

WHEN IS ZERO CLIENT NOT ZERO CLIENT?

THE ZERO CLIENT VALIDATION BREAKTHROUGH

BY IRV SEGAL, SENIOR CONSULTANT, SYSGEN, INC.

Zero client software seems to be all the rage in today's 21 CFR, Part 11 conscious market. It sure makes for a slick marketing campaign. But what does it really mean for firms struggling with validating a web-based computer-related system that supports FDA regulated functions?

Software vendors would have you believe that this means you don't have to place client systems under validation controls. Whether they explicitly state this in their marketing materials or verbally express this during a sales presentation, the inference is that a zero client software Web-based system does not require any validation controls on client systems, since there is no software to install on the client PCs.

At least, that's what these software vendors are claiming. Are they right? Not necessarily.

What does Zero Client Software Really Mean?

As with many aspects of validation there is a terminology disconnect between what a computer technician refers to as zero client, and what zero client means from a quality perspective.

To a computer technician the phrase "zero client software" means that when implementing a given software package there are no software components that must be manually installed on the client systems. This is truly a technical advantage because it dramatically reduces the time required to deploy a new software application.

But from a quality perspective there really is no such thing as zero client software.

Quality engineers are not concerned with whether or not the software is manually installed on client machines. The concern here is whether or not software instructions are being processed by the client system, regardless of how those instructions get there. As of this writing I do not believe anyone has developed technology enabling a client system

to interact with a host without processing some level of software instruction. Even truly Web-based systems send at least some form of HTML to the client system. And the client system must properly interpret these instructions in order to display the Web page. This requires that the client system have software (i.e. a Web browser) that is capable of properly interpreting the HTML instructions sent from the host.

To prove my point, simply look at the client system requirements these zero client software vendors specify. There is typically a limitation on the browser version (i.e., Microsoft Internet Explorer v 5.01 or higher required, etc.) and sometimes on the operating systems that are compatible (Windows PC only). If there were truly zero software instructions being interpreted on the client system, why would these client system requirements be necessary?

Obviously even zero client Web-based software has some form of software being sent to the client system for interpretation.

That is why from a quality perspective there is no such thing as a zero client software system. So when we talk about zero client software from a quality perspective (which is the focus of this article), we will refer to it as "zero" client software.

Is One "Zero" Client Software Design Better Than Another?

The answer to that question is a resounding YES! I say resounding because it sets the stage for an exciting validation breakthrough that can be accomplished given the proper software design approach.



"PROPERLY DESIGNED SOFTWARE IS SOFTWARE THAT BY INTENTIONAL DESIGN IS STRICTLY LIMITED TO SOFTWARE INSTRUCTIONS THAT HAVE BEEN STANDARDIZED TO SUCH A DEGREE THAT IT IS REASONABLY RELIABLE WITHOUT ANY CONCERN ABOUT THE ENVIRONMENT ON THE CLIENT SYSTEM."

For the first time since computer-related system validation began sending shivers down the spines of IT directors everywhere, there is an opportunity to eliminate client system installation and operational qualifications, and forego change controls as well.

However, that opportunity can only be realized with properly designed Web-based software.

What is this "properly designed" Web-based software? For the purpose of eliminating client system validation, "properly designed" software is software that by intentional design is strictly limited to software instructions that have been standardized to such a degree that it is reasonably reliable without any concern about the environment on the client system.

In other words, software that will consistently perform as expected on any computer capable of accessing the Internet, period. No browser or operating system limitations. No minimum processor speeds. Nothing else. The only requirement for a client system interacting with this "properly designed" software is that it is capable of accessing and browsing the Internet. That, is a "properly designed" Web-based system from a quality perspective, because it paves the road to zero client validation.

The HTML Advantage

What is this magical software that works with any Internet-capable client system?

Pure HTML.

Not JavaScript. Not ActiveX Controls. Not VBScript. No Java Applets. Just plain old-fashioned HTML.

