Tools and Solutions for Modernizing Your IBM i Applications

Discover application modernization tools

Create mobile, web, and client solutions

Manage security using SkyView Policy Minder

Tim Rowe

Redpaper

ibm.com/redbooks

International Technical Support Organization

Tools and Solutions for Modernizing Your IBM i Applications

September 2014

REDP-5095-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (September 2014)

This edition applies to IBM i 6.1 and IBM i 7.1 and later.

This document was created or updated on May 31, 2017.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper™ publication is a companion to the *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185 IBM Redbooks® publication. It describes independent software vendor (ISV) and Business Partner tools that can be used to modernize your IBM i applications. It includes the following types of tools:

- Mobile, web, and client solutions
- Database modernization tools
- Security
- Tools for understanding and modernizing RPG and COBOL

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

Tim Rowe is the Business Architect for Application Development and Systems Management for IBM i. He is responsible for ensuring that the IBM i operating system has the features, function, languages, and tools IBM i developers require to run, maintain, and write modern applications to run business. He has worked in the web and IBM i middleware space for the past decade. He regularly speaks at conferences to help customers understand how IBM i is a modern platform for running the most modern applications. He has been a part of the IBM i family for over 20 years, and has worked on many different layers of the operating system.

Thanks to the following companies for their contributions to this project:

Adsero Optima
Arcad
ASNA
BCD Software
CNX Corporation
Fresche Legacy
IBS
LANSA
looksoftware
ProfoundLogic
Resolution Software
Rocket Software
SkyView Partners
Surround Technologies
SystemObjects

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  https://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Mobile, web, and client solutions

This chapter introduces mobile, web, and client solutions. It includes the following sections:

- ASNA
- looksoftware
- BCD Software
- CNX Corporation: Valence Web Application Framework for IBM i
- ProfoundLogic
- Rocket Software
- SystemObjects
- Lansa
- IBS

1.1 ASNA

ASNA offers Mobile RPG to enable mobile capabilities and Wings to provide RPG application display file modernization.

1.1.1 ASNA Mobile RPG brings mobile computing capabilities to RPG programmers

Consumer demand drove the first wave of mobile use into record-setting levels. For example, with 1.5 million device activations per day, it is projected that a billion Android devices will be activated by 2014. During this first wave, smartphones and tablets have been elevated from specialty, luxury use to everyday commodity use. Just as these devices have become commonplace to consumers, they are also rapidly becoming commonplace in the enterprise.[1]

Businesses are quickly finding that mobile devices not only add user convenience, but more importantly, that their use enables substantial workflow improvements that save money and time. In this day of narrow margins and demanding competitive conditions, mobile computing in the enterprise is no longer seen as just a convenience. It is seen as necessary to maintain a competitive business position.

Today's savvy IBM i decision makers are able to articulate specific use cases and business advantages that mobile devices can offer them. The challenge for most IBM shops is how to acquire the software that is needed to power IBM i mobile applications.

By using ASNA Mobile RPG, IBM i *programming teams can use their existing RPG programming skills* to build the next generation of IBM i applications that target mobile devices.

[1] 1. Mobile device distribution and usage
http://www.androidpit.com/schmidt-android-activations-1-5-million-a-day-1-billion-by-2014
http://www.ubergizmo.com/2013/07/google-sees-1-5-million-android-device-activations-daily/

You can use Mobile RPG to create smartphone and tablet apps with nothing but ILE RPG. Figure 1-1 shows an example application.

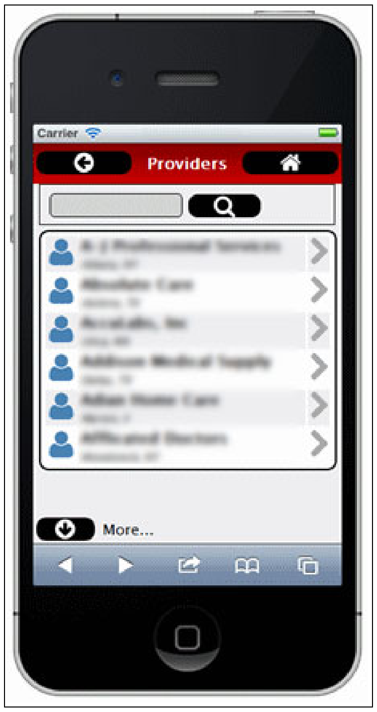


Figure 1-1 Example application

Contents

ASNA Mobile RPG works by intercepting the workstation file output from an RPG program and redirecting it to a mobile device display file. The RPG program does 100% of the work (that is, logic, validation, file I/O, and so on). In fact, its logic is unaware that its workstation file output is intercepted. As far as the RPG program is concerned, it is reading and writing to a traditional display file.

The Mobile RPG enabler for this workstation file interception is the IBM Rational® Open Access for RPG Edition API. Open Access for RPG Edition captures the workstation file data at runtime and redirects it to the ASNA Mobile RPG display file (an example is shown in Figure 1-1). This mobile display file is created with the Mobile RPG drag-and-drop mobile display file designer. The Mobile RPG mobile display file does all the work. The RPG programmer does not have to learn HTML, JavaScript, CSS, Objective C, Java, or any other traditional mobile technologies. ASNA Mobile RPG can make any RPG programmer a mobile application developer in a matter of days.

Did you know?

IBM Rational Open Access for RPG Edition is packaged with V6R1 and V7R1. Although Open Access for RPG Edition was originally a billable feature, that is no longer the case. There is no additional charge to use Open Access on your IBM i system. To compile an RPG program, a programmer who is using Open Access for RPG Edition requires only a licensed RPG compiler (as is required for compiling any other RPG program).

Open Access for RPG Edition has the capacity to intercept any file data from an RPG program at a low level. However, Open Access for RPG Edition is only an enabler. To offer a complete solution, Open Access for RPG Edition needs to hand off RPG file data to an Open Access handler. Open Access for RPG Edition handlers transform intercepted data into something useful. Mobile RPG is packaged with its own Open Access for RPG Edition handler. This handler then passes workstation file data, including hidden fields and all indicator values, to the Mobile RPG mobile display file. IBM Open Access for RPG Edition provides a powerful, reliable, and performant way to gracefully extend RPG's abilities.

Business value

Although many IBM i businesses need mobile IBM i applications, they are often challenged with exactly how to acquire them. Creating mobile applications the conventional way requires learning several languages and technologies. It is not uncommon for a programming team to need to spend months learning technologies such as JavaScript, CSS, HTML5, Objective C, and Java to create great mobile applications.

With the advent of ASNA Mobile RPG, this is no longer the case. Mobile RPG empowers your existing team to create superb mobile applications, targeting smartphones and tablets using nothing but their RPG programming skills. No PC-based code needs to be written for a Mobile RPG application: 100% of the application is written with ILE RPG. No obscure ILE libraries are required and there are not any constraints on your RPG. You can use embedded SQL, record level access, fixed format, free format, or anything else you can think of. You can use Mobile RPG to write RPG your way.

Because learning Mobile RPG is so quick, RPG coders can be productive with Mobile RPG in a matter of days.

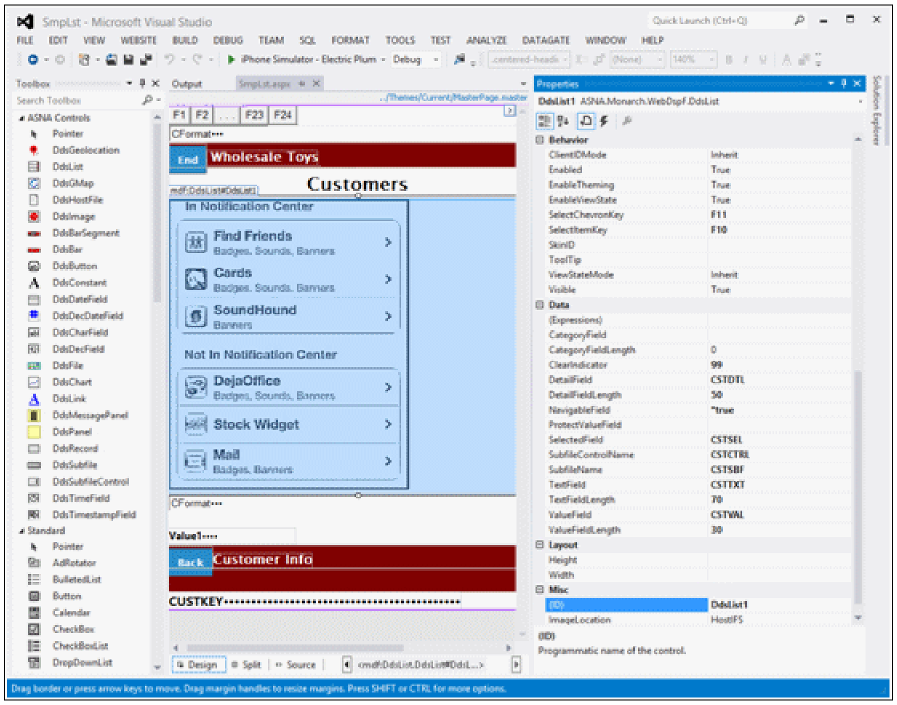
Solution overview

ASNA Mobile RPG uses a three-step workflow to create a mobile application. This next section demonstrates this workflow by creating a list in a smartphone application. This section provide an overview of the process. A short video on creating an ASNA Mobile RPG application is available for viewing on the ASNA website.

Step 1. Create the mobile display file

Figure 1-2 on page 5 shows the design-time view of creating an ASNA Mobile RPG display file. On the left are the ASNA Controls (user interface elements) that are available for creating the user interface (UI). There are controls for elements such as lists, maps, charts, images, buttons, input/output fields, and navigational bars. Figure 1-2 on page 5 shows that a DdsList control is placed on the form. On the left, you can see the list control properties such as ClearIndicator (99), the SubfileControlName, and SubfileName (CSTCTRL and CSTSBF), the TextField (CSTTXT), the DetailField (CSTDTL), and the ValueField (CSTVAL).

No code is written here. All that is necessary is to assign values (as described earlier) to the properties.



DdsList1 ASNA.Monarch.WebDspf.DdsList

Behavior	
ClientIDMode	Inherit
Enabled	True
EnableTheming	True
EnableViewState	True
SelectChevronKey	F11
SelectItemKey	F10
SkinID	
ToolTip	
ViewStateMode	Inherit
Visible	True
Data	
(Expressions)	
CategoryField	
CategoryFieldLength	0
ClearIndicator	99
DetailField	CSTDTL
DetailFieldLength	50
NavigableField	*true
ProtectValueField	
SelectedField	CSTSEL
SubfileControlName	CSTCTRL
SubfileName	CSTSBF
TextField	CSTTXT
TextFieldLength	70
ValueField	CSTVAL
ValueFieldLength	30
Layout	
Height	
Width	
Misc	
(ID)	DdsList1
ImageLocation	HostIFS

(ID)
Programmatic name of the control.

Drag border or press arrow keys to move. Drag margin handles to resize margins. Press SHIFT or CTRL for more options.

Figure 1-2 ASNA Mobile RPG mobile display file design-time experience

Step 2. Export the mobile display file to an IBM i display file

The Mobile RPG display file designer is then used to export the mobile display file to the IBM i as a traditional display file. This traditional display file is used as a target display file against which an ILE RPG program is compiled.

The list example in Figure 1-3 shows a fragment of the DDS that is generated by the Mobile RPG export process. The export process also compiles this DDS into a display file object. Viewing the DDS is not generally something you do in the course of using Mobile RPG. However, seeing it is helpful for understanding how things are working. You can then write an ILE RPG program that references this traditional display file.

Remember, this display file is only used to compile the RPG program. At runtime, Open Access redirects the RPG program's WORKSTN file output to the Mobile RPG display file.

```
A          R RCSTINFO
A                                    OVERLAY
A            CFORMAT       10A  I  1  2
A            CUSTKEY       50A  O  1 26
A          R CSTSBF                  SFL
A            CSTSEL         1A  H
A            CSTTXT        50A  O  1  5
A            CSTVAL        30A  H
A            CSTDTL        50A  O  3  1
A          R CSTCTRL                 SFLCTL(CSTSBF)
A                                    SFLSIZ(2)
```

Figure 1-3 DDS that is generated from ASNA Mobile RPG mobile display file

Write an RPG program that compiles with the exported display file

The RPG programmer then writes an RPG program to populate that subfile. As far as the RPG programmer and the RPG program are concerned, the traditional display file that is created with the Mobile RPG export process is the target output for the subfile data. However, at runtime, the program's workstation file data is redirected, through Open Access, to the Mobile RPG display file. The subroutine to populate this file is shown in Example 1-1.

Example 1-1 The ILE RPG written to populate the ASNA Mobile RPG DdsList control

```
* Load customer list.
C     LOADCSTLST    BegSr
C                   Eval      *IN99 = *On
C                   Write     CSTCTRL
*
C                   Eval      CSTRRN = 0
C                   Eval      ROWCOUNT = 0
*
C     ROWCOUNT      DowLT     16
C                   Read      CUSTMASTL2
C                   If        NOT %EOF
C                   Eval      ROWCOUNT = ROWCOUNT + 1
C                   Eval      CSTRRN = CSTRRN + 1
C                   Eval      CSTSEL = '0'
C                   Eval      CSTTXT = CMNAME
C                   Eval      CSTDTL = %TRIM(CMCITY) + ', ' + CMSTATE
C                   Eval      CSTVAL = %CHAR(CMCUSTNO)
C                   Write     CSTSBF
C                   Else
C                   Leave
C                   EndIf
C                   EndDo
C
C                   Eval      *IN99 = *Off
C                   Write     CSTCTRL
C                   EndSr
```

Solution architecture

As previously covered, ASNA Mobile RPG uses Open Access as its enabler. For a more detailed description of how Open Access works with Mobile RPG, consider how a traditional application works without Open Access. As (the highly abstracted) Figure 1-4 shows, when the RPG program writes a WORKSTN record, that data is passed on to the workstation control and then emitted as a 5250 data stream to a 5250 device.

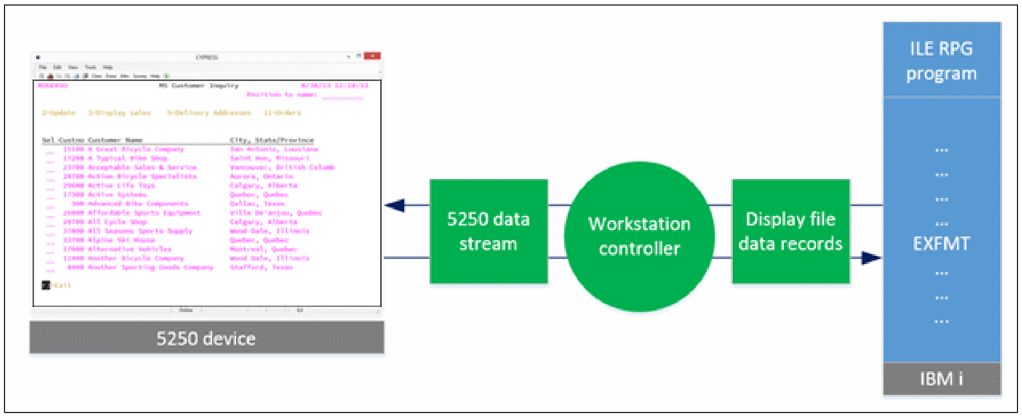


Figure 1-4 WORKSTN file data flow with traditional RPG program without Open Access

With Open Access and Mobile RPG, the RPG program still writes a traditional WORKSTN record. However, when the RPG program has the Mobile RPG Open Access (OA) handler registered with it (by using the WORKSTN file's HANDLER continuation line), the Open Access API passes the WORKSTN data to the Mobile RPG OA handler. That handler emits an ASNA display file data set. This data includes all the fields that are written to the WORKSTN record, in addition to hidden fields, special fields, and all indicator values. This data structure is then passed to the Mobile RPG presentation layer where it is rendered by the Mobile RPG display file. The complete process is shown in Figure 1-5.

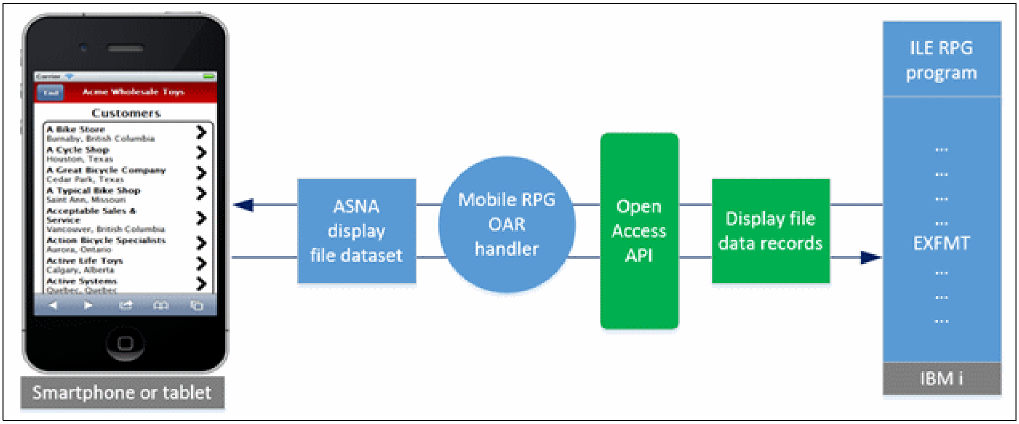


Figure 1-5 WORKSTN file data flow with ASNA Mobile RPG and IBM Open Access

Mobile interface element controls

Mobile RPG provides a set of user interface elements with which to build your mobile user interface. These controls include elements such as maps, charts, images, input fields, labels, buttons, lists, navigation bars, and record formats. These controls have their HTML, JavaScript, and CSS predefined. They are all controlled through RPG fields and indicators. The RPG programmer does not need to learn HTML, JavaScript, or CSS.

These controls are also aware of facilities on the mobile device. So tasks such as initiating a phone call, sending a text message, opening a browser, scanning a bar code, taking a picture, and recognizing the GEO location of the phone are all built into Mobile RPG.

Recall that a plain RPG program populates the Mobile RPG user interface elements. Therefore, each of these elements must be able to be surfaced to the RPG program through a traditional display file. Many of Mobile RPG display elements have a one-to-one relationship with corresponding elements typically defined with DDS. For example, labels and input fields map directly to DDS input fields and output constants. However, several of the Mobile RPG controls do not have a direct analog in DDS. In this situation, the RPG subfile is used.

The power of the subfile for Mobile RPG

Several of the Mobile RPG user interface elements are surfaced to the RPG program, through the exported display file, as a simple RPG subfile. Consider a simple example. Say that you need to map the route between the San Antonio Airport and the ASNA offices in San Antonio.

After dragging and dropping the DddsGMap control on the Mobile RPG mobile display file, you set the property values listed in Table 1-1.

Table 1-1 Property values

Property name	Value	Description
Address Field	Location	Subfile field name
AddressFieldLength	50	Subfile field name length
ClearIndicator	99	Clear subfile when this indicator is on
SubfileControlName	MapCtrl	Subfile controller name
SubfileName	MapSbf	Subfile name

Then, the Mobile RPG display file is exported to the IBM i system. The basic ILE RPG to render the map is shown in Example 1-2 in free format. Populating multiple map points is probably done by reading data from a file, but the file and that loop are omitted for simplicity.

Example 1-2 ILE RPG to map the route for two addresses

```
FHELLODSPF CF   E             WORKSTN Handler('MOBILERPG')
F                                     SFile(MAPSBF:MAPRRN)

D MAPRRN          S              4P 0

 /free
  *IN99 = *ON;
  Write MAPCTRL;

  MAPRRN = 1;
  LOCATION = '14210 Northbook, San Antonio, TX';
  Write MAPSBF;

  MAPRRN = 2;
  LOCATION = 'San Antonio Airport, San Antonio, TX';
  Write MAPSBF;

  *IN99 = *Off;
  Write MAPCTRL;
```

```
ExFmt HomeMenu;
Select;
    When *In03;
    *INLR = *ON;
    Return;
EndSl;
```

Assuming that you know RPG, the code is self-explanatory. You can see that the graph control is surfaced to the underlying RPG program as a simple subfile, and that simple code is used to populate the subfile. You do not need to worry about folding subfiles or other subfile minutiae with Mobile RPG. Its subfile use is simple.

At runtime, the RPG program runs as though the traditional display file was being used, but Open Access and Mobile RPG OA handler redirect the WORKSTN file data to Mobile RPG display file.

As you can see, the subfile makes populating the Mobile RPG DdsGMap control simple. Mobile RPG employs a similar process to populate lists and charts (such as horizontal and vertical bar and pie charts).

ASNA DataGate

ASNA DataGate, packaged with Mobile RPG, is an ASNA IBM i host server product that transports the WORKSTN data between the IBM i and the Mobile RPG mobile display file. ASNA DataGate is secure (it obeys all IBM i imposed security and authorities) and performant. Before using a Mobile RPG app, a user must sign in to the app with a valid IBM i user profile and password.

Usage scenarios

Mobile RPG is intended for you to use to provide mobile computing, which is driven by your IBM i system and its enterprise data, to your employees, business partners, and customers.

Supported platforms

The ASNA Mobile RPG smartphone support currently includes Apple iOS version 5 and version 6; Android OS Gingerbread, Ice Cream Sandwich, and Jellybean; and Windows 8 smartphones. Its tablet support includes Apple iPad 2 and the iPad Mini, Samsung Player 4, Asus Transformer; Google Nexus 7; and Microsoft Surface RT and Pro. The latest comprehensive list of device support is available on the ASNA website at:

http://devnet.asna.com/downloads/Pages/ASNAMobileRPG61.aspx

Related information

For more information about ASNA Mobile RPG, see the following link:

http://asna.com/us/products/mobile-rpg/

For ASNA Mobile RPG installation notes and device support, see:

http://devnet.asna.com/downloads/Pages/ASNAMobileRPG61.aspx

For an introductory video to ASNA Mobile RPG, see:

http://asna.com/us/products/mobile-rpg/videos/

For an ASNA Mobile RPG FAQ, see:

http://asna.com/us/products/mobile-rpg/faq/

For the Mobile Computing and the IBM i white paper, see:

http://asna.com/us/2012/mobile-computing-and-the-ibm-i/

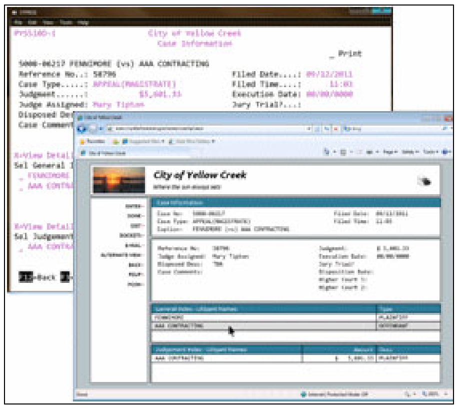
1.1.2 ASNA Wings enables RPG application display file modernization

IBM i customers struggle with the limits and constraints of the IBM i native character-based user interface. For many businesses, the limits of the “green-screen” user interface are annoying and frustrating. In addition to that frustration, some IBM i ISVs decided that character-based applications are not competitive.

The challenge of the limits and stigma of the IBM i 5250 character-based user interface led to many screen scrapers in the 1990s. Although many shops attempted to use these products, these products imposed their own constraints. Many of these products are quick fixes, not permanent solutions to the real problem.

In 2012, IBM announced Rational Open Access RPG Edition. Open Access for RPG Edition is an API, but it is an important API. Open Access for RPG Edition allows you to intercept ILE RPG file data at a previously unobtainable, low level. This API is a superb solution to the limitations of the IBM i character-based user interface.

ASNA Wings harnesses the power of the IBM Rational Open Access API to provide superb, browser-based alternatives to the traditional green-screen user interface. Because Wings' browser-based displays are not constrained by the 5250 data stream, you can create effective functional and cosmetic enhancements to these displays. The result is a modern user experience for your RPG applications as shown in Figure 1-6.



```
                          City of Yellow Creek
                           Case Information
                                                          _ Print
 5000-06217 FENNIMORE (vs) AAA CONTRACTING
 Reference No..: 58796                    Filed Date....: 09/12/2011
 Case Type.....: APPEAL(MAGISTRATE)       Filed Time....:      11:03
 Judgment......:          $5,601.31       Execution Date: 00/00/0000
 Judge Assigned: Mary Tipton              Jury Trial?...:
```

Figure 1-6 ASNA Wings provides a modern, extensible alternative IBM i user interface

ASNA Wings works by intercepting the WORKSTN file output from an RPG program and redirecting it to a browser-based alternative display. This alternative display is created automatically by Wings from the character-based display's DDS specifications. All DDS rules (for example, position cursor and conditional attributes such as input inhibited) are obeyed by the Wings display. The RPG program does all of the work (logic, validation, file I/O, and so on). In fact, its logic is unaware that its workstation file output has been intercepted. As far as the RPG program is concerned, it is reading and writing to a traditional display file.

The ASNA Wings enabler for this workstation file interception is the IBM Rational Open Access for RPG Edition API. Open Access for RPG Edition captures the workstation file data and, at runtime, redirects it to the ASNA Wings RPG display file (the foreground of Figure 1-6 on page 10). After Wings generates its alternative browser-based display, you can easy add functional enhancements (such as data export to Excel or calling business partner web services) and cosmetic improvements such as images and styling.

ASNA Wings includes a built-in, browser-based 5250 emulator. This emulator provides a great way for you to selectively and incrementally improve a group of display files. Any display files that are not yet modernized by Wings are displayed in a nondisruptive fashion in the Wings emulator. Wings' displays are also fully compatible with Apple and Android tablets. This compatibility provides users with a great portable workstation.

Did you know?

IBM Rational Open Access for RPG Edition is packaged with V6R1 and V7R1. Although Open Access for RPG Edition was originally a billable feature, it is now available at no additional fee on your IBM i. Compiling an RPG program with Open Access for RPG Edition requires only a licensed RPG compiler (the same as for compiling any other RPG program).

Open Access for RPG Edition has the capacity to intercept any file data from an RPG program at a low level. However, Open Access for RPG Edition is only an enabler. To offer a complete solution, Open Access for RPG Edition needs to hand off RPG file data to an Open Access handler. Open Access for RPG Edition handlers transform intercepted data into something useful. ASNA Wings is packaged with its own Open Access for RPG Edition handler. This handler then passes workstation file data, including hidden fields and all indicator values, to the Wings browser-based display file.

Business value

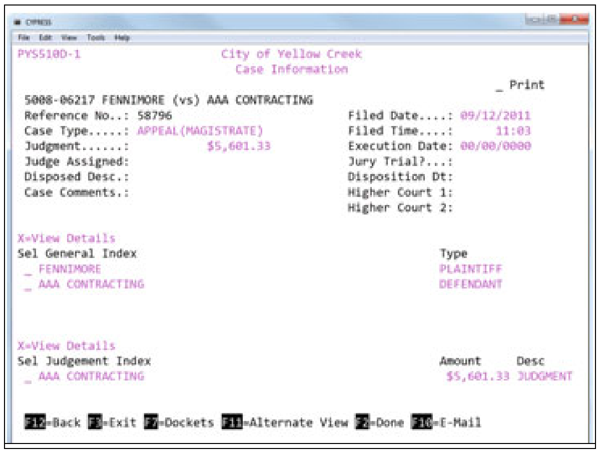
Many IBM i businesses have a keystone RPG application that they have been using for years (often decades). This application has often been customized and enhanced over the years to enable the business to deliver its unique value to its customers. Without this keystone application, many IBM i businesses might be dramatically impaired. However, although this application does many things that the business needs, because of its character-based dependency, the application cannot generate a modern interface that the business needs. In most cases, that application was written for 20- or 30-year-old specifications.

In evaluating their frustration with their keystone applications, shops might consider rewriting or replacing their application with a prewritten package. Keystone applications often have hundreds of thousands (or often millions) of lines of code, so rewriting it is too expensive and too time consuming. A prewritten package might offer an application rewrite alternative, but a prewritten package often is not able to offer the unique software features the business needs to maintain its competitive advantage. Tweaking the prewritten package to provide those features can be prohibitively expensive. The reality for many IBM i shops is that, despite frustrations and limitations, their keystone applications are hard to replace.

These issues make effective IBM i application modernization the perfect answer to the challenge. IBM Open Access provides a more effective, more pleasing user interface for keystone applications. Open Access for RPG Edition has made application modernization a strategic, long-term solution. With ASNA Wings and Open Access, you can modernize your keystone application in a stepwise, rational fashion. By doing so, you are laying the groundwork for further application growth and improvement later.

Solution overview

ASNA Wings provides a Windows based design aid that is used to import and modernize traditional display files. You can import a few display files or use the Wings FlightPlan facility, which allows you to import many (perhaps hundreds) of display files in a single operation. See Figure 1-7.

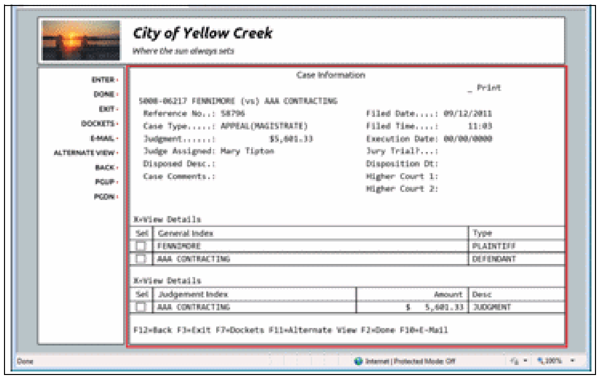


```
PYS510D-1                    City of Yellow Creek
                               Case Information
                                                              _ Print
 5008-06217 FENNIMORE (vs) AAA CONTRACTING
 Reference No..: 58796                       Filed Date....: 09/12/2011
 Case Type.....: APPEAL(MAGISTRATE)          Filed Time....:      11:03
 Judgment......:          $5,601.33          Execution Date: 00/00/0000
 Judge Assigned:                             Jury Trial?...:
 Disposed Desc.:                             Disposition Dt:
 Case Comments.:                             Higher Court 1:
                                             Higher Court 2:

X=View Details
Sel General Index                                         Type
 _ FENNIMORE                                              PLAINTIFF
 _ AAA CONTRACTING                                        DEFENDANT


X=View Details
Sel Judgement Index                                       Amount     Desc
 _ AAA CONTRACTING                                         $5,601.33 JUDGMENT

 F12=Back F3=Exit F7=Dockets F11=Alternate View F2=Done F10=E-Mail
```

Figure 1-7   Traditional “green-screen” display

Figure 1-8 shows the “raw” rendering of the display in Figure 1-7 after it is imported with ASNA Wings. The area in the red outline is the part of the display that is automatically generated from the display file's DDS by ASNA Wings. The rest of the page, the header, the sidebar, and the footer are customizable (including colors, images, and layout) and easily repeatable across all of the display files you import.

The part of the display in the red outline is the part that provides fidelity with the original display. This part is generated automatically by Wings. When rendered, all aspects of the original display are provided. All of the function keys are features such as message files, folding subfiles, and conditional indicator-driven attributes are available. See Figure 1-8.



```
                         Case Information
                                                          _ Print
5008-06217 FENNIMORE (vs) AAA CONTRACTING
 Reference No..: 58796                     Filed Date....: 09/12/2011
 Case Type.....: APPEAL(MAGISTRATE)        Filed Time....:      11:03
 Judgment......:          $5,601.33        Execution Date: 00/00/0000
 Judge Assigned: Mary Tipton               Jury Trial?...:
 Disposed Desc.:                           Disposition Dt:
 Case Comments.:                           Higher Court 1:
                                           Higher Court 2:
X=View Details
```

Sel	General Index	Type
☐	FENNIMORE	PLAINTIFF
☐	AAA CONTRACTING	DEFENDANT

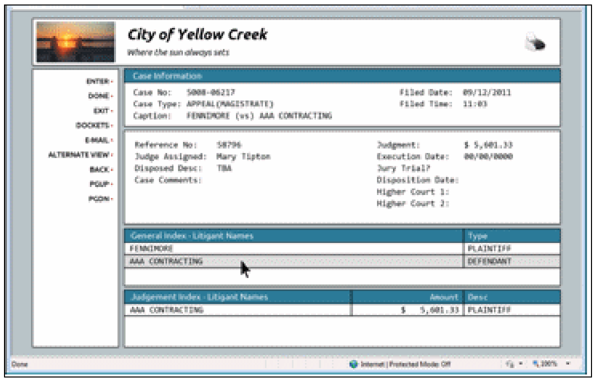
X=View Details

Sel	Judgement Index	Amount	Desc
☐	AAA CONTRACTING	$ 5,601.33	JUDGMENT

```
F12=Back F3=Exit F7=Dockets F11=Alternate View F2=Done F10=E-Mail
```

Figure 1-8   Display that is rendered with ASNA Wings

The display in Figure 1-9 shows how the Wings raw display can look with a few enhancements. In this case, some areas of the window are more clearly delineated by adding some CSS style sheet rules. The page is made a little more mouse friendly by setting a few properties of the Wings subfile control. Wings allows you to easily identify one of a subfile row's values as the subfile's default action when a row is clicked. This identification makes the display even more mouse friendly and easier for newer users.



Case Information

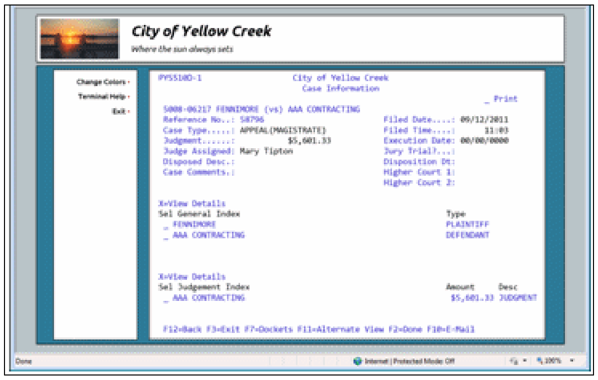
```
Case No:    5008-06217                          Filed Date:  09/12/2011
Case Type:  APPEAL(MAGISTRATE)                  Filed Time:  11:03
Caption:    FENNIMORE (vs) AAA CONTRACTING

Reference No:     58796                Judgment:          $ 5,601.33
Judge Assigned:   Mary Tipton          Execution Date:    00/00/0000
Disposed Desc:    TBA                  Jury Trial?
Case Comments:                         Disposition Date:
                                       Higher Court 1:
                                       Higher Court 2:
```

General Index - Litigant Names	Type
FENNIMORE	PLAINTIFF
AAA CONTRACTING	DEFENDANT

Judgement Index - Litigant Names	Amount	Desc
AAA CONTRACTING	$ 5,601.33	PLAINTIFF

Figure 1-9 The previous display rendered with ASNA Wings with a few enhancements.

Figure 1-10 shows how the display in Figure 1-7 on page 12 looks when it is rendered with the Wings built-in browser-based 5250 emulator. As you can see, the same general appearance that the Wings imported displays use is also used by the 5250 emulator. This reuse minimizes the work for the user to move from a Wings modernized display to the emulator.



```
PYS510D-1                  City of Yellow Creek
                             Case Information
                                                              _ Print
 5008-06217 FENNIMORE (vs) AAA CONTRACTING
 Reference No..: 58796                      Filed Date.....: 09/12/2011
 Case Type.....: APPEAL(MAGISTRATE)         Filed Time.....:      11:03
 Judgment......:           $5,601.33        Execution Date: 00/00/0000
 Judge Assigned: Mary Tipton                Jury Trial?...:
 Disposed Desc.:                            Disposition Dt:
 Case Comments.:                            Higher Court 1:
                                            Higher Court 2:

X=View Details
Sel General Index                                      Type
 _ FENNIMORE                                           PLAINTIFF
 _ AAA CONTRACTING                                     DEFENDANT


X=View Details
Sel Judgement Index                                    Amount     Desc
 _ AAA CONTRACTING                                      $5,601.33 JUDGMENT

 F12=Back F3=Exit F7=Dockets F11=Alternate View F2=Done F10=E-Mail
```

Figure 1-10 Original rendered with the Wings browser-based emulator

The Wings emulator is important because it helps you incrementally import and enhance display files in an incremental fashion. It is also important because you might have some programs for which the displays cannot be imported. Recall that IBM Open Access works only with ILE RPG programs. If you have a CL program with a display file, an old COBOL program, or even use WRKSPLF in the course of a user workflow, the Wings emulator can gracefully take over and display those programs in a nondisruptive fashion. For IBM RPG/400® programs, you can easily use the IBM i **`CVTRPGSRC`** command to transform your RPG/400 to ILE RPG so those programs can be modernized with Wings.

Alternative user experience

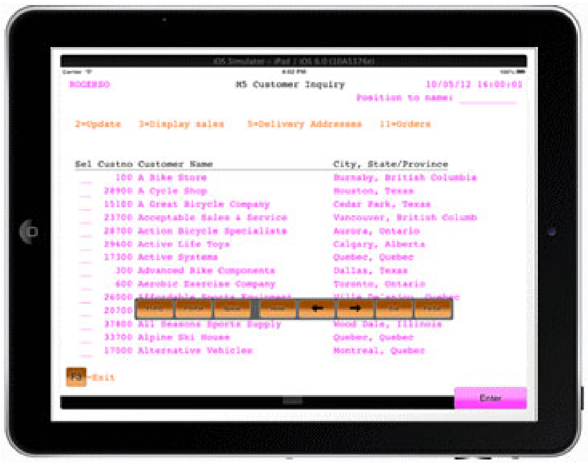
Wings supports a user workflow, managed by the IBM i library list, to provide your users with access to either the modernized displays or the traditional 5250 displays. This is often important in those cases when you want a class of users to continue to use the 5250 emulator (manufacturing workers on a shop floor, for example).

Portable workstations

Many IBM i businesses today need mobile solutions. In many cases, they need custom mobile solutions that perform focused, specific tasks that are tailored to run on smartphones and other pocket computers. For that need, ASNA provides ASNA Mobile RPG:

http://asna.com/us/products/mobile-rpg/

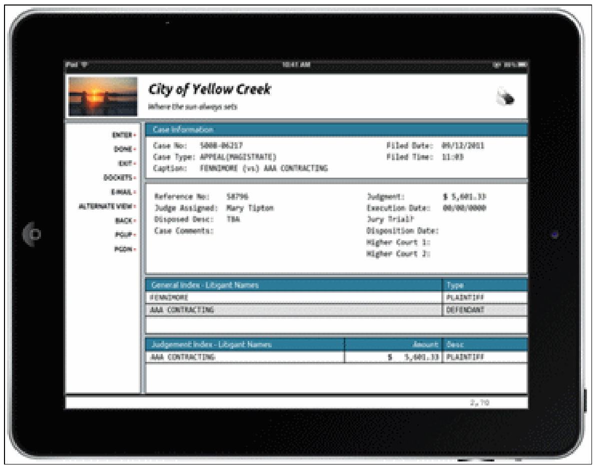
However, there is also a class of mobile that is called portable mobile in this document. In this case, a business has a group of users for whom the organization needs to provide quick, mobile access to existing IBM i applications. These applications are traditional 24 x 80 or 27 x 132 applications and smartphones are often poor devices for display applications with this form factor. Users often do not tolerate the finger pinching and sliding to try to move the salient part of the window they need to see into view on a smartphone's limited viewport. However, modern tablets, both 7" and 10", make great devices for providing portable access to existing IBM i applications. Use cases for this type of computing include medical professionals in a hospital; managers who are working on an assembly line; and mobile realtors who need to update back-office information in real time. See Figure 1-11.



```
ROCERSO                    M5 Customer Inquiry               10/05/12 16:00:01
                                                     Position to name: ________

 2=Update   3=Display sales   5=Delivery Addresses   11=Orders

 Sel Custno Customer Name                       City, State/Province
 __     100 A Bike Store                        Burnaby, British Columbia
 __   28900 A Cycle Shop                        Houston, Texas
 __   15100 A Great Bicycle Company             Cedar Park, Texas
 __   23700 Acceptable Sales & Service          Vancouver, British Columb
 __   28700 Action Bicycle Specialists          Aurora, Ontario
 __   29600 Active Life Toys                    Calgary, Alberta
 __   17300 Active Systems                      Quebec, Quebec
 __     300 Advanced Bike Components            Dallas, Texas
 __     600 Aerobic Exercise Company            Toronto, Ontario
 __   26000 Affordable Sports Equipment         Ville Sn'aulcu, Quebec
 __   20700
 __   37800 All Seasons Sports Supply           Wood Dale, Illinois
 __   33700 Alpine Ski House                    Quebec, Quebec
 __   17000 Alternative Vehicles                Montreal, Quebec

 F3=Exit
```

Figure 1-11 Wings emulator at work on an iPad

These applications can be displayed with Wings using the emulator as shown in Figure 1-11, or with a Wings enhanced display as shown in Figure 1-12. In either case, the displays are touch-sensitive, render well, and do not require pinching and sliding to bring parts of the display into view.



City of Yellow Creek

Where the sun always sets

Case Information

Case No: 5008-06217 | Filed Date: 09/12/2011
Case Type: APPEAL(MAGISTRATE) | Filed Time: 11:03
Caption: FENNIMORE (vs) AAA CONTRACTING

Reference No: 58796 | Judgment: $ 5,601.33
Judge Assigned: Mary Tipton | Execution Date: 00/00/0000
Disposed Desc: TBA | Jury Trial?
Case Comments: | Disposition Date:
Higher Court 1:
Higher Court 2:

General Index - Litigant Names	Type
FENNIMORE	PLAINTIFF
AAA CONTRACTING	DEFENDANT

Judgement Index - Litigant Names	Amount	Desc
AAA CONTRACTING	$ 5,601.33	PLAINTIFF

Figure 1-12 A Wings modernized display on an iPad

Solution architecture

As previously covered, ASNA Wings uses IBM Open Access as its enabler. Consider how Open Access works with ASNA Wings. First, consider how a traditional application works without Open Access. As (the highly abstracted) Figure 1-13 shows, when an RPG program writes a WORKSTN record, that data is passed on to the workstation control and then emitted as a 5250 data stream to a 5250 device.

