ASNA Staged Application Modernization and Migration with Wings and Monarch

Most IBM i-centric businesses face one or both of two core challenges today:

Challenge 1: Frustrations with the green-screen user interface. The frustrations associated with the green-screen user interface are legendary and universal. Although a green-screen does indeed provide a superb heads-down data entry experience, that isn't the point. Despite the positives the green-screen offers, the overwhelming negative is that the green-screen is used, almost universally, as a measure of the quality of the application. This is especially true for ISVs trying to sell a green-screen application, but it's also true for the vertical RPG application used 100% internally. It's the rare prospect today that would consider buying a spanking-new character based application. And lest you think that because your application is only used internally you don't need to worry about your application's perception—try telling that to the new CIO with a Windows or Linux background. Alas, the merits of an application aren't measured by how infrequently you need to IPL the box or how the database never gets corrupted, the gold standard is if the application offers pull down menus, radio buttons, and gradient color transformations.

Challenge 2: The need to migrate off of RPG and perhaps even the IBM i platform. RPG, which is well into its fourth decade of service to many businesses, has done a superb job providing the backbone of the enterprise application portfolio. Many RPG-centric businesses today, though, are starting to feel two RPG pinches:

***The inability of RPG to quickly provide the features the business needs to stay competitive*.** IBM has done a superb job over the years of propping up RPG to make it keep pace with modern demands—but despite these efforts, in today's e-commerce, browser-based, object-oriented world, RPG is clearly starting to show its age with constraints such as its inability to natively produce a browser-based user interface, its lack of modern mechanisms for business-to-business commerce, and its inability to provide a reusable framework for quickly scaffolding new applications. These all limit he choices you have for building new RPG applications or enhancing existing ones.

The shrinking, and ultimate demise, of the RPG talent pool. Universities don't produce new RPG programmers anymore. They haven't for years. For many businesses, the day is rapidly approaching when the venerable internal RPG team will close its briefcase for the last time and head out to live the rest of its years in retirement. In virtually every case, this doesn't mean just the retirement of RPG application programmers. Rather it means the retirement of the core group of developers who know, deeply, how your business and your applications work. The documentation, the data flows, the application specifications for the million-plus line of RPG on which your business is nearly wholly dependent, exists only in the heads of these RPG coders who are rapidly heading out for the beach. Or the retirement home. Or the interstate in their RV. Never to return to their desks in your office.

It's very clear that businesses using the IBM i today have some very deep thinking to do about the future of their RPG application portfolio. In the past, making strategic decisions about your IBM i were tough—partly because IBM's strategies-du-jour (Java, no PHP, no EGL, no wait, PHP, for sure this time) didn't help in any way. Partly because there were few decisions you could make that weren't the IT equivalent of putting all your money on Red 21 in Las Vegas. Decision makers need an alternative they can select that gets them tactical results quickly, but also provides the ability to later grow into an enterprise-wide strategic solution.

Let's be very clear about the challenge: IBM i customers need to ensure that their RPG application portfolio can continue to provide business value well into the future.

Some of the issues involved are driven by UI frustrations; some are driven by uncertainty about the ability of the RPG application to persist into the future. Traditionally, the tactical answer to the UI frustration was to use some form of screen scraper. Something that at some level attempted to trick the user into thinking the 5250 data stream wasn't a 5250 data stream. While some of these products did provide some level of sophistication, most users can spot a screen scraper a mile away. Beyond the off-by-one aspects of screen scraping, another issue with them is that the investment they require, and the results they deliver, won't do you any good should you ever need to migrate your RPG application away from the IBM i.

On the low end, screen scraping is just a stop-gap measure—something that temporarily solves the problem until you find the real answer. And that real answer has been elusive. There are RPG transformation or migration products, but those tools have a huge development cycle and often take years to deliver their full results (we know, we sell one!). IBM i decision makers are stuck in quite a quandary—do something tactical with little long-term payoff, or do something strategic that provides a long-term payoff but delivers very little, if anything, in the short-term. The issue of deciding between the tactical or strategic solution has had a paralyzing effect on many IBM i decision makers. Simply improving the UI is too short-term, and migrating the application is too long-term. If only there was a rational middle ground.

ASNA's staged migration strategy

ASNA's staged application modernization/migration strategy offers the graceful middle ground you've been looking for. You can use ASNA Wings to quickly get an affordable, easily improved user interface for your RPG programs. Then later, when or if, you need to migrate your RPG application portfolio off of the IBM i, you can do so with ASNA Monarch and reuse 100% of the investment you made with Wings to improve the user interface.

Staged application modernization/migration with ASNA Wings and ASNA Monarch makes it easy for you to get improved user interface results quickly, but also be able to answer to the CFO that your investment in an improved UI is protected should the day come when you need to sunset the IBM i and move your RPG application portfolio to a platform such as .NET.

This document explains the details of ASNA's staged application modernization/migration strategy and what it can do for you.

Introducing ASNA Wings and ASNA Monarch

Before we get into the details of the ASNA staged strategy, let's first get a handle on what ASNA Wings and ASNA Monarch are.

ASNA Wings

ASNA Wings™ is a product for IBM i RPG programs that modernizes traditional 5250 green-screen user interfaces. Wings uses IBM's Open Access for RPG to naturally extend RPG applications to Microsoft's ASP.NET platform. ASNA Wings doesn't require intense upfront analysis or any business logic migration. Only the presentation layer is transformed, so all back-end programming logic and DB access remains intact on the IBM i. Wings creates standard browser pages for the user interface. Wings produces results very quickly with no disruption to the business, and Wings requires very little training. The very same RPG source code can be used to create a program object for Wings display file use and one for the traditional display file use. You won't need two versions of your RPG source when you modernize the UI with Wings.

