform Requirements | 9 |
| The Solution: The dynamicsoft Application Server | 11 |

## ABSTRACT

The Internet telephony industry is poised for change. To date, the success that the industry has enjoyed has been the result of offering reduced service at significantly lower prices. However, the difference in price between traditional circuit-switched telephony and IP telephony is rapidly diminishing for consumers. Unfortunately, service providers cannot buy customer loyalty by offering low prices. Whenever a competitor reduces its prices by a sufficient amount, a service provider can lose customers and revenue. The resultant problem is that Internet telephony Service Providers (ITSP) must find ways to retain current customers (i.e., increase customer loyalty), attract new customers and generate revenue. The solution to their problem is to offer "sticky" services. Sticky services are services that attract new customers and help retain existing ones. An ITSP derives the maximum benefit from these sticky services if these new services differentiate them from their competitors both inside and outside the Internet telephony industry.

So, how do service providers build and deploy sticky services that differentiate them from their competitors?

In this paper, we propose an **Application Server Architecture** that enables service providers to build new and innovative services that seamlessly combine Web content, email, instant messaging (IM) and presence with Internet telephony.

## INTRODUCTION

The Internet telephony industry was born in the mid-1990s, primarily as a client software market for end user PC-to-PC calls over the Internet. It quickly evolved into a services market, with companies offering international calls, long distance, calling card and prepaid calling services. Recently, companies like Dialpad.com[1] have been offering free PC-to-phone service, using advertising to generate revenue.

The common theme throughout the evolution of the Internet telephony industry has been cost savings. From its inception, Internet telephony has been about low cost communication. However, the differences in cost between circuit-switched telephony and Internet telephony will not last forever. ITSPs cannot rely upon "low cost" to continue to be their primary service differentiator. The reasons are clear:

- Circuit-switched telephony is getting cheaper all the time. Over the last few years, U.S. domestic long distance rates have fallen from 25 cents per minute to around 4 cents per minute at the time of writing. Providers are experimenting with flat rate plans, which will further reduce the costs of telephony. As such, the cost reduction of voice over IP (VoIP) compared to traditional telephony continually shrinks.
- Customer retention is difficult when cost is the only factor. If a competitor sufficiently reduces their rates, customers will leave their current provider. "Low cost" is not sticky.
- PC-to-PC VoIP is gaining momentum and will be free. With the growing availability of always-on, broadband IP access in the home, through technologies like Digital Subscriber Line (DSL), cable modems and fixed wireless, customers will soon be able to have direct access to high quality VoIP services. At this point, basic VoIP will be free. Widely available, free, PC-to-PC and PC-to-phone services make it difficult to justify metered phone-to-phone services.

All of this poses a problem for the service provider:

***If the cost differentials that attract customers to VoIP services are disappearing, where are the revenue opportunities for service providers in the future?***