But doesn't HTML have browser compatibility issues? Of course it does, if you use the latest version of HTML. But who says you have to use the latest version? Earlier versions, though limited in functionality, have a big advantage.

They have the advantage of time. Over time all widely used commercial browsers adapt to newer HTML versions. The once inevitable compatibility issues of older Web-based software slowly dissipate into oblivion.

And by evaluating the performance of these unchanged older, Web-based software applications over time we have seen how reliable they become. Because of this standardization process they reach the point where they consistently produce expected results regardless of the client system's environment.

Left unchanged, these software applications become highly reliable. Which is how they open the door to something far more exciting than "zero" client software. They provide the path to zero client validation! It is their inherent reliability that opens the door to that path.

How old of a version of HTML would you have to use to achieve this level of reliability? Where do you draw the line? You'd have to find a version of HTML that no longer has a single compatibility issue with any of today's commercial browsers. It would have to be a version that has been experiencing zero compatibility issues for a length of time that reasonably assures its reliability. But where do we draw this line?

To find out I called Eric June, Chief Software Architect at AssurX, Inc. AssurX was the first vendor to tout the "zero" client software buzzword in the Part 11 arena. They produce CATSWeb, a Web-based, Part 11 compliant, Corrective/Preventive Action (CAPA) system. They are also the only vendor I have seen that can claim without question to have a system where the required level of reliability is inherent enough to forgo validation on client systems.

"When we realized how big a concern compatibility would be", June said, "especially to validation managers, we made a steadfast commitment to adhere to HTML v3.0 standards. Our competitors may deliver a few more superficial bells and whistles than we do, but we deliver a system that eliminates the need to validate client systems. Our customers see this as a huge financial and logistical advantage, when compared to the minor enhancements that JavaScript and other executables offer."

HTML v3.0 is essentially the version of HTML upon which the Internet boom was launched. Today you would be hard-pressed to find a

browser still in use that would have a compatibility issue with HTML v3.0. In fact, a look at the system requirements for CATSWeb essentially states that client systems must have at least Internet Explorer v3.01, Netscape Navigator v3.0, or any equivalent browser. You'd probably have to rummage around a computer scrap yard to find any system still running these archaic browser versions.

That means we can reasonably assume that any Internet-capable client system in use today meets the minimum requirements for accessing the CATSWeb software. Hence, there is no need to place these systems under any validation controls. I believe June is right on the money about the huge financial and logistical advantage CATSWeb delivers. Any validation manager who has validated a large-scale computer-related system can tell you that the time, expense, and logistical headaches that are part and parcel of controlling client systems is a very big problem indeed.

The validation effort required to properly control client systems is so monumental that it has prompted some firms to take aggressive and somewhat risky positions. One consultant I spoke with while researching this subject informed me that one of his larger clients simply performs random tests of various configurations and documents that these systems perform as expected under those configurations. But they do not properly qualify each client system, and they surely have no basis for controlling changes to those systems.

Suppose you have a server hosting a regulated application for an international concern. Further suppose that there are 500 client systems accessing that server. Each of these devices must be placed through the paces of an IQ/OQ and have a properly established basis for change control. How would you even begin to identify all 500 client systems? Are they all company owned? Are they being taken off-site? Who makes sure that these systems are properly controlled during future changes? How do you coordinate the change control training that these 500 users must now undergo?

Furthermore, even if we conquer the validation of these 500 client systems, how would we control access to a Web-based system? How could we prevent access from Internet cafes or the local Kinko's, where we have no controls whatsoever on the client systems?

The answer is that we cannot control this at all. Which raises an interesting point. If we are unable to fully control the client systems that access our Web-based applications how can we justify using anything other than a system like AssurX's CATSWeb? It is literally impossible to open a system up to Web-access and gain the proper controls over all client systems.

How Does HTML Equal Zero Client Validation?

You may be wondering how the level of reliability delivered by versions of HTML such as v3.0 translate into forgoing client system validation. After all, these validation activities seem to have become a way of life around computer-related systems in a regulated environment. How can we suddenly just do away with them? Won't regulatory inspectors be looking for validation controls? How do we justify this?