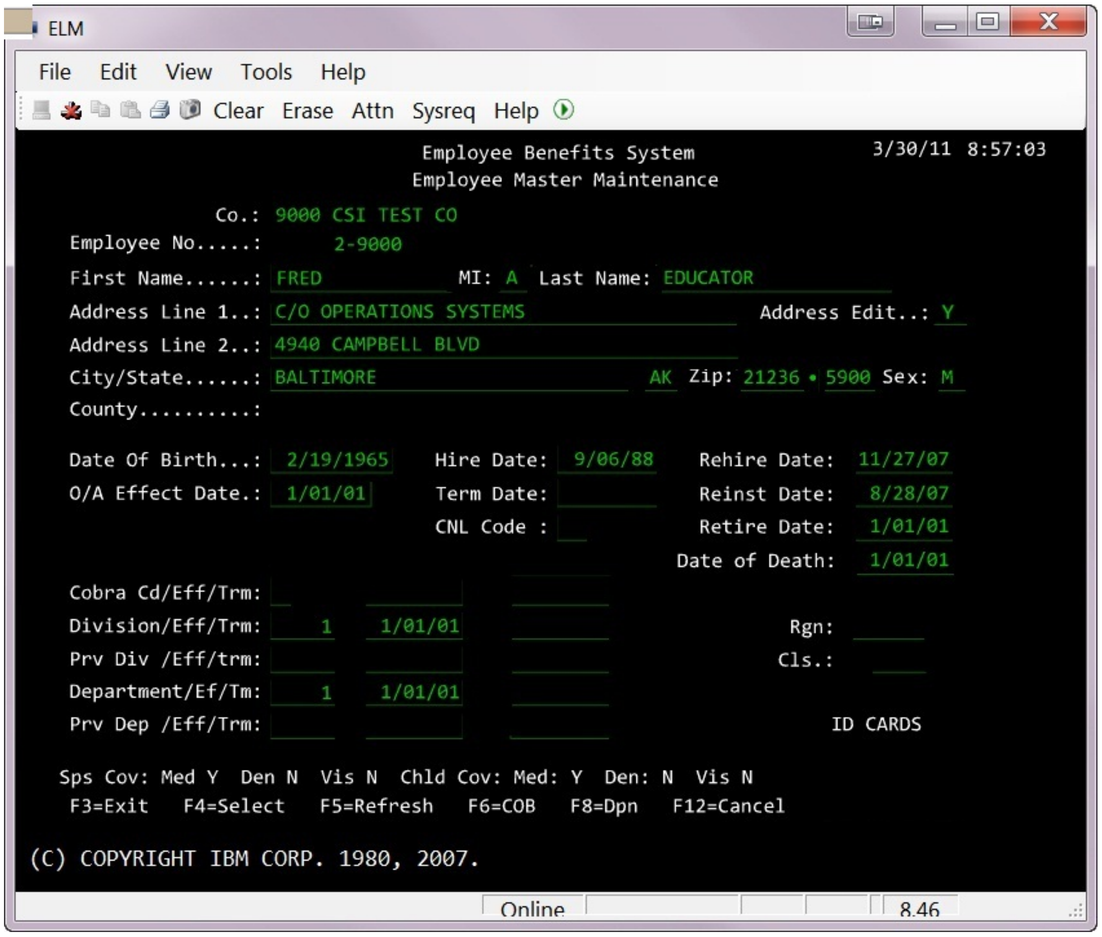
ASNA Monarch

ASNA Monarch™ is a migration suite that migrates RPG (both ILE RPG and RPG/400), DDS, and CL to .NET. Like Wings, Monarch transforms display files to browser pages (in fact, Wings uses Monarch's DDS translation agent to create its browser pages). The migrated application runs in the .NET environment and can connect to the IBM i DB2 database or Microsoft's SQL Server. One of Monarch's components is a powerful RPG application analyzer that examines program dependencies. This analyzer's output provides the needed roadmap to successfully migrate an application. A Monarch migration provides substantially more capabilities for further modernization and customization—but seeing results from a Monarch migration require substantially more time. With Monarch more upfront planning and analysis is required and there are some special skills required when migrating RPG applications to .NET.

To migrate or not

The decision to migrate an RPG application portfolio off of the IBM i is indeed a challenging decision to make. Results are powerful, but are also strategic and long-term. But the introduction of Wings into the mix mitigates the need to make a whole-hog migration decision right away. With ASNA's staged modernization/migration strategy, you can start by modernizing display files first with very minimal effort. Later, as your business needs become more fully defined (and, perhaps, as your confidence with .NET grows as you use Wings), you can step up to Monarch to perform a full migration. The combination of ASNA Wings/Monarch is the only staged modernization/migration available today for the IBM i.