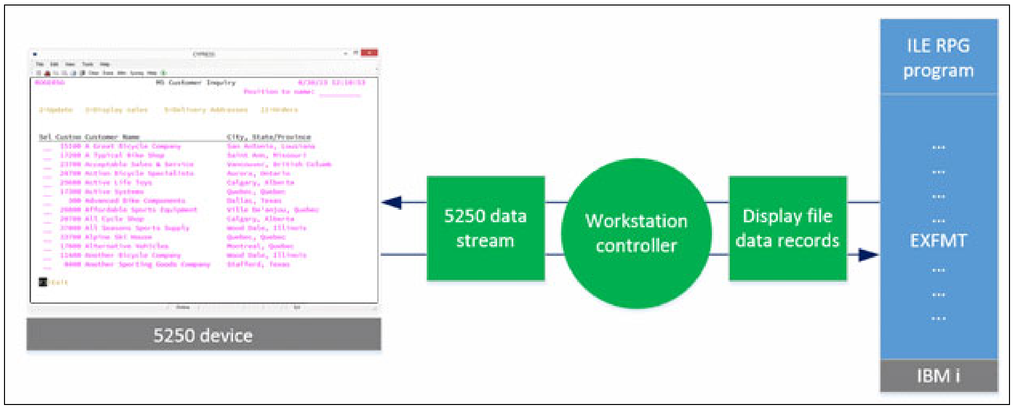


Figure 1-13 WORKSTN file data flow for a traditional RPG program without Open Access

With Open Access and ASNA Wings, the RPG program still writes a traditional WORKSTN record (Figure 1-14). The only modification the RPG program needs is to have the RPG “Handler” keyword added as a continuation to the WORKSTN file declaration. When this handler is present, the Open Access API passes the WORKSTN data to the Wings RPG OA handler, which emits an ASNA display file data set. This data includes all the fields that are written to the WORKSTN record, in addition to hidden fields, special fields, and all indicator values. This data structure is then passed to the ASNA Wings presentation layer, where it is rendered to the user in a browser.

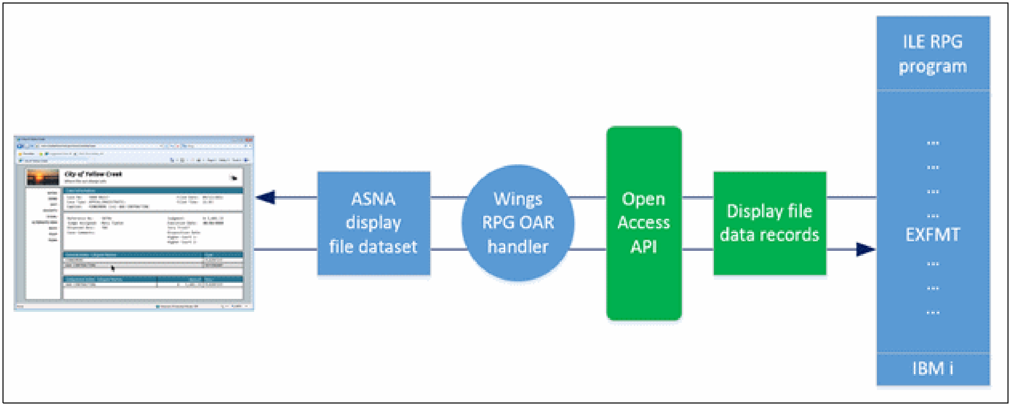


Figure 1-14 WORKSTN file data flow with ASNA Mobile RPG and IBM Open Access

Unlike screen scrapers, Wings eliminates the 5250 data stream. Wings passes a semantic buffer of data to the presentation layer-where the record format fields are available by field name. All indicators and hidden fields are also available in this data structure. This data structure is handy for functionally extending the presentation layer. For example, you can add web service calls in the presentation layer by using these fields, and the underlying RPG itself has no knowledge of its presentation layer being changed.

The Wings presentation layer has a high level of fidelity with display file rules as traditionally defined by DDS. For example, Mobile RPG's input fields have a PositionCursor attribute, the value of which is an indicator. Thus, if that indicator is on in the ASNA display file data set, the cursor is positioned at that field.

ASNA DataGate

ASNA DataGate, packaged with ASNA Wings, is an ASNA IBM i host server product that transports the WORKSTN data between the IBM i and the Mobile RPG mobile display file. ASNA DataGate is secure (it obeys all IBM i imposed security and authorities) and performant. Before using an ASNA Wings application, a user must sign in to the application with a valid IBM i user profile and password.

Usage scenarios

ASNA Wings is intended for you to use to provide modernized displays for your existing IBM i applications. It provides a browser-based rendering, and no other software than a browser needs to be installed on your users' computers. ASNA Wings also works on 7" and 10" Apple and Android tablets to provide portable workstations for your RPG applications.

Supported platforms

ASNA Wings works with modern browsers, including Chrome, IE (7 and up), Firefox, and Safari.

ASNA Wings requires IBM i5/OS™ V6R1 or V7R1.

Related information

For more information about ASNA Wings, see:

http://asna.com/us/products/wings/

1.2 looksoftware

looksoftware openlook helps you develop new applications or enable existing RPG ILE applications for RPG OA. The newlook software is an IDE that helps you create rich interfaces and take advantage of RPG OA.

1.2.1 openlook

The openlook software enables you to:

1. Easily and quickly develop new, modern graphical RPG applications for rich desktop, the web, and mobile devices, using IBM RPG Open Access (RPG OA)
2. Enable existing RPG ILE applications for RPG OA
3. Seamlessly switch between RPG OA enabled and non-OA (refaced 5250) screens for maximum flexibility
4. Use Open Display Files to remove the limitations of DDS

This section covers developing modern RPG applications with openlook, which is coupled with newlook for design and layout of forms

openlook: An overview

openlook is a technology that enables RPG developers to use their existing skills and source code. Coupled with newlook, it makes designing and developing state-of-the-art modern RPG applications a highly productive experience. Developing with openlook provides three core benefits to developers:

- Simplicity. Through existing RPG developer skills and newlook's advanced intuitive designer, developers can build modern applications with minimal training. The experience of developing is intuitive, and common layout tasks are automated.
- Productivity. Everything in openlook is tightly integrated and geared towards finishing projects more quickly and easily. You can develop for rich clients, mobile devices, and browsers, using familiar RPG.
- Satisfaction. newlook's designer is easy, self-evident, and fun to work with. Many of the core tasks that a developer requires to do are either automated or just one mouse click away.

RPG Open Access and openlook

Despite being over 50 years old, RPG is still by far the most popular development language on IBM i. It is estimated that over 80% of all development on IBM i today is conducted with RPG. It continues to be enhanced and improved by IBM, and is highly lauded within the IBM i community as a proven foundation for mission-critical business applications.

This achievement is amazing when you consider the massive technological advancements that have taken place in the last 20 years. Today's businesses and their users are increasingly dependent on technology to do tasks, with the Cloud and mobile devices now adopted in everyday lives.

That ubiquity is why RPG Open Access and openlook are so important. As 5250 becomes less and less able to support the demands of users needs, RPG evolves to handle today's requirements.

Here are some of the core functions that today's users and businesses demand:

- Rich graphical applications with efficient navigation and easy access to information
- Desktop integration with applications such as Microsoft Excel, Word, Outlook, and SharePoint
- The ability for casual users, remote personnel, and customers to access data securely from the web
- The ability to access applications from mobile devices such as smartphones and tablets
- Integrating with devices to handle tasks like image integration, signature capture, bar code scanning, and GPS mapping.

Traditional RPG uses DDS to define a 24x80 or 27x132 5250 window and then uses a Workstation as shown in Figure 1-15.

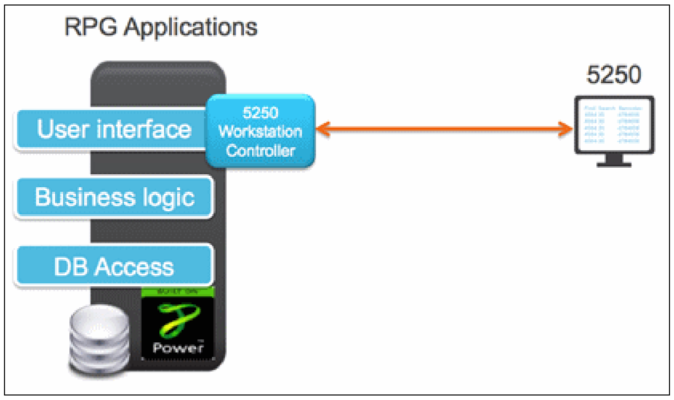


Figure 1-15 Dependency on 5250 workstation controller

RPG Open Access removes the dependency on 5250 and replaces it with an open architecture as shown in Figure 1-16.

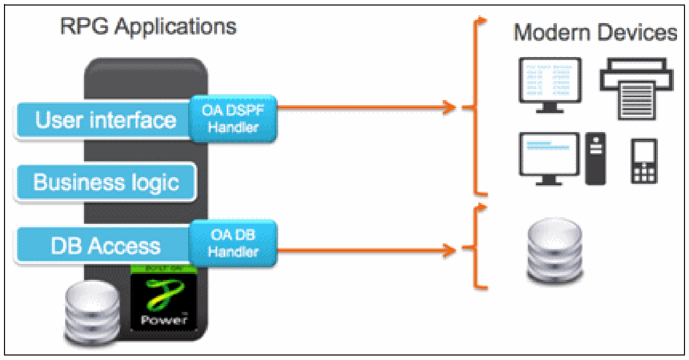


Figure 1-16 Dependency on 5250 is removed

Because there are so many possibilities with Open Access, IBM collaborated with vendors, including looksoftware, to devise handlers for specific tasks. Because looksoftware has the skills, technologies and experience for supporting modern graphical interfaces, including rich desktop, web, and mobile, its focus is enabling RPG developers to have the best possible experience with implementation for those interfaces.

looksoftware's openlook solution includes a single handler as shown in Figure 1-17.

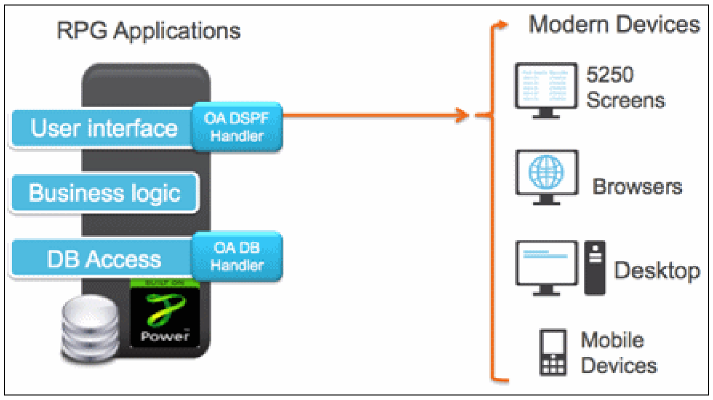


Figure 1-17 Single handler

openlook is designed to be seamless, efficient, and productive so that developers can focus on developing in RPG and easy layout of graphical forms. It offers the power and flexibility to support today's and tomorrow's interfaces and devices.

openlook shields developers from technical “plumbing” so that they can focus on building and delivering applications as productively as possible (Figure 1-18).

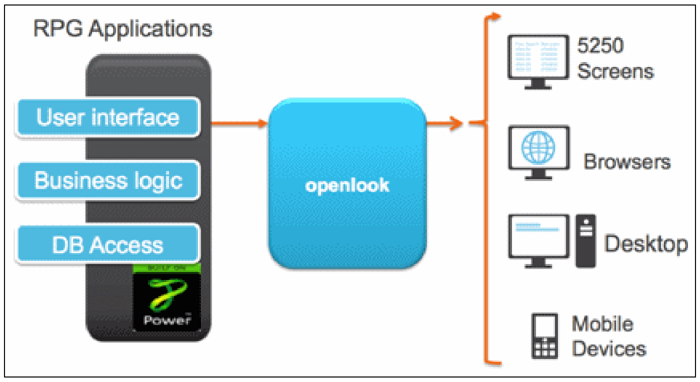


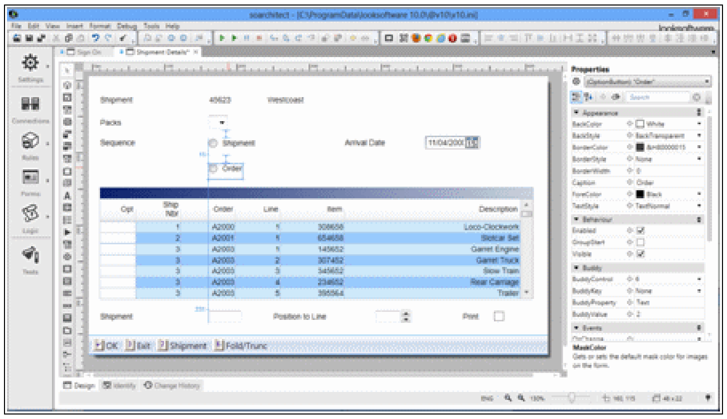
Figure 1-18 openlook shields developers from technical “plumbing”

Designing forms with newlook

When designing graphical interfaces such as web, mobile and rich desktop, have a design environment that can make the process as easy and flexible as possible. newlook is the leading design environment for IBM i that supports multiple interfaces with a single effort.

Design and layout of graphical forms is an intuitive and straightforward experience, especially if you are already familiar with a modern graphical integrated development environment (IDE). However, you must have clear objectives of what you need to deliver before starting. You need to set up the environment that works best with your existing development processes.

The newlook IDE has a familiar interface for developers with modern graphical IDE experience, and it is easy to learn for newcomers. See Figure 1-19.



Shipment 45623 Westcoast

Packs

Sequence ○ Shipment Arrival Date 11/04/200

○ Order

Opt	Ship Nbr	Order	Line	Item	Description
	1	A2000	1	308658	Loco-Clockwork
	2	A2001	1	654658	Slotcar Set
	3	A2003	1	145652	Garret Engine
	3	A2003	2	307452	Garret Truck
	3	A2003	3	345652	Slow Train
	3	A2003	4	234652	Rear Carriage
	3	A2003	5	395564	Trailer

Shipment Position to Line Print

OK Exit Shipment Fold/Trunc

Figure 1-19 The newlook IDE

When you start the newlook IDE, you either need to create a solution or open an existing one. The solution contains the entire newlook project within a single folder. This configuration simplifies management and maintenance of the modernization project. newlook allows for many configurations when you are working in a team with multiple developers.

Through automated smart-guides, rulers, and enhanced toolbars, newlook anticipates what a developer is working on and helps with the process of designing forms. Further additions such as the hints system make suggestions to help guide a developer intuitively through the design and layout process.

newlook's designer toolbar provides access to the layout of forms as shown in Figure 1-20.

Figure 1-20 The newlook designer toolbar

Developing with openlook

There are times when you cannot Open Access an existing 5250 window or application. Perhaps it is a package where you do not have the RPG source code, or it is an OS window. Perhaps it was developed in a much older version of RPG or even COBOL. If that is the case, newlook can reface those screens by using its dynamic recognition engine.

openlook has been designed to seamlessly switch between OA and non-OA enabled screens for a holistic and flexible approach to modernization.

openlook is part of a holistic solution that can seamlessly switch between OA and non-OA as shown in Figure 1-21.

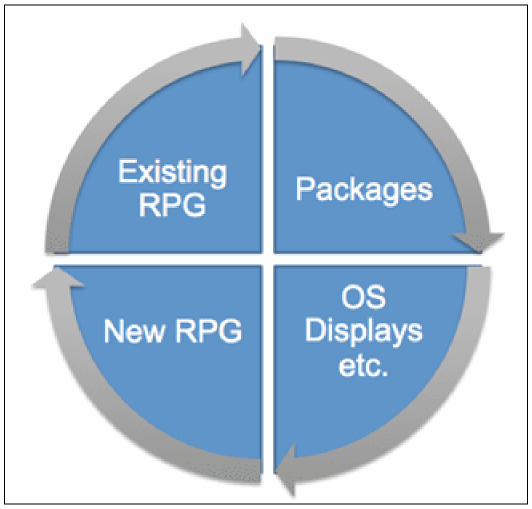
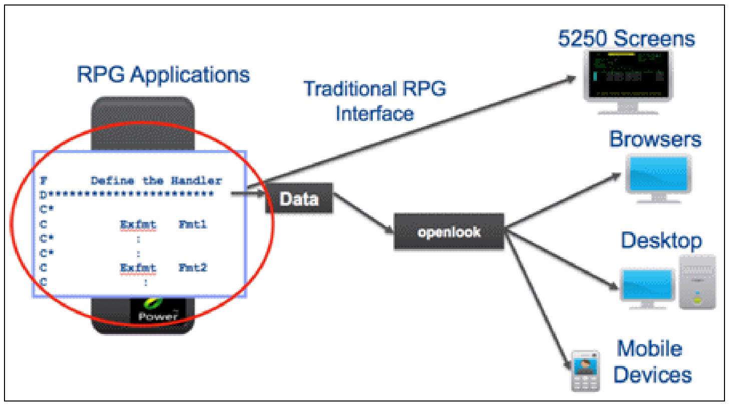


Figure 1-21 Holistic solution

For existing RPG applications that are available to OA enable, only an added line of code to the RPG source is needed to define the handler and direct output.

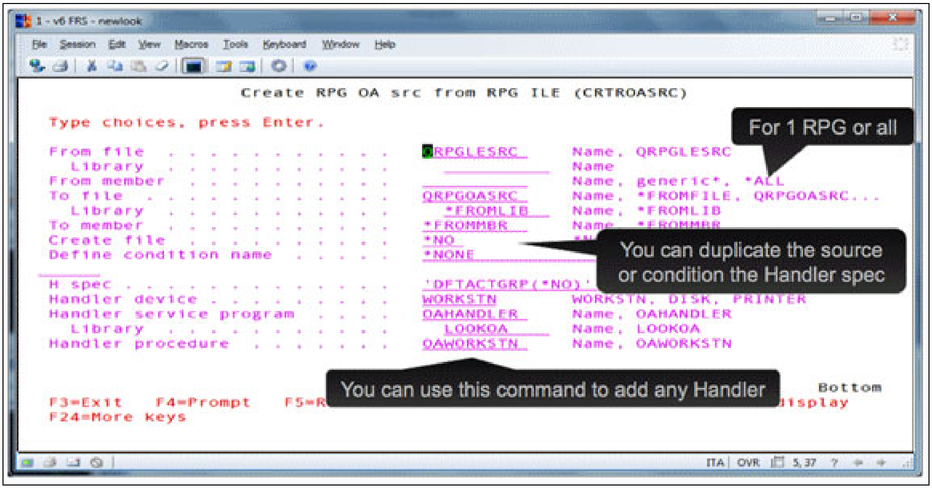
openlook can easily enable an existing RPG program for OA by adding only one line to define the handler as shown in Figure 1-22.



```
F        Define the Handler
D*************************
C*
C              Exfmt    Fmt1
C*               :
C*               :
C              Exfmt    Fmt2
C                :
```

Figure 1-22   Enabling an existing RPG program for OA

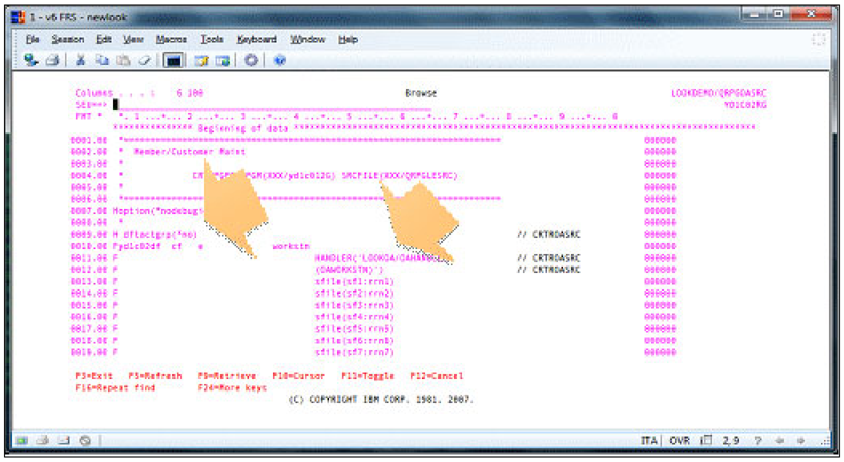
Enabling existing programs for RPG OA is easy. openlook provides the command CRTROASRC to create RPG OA source from an existing RPG ILE program (or many, or all) as shown in Figure 1-23.


1 - v6 FRS - newlook
File Session Edit View Macros Tools Keyboard Window Help
For 1 RPG or all
You can duplicate the source or condition the Handler spec
You can use this command to add any Handler
ITA OVR 5, 37

```
                    Create RPG OA src from RPG ILE (CRTROASRC)

 Type choices, press Enter.

 From file  . . . . . . . . . . .   QRPGLESRC     Name, QRPGLESRC
   Library  . . . . . . . . . . .                 Name
 From member  . . . . . . . . . .                 Name, generic*, *ALL
 To file  . . . . . . . . . . . .   QRPGOASRC     Name, *FROMFILE, QRPGOASRC...
   Library  . . . . . . . . . . .     *FROMLIB    Name, *FROMLIB
 To member  . . . . . . . . . . .   *FROMMBR      Name, *FROMMBR
 Create file  . . . . . . . . . .   *NO
 Define condition name  . . . . .   *NONE

 H spec . . . . . . . . . . . . .   'DFTACTGRP(*NO)'
 Handler device . . . . . . . . .   WORKSTN       WORKSTN, DISK, PRINTER
 Handler service program  . . . .   OAHANDLER     Name, OAHANDLER
   Library  . . . . . . . . . . .     LOOKOA      Name, LOOKOA
 Handler procedure  . . . . . . .   OAWORKSTN     Name, OAWORKSTN

                                                                     Bottom
 F3=Exit   F4=Prompt   F5=R                                        isplay
 F24=More keys
```

Figure 1-23   Using the CRTROASRC command

The command creates a duplicate copy of the original source and adds a workstation handler command (Figure 1-24).



```
 Columns . . . . 1    6 100                 Browse                        LOOKDEMO/QRPGOASRC
 SEU==> █                                                                            YD1C02RG
 FMT *  *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
        *************** Beginning of data ***********************************************************
0001.00  *****************************************************************           000000
0002.00  *  Member/Customer Maint                                                     000000
0003.00  *                                                                            000000
0004.00  *            CR[illegible]PGM(XXX/ydlc012G) SRCFILE(XXX/QRPGLESRC)           000000
0005.00  *                                                                            000000
0006.00  *****************************************************************           000000
0007.00 Hoption(*nodebug[illegible]                                                   000000
0008.00  *                                                                            000000
0009.00 H dftactgrp(*no)                                         // CRTROASRC         000000
0010.00 Fydlc02df  cf   e              workstn                                        000000
0011.00 F                                 HANDLER('LOOKOA/OAHAN[illegible]  // CRTROASRC         000000
0012.00 F                                 (OAWORKSTN)')          // CRTROASRC         000000
0013.00 F                                 sfile(sf1:rrn1)                             000000
0014.00 F                                 sfile(sf2:rrn2)                             000000
0015.00 F                                 sfile(sf3:rrn3)                             000000
0016.00 F                                 sfile(sf4:rrn4)                             000000
0017.00 F                                 sfile(sf5:rrn5)                             000000
0018.00 F                                 sfile(sf6:rrn6)                             000000
0019.00 F                                 sfile(sf7:rrn7)                             000000

 F3=Exit   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle   F12=Cancel
 F16=Repeat find        F24=More keys
                                (C) COPYRIGHT IBM CORP. 1981, 2007.
```

Figure 1-24 CRTROASRC command results

You can start your modernization project with a refacing approach to get quick results. Development and enablement in OA can then take place gradually to take advantage of the RPG environment.

Refacing is still critical for OS screens, or where you do not have source. With Open Access, developers have far more control of the UI. See Figure 1-25.

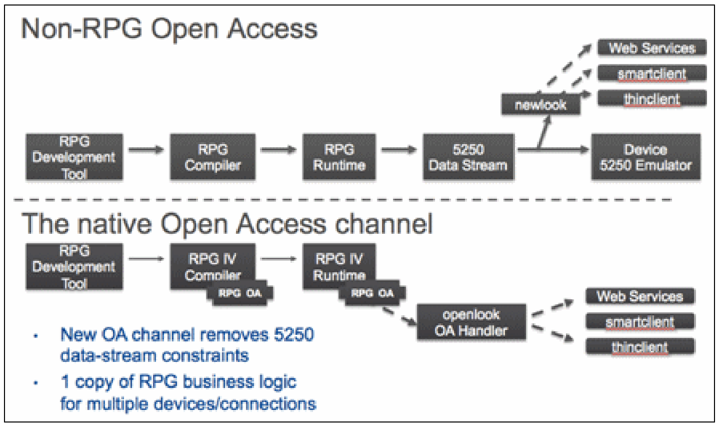


Figure 1-25 Benefits of using Open Access to control the UI

As you develop and deploy RPG OA enabled applications, the user has a seamless modernized experience as they navigate around the system.

The architecture of openlook enables developers to use both OA and non-OA as shown in Figure 1-26.

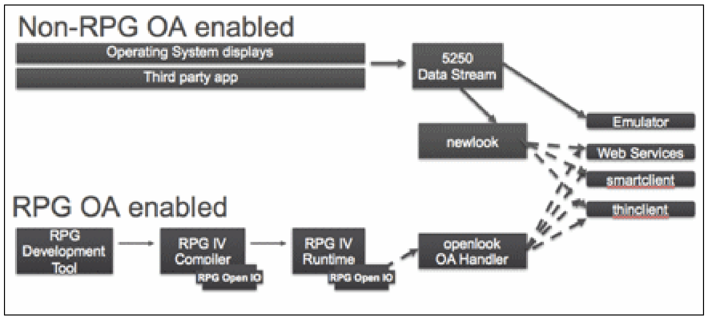


Figure 1-26 Developers can use both OA and non-OA

Within a short amount of time, you can have a modernized version of your 5250 applications, along with the foundation for extending, replacing, and developing new applications in almost limitless ways (Figure 1-27).

Figure 1-27 Developing new types of applications

Open Access Metadata Open Standard (OAMOS)

The RPG Open Access Metadata Open Standard (OAMOS) is established to provide an industry standard for Open Access solutions (Figure 1-28). For more information, refer to the *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

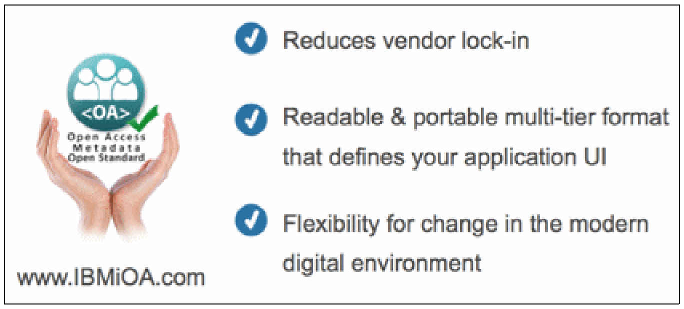

<OA>
Open Access Metadata Open Standard
www.IBMiOA.com
Reduces vendor lock-in
Readable & portable multi-tier format that defines your application UI
Flexibility for change in the modern digital environment

Figure 1-28 OAMOS

For more information and the latest announcements, see:

http://www.ibmioa.com

Open Display Files (OpenDSPF)

Open Display Files enable to developers to build new RPG applications starting with form layout and design, by using newlook, the most advanced IBM i design environment available.

Embedded into the entire looksoftware suite of products, Open Display Files (Figure 1-29) enable RPG developers to develop for rich desktop, any browser, mobile devices, and the cloud entirely within RPG code. This removes the need to learn any other language or use different development tools.

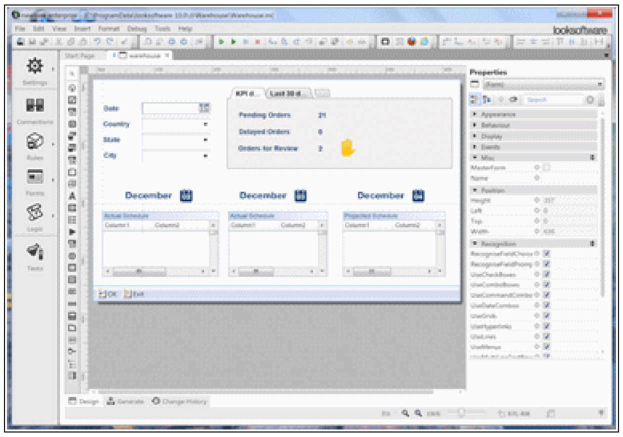


Figure 1-29 Open Display Files

Before Open Display Files were introduced, DDS was used for format, field, and buffer definition, which were needed by the RPG compiler. Also, the UI definition such as field positioning and field attributes were also contained. The UI definition was limited to the constraints of 5250. See Figure 1-30.

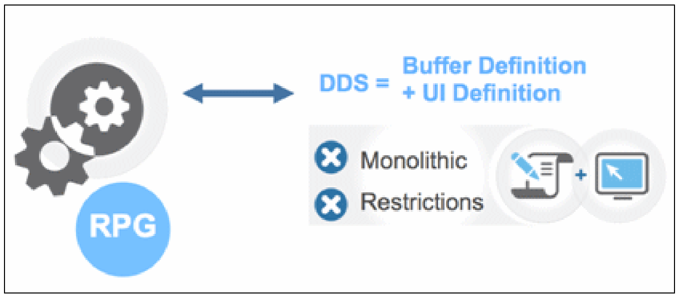


Figure 1-30 The UI is limited to the constraints of 5250

Now, with Open Display Files, DDS is used only for the buffer definition (which is still needed by the RPG compiler).

All UI definition is handled by OAMOS in XML, where there are now no limits (Figure 1-31).

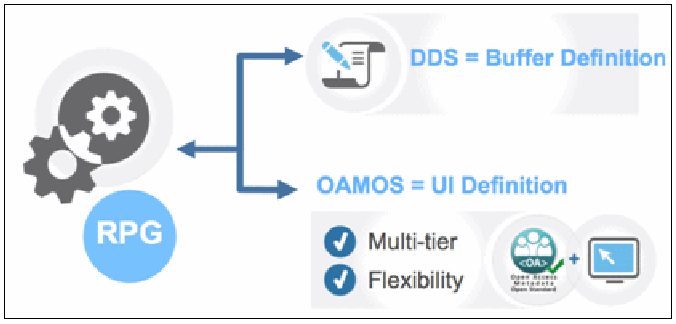


Figure 1-31 Limits are removed

Benefits of openlook

These four fundamentals are behind openlook, and are the key to delivering successful modernization of applications:

1. Applications Anywhere. Delivering the most optimal experience to your users based on their accessibility requirements enables you to support your business functions as effectively and productively as possible. Today's applications support rich desktops, web browsers, and mobile devices.
2. Seamless Integration. Many of today's applications are inefficient because they are not interdependent with other applications. Often users run separate applications and manually copy and paste or rekey data.
3. Power With Simplicity. Delivering applications with multiple interfaces and integration are great concepts, but often they require complex skills and lengthy implementation times. It is important to have a technology that makes it easy to implement, manage, and maintain your modern applications.
4. Single Development Effort. Often development tools can perform only partial functionality, requiring developers to learn and work with multiple IDEs.

Contact looksoftware to request a free assessment of your application.

1.2.2 newlook

newlook enables you to:

1. Easily and quickly modernize existing IBM i applications to rich desktops, the web, and mobile devices
2. Unify and integrate multiple applications by using web services and industry-standard APIs
3. Develop new functions for modern devices and interfaces by using RPG OA with openlook
4. Rapidly generate new applications using .NET with renew

This section describes the newlook IDE and how it forms the foundation for tasks 1 and 2. Developing with RPG OA (openlook) and generating applications with .NET (renew) are covered in more detail in separate sections.

newlook is a leading IDE for IBM i applications. It makes modernizing, designing, and developing state-of-the-art IBM i applications a highly productive experience. Modernizing with newlook is a good choice because it incorporates:

- Simplicity. Through newlook's advanced designer tools, it is easy to design forms. Developing is more intuitive and common layout tasks are automated.
- Productivity. Everything in newlook is tightly integrated and geared towards finishing projects more quickly and easily. You can develop for rich clients, mobile devices, browsers, RPG Open Access, and non-OA, all from the same designer.
- Satisfaction. The newlook designer is easy to use. Many of the core tasks that a developer must do are either automated or just one click away.

Developing with newlook

Getting started with newlook is an intuitive and straightforward experience, especially if you are already familiar with a modern graphical IDE. However, have clear objectives of what you need to deliver before starting. You must set up an environment that works best with your existing development processes.

The newlook IDE has a familiar interface for developers with modern graphical IDE experience, and it is easy to learn for newcomers (Figure 1-32).

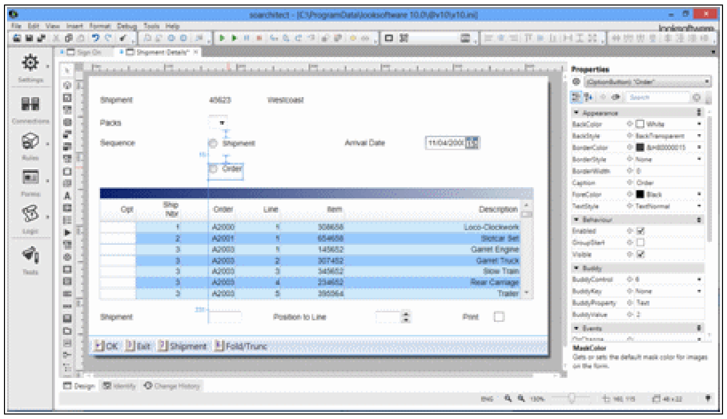


Figure 1-32 The newlook IDE

After you start the newlook IDE, you either need to create a new solution or open an existing one. The solution contains the entire newlook project within a single folder. This configuration simplifies management and maintenance of the modernization project. newlook allows for many configurations when you are working in a team with multiple developers.

It is critical that you have global themes, layout, and preferences defined and set up-front. Also, global filters are a mechanism to look for recurring recognition patterns. This minimizes the effort that is required at the individual window level.

Through automated smart-guides, rulers and enhanced toolbars, newlook anticipates what a developer is working on and helps with the process of designing forms. Further additions, such as the hints system, make suggestions to help guide a developer intuitively through the design and layout process.

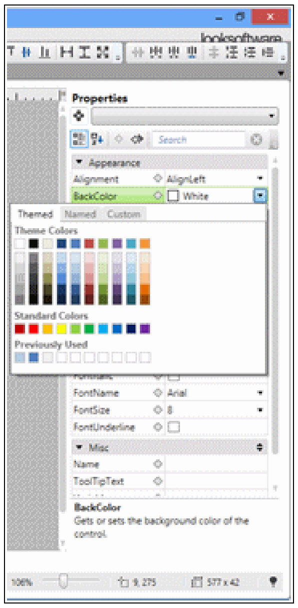
The newlook designer toolbar provides access to the layout of forms as shown in Figure 1-33.

Figure 1-33 The newlook designer toolbar

newlook's designer makes it easy to develop interfaces with the following features:

- Control snapping and smart guides to make high-quality graphical design easy and intuitive
- Control palette to support multi-channel controls, their behavior, and integration
- Intelligent rulers to produce perfect layout
- Control painter to make it simple to apply identical properties to multiple controls
- In-place editing to enable changes exactly where you see them
- Designer zooming to make design consistent across all window sizes
- Extended workspace to allow easy design enhancement of busy screens
- Device form size support to make designing for mobile devices a breeze
- Designer hints that enable the IDE to anticipate wanted actions and guide you to the quickest result
- Single-click browser preview to allow instant testing of your solution with any browser

With the properties sheet, developers have powerful and fine-grain control of form layout (Figure 1-34).



Appearance
Alignment AlignLeft
BackColor White
FontName Arial
FontSize 8
FontUnderline
Misc
Name
ToolTipText

BackColor
Gets or sets the background color of the control.

Figure 1-34 Properties sheet

The foundation of newlook

Four fundamentals behind newlook are key to delivering successful modernization of applications:

1. Applications anywhere. Delivering the most optimal experience to your users based on their accessibility requirements enables you to support your business functions as effectively and productively as possible. Today's applications support rich desktops, web browsers, and mobile devices.

   newlook enables you to take any existing 5250 application and provide:

   – Optimized interfaces for rich desktop, web and mobile devices, including smartphones and tablets

   – Easy cloud delivery

   – The correct interface for the correct user with responsive and secure connectivity to IBM i applications

2. Seamless integration. Many of today's applications are inefficient because they are not interdependent with other applications. Often, users run separate applications and manually copy and paste or rekey data.

   newlook enables developers to:

   – Unify multiple applications into one

   – Use all of your devices' capabilities, including Windows PCs, smartphones, and tablets

   – Harness the web with existing IBM i applications

3. Power with implicity. Delivering applications with multiple interfaces and integration are great concepts, but often they require complex skills and lengthy implementation times. It is important to have a technology that makes it easy to implement, manage, and maintain your modern applications.

4. Single development effort. Often development tools have only partial functionality, requiring developers to learn and work with multiple IDEs.

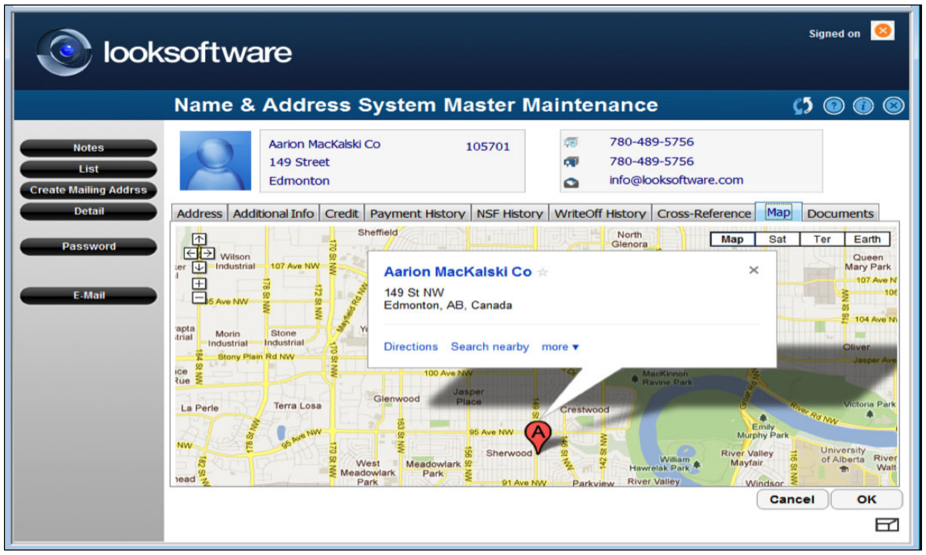
newlook is a single IDE that enables developers to:

– Develop once and deploy to many interfaces
– Use existing skills, including RPG, .NET, HTML, JavaScript, and CSS
– Work with the most advanced design environment for IBM i

Developing rich desktop solutions

Often the starting point for modernization of IBM i applications is replacing older 5250 applications with new updated graphical interface access on rich desktops. Rich desktops are often the ideal replacement for existing 5250 users who are running data-intensive tasks like order entry and customer service. Rich desktop clients have high-performance characteristics that include keyboard buffering and native integration to other applications, including the underlying operating system. A modern rich client can provide many other benefits. These benefits are described in this section.

looksoftware's rich desktop offering, smartclient, runs natively on Microsoft Windows and provides all the benefits that over 90% of PC users worldwide already interact with everyday. Because it runs natively, it has full access to the Windows API and can improve the experience for users who are running Microsoft PCs. See Figure 1-35.



Aarion MacKalski Co 105701
149 Street
Edmonton

780-489-5756
780-489-5756
info@looksoftware.com

Aarion MacKalski Co
149 St NW
Edmonton, AB, Canada

Figure 1-35 The looksoftware smartclient offering

Core benefits of modernizing 5250 applications with smartclient include:

- Drastically reduce the amount of navigation around your applications
- Combine multiple screens into one
- Add visual icons and other graphical aids that makes your applications more intuitive
- Add charts and graphics for a visualization of data
- Embed images (product renderings, diagrams, and so on)
- Integrate to desktop applications such as Microsoft Word, Excel, and Outlook
- Integrate with other enterprise applications
- Reduce training times for new users

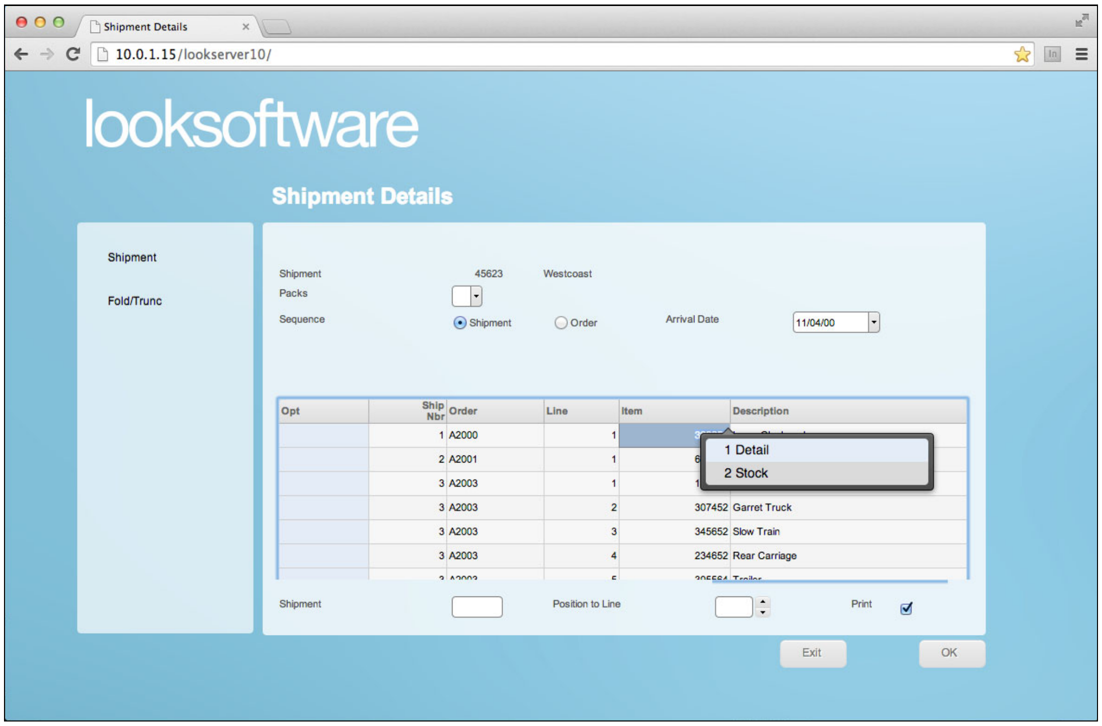
See Figure 1-36.

Figure 1-36   Benefits of modernizing 5250 applications with smartclient

Developing web browser solutions

Zero deployment for external and casual access for remote users provides significant business benefits. This access provides automated access to applications and data that otherwise requires manual methods such as calling into headquarters and having an internal user look up something.

looksoftware's thin client supports all popular browsers, and uses HTML5 and CSS3 for responsive and functional web delivery of applications. See Figure 1-37.