***How can VoIP service providers continue to retain current customers and attract new ones?***

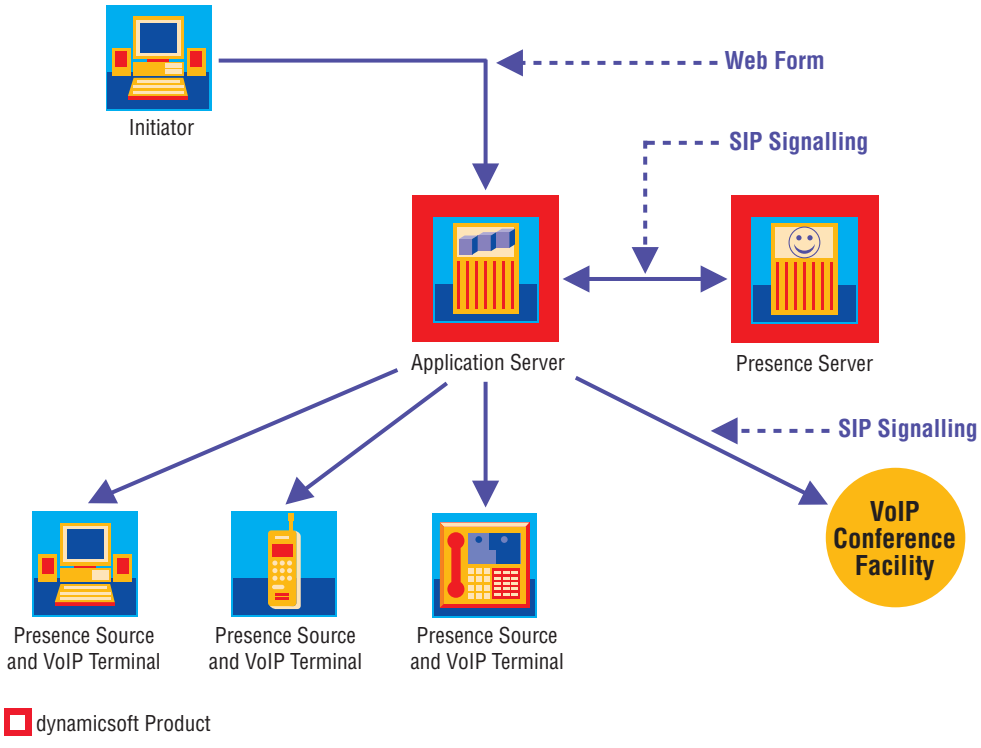
## THE ANSWER

The answer to this question is simple: service providers must offer their customers new and different services. VoIP provides a unique opportunity to define new and innovative applications that can provide customer retention, revenue and service provider differentiation. This opportunity arises because of the "I" in VoIP – Internet. The Internet already provides numerous applications – Web, email, presence, IM, e-commerce, gaming, collaboration and so on. With VoIP, these other applications can be integrated with voice (and video; VoIP includes both Voice over IP and Video over IP), providing new features and services that are impossible in the Public Switched Telephone Network (PSTN).

### *Auto Conference: An Example*

An example of such a service is Auto Conference. This service addresses a common problem – setting up conference calls. Normally, this is done by trying to schedule a call based on the schedules of the participants. However, ad-hoc meetings, phone calls and last minute emergencies make this difficult. What you really want is to have the conference automatically scheduled based on the real availability of the participants. This service does exactly that. A user of the service goes to a Web page and enters the identities of those participants desired for the conference in a form. The service then uses *presence* (also known as buddy lists, instant messengers, etc.) and subscribes to those participants. The application keeps track of when each of the participants comes online and offline, idle and not idle. When it finds that all of them are online and available, it initiates a call to each participant and connects them to a conference bridge. The service is depicted in Figure 1.

***FIGURE 1: AUTO CONFERENCE SERVICE***



### *Service Exponentiation Effect: A Service Explosion*

This Auto Conference service is interesting because it combines three separate Internet applications together – Web (for the initiation of the service), presence and voice. It is these ingredients which has yielded something innovative and new.

This service is only the beginning. The range of possible services enabled by combining voice and video with Web, email, IM and presence is huge. In fact, we theorize that the number of such combinations grows exponentially with the number of different components being combined together. This phenomenon, which we call the *Service Exponentiation Effect*, is depicted in Figure 2.

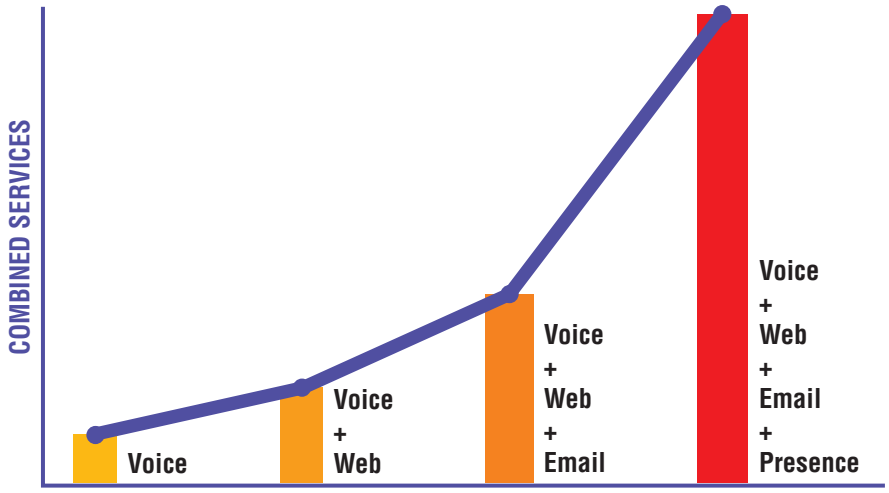



*FIGURE 2: SERVICE EXPONENTIATION EFFECT*

The service exponentiation effect has important implications for service providers. It means that combining voice with other applications provides a means for differentiation. The large application space means that there is lots of room for different service providers to do different things. This differentiation means that applications can be unique to a particular provider and innovative at the same time. The combination of these serve as a tool to attain customers. That solves half the problem of stickiness.