Let's take a quick look at the five stages of modernization/migration available with ASNA Wings and Monarch. To help provide a frame of reference for the various stages, consider the green-screen shown in **Figure**1. This example green-screen will help illustrate some of the stages.



```
                         Employee Benefits System              3/30/11  8:57:03
                        Employee Master Maintenance
                 Co.: 9000 CSI TEST CO
  Employee No.....:        2-9000
  First Name......: FRED            MI: A  Last Name: EDUCATOR
  Address Line 1..: C/O OPERATIONS SYSTEMS                  Address Edit..: Y
  Address Line 2..: 4940 CAMPBELL BLVD
  City/State......: BALTIMORE                         AK  Zip: 21236 • 5900 Sex: M
  County..........:

  Date Of Birth...:  2/19/1965    Hire Date:  9/06/88     Rehire Date:  11/27/07
  O/A Effect Date.:  1/01/01      Term Date:              Reinst Date:   8/28/07
                                  CNL Code :              Retire Date:   1/01/01
                                                        Date of Death:   1/01/01
  Cobra Cd/Eff/Trm:
  Division/Eff/Trm:     1      1/01/01                          Rgn:
  Prv Div /Eff/trm:                                            Cls.:
  Department/Ef/Tm:     1      1/01/01
  Prv Dep /Eff/Trm:                                               ID CARDS

 Sps Cov: Med Y  Den N  Vis N  Chld Cov: Med: Y  Den: N  Vis N
  F3=Exit   F4=Select   F5=Refresh   F6=COB   F8=Dpn   F12=Cancel

(C) COPYRIGHT IBM CORP. 1980, 2007.
```

Figure1. Typical green-screen display.

Stage 1. Display file modernization

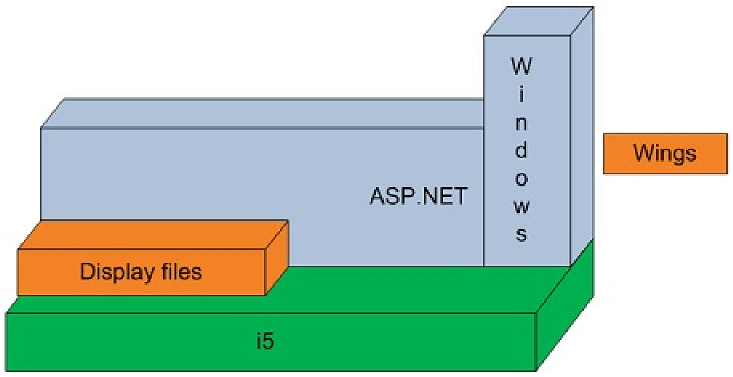


Figure 2. State 1 of ASNA's RPG application modernization strategy.

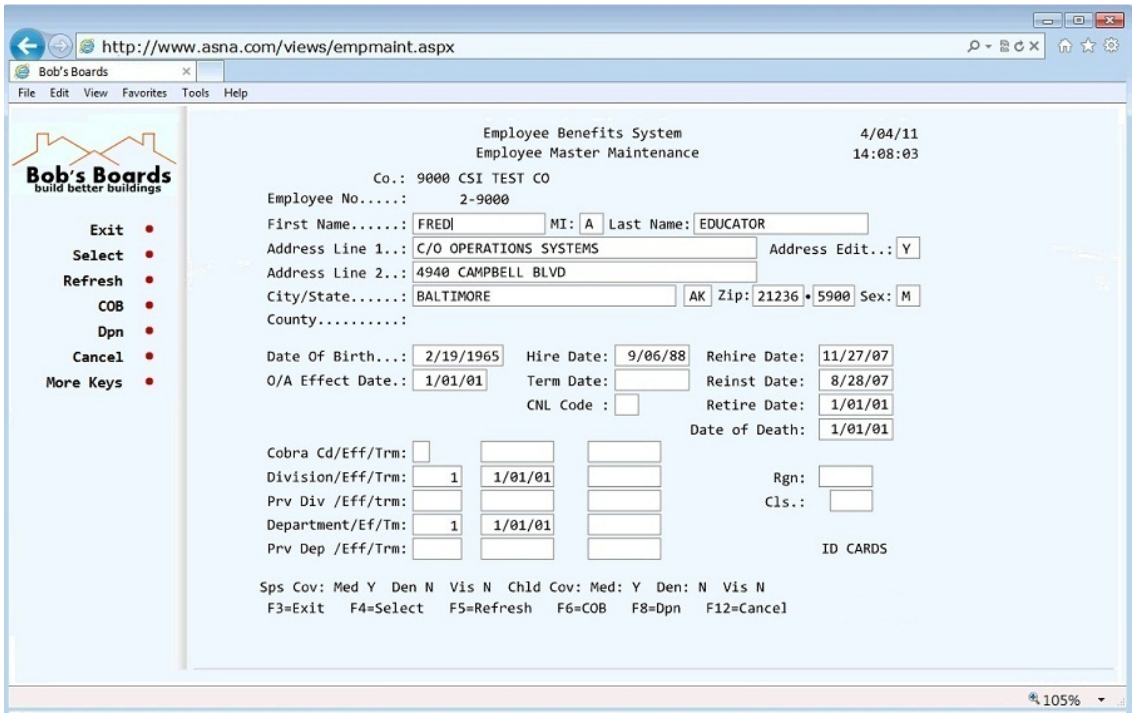
Stage 1 of ASNA's modernization/migration strategy is to use ASNA Wings to modernize your existing green-screen display files (you can select as many or as few as you want initially). This process's workflow is very simple. When you use Wings to modernize your display files, you'll always have the option to continue to use your traditional display files as well. We've found that it's often very important to businesses to have both the modernized, as well as the non-modernized, versions of their RPG applications available—at least until all users are comfortable and ready to use the modernized version.

Because Wings uses ASNA's browser-based emulator, you can modernize as many or as few display files as necessary. With Wings, *all* display files are displayed in the browser. Modernized display files are presented with their new look and feel while any displays not modernized are displayed with the ASNA emulator. Thanks to Wings quick results and the ASNA emulator, Wings provides virtually no disruption to the business.