Shipment	45623	Westcoast
Packs		
Sequence	[✓ Shipment] Order	Arrival Date 11/04/00

Opt	Ship Nbr	Order	Line	Item	Description
	1	A2000	1	[illegible]	[illegible]
	2	A2001	1	[illegible]	[illegible]
	3	A2003	1	[illegible]	[illegible]
	3	A2003	2	307452	Garret Truck
	3	A2003	3	345652	Slow Train
	3	A2003	4	234652	Rear Carriage
	3	A2003	5	305564	Trailer

Shipment   Position to Line   Print ✓

Figure 1-37   The looksoftware thin client

looksoftware's lookserver and thin client allow you to deploy your IBM i applications to any popular browser over the web. Using the HTML, JavaScript, and CSS industry standards ensures maximum interoperability and flexibility. These applications also allows you to work with your security requirements (VPN, HTTPS, and so on) to ensure that your system is protected.

Developing for web browsers is almost identical to developing for rich clients. However, there are some things to consider based on some behavioral characteristics that thin client has over smartclient. See Figure 1-38.

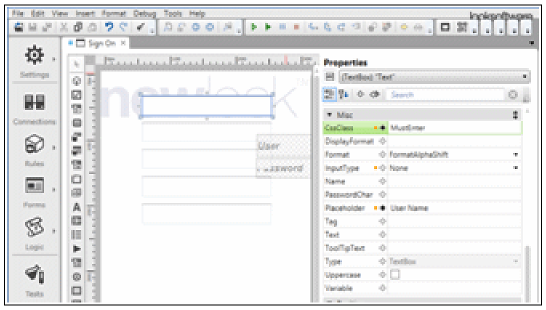


Figure 1-38 Developing for web browsers

lookserver uses skins to help manage and maintain different displays for the same applications. Skins provide a way to use templates for browser and mobile interfaces that not only help applications look great, but also work great. They can transform the UI to work in a way that is optimal for the device and user. A design that is based on UI preferred practices can dramatically reduce the effort to get your solution to market.

Developing mobile solutions

newlook provides enhanced support for smartphones, tablets, and browser delivery that make it easier than ever to deliver high-quality mobile solutions. Popular devices, including Apple iPhone, iPad, and Android devices, are automatically supported so that it is possible to get great results in a short amount of time with little effort. See Figure 1-39.

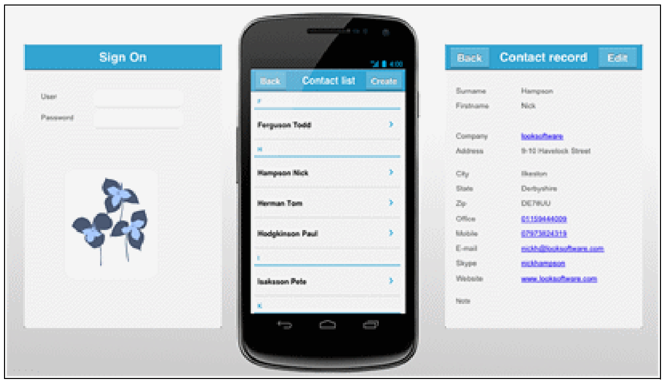


Figure 1-39 newlook and lookserver make development and deployment of IBM i applications to mobile

Templates are provided that help with general layout tasks and styling, such as fonts, colors, and spacing. Not only is this method productive, but it shields developers from having to learn and maintain HTML5 and CSS3 code.

newlook enables developers to perform these tasks:

- Instantly modernize any 5250 application for access from many modern mobile devices
- Use pre-set form sizes for devices that include Apple iPhone, iPad, and Android phones and tablets
- Test deployment with lookserver with one-click
- Take advantage of support for HTML5 and CSS3

See Figure 1-40.

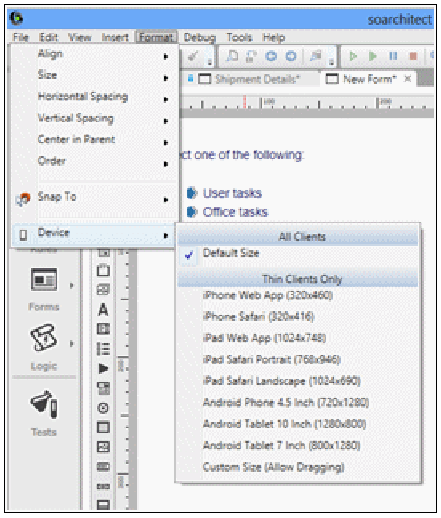


Figure 1-40 Pre-set device layouts for popular mobile interfaces, including iOS and Android devices

Figure 1-41 shows some examples of how tablets and smartphones render IBM i applications that are modernized with newlook.

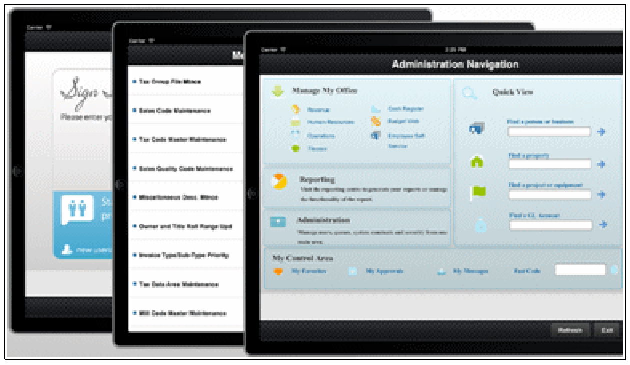


Figure 1-41 iPad example of an IBM i application that is modernized with newlook

Figure 1-42 shows an iPhone example of an IBM i application, with an iOS look that is optimized for touch.

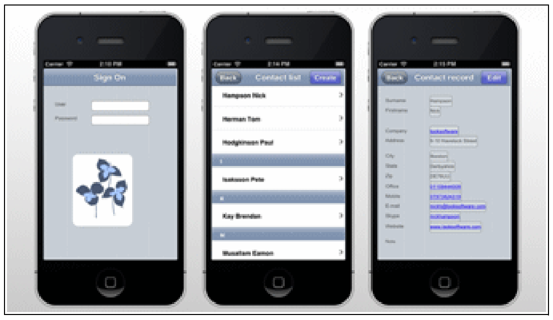


Figure 1-42 An iPhone example of an IBM i application

Integrating applications

Although the focus so far in this section is on accessibility from today's popular devices and interfaces, newlook also has extensive capabilities to unify and integrate applications.

Here are some of the common integration tasks that are implemented with newlook:

- Combining multiple screens into one
- Automating creation of Word documents based on live IBM i application data
- Downloading subfiles to Microsoft Excel spread sheets
- Adding image integration to a product database
- Embedding a Google Map with a customer address record
- Using a web service to get real-time access to package tracking
- Enabling signature capture with a mobile device
- Scanning a bar code with a mobile device camera

newlook enables almost limitless integration that includes these built-in developer functions:

- JavaScript
- VB Script
- Macro programming
- Consuming web services
- Direct data access (DDM, ODBC, ADO, and so on)
- HTML

Developers can also encapsulate functionality of existing 5250 applications and expose them as APIs or Web Services, including all business rules and validations.

For more information, see:

`http://www.looksoftware.com`

1.3 BCD Software

This section describes the BCD Software Presto and WebSmart products.

1.3.1 Presto

Many organizations still access some or all of their IBM i applications through text-based 5250 terminals or terminal emulators. However, these applications do not take advantage of the productivity benefits that the web provides, and many users, managers, and even IT staff perceive the IBM i as outdated solely because of 5250-based “green screens”. As a result, IBM i organizations are facing increased pressure to modernize their 5250 application interfaces.

Web-based graphical user interfaces (GUIs) provide the following benefits:

- They improve the organization's image with a more contemporary look.
- They reduce training costs and can speed up data entry by streamlining workflows and simplifying the user experience with a browser-based interface that is commonly used.
- You can use the full power of web browsers, including advanced UI components and visual elements like images, video, and charts.
- They are easier to deploy than 5250 emulators because users can easily access them from browsers on PCs, Macs, tablets, and smartphones.

- You can repurpose your heritage applications so business partners, vendors, and even customers can access them.

Solution overview

BCD Presto is a web enablement and modernization tool that helps IBM i programmers meet the shortest deadlines by automatically rendering IBM i green screens as web pages. You do not need to select which applications to modernize because Presto uses the 5250 datastream to web enable all RPG and COBOL programs on the fly without requiring the source code. With Presto's 5250 data stream approach, all of your third party, older RPG II-IV programs and those missing the source code are given a web GUI. You can also give new RPG programs a web GUI using Presto's RPG Open Access (OA) handler.

On installation, Presto automatically transforms all of your function keys and menu items into clickable buttons and links, and provides you with other global configuration options. Presto includes a visual editor that makes it easy to add new functions such as drop-down boxes, date pickers, and images to your web-enabled screens without needing to code HTML. Presto Designer also provides access to the HTML, CSS, and JavaScript, which gives programmers who prefer to code by hand the flexibility to do so.

Presto web-enabled screens are easy to deploy because users need only a browser to securely access the web-enabled screens on PCs, Macs, tablets, and smartphones. No ActiveX, Windows Servers, or other software is required.

A phased approach to modernization

Some organizations take a phased approach to modernization with Presto. At first, they deploy the consistent look and experience that Presto provides on installation for most of their screens. In the second phase, they further enhance and add new functions to their most-used screens. In the last phase, they might integrate some of their web-enabled screens with other web applications and technologies to further improve work flows. This process is shown in Figure 1-43.

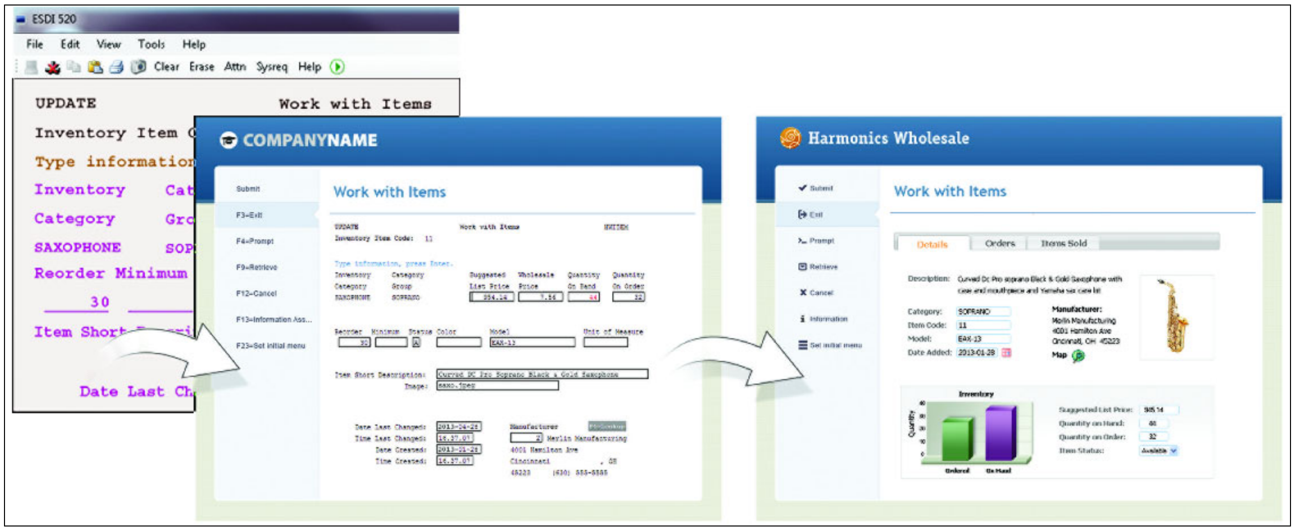


Figure 1-43   A 5250 window transformed to a web page by Presto (Phase 1), then further enhanced (Phase 2 and 3)

Phase one: Rapid results

Presto allow you to take advantage of the modern look and new functions that browsers provide by deploying your 5250 applications as web applications almost immediately.

As soon as you install Presto on your IBM i, which can be done in under 30 minutes, you can access all of your IBM i screens as web pages from a browser. Presto accomplishes this goal by automatically intercepting the 5250 data stream, and dynamically translating and reshaping your programs to run as web pages (Figure 1-44 on page 34). This process does not require any source code changes or manually identifying your screens. Because Presto is a web enablement tool and not an emulator, test your screens before you deploy them to your users.

RPG OA support is included with Presto so you have more options to deliver modern web applications. You can use Presto's RPG OA handler to give new RPG programs a web GUI, transform subfiles into scrollable grids with sortable columns and automatically add datepickers to date fields. You can also seamlessly intermingle OA screens with screens that were web-enabled using Presto's 5250 datastream approach.

Figure 1-44 shows the Presto runtime architecture.

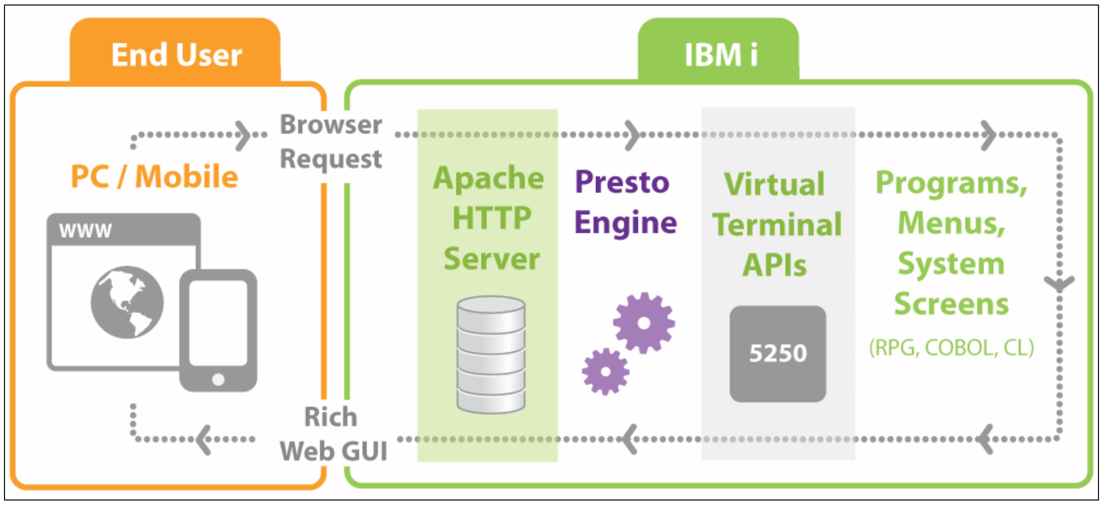


Figure 1-44 The Presto runtime architecture

Presto perform these tasks:

1. The user browses to a URL in their browser on their PC or mobile device. That URL points to the IBM i.
2. The browser communicates with the IBM HTTP Server (powered by Apache), which processes the request and evokes the Presto Engine.
3. The Presto Engine starts the Virtual Terminal session. It processes requests in both directions, rendering your 5250 panels as web pages to the browser by transforming the 5250 data stream into HTML, CSS, and JavaScript. It also returns data in the format that the Virtual Terminal requires.
4. The original RPG or COBOL programs and menus run in the background. Because Presto works on the 5250 data stream level, your programs are not affected in any way and can still be accessed on 5250 emulators by users who prefer 5250 panels.

You can also enhance your panels by using the Presto global environment settings. A few examples of global transformations are the Presto automatic conversion of function keys and menu options into clickable buttons and links, and the ability to select a skin that applies a modern web look to all of your screens. You can also customize the skins to match your organization's graphic design. The benefit of global changes is that they enhance all of your screens without having to individually customize them.

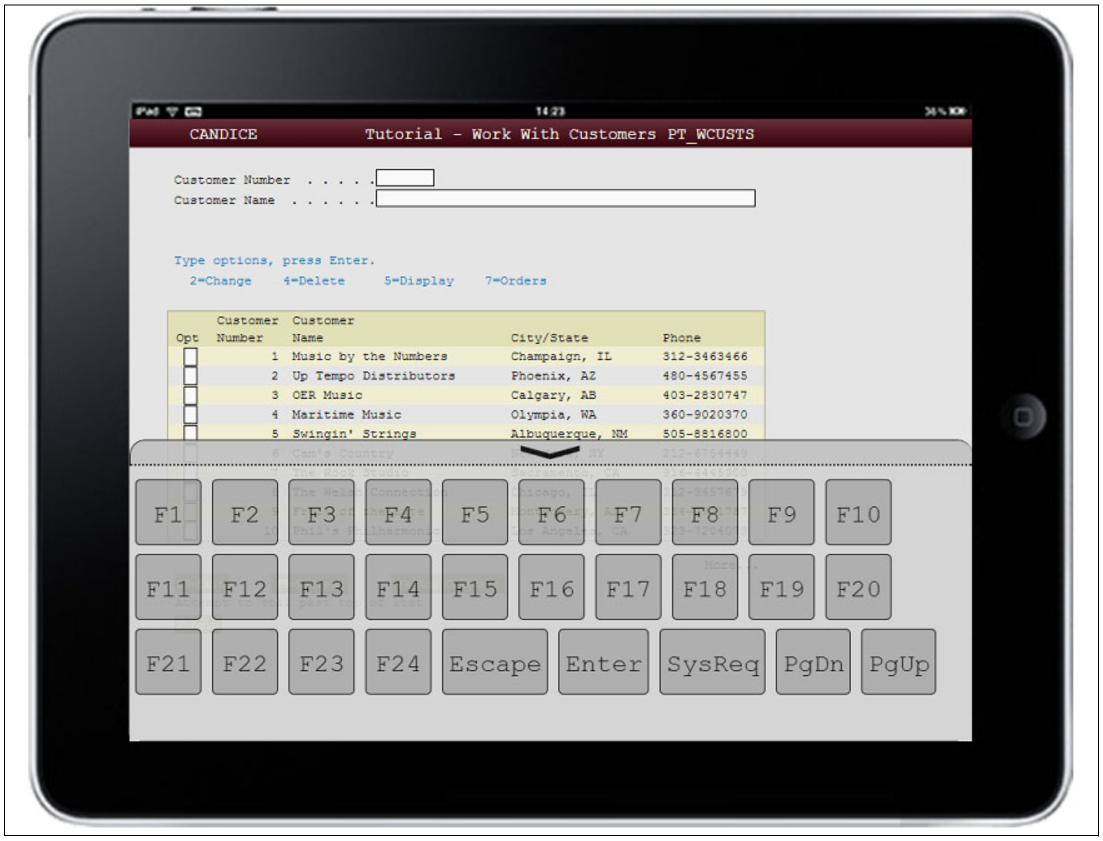
Critical to any enterprise solution, Presto supports multiple environments and languages. Presto environments are a set of global settings for a group of programs or screens on your system, providing you with test and production environments. If you are an independent software vendor (ISV) who needs to web enable your existing 5250 solutions, Presto environments give you the ability to simultaneously have independent versions of the same web-enabled applications. ISVs can use environments to make the same application have a customized look for each client.

Presto supports your IBM i and database language settings. Your web-enabled screens display the language that you configure with your IBM i and database.

Accessing screens from mobile devices

Extending the function of your native IBM i-based applications to mobile devices helps extract more value from your investment in IBM i technology by improving business processes. Presto includes built-in features to support mobile devices. When an iPhone, iPad, or Android device is used to access a web-enabled green screen, the Presto built-in intelligence automatically detects the device type and displays the mobile layout. The mobile layout can also be customized for other devices.

The Presto mobile layout hides the top banner of the window to make more efficient use of the screen size and reduce the size of the page so it loads faster. It also includes a virtual keyboard for function keys (Figure 1-45). This keyboard is significant because some mobile devices, including iPhones and iPads, do not have function keys on their keyboards, which is a part of almost every 5250 application.



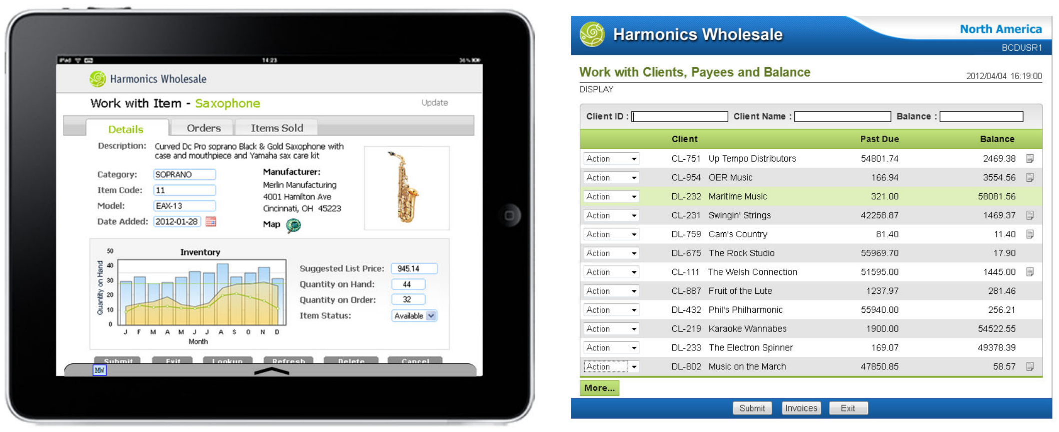
```
CANDICE              Tutorial - Work With Customers PT_WCUSTS

Customer Number . . . . .
Customer Name  . . . . . .

Type options, press Enter.
  2=Change    4=Delete    5=Display    7=Orders

     Customer  Customer
Opt  Number    Name                     City/State         Phone
            1  Music by the Numbers     Champaign, IL      312-3463466
            2  Up Tempo Distributors    Phoenix, AZ        480-4567455
            3  OER Music                Calgary, AB        403-2830747
            4  Maritime Music           Olympia, WA        360-9020370
            5  Swingin' Strings         Albuquerque, NM    505-8816800
```

Figure 1-45   A Presto web-enabled window displayed in an iPad with its virtual keyboard expanded

This first stage might be sufficient for some, or even all, of your screens, regardless of whether they are accessed from desktops or mobile devices. From this starting point, you can make further customizations that add more business value to your applications.

Phase two: Add new functionality

Presto makes it easy to further enhance screens by adding new UI elements and functionality like drop-down boxes (Figure 1-46). There are several different ways to customize your web-enabled desktop or mobile screens, including the Presto Visual Editor, SQL Queries, and changes to the HTML, CSS, and JavaScript. See Figure 1-46.



Work with Item - Saxophone

Description: Curved Dc Pro soprano Black & Gold Saxophone with case and mouthpiece and Yamaha sax care kit

Category:	SOPRANO
Item Code:	11
Model:	EAX-13
Date Added:	2012-01-28

Manufacturer:
Merlin Manufacturing
4001 Hamilton Ave
Cincinnati, OH 45223
Map

Suggested List Price:	945.14
Quantity on Hand:	44
Quantity on Order:	32
Item Status:	Available

Work with Clients, Payees and Balance 2012/04/04 16:19:00

DISPLAY

Client ID : Client Name : Balance :

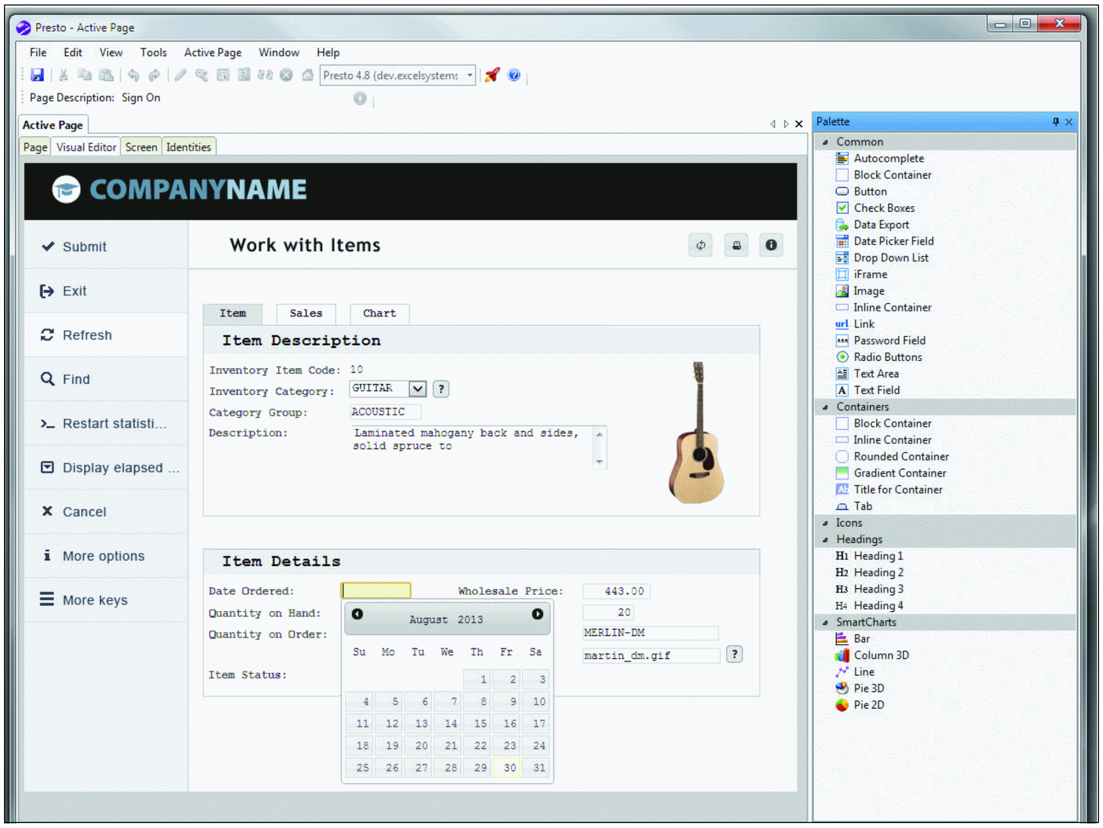
	Client		Past Due	Balance
Action	CL-751	Up Tempo Distributors	54801.74	2469.38
Action	CL-954	OER Music	166.94	3554.56
Action	DL-232	Maritime Music	321.00	58081.56
Action	CL-231	Swingin' Strings	42258.87	1469.37
Action	DL-759	Cam's Country	81.40	11.40
Action	DL-675	The Rock Studio	55969.70	17.90
Action	CL-111	The Welsh Connection	51595.00	1445.00
Action	CL-887	Fruit of the Lute	1237.97	281.46
Action	DL-432	Phil's Philharmonic	55940.00	256.21
Action	CL-219	Karaoke Wannabes	1900.00	54522.55
Action	DL-233	The Electron Spinner	169.07	49378.39
Action	DL-802	Music on the March	47850.85	58.57

Figure 1-46 Two Presto web-enabled screens with added UI elements

Presto includes a Windows-based IDE called the Presto Designer. Programmers and IT staff with no web development experience can be productive immediately by using the Presto Designer Visual Editor (Figure 1-47 on page 37). The visual editor provides these features:

- Drag fields around the window.
- Change the colors, fonts, and other properties of elements or text on the window.
- Add visual elements like tabs, images, and charts.
- Add UI elements, such as date pickers, drop-down boxes, and autocompletes that speed up data entry and reduce keying errors.
- Add links to PDF or Word documents.
- Add JavaScript events to fields for a more responsive user experience.
- Further enhance screens for mobile devices.

Figure 1-47 shows a window enhanced by using Presto Visual Editor.



Presto 4.8 (dev.excelsystems

Page Description: Sign On

Work with Items

Item Description

Inventory Item Code: 10
Inventory Category: GUITAR
Category Group: ACOUSTIC
Description: Laminated mahogany back and sides, solid spruce to

Item Details

Date Ordered:
Wholesale Price: 443.00
Quantity on Hand: 20
Quantity on Order:
MERLIN-DM
martin_dm.gif
Item Status:

August 2013

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 1-47 The Presto Designer showing window enhanced with Presto Visual Editor

The Presto Designer also gives you full access to HTML, JavaScript, and CSS so you have the flexibility to add any other enhancements that you might want to make.

SQL Queries for database-driven UI elements

Some of the UI elements that you add through the Visual Editor are driven by SQL Queries that you define in Presto. SQL Queries allow you to extend your screens beyond the 5250 data stream by accessing data that is not in the original program. They are most frequently used to dynamically populate drop-down boxes, autocompletes, and charts with data from IBM DB2® files, and to export the DB2 query result to a spreadsheet or HTML page. They can also be used to retrieve and update DB2 data that was not part of the original program. SQL Queries are non-intrusive because they do not require any changes to the underlying program object or source code.

Phase three: Integrated web environment

Phase three integrates web-enabled screens with other web applications and technologies. Depending on the application and your organization's requirements, you might decide to carry only a small percentage of screens (if any) forward to phase three. Having the flexibility to enhance screens at this level helps add more sophisticated functionally and future-proof your applications.

The following are examples of these types of enhancements:

- JSON, Ajax, and other technologies to call other programs and perform web services for approving a credit card, tracking a shipment, or displaying a location in a map.

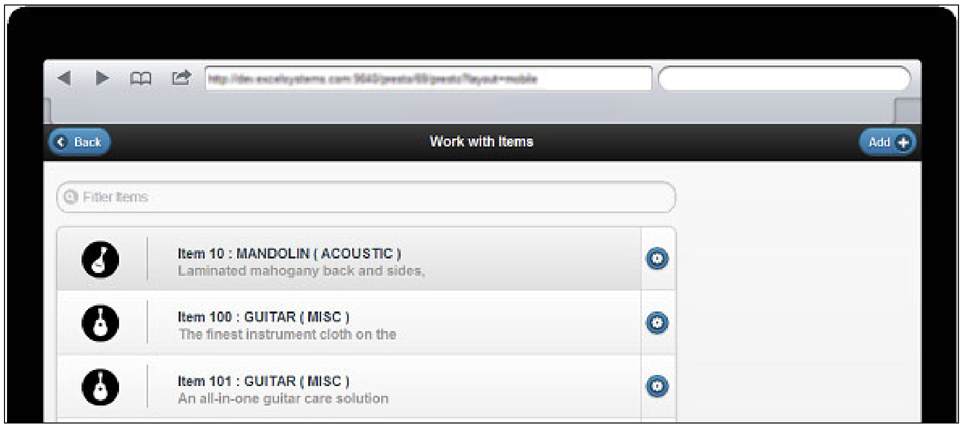
- Linking to other web applications (PHP, RPG, .Net, and so on), including new web applications that are created with WebSmart, which is part of BCD's modernization suite.
- Adding web plug-ins and scripts like jQuery, which is included and used extensively in Presto.

Mobile Optimization with jQuery and Phonegap

While Presto offers mobile features, it also provides the flexibility for you to take your applications to the next level. You can integrate Presto with other technologies to provide the mobile appearance that users expect and to take advantage of the device's hardware features (such as GPS and camera):

- Use jQuery Mobile to provide a mobile-optimized design that enhances the user experience when screens are accessed from mobile devices (Figure 1-48). Using jQuery Mobile with Presto requires you to change the HTML.
- Integrate with Phonegap to use the device's hardware features. These features can include accessing the camera to take photos or scan a bar code, and using the GPS for geolocation.

Figure 1-48 shows a Presto web-enabled window.



Item 10 : MANDOLIN (ACOUSTIC)
Laminated mahogany back and sides,

Item 100 : GUITAR (MISC)
The finest instrument cloth on the

Item 101 : GUITAR (MISC)
An all-in-one guitar care solution

Figure 1-48 A Presto web-enabled window that optimized for mobile devices by using jQuery Mobile

Modernization suite

You can optionally combine Presto with other BCD tools to further modernize your IBM i applications and processes by building your own customized and integrated solution. Start with one tool, a couple of tools, or the complete BCD ClearPath modernization suite to meet your needs:

http://www.bcdsoftware.com/iseries400solutions/clearpath/

WebSmart is the BCD rapid desktop and mobile web application development tool for creating new PHP or RPG web applications. You can link your Presto web-enabled screens to your WebSmart web applications (and any other web application). You can also use WebSmart to add more sophisticated functionality to your Presto screens, including emailing, uploading files, and processing web services:

http://www.bcdsoftware.com/iseries400solutions/websmart/

Presto optionally includes the BCD Nexus Portal. Nexus provides a secure single point of access to your enterprise information, including your Presto web-enabled screens, other web apps and pages, documents, dashboards, and productivity tools. It also includes a web menu system and single sign-on for your Presto apps that you can use to hide some 5250 navigation menus:

http://www.bcdsoftware.com/iseries400solutions/nexusportal/

Security

Because Presto runs as a layer over your green screens, many of their security conventions, such as object authorities, are inherited by your web-enabled screens. The web layer in Presto can provide extra security in the form of encryption. Many of the same security considerations that apply to conventional web applications also apply to Presto web-enabled applications. Consider the following concerns:

1. Data transmission between client (browser) and server (IBM i/Presto server). To secure data transmission, use Secure Socket Layers (SSL). SSL encrypts data before transmission, sends it encrypted, then decrypts it. All major commercial B2C sites use SSL.
2. Network configuration (firewalls, VPN, and so on). A firewall allows you to control access to your network, including your IBM i system. By default, all external access to your network should be denied. You can then open a port to the IBM HTTP Server if you want to allow external access to your web-enabled screens. You also control whether you want to require a VPN to allow remote access to screens.
3. IBM HTTP Server (powered by Apache). The default Apache configuration that is shipped with Presto only makes Presto-related functionality available. Any changes to the configuration are subject to the standard Apache considerations.
4. IBM i user profile and object security. Presto uses the user's IBM i user profile so that all of their object authorities are automatically retained. You can also specify a device description in Presto.
5. SSO with auto-logins. The Presto auto-logins feature allows you to configure a way for authorized users to go directly to a program or menu without another login through Presto. This feature is primarily intended to be used with Nexus, the BCD web portal, although you can also tie it into your own portal software or web applications by writing your own API.

Required skills

The skills that you need depend on the level of customizations that you plan to make to your web-enabled screens. Using Presto and making changes using the Visual Editor requires minimal HTML skills. BCD has detailed documentation, code samples, and a technical support team to assist you as you learn.

As your customizations become more sophisticated, an understanding of HTML is greatly beneficial. If you choose to make phase three customizations by integrating web applications and technologies with your screens, you also need a deeper understanding of web technologies.

Resources

For more information, see the following resources:

- Data sheet

  http://www.bcdsoftware.com/iseries400solutions/presto/PDFs/ibmi-web-enablement-modernization.pdf

- Webinars

  http://www.bcdsoftware.com/ibmiwebinars/ondemand/

- White papers:

  – IBM i Web Enablement: Five Decisions to Make Before You Start

    http://iprodeveloper.com/bcd-business-computer-design/ibm-i-web-enablement-five-decisions-make-you-start

Supported platforms

Presto requires IBM i V5R4 or later, and uses the free IBM HTTP Server (powered by Apache). Users can access Presto screens from PCs, Macs, tablets and smartphones by using browsers such as Chrome, Firefox, Internet Explorer, and Safari. It does not use IBM WebSphere®, Windows Server, or ActiveX so there is no need to purchase or configure any of those products.

Download and ordering information

For download and ordering information, see the following link:

http://www.bcdsoftware.com/richgui

1.3.2 WebSmart ILE, PHP, and Mobile

Web applications can play a strategic role in any IBM i organization because they provide several benefits:

1. Improved workflow by extending your applications to employees, business partners, and customers. Giving these groups access to web applications that allow them to self-manage inventory, pricing, and orders improves customer service.
2. Increased revenue (B2B and B2C e-commerce) and improved decision making by providing real-time information to the people who need it.
3. The familiar appearance that web applications provide can reduce training costs for employees who are unfamiliar with 5250 screens. They also increase user productivity with intuitive UIs that use drop-down boxes, menus, calendars, tabs, and images that are used in users' everyday lives.
4. Web applications give your IBM i system a modern look to match its cutting-edge technology. This look can go a long way in changing the perception that the IBM i is outdated.
5. Users with appropriate permissions can access web applications from anywhere on PCs, Macs, tablets, and smartphones because all they need is a browser to do so.
6. Web applications require no interactive processor power on your IBM i, potentially saving you thousands of dollars in server hardware upgrade fees.

Solution overview

WebSmart is a multi-award-winning, rapid web application development tool for creating desktop or mobile web applications. WebSmart is easy for IBM i and other programmers to understand, even if they do not have web development experience. Its program templates provide immediate results and reduce the web development learning curve.

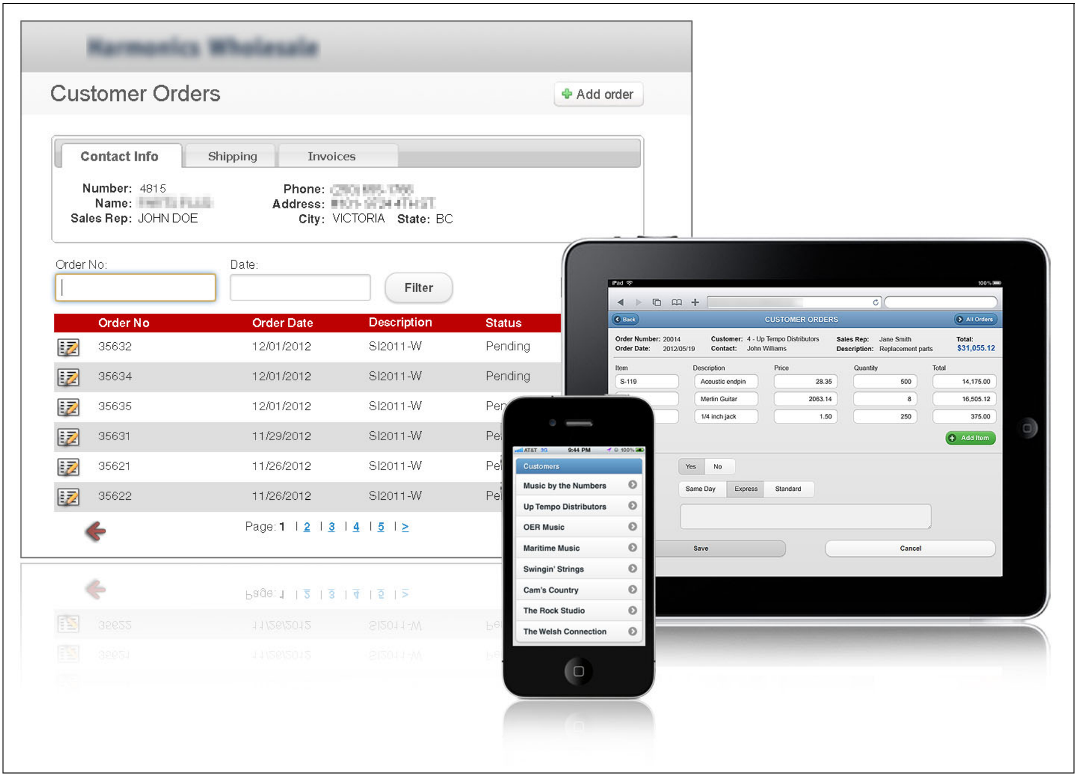
WebSmart comes in two editions (WebSmart Mobile is included in both editions):

- ILE, for creating IBM i-centric RPG web applications that access DB2, MS SQL and MySQL databases. The apps run on IBM HTTP Server (powered by Apache).

- PHP, for creating open source PHP applications that run on multiple platforms (IBM i, Windows, Linux, and UNIX) and access databases such as DB2, MySQL, and Oracle. WebSmart PHP applications run with Zend Server if they are hosted on the IBM i.

You can use either edition to create order entry, invoicing, inquiries, B2B/B2C shopping carts, dashboards, and other web applications. With WebSmart, you are creating new applications that can use existing RPG code, including validation and pricing routines. WebSmart supports many web technologies such as web services, Ajax, and jQuery to build rich web applications.

Both WebSmart editions include templates that jumpstart your web application development by generating the initial business and client-side logic, in addition to a full-featured IDE to rapidly build any type of desktop or mobile web application. The IDE includes a code editor, project and change management, and more. WebSmart also integrates with commercial change management solutions from other vendors. See Figure 1-49.



Number: 4815
Name: [redacted]
Sales Rep: JOHN DOE
Phone: [redacted]
Address: [redacted]
City: VICTORIA State: BC

Order No:
Date:

	Order No	Order Date	Description	Status
	35632	12/01/2012	SI2011-W	Pending
	35634	12/01/2012	SI2011-W	Pending
	35635	12/01/2012	SI2011-W	Pen
	35631	11/29/2012	SI2011-W	Pe
	35621	11/26/2012	SI2011-W	Pe
	35622	11/26/2012	SI2011-W	Pe

Page: 1 | 2 | 3 | 4 | 5 | >

Order Number: 20014
Order Date: 2012/05/19
Customer: 4 - Up Tempo Distributors
Contact: John Williams
Sales Rep: Jane Smith
Description: Replacement parts
Total: $31,055.12

Item	Description	Price	Quantity	Total
S-119	Acoustic endpin	28.35	500	14,175.00
	Merlin Guitar	2063.14	8	16,505.12
	1/4 inch jack	1.50	250	375.00

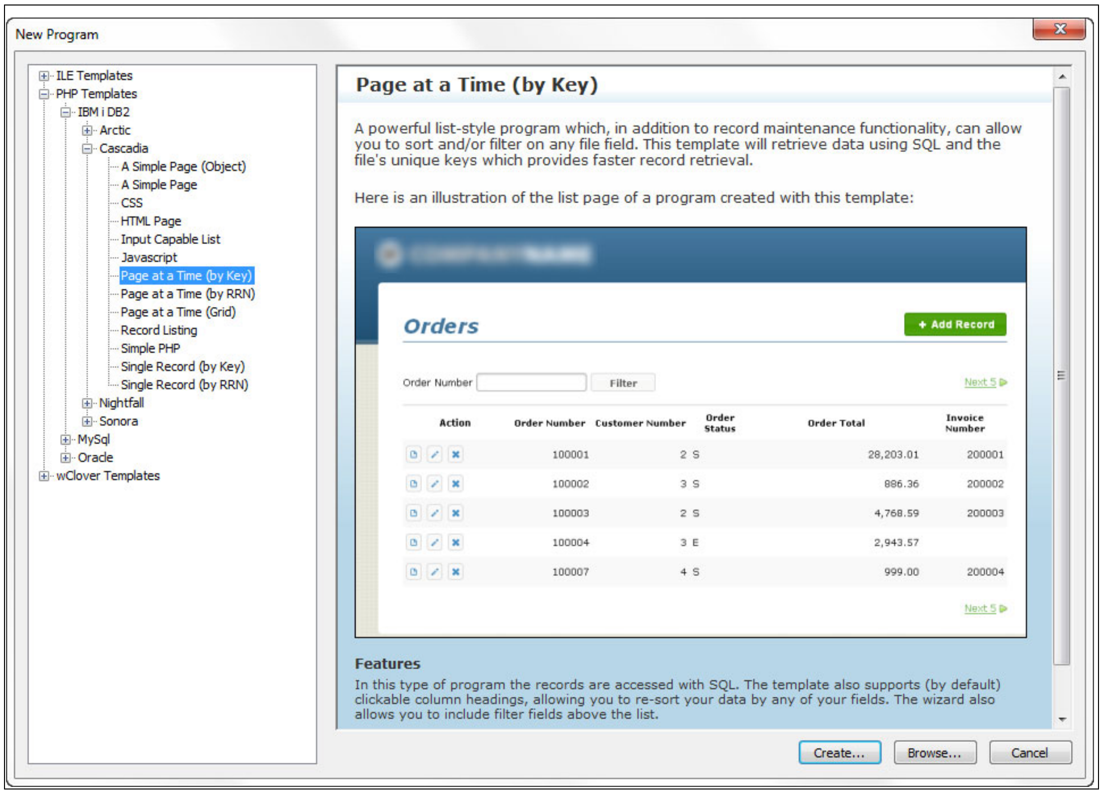
[✓ Yes] No
Same Day [✓ Express] Standard

Customers:
- Music by the Numbers
- Up Tempo Distributors
- OER Music
- Maritime Music
- Swingin' Strings
- Cam's Country
- The Rock Studio
- The Welsh Connection

Figure 1-49 Web applications that are developed with WebSmart

How it works

WebSmart consists of three main components: one on the PC and two on the server. The core WebSmart component is a Windows based IDE. The WebSmart IDE includes intelligent program templates (Figure 1-50) that prompt you for files and fields, then generate the starting HTML, CSS, and RPG or PHP.



