



## Screen Scraping: A Wolf in Sheep's Clothing

## Copyright

Copyright © 2004 ASNA, Inc. All rights reserved.

## Restricted Rights

This document may not, in whole or in part, be photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without the prior consent, in writing, from ASNA Inc. Information in this document is subject to change without notice and does not represent a commitment on the part of ASNA, Inc.

## Trademark

ASNA<sup>®</sup>, ASNA Visual RPG<sup>®</sup>, ASNA DataGate<sup>®</sup>, ASNA Monarch<sup>®</sup> and/or other ASNA products referenced herein are either trademarks or registered trademarks of ASNA Inc. All other product and company names mentioned herein may be the trademarks of their respective owners.

## Table of Contents

What is Screen Scraping? .....	4
Vendor Positioning .....	5
The Reality.....	6
The Issues.....	6
A Scenario.....	7
How the ASNA Solution Set Compares .....	8
About ASNA.....	9

## What is Screen Scraping?

There is much hype in the market place around screen scraping as a potential strategy for modernizing or extending legacy applications. It started out as a point solution for windows to iSeries<sup>[1]</sup> on a screen-by-screen basis. Now they are positioning themselves as a comprehensive strategy for modernization. It is called “non-intrusive” and “easy”, because you do not have to be (in the case of the iSeries) an RPG programmer to use it. The screen-scraping concept took hold in the iSeries world back in the early 90’s, when it was touted as the most effective method for putting a “windows” front end on iSeries applications in MS Windows and in OS/2. At the time the concept made a great deal of sense in many cases, since developing scalable client-server applications for the iSeries was next to impossible.

Over the years, screen scraping has slowly morphed into a progression of increasing “intelligence” on the client side as vendors provided capabilities to move fields around on the newly formed screens and eventually to even be able to combine fields from varying green screens onto new windows. This allowed a business to in effect re-engineer their business through revamping the progression of data entry, update and retrieval.

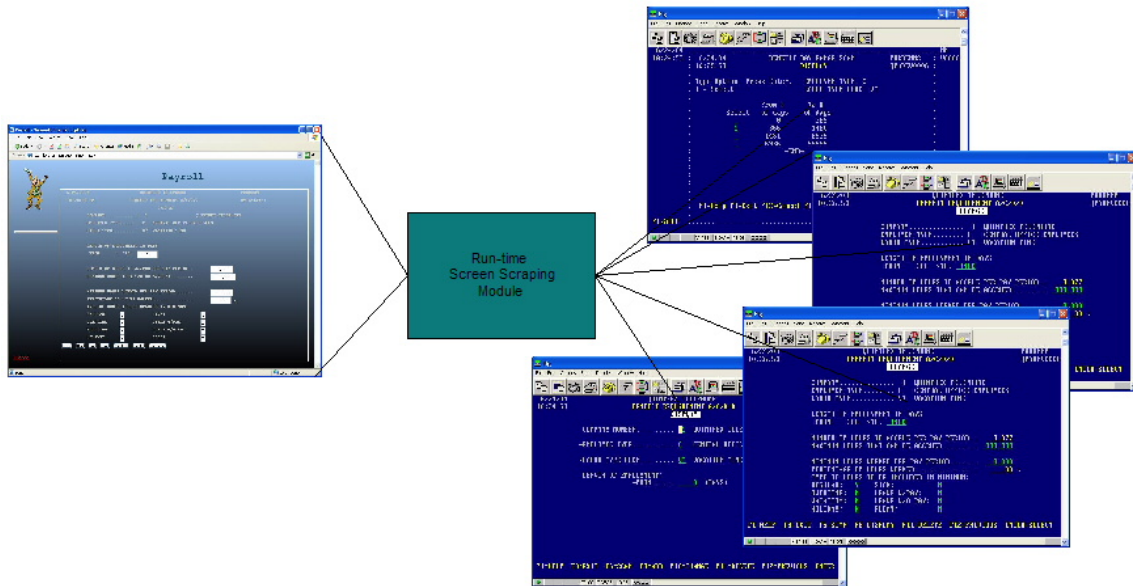


Figure 1. Showing multiple fields from different screens going to one window.