Wings requires no complex analysis. You simply, from within Visual Studio, select the display files to modernize by point and click. When you modernize a display file with Wings, the original DDS is read from the IBM i and that display file definition is used to generate a browser-based equivalent of that display file in Visual Studio. The implementation of these alternative display files are Microsoft .NET ASPX pages. These pages are hosted on a Windows Web server. Remember, too, that because Wings uses IBM's Open Access for RPG, *all* RPG logic and file IO *remain on the IBM i.*

On the RPG side, you tell the RPG program that it is to use the Wings Open Access "handler" to manage its display file data. This is done by providing a single keyword to the workstation file declaration and recompiling the RPG program. With the Wings handler specified, display file is routed through the Wings Open Access handler rather than through the 5250 workstation controller. For all intents and purposes, the RPG program is oblivious to the fact that it's bypassing the green display files and using the Wings browser-based pages instead.

Figure 3 shows what a default Wings display file modernization looks like. The core part of the display is a pretty faithful rendering of the original green-screen. A few minor features are added automatically, such a pop-up calendar for entering date values but for the most part, the intent of the default display is to get the UI rendered in the browser with as little fuss as possible.

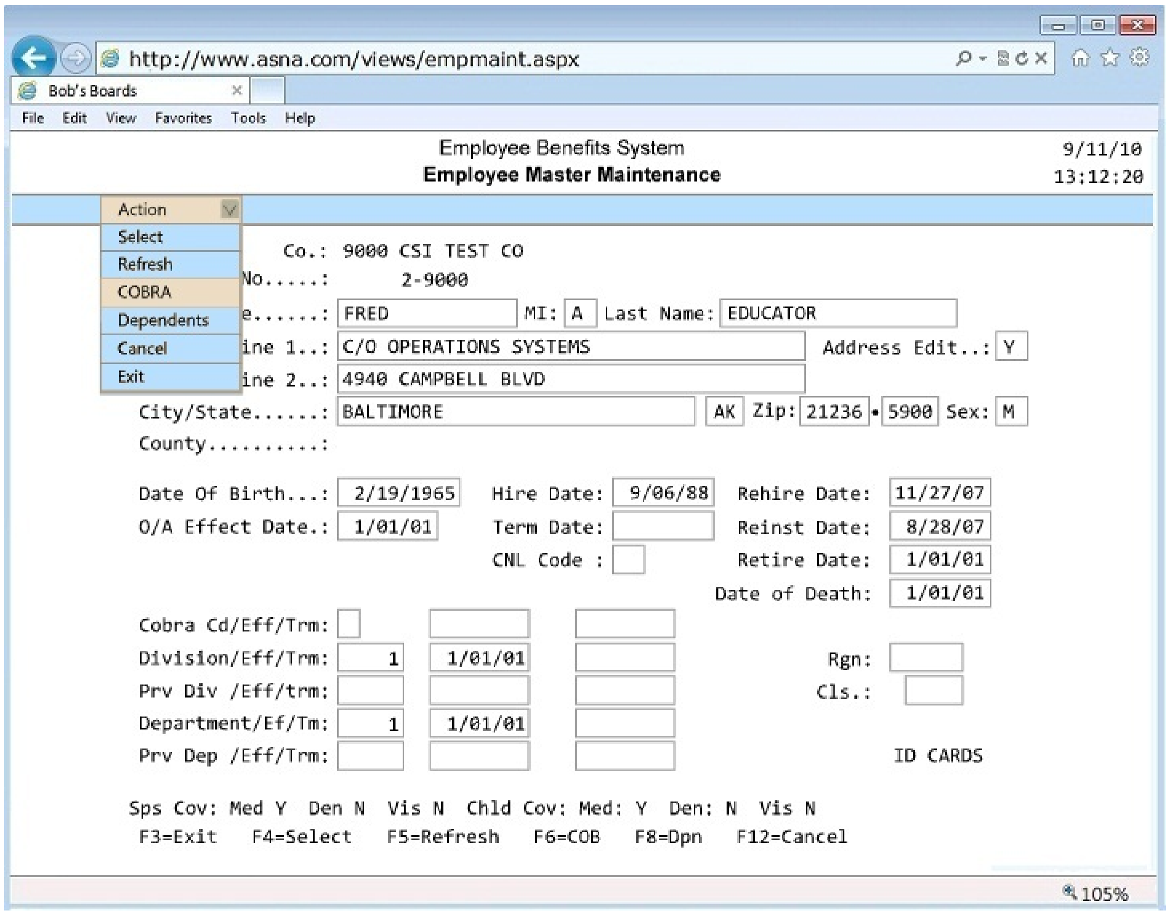


```
                       Employee Benefits System                    4/04/11
                      Employee Master Maintenance                 14:08:03
              Co.: 9000 CSI TEST CO
Employee No.....:        2-9000
First Name......: FRED         MI: A  Last Name: EDUCATOR
Address Line 1..: C/O OPERATIONS SYSTEMS                Address Edit..: Y
Address Line 2..: 4940 CAMPBELL BLVD
City/State......: BALTIMORE                  AK  Zip: 21236 • 5900 Sex: M
County..........:

Date Of Birth...: 2/19/1965   Hire Date:  9/06/88   Rehire Date: 11/27/07
O/A Effect Date.:  1/01/01    Term Date:            Reinst Date:  8/28/07
                              CNL Code :            Retire Date:  1/01/01
                                                  Date of Death:  1/01/01
Cobra Cd/Eff/Trm:
Division/Eff/Trm:     1    1/01/01                         Rgn:
Prv Div /Eff/trm:                                          Cls.:
Department/Ef/Tm:     1    1/01/01
Prv Dep /Eff/Trm:                                                ID CARDS

Sps Cov: Med Y  Den N  Vis N  Chld Cov: Med: Y  Den: N  Vis N
F3=Exit   F4=Select   F5=Refresh   F6=COB   F8=Dpn   F12=Cancel
```

Figure 3. Wings' (or Monarch's) default display of the green-screen from Figure 1.