Solving the other half – retention – is harder. That is because once other service providers learn about that cool service and build it themselves, the customers no longer have a reason to stay with the original provider. Thus, it is imperative that a service provider employ several strategies for retaining differentiation in its offerings:

- Rapid Development: To stay ahead of this game, providers need to be able to quickly turn their service concepts into deployable services. In the circuit-switched networks, the service development process is slow and costly. Service providers had to rely upon their equipment vendors to deploy new services. Equipment vendors are very slow to develop new services whether standard or custom. Standard services enabled by equipment vendors do not provide their customers differentiation. This model will not work anymore since service providers need tools for rapid service development.

*THE SOLUTION TO THE SERVICE PROVIDERS' DILEMMA (HOW TO OBTAIN AND RETAIN CUSTOMERS, AND PRODUCE REVENUE FROM THEM) IS CLEAR: VALUE ADD COMBINED SERVICES.*

- Rapid Deployment: Services must be deployed even more quickly than they are developed. In the circuit-switched networks, service deployment may require actually installing new equipment. Failing that, it may require complicated rollout procedures. This model also will not work anymore since service providers need tools for rapid service deployment.
- Specialization: One strategy that a service provider can employ to discourage other service providers from copying its service offerings is to specialize. There are many vertical markets for communications services with very specialized needs – medicine, law, emergency services and call centers. Only a handful of these markets were ever addressed by the circuit-switched world because (except for all but a few lucrative markets) the cost of developing customized applications was too high and the sales volumes were too low. But, in the VoIP space, if the costs of development and time-to-market can be reduced, the vertical communications market could serve as an ideal source of differentiation.

Thus, the solution to the service providers' dilemma (that help attract and retain customers and produce revenue from them) is to provide sticky services that integrate Web, email, IM and presence with voice and video.

These services must be provided by a platform that enables:

- Rapid Development
- Rapid Deployment
- Enhancement
- Opportunities for Specialization

What would a platform look like to meet such needs?

## CHOOSING A PLATFORM

***THE ANSWER IS CLEAR – THE WEB WAY IS THE IDEAL SOLUTION.***

Platforms for building services are not new; such components have existed both in the traditional telephony world (in systems such as the Intelligent Network (IN)) and in the traditional Internet Web world (in systems like Enterprise Java Beans (EJB) Application Servers). As VoIP lives at the intersection of both worlds, the question is this: which model is more appropriate for VoIP services?

The answer is clear – the Web way is the ideal solution. In the sub-sections below, we examine the differences from several angles. Much of the differences relate to the very unique APIs and programming tools used on these systems. An API is the heart of an application development platform. It is the interface used to create new applications. The speed in which an application can be developed, the types of services applications can provide and the robustness of applications all depend on the API provided by an application server.

In the telephony world, the APIs often used are Parlay[2], the Telephony API (TAPI™), the Java™ Telephony API (JTAPI)[3], in addition to a host of proprietary APIs that are specific to each

system. In the Web world, the most commonly used APIs are servlets, the Common Gateway Interface (CGI)[4], Active Server Pages (ASP) and Java Server Pages (JSP).

Traditional IN platforms are typically closed; few people are trained in their usage, limiting service development to a select few. In the Web world, application platforms are based on open and widely-known interfaces. These interfaces are standard training for computer science graduates. The result is that large communities of developers are available to build Web services.

### *Development Cycles*

Traditional voice APIs are not generally understood by the typical software developer. They usually require expert training and experience in the field before they can be used. This is because they rely heavily on telecom concepts (such as call models), telecom services and telecom technologies. The APIs are usually encapsulated across multiple, lengthy documents, which are not self-contained and easily read. Limited external documentation (in the form of books, tutorials and tools) is available. Using the APIs requires access to specific vendor systems on which to test. These systems are not widely available.

This is in stark contrast to traditional Web application development. The APIs used there, which include servlets, CGI and EJB, are much more widely understood. The API specifications are freely available. They are generally small, self-contained and written with the developer in mind. Typically, extensive additional materials exist on these APIs, in the form of books, Web tutorials and slideware. Servers on which these APIs can be executed are widely available for little to no cost and often are included within Integrated Development Environments (IDEs).