Over time, the screen scraper vendors evolved to a new model – that of encapsulating the screen data streams into web services to be consumed in an SOA model. For instance, imagine a green screen, which shows an invoice, based on an invoice number that is entered.

[1] Throughout this document, the term ‘iSeries’ is used to refer to the S3x, AS/400, iSeries and i5 platforms.

The web service that would encapsulate those steps would receive as input the invoice number and would in turn send that invoice number to the screen data stream, which would in turn go to the iSeries based program which would look up the invoice and ship that back out via the screen data stream. The web service would scrape that data out of the stream and translate it to XML for use by whichever application consumed it. This is the latest incarnation of screen scraping, which is built on the concept of allowing legacy applications to be brought into a web services environment.

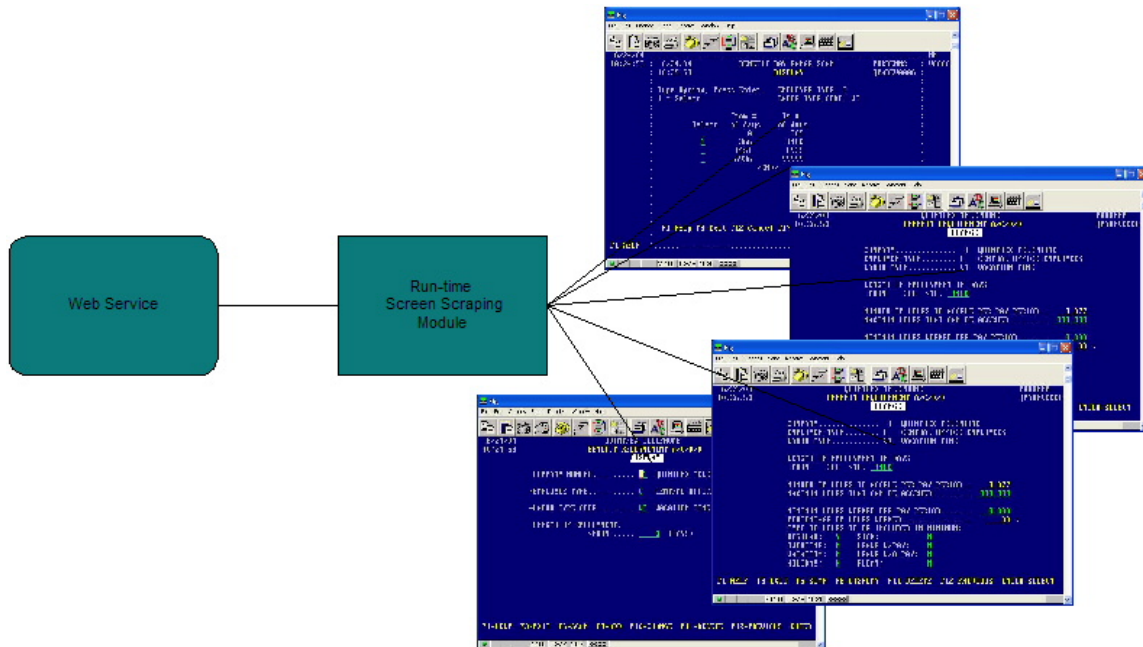


Figure 2. Showing the web-service example.

## Vendor Positioning

There are many vendors out there who provide this functionality, including Jacada, Seagull, WRQ and IBM. They position this solution as “non-intrusive” in that one need not know the inner workings of the applications on the iSeries and further will not have to make any changes on the iSeries. It is completely hands-off from the iSeries side. Also, the vendors position it as a rapid deployment solution given that the web services are relatively easy to build around the green screen data streams.