Page at a Time (by Key)

A powerful list-style program which, in addition to record maintenance functionality, can allow you to sort and/or filter on any file field. This template will retrieve data using SQL and the file's unique keys which provides faster record retrieval.

Here is an illustration of the list page of a program created with this template:

Orders

Action	Order Number	Customer Number	Order Status	Order Total	Invoice Number
	100001	2	S	28,203.01	200001
	100002	3	S	886.36	200002
	100003	2	S	4,768.59	200003
	100004	3	E	2,943.57	
	100007	4	S	999.00	200004

Features

In this type of program the records are accessed with SQL. The template also supports (by default) clickable column headings, allowing you to re-sort your data by any of your fields. The wizard also allows you to include filter fields above the list.

Figure 1-50 The WebSmart template selection window

Although the templates produce fully functioning web applications with no coding on your part, you can also customize them in the WebSmart IDE to create any type of web application. WebSmart ILE uses a language that is called PML, which is easy for RPG programmers to understand. If you are using WebSmart PHP, you can customize the open source PHP code or plug in free PHP scripts. The WebSmart IDE also includes productivity features such as an HTML editor, code completion, and a debugger.

WebSmart ILE creates an ILE RPG program object and WebSmart PHP creates a PHP file. The result is a stateless web application that works in the same way as other web applications that users are familiar with. This result means that the browser's “back” and “refresh” functions work like any other web application. WebSmart web applications are also easily scalable.

In addition to the WebSmart IDE, there are two server components:

- A code generator that produces the server-side programs for WebSmart ILE only. With WebSmart PHP, PHP files are created on the PC, then transferred to a server with FTP.
- A web application server that facilitates running the generated WebSmart ILE programs. WebSmart PHP programs run with Zend Server and are hosted on IBM i.

The three WebSmart components work seamlessly together to help you code and produce web applications without having to master the complexities of coding web applications by hand.

WebSmart runtime architecture

WebSmart web applications are delivered to the user by the following mechanisms (Figure 1-51):

- The user requests a URL on an IBM i server from a browser (PC).
- The free IBM HTTP web server (powered by Apache) accepts the request and processes it.
- The WebSmart program is evoked by the web server.
- The web application server is used with WebSmart ILE to provide all the supporting functions for the programs to interact with a browser, such as session information handling and library lists. For WebSmart PHP, Zend Server is involved in this process.
- If the program is designed for database file access, it establishes a connection to the target database.
- The WebSmart program reads input parameters from the user request (by using the web server).
- The WebSmart program constructs a web page by merging static pieces of HTML with live data from the target database and writes the page to the web server.
- The web server sends the web page to the user through the browser.

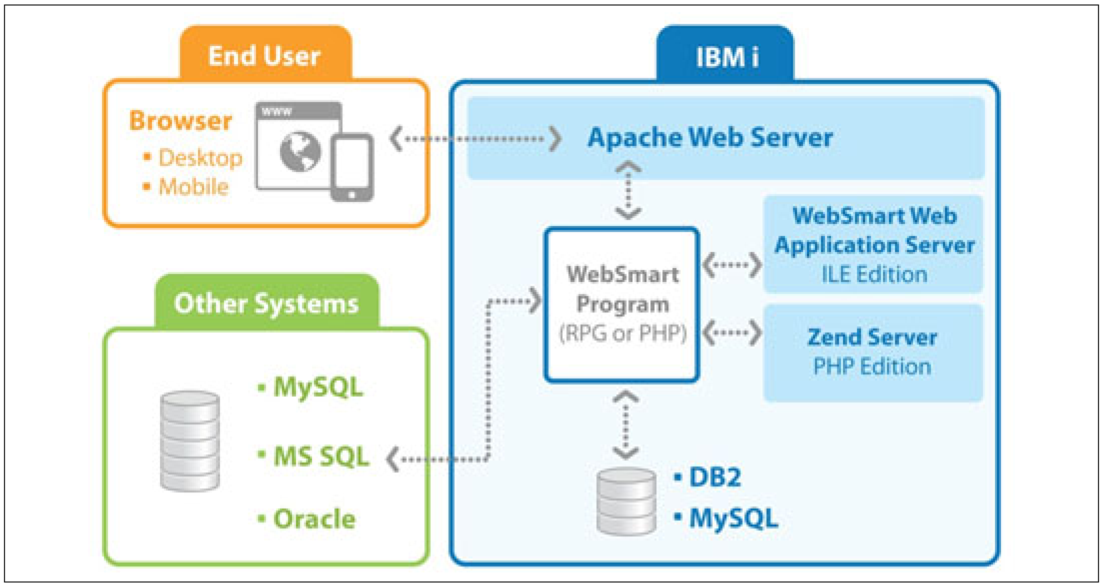


Figure 1-51 WebSmart ILE runtime model

WebSmart Mobile

WebSmart Mobile is included in WebSmart ILE and PHP, and consists of mobile templates and IDE features that are designed to provide quick results even if you have no mobile development experience. Using WebSmart Mobile's templates, you can create web and hybrid applications that have touch gestures and automatically resize when you access them from different-sized screens or rotate your device. The templates use the jQuery Mobile framework, which is an easily theme-able, touch-optimized web framework for tablets and smartphones. For more information about the jQuery Mobile framework, see:

`http://jquerymobile.com/`

You can then customize your mobile web applications in the WebSmart IDE. It includes point-and-click code snippets so you can quickly insert new mobile UI widgets, including inline buttons, check boxes, radio buttons, search, sliders, toggle switches, and navigation bars. The HTML5 snippet inserts mobile-optimized input fields, such as email address, telephone number, or URL, that automatically open the correct keyboards, and date or color inputs that open date pickers or color palettes.

Figure 1-52 shows Mobile web applications developed with WebSmart.

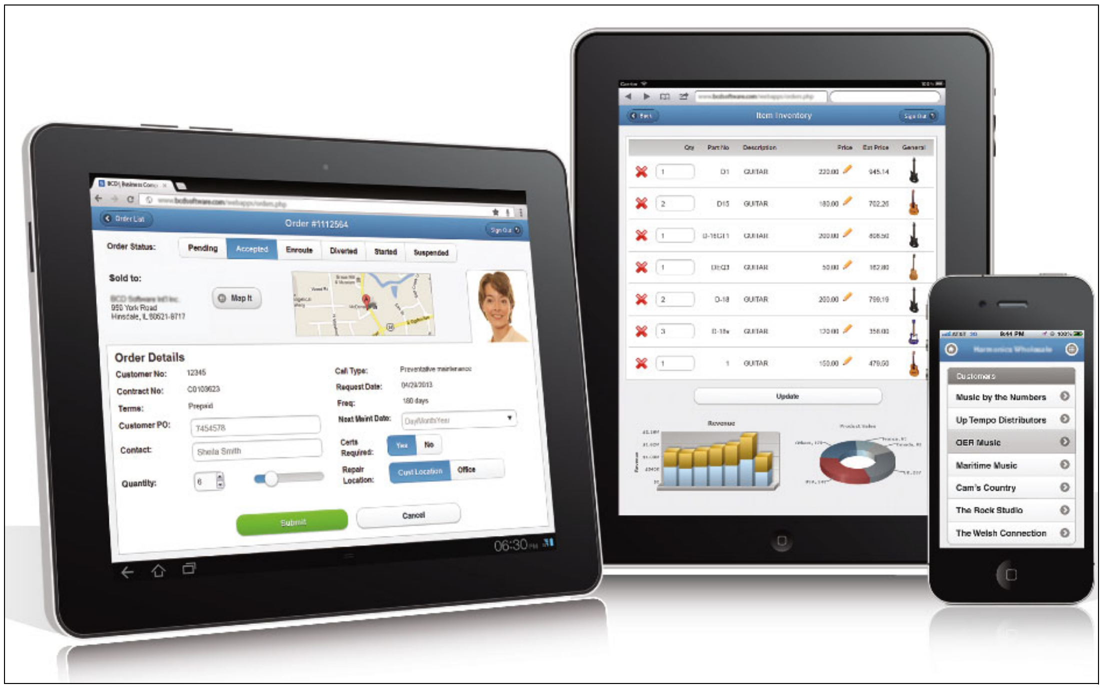


Figure 1-52 Mobile web applications that are developed with WebSmart

WebSmart Mobile also provides the flexibility for you to integrate with other technologies like Phonegap to take advantage of the device's hardware features. This integration can include accessing the camera to take photos or scan a bar code, or using the GPS for geolocation.

Features

WebSmart has many features that are designed to boost your productivity as you write desktop or mobile applications.

Intelligent templates

WebSmart includes templates that guide you through creating dynamic desktop or mobile web applications that access IBM i and multi-platform data. The templates generate the initial RPG or PHP, HTML, CSS, and JavaScript so you start with a fully functional desktop or mobile web application even if you have no prior web development experience. You can then use the WebSmart IDE to create web applications for your business needs.

You can choose from several different functional templates such as logins and grids, which use SQL or RLA. WebSmart PHP includes both procedural and object-oriented templates.

Customizable template themes

WebSmart's four customizable template themes allow you to “skin” a web application with your corporate identity, including logo, graphics, and colors. The themes are based on jQuery UI, which makes them more extensible as you adapt them to your own design needs. You can also easily customize the appearance of the new template themes by using ThemeRoller.

For more information about the jQuery UI, see the following link:

http://jqueryui.com/

For more information about ThemeRoller, see:

http://jqueryui.com/themeroller/

Browser-based access

Applications that are developed with WebSmart are easy to deploy because users need only a browser to access them on PCs, Macs, tablets, and smartphones. There is no need to install any additional software or deploy mobile applications from an app store.

Code editor

The WebSmart IDE includes many features that increase programming speed, including a syntax checker, code completion, drag-and-drop fields, and an interactive debugger. WebSmart ILE and PHP share design infrastructure and function. As a developer who is using WebSmart, you can move seamlessly between writing ILE and PHP applications.

Text-based and visual HTML design tools

WebSmart simplifies HTML coding and web page design with visual and text-based tools. The WebSmart IDE includes wizards to add UI elements, drag-and-drop fields, and color coding. It also integrates with third-party visual editors like Adobe Dreamweaver.

Use RPG and your IBM i

WebSmart was built with RPG programmers in mind and helps them succeed in the world of web development. WebSmart ILE programs generate ILE RPG and take advantage of RPG IV and ILE features, such as bound modules, service programs, subprocedures, and free-format RPG. The WebSmart IDE also includes wizards to call back-end RPG programs.

WebSmart PHP includes a SmartSnippet to call RPG or COBOL programs. It uses the new open source PHP toolkit for IBM i, which requires less code to call a program.

Separation of client-side code from logic

In WebSmart, the client-side code (HTML, CSS, JavaScript) is separated from, instead of intermingled with, the logic (RPG or PHP). This approach has these advantages:

- The code is easier to read and is similar to how RPG and DDS are separated.
- It is simpler for people with different skill sets (developers and designers) to work on the same program.
- It is easier to reuse RPG or PHP and HTML code snippets.

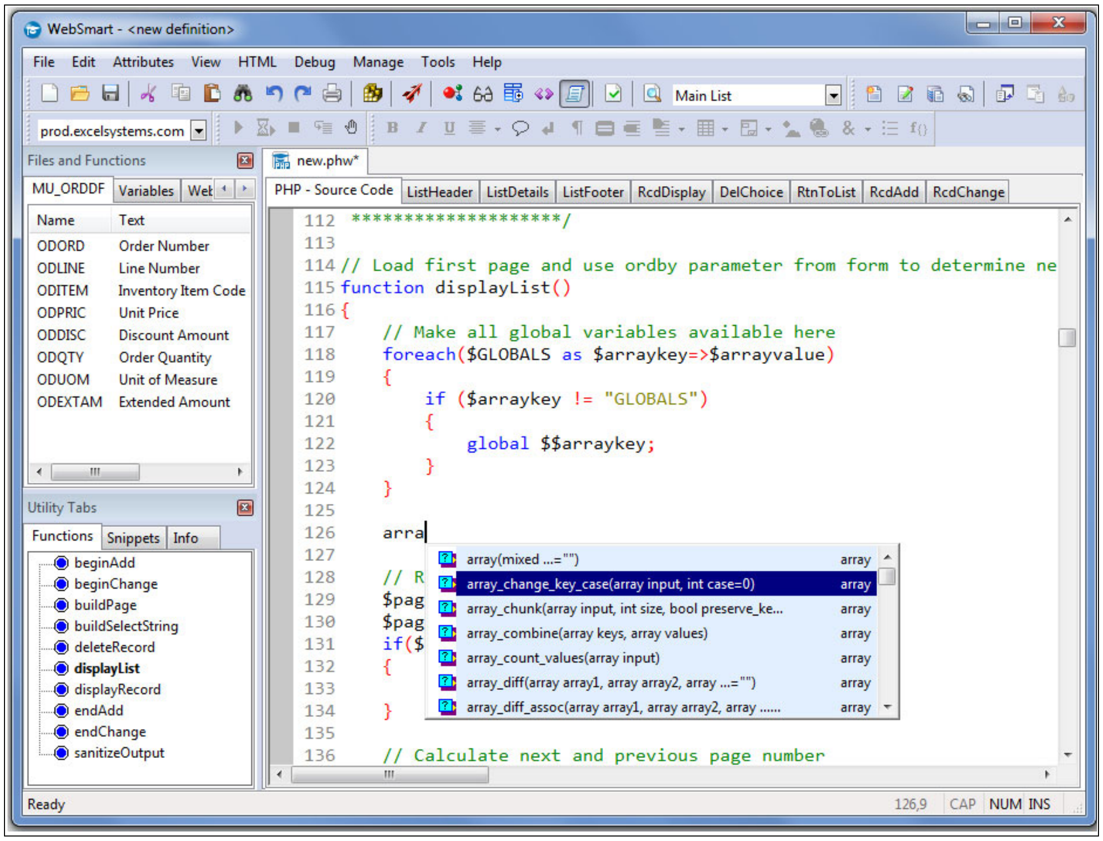
Central repository

WebSmart has a central repository where you can define business, database, and presentation rules at the database level, effectively providing powerful data modeling capabilities. When extensions are defined in the repository, the templates can use them.

5,000+ PHP functions are integrated into the IDE

In WebSmart PHP, the 5000+ related PHP functions are organized into tabs and integrated into the WebSmart IDE development environment with real-time PHP syntax checking, prompting, color-coding, and formatting.

As you add your PHP functions, WebSmart PHP shows you a prototype of the function along with tooltips to explain any parameters that are used by the function. The IDE also supports context-sensitive help for any function. All of these features help reduce errors in your code, even before you start testing. See Figure 1-53.



Name	Text
ODORD	Order Number
ODLINE	Line Number
ODITEM	Inventory Item Code
ODPRIC	Unit Price
ODDISC	Discount Amount
ODQTY	Order Quantity
ODUOM	Unit of Measure
ODEXTAM	Extended Amount

- beginAdd
- beginChange
- buildPage
- buildSelectString
- deleteRecord
- **displayList**
- displayRecord
- endAdd
- endChange
- sanitizeOutput

```
********************/

// Load first page and use ordby parameter from form to determine ne
function displayList()
{
    // Make all global variables available here
    foreach($GLOBALS as $arraykey=>$arrayvalue)
    {
        if ($arraykey != "GLOBALS")
        {
            global $$arraykey;
        }
    }

    arra

    // R
    $pag
    $pag
    if($
    {

    }

    // Calculate next and previous page number
```

```
array(mixed ...="")                                  array
array_change_key_case(array input, int case=0)       array
array_chunk(array input, int size, bool preserve_ke...  array
array_combine(array keys, array values)              array
array_count_values(array input)                      array
array_diff(array array1, array array2, array ...="")  array
array_diff_assoc(array array1, array array2, array ......  array
```

```
Ready                                    126,9   CAP  NUM  INS
```

Figure 1-53   WebSmart IDE displaying PHP functions and code completion

Strong integration with DB2

The WebSmart IDE displays DB2 file (and other supported database) attributes, field lists, and file keys. This is a great help as you write your code and create joins. If you are creating applications to run on the IBM i, WebSmart has functions that help you use library lists at runtime.

SmartCharts

WebSmart includes SmartCharts so that you can design and integrate real-time animated two and three-dimensional bar charts, pie charts, executive dashboards, and more in your web applications.

Built-in FTP client

The WebSmart IDE comes with its own integrated FTP client that you can use to connect to multiple servers, IBM i or otherwise. You upload or download files through the folder tree interface, which makes it easy to manage development efforts across multiple servers, platforms, and databases.

Integrated suite of projects

You can optionally combine WebSmart with other BCD tools to further modernize your IBM i applications and processes by building your own customized and integrated solution. Start with one tool, a few tools, or the complete BCD ClearPath modernization suite to meet your needs. For more information, see:

http://www.bcdsoftware.com/iseries400solutions/clearpath/

Clover, a real-time web report generator, is both a stand-alone solution and an integrated add-on component to WebSmart. You can work seamlessly with the reporting features of Clover and the web development features of WebSmart because they share an IDE.

For more information about Clover, see:

http://www.bcdsoftware.com/iseries400solutions/clover/

WebSmart optionally includes the BCD Nexus Portal, making it even easier to secure your web applications. Nexus organizes your web applications in menus so users can quickly access them. It also provides a secure single point of access to other web applications, documents, and dashboards.

For more information about Nexus Portal, see:

http://www.bcdsoftware.com/iseries400solutions/nexusportal/

Presto is the BCD web enablement and modernization tool for existing green screens and new RPG applications. You can link your Presto web-enabled screens to your WebSmart web applications. You can also use WebSmart to add more sophisticated function to your Presto screens, including emailing, uploading files, and processing web services.

For information about Presto, see:

http://www.bcdsoftware.com/iseries400solutions/presto/

Security

Web applications that are developed with WebSmart run the same way as any other web application. Therefore, the same security considerations apply. These things are some of the areas you can address to ensure that your web applications are secure:

- Data transmission between client (browser) and server. WebSmart supports SSL to secure data transmission. SSL encrypts data before transmission, sends it encrypted, and then decrypts it. All major commercial B2C sites use SSL.
- Free IBM HTTP web server (powered by Apache). The Apache configuration ensures that all users are restricted to only the areas (libraries and programs, IFS directories, and files) that you explicitly open for access. You can also impose authorization schemes on these resources, requiring users to type a valid user ID and password before they are allowed to proceed further.
- User profile security. WebSmart includes functions for authenticating against IBM i user profiles and passwords. You can also validate against database files or validation lists.
- Data encryption. You can encrypt and decrypt data with 128-bit AES encryption.
- Session management. WebSmart includes session management features with unique session IDs to eliminate the possibility of hacking URLs to create and expire sessions, and to make parameter passing easier.
- XSS and SQL injection. WebSmart includes functions to encode and sanitize data from malicious users.

Comparing WebSmart ILE and PHP

This section contains some considerations to take into account when you choose WebSmart ILE, WebSmart PHP, or both editions. Although the PML and PHP languages cannot be combined in a single program, there are no limits to combining PHP and ILE programs within an application.

You can transparently move from PHP to ILE programs by using Ajax, links, forms, menus, or redirects, taking advantage of the strengths of both environments. You can also work with both PHP and ILE definitions at the same time, in the same IDE session.

Table 1-2 shows a comparison of WebSmart ILE and WebSmart PHP.

Table 1-2 Comparison of WebSmart ILE and WebSmart PHP

Consideration	WebSmart ILE	WebSmart PHP
Server language	Programming is done in PML, which generates RPG ILE objects.	Programming is done in open source PHP. The result is a PHP file.
Databases	DB2/400, MySQL, and MS SQL	DB2/400, MySQL, and Oracle
Server platforms	IBM i	IBM i, Linux, UNIX, or Windows. (Runs with Zend Server if you are hosting on IBM i).
Performance and scalability	Both editions create stateless web applications that run in batch and are easy to scale.	
Runtime object requirements	WebSmart Web Application Server	Zend Server installed on the IBM i
Object oriented	Uses RPG ILE programming model	Includes procedural and OO templates
Mobile support and templates	Yes	Yes

Resources

For more information, see the following resources:

- Data sheets
  - WebSmart ILE

    http://www.bcdsoftware.com/iseries400solutions/websmart/PDFs/websmart-ile-brochure.pdf
  - WebSmart PHP

    http://www.bcdsoftware.com/iseries400solutions/websmart-php/PDFs/websmart-php-brochure.pdf
- WebSmart Mobile

  http://www.bcdsoftware.com/iseries400solutions/websmart/editions/websmart-mobile.htm
- WebSmart PHP white paper

  http://www.bcdsoftware.com/iseries400solutions/websmart/phpinforequest.php
- Webinars

  http://www.bcdsoftware.com/ibmiwebinars/ondemand/

Supported Platforms

WebSmart ILE requires at least V5R4 of IBM i.

WebSmart PHP requires Apache and PHP. IBM i is not required for WebSmart PHP. If you are using IBM i, at least V5R4 is required and Zend Server must be installed.

Download and ordering information

For download and ordering information, see the following link:

`http://www.bcdsoftware.com/websmartfaster`

1.4 CNX Corporation: Valence Web Application Framework for IBM i

Running on the native Apache Server, Valence provides a framework for new web and mobile application development on IBM i. Valence uses standard ILE RPG to handle business and database logic on the server, connecting to an extensive JavaScript/HTML5 framework to handle the user interface on the browser or mobile device. Developers use standard Ajax techniques to send and receive data between the browser and their RPG programs, applying a programming paradigm that is common to web applications across many platforms today.

The Valence framework includes:

- Ext JS. A collection of highly configurable components for the browser (such as forms, grids, charts, graphs, trees, and toolbars) that can be set up to send data to and receive data from RPG programs
- Sencha Touch. Functionally similar to Ext JS, but with finger-friendly components that are designed explicitly for mobile devices
- Valence Portal. Provides a configurable browser page for users to log in to your IBM i system and launch applications. The Portal includes built in security and an icon-based menu that serves as an effective GUI replacement for 5250-based menus.
- Valence RPG Toolkit. A collection of procedures that can be called from within any ILE RPG program to facilitate communications with the browser or mobile device that is running Valence.
- Valence Nitro Utilities. Powerful tools for creating graphical dashboards and queries, editing JavaScript source, browsing IBM i database files, and creating applications with the Sencha Architect IDE.

Figure 1-54 shows the login window for the Valence portal.

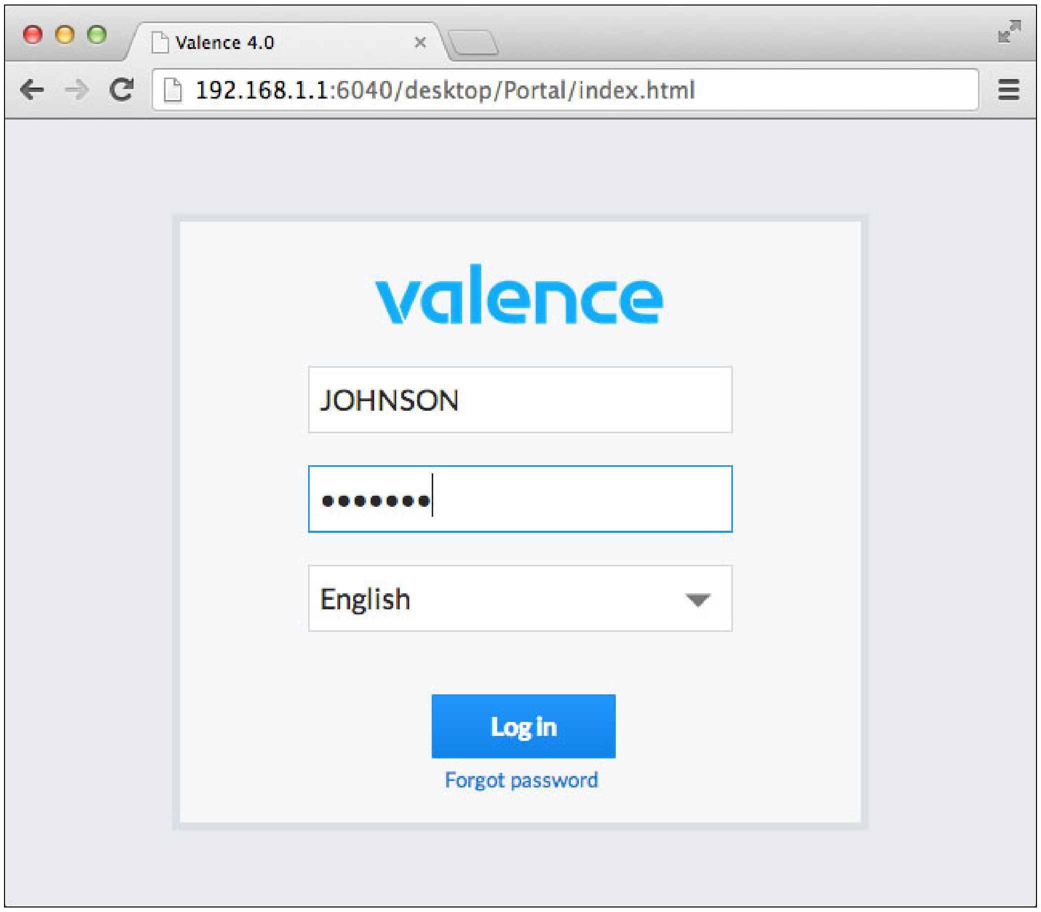


Figure 1-54 Valence portal login

Figure 1-55 shows the Valence Portal App Launcher.

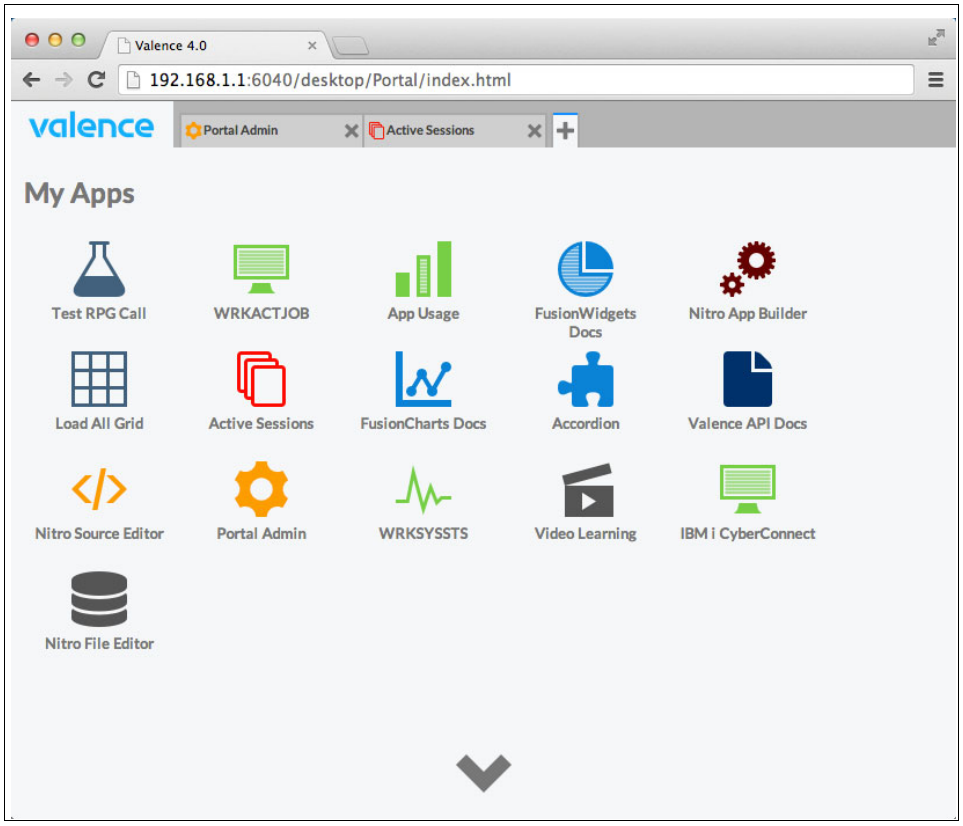


Figure 1-55 Valence Portal App Launcher

The case for learning JavaScript frameworks to provide a UI for RPG

A developer building IBM i-based web and mobile apps has many possible development paths to take. There is no shortage of different front-end and back-end technologies vying for your attention on this front.

The Valence solution assumes that it is best to start with what you know and work from there. An experienced RPG programmer who knows their existing business data and system logic is usually best served by technologies that complement, instead of replace, their RPG skills. In other words, there is less value to spending time learning new server-side programming techniques if RPG programs can already be made to effectively “talk” to the browser. It is better to spend time mastering the browser-side development tools and languages, which are critical to know regardless of the back-end language or platform that is being used.

Valence includes tools that enable RPG to seamlessly communicate with the browser by converting RPG output into JavaScript Object Notation (JSON), the native language of JavaScript, and vice versa. Using these tools, an experienced RPG developer can quickly create back-end programs that send data to, and receive data from, front-end Ext JS or Sencha Touch components. This allows them to focus their time on learning the front-end technology and hone skills that are most pertinent to web and mobile app development.

The core components of the Valence front end are Ext JS and Sencha Touch. These are frameworks that are used by millions of developers around the world to extend HTML5 concepts and rapidly produce web and mobile apps with state-of-the-art user interfaces. JavaScript frameworks make it possible to create applications with highly sophisticated UI elements such as charts, grids, forms, trees, and so on. These components are the kinds of components users have come to expect in Web 2.0-style applications regardless of platform. Learning the basics of JavaScript development and the APIs behind frameworks such as Ext JS should be a high priority for any IBM i business application developer who is looking to interface with the browser.

Understanding Valence versus green screen

Before learning about the Valence application development fundamentals, it is important to have a high-level understanding of the basic differences between Ajax-based web applications (Valence) and conventional interactive 5250 character-based interface ("green screen") programs.

First, consider what is similar. In a standard green screen program, the RPG program does not have any direct connection to what the user sees on the screen. That is, the DDS file contains the code that controls where fields are located, how they appear, which function keys can be pressed, and so on. Likewise, in Valence applications, the RPG program does not have any direct control over what the user sees on the browser page. Instead, JavaScript logic on the front end controls the HTML that determines what the users see and what actions they can take.

But that is where the similarities end. Unlike DDS files, your front-end JavaScript has a certain degree of autonomous control over many window elements without any direction from the back end. For example, fields can move around or have their display properties adjusted, items can appear or disappear, and so forth, This capability means that developers can put fairly sophisticated interface-altering logic into the front-end code that limits the number of times the browser program must call a back-end RPG program for instructions. Generally speaking, the Valence front end needs to call the back end only when it needs to run server-side business logic, retrieve data to present to the user, or save data to the IBM i database. Everything involving presentation of data on the window is controlled entirely by the front end.

Another fundamental difference between green screen programs and Valence applications is the concept of user jobs on IBM i. In the interactive green screen world, upon logging in each user is given a dedicated job that you can see as a separate listing in WRKACTJOB. This is not the case when a user logs in to Valence to run apps. A user is only "active" during the split second it takes to service a response to a click or event that they make on the web page. In WRKACTJOB, you can see a number of CGI jobs in the QHTTPSVR subsystem, each of which can service an unlimited number of users who are running apps in the Valence Portal. These CGI jobs are technically not much different from batch jobs that are running RPG programs to generate reports.

Finally, Valence apps, and JavaScript programs in general, operate in an asynchronous fashion. In a green screen application, the entire page is written in chunks (record formats) by a single RPG program, typically concluded with an EXFMT operation. At this point the program "freezes" and waits for the user to do something. A Valence application, by contrast, uses an asynchronous approach in which any page can have many independent user interface components, each populated with data that might come from separate RPG programs or procedures. The user can take an action that results in one component being updated while others remain unchanged, a concept commonly referred to as Ajax. At no point in the process is there a back-end RPG program that is waiting on anything to happen in the interface. Instead, it is the user's actions in the browser, typically the act of clicking something, that dictates what happens next.

These Valence UI concepts might seem a bit unfamiliar at first, but they are normal on every platform that is not based on a 5250 underpinning. Just about any "shopping cart" type website is a good example of these principles in action, and RPG is capable of functioning in this manner as well. It boils down to approaching the design and development of the front end with a different mindset. The back-end development follows naturally.

Rethinking the user interface

If your company has been running on IBM i, or any of its predecessors of various names, for any significant length of time, you probably have many years of experience building 5250 applications in RPG.

However, this situation is where IBM i developers often get stuck when they are moving into the Web 2.0 world of Valence. When it is time to develop new applications using Ext JS, do not think of the user interface in 5250 terms. Here are a few key points to keep in mind when you are creating Valence apps on IBM i:

1. Do not limit yourself to a one-for-one program replacement mentality.

   Your users are likely accustomed to calling a separate 5250 program for nearly every function they do on the system. It is probably second nature to them to go to a menu and run program “A” to do one task, then exit the program (or flip to another 5250 session) and run program “B” to do another task.

   This is contrary to the norm in the Web 2.0 world, where a single application is typically built to handle many related tasks. GUI features make it possible to seamlessly and elegantly bundle many related tasks into a single app. For example, in an inventory application you might bundle material movement, order picking, location look-ups, and putaways into a single application to enable your users to jump from one related task to another without leaving the application.

   Therefore, do not limit yourself to thinking your new Valence apps are one-for-one replacements for your existing green screen programs. When you think creatively, you might find ways to make certain processes much more efficient for your users.

2. Avoid the multi-5250 panel mindset.

   When your green screen programs need to present or obtain more data than can fit on a 24x80 (or 27 x 132) text panel, you are often forced to break your applications into separate screens or record formats and use function keys or other methods to move users to the next page of data. Users and developers might come to think of certain data entry processes as inherently multi-page or multi-screen operations because they are accustomed to entering the data in that manner.

   This multi-panel approach is no longer necessary when you are creating browser-based applications. The data can be presented in a vertically scrolling page, multiple tabs, or one of many other display methods that are available through Ext JS. When you are designing a new application, avoid forcing users to take an action every time they use the application to complete their data. For example, if your current 5250 program consists of four screens, and the user almost always must interact with three of them to complete a common task, consider making the data from all three of those screens into one page in your new application. Make data from the lesser-used panel obtained by clicking a button or tab. This design leads to less work for your users to get the particular task completed, in addition to making a cleaner interface.

3. Less is more.

   Developers often tend to think of a computer screen as a “canvas” to be filled with as much data as possible for the user to work with. In the 5250 world, this often leads to clever abbreviations and truncated fields to squeeze as much as possible into the page. While your seasoned users might be accustomed to such busy interfaces, users who are accustomed to the web world have a hard time adapting to these displays.

   When you are embarking on a project to create a new Valence application as a replacement for a green screen program (or programs), take a moment to rethink what is being presented to the user. Quantify what information is truly needed to accomplish the tasks at hand. You have the freedom to include everything that is on the green screen and much more in your Valence application. However, consider that some of the information might be superfluous to the task at hand. Each element that you add to the page is another item that users need to mentally accommodate,. Starting fresh, you have the opportunity to create much cleaner interfaces that list just what the user needs. The “80-20” rule is a good approach to consider: include fields that accommodate 80 percent of the users, and put the extra fields in a separate section or tab that they can easily access when needed.

Screens that contain fewer fields through which users must navigate are easier to implement, easier to understand, and ultimately easier to train new users on. If possible, have a user interface expert help design elements of the user interface. Typically a UI expert is also versed in styling elements (CSS) that can be put to use to make the new pages more effective for the users. The resulting applications can be visually stunning, particularly when compared to their green screen predecessors.

4. Empower your users to customize the pages to their liking, particularly with grids.

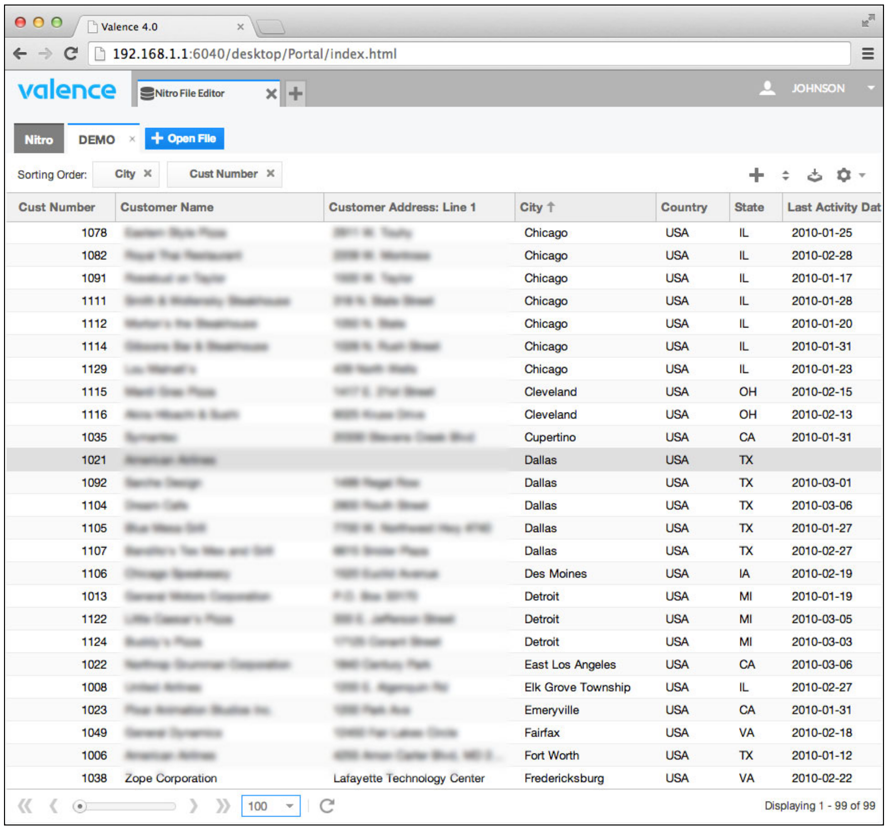
   Subfiles are naturally pervasive in the 5250 world, serving business users' need to work with lists of data. The equivalent in the Valence world, the Ext JS grid, is likewise going to be used heavily in your new applications. This grid can help you deliver features that users love with minimal programming effort on your part.

   When you are setting up the back end RPG for a grid, try to retrieve and send all the data that a user might need to see in the list to efficiently handle their task. You are not limited to only the fields that were in the former subfile. For example, if you are including a product number in the grid, why not also include the product description and other potentially useful information as well (item type, unit of measure, and so on). However, keeping the “less is more” directive in mind, you might want to make these extra fields set to “hidden” by default, so users who need them can turn them on using the grid's column controls. Users can quickly become enamored of their new ability to resequence the grid columns into an order that best suits them, another feature inherent to the UI component (no programming is required).

   Every grid can be made to be sortable by clicking any column heading, a feature that web users come to expect in any list. Unless your grid is of the “load all” variety, in which case this sort feature is automatic, it is going to be your RPG program's responsibility to handle the sort. This process is where SQL becomes helpful because dynamic sorting is one of the specialties of SQL. For this reason, try to avoid relying on native I/O (SETLL and READ/READE) to load the grids as you might have done in the past with subfiles. Take some time to master embedded SQL and stored functions or procedures so that all the work can be done within the SQL select statement. This time is helpful, particularly whenever a new column is needed. Adding a column is typically just a matter of adding an extra field to your SQL select statement, perhaps along with a join file.

   Also, be sure to make your grids stateful. This is a configuration property that, when activated in your code, automatically places user customizations to field sequence, field size, and so on, in a browser cookie or session storage. It reapplies them when they return to the application on another day. Such features help to make the transition from green screen to web (the “buy-in” for your users) that much more successful.

Figure 1-56 shows an example of user-customized columns.