The result is a significant difference in the timetables to develop and deploy applications. The reasons are simple:

- With a much larger community of expertise in Web APIs, it is easy to find third-party developers or to quickly hire a trained staff that can be immediately effective.
- The simplicity of the Web APIs, compared to the legacy PSTN APIs, makes it easy for new developers to come up to speed. Coupled with the ready availability of books and tutorials, the learning curves are substantially different.
- The simplicity of the Web APIs make them easy to program, resulting in more rapid engineering cycles.
- The wide availability of platforms to run the applications makes testing and lab evaluation much faster.

Overall, the result is an accelerated time from concept to deployment. This is evident by the tremendous pace of Web application development. In the Web, there are literally thousands of different applications. These cover every area of interest, from bartering, trading and commerce applications; to medicine, law and technology advice systems; to food, restaurant, shopping and buying applications; to simple search engines. In contrast, only a few dozen applications have seen significant use and deployment over the PSTN.

### *Models of Abstraction*

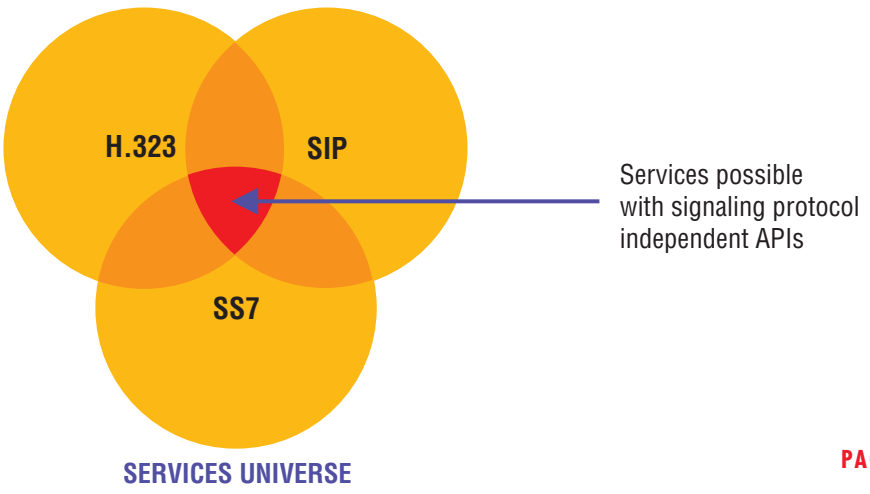
Traditional voice platforms and APIs focus on "call models" (such as the IN defined Basic Call State Model, or BCSM) as the fundamental model of abstraction. These call models represent the processing space of a switch on a call as a finite state machine. An application is constructed by processing events related to the call – which by definition, are the state transitions that occur in the call model. The processing usually includes manipulation of the call in some way, by either forcing it into a specific state or performing some defined operation that is allowed within the specific state (such as a transfer or hold). These states are fixed and pre-defined thus limiting the types of applications that can be supported.

The APIs used for building Web applications are much different. They are transactional in nature. Servlets and CGI are both strongly rooted in the request/response paradigm of HTTP. These APIs also allow access to much of the protocol details if needed, but do not require anything beyond the basics. However, the APIs in no way try to hide the details of the protocol.

When considering APIs for VoIP service development, the two models have profound implications on the types of services that can be constructed. The call model-based APIs generally have the property that they are independent of the underlying signaling protocols. APIs like Parlay can be used for SIP, H.323 and SS7-based services. While this may appear to be a strength, it is actually a weakness. The problem lies in the fact that all protocols are not created equal. The set of services enabled by SIP is not the same as those enabled by H.323, which is different still from the set of services enabled by SS7. This is shown in Figure 3. The caption depicts the services universe and the set of services enabled by three specific signaling protocols. The figure clearly shows the APIs that are independent of the signaling protocol (and thus work for any protocol) can only enable services at the intersection of those possible with the signaling protocols. This, unfortunately, eliminates the possibility of those APIs providing any new services that would otherwise be enabled by the protocols. SIP, in particular, enables numerous new services through its integration with the Web and URLs; all of these capabilities would be limited by an API such as JTAPI.

In contrast, Web style APIs have a much better services model. Since they allow access to the capabilities of the protocols, application writers can make use of those protocol services that enable new applications. The use of Web style APIs means that it is easy to access back-end services that have already been built for use in Web applications.

***FIGURE 3: APIs AND THE SERVICES THEY ENABLE***



### *Vertical Markets*

Traditional voice platforms have been unsuccessful in delivering vertical telecommunications applications. This is due, in large part, to the lack of expertise in these systems and their APIs across the software developer population. It is also due to the smaller volume of sales that can be generated by vertical applications. When coupled with the long development cycles and high cost of operations, the poor profitability of these applications has prohibited them from being deployed.