However, the bulk of their positioning addresses the SOA model, Composite Applications and business process improvement. With their technology, it is possible to utilize the mission critical functionality of the legacy applications in heterogeneous environments, incorporating any windows, web or host-based custom or packaged application functions. Furthermore, with their technology, without changing the legacy application in any way, the customer can redesign business process flow, or even create new business process capabilities through assembling web-services in new ways. So the screen-scraping vendors posit that customers can enable customers to repurpose their most valuable assets and rapidly turn them into services which enable new environments and architectures.

## The Reality

Sounds great, huh? Oooh! Non-intrusive! Wow! Repurposing assets! How cool!

Well, there are some pretty significant problems with the model.

Back in the 90s, screen scraping was adopted in many cases and did well as a stop-gap to getting to the iSeries from Windows. Back then it made sense for a few point solutions that absolutely had to be accessible via windows. However, it did not make sense from a broad application perspective for the simple reason that it was “non-intrusive”. While the vendors positioned non-intrusive as a benefit, it was in actuality the Achilles heal of the model, because the applications that were scraped could not be changed without significant pain.

It quickly became apparent that anyone who created a new business process flow based on any number of 5250 data streams had inadvertently “locked” the code on the back end, because if they changed it, it would result in unpredictable results at best and just plain false information at worst.

At that time, iSeries shops were still maintaining and updating the functionality of the iSeries-based applications sufficiently that the screen scraping applications would quickly fall out of date. Any reasonably sophisticated application that re-engineered a set of business processes would quickly become unwieldy when updates to the underlying code were necessary. This problem was the reason that screen-scraping did not see wide and deep adoption in the iSeries community. It was much more pain than it was worth. Though the end-use of the model has changed (from new windows to web-services), the fundamental problem of forcing code to be static has not changed. However, today there is less change being applied to the back-end applications in the iSeries environment. As such, the screen scraper model does have more appeal on the face of it, given that the static nature of the code is not as large a problem. However, there are other, even more troubling issues with the screen scraping model.

## The Issues

The first is somewhat esoteric, but nonetheless presents a potentially serious level of complexity for developers. It feeds into what is the largest fundamental flaw for screen scraping, which is addressed in a moment. Applications on the iSeries are typically quite large, consisting of thousands (if not tens of thousands) of programs and millions of lines of code. The underlying database is typically quite large and complex as well and is normalized to between the 2nd and 3rd levels of normalization. Business application code developers in any environment live and die by the fundamental concepts of the relational database.

However, screen scraping does not. It has nothing to do with relational databases. Instead, the data model for the screen scraping concept is built on fields from screens, not fields and tables in a relational model. As such, when implementing a screen scraping

model with an enterprise wide application, the development environment quickly builds what is in effect a non-normalized and non-relational data model based on screens. This makes it very difficult to enforce some of the precepts of relational databases, such as referential integrity and “one fact in one place” (and of course semantic dependencies go through the roof!).

Much of this problem is manifested in the way that data is presented on screens in business applications. The data on a screen is often derived from the combination of multiple fields from a database. Further, they may be “temporally massaged” – meaning that it is massaged for presentation for a specific point in time (“Days Past Due” as a derived value on a screen, for instance). What this means is that the development staff has to build and maintain yet another metadata model for the application that is specific to the screens from the application – one that is necessarily complex as it is built based on thousands of screens.

That is not a trivial task and certainly injects a level of complexity in application development that would be expensive in time and money to maintain and it would have a significant impact on how web services could (and could not be) built and maintained.

All of that being said, the extra abstracted metadata layer of the database is not the biggest issue with screen scraping. The fundamental flaw of the screen scraping model is that it vastly limits an organization’s options for migration down the road. The model presupposes that the underlying legacy system will always be available, that the platform on which it lies (in this case, the iSeries) will always be available, will always be supported, will always be maintained and that the organization will never want to migrate the mission critical business applications off of it. That is a dangerous assumption.