There are alignment tweaks required from time to time, but for the most part, what you see is what you get. The Windows display shown in Figure 3 is virtually untouched by human hands (this what we often refer as the "row" modernized display). The general look and feel of the page is provided by a templating feature in .NET called "master pages." In this case, the master page used will provide the Bob's Boards image in the upper left-hand corner and dynamically list the currently available function keys in a vertical list on the left—for every display.

The master page can be used to introduce rather dramatic wholesale changes to the look and feel of the application—without needing to touch every page by hand. For example, Figure 4 below shows what the default display would like if you swapped out Figure 3's master page for one that displays a horizontal menu bar.



```
                         Employee Benefits System                              9/11/10
                       Employee Master Maintenance                            13:12:20

                    Co.:  9000 CSI TEST CO
                  No.....:       2-9000
                  e......: FRED              MI: A  Last Name: EDUCATOR
                  ine 1..: C/O OPERATIONS SYSTEMS                   Address Edit..: Y
                  ine 2..: 4940 CAMPBELL BLVD
    City/State......: BALTIMORE                  AK  Zip: 21236 - 5900  Sex: M
    County..........:

    Date Of Birth...:  2/19/1965   Hire Date:  9/06/88   Rehire Date: 11/27/07
    O/A Effect Date.:  1/01/01     Term Date:            Reinst Date:  8/28/07
                                   CNL Code :            Retire Date:  1/01/01
                                                       Date of Death:  1/01/01

    Cobra Cd/Eff/Trm:
    Division/Eff/Trm:      1    1/01/01                           Rgn:
    Prv Div /Eff/trm:                                             Cls.:
    Department/Ef/Tm:      1    1/01/01
    Prv Dep /Eff/Trm:                                               ID CARDS

   Sps Cov: Med Y  Den N  Vis N  Chld Cov: Med: Y  Den: N  Vis N
    F3=Exit   F4=Select   F5=Refresh   F6=COB   F8=Dpn   F12=Cancel
```

Figure 4. Using a different master page to display a horizontal menu bar.

The options shown in the pull down in Figure 4 menu are rendered dynamically from the function options the screen originally offered. If you're happy with the function key wording, you don't have to do anything else to any one page to apply this look and feel to all pages. However, you can also override the words shown in the menu. For example, "COB" was mapped to "COBRA" for the menu action.

Stage 2. Enhance the modernized user experience

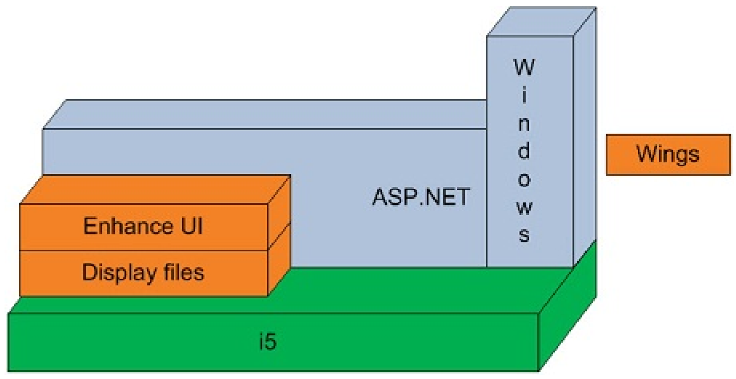
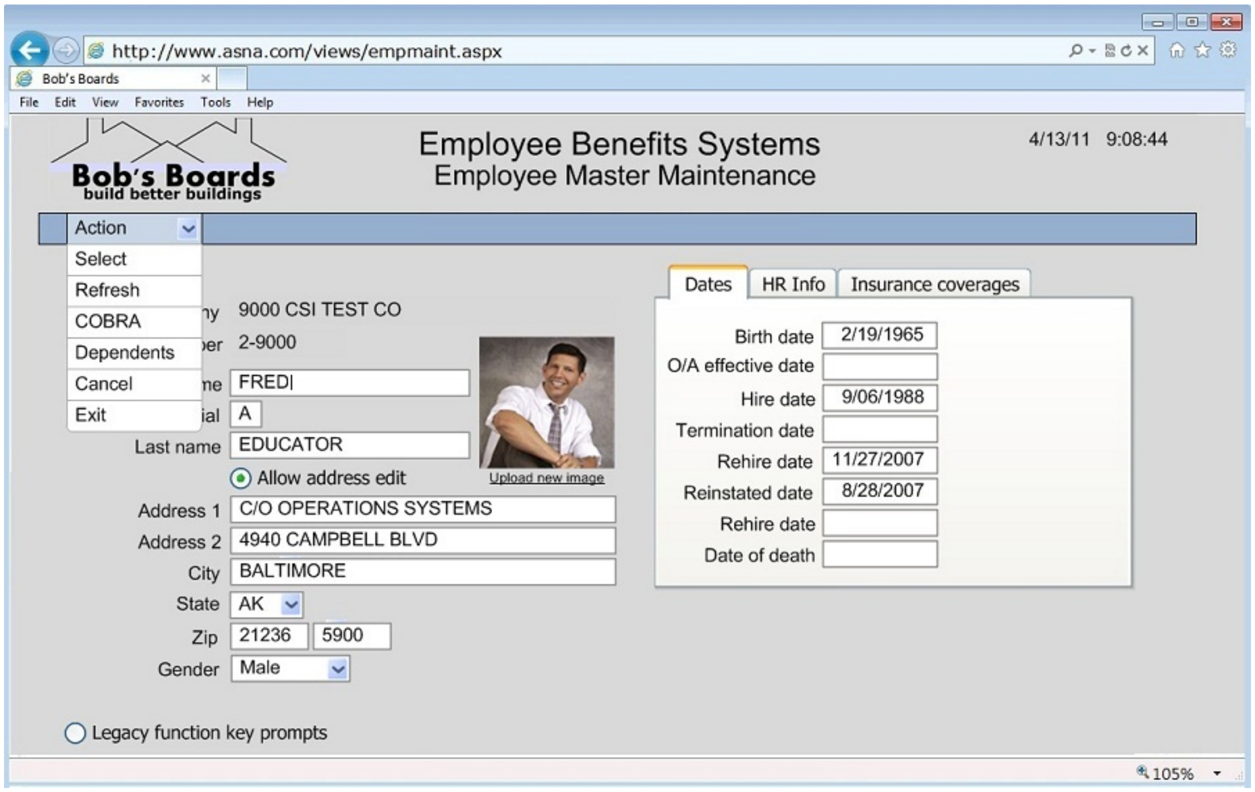