Interestingly, this has not been the case on the Web. Nearly every vertical market (lawyers, doctors, animal lovers, stock brokers, etc.) has a dozen Web application providers in its space. Clearly, Web platforms have been very successful in delivering highly specialized applications and services. The reasons are simple:

- Development costs of these applications are relatively low. There are countless Web development and consulting firms that can quickly build applications. The wide availability of platforms on which to run them makes it easy for these firms to develop and test the applications before they are launched.
- Operation costs of these applications are relatively low due to the wide availability and ease of use of the platforms they run on.

### *Summary*

Clearly, the Web model for service development and deployment has advantages for VoIP. The wealth of developers experienced in the models and APIs, the quick turnaround times on development, the availability of training resources and tutorials, the easy access to platforms and the ability to develop new services, point towards the Web model as the clear choice for building a next-generation VoIP services platform.

## CREATING PLATFORM REQUIREMENTS

Our conclusion that an application server architecture must be built upon Web technologies, be open and easy to use with people with traditional Web development experience and enable new services that combine voice, video, IM, presence and Web, generates three architectural requirements for an application server.

**Web, IM, presence, email, voice and video must be integrated at the services layer.**

Until now, the notion of “integrated networks” has really been integration at layer 3. A single data network would deliver both voice services and data services. However, the platforms that deliver these services and the services themselves, are still separated. To build the integrated services we have been discussing, these networks and platforms must be joined at both the network and the services layer. What does this mean?

- Integrated APIs: The application server must provide APIs that allow for manipulation of voice *and* Web *and* presence *and* IM *and* email; not separately, but together.
- Integrated Deployment: The process of deployment should support the deploying of the voice piece *and* the Web piece *and* the presence piece *and* the IM piece *and* the email piece together, as a single, unified application, rather than requiring the application provider to deploy each piece separately.
- Integrated Management: The process of management should allow the application provider to manage the Web piece *and* the voice piece *and* the presence piece *and* the IM piece *and* the email piece all from the same console, as if they were a single manageable entity, not many distinct and isolated entities.

**Call model-based APIs are not sufficient.**

As we have already discussed, traditional, call model-based APIs are not sufficient for creating next-generation services. Application server platforms therefore must provide APIs, preferably built on Web technologies, which can provide the tools needed to build both traditional applications and new ones. The distinction between a traditional API and traditional application is important here. An API that provides access to new services can still be used to build old applications, but an API that does not provide access to the capabilities of next-generation protocols cannot provide new applications.

**New feature and service composition tools are needed.**

In the traditional telephony world, service providers provide more than a single feature or service – they provide many of them. In fact, a single customer can have multiple services and applications enabled for them at the same time. This has led to the problem of service interaction – how to deal with the interactions between multiple services, written independently of each other, when they try and handle the same call. A classic example of the service interaction problem is the combination of outbound call screening and call forwarding. Outbound call screening will block outgoing calls to specific numbers, often 900 numbers. Call forwarding allows a call for a specific destination to automatically be redirected to another. A problem occurs when both services operate on a single call. If a subscriber has outbound call screening enabled, but calls a number that has call forwarding enabled, the forwarding could cause the call to be forwarded to one of the disallowed numbers. The result is that the screening is effectively disabled. A more intelligent ordering of service execution can solve this problem – the forwarding service should run first and then the screening service should be performed.

Unfortunately, combined services make this problem more difficult. We now have to worry about the combination of services and applications which themselves combine voice, Web, email, IM and presence. For example, imagine an “IM screening” service that disallowed outbound instant messages to selected chat rooms. Similarly, one could imagine an “IM forwarding” service, which took instant messages received for a particular user and forwarded them to another. If these two services are running at the same time, an interaction problem (similar to the one described for voice) can occur for IM as well.

As a result, application servers must provide new tools for managing service interaction that allow them to work across multiple application protocols.

## THE SOLUTION: DYNAMICSOFT APPLICATION SERVER

The dynamicsoft Application Server is the first truly converged application execution platform. It provides exactly the solution demanded by our analysis above – it is built upon Web models and APIs, provides unified access to voice, Web, IM and presence services, provides service interaction management for converged applications and opens up development to the masses by building off of open, well-established APIs.

Its architecture is depicted in Figure 4.