Any organization which maintains its competitive advantage (its business processes, methods, etc.) on a platform will always want to keep its options open to migration to other platforms. Too many companies have been burned because they built their business processes on technologies that have gone the way of the dodo. This does not necessarily mean that they are imminently planning to migrate, but if they find that they need to, they want to have as many options available to them as are economically and realistically feasible. This is where the screen scraping model falls down.

## A Scenario

Imagine that you have the typical business application built on the iSeries: Thousands of programs and screens, potentially millions of lines of code. Now imagine that you used a screen scraping technology to build a large, functionally rich set of web services to “front-end” all, or at least many, of your screens and programs. Over several years, your developers build potentially thousands of web services, which are consumed by other web services or web servers, or BPM tools such as BizTalk or portal applications like SharePoint or even custom composite applications to facilitate new business processes incorporating function and information from a myriad distinct applications. The list goes on and on.

Now suppose that company that sold and supported the legacy platform, decide to no longer support that platform. Or perhaps the vendor of that platform begins a forced effort to migrate you somewhere that you just plain do not want to go (to java and Websphere, for instance). But wait a minute... You have all those applications built on all of those web services that are built on all of those screens on a platform that is going away.

Now what do you do? Well, first your blood runs cold. Then you update your resume because you recall that you came up with the idea in the first place. Or you decide to forge ahead. By moving your data from that legacy platform and then meticulously resolving all of those old screens to the underlying database fields, going through all of that old RPG code to see how the database fields were massaged, then going through and re-writing all those thousands of functionally rich web services so that they can still be used by all those applications that you built on top of them. Pretty non-intrusive, huh? This is a somewhat extreme example. But the point to not be lost is that the screen scraping model limits a company's options by only prolonging addressing the underlying issue – that of being in charge of your own destiny as it relates to your business application platforms. The screen scraping model negates this basic tenet. It forces a company to become even more reliant on the platform of the legacy environment than less. And if the company decides for what ever reason that they wish to migrate to another platform down the road, the screen-scraping model makes it vastly more difficult than less. The screen scraping model grossly limits one's options for future platform decisions.

## How the ASNA Solution Set Compares

The ASNA solution set takes a different approach to modernization and extension. The fundamental difference between screen scrapers and the ASNA technology is that it does not rely on the 5250 data streams for data stores. Instead, it goes directly to the database. As such, the only metadata model is that of the database.

Furthermore, the ASNA solution set is specifically engineered so that a company can continuously maintain its options to migrate from the ISERIES to .NET if it so desires. The legacy system is not “locked” into the legacy platform. Since the ASNA solution set facilitates bringing the actual code over to Visual Studio .NET and facilitates direct database access on the iSeries or SQL Server, a company can bring mission critical business applications over to the .net environment making them available in a web services oriented architecture for use within composite applications, BPM tools, portals, etc.

DataGate, as a middleware solution works equally with the iSeries database as well as the MS SQL. So the database, when necessary, can be migrated over to MS SQL – a move that is transparent to the code. There is no need to change the web services when the move is made.

The similarities between the offerings of the screen scrapers are that they both provide the ability to make iSeries applications available in a modern, web services environment, adhering to the standards of service oriented architecture, which in turn can be used in composite applications in heterogeneous environments. But the similarities end there. With ASNA, customers go direct to the database, while with screen scrapers, developers must build and maintain a complex abstracted metadata model with which to work. With ASNA, the underlying business application itself can evolve and change if necessary, while with screen scrapers the underlying business application necessarily cannot evolve or change; it must remain static.

And most importantly, with ASNA, a customer always has options open to them to migrate off of the iSeries if they so desire, while with screen scrapers, the customer is literally locked into a legacy platform that may very well go away.

## About ASNA

San Antonio-based ASNA (Amalgamated Software of North America) was established in 1982 and develops and markets unique software products that evolve IBM AS/400 and iSeries systems. Aligned with Microsoft's .NET initiative, ASNA is the only company to offer a thoroughly conceived, standards-based extension and migration path that solves its customers' business challenges. For more information about ASNA: <http://www.asna.com/>