Figure 5. Stage 2 of ASNA's RPG application modernization strategy: customizing and tailoring the modernized pages' look and feel.

The Wings' Stage 1 process is primarily a passive process. That is, you select the display files to modernize and Wings does the rest. With Stage 1 modernization, especially with an attractive master page, the default Wings presentation is very appealing. However, except for the few changes the master page imposes, that presentation is a faithful browser-based version of the original 5250 display file.

To get the most out of Wings display file modernization, you'll probably want more than the basic presentation. Stage 2 is where you do that. Stage 2 represents manual design and coding work done to further improve and enhance the user interface. In Stage 2 you'd use VB.NET, C#, or AVR to add further functionality and user interface enhancements to the Wings display file—at the presentation layer. The original RPG program is rarely, if ever, changed.

Stage 2 may be primarily cosmetic, but the bigger payoff is when you add substantial functional enhancements. Cosmetically, you may swap out the default Wings display file for a third-party grid; functionally you may replace an F4 lookup panel with an Ajax-driven incremental search or integrate Web services with the display. It's important to remember that Stage 2 enhancements are usually done only at the presentation layer, so there are no changes or testing required for the original RPG application.


http://www.asna.com/views/empmaint.aspx

Bob's Boards

File Edit View Favorites Tools Help

Bob's Boards
build better buildings

Employee Benefits Systems
Employee Master Maintenance

4/13/11 9:08:44

Action: Select, Refresh, COBRA, Dependents, Cancel, Exit

ny	9000 CSI TEST CO
ber	2-9000
me	FREDI
ial	A
Last name	EDUCATOR
	Allow address edit
Address 1	C/O OPERATIONS SYSTEMS
Address 2	4940 CAMPBELL BLVD
City	BALTIMORE
State	AK
Zip	21236 5900
Gender	Male

Upload new image

Dates | HR Info | Insurance coverages

Birth date	2/19/1965
O/A effective date	
Hire date	9/06/1988
Termination date	
Rehire date	11/27/2007
Reinstated date	8/28/2007
Rehire date	
Date of death	

Legacy function key prompts

105%

Figure 6. The original page from Figure 3 or Figure 4 with custom modernization done to it.

Figure 6 illustrates the kind of effect you can achieve with Stage 2 enhancements. In this case, several enhancements have been done to the page. First, you'll notice that variation of the master page introduced with Figure 4 is used to provide a horizontal pull down menu. This master page also provides the radio button at the bottom of the page to toggle the display of the original legacy function key prompts (eg, F3=Exit). Many times Stage 2 modernizations are done to make application more accessible to someone who doesn't have a lot of experience using green-screen applications. This type of user is probably more likely to use the pull down menu than to use the function keys. So for them, the legacy function key prompt line is just noise. But, to placate experienced green-screen users, that prompt line is available if desired.

You'll also notice that sections of the original screen have been segregated into their own tabs (as represented by the "Dates," "HR Info", and "Insurance coverages" tabs).

Also notice how what was an input field to allow address changes has been changed to a radio button.

In addition to these mostly cosmetic changes (although it could be argued that segregating the data into tabs offers some functional enhancement), the primary functional change added to the application is the addition of the ability display the employee’s picture. In this case, a convention is used to locate that picture’s image file. For example, the employee’s number is used to build the file name (eg, the employee image file name for Figure 6 would be “2-9000.jpg”). Conventions like these allow images to be displayed without the requirement for adding an “image file name” field in the employee master file. There is also an “upload new image” link under the image for image maintenance. And, just as one more reminder, this sort of application change happens in the presentation layer—the RPG program has no knowledge of its display’s newfound ability to upload a new image.

The ability to perform Stage 2 UI enhancements is one of the things that substantially differentiates Wings from screen scrapers. With screen scrapers, you have only the unadorned 5250 data stream with which to customize the display. With Wings, you get named fields, and indicators, for all of the data sent from the RPG program to the display. Having access to this semantically-defined display data allows a nearly unlimited set of possibilities for enhancing the cosmetic appeal and functional capabilities of the Wings display.