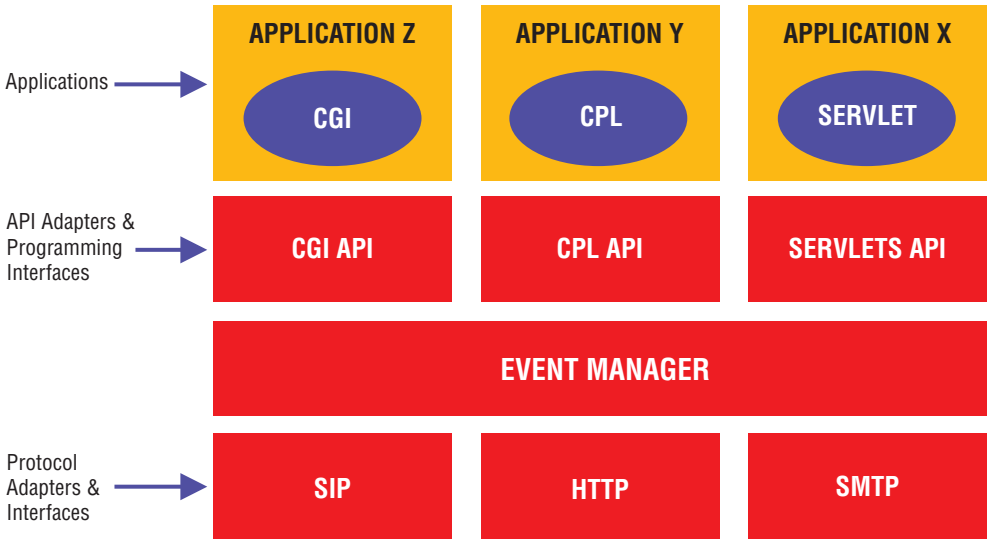



*FIGURE 4: DYNAMICSOFT APPLICATION SERVER ARCHITECTURE*

The Application Server is composed of a number of components:

- Protocol Engines provide support for the various protocols that can be used by the Application Server to deliver applications. These include the Hypertext Transfer Protocol (HTTP)[5] for Web and the Session Initiation Protocol (SIP)[6] for IM, presence, voice and video.
- The Event Manager (EM) is a protocol-independent rules engine and service composition tool. Based on patent pending dynamicsoft technology, the Event Manager is capable of determining which applications need to handle a particular protocol event and can compose them together based on administrator directives for service interaction handling. After determining which applications will handle which events, the EM passes the protocol event to one of several containers. The EM allows a single "call", or more generically, any kind of protocol event, to be handled by multiple applications, each of which may even run in separate containers. It automatically manages their interaction, hiding the existence of any other application from each one. This allows applications to be developed independently of each other.

- Containers are sub-systems that provide API support for delivering services. The containers are the embodiment of the converged APIs that we have been discussing. Containers also provide management and configuration interfaces to the system administrator.

The dynamicsoft Application Server presently supports three containers:

  - The Unified Servlet Container, as its name suggests, hosts both HTTP and SIP servlets and supports the sharing of session state between them. The container also allows application writers to initiate requests, an important component of many converged services.
  - The Unified CGI Container, like the Unified Servlet Container, is a hosting environment for both HTTP CGI scripts and SIP CGI scripts (written in languages like PERL). It builds upon the traditional CGI models by supporting sessions through a rich lifecycle management system. Similar to the Unified Servlet Container, scripts can send requests if needed.
  - The CPL Container is an extensible execution environment for CPL scripts. It provides the service provider with an extremely rapid way to develop and deploy a fairly broad set of services and applications. Additionally, the structure of the CPL language makes CPL scripts ideal for end user created services. To that end, a variety of deployment models are supported for the storage and maintenance of scripts.

Some additional features of the dynamicsoft Application Server include:

- Carrier-grade scalability and reliability, including support for load balancing and centralized management over groups of servers
- Easy to use, GUI-based tools for administration, provisioning and subscriber management
- Support for billing, monitoring and usage recording

[1] *http://www.dialpad.com*

[2] *http://www.parlay.org/*

[3] *http://java.sun.com/products/jtapi/*

[4] *http://hoohoo.ncsa.uiuc.edu/cgi/*

[5] *R. Fielding, et. Al., "The Hypertest Transfer Protocol v1.1", IETF Request For Comments RFC2616, June 1999*

[6] *M. Handley, E. Schooler, H. Schulzrinne, J. Rosenberg, "The Session Initiation Protocol (SIP)", IETF Request for Comments RFC2543, April 1999*

**CONTACT US @ +1 973.952.5000**

**www.dynamicsoft.com**

**sales@dynamicsoft.com**

dynamicsoft®