The answer lies in the fundamental basis for establishing the need to validate computer-related systems in the first place. The rationale behind validation controls is often misunderstood. Technical folks often see it as a necessary evil that comes along with life in a regulated environment. The red tape is simply inescapable. Even validation engineers often don't take the time to think about why they do what they do.

Believe it or not, the FDA did have sound fundamental reasons for requiring computer-related systems to be validated. In fact, it is the same reason that the government established the FDA in the first place. FDA was established to assure that the products they regulate are of a reasonable level of quality and produced in a consistently reliable manner. Everything the FDA does is in furtherance of that purpose.

When the FDA decided that computer-related systems require validation controls it was

●

"THE VALIDATION EFFORT REQUIRED TO PROPERLY CONTROL CLIENT SYSTEMS IS SO MONUMENTAL THAT IT HAS PROMPTED SOME FIRMS TO TAKE AGGRESSIVE AND SOMEWHAT RISKY POSITIONS."

●

because these systems were being used in the production and processing of the products they regulate, and because there was no other means to assure that these systems would produce a consistently reliable result. Had computer systems been producing consistently reliable results, the FDA would have never dreamed of implementing validation controls over these devices.

Computer-related systems are comprised of many different components that do not conform to a single standardized technology. This introduces so many compatibility issues that validation controls are the only way to ensure the reliable performance of these systems.

Will computer-related systems ever achieve a level of standardization that makes their reliability a reasonable assumption? Probably not completely, at least not in our lifetimes. But there is now a portion of these systems that can be deemed to have achieved this level of reliability.

The use of HTML in a manner such as AssurX's CATSWeb product creates a system where the client computers can reasonably be expected to perform in a consistently reliable manner. And this technology has proven itself over a period of time to demonstrate this reliability.

Something Old is New Again

Amazingly, by properly designing a Web-based application with a version of HTML that has achieved a level of standardization that is proven consistently reliable, we have turned the smart client system right back into the old dumb terminals of yesteryear. But these new dumb terminals have one strategic advantage over their predecessors: They are not a hard-wired component of the computer-related system.

The one advantage of the new dumb terminal (i.e. the client system in a Web-based environment) is that we no longer hard-wire these devices to the host. They are in fact not even a component of the system. They represent a standardized, reliable communications device that is used to retrieve and send Web pages to

the host server, like a fax machine that sends and receives documents via telecommunication technology.

For the first time in history this provides the opportunity to eliminate a large portion of the validation nightmare for computer-related systems. Apparently it took the foresight of one lonely Chief Software Architect to get to this point.

Why Don't All Part 11 Compliant Software Vendors Use Pure HTML?

It is a wonder that more software firms engaged in the highly competitive Part 11 compliant market don't make this connection. Old, standardized, proven-reliable technology leads to less validation. Less validation leads to lower operating costs and easier implementations, which lead to happier customers. And happier customers lead to more sales.

There is only one thing standing in the way of zero client validation controls becoming a widespread reality: Software technicians.

The very people who engineer these software applications are the ones keeping their customers locked in validation chains. The programmers, analysts and architects who design and build "zero" client software don't want to work with older technology.

We should thank the validation gods that there was at least one Chief Software Architect out there who really got it. Let his foresight and steadfast determination be a lesson to software architects everywhere.

ABOUT THE AUTHOR

Irv Segal is a Senior Consultant with SysGen, Inc., a consulting firm specializing in FDA-regulated computer systems. In addition to over fifteen years experience in computer programming, systems analysis and project management, he has more than thirteen years of experience creating, validating, and auditing FDA-regulated computer-related systems for a variety of pharmaceutical and medical device manufacturers.

Irv Segal may be reached via email:
irv.segal@sysgeninc.com

© 2002 SysGen, Inc. All rights reserved.

AssurX, Inc.

305 Vineyard Town Ctr.

Suite 374

Morgan Hill, CA 95037

Tel 408.778.1376

Fax 408.776.1267

www.assurx.com

AssurX