Cust Number	Customer Name	Customer Address: Line 1	City	Country	State	Last Activity Dat
1078	[redacted]	[redacted]	Chicago	USA	IL	2010-01-25
1082	[redacted]	[redacted]	Chicago	USA	IL	2010-02-28
1091	[redacted]	[redacted]	Chicago	USA	IL	2010-01-17
1111	[redacted]	[redacted]	Chicago	USA	IL	2010-01-28
1112	[redacted]	[redacted]	Chicago	USA	IL	2010-01-20
1114	[redacted]	[redacted]	Chicago	USA	IL	2010-01-31
1129	[redacted]	[redacted]	Chicago	USA	IL	2010-01-23
1115	[redacted]	[redacted]	Cleveland	USA	OH	2010-02-15
1116	[redacted]	[redacted]	Cleveland	USA	OH	2010-02-13
1035	[redacted]	[redacted]	Cupertino	USA	CA	2010-01-31
1021	[redacted]		Dallas	USA	TX	
1092	[redacted]	[redacted]	Dallas	USA	TX	2010-03-01
1104	[redacted]	[redacted]	Dallas	USA	TX	2010-03-06
1105	[redacted]	[redacted]	Dallas	USA	TX	2010-01-27
1107	[redacted]	[redacted]	Dallas	USA	TX	2010-02-27
1106	[redacted]	[redacted]	Des Moines	USA	IA	2010-02-19
1013	[redacted]	[redacted]	Detroit	USA	MI	2010-01-19
1122	[redacted]	[redacted]	Detroit	USA	MI	2010-03-05
1124	[redacted]	[redacted]	Detroit	USA	MI	2010-03-03
1022	[redacted]	[redacted]	East Los Angeles	USA	CA	2010-03-06
1008	[redacted]	[redacted]	Elk Grove Township	USA	IL	2010-02-27
1023	[redacted]	[redacted]	Emeryville	USA	CA	2010-01-31
1049	[redacted]	[redacted]	Fairfax	USA	VA	2010-02-18
1006	[redacted]	[redacted]	Fort Worth	USA	TX	2010-01-12
1038	Zope Corporation	Lafayette Technology Center	Fredericksburg	USA	VA	2010-02-22

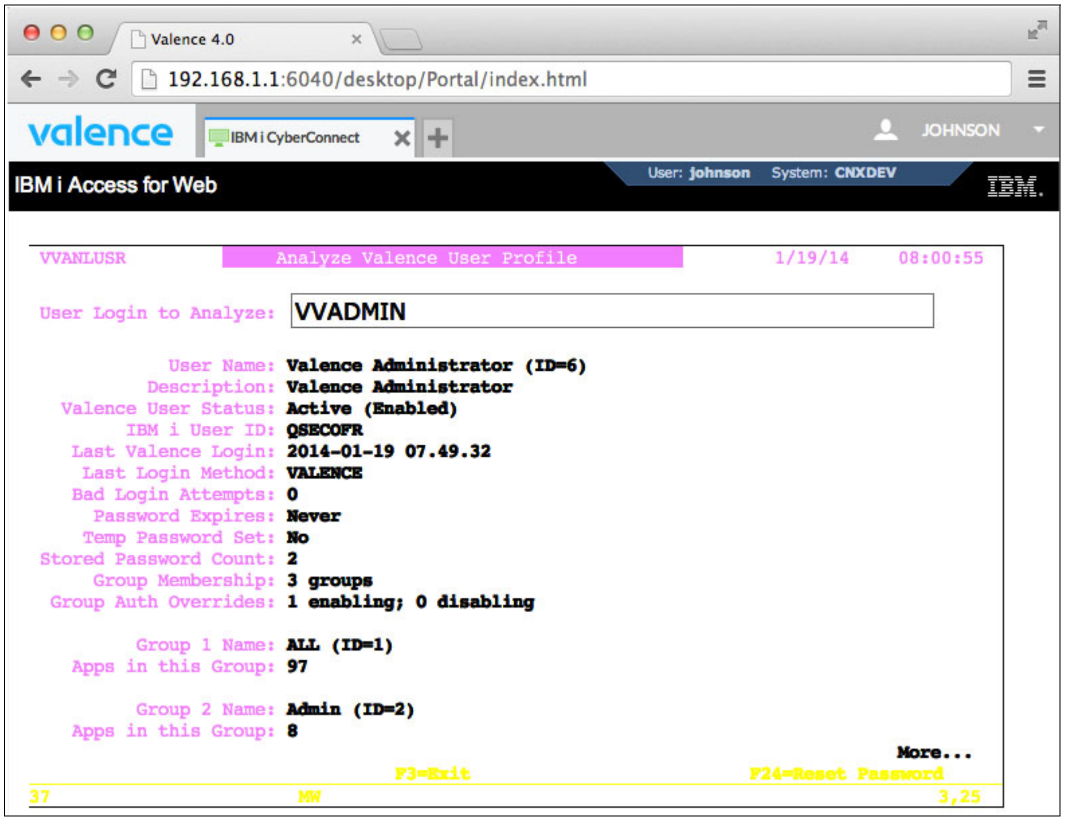
Figure 1-56 Nitro File Editor Grid App (example of user-customized columns)

Accommodating existing 5250 programs

As mentioned previously, most established IBM i shops have a large collection of 5250-based applications. These often-complex interactive programs cannot be rewritten or retrofitted to work in the web overnight. In such cases, introducing new web applications to an IBM i shop means running a mix of new browser-based applications alongside a number of heritage green screen programs for quite some time. This used to mean that users had to have one or more separate Client Access emulation sessions open, flipping between those sessions and the web browser throughout the day. This process can be cumbersome to users and hamper acceptance of new web solutions, no matter how much of a UI improvement the new applications might provide.

IBM i Access for Web offers a solution to this problem by making it possible to run interactive green screen programs within the browser. Beginning with Valence 4.0, IBM i Access for Web support is included within the Valence Portal, so developers can create Valence Apps that are links to existing interactive green screen programs. Users start the wanted green screen application within the Portal. They run the same program in the browser as they ran in Client Access and do not need to leave the Portal in the process. Developers and users can set up macros for these green screen applications so the user is automatically placed into a specific point of the program. A green screen program can also be called with special parameters to indicate that it is being run from the browser, in case the developer wants to alter the program's behavior. It is also possible to mix green screen applications with Valence applications, so that a click or action in a Valence application can trigger a separate green screen application to open with information specific to that action.

Figure 1-57 shows an example of running a 5250 application using IBM i Access.


Valence 4.0
192.168.1.1:6040/desktop/Portal/index.html
valence
IBM i CyberConnect
JOHNSON
IBM i Access for Web
User: johnson System: CNXDEV
IBM.

```
VVANLUSR            Analyze Valence User Profile              1/19/14     08:00:55

 User Login to Analyze: VVADMIN

             User Name: Valence Administrator (ID=6)
           Description: Valence Administrator
   Valence User Status: Active (Enabled)
        IBM i User ID: QSECOFR
    Last Valence Login: 2014-01-19 07.49.32
     Last Login Method: VALENCE
    Bad Login Attempts: 0
      Password Expires: Never
     Temp Password Set: No
 Stored Password Count: 2
      Group Membership: 3 groups
  Group Auth Overrides: 1 enabling; 0 disabling

          Group 1 Name: ALL (ID=1)
    Apps in this Group: 97

          Group 2 Name: Admin (ID=2)
    Apps in this Group: 8
                                                                     More...
                                F3=Exit                  F24=Reset Password
37                        MW                                           3,25
```

Figure 1-57 Running a 5250 application in Valence by using IBM i Access

Back-end RPG Development in Valence

Syntactically, RPG development in Valence is not much different from any other RPG programming, except when it comes to the logic that is used to interact with the user. Instead of writing subfile records or window fields followed by an EXFMT, you are writing to arrays and data structures followed by a "Send to Browser" procedure call.

When you are developing the RPG side of a Valence app, there are two key points for programmers to remember:

1. There can be no interactive display logic included in the RPG program.
2. The RPG program is not stateful; it runs in batch and ends after the page is built or the user request is served.

Both of these points fall under the principle of separating business logic from display logic, a concept that can be unfamiliar at first to developers who are accustomed to building interactive RPG programs. Unlike traditional RPG with interwoven 5250 display file logic, programs that talk to Valence applications are more akin to programs that run in batch. They do not sit and wait for a user to take an action. Instead, RPG programs that are serving Valence applications accept all the input parameters that they need, do something (or a series of things), send data back to the browser, and then end.

The process of moving data between the browser and the RPG program is where things most noticeably digress from the 5250 world. Valence handles this process by using its included RPG Toolkit, an IBM i service program. When this service program is bound to an RPG program, it activates a collection of special procedures that the developer can use to interact with the browser. Think of this collection as a bunch of functions that provide the browser equivalent to READ and WRITE for display files, along with many more functions. These routines are automatically made available to your RPG programs through use of some simple copy source in your D specs.

All of the Valence Service Program procedures or APIs are documented here. These procedures include utility functions for using RPG to send emails, creating PDFs, working with IFS data, and much more, but the most important routines are the routines that are used to communicate with the browser front end. These routines are variants of procedures that are called VVIN and VVOUT (for consistency of identification, all Valence procedures and objects begin with “VV”).

A typical Valence RPG program begins by doing a “VVIN” to retrieve data from the browser. Say, for example, that you have an inventory inquiry application that is defined in the Portal to call an RPG program called INV100. The first thing INV100 is likely to do is to determine what information the user wants to see. Do they want a list of warehouses? Or a list of inventory for a particular item? Accomplish this goal by pulling in the value of a character field set in the browser front-end code. A common name for such a field is “action” because it is telling the RPG program what action to run.

To get this “action” value from the browser, use the Valence vvIn_Char procedure. The free form RPG code looks like something like Example 1-3.

Example 1-3 Free form RPG code

```
// determine what the user wants to do
action = vvIn_Char('action');

If action = 'list_warehouses';
  exsr listWhses;
elseif action = 'list_products';
  exsr listProds;
elseif action = 'list_inventory';
  exsr listInv;
endif;

*inlr=*on;
```

For sake of example, say the action received by the RPG program is to provide a list of inventory to the user. The program then calls a “list inventory” subroutine (or better still, a procedure) that pulls in the inventory for a particular warehouse and returns the list to the user.

In a traditional green screen program, you might accomplish this inventory list by using native I/O (reads) against one or more physical or logical files and loading up a subfile, one record at a time. You might use the same approach to send data to the browser, only you are loading a data structure array one record at a time, then sending the loaded array to the browser. That RPG code might look something like Example 1-4.

Example 1-4 Loading a data structure and sending the array to the browser

```
begsr listInv;
// load up array with inventory data & send to browser
whse = vvIn_Char('warehouse');
setll whse lotbinPF;
reade whse lotbinPF;
dow not %eof;
```

```
  x+=1;
  invDS(x).product  = prdno;
  invDS(x).location = aisle+row+tier;
  invDS(x).quantity = onhand-issued+recvd+adjustd;
  reade whse lotbinPF;
enddo;
vvOut.rootname='inventoryGrid';
vvOut.object  ='invDS';
vvOut_toJSON(vvOut:%addr(invDS));
endsr;
```

In Example 1-4 on page 57, a data structure array, which is defined from an external file object or record format that is called invDS, is loaded. Then the vvOut_toJSON service program procedure is called to send the array to the browser in JSON format, the "native tongue" for JavaScript. The "rootname" is the arbitrary name that is given to the data set, which the front end is looking for to match up to a grid. The "object" is just a reference to the file object so the service program knows how the array is constructed.

If you are comfortable with SQL, a better way to accomplish the same thing is to create an SQL select statement and route the executed results to the browser in JSON format, as shown in Example 1-5.

Example 1-5 Creating an SQL statement and routing the results to the browser in JSON format

```
begsr listInv;
// create SQL stmt to obtain inventory data & send to browser
whse = vvIn_Char('warehouse');
stmt = 'select prdno, aisle||row||tier as location, '+
              'onhand-issued+recvd+adjustd as quantity'+
       ' from lotbinPF '+
       'where house='''+whse+''' '+
       'order by aisle, row, tier';
vvOut.rootname='inventoryGrid';
vvOut_execSQLtoJSON(vvOut:stmt);
endsr;
```

In both cases, your result might be mimicking what was done in an existing green screen program. This situation is where you as a Valence developer need to stop and think creatively. What can you do to make this application even better for the users? You might want to add more columns that did not fit within the confines of the green screen, such as product description, last activity date, and so on. In a browser grid, users cannot only scroll infinitely left and right, but they can also resize, rearrange, or hide columns as wanted, with no additional programming required.

This section gives you an introduction to the programming in RPG with Valence. There are some new procedures to learn, but otherwise not that much different from standard RPG code. This short learning curve gives RPG developers more time to focus on the front-end logic.

Front-end Ext JS/Sencha Touch Development

If you look at the source behind almost any web page, you find a lot of HTML, the basic building block for almost everything that you see in your browser. Although creating a web page entirely through manual entry of raw HTML code is possible, few people create pages that way, just as few people write computer programs in assembly language.

JavaScript Frameworks serve as one common way to abstract programmers from most, if not all of, the intricacies of HTML coding. Complex user interface components like grids, the web equivalent to subfiles, might take many hundreds of lines of HTML code to create. However, they require only a handful of config lines in the Valence Framework when you use Ext JS. This simplicity allows developers to create highly functional web pages with much fewer lines of code. They need to learn only how to interact with the framework.

For RPG programmers, this part of the application development process (coding in JavaScript and Ext JS or Sencha Touch) is where the learning process reaches high gear. When you are figuring out how these frameworks operate you, are acquiring valuable skills for creating web and mobile applications regardless of what back-end platform you are working with. Many new development tools have been developed to help make this learning curve a little shorter.

Among the more prominent of these tools is Sencha Architect, the Sencha IDE for developing the front-end part of an application. Direct integration with Sencha Architect is included in Valence 4.0 and later. This enables developers to visually develop much of their Valence applications by dragging and dropping Ext JS or Sencha Touch components into a development “canvas”. This process accelerates the process of creating web and mobile applications because formerly all the coding behind an Ext JS or Sencha Touch application needed to be entered manually.

Figure 1-58 shows the development window for Sencha Architect.

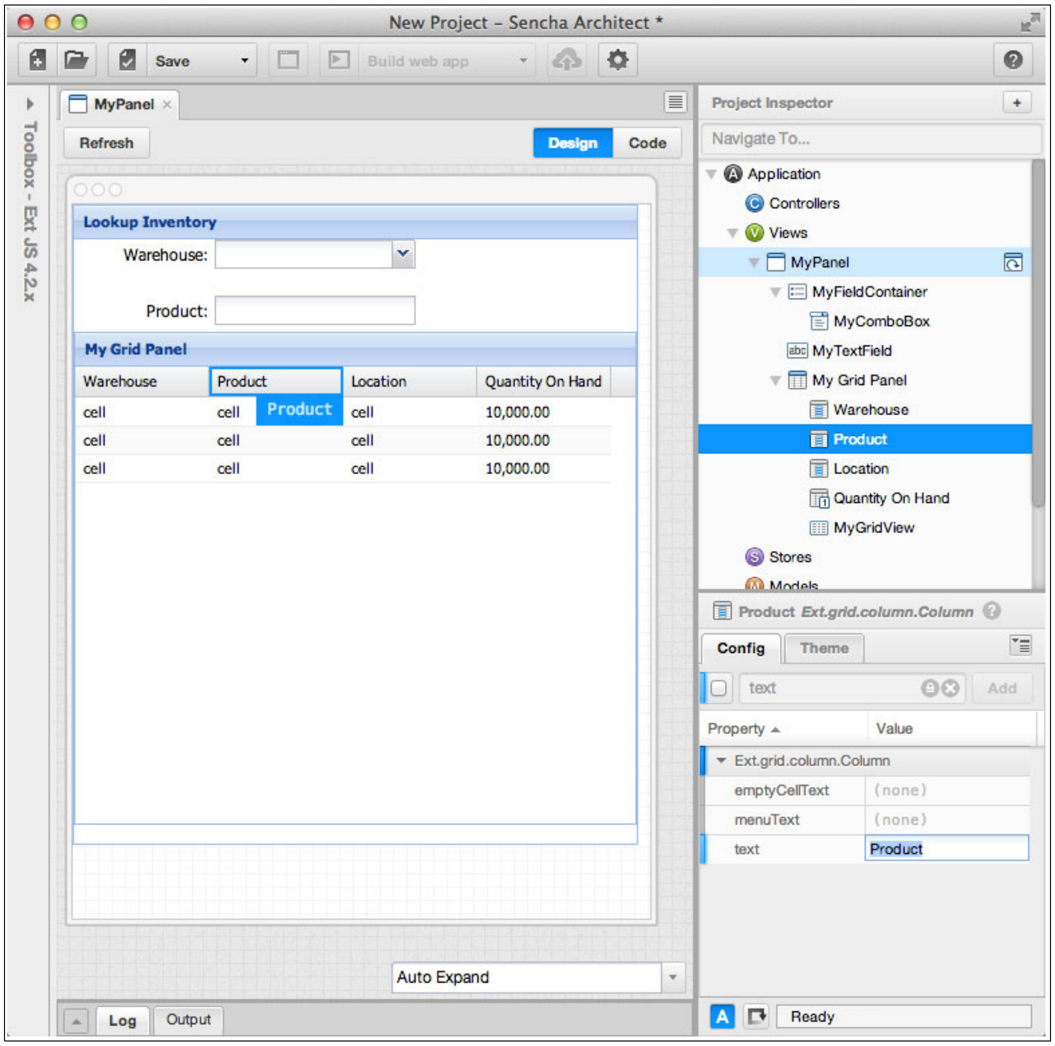


Figure 1-58 Front-end Valence App Development with Sencha Architect

Although developers still need to learn and understand new design concepts, particularly the various configurations and properties of the myriad of available UI components in Ext JS and Sencha Touch, the use of Sencha Architect helps developers work with the more complex aspects of the framework. When you are visually designing a layout in Sencha Architect, the configuration of the design is stored in a meta language. This language is translated into JavaScript files that are sent to the browser when the application is run. To modify a screen layout, open your design, make changes, and save. Sencha Architect regenerates all of the JavaScript again.

Architect is typically used in a manner that creates Ext JS and Sencha Touch applications that follow an MVCS (Model-View-Controller-Store) design paradigm, in which the front end is separated by function roughly as follows:

- Model. Holds the definitions of the data that is used in the application. From an RPG perspective, think of a model as a collection of data structures and information about how they are loaded.
- View. The visual piece of the application, what you see in the “canvas” of Architect. A view is where you control what the user sees.

- Controller. Where logic is run in response to an action that is taken in the View. It might result in a call to an RPG program, a change in the View, a (re)load of a Model, or any other type of action.
- Store. Working with the Models, a Store is where local data (lists) is stored on the front end. It is typically referenced within the Controller code when it is accessing data.

The MVCS approach to Ext JS and Sencha Touch application creation results in code that is easily modified or extended. The front-end piece of the inventory application example set up in the RPG examples of the previous section includes a Model that holds code that lists the fields to be expected from the SQL statement and the RPG call that populates them. The View contains the configuration information for the prompt window and the grid that shows the inventory data. The Controller manages the interaction between the two. The Store houses local lists of data, such as warehouses and on-hand inventory, after they are sent to the front end by an RPG call. You can see examples of this process in action by downloading the Valence Framework and navigating to the “Examples” folder that is created on your IFS during installation.

Valence system requirements

Valence runs on IBM i at V5R4 or later. To use the IBM i Access for Web tools, licensed program 5770XH2 must be installed.

Valence can be used on IBM i for free under terms of the Community Developer License, with extra features and capabilities enabled under the paid license terms of Valence Enterprise. More information is available at:

http://www.cnxcorp.com/valence

1.5 ProfoundLogic

This section describes Profound Logic Genie and Profound UI Rich Displays.

1.5.1 Genie from Profound Logic Software

The Profound Logic Software modernization solution is called Profound UI. It is a suite of tools for modernizing your user interfaces. These tools include the Rich Display module, Atrium, and Genie. The Rich Display module, which uses Rational Open Access for RPG, provides a modern way to develop rich graphical interfaces that run in a web browser or mobile device. Atrium is a modern web portal or menu system for starting applications. Genie translates green screens into web pages in real time.

This solution guide focuses on Genie, which is a tool to reface, or screen scrape, your 5250 applications. With Genie, you can give your screens a modern look without investing heavily in development time and tools. In fact, within seconds of installing Genie, you can run all of your 5250 applications in a web browser.

How Genie works

Genie includes a 5250 emulator that is similar to IBM i Access (previously known as Client Access), except that it runs in the HTTP server on the IBM i operating system. Genie converts each 5250 panel into a window that is suitable for display in a web browser. Genie is written in native ILE languages such as RPG and C++, and therefore has about the same performance characteristics as traditional green-screen applications.

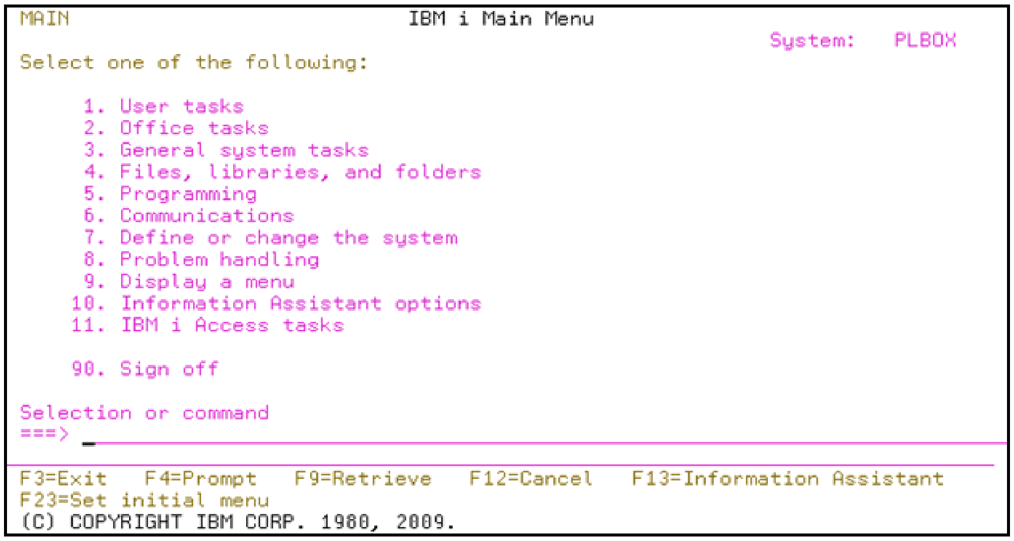
In addition to providing 5250 emulation in a browser, Genie is fully theme-able. You can create themes (which Genie calls “skins”) that specify different appearances to fit your environment. This includes color schemes, logos, fonts, scripts, and more, allowing you customize the Genie experience to fit your company's environment.

Each Genie skin has options for automatic window modernization, which turn function keys into buttons, make menus clickable, and change subfiles into modern-looking grids.

In addition to customizing Genie with your own skin, each window can be transformed to take better advantage of the new browser environment. In Genie, this is done by choosing and finding a string on the window that uniquely identifies that particular window. When Genie sees this “screen identifier”, it can apply extra transformations, such as changing colors and fonts, or adding new graphic elements such as pictures, charts, or even file upload/download capabilities. These extra graphic elements can be added “on-window” as you are viewing it in your web browser using Genie's window designer.

Automatic Transformations

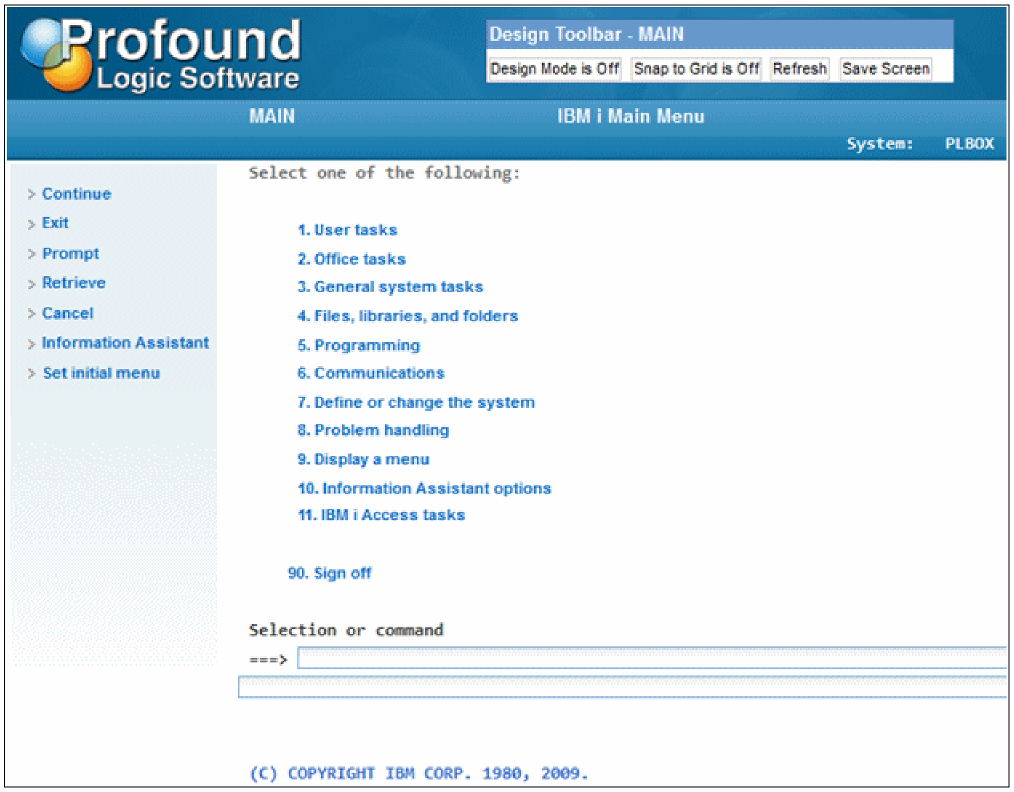
There are certain elements on a window that Genie can automatically transform to give it a more modern web-like appearance or behavior. For example, in Figure 1-59, you can see the familiar IBM i Main Menu as you see it in a traditional 5250 emulator.



```
MAIN                              IBM i Main Menu
                                                             System:   PLBOX
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. IBM i Access tasks

    90. Sign off

Selection or command
===> _

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2009.
```

Figure 1-59 IBM i Main Menu in a 5250 emulator

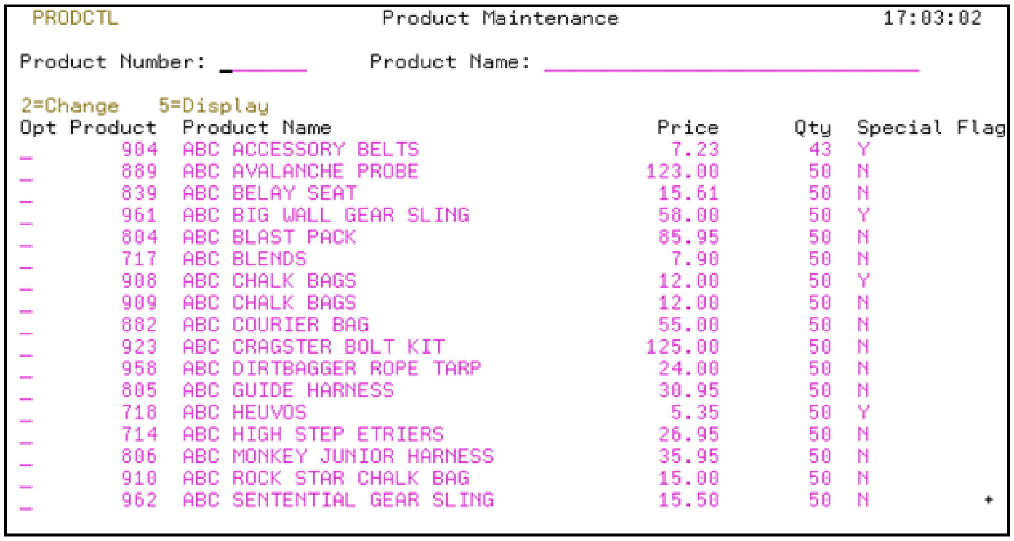
Genie automatically enhances this window, as shown in Figure 1-60. The fonts are changed to more web-friendly fonts, and the colors are set up according to a blue-and-white color scheme that is defined in a Genie "skin." The skin is also set up to automatically transform the function keys into a menu on the left side of the window, and make each menu option into a hyperlink that can be clicked to run the option. However, you can also use the traditional method of typing a number and pressing Enter. All of these changes are done automatically, just by opening this window in Genie.



```
MAIN                       IBM i Main Menu
                                              System:    PLBOX
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
     10. Information Assistant options
     11. IBM i Access tasks

    90. Sign off

Selection or command
===>

(C) COPYRIGHT IBM CORP. 1980, 2009.
```

Figure 1-60 IBM i Main Menu in Genie

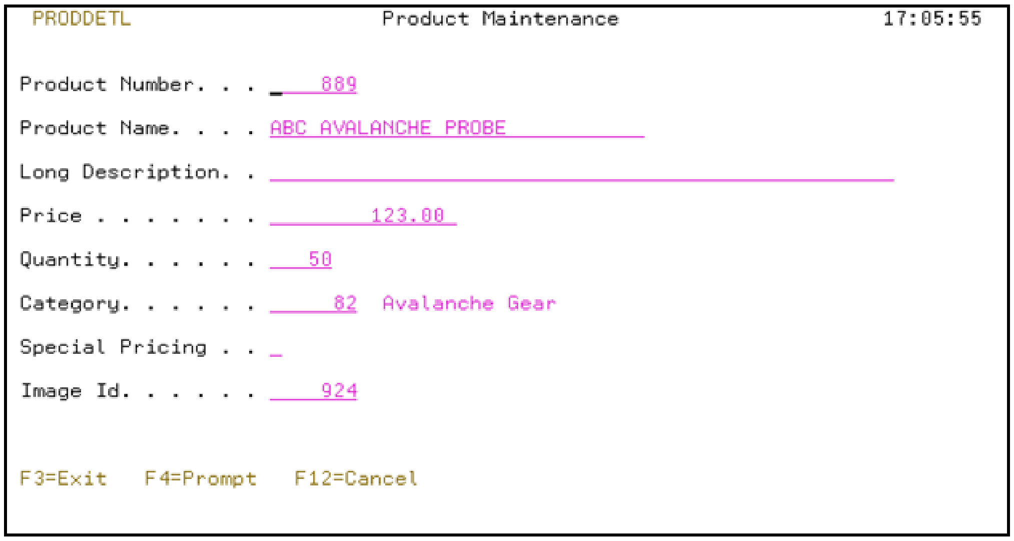
Genie can be used to transform your own applications, in addition to the screens that are provided by IBM. Consider how you might transform a typical subfile application into a web accessible program by refacing it with Genie. Figure 1-61 shows a typical IBM i application panel that contains a subfile.



```
 PRODCTL                       Product Maintenance                     17:03:02

Product Number: _______     Product Name: ______________________________

2=Change    5=Display
Opt Product  Product Name                         Price      Qty  Special Flag
_        904  ABC ACCESSORY BELTS                  7.23       43  Y
_        889  ABC AVALANCHE PROBE                123.00       50  N
_        839  ABC BELAY SEAT                      15.61       50  N
_        961  ABC BIG WALL GEAR SLING             58.00       50  Y
_        804  ABC BLAST PACK                      85.95       50  N
_        717  ABC BLENDS                           7.90       50  N
_        908  ABC CHALK BAGS                      12.00       50  Y
_        909  ABC CHALK BAGS                      12.00       50  N
_        882  ABC COURIER BAG                     55.00       50  N
_        923  ABC CRAGSTER BOLT KIT              125.00       50  N
_        958  ABC DIRTBAGGER ROPE TARP            24.00       50  N
_        805  ABC GUIDE HARNESS                   30.95       50  N
_        718  ABC HEUVOS                           5.35       50  Y
_        714  ABC HIGH STEP ETRIERS               26.95       50  N
_        806  ABC MONKEY JUNIOR HARNESS           35.95       50  N
_        910  ABC ROCK STAR CHALK BAG             15.00       50  N
_        962  ABC SENTENTIAL GEAR SLING           15.50       50  N          +
```

Figure 1-61 Traditional 5250 panel with a subfile

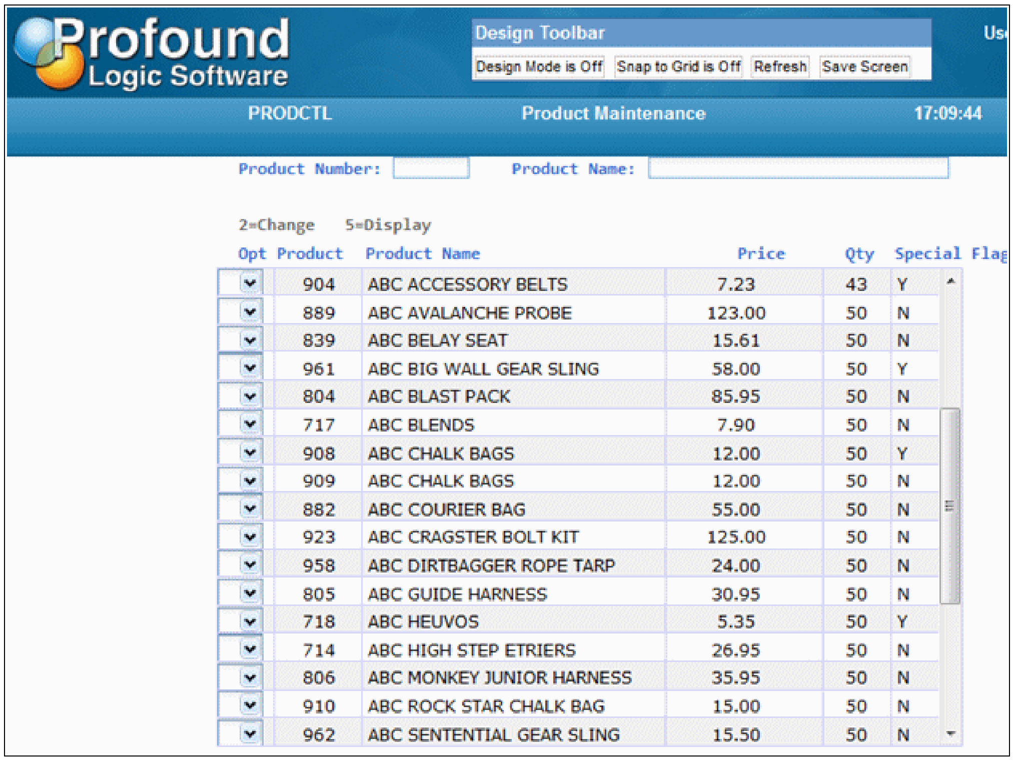
This application is a maintenance program for a product master file. The user can place a 2 next to any item to change it. When they do that, they see the window in Figure 1-62.



```
 PRODDETL                      Product Maintenance                     17:05:55


Product Number. . . _   889

Product Name. . . . ABC AVALANCHE PROBE

Long Description. . 

Price . . . . . . .         123.00

Quantity. . . . . .    50

Category. . . . . .      82  Avalanche Gear

Special Pricing . . _

Image Id. . . . . .     924


F3=Exit    F4=Prompt    F12=Cancel
```

Figure 1-62 Traditional 5250 panel that contains product detail

This type of screen is common in the 5250 environment. If it is viewed in Genie, it can automatically detect the subfile and transform it into a modern-looking grid on the web page that is displayed. Figure 1-63 shows what that looks like.



PRODCTL Product Maintenance 17:09:44

Product Number: Product Name:

2=Change 5=Display

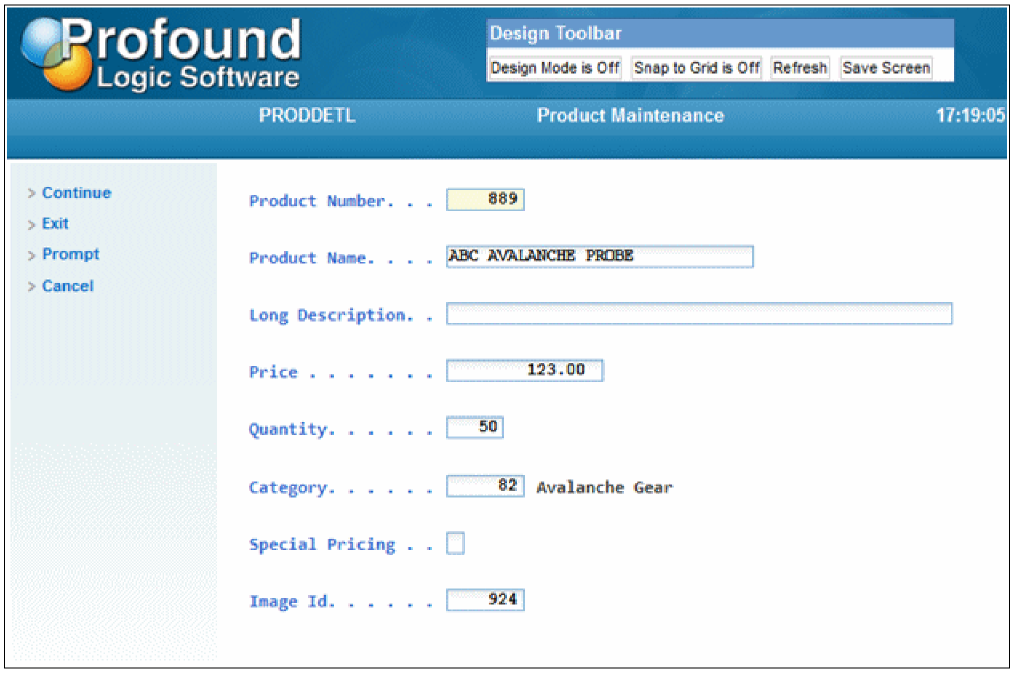
Opt	Product	Product Name	Price	Qty	Special Flag
	904	ABC ACCESSORY BELTS	7.23	43	Y
	889	ABC AVALANCHE PROBE	123.00	50	N
	839	ABC BELAY SEAT	15.61	50	N
	961	ABC BIG WALL GEAR SLING	58.00	50	Y
	804	ABC BLAST PACK	85.95	50	N
	717	ABC BLENDS	7.90	50	N
	908	ABC CHALK BAGS	12.00	50	Y
	909	ABC CHALK BAGS	12.00	50	N
	882	ABC COURIER BAG	55.00	50	N
	923	ABC CRAGSTER BOLT KIT	125.00	50	N
	958	ABC DIRTBAGGER ROPE TARP	24.00	50	N
	805	ABC GUIDE HARNESS	30.95	50	N
	718	ABC HEUVOS	5.35	50	Y
	714	ABC HIGH STEP ETRIERS	26.95	50	N
	806	ABC MONKEY JUNIOR HARNESS	35.95	50	N
	910	ABC ROCK STAR CHALK BAG	15.00	50	N
	962	ABC SENTENTIAL GEAR SLING	15.50	50	N

Figure 1-63 Subfile that is transformed in real time by Genie

As with the menu example, no code changes were needed to transform the window. Genie automatically detected that it was a subfile, and converted it into a grid. Genie saw the text “2=Change 5=Display” at the top, and automatically converted it into a drop-down box for each row of the subfile, so the user can select from these options. It also added a scroll bar on the right side.

Unfortunately, because Genie only knows what text is on the display (because it acts as a 5250 emulator) it does not have any way to know the total size of the subfile. Therefore, it cannot do “free scrolling” like a user might expect from a grid. Instead, it does the best that a screen-scraper can do. When you pull the scroll bar down, it sends a page down to the server, and when you pull up on the scroll bar, it sends a page up.

However, the second window of this application has little that Genie can do automatically. It is converted to a web page, as shown in Figure 1-64, but that page looks much like the original 5250 panel did.



Product Number. . . 889

Product Name. . . . ABC AVALANCHE PROBE

Long Description. .

Price 123.00

Quantity. 50

Category. . . . . . 82 Avalanche Gear

Special Pricing . .

Image Id. 924

Figure 1-64 Product Details Before Genie Enhancements

The Design toolbar at the top of the display is normally configured to be accessible to developers who might want to customize a window, and is not typically available to users. Click the **Design Mode is Off** option on this toolbar to switch on design mode. When it is on, you can make enhancements to the refacing of this window so that it looks more modern.

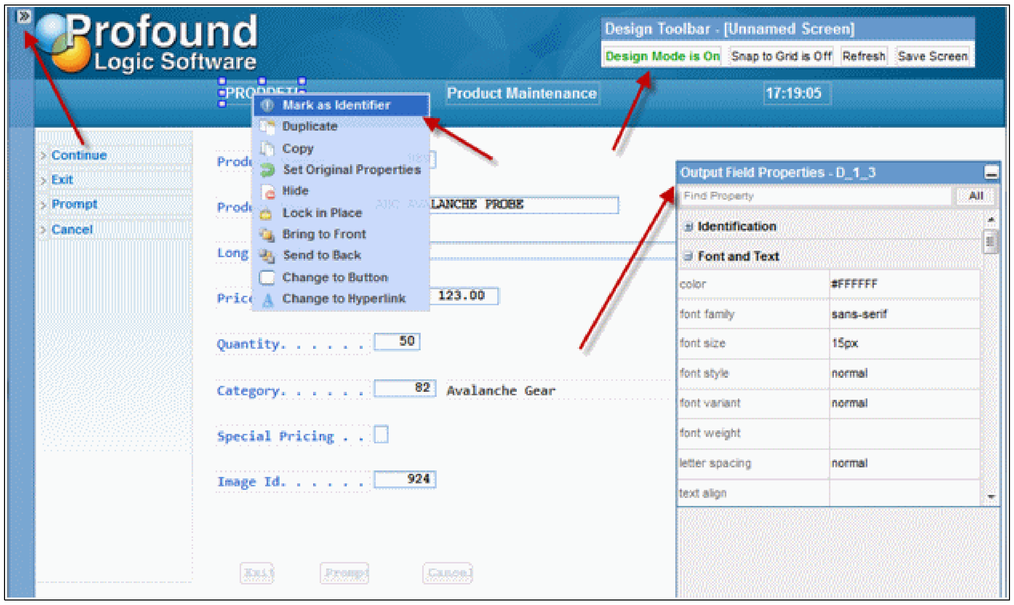
In this example, the following improvements are made while in Genie's design mode:

1. The Special Pricing field is normally set to Y if special pricing is wanted, or blank if it is not. Change that to a check box, so the user can check it to designate special pricing.
2. Change the Category field to be a drop-down box (or a “select box” in HTML terms) so that the user does not need to know the category number. The user can then select the category from a list.
3. Include a picture of the product on the display. This file has a field that is called Image Id that corresponds to a picture in the IFS. Genie displays that picture.

To do any of these things, Genie must be told how to detect this particular window because enhancements are made only when the window is displayed. Remember, Genie is a 5250 emulator. It does not know what operations your RPG program is running. It does not know the name of your record formats or display files. All that it knows is that IBM i system asked it to display a window that has certain text strings, and certain input fields, in different places. But, due to the way this window is written, it always has the string 'PRODDETL' in row 1, column 3. Genie can make extra enhancements any time it sees this string. To make enhancements:

1. Click “Design Mode is Off” to toggle design mode to on. This step is needed to make changes.
2. Right-click the “PRODDETL” string and specify “Mark as Identifier”

Figure 1-65 shows what the window looks like when the **Mark as Identifier** option is selected. A button in the upper left can be clicked that expands into a long list of graphical elements (or “widgets” as they are called by Profound Logic) that can be dragged onto the Genie display to add graphical features. The Properties box is titled “Output Field Properties - D_1_3” because it is showing the properties of the PRODDETL field that is in column 1, row 3. This property box can be used to change anything about that field, including the field type, font, color, value of the field, and so on. If a different field on the window is clicked, the property box changes to show the properties of the selected field. For now, only PRODDETL is set as the identifier so that Genie recognizes this window.