Remember that Stage 1 and Stage 2 changes are wholly contained within the Wings environment. Thus, the original RPG application, and its file IO continue to perform as it always has—without regard for its effective new Wings user interface.

While some Web developer skills are required for some Stage 2 enhancements, you don’t have to be a Web development expert to add very powerful UI improvements with Stage 2. Because Wings generates standard .NET ASPX browser pages, you can use virtually any third-party ASPX control or custom Web development technique (employing things such as CSS, JavaScript/jQuery, and perhaps Ajax) to improve the

UI. Wings doesn't constrain the potential UI enhancements you can achieve with a vendor-provided proprietary set of UI widgets. With Wings, your imagination is the only limit to the ways you can provide your users a better UI. Google maps, graph controls, consuming business partner Web services, tab controls—those kinds of UI improvements are all easily incorporated with ASNA Wings.

Stage 3. Migrate interactive RPG programs to .NET

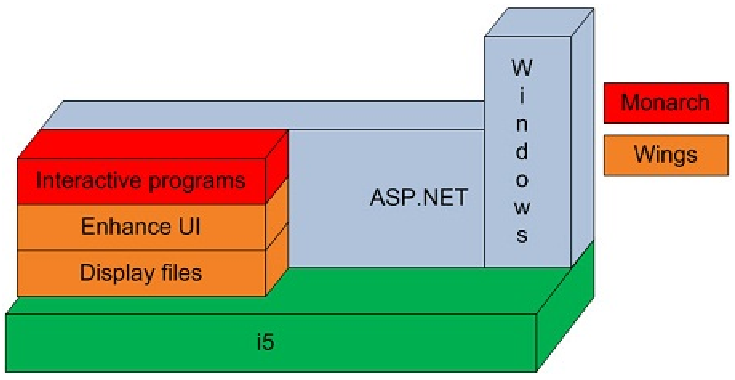


Figure 7. Stage 3 of ASNA's RPG application modernization strategy: Migrate interactive RPG programs to .NET

Several vendors offer some form of Stage 1 and Stage 2 UI enhancement. One of the problems, though, with our competitor's offerings is that their product's Stage 1 and Stage 2 enhancements are dead-ends. That is, those enhancements are very IBM i dependent and can't be reused in any other application context.

After having performed Stage 2, whether you proceed with Stages 3 and higher depends on your long application modernization strategy. While Stage 1 and Stage 2 modernizations are certainly strategic, they are low-cost and low effort, high-payoff enhancements that require few special skills. Moving to Stage 3, though, requires long-term commitment to the migration process, a long-term commitment to the technologies required, and usually efforts by both the ASNA Services team for the actual code migration and efforts by your team for testing and deployment.

Most of the time, a fork in the road that directs you to Stage 3 is based on the decision you make as to whether you are going to keep the IBM i platform for the foreseeable future. If you have a strategy (either short-term or long-term) to migrate off the IBM i platform, you are probably interested in Stages 3 and higher. However, you don’t necessarily have to be moving off of the IBM i for at least Stages 3 and 4 to have merit. If you simply need your RPG programs to do more than they can currently do (interface effectively with business partner Web services, for example), Stages 3 and 4 may indeed be for you as well.

In Stages 1 and 2, all program logic and file I/O remains on the IBM i—it's just the presentation layer that is hosted in Windows. Stage 2 provides the ability to add cosmetic and functional display enhancements—while requiring no changes to the original RPG application (and remember that with Stages 1 and 2, all RPG logic and file IO remain on the IBM i). For many shops, this capability meets all modernization requirements and they need go no further with the ASNA modernization/migration stages.

However, there may come a time when your interactive programs need changes that aren't rational to make in the green-screen RPG environment. For example, you may have the need for program logic itself (and not just the presentation layer) to make a Web service call, or you may need to add the ability for the interactive program to communicate concurrently with both the IBM i's DB2 and with Microsoft's SQL Server.

For these cases, with Stage 3 you use ASNA Monarch to migrate interactive RPG programs to .NET. The target language for this conversion can be either ASNA Visual RPG (AVR) or Microsoft C#. Once migrated, the program logic is executed on the .NET Web server. This migrated code can then be further enhanced and modified by .NET programmers with all of the programming capabilities available in the.NET Framework. The work that you've done in Stages 1 and 2 to modernize and enhance the UI continue to persist in this stage. All of your investment in improving the presentation layer is preserved in Stage 3. If you need to migrate an interactive program for which you haven't yet modernized its display files that can also be done in Stage 3. The ability to perform this staged modernization/migration is unique to ASNA's modernization/migration offerings. No other vendor can offer the non-disruptive modernization/migration that the Wings/Monarch combination offers.

For IBM i-based businesses that have made the decision to migrate their RPG application portfolio to .NET, Stage 3 may also be considered the entry point in .NET migration. These businesses can jump directly into Stage 3 to start a full application migration. Although Wings modernization projects upgrade nicely to Monarch migration projects, you don't have to start with Wings to migrate your applications. If you know that migrating your RPG business logic and file IO is your endgame, you can start directly at Stage 3. Note that Wings (because of IBM Open Access imposed constraint) requires ILE RPG (or RPG 400 programs converted with CVTRPGSRC). ASNA Monarch imposes no such constraint—it can migrate either RPG III (RPG/400) or ILE RPG (RPG IV).

For IBM i-based business that don't know if they'll ever need to migrate their RPG application portfolio to Wings, think of Stage 3 and higher simply as asset protection. You might not ever want to migrate your RPG to .NET with Monarch, but if you ever do, the presentation layer work you've done with Wings is preserved and useable by Stages 3 and higher.