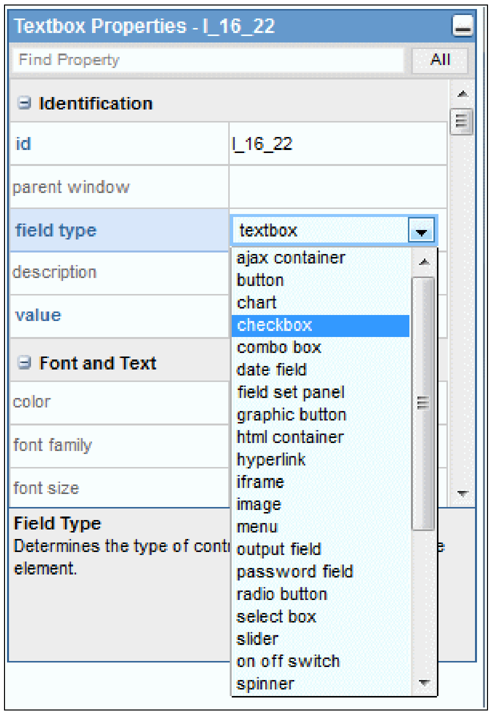


Prod
Prod ... LANCHE PROBE
Long
Pric ... 123.00
Quantity. 50
Category. . . . . . 82 Avalanche Gear
Special Pricing . .
Image Id. 924

color	#FFFFFF
font family	sans-serif
font size	15px
font style	normal
font variant	normal
font weight	
letter spacing	normal
text align	

Figure 1-65   Marking the identifier in the Genie designer

To change the Special Pricing field to a check box, use the properties box. Click the blank next to the words “Special Pricing”, and in the properties box, change its field type to **checkbox**, as shown in Figure 1-66. There are also properties in that box for the value that is sent to the 5250 panel when the box is checked or cleared. These properties can be set to Y, and blank, so that when the user checks the box, a Y is sent to the server, and when it is cleared, a blank is sent. The underlying 5250 program does not know that anything is changed, so neither the screen nor the program needs to be modified.


Textbox Properties - I_16_22

Find Property | All

Identification

id	I_16_22
parent window	
field type	textbox
description	
value	

Font and Text

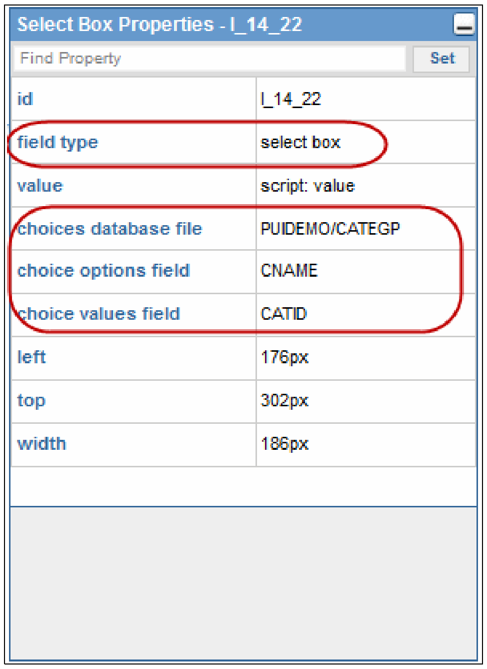
color	
font family	
font size	

Field Type
Determines the type of contr... element.

Figure 1-66 Changing the field type to a check box

The same technique can be used to change the Category field to a drop-down. In this case, use a field type of “select box” (instead of the check box that is used for the Special Price field). It is also possible to tell Genie where to get the values to display in the drop-down menu. This can be done by pointing Genie at a physical file on the system, by setting the choices database file property. The choice options field property is where you tell Genie the field name of the choices to be displayed to the user. The choice values field is the field name that contains the category number to send back to the 5250 application. The user can now select a category name in a drop-down box, and the 5250 application gets the category number, as though nothing had changed.

Figure 1-67 shows the properties that changed to make the field into a drop-down box.

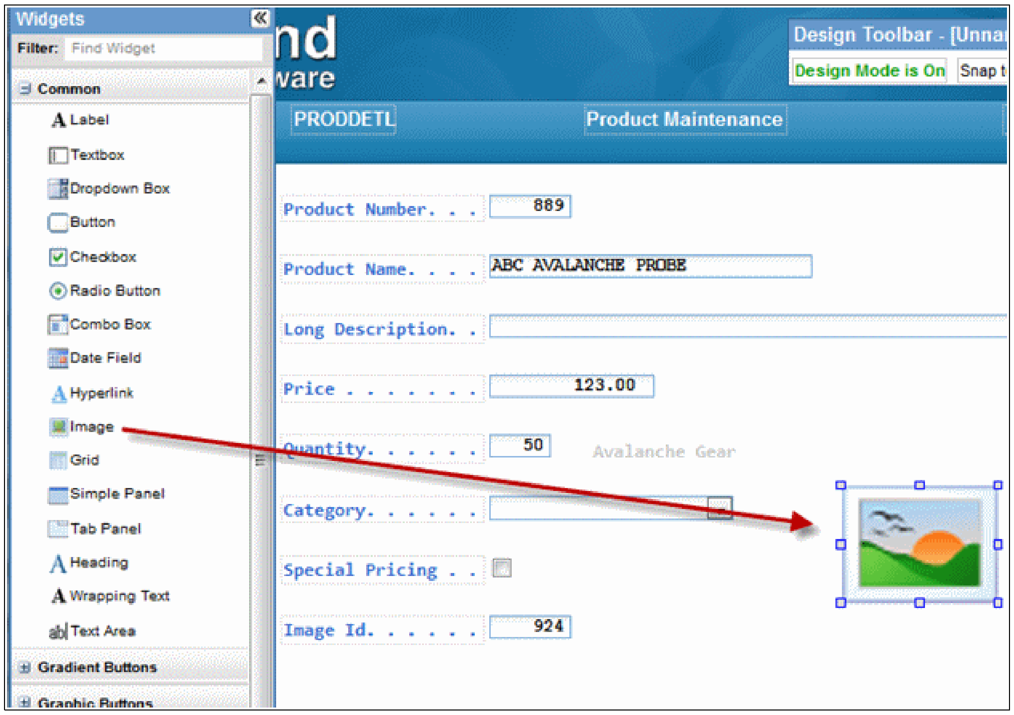


Select Box Properties - I_14_22

id	I_14_22
field type	select box
value	script: value
choices database file	PUIDEMO/CATEGP
choice options field	CNAME
choice values field	CATID
left	176px
top	302px
width	186px

Figure 1-67 Converting the category to a drop-down box

To enable an image to be displayed, a new element must be added to the window. Click the button in the upper-left corner to expand the Widgets dialog, and drag an image widget to the window, as shown in Figure 1-68.

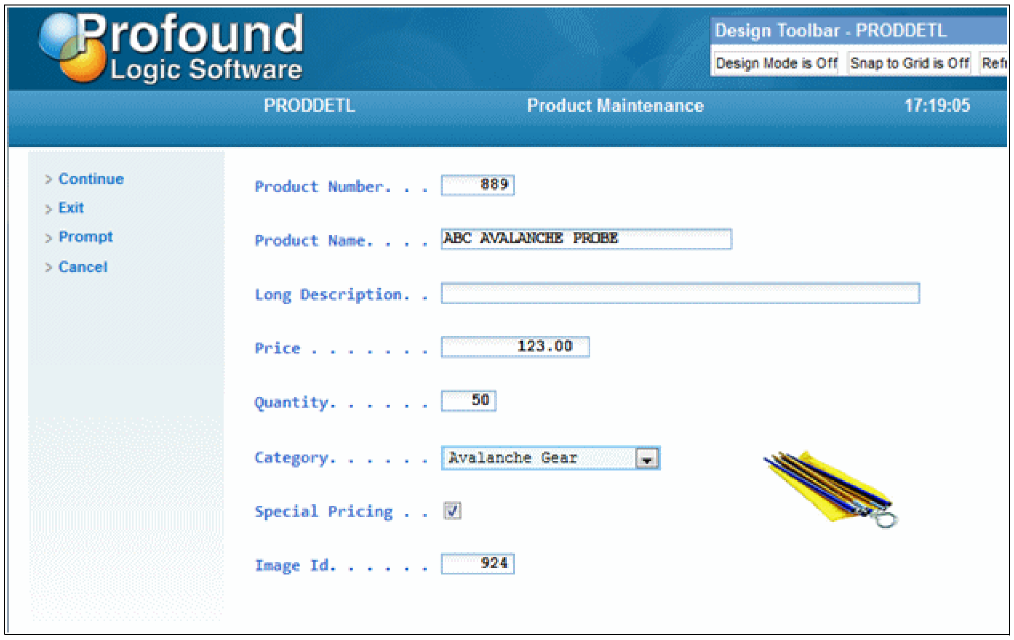


Product Number. . .	889
Product Name. . . .	ABC AVALANCHE PROBE
Long Description. .	
Price	123.00
Quantity.	50 Avalanche Gear
Category.	
Special Pricing . .	
Image Id.	924

Figure 1-68 Dragging an image from the Widgets dialog

The images are in the IFS, within the document root of the HTTP server. To display a different image depending on the value of the Image Id field on the window, write a small bit of code to run when this image is displayed. To do that, set the “image source” property of the image to script: "/profoundui/userdata/product_images/" + get("I_18_22") + ".gif". This tells it to build an IFS path name, containing an IFS directory name, followed by the contents of the I_18_22 field, followed by .gif image. Because this image ID is 924, the result of this bit of code is "/profoundui/userdata/product_images/924.gif". However, when the value of the Image Id field is a different number, it changes accordingly. Any of the fields in Genie can be scripted this way, allowing a great deal of flexibility.

When the window customizations are complete and design mode is exited, the window looks like what is shown in Figure 1-69.



PRODDETL Product Maintenance 17:19:05

Product Number. . . 889

Product Name. . . . ABC AVALANCHE PROBE

Long Description. .

Price 123.00

Quantity. 50

Category. . . . . . Avalanche Gear

Special Pricing . . ✓

Image Id. 924

Figure 1-69 Product Detail Screen with Genie Enhancements

As you might have noticed, there are pros and cons to a screen-scraper interface like Genie. The underlying programs do not know that a modern, web-based window is used, and all input is coming from a 24x80 (or 27x132) text image of a display. Genie never knows the display file name, record format name, or field names of the fields. It can only display (or transform) what fit on that original text display. For example, it is not possible to allow dragging and dropping, or make a subfile with 50 rows on the window at once, or have a “true” scroll bar. However, you can get up and running quickly with a screen-scraper solution like Genie, and you do not need to have the source code of the application. You can even show IBM-supplied screens (such as Work with Spooled Files, for example) in the application.

Genie can also display screens that are created outside of the 5250 environment using the Rich Display module. They can be run from the same menus as traditional 5250 screens, allowing an easy upgrade path to Rich Displays that are not limited by the underlying 5250 screens.

Transforming to modern web pages

Genie works by transforming 5250 screens in real time into modern-looking web pages, but it also can work together with the Profound Logic Rich Display solution. With Genie, you can be up and running immediately, and you can transform any 5250 display, including those that you do not have source code for. However, with the Rich Display solution, you can go beyond the limitations of the 5250 protocol to support modern features such as drag and drop, unlimited window space, and mobile applications.

The integration between these two products allows you to run them both together. For example, you might start with Genie today, and gradually convert to Rich Displays over time. By combining the two, you can have a traditional 5250 menu that runs both types of programs. As you change, you can gradually replace the 5250 displays with the new Rich Displays without disrupting the business processes. The new Rich Displays can switch back to 5250 displays to incorporate operating system tools, such as the Work With Spooled Files command. Integrating Rich Displays with Genie gives you the best of both worlds.

Resources

To learn more, get a sales quote, see a demonstration, or download a free trial of Profound UI, contact Profound Logic at 1-877-224-7768 or sales@profoundlogic.com. You can also visit the website at:

http://www.profoundlogic.com

1.5.2 Profound UI Rich Displays

The Profound Logic Software modernization solution is called Profound UI, and it is a suite of tools for modernizing your user interfaces. These tools include Genie, which translates green screens into web pages in real time. Atrium is a modern web portal or menu system for starting applications. Profound UI Rich Displays provides rich, modern, graphical user interfaces for RPG programs. These displays can run in a web browser or on a mobile device.

This section focuses on Rich Display tool and its Visual Designer, which is designed to make it easy for programmers with RPG skills to build new applications with modern displays rapidly. There is a conversion module available that can be used to convert your existing 5250 displays into rich displays so that you can preserve your investment in your business applications.

Profound Rich Displays: Modernize with RPG Open Access

The Profound Logic Rich Display module is intended to be a native GUI for ILE RPG. It provides an easy-to-use, WYSIWYG window designer that can be used to build displays in a manner that is similar to what is done in other environments such as Microsoft Visual Basic. When designed and compiled, a Rich Display is run from an RPG program using the standard EXFMT opcode, or by using other file operations like READ, WRITE, and READC. Even though RPG programs can show rich displays with the same opcodes that are traditionally used for 5250 displays, rich displays do not use 5250 display. Instead, Profound UI uses an open access handler so that the RPG opcodes like EXFMT call the Profound Logic routines instead of the traditional 5250 routines provided by the operating system.

The Profound UI Rich Display window designer runs in a web browser, and is served from the IBM HTTP Server for i. No additional servers or PC software are required. This simplicity makes it easy to install, upgrade and maintain because installation, upgrades, and backups are all done in a single place: your server that is running IBM i.

Figure 1-70 shows what the Profound UI Rich Display designer looks like. On the left side is a toolbar that contains graphical elements such as text fields, date pickers, buttons, graphs, charts, images, and drop-down lists. These elements are referred to as “widgets”. You can drag them onto your canvas to build a modern graphical screen for your RPG program. Many window formats can be placed within the same Rich Display File, in a similar manner to the way that multiple formats can exist in a traditional 5250 panel. On the right side of the display, you can see a pane for managing those record formats. Beneath that pane, there is a properties pane where you can control all of the details of each widget on the window, including colors, fonts, sizes, styling, and much more.

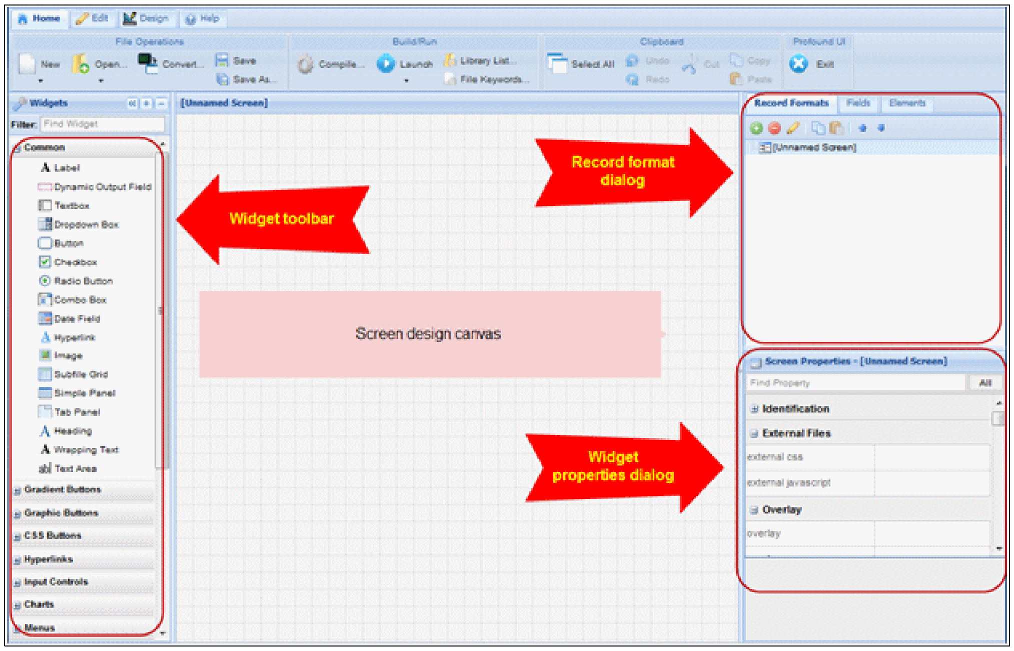
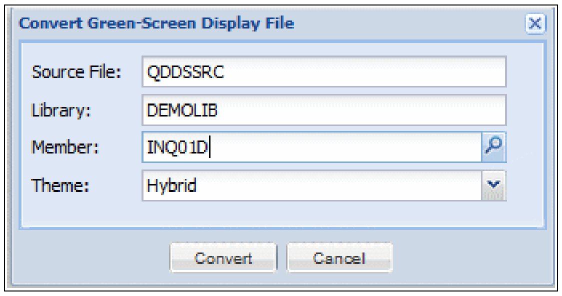


Figure 1-70 The rich window designer

For a shop with existing applications, one helpful tool is the **Convert** button that is near the top of the designer window. The convert function reads the DDS source code for a traditional 5250 display and converts that display file into a new, modern Rich Display file. When you click **Convert**, you see the window shown in Figure 1-71.

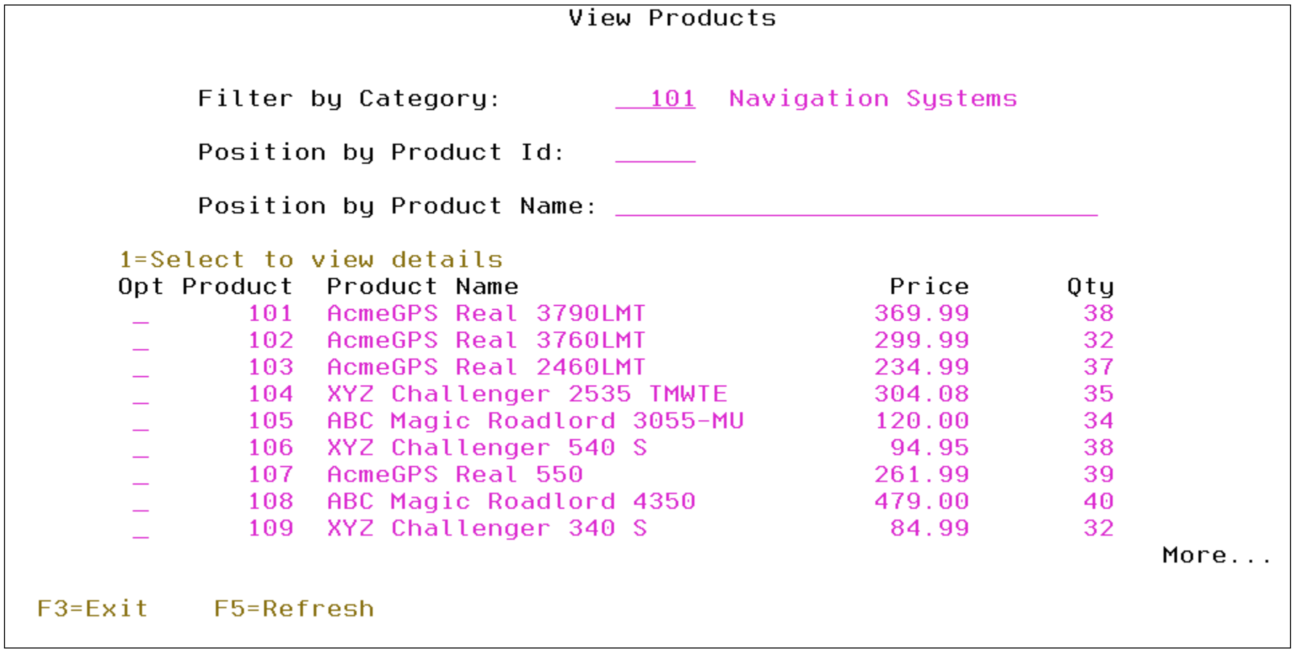

Convert Green-Screen Display File

Source File:	QDDSSRC
Library:	DEMOLIB
Member:	INQ01D
Theme:	Hybrid

Figure 1-71 The Convert window

You can use the **Theme** option to control what the converted window looks like, including the types of elements that are used, the color scheme, and more. You can use one of the pre-built themes, or you can create your own theme, including your own scripting, to customize the conversion even more.

For example, Figure 1-72 shows a typical green-screen product inquiry program.



```
                            View Products

       Filter by Category:         ___101  Navigation Systems

       Position by Product Id:     _____

       Position by Product Name:   ______________________________

     1=Select to view details
     Opt Product   Product Name                   Price      Qty
      _      101   AcmeGPS Real 3790LMT          369.99      38
      _      102   AcmeGPS Real 3760LMT          299.99      32
      _      103   AcmeGPS Real 2460LMT          234.99      37
      _      104   XYZ Challenger 2535 TMWTE     304.08      35
      _      105   ABC Magic Roadlord 3055-MU    120.00      34
      _      106   XYZ Challenger 540 S           94.95      38
      _      107   AcmeGPS Real 550              261.99      39
      _      108   ABC Magic Roadlord 4350       479.00      40
      _      109   XYZ Challenger 340 S           84.99      32
                                                                 More...

  F3=Exit    F5=Refresh
```

Figure 1-72 A 5250 inquiry panel before conversion

Figure 1-73 shows the same inquiry window in the designer after conversion.

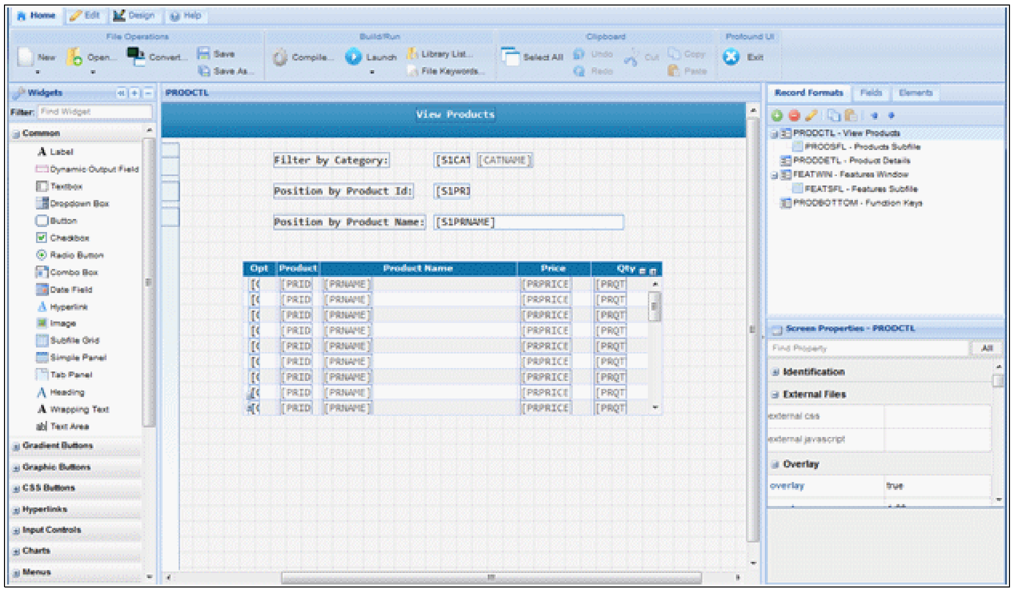


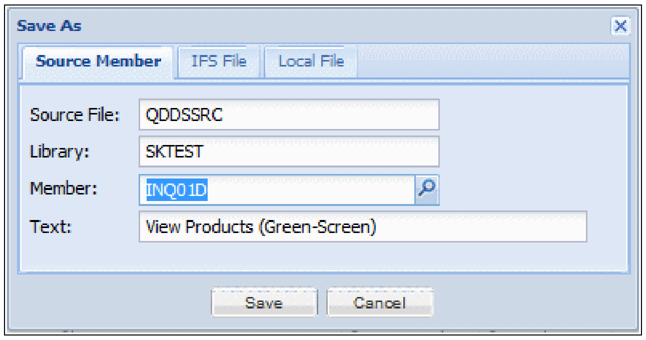
Figure 1-73 Inquiry window in the designer after conversion

In only a second, you have a rich display that matches the original green-screen. However, there are a few important things to notice:

- The field names are converted, and keep the same as they were in the original green-screen.
- All of the record formats are converted, and the format names are also unchanged.
- The subfile now has a “real” scroll bar that scrolls up and down the way a user expects.

- The subfile columns can now be sorted by clicking the heading (no coding is required).
- By turning on a flag in the subfile widget, you can make it exportable to Microsoft Excel (no coding is required).
- You are no longer limited by what fits on a 5250 panel. You can make the screen as wide or tall as wanted, enable dragging and dropping, or even put 200 rows of the subfile on the window at once.

The old 5250 display file is now a Rich Display file that you can edit in the designer. You can change it any way that you want. But, so far, it is only been in the designer window, so the next step is to save the work to disk. To do that, click the **Save As** option in the designer, which opens the window shown in Figure 1-74.



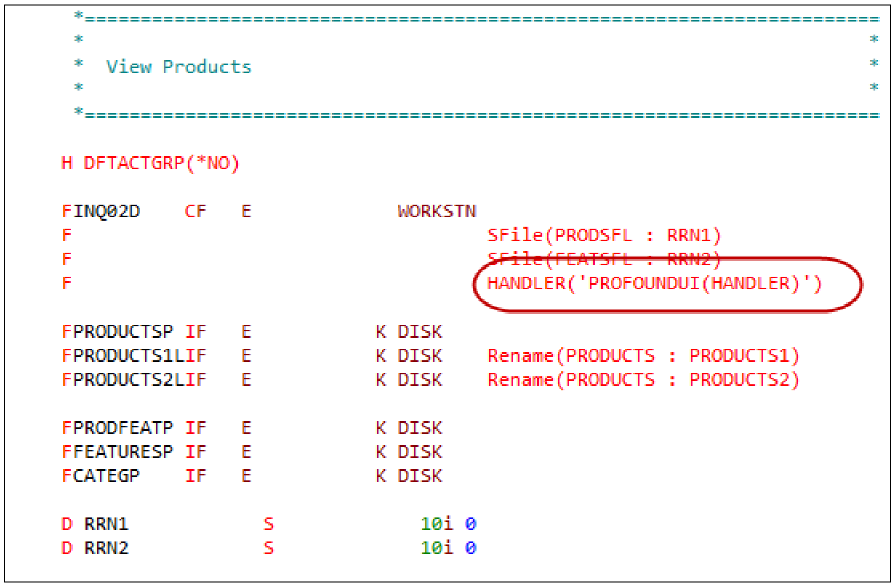
Source File: QDDSSRC
Library: SKTEST
Member: INQ01D
Text: View Products (Green-Screen)

Figure 1-74 The Save As window

This window allows you save your converted display file to disk. At the top, separate tabs allows you save it to a source member, the IFS on your IBM i system, or even a local file on your PC. In this example, it is saved to a member in the QDDSSRC file, and in a different library from the original, to keep both the original green-screen and the new Rich Display file.

The saved display file is kept in DDS format, but it uses the (IBM supplied) HTML DDS keyword to store all of the rich display information. You can still compile the display file with the IBM tools, such as PDM, or the `CRTDSPF` command, or you can compile it by clicking **Compile** in the designer. In all cases, it generates a display file object in a library on the IBM i system that is suitable for use in an RPG program. However, this new display file does not work on a 5250 terminal. It is now a rich display file that can be viewed only in a web browser or on a mobile device. You can run it directly from the IBM HTTP Server (powered by Apache), or you can run it through the Genie interface.

Before you run it, you need to update the RPG program so that it no longer uses the operating system 5250 routines for this display, but instead uses the Profound Logic Rich Display Handler. This update is done by adding the HANDLER keyword to the F-spec in the RPG program, as shown in Figure 1-75. This update is the only change that you need to make to the RPG code. The remainder of the program works the same as it did before.



```
      *====================================================================
      *                                                                   *
      *  View Products                                                    *
      *                                                                   *
      *====================================================================

     H DFTACTGRP(*NO)

     FINQ02D    CF   E             WORKSTN
     F                                     SFile(PRODSFL : RRN1)
     F                                     ~~SFile(FEATSFL : RRN2)~~
     F                                     HANDLER('PROFOUNDUI(HANDLER)')

     FPRODUCTSP IF   E           K DISK
     FPRODUCTS1LIF   E           K DISK    Rename(PRODUCTS : PRODUCTS1)
     FPRODUCTS2LIF   E           K DISK    Rename(PRODUCTS : PRODUCTS2)

     FPRODFEATP IF   E           K DISK
     FFEATURESP IF   E           K DISK
     FCATEGP    IF   E           K DISK

     D RRN1            S             10i 0
     D RRN2            S             10i 0
```

Figure 1-75   The RPG Code with the HANDLER keyword

One small change to the RPG program is all that is needed. This way, companies can preserve their investment in their existing RPG programs, and continue to use their existing in-house RPG talent to maintain them. The IBM RPG compiler and tools are still used to maintain your RPG programs, as before. Rational Developer for i is the tool that is pictured in Figure 1-75, but you can also use SEU or PDM.

If you want to modernize your RPG code or database logic, you can do so in a separate phase of your modernization project. Profound UI allows you to create modern screens that work with either modernized RPG code, or with your existing traditional code. Modernization can occur incrementally, at your own pace.

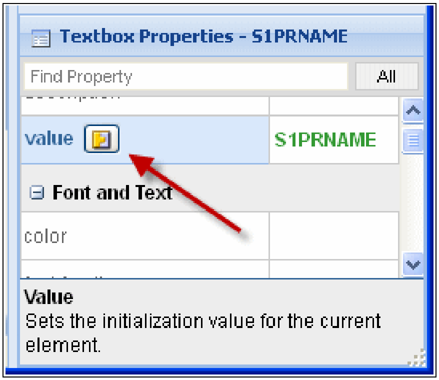
After your RPG program is recompiled, you can run it in a web browser, and you now have a modern web-based application. It has a scroll bar that scrolls freely, not just one page at a time, the way a user might expect. You can also click the column heading to sort the subfile by that column, automatically. So far, aside from adding the HANDLER keyword, no programming is required. The designer does all of the work.

A deeper look at the converted window

The Rich Display designer is not only for converting screens, but is also used to build new screens when you need to write a new application. It also maintains the converted display file for the life of the application.

The window is made up of graphical elements that are called “widgets”. Each widget represents one item on the window. It might be a label that prints static text on the window, an output field that shows text from an RPG program, a textbox that provides an input field for the user to type, a button, a date picker, a calendar, or any of dozens of other graphical widgets.

You can select any widget by clicking it with the mouse, and then you can change its properties in the Properties window in the lower-right corner of the window. For example, this application has a text box where the user can type a product name. You can see it in Figure 1-76 where it says [S1PRNAME] on the display. If you click that text box, the properties window changes to show all of the properties for this text box. The properties window has a scroll bar on the right that can be used to view more properties. Figure 1-77 shows the properties window after you scroll down to the value property and click the blank next to it.



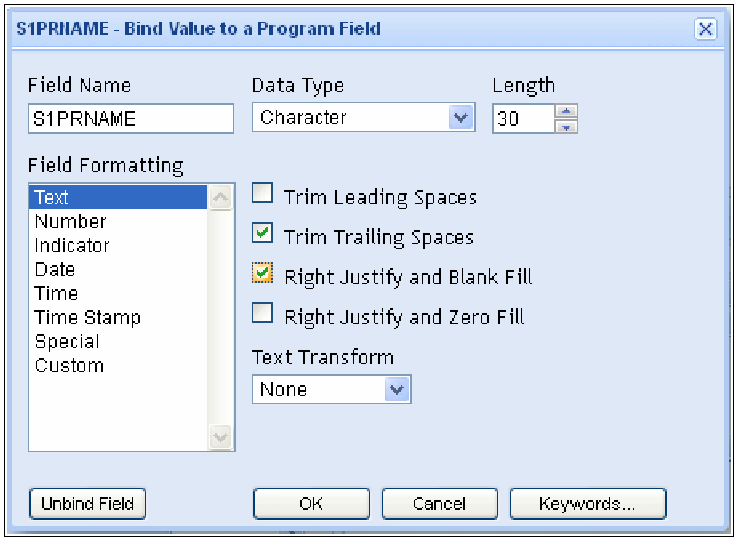
value: S1PRNAME

Value
Sets the initialization value for the current element.

Figure 1-76   Window for a text box with a binding button

The value property of a text box shows the value that is displayed in the text box, or that the user typed into the text box. In this case, that value is connected to a variable in an RPG program. Notice the small yellow button that appears when you click the value property. That button is called the “binding” button, and it is used to connect, or “bind”, a property to an RPG variable.

Clicking the yellow binding button shows the window shown in Figure 1-77. In this window, you can select the field name that to use in your RPG program, and you can define it in any of the data types that are supported by DDS (character, graphic, numeric, date, time, indicator, and so on). Alternately, you can select **Use Reference File** to get the definition from a physical file on disk, like the REF and REFFLD capabilities that you might use in green-screen programming.



Field Name: S1PRNAME
Data Type: Character
Length: 30

☐ Trim Leading Spaces
☑ Trim Trailing Spaces
☑ Right Justify and Blank Fill
☐ Right Justify and Zero Fill

Text Transform: None

Figure 1-77   The binding dialog

There are separate Field Formatting options for each of the possible data types, or for advanced users, you can write your own formatting code in JavaScript. All of the options you expect to find in the green-screen environment are available here, including blank fill, zero fill, edit words, currency symbols, negative number formatting, and so forth.

Although this example shows binding the value field, it is not the only property that can be bound. You can bind almost any property in the properties window to an RPG variable. For example, if you want to change the color of a field based on some calculation in your RPG program, bind a field to the color property. Then, the RPG program can set the color to a string such as 'Red' or 'Green', and the field displays in that color.

To add more elements to the window, you can drag other widgets from the Widgets toolbar on the left to the display. You can bind the properties of your new widget to RPG fields by using the same binding dialog.

Enhancing your screens in the Profound UI Designer

This example shows how to change the [S1CATID] (filter by category) field at the top of the display to be a pull-down box so the user can select the category name:

1. Because the category name is displayed in the drop-down, you no longer need the [CATNAME] field. Click the [CATNAME] field and then the **Delete** button on the keyboard to remove it.
2. Click the [S1CATID] field, and in the properties window, change its field type from **text box** to **select box** (**select box** is the HTML name for a drop-down).
3. Grab the right edge of the field and drag it to the right to make it large enough to contain the category name.
4. In the properties window, scroll down to the “choices database file” property, and change it to the library and file name of a physical file that contains all of the categories. In this example, type DEMOLIB/CATEGP, which is the library and file name of the category physical file.
5. In choices options field, type CATNAME, which is the field name that contains the category from the physical file. If you do not remember the field name, there is a button that you can click, similar to the green-screen F4=Prompt option, that shows the fields in the file.
6. In the choices values field, type CATID (or click the prompt button) to specify the name of the field with the value to be sent back to the RPG program. This is the category ID number.

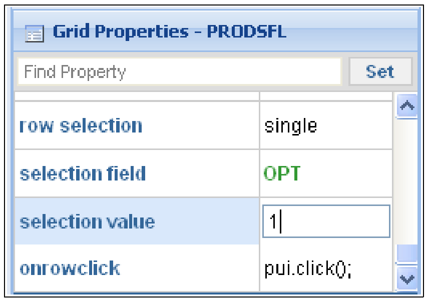
With these changes, Profound UI builds a drop-down list that contains all of the categories in the CATEGP file for the user to select from. The RPG program does not need to be changed at all because Profound UI still sends the category number when the user clicks the category name.

The next enhancement is to enable the user to be able to click a record in the subfile, and have it automatically take them to the second window. This is so they do not need to type a 1 in the Opt column.

1. Highlight the word Opt in the subfile heading, and press the delete key on the keyboard to delete it.
2. Highlight the [OPT] field in the subfile, and press the delete key on the keyboard to delete it.
3. Leave the column in the subfile for now because it can be reused as a place for the product image.
4. Click the subfile, and then find the row selection, selection field, and selection value properties in the Grid Properties window.

5. Change row selection to single. This is a feature of subfiles in Profound UI that makes it easy to allow the user to click a record to select it.
6. When a row is selected, you can optionally have Profound UI add a hidden field to the subfile record that contains a value. This is done by binding (with the binding dialog activated by the yellow button) the selection field property to an RPG variable, and setting the selection value property to the value to be placed in the RPG variable. In this case, the RPG program is already designed to look for a value of 1 in a field that is named OPT. Therefore, bind the selection field property to a character field named OPT, and set the selection value to 1.
7. Set the onrowclick property to pui.click(). The onrowclick property allows you to define some code to run when the user clicks a subfile row. When you call the pui.click() API with no parameters, it acts like the user pressed the Enter button, and submits the window back to the RPG program.

These changes are shown in Figure 1-78. When you have made these changes, you need to save your work and recompile the display file. The result is that clicking a row in the subfile automatically selects it and takes you to the second window to see the item details.



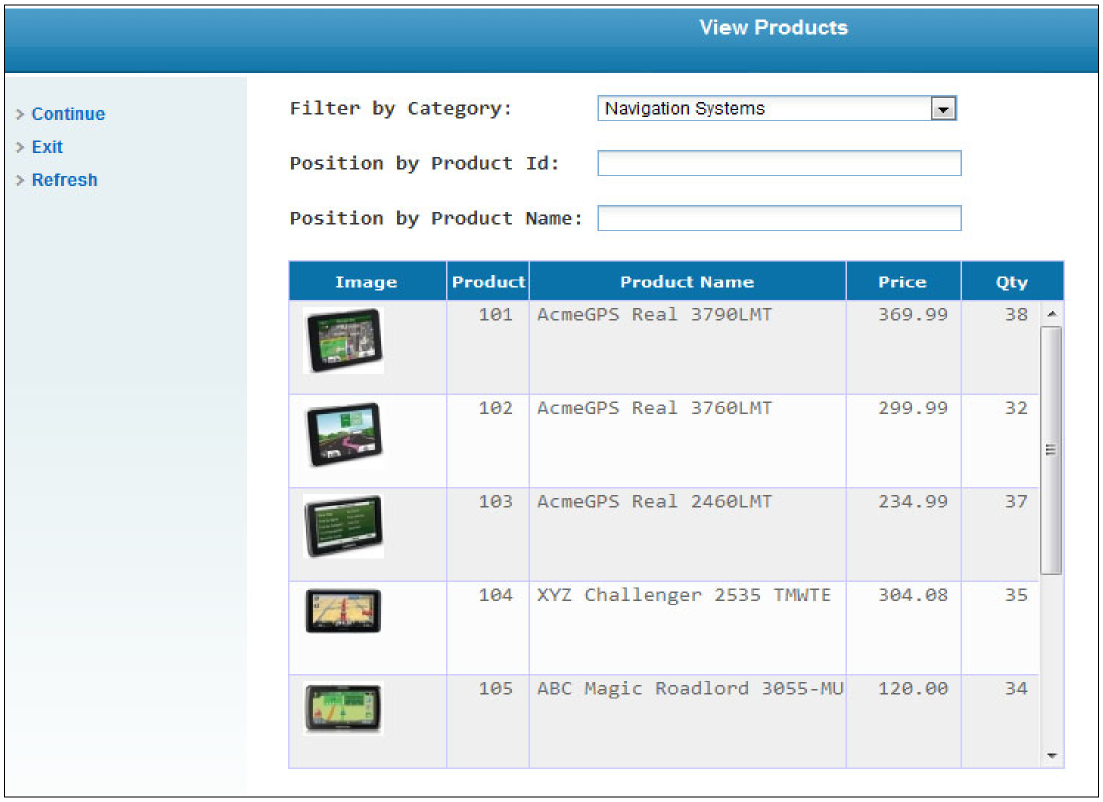
row selection	single
selection field	OPT
selection value	1
onrowclick	pui.click();

Figure 1-78 Options to select a row with a mouse click

To add a product picture into the subfile, perform these steps:

1. Grab the grid line in the subfile, and drag it to make the size of the cells taller. You need more space to insert a picture.
2. Drag an image widget from the widget toolbar into the subfile.
3. Bind the image source property of the image to an RPG variable. In this example, the database record contains a field that is named PRIMAGE that contains the IFS path of the product image, so this is the field that you bind to the image source.
4. In this example, the RPG program is already loading the database record that contains the PRIMAGE field, so the write to the subfile automatically picks up the new field. You do not need to change the RPG code, but you do need to recompile the program.
5. Save and recompile the display file.
6. Recompile the RPG program.

After these changes, your window looks las shown in Figure 1-79. As you can see, it now contains images, a true scroll bar, the ability to sort by column, the ability to click a subfile record to select it, and more. All of these things can be done in only a few minutes of work.



Filter by Category: Navigation Systems

Position by Product Id:

Position by Product Name:

Image	Product	Product Name	Price	Qty
	101	AcmeGPS Real 3790LMT	369.99	38
	102	AcmeGPS Real 3760LMT	299.99	32
	103	AcmeGPS Real 2460LMT	234.99	37
	104	XYZ Challenger 2535 TMWTE	304.08	35
	105	ABC Magic Roadlord 3055-MU	120.00	34

Figure 1-79 The subfile with images and click to select

If you want to compare the features of two units, select one unit, open the “show features” window, and write down the features. Then select the other unit and open the “show features” window of that second unit and compare each feature and see which unit best meets your needs. However, with Profound UI, you can create a drag-and-drop comparison of different navigation units. To do so, complete these steps:

1. Drag a new subfile widget onto the window to contain the features. Give it three columns to contain the feature name, the first GPS to compare, and the second GPS to compare.
2. Drag an output field for the feature name to the first column, and bind it to an RPG variable.
3. Drag an image widget to the window, resize it to be small enough to fit in the subfile, and then drag it into the subfile.
4. Bind the image source property of the new widget to an RPG variable. The RPG program sets this variable to point to a picture of a check mark if the feature is present on a GPS model, or to a red X if it is not present in the GPS model.
5. Repeat steps 3 and 4 for the last column, so that you can list the features of a second GPS.
6. Drag two image widgets to the heading line of your subfile. Name one of them Image1, and the other Image2. Use these widgets to display the product image of the GPS unit that the user drags to the subfile.

7. On the original subfile, set the **allow drag** property to true, and the drop targets to Image1 and Image2. You can then drag any subfile record to either the image1 or image2 properties that were added in step 6.
8. In the window-level properties, use the binding dialog to bind the dd record number ("dd" is short for "drag and drop") to the DDRRN RPG variable. When someone drags a record to one of the drop targets, this field is set to the subfile record number that corresponds to where the record was dragged from.
9. Also, in the window-level properties, use the binding dialog to bind the target element ID property to an RPG field named DDELEM. The RPG program can read this variable to find out where you dragged the image to.

You have enabled dragging and dropping in your rich display file. The RPG program also needs to be modified. It can use the DDRRN field to read the original subfile to get the record that was dragged, and then based on the contents of DDELEM, it can load the feature comparison subfile to show the data for the unit that was dragged.

Figure 1-80 shows the completed panel with drag-and-drop enabled. This window is far beyond anything that can be done in a green screen application, even one that is enhanced with a screen-scraper like Genie. With these capabilities, you can build business applications that are easy to use, reduce the training time for new users, and take advantage of your existing investment in RPG.

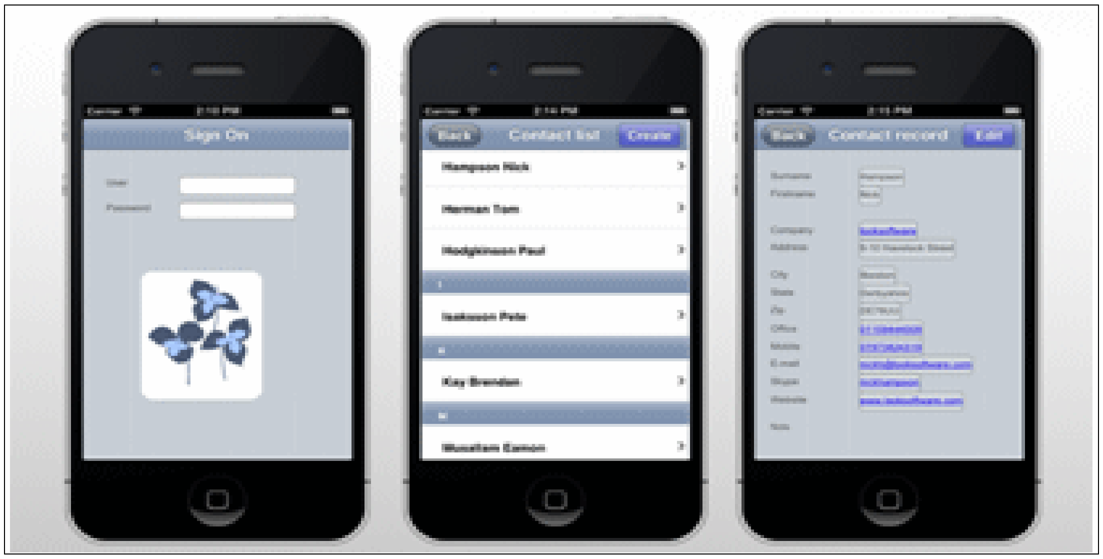


Figure 1-80 Drag to Compare GPS Units

Preparing for the future

The Rich Display solution allows you to go beyond the limitations of the 5250 protocol in the simplest, most intuitive way. However, it does require you to have the source code for any panel you want to convert, and there is a transition period while you update your applications. The Rich Display module can be integrated with the Profound Logic Genie module. Genie is a tool that transforms 5250 displays into web pages in real time (it is sometimes called a "screen-scraper.")

The integration between these two products helps with the transition period from 5250 to Rich Displays. For example, you might start with Genie today, and gradually convert to Rich Displays over time. You can run either 5250 screens or Rich Displays within the Genie environment. As you make the transition, you can gradually replace the 5250 displays with the new Rich Displays without disrupting the business processes. The new Rich Displays can switch back to 5250 displays if they want to incorporate operating system tools, such as the Work With Spooled Files command. Integrating Rich Displays with Genie gives you the best of both worlds.

Resources

To learn more, get a sales quote, see a demonstration, or download a free trial of Profound UI, contact Profound Logic at 1-877-224-7768 or `sales@profoundlogic.com`. You can also see the website at:

`http://www.profoundlogic.com`

1.6 Rocket Software

This section describes Rocket LegaSuite.

1.6.1 Rapid Application Modernization with Rocket LegaSuite

Today's workplace has undergone a profound technological shift. The proliferation of web and mobile technologies puts at risk the decades of investment that organizations have in their enterprise applications. To remain competitive, business and IT executives must quickly provide innovative applications to their customers and employees. User expectations are changing, too. A new generation of employees expects the same level of ease and usability in work applications as they experience with consumer-oriented mobile and web applications. These pressures are causing organizations to reconsider how to manage the large investments that they have in business-critical enterprise applications.

The stereotypical enterprise application presents a text-based, keyboard-driven interface to users. Training new employees to use these “green-screen” applications costs more in terms of time and money than training employees to use the more modern user interfaces that most of today's workforce is accustomed to.

Although experienced users and developers find text-based interfaces to be high-performing, novice users who are more accustomed to web and mobile applications might become frustrated by the lack of interactivity and mobility in green-screen applications.

Similarly, the architecture of many enterprise applications makes it challenging to access and share data and information with the web and mobile applications that are proliferating in many organizations. Many enterprise applications were designed when centralized computing was the norm, so they do not easily fit into to the current IT environment.

Organizations have tried various approaches to modernize enterprise applications, including converting existing code to newer languages or completely replacing their application infrastructure with brand-new technology. However, these invasive approaches present significant drawbacks. Rehosting applications does not often result in improved function and usability. And replacing existing enterprise applications with entirely new applications is costly, time-consuming, and introduces a high level of risk and disruption to day-to-day operations.

As a result, business and IT executives are enthusiastic about application modernization strategies that add new capabilities to enterprise applications and facilitate their integration with emerging technologies.

Using Rocket LegaSuite, organizations can quickly enhance their existing enterprise applications with modern user interfaces and reusable services that use existing application investments. LegaSuite facilitates the development of solutions that offer a faster time to market and a lower cost point than more invasive modernization approaches.

Business value and solution overview

Rocket LegaSuite allows organizations to gain more value from their trusted enterprise assets by improving and updating application function, extending applications for use with web and mobile architectures, automating application processes and navigation, and integrating with other enterprise applications.

Organizations can quickly switch from working with text-based interfaces to working with engaging, intuitive applications without incurring the risks that are associated with changing their existing enterprise applications. All of the original application business logic and data is preserved, and new function is presented in a format that is easy to learn.

With LegaSuite, organizations can quickly deploy modernized applications that are more valuable than the sum of the underlying parts. And because LegaSuite uses existing assets, organizations can achieve their application modernization goals in a short time, with low cost and low risk. See Figure 1-81.

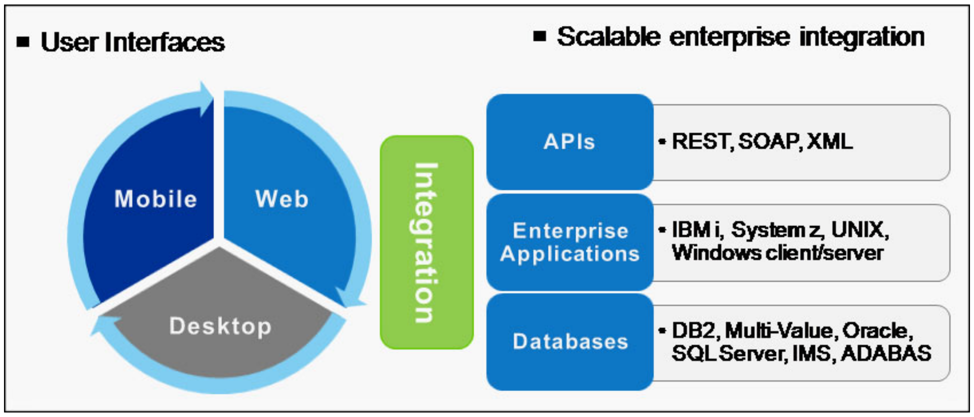

User Interfaces
Scalable enterprise integration
Mobile
Web
Desktop
Integration
APIs
• REST, SOAP, XML
Enterprise Applications
• IBM i, System z, UNIX, Windows client/server
Databases
• DB2, Multi-Value, Oracle, SQL Server, IMS, ADABAS

Figure 1-81 LegaSuite enables user interfaces and a range of application integration

LegaSuite facilitates the following types of application modernization projects:

- Creating easy-to-use, high-performance desktop, web, and mobile browser-based applications that suit the needs of different user audiences.
- Extending enterprise applications beyond the back-office to new user communities and new devices.
- Simplifying applications so that users who are not subject matter experts can use and access them.
- Modifying application workflows to match business processes, increasing efficiency and reducing errors.
- Integrating with other applications and middleware through industry standard APIs such as REST and SOAP.

Solution architecture

LegaSuite is an application platform that developers use to modernize and build applications around mission-critical IBM i, IBM System z®, UNIX, and Microsoft Windows client/server applications.

The LegaSuite workbench includes the UX Designer and the Service Builder:

- ► Developers use the UX Designer to create modernized desktop, web, and mobile user interfaces as HTML5, Java, or Windows clients
- ► Developers use the Service Builder to build and deploy RESTful and SOAP services that are based on existing enterprise applications.

Development workbench

LegaSuite provides an Eclipse-based IDE that developers can use to preform these tasks:

- ► Design easy-to-use applications. Use the WYSWYG page editor, and drag-and-drop controls and widgets.
- ► Integrate user interface controls and widgets with existing applications. Drag and drop database queries, web services, and fields that are automatically generated from the screens in existing enterprise applications.
- ► Build SOA interfaces from existing application screens, programs, and data.
- ► Add custom code to applications and interfaces. See Figure 1-82.

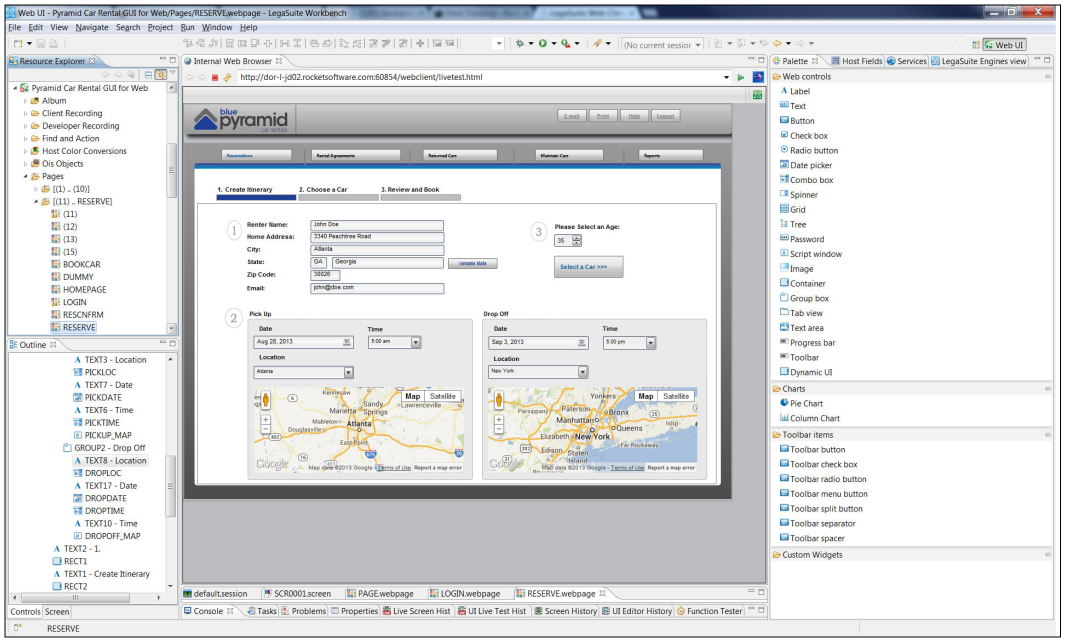


Figure 1-82 LegaSuite development workbench

UX designer

Developers use the UX Designer to define the layout of a new user interface for an existing enterprise application. For desktop and web user interfaces, LegaSuite automatically generates a page layout that is based on the screens of the existing application. For mobile user interfaces, page layouts are independent of existing enterprise applications and use RESTful services to communicate with distributed applications.

The UX Designer also provides the following user interface controls to customize the user experience:

- ► Page headers and footers
- ► Navigation menus
- ► Tree navigation
- ► Buttons and hyperlinks
- ► Grid views and list views

- Text input fields
- Drop-down boxes
- Toggle switches
- Sliders
- Swipeable lists
- Collapsible panes
- Graphs

Desktop user interfaces

LegaSuite can transform green-screen applications into Windows and Java user interfaces for desktop applications, which are based on window maps, business logic, and data from the screens of the existing enterprise application. LegaSuite automatically generates a graphical user interface from text-based screens. Then, developers can customize the generated user interface to optimize application processes and integrate the application with third-party applications, such as Microsoft Office.

Modernized desktop applications are best suited for users who work onsite with enterprise applications for several hours a day and are accustomed to subsecond application response time.

Web user interfaces

Developers can create engaging user experiences that are customized for web browsers and mobile browsers. For example, developers can do the following types of tasks:

- Create web-friendly versions of enterprise applications and deploy them as HTML5 applications.
- Define the behavior of an application when it runs on different browser sizes.
- Integrate third-party controls, applications, and content by using a JavaScript widget.
- Create client-side mashups.

Mobile user interfaces

Developers can build a hybrid mobile interface once and deploy it to many mobile platforms and mobile browsers. Features include:

- Native delivery to iOS, Android, Windows Phone, and any mobile browser that supports HTML5.
- Ready-to-use iOS, Android, and mobile browser skins.
- Automatically changing device panel size and orientation to suit a device.
- Using native device features, such as cameras, location services, and local storage.

Service-oriented architecture (SOA) Interfaces

In addition to user interfaces, LegaSuite can SOA-enable application functions from a wide range of enterprise platforms. Developers can do the following types of tasks:

- Integrate services with application window logic, programs, and data.
- Convert text-based panel functions into production-grade, scalable web services. The developer navigates through the application and selects inputs and outputs.
- Use the recording wizard to capture navigation logic and identify the input and output fields of a host application. LegaSuite packages the navigation sequence as a microflow that can be published as a REST, SOAP, or XML service.
- Deploy SOA interfaces on their own, or incorporate them into an existing architecture.
- Manage various application logic, including list structures and window attributes.

- Create a scalable, high-performance pluggable server architecture that accommodates all enterprise office application integration.
- Compose services from disparate sources into new business services.
- Implement server capabilities such as message transformations, connection pooling, composite services, enterprise data mashups, enterprise level security, logging, auditing, scheduling, publish-subscribe, and queuing

Collector

The Collector expedites development by automatically gathering information about the screens and fields in an enterprise application. Whether an application uses display files (DDS, DSPF) or window maps (BMS, MFS, and so on), developers can use those maps as the basis for a modernization project. Developers can perform the following tasks:

- Use existing application field names, attributes, and hidden or non-display field information in the modernized application.
- Use the layout of the host window as the starting point for the user interface layout.
- Perform user interface development offline without a live connection to the existing enterprise application.

For applications such as those that run on UNIX and do not use window maps or definitions, developers can connect to the host, capture each window, and build the user interface. See Figure 1-83.

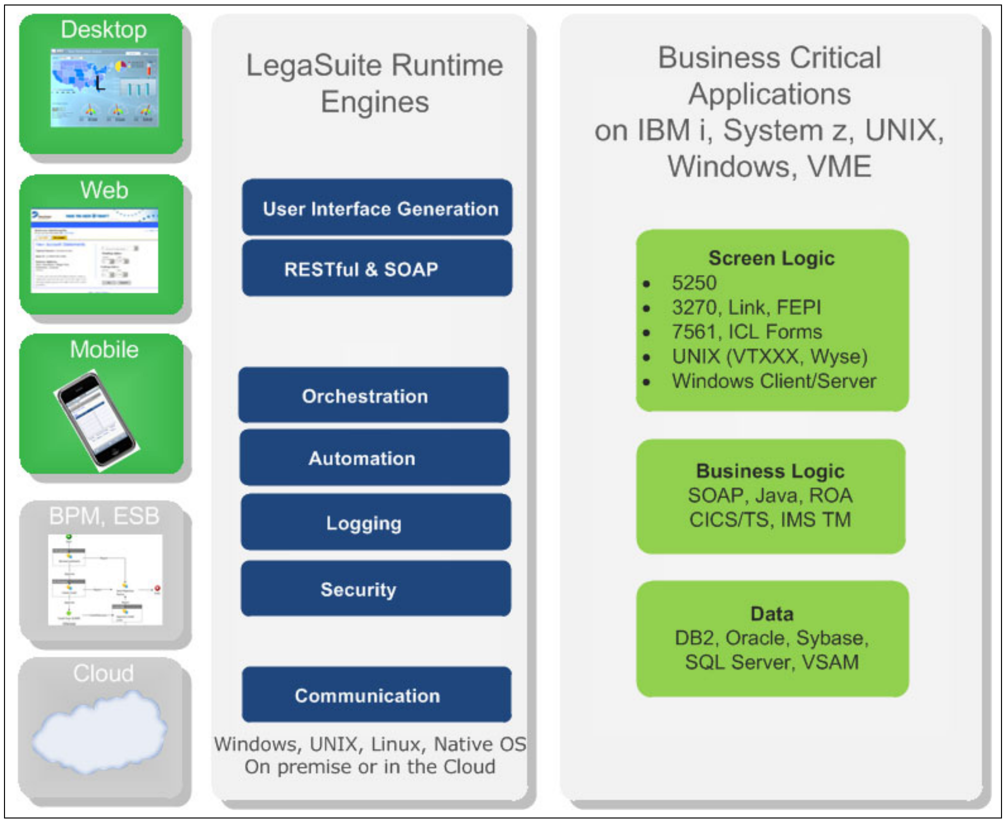


Figure 1-83 LegaSuite architecture diagram

LegaSuite runtime engines

After a developer creates a modernized user experience or SOA interface, the next step is to deploy the interface. With LegaSuite runtime engines, developers can perform these tasks:

- Deploy and run modernized interfaces as an HTML5, Java, or Windows applications
- Deploy and run microflows as REST, SOAP, or XML interfaces

Runtime engines can be deployed in the cloud for ease of access or on premises for tight data control. LegaSuite runtime engines that are run on various operating systems, such as Microsoft Windows, UNIX, and Linux, and on native operating systems such as IBM i5/OS and IBM z/OS®.

LegaSuite runtime has these features:

- Scalability. Session pooling, response caching
- Manageability. REST APIs for management, extensive logging, scheduled downtimes
- High performance.
- Security. Single sign-on, role-based security, and encryption

Security

LegaSuite uses the security of the existing enterprise applications. So users who are not allowed to access functions and data in the existing application cannot access them through LegaSuite.

In addition to using all IBM i, IBM System z, UNIX, and Windows platform security, LegaSuite provides these additional security features at development time and run time:

- Component Security. LegaSuite components are digitally signed, and developers can encrypt user interfaces and SOA interfaces.
- Data Encryption. LegaSuite supports HTTPS, SSL, and SSH encryption at development time and run time to ensure the integrity of in-transit data.

Single sign-on

LegaSuite supports single sign-on (SSO) methods, including Kerberos, Oblix, and IBM Enterprise Identity Management (EIM). All deployment methods use LDAP to access user credentials and other user-specific information. Implementations can include internet user, intranet user, and host user authentication.

Server and Client Management and Monitoring

LegaSuite comes with a secure, HTML-based server management console that administrators use to start, stop, and manage all instances of the run time.

This console is password-protected and can be SSL-encrypted to provide more security during remote administration.

In addition to managing the user interface run time, the console provides logging and troubleshooting capabilities:

- Server configuration logs
- Connection logs
- Event logs
- Server process logs

The server process log tracks specific activities, such as the number of user interface presentations that are delivered, the number of host screens that are navigated, and the number of lines of code that are run. All LegaSuite logging features are also available through a REST API.

Usage scenarios

LegaSuite is an industry-agnostic solution that enterprises use to optimize the user experience for enterprise applications and share business-critical information across an organization. Real-world usage include these scenarios:

- A multi-billion dollar apparel conglomerate uses LegaSuite to deliver a mobile version of a highly customized merchandise management system to hundreds of district and store managers who use the Apple iPad.
- A large vehicle insurer uses LegaSuite in a service-oriented architecture to generate web services from a policy and claims application that runs on IBM i and deliver the services to a .NET-based web portal that insurance brokers use. The company also uses LegaSuite to SOA-enable their IBM i application that delivers insurance quotes to customers who use a web-based self-service application.
- A major financial services company uses LegaSuite to modernize a text-based 5250 interface for a business-critical loan-processing system. Over 1,300 branches of the company use the new user interface, which has dramatically decreased training time and reduced employee turnover.
- A leading cruise line modernized their green-screen reservation system on the IBM i. Now call center representatives can quickly make reservations and take advantage of desktop integration with Microsoft Office applications.

Supported platforms

LegaSuite supports the following platforms.

User interface clients

The following web and mobile browsers are supported:

- Microsoft Internet Explorer
- Mozilla Firefox
- Google Chrome
- Apple Safari
- Opera

The following operating systems (for desktop client deployments) are supported:

- Microsoft Windows Vista, Windows 7, or Windows 8
- Macintosh OS/X with Java 1.6 or higher
- Linux distributions with Java 1.6 or higher

Development: LegaSuite Workbench

LegaSuite Workbench has the following system requirements:

- Microsoft Windows Vista, Windows 7, or Windows 8 Operating System (32- or 64-bit)
- 2 GB RAM minimum required, 4 GB or more recommended
- 2 GB disk minimum required, 10 GB or more recommended

Deployment: Runtime engines

The runtime engines have the following requirements.

Microsoft Windows 2003/2008 / 2008 R2 server (32- & 64-bit):

- Physical and virtual/PaaS deployments are supported
- 2 GB RAM minimum required, 4 GB or more recommended

- 2 GB disk minimum required, 10 GB or more recommended
- IIS 7 or any standard Java Application Server at JVM 1.6 or higher

IBM i with i5/OS V5R4, V6R1, or V7R1:

- Physical and virtual/PaaS deployments are supported
- 2 GB RAM minimum required, 4 GB or more recommended
- 2 GB disk minimum required, 10 GB or more recommended
- Any standard Java Application Server at JVM 1.6 or higher (Apache Tomcat, IBM WebSphere Application Server)

IBM System z, UNIX, and Linux:

- Physical and LPAR/virtual/PaaS deployments are supported
- 2 GB RAM minimum required, 4 GB or more recommended
- 2 GB disk minimum required, 10 GB or more recommended
- System z: Base Linux System SuSE Linux Enterprise Server 11 as a guest OS under IBM z/VM®
- IBM AIX®: Version 6.1
- HP-UX RISC: Version 11.23
- HP-UX IBM IA® (Intel): Version 11.31
- Solaris Sparc or Intel: Version 10
- Linux Intel: Variety of distributions at Kernel 2.4.7-10
- Any standard Java Application Server at JVM 1.6 or higher (ApacheTomcat, IBM WebSphere Application Server)

SOA Interface runtime Engines

Microsoft Windows 2003/2008 / 2008 R2 server (32- and 64-bit):

- Physical and virtual/PaaS deployments are supported
- 1 GB RAM minimum required, 2 GB or more recommended
- 1 GB disk minimum required, 2 GB or more recommended
- JVM 1.6

IBM i with i5/OS V5R4, V6R1, or V7R1:

- Physical and virtual/PaaS deployments are supported
- 1 GB RAM minimum required, 2 GB or more recommended
- 1 GB disk minimum required, 2 GB or more recommended
- JVM 1.6

Linux for System z /UNIX / Linux:

- Physical and LPAR, virtual, or PaaS deployments are supported
- 1 GB RAM minimum required, 2 GB or more recommended
- 1 GB disk minimum required, 2 GB or more recommended
- Linux for System z: Base Linux System SuSE Linux Enterprise Server 11 as a guest OS under z/VM
- IBM AIX: Version 6.1
- HP-UX RISC: Version 11.23
- HP-UX IA (Intel): Version 11.31
- Solaris Sparc or Intel: Version 10
- Linux Intel: Variety of distributions at Kernel 2.4.7-10
- JVM 1.6

IBM z/OS Native CICS

- IBM z/OS - All Versions
- IBM CICS® Transaction Server 3.1, 3.2, 4.1, 4.2, 5.1
- IBM CICS Web support

1.6.2 Contact information

For sales information about products that are discussed in this section, contact Rocket Software at `info@rocketsoftware.com`.

For more information, see the following links:

`http://www.rocketsoftware.com`
`http://www.rocketsoftware.com/brand/rocket-legasuite`

1.7 SystemObjects

This section describes SystemObjects SmartPad4i.

1.7.1 IBM i web and mobile development with SmartPad4i from SystemObjects

Today there are more smartphones and tablets connected to the Internet than PCs and this number is growing fast.

Since 1991, the SystemObjects mission is to provide IBM i developers with development tools to face new challenges. Its flagship product, Delphi/400, modernizes RPG and COBOL business logic with graphical user interfaces, and it is used today by thousands of developers around the world.

To meet requirements for applications on mobile devices, SystemObjects provides SmartPad4i, a solution to create graphical applications for the web, smartphones, and tablets using your existing RPG and COBOL skills.

Business value

Today's businesses have no walls, making it increasingly essential for IBM i developers to create mobile applications for their customers, suppliers, and business partners.

The IBM i is a transactional powerhouse that contains everything a modern server needs. It provides integration, security, resiliency, and scalability and eases of administration. In addition, IBM i developers have proven that they can develop practically any type of application by using RPG and compatible technologies. Their applications embody exceptional knowledge of business rules and mastery of their database structure.

The one thing the IBM i lacks is an intuitive way to build modern user interfaces that can be easily deployed to any browser or mobile device. Because IBM i development teams tend to be small and busy, they have limited time to learn how to develop new mobile apps, especially for all of the operating systems that are required: iOS, Android, BlackBerry, Windows, and so on.

SmartPad4i fills this gap. Built by RPG developers for RPG developers, it is an easy-to-learn tool that handles all of the complexities of developing multi-OS mobile apps. It allows IBM i developers to focus on developing efficient business applications.

SmartPad4i key features

The SmartPad4i product design comes from listening to what customers and prospects say about their development strengths, challenges, and requirements. Its key features deliver fully functional mobile applications after a learning curve that is typically measured in days rather than weeks.

- Develop applications once by using RPG or COBOL and use them everywhere, for mobile devices and Internet browsers; for the PC and Mac; for iOS, Android, BlackBerry, and Windows.
- Work with mobile device features, such as the mobile address book, geo-location, and camera. SmartPad4i applications are "hybrid", containing several standard mobile functions that share data with your IBM i. For example, as a standard feature of SmartPad4i, you can send an email from an RPG program by using the mobile address book.
- There is no need to develop for the mobile operating system. The SmartPad4i global app for iOS, Android, and BlackBerry can be downloaded from any mobile app store by searching for SP4i. After downloading and installing the app, the user enters the address of the web application server only once to access SmartPad4i applications on the mobile device.
- Use a web application server to handle potentially rapid growth in the number of transactions and users.

SmartPad4i architecture

SmartPad4i is composed of three modules:

- Development creates a new web or mobile application.
- Generator generates up to 100% of the code for new standard database applications like lists, updates, and inserts.
- Deployment runs all the SmartPad4i applications from your browser, smartphone, and tablet.

SmartPad4i Development

This module creates any new application in RPG, ILE RPG, COBOL, or CLP. The application can work with multiple web browsers (Internet Explorer, Chrome, Firefox, and Safari) and all smartphones and tablets that have SmartPad4i's pre-packaged global mobile app installed (available for iOS, Android, and BlackBerry).

SmartPad4i Development follows these basic principles:

- Replace DDS with HTML, preserving the separation of the presentation layer and the business logic.
- Remove all DDS constraints, such as the maximum number of rows or columns that are displayed on a page.
- Present data by using HTML components such as images, links, check-boxes, lists, and so on, for a sleek, modern UI.

The code that is generated by SmartPad4i Development manages communication between the HTML page and the RPG program through data structures. It uses simple data structures for record type formats and multiple occurrence data structures for lists. You can display lists with 1 to 9999 rows and any number of columns.

The RPG program treats clickable objects (buttons, images, hyperlinks, and so on) as function keys. You set a return value for each clickable object.

To make it easy to maintain the application, the generated code is one program for one HTML page. You add business logic to the generated program and specify the order of displayed pages by using your RPG or COBOL code. You can enhance the generated source code within the YOURCODE tags. All changes will be preserved during future regenerations of the program. The programs generated by SmartPad4i are standard batch programs and can be debugged by using the STRDBG command. You can call another program that is developed with SmartPad4i by using a simple CALL command, with or without parameters. Similarly, you can call any existing batch program, including batch CL commands.

Data access is done by using record level I/O operations or by using SQL. To display data in an HTML page, you move the data (from the database or from a calculation) to the output data structures. To read user input, you move the data from the input data structures to variables in your program.

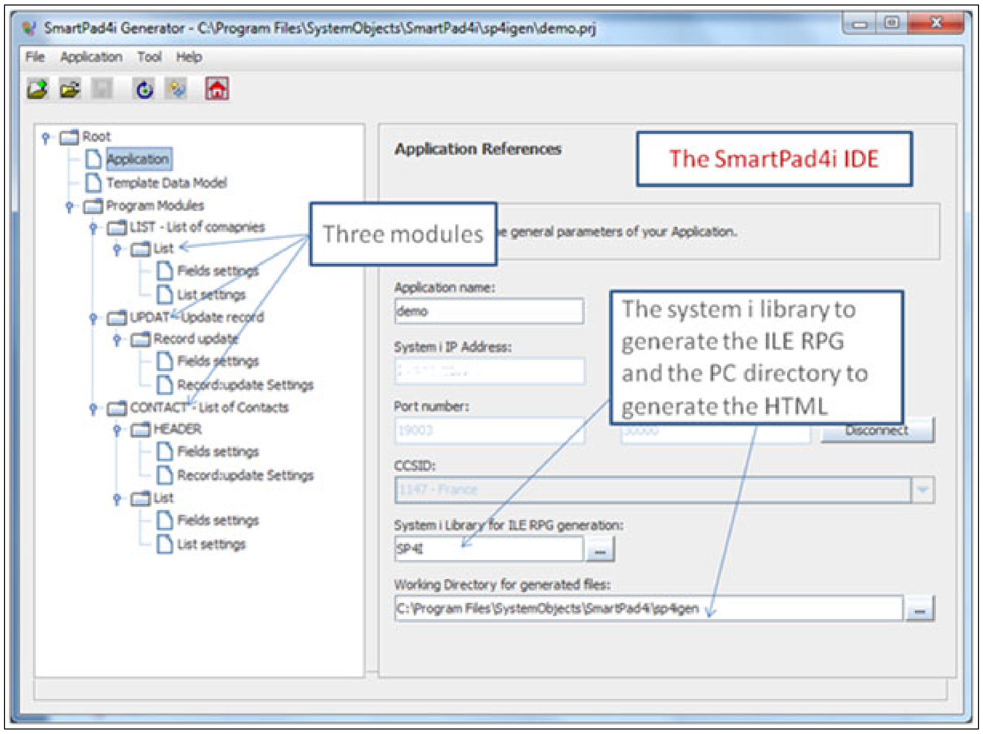
Error management is handled as before. You can highlight any field (by using a CSS class), display an error message, and position the cursor in the field in error.

You can choose whether to display pages of different sizes, based on the page size of the device that is being used.

SmartPad4i supports JavaScript frameworks like JQuery Mobile. It also supports new HTML features such as multi-tab input forms and more.

SmartPad4i Generator

You use the SmartPad4i Generator with its graphical IDE to create applications. Figure 1-84 shows an example of the SmartPad4i Generator IDE.



...e general parameters of your Application.

Application name: demo

System i IP Address:

Port number: 19003

CCSID: 1147 - France

System i Library for ILE RPG generation: SP4I

Working Directory for generated files: C:\Program Files\SystemObjects\SmartPad4i\sp4igen

Figure 1-84 SmartPad4i Generator IDE

An application is a set of modules. A module is:

- A program that is generated in ILE RPG that accesses data by using SQL.
- An HTML page that is generated from a template that incorporates the graphical appearance of your company plus special keywords.

The data is accessed by using a data model:

- A physical file.
- An SQL table.
- A set of joined SQL tables. The definition of these joins is made by using the IDE.

The generated program can display lists of items, update fields in a table, insert a new row in a table, delete a row in a table, or any combination of those functions.

To specify the type of program to generate, you add keywords to the HTML template. For example, inserting a tag for list causes the generator to place your actual list with real data in that specific position. The HTML template can use one or more data models.

Using the IDE, you can define these attributes of the data fields:

- Input, output, or hidden fields
- Color, fonts, styles, and so on, using CSS classes
- Edit code for the data presentation
- Number of rows that are displayed per page for lists
- Sort functions (all columns can be sorted)
- Search function on a column (all columns can be searched)
- Data validation rules
- Calendar for date fields
- Ajax “Suggest Fields” to enable the Google-like function to search a database after each character is entered

The IDE creates the links between modules, with or without passing parameters.

The generated code is ILE RPG. It contains all of the instructions for presenting and updating the data, handling paging lists, editing, sorting, searching, and so on.

SmartPad4i Deployment

SmartPad4i Deployment is responsible for the execution of programs created by SmartPad4i Development or generated by SmartPad4i Generator. The intelligence in Deployment makes it possible for all Internet browsers, smartphones, and tablets to use the same program.

To handle peak loads or many simultaneous hits, SmartPad4i uses a web application server such as the following servers:

- IBM HTTP Server and IBM WebSphere Application Server
- IWAS, Integrated Web application server for i
- Apache Tomcat

SmartPad4i Deployment performs the following functions:

- Creates a job in the SmartPad4i subsystem for each Internet or mobile user
- Controls access by using IBM i profiles
- Manages the link between the HTML page and the program

- Cleans up potential problems that are caused by Internet access. SmartPad4i generates an ENDJOB to finish the job in progress and close open tables, unlock records, run the COMMIT/ROLLBACK function, and suppress any ghost jobs from running on the IBM i.

The processor consumption of a program that is generated by SmartPad4i is similar to programs that run in 5250 mode.

Your first SmartPad4i application

This section provides an example of how to create a SmartPad4i application. All of the source files for this example can be downloaded at:

`http://www.systemobjects.com/helloi.zip`

Installation and setup

To create your first application, use SmartPad4i Express. SmartPad4i Express contains the entire SmartPad4i product, which is bundled with a Tomcat 7.0 server for your PC. The SmartPad4i Express version is easy to install and circumvents any potential issues that are related to installation, configuration, and update of your web application server.

The SmartPad4i Express version installs these items:

- Libraries on your IBM i
- Two subsystems on your IBM i
- Tomcat 7.0 server on your Windows PC
- The entire SmartPad4i product on your PC

The installation starts the subsystems. You might want to consider changing the IBM i IPL procedure so that it also starts the SmartPad4i subsystems. Use a user profile with QSECOFR authority to allow the installation procedure to create libraries and subsystems.

After installation, the SmartPad4i folder appears on your PC's Start menu (Figure 1-85).

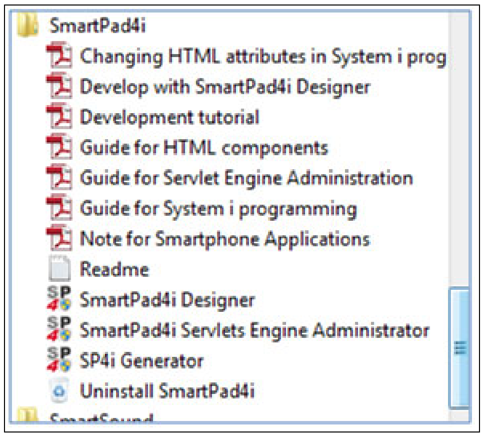

SmartPad4i
Changing HTML attributes in System i prog
Develop with SmartPad4i Designer
Development tutorial
Guide for HTML components
Guide for Servlet Engine Administration
Guide for System i programming
Note for Smartphone Applications
Readme
SmartPad4i Designer
SmartPad4i Servlets Engine Administrator
SP4i Generator
Uninstall SmartPad4i

Figure 1-85   The SmartPad4i Folder

Use the Guide for Servlet Engine Administration to configure the web application server.

You should have these directories on your system (Figure 1-86).

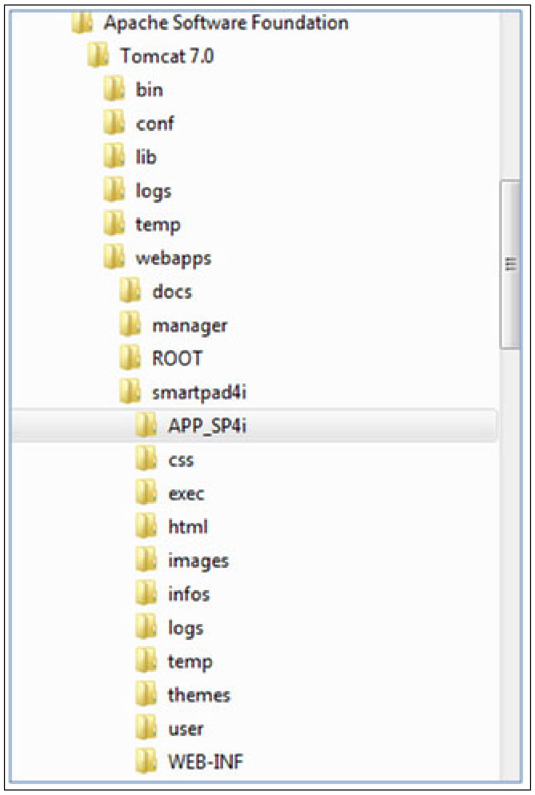


Figure 1-86 The installed directories

Developing your first application

This section explains these procedures:

- Create your first HTML page
- Create your first SmartPad4i RPG program
- Deploy the program on your web application server
- Run the program in your browser
- Run the program with your smartphone or tablet with the SP4i app for Android, iOS or BlackBerry

1. Create your first HTML page

   Using your favorite HTML editor, create a page that is named first_page.html. In this page, define these objects:

   – A form

   – Two input fields

   – Two buttons

   The simplest way to create these objects is to add a table with four rows and two columns inside your form. Using the editor that is shown in Figure 1-87 on page 95, drag the objects from the right column onto the table.

   To ensure that your page resizes itself to your screen, browser or smartphone, add the following line before the </head> tag:

```
<meta name="viewport" content="width=device-width" />
```

To add a title, insert a <div> tag above your table. For a large title, use the <h1> tag.

Figure 1-87 shows the HTML page in the development environment.

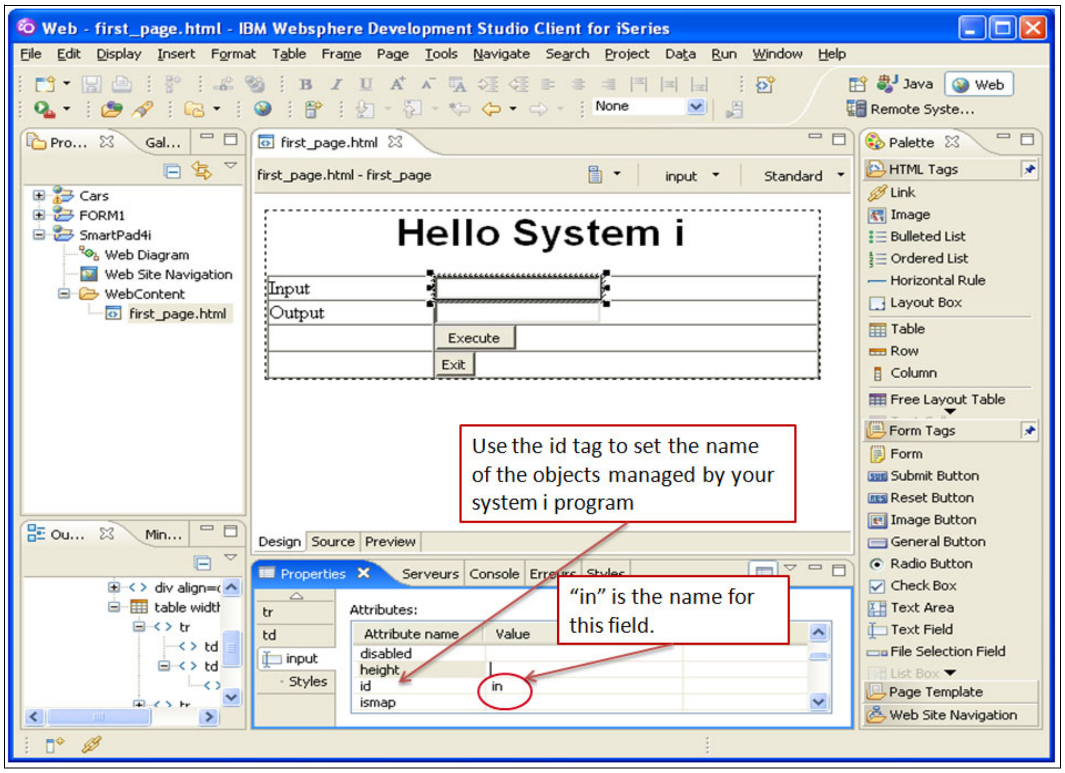


Figure 1-87 WebSphere Development Studio

Using this HTML form, the SmartPad4i program reads the first field and displays the result in the second field when the user clicks the **Execute** button.

You need to give a unique name to each object on this HTML page:

– The first input field is named **`in`**
– The second field is named **`out`**
– The first button is named **exec**
– The second button is named **exit**

Here is a part of the HTML code (Example 1-6)

Example 1-6 Part of the HTML code

```
<form >
<div align="center"><h1>Hello System i</h1></div>
<table width="100%" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td width="30%">Input</td>
    <td width="70%"><input type="text" name="in" id="in"></td>
  </tr>
  <tr>
    <td>Output</td>
    <td><input type="text" name="out" id="out"></td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="button" name="exec" id="exec" value="Execute"></td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="button" name="exit" id="exit" value="Exit"></td>
  </tr>
</table>
</form>
```

Next, create a directory in your web application server under the html directory. Name the directory HELLOI. This name can be the name of the library that is used to store your source programs on your IBM i. Create this library on your IBM i system.

On your PC, you should have a directory structure like this:

`\Apache Software Foundation\Tomcat 7.0\webapps\smartpad4i\html\HELLOI`

Store the `first_page.html` file in the HELLOI directory of the web application server on your system.

2. Create your first SmartPad4i RPG program. In the SmartPad4i Designer, complete these steps:

   – Using the File menu, create a new project named Helloi.

   – Log on to your IBM i system.

   – In the **Options** menu, select **Select Html File Pathname** (Figure 1-88):

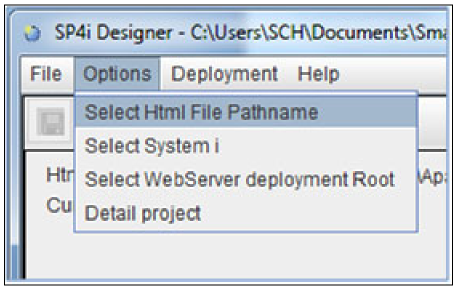

SP4i Designer - C:\Users\SCH\Documents\Sm
File
Options
Deployment
Help
Select Html File Pathname
Select System i
Select WebServer deployment Root
Detail project

Figure 1-88 SmartPad4i Options

   – Choose the HELLOI directory under the html directory.