Stage 4. Migrate RPG batch programs to .NET.

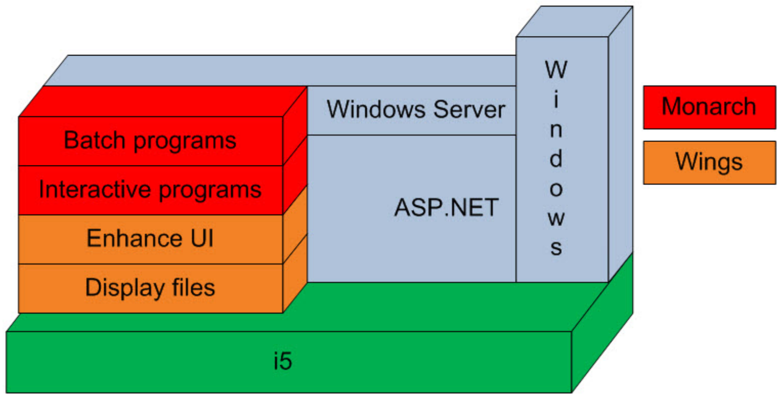


Figure 8. Stage 4 of ASNA’s RPG application modernization strategy: migrate RPG

batch programs to .NET

In Stage 3, RPG interactive programs are migrated to .NET. This means that RPG logic and file IO is no longer executed as i5 program objects, but rather as .NET assemblies. Stage 4 is general extension to Stage 3—where your batch programs are also migrated to the .NET environment. This may include printing processes (where Monarch provides Windows-based print file alternatives to your RPG print files, for example) and job workflow processes.

ASNA Monarch provides the investigative tools needed to identify interactive and batch programs, as well as program and display file dependencies. ASNA Monarch's rich discovery tools provide the long-term road map needed to effectively and successfully migrate your entire RPG application to .NET.

Stage 5. Migrate DB2 database to SQL Server

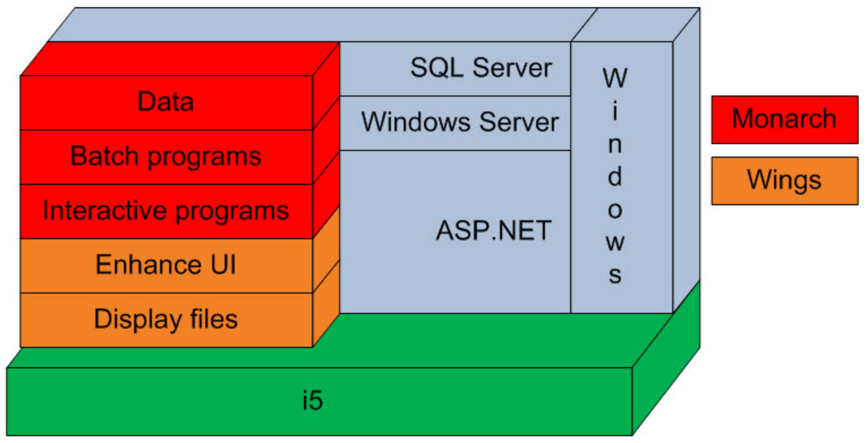


Figure 9. Stage 5 of ASNA's RPG application modernization strategy: migrating from DB2 to SQL Server.

Stage 5 applies specifically to those IBM i-based businesses who are on the strategic path to sunset their IBM i platform and migrate their application portfolio to .NET.

Monarch-migrated programs connect to the underlying database through ASNA DataGate. ASNA DataGate can transparently connect to either the IBM i DB2 database or Microsoft SQL Server. In the case of using Microsoft SQL Server, the RPG file level IO operations are translated in the DataGate layer to SQL operations. From a coding perspective, the migrated code continues to use record level access idioms (that is, record level operations such as CHAIN, SETLL, and READ persist in the .NET code). This is achieved either by using AVR as the migration target language or, when C# is used as the target language, with Monarch-provided C# library that provides C# versions of RPG file IO idioms.

The dual connectivity offered by DataGate minimizes and nicely compartmentalizes the disruption of migrating an RPG application portfolio off of the IBM i platform. The migration efforts can first be focused on migrating the applications, saving the database conversion for the last step. Once the apps are working, the work on the database conversion can begin.

Once Stage 5 is completed, what was once an IBM i-bound RPG application will have been granted a new life as an AVR or C# application residing in the .NET environment and using SQL Server as its underlying database.

ASNA staged modernization/migration summary

Not every business needs every stage of the possible ASNA staged modernization/migration process. The point of discussing them here with Wings is to ensure that Wings customers understand that their investment in display file modernization is protected should they later find the need to sunset the IBM i and migrate their RPG application portfolio to .NET.

ASNA staged modernization/migration summary

You can use ASNA Wings and ASNA Monarch to achieve a well-staged and incremental application modernization for your RPG application portfolio. Using Wings for Stages 1 and 2, you get very fast, high-quality results. Depending on your application modernization strategy, you may or may not need Stages 3 and higher. However, if your needs today don’t include migrating off the IBM i platform, it’s good to know that those stages are available should the need ever arise.

No other vendor offers the non-disruptive modernization/migration options and capabilities that ASNA Wings and ASNA Monarch offer. And, not only are our products the best, but when you select ASNA, you’re selecting a company with a 30-year track history of making customers happy!

Your RPG applications no longer need to be bound to a sad, dreary, 5250 green-screen user interface. With ASNA you can give your RPG applications the happy user interface they deserve!