   – In the **Options** menu, click **Select WebServer deployment Root**.

   – Choose the `webapps` directory under the Tomcat 7.0 directory.

   – Click the **+** button under the Deployment menu (Figure 1-89):

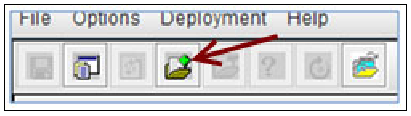
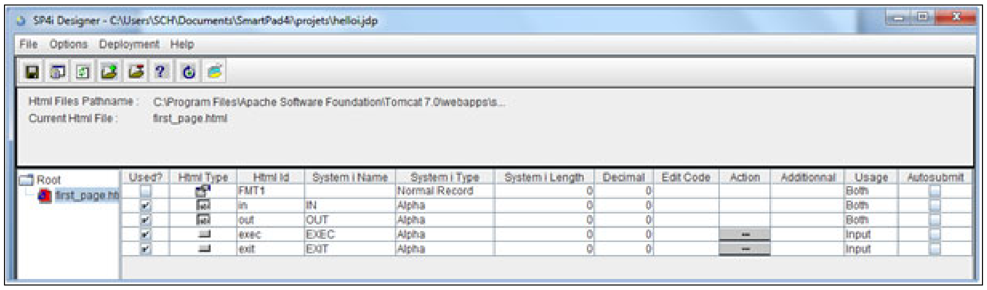


Figure 1-89 Fig6: SmartPad4i Menu

3. Select first_page.html to open the window shown in Figure 1-90.


SP4i Designer - C:\Users\SCH\Documents\SmartPad4i\projets\helloi.jdp
File Options Deployment Help
Root
first_page.ht

Html Files Pathname : C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\s...
Current Html File : first_page.html

Used?	Html Type	Html Id	System i Name	System i Type	System i Length	Decimal	Edit Code	Action	Additionnal	Usage	Autosubmit
☐		FMT1		Normal Record	0	0				Both	☐
✓		in	IN	Alpha	0	0				Both	☐
✓		out	OUT	Alpha	0	0				Both	☐
✓		exec	EXEC	Alpha	0	0		...		Input	☐
✓		exit	EXIT	Alpha	0	0		...		Input	☐

Figure 1-90 SmartPad4i Designer

As you can see, each object on the HTML page is on a separate line in the SP4i Designer.

– In the IBM System i® Length column, enter 20 for the two fields, IN and OUT.

– For the two buttons (EXEC and EXIT), click the button in the Action column. Leave the value set to 1 for EXEC and enter the value F3 for EXIT. In the RPG program, the JCACTN variable is used to indicate which button the user selected.

– Save your project by using the first icon under the menu bar.

– Click the **Deployment** menu and select **Deploy Project to the System i** as shown in Figure 1-91.

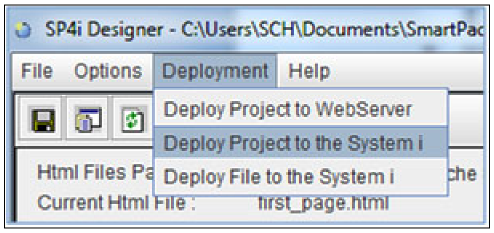
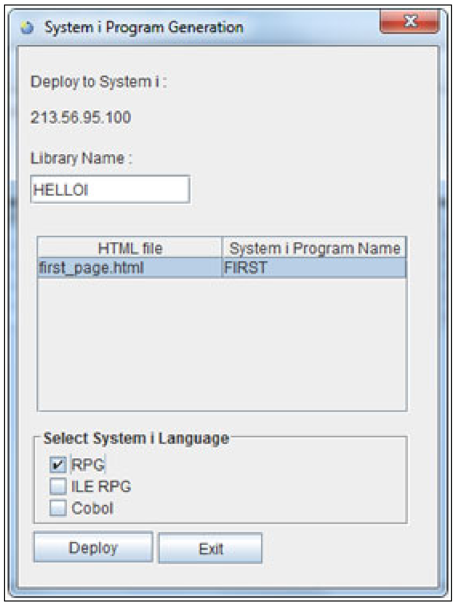

SP4i Designer - C:\Users\SCH\Documents\SmartPac
File Options Deployment Help
Deploy Project to WebServer
Deploy Project to the System i
Deploy File to the System i
Html Files Pa
che
Current Html File : first_page.html

Figure 1-91 SmartPad4i Deployment menu

Next, as shown in Figure 1-92:

– Enter the library name HELLOI
– Enter the program name FIRST, then press **Enter**
– Select the language **RPG**



Deploy to System i :

213.56.95.100

Library Name :

HELLOI

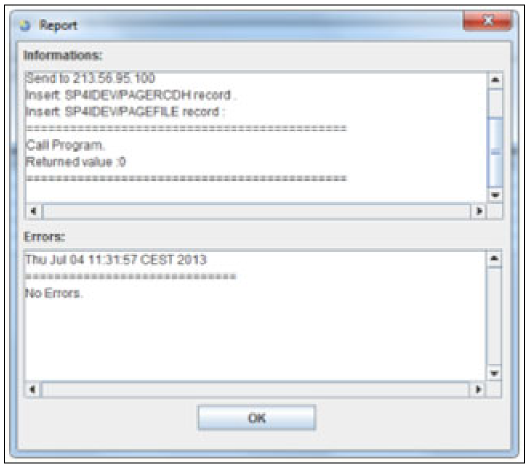
HTML file	System i Program Name
first_page.html	FIRST

Select System i Language

[✓ RPG]
ILE RPG
Cobol

Figure 1-92 SmartPad4i Program Generation Screen

– Click **Deploy** to get this window (Figure 1-93).



Informations:

```
Send to 213.56.95.100
Insert SP4IDEV/PAGERCDH record .
Insert SP4IDEV/PAGEFILE record :
================================================
Call Program.
Returned value :0
================================================
```

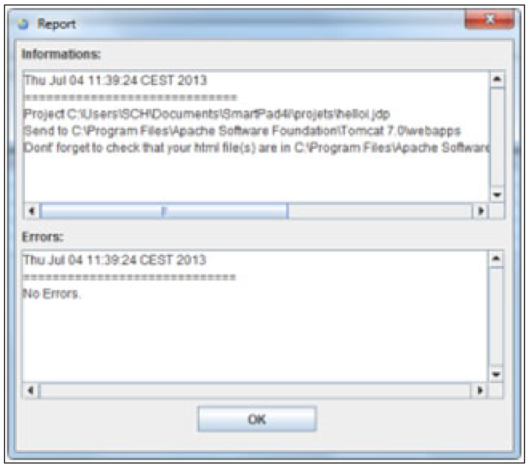
Errors:

```
Thu Jul 04 11:31:57 CEST 2013
================================
No Errors.
```

Figure 1-93 Program Deployment

– Click **OK**.
– Click the **Deployment** menu and select **Deploy Project to WebServer**.

If the deployment was successful, you see the window shown in Figure 1-94).



```
Thu Jul 04 11:39:24 CEST 2013
==============================
Project C:\Users\SCH\Documents\SmartPad4i\projets\helloi.jdp
Send to C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps
Dont' forget to check that your html file(s) are in C:\Program Files\Apache Softwar
```

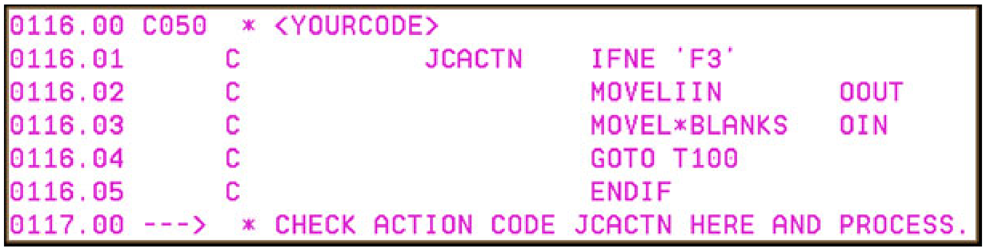
```
Thu Jul 04 11:39:24 CEST 2013
==============================
No Errors.
```

Figure 1-94 Web Application Server Deployment

- Click **OK**
- You are finished. You are done with SmartPad4i Designer and this page.

You have created two programs in the HELLOI library: an RPG program that is named FIRST and a CLP program that is named SETENV, which is already compiled. You can run it to put your library in the library list.

- Edit your FIRST program in the HELLOI library.
- Under the line 0116.00, enter the code shown in Figure 1-95.



```
0116.00 C050  * <YOURCODE>
0116.01       C            JCACTN    IFNE 'F3'
0116.02       C                      MOVELIIN       OOUT
0116.03       C                      MOVEL*BLANKS   OIN
0116.04       C                      GOTO T100
0116.05       C                      ENDIF
0117.00 --->  * CHECK ACTION CODE JCACTN HERE AND PROCESS.
```

Figure 1-95 The RPG generated code

- The first line tests whether the JCACTN variable is filled with the return value (specified in the SP4i Designer) when the user clicks a button.
- If it is not the F3 return value:
  - Move the input value in the first field to the second field (this program is a trivial program)
  - Clear the first input field
  - Go to the T100 tag to send the HTML page back to the browser.
- You named the first field IN. However, the SP4i program uses this field as an external data structure. SP4i adds the prefixes "I" for input and "O" for output. For that reason, the program reads IIN, writes to OOUT, and clears OIN.
- Compile your program to complete the process.

4. Deploy the program on your web application server.

To run your program in your browser and to prepare to run it in your smartphone or tablet, you need to customize the start.html page that is stored in the APP-SP4i directory that is installed on your web application server.

In this directory, you find two start page templates:

– Select `start_logon.html` if you want to prompt for the IBM i user ID and password before you run the program

– Select `start_nologon.html` if you want to use the autologon feature of SmartPad4i

In this tutorial, use the `start_logon.html` file to prompt for the user ID and password. Make the following changes to the file and save it:

– Change the PGMAPP value to FIRST

– Change the LIBENV value to HELLOI

– Example 1-7 shows the code and the changes that you need to make:

Example 1-7 Code and changes that you need to make

```
<input name="PGMAPP" type="hidden" id="PGMAPP" value="FIRST" size="10">
<input name="LIBAPP" type="hidden" id="LIBAPP" value="*LIBL" size="10">
<input name="PGMENV" type="hidden" id="PGMENV" value="SETENV" size="10">
<input name="LIBENV" type="hidden" id="LIBENV" value="HELLOI" size="10">
<input name="PARM" type="hidden" id="PARM" size="1"></td>
```

– Start the Tomcat 7.0 service on your PC if needed.

5. Run the program in your browser

– In your browser, enter the following URL:

`http://localhost:8080/smartpad4i/APP_SP4i/index.html`

– The window shown in Figure 1-96 opens.

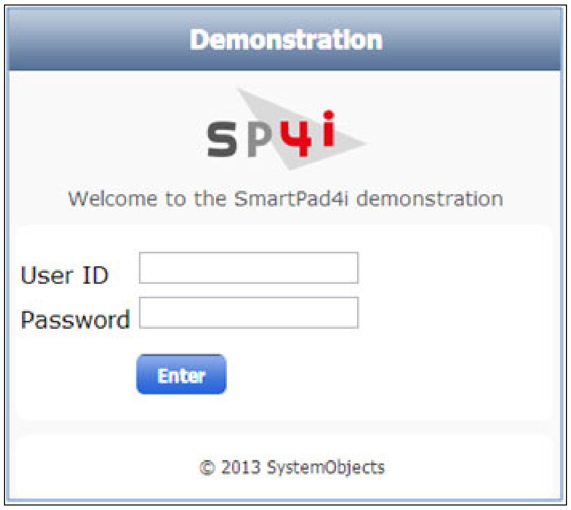


Figure 1-96 SP4i main menu

– Click **Set Your Server Address** and enter the addresses of your web application server. A start page opens as shown in Figure 1-97.

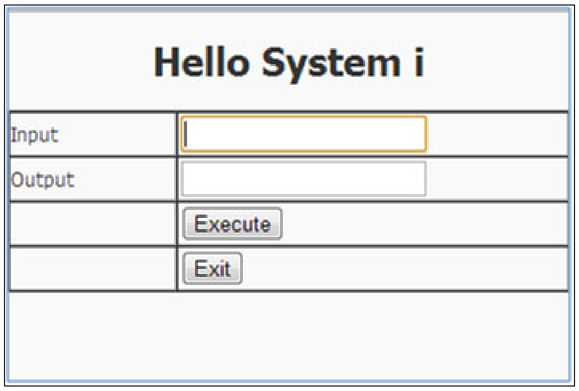


Figure 1-97 The Hello I Program

6. Run the program with your smartphone or tablet

   To run this application on your smartphone or tablet:

   – Download and install SP4i from iTunes for iOS, Google Play for Android, or BlackBerry World.

   – Make sure that the PC that is running your web application server has a public IP address. If your PC does not have a public IP address and you are connected by WiFi, you can proceed if your smartphone and your PC are in the same domain.

   – Run the SP4i application on the smartphone.

   – Change the addresses as shown in the previous “Run your program in your browser” section, changing *localhost* to the IP address of your PC that is running the web application server.

   – Run the SmartPad4i application on your smartphone.

You have completed your first SmartPad4i mobile application.

Supported Platforms

SmartPad4i supports these platforms:

- Internet Explorer, Chrome, Firefox, Safari
- All iOS devices
- All Android devices
- All BlackBerry devices (OS 10)

SmartPad4i works with these applications:

- IBM i V5R1 and later
- WebSphere Application Software V3.5 and later
- iWAS V6R1 and later
- Tomcat V4.0 and later

Ordering information

You can request a demonstration and free trial version from the SmartPad4i website:

http://www.systemobjects.com/homesp4i.html

To request a demonstration, see the following website:

http://www.systemobjects.com/redbook_demo.html

To order SmartPad4i, contact SystemObjects at info@systemobjects.com. Product licensing consists of the following items:

- Development

  A single Development license includes SmartPad4i Development and SmartPad4i Generator for any number of developers. It covers all development and testing activity on any single IBM i partition or system, regardless of model or software group.
- Deployment

  A single Deployment license includes SmartPad4i Deployment for any number of users accessing SmartPad4i applications that are running on one IBM i partition or system, regardless of model or software group.
- For High Availability environments, an optional Backup Deployment license is also offered.

1.8 Lansa

This section describes LANSA LongRange, RAMP, and Visual LANSA.

1.8.1 LongRange

LongRange is the first native mobile application builder for the IBM i community that enables developers to use their existing RPG, COBOL, and DDS skills to quickly create native mobile applications for iPhone, iPad, and Android devices that fully integrate with their IBM i application and data. LongRange does not require programming on the mobile device, which means developers do not need to worry about learning new languages such as Java (Google Android), Objective-C (Apple iOS), or other coding range techniques like HTML, CSS, and JavaScript. However, LongRange applications can still take photos, record audio or videos, capture signatures, scan bar codes, synchronize documents, use maps and geo-location information, and integrate those features with IBM i data and applications.

What is LongRange?

LongRange consists of a server-side management service (LongRange Server), an app that runs natively on a mobile device (LongRange mobile app) and a developer tool (LongRange Studio) as shown in Table 1-3.

Table 1-3 LongRange components

Software	Description
LongRange server	LongRange Server is the interface between IBM i programs and the LongRange mobile app. It manages requests from the app and calls the appropriate IBM i programs.
LongRange mobile app	LongRange mobile app is the intermediary between the mobile device user and the LongRange server. It responds to user actions and displays screens that are generated by IBM i programs.
LongRange Studio	LongRange Studio is the tool that developers use to define the application model and content in an application schema.

To use the LongRange application, the team must complete these steps:

- Administrators download and install LongRange Server on an IBM i server.
- Developers download and install LongRange Studio on a desktop or notebook computer.
- Users download the LongRange mobile app to their mobile device, connect to the server, and are then ready to use the application.

How does LongRange work?

The LongRange mobile app runs on both Apple iOS and Android mobile devices. The LongRange Server runs on an IBM i server. LongRange mobile apps securely connect to the LongRange Server to run IBM i programs that are written in RPG, COBOL, or CL.

When a mobile device user calls a function in the LongRange mobile app, the action sends a request to the LongRange Server, which calls the associated RPG, COBOL, or CL program. The program does its processing and outputs a data stream of information through XML. The LongRange Server then sends the data stream to the LongRange mobile app, which presents the information about the mobile device. The LongRange mobile app is faster than browser-based mobile apps when it renders screens and responds to user actions because the app is designed to run natively on the mobile device.

Figure 1-98 illustrates the LongRange deployment.

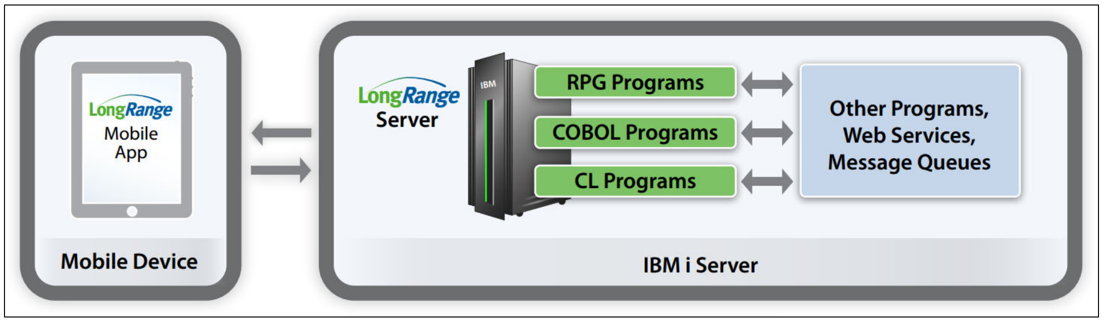


Figure 1-98 How LongRange works

The LongRange mobile app

LongRange is a native mobile app builder that is composed of panes for Navigation, Tabs, Commands, and a Content Area where developers add form views (screens that are generated by IBM i programs), web views (HTML applications, pages, and sites), and document views. This configuration allows developers to add and change programs without having to redistribute the whole application.

Figure 1-99 shows the layout of the LongRange Mobile App.

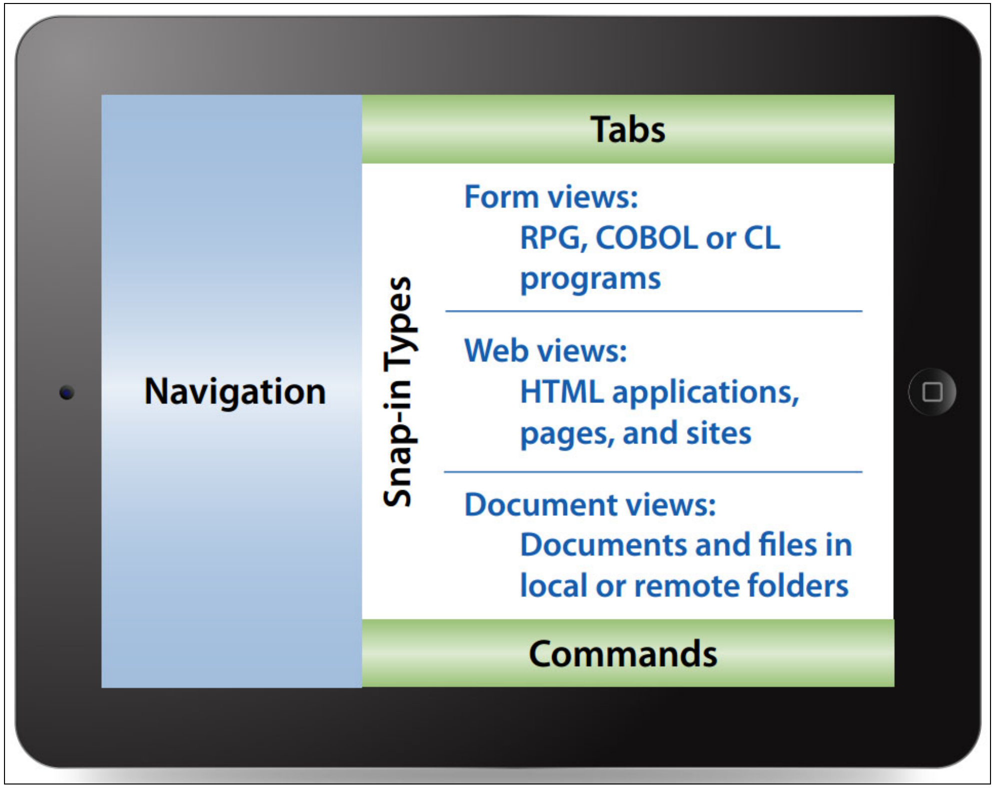

Tabs
Navigation
Snap-in Types
Form views:
RPG, COBOL or CL programs
Web views:
HTML applications, pages, and sites
Document views:
Documents and files in local or remote folders
Commands

Figure 1-99 LongRange Mobile App layout

Table 1-4 describes the elements of the layout and explains their purpose.

Table 1-4 Elements of the layout and their purpose

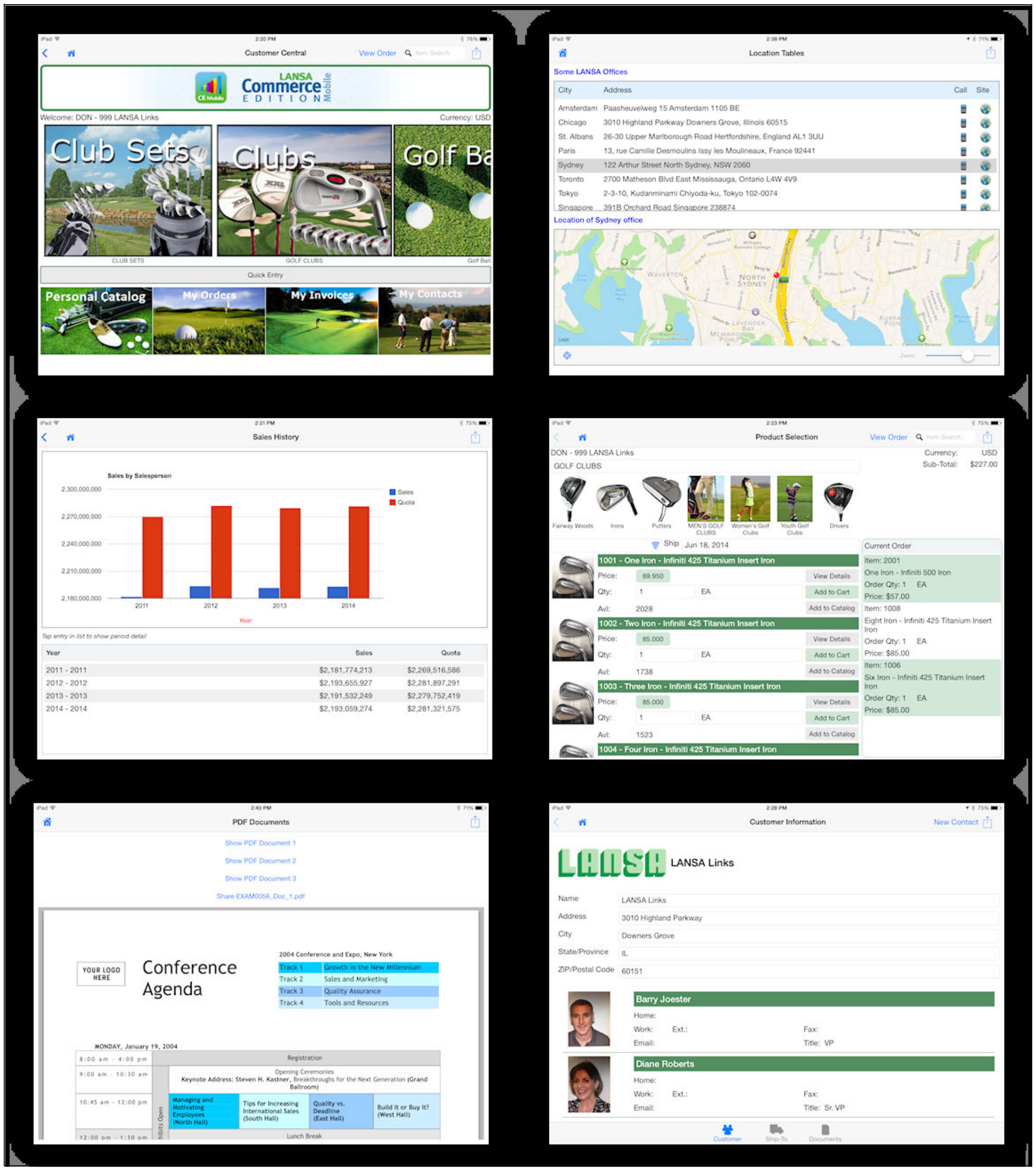
Element	Purpose
Commands	Commands are user actions; examples are save and cancel.
Document views	Document views show files and folders either on the mobile device or on a remote server.
Form views	Form views are screens that are generated by RPG, COBOL, or CL programs that the LongRange mobile app renders on the mobile device.
Navigation	Navigation is the equivalent of menus and shows applications accessible from the mobile device.
Tabs	Tabs provide concurrent views of information from multiple IBM i programs and allow users to quickly switch between views.
Web views	Web views are HTML applications, web pages, or sites.

Form views typically contain data and labels (text that describes the data). Developers can build applications using RPG, COBOL, or CL programs that are started by using form views, or composite applications that are using combinations of form views, web views, and document views.

LongRange includes a feature that is called “document view” that displays documents and files that are in folders that are stored on the mobile device or on the IBM i server. Users can take photos and store them in folders, accept documents from other apps on the mobile device, and save them in folders on the device or a server. The LongRange document view element can synchronize documents that are created on the mobile device with the IBM i server, and vice versa.

Characteristics of a LongRange application

LongRange operates and looks like any other native mobile application. It processes input and output from IBM i applications with an enhanced user interface on Apple iOS and Android mobile devices. See Figure 1-100.



Some LANSA Offices

City	Address
Amsterdam	Paasheuvelweg 15 Amsterdam 1105 BE
Chicago	3010 Highland Parkway Downers Grove, Illinois 60515
St. Albans	26-30 Upper Marlborough Road Hertfordshire, England AL1 3UU
Paris	13, rue Camille Desmoulins Issy les Moulineaux, France 92441
Sydney	122 Arthur Street North Sydney, NSW 2060
Toronto	2700 Matheson Blvd East Mississauga, Ontario L4W 4V9
Tokyo	2-3-10, Kudanminami Chiyoda-ku, Tokyo 102-0074
Singapore	391B Orchard Road Singapore 238874

Tap entry in list to show period detail

Year	Sales	Quota
2011 - 2011	$2,181,774,213	$2,269,516,586
2012 - 2012	$2,193,655,927	$2,281,897,291
2013 - 2013	$2,191,532,249	$2,279,752,419
2014 - 2014	$2,193,059,274	$2,281,321,575

Name	LANSA Links
Address	3010 Highland Parkway
City	Downers Grove
State/Province	IL
ZIP/Postal Code	60151

Figure 1-100 Enhanced user interface

A LongRange application has these key features:

- The app behaves the way a mobile device user expects.
- The IBM i user interface is extended with touch and enhanced visually by mobile devices.

- Applications can send and receive files between a mobile device and a server.
- IBM i applications can use the device-side capabilities, including the camera and geo-location services.

The DDS provides the layout and data content that the LongRange mobile app displays on a mobile device. Screen layouts automatically adapt from portrait to landscape and landscape to portrait as the mobile device changes orientation. Developers do not need to design layouts and content for each screen size and orientation because layouts and content automatically adjust when displayed on small or large screen sizes.

Prototyping and designing applications

You can use LongRange Studio to rapidly create a complete application prototype and demonstrate it to users. Building a prototype helps to define the parts that need to be built, estimate the programming effort, and identify what is missing from the design.

Prototypes allow users to understand how an application operates and see what the screens look like without having to build the entire application, aligning user and developer expectations of the application.

After approval of the prototype, developers build the application progressively, adding each completed program into the prototype and gradually turning the prototype into a working application.

This development methodology allows users to see an application that grows and adapts quickly to changing requirements.

Application development

To build an app using LongRange, developers create application content such as forms (or screens) in an IBM i program and an application schema with LongRange Studio.

IBM i development tools

Developers can use RPG, COBOL, and LANSA to develop applications for use with LongRange. These applications can also call other programs on IBM i servers, use data queues, interact with the database, and more. From a developer's perspective, the applications are just like any other IBM i application.

LongRange complements existing developer tools. Therefore, developers can use their existing IBM i application development tools and LongRange Studio to develop applications for use with the LongRange mobile app.

Developers who are familiar with the Programming Development Manager (PDM) and the Source Entry Utility (SEU) require a developer workstation, LongRange Studio, and terminal access to PDM and SEU. See Figure 1-101.

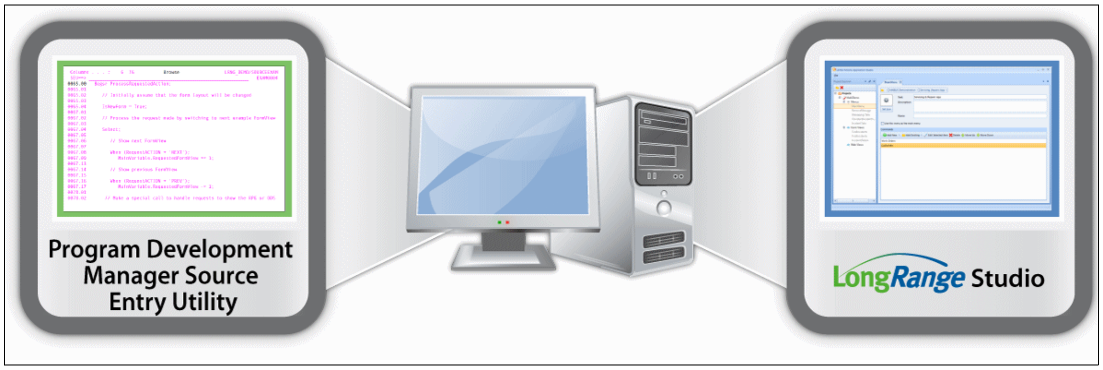


Figure 1-101 Requirements for PDM and SEU developers

Developers who are familiar with the IBM Rational Developer for Power Systems™ require a developer workstation, LongRange Studio, and the IBM Rational development tools. See Figure 1-102.

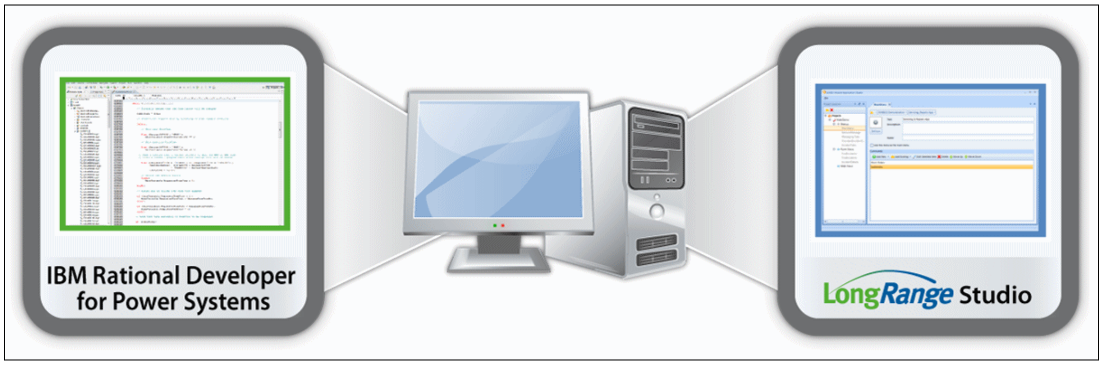


Figure 1-102 Requirements for IBM Rational developers

LongRange Studio

LongRange Studio is a development tool that is used to define the static parts of your application. These include the application menus, the caption and icon of the menu items, the caption of a form, which program to call when a menu is selected, and various other things. See Figure 1-103.

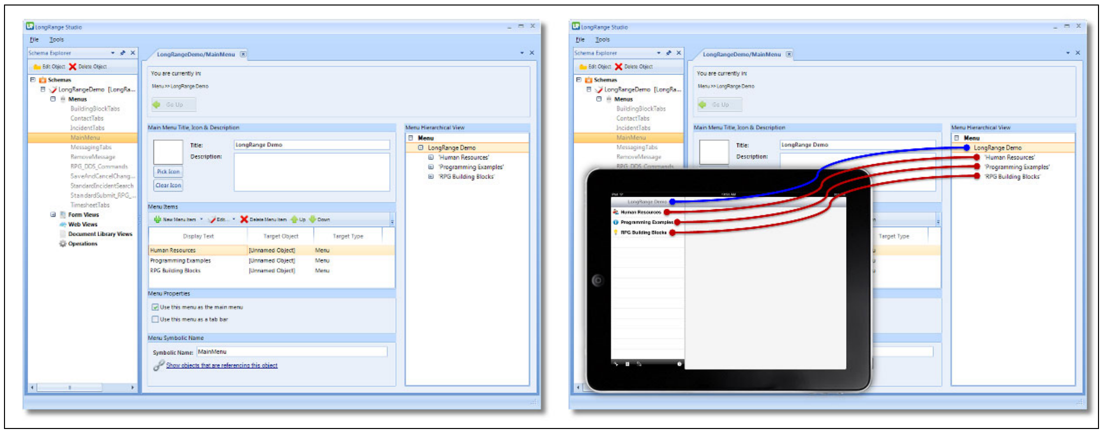


Figure 1-103   LongRange Studio captures

Developers use LongRange Studio to describe apps by building application schemas. Application schemas define the static parts of an application:

- Navigation. Menu structure hierarchy and menu item icons
- Form Views. Typical forms that show visual controls like labels and text boxes that present information to users
- Web Views. HTML, CSS, and JavaScript pages
- Document Views. File and folder explorers show lists of files and folders that contain documents, presentations, photos, and movies (videos) that are associated with the application
- Commands. Actions available to users such as save or cancel

The application schema also includes a reference to the IBM i program to call when a user requests a form view.

Access to device-side features

Applications that use LongRange have access to files that are stored on mobile devices, including photos and data. Suppose a business requirement of an insurance claim system is to capture a photo of a motor vehicle accident scene. An assessor with a mobile device can take photos, then use the application to save the photos in the database on an IBM i server, or save the files in a folder in the server's file system.

Sending files to a mobile device from the server

This feature of LongRange expands the capabilities of IBM i applications beyond the information displayable on a window. With access to device-side features, including the camera, geo-location services, and files that are stored on the mobile device, applications can capture, manage, and display not only screens, but also photos, documents, maps, and mobile device geo-location information.

Sample applications and templates

LongRange includes an extensive collection of sample applications and templates. The sample applications range from the introductory to fully functional applications. When you run a sample application you can see not only the screens and user actions, but also view the program code and data description specifications.

Testing and debugging

Developers need Apple iOS or Android mobile devices to test their applications.

Developers can use their existing debugging tools to track down errors in an application. Finding and debugging errors in applications that are developed by using LongRange is no different from finding and debugging errors in any other application.

To assist developers in finding errors, LongRange provides two levels of tracing: one at the application level and a second at the system level. Developers can use application tracing during testing or debugging to track activity of individual applications (or programs). System-level tracing has a wider scope and produces much more trace data than application-level tracing.

Deployment

Deploying the LongRange mobile app is as simple as downloading the app from the Apple App Store or Google Play and configuring communications with an IBM i server.

Updating the mobile app is the same as enhancing and maintaining any other IBM i application. After you deploy application updates to your production system, they are immediately available to users of the LongRange mobile app, without having to download or update anything on the mobile device.

Security

Security is an integral part of LongRange and is designed to protect data and applications while making those applications available to remote users who are operating mobile devices. LongRange supports a number of physical implementations, each intended to provide different levels of scalability and performance without compromising security. LongRange has the following security support and requirements:

- LongRange supports the IBM i security and authentication mechanisms up-to and including the highest security level (level 50).
- User identification and passwords are encrypted before transmission.
- User identification and passwords cannot be cached on the device.
- LongRange supports log-in from specific IP addresses.
- LongRange supports log-in from specific device names.
- LongRange supports reverse proxy.

LongRange Server supports SSL and Transport Layer Security (TLS) protocols, which allow for secure authentication, encryption, non-repudiation, and VPN technologies. Both SSL and VPNs prevent eavesdroppers from listening and intercepting traffic between a browser and a server.

Communications traffic between a mobile device and the server is compressed, which offers two major benefits. First, traffic is harder to monitor with sniffer devices. Second, when using SSL or VPN technologies, the amount of data that is transmitted between a server and a mobile device is reduced before the SSL or VPN encryption or decryption processing takes place. Compressing communications traffic achieves considerable savings in bandwidth and processor cycles.

The applications run on the server, so no business logic runs on the mobile device. Therefore, applications and the database are not exposed to the Internet or internal networks.

Consistent quality and application management

Developers are not always the best designers of user interfaces, and even the ones who can produce an acceptable user interface have differing ideas about how to lay out a window.

The LongRange mobile app is built to the operational and user interface design standards of mobile device platforms. It manages the user interface for the developer. IBM i developers using LongRange always produce a consistent user interface that conforms to the quality standards of the mobile device platform.

Application management is a simple task for administrators and developers. Because LongRange can be downloaded from an app store, administrator workload is reduced because they do not need to schedule and rollout app upgrades. Applications that are built for use with LongRange need no special management. Their deployment is the same as any other IBM i application, and developers and administrators are already familiar with these processes.

Ongoing maintenance and enhancement

Developers maintain and enhance IBM i applications that are built for use with LongRange by using the same tools and methods they use with their existing IBM i applications.

Online documentation and trial software

Companies can trial fully functioning versions of LongRange Studio and LongRange Server free for 45-days before licensing the software.

Trial software and online documentation is available at:

`http://www.longrangemobile.com`

1.8.2 LANSA Rapid Application Modernization Process (RAMP)

Introduction to RAMP

The LANSA Rapid Application Modernization Process (RAMP) is a single, integrated solution that addresses both tactical and strategic modernization requirements. RAMP is a reengineering product that allows companies to rapidly consolidate their applications into a graphical application framework and then incrementally replace heritage programs with new, reengineered components.

RAMP is the perfect hybrid between these two drastically different approaches: rewriting and refacing. The RAMP framework consolidates function from refaced existing applications and newly written components into one composite application that enables IBM i organizations to attack their immediate tactical issues while providing a long-term redevelopment strategy.

Modernizing with RAMP

RAMP provides an iterative and incremental application modernization approach to deliver a modernized version of companies' existing applications and a foundation for reengineering, enhancements, and new development.

Companies can easily repurpose the valuable parts of their existing applications into a new modern solution without the developmental obstacle of rigid business processes.

Companies can then gradually replace parts of their heritage applications with new components built by using the powerful LANSA development tools without the cost and time of a complete rewrite. See Figure 1-104.

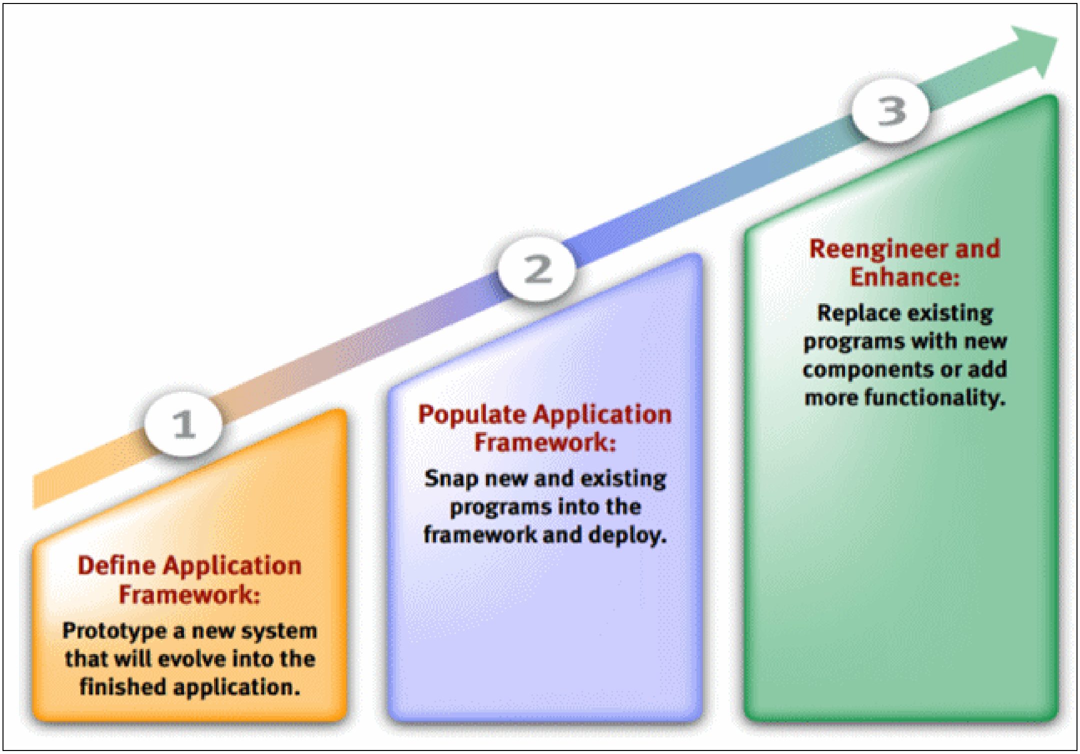

1
Define Application Framework:
Prototype a new system that will evolve into the finished application.
2
Populate Application Framework:
Snap new and existing programs into the framework and deploy.
3
Reengineer and Enhance:
Replace existing programs with new components or add more functionality.

Figure 1-104 Gradually replacing parts of heritage applications

Step 1: Define the application framework

RAMP takes application modernization to an entirely new level by using an Instant Prototyping Assistant to generate a working prototype of a new application. Upon completion, the prototype is a fully executable application that can be presented to users to gather their feedback. All of the application navigation, organization, integration, user settings, and security are built into the framework.

Because RAMP does not use companies' heritage application 5250 navigation and behavior as the foundation, companies can design the application that they want by using the Instant Prototyping Assistant to generate a new application. No programming is required during this prototyping phase, so the application prototype can be defined by a business analyst or an expert user. A “prototype” usually must be disposed of in favor of the final product, but that is not the case with RAMP. The prototype evolves into the finished application. Within a short time, typically days, companies can prototype the overall navigation and behavior of their application with a modern appearance that is provided by the RAMP framework.

Step 2: Populate the application framework

Next, using RAMP's wizard-based isolation engine, called the Application Navigation Assistant, companies isolate the valuable parts of their application, hiding the navigation that is required to run these working components. Populating the framework is accomplished by adding working components into the prototype. This step is where the prototype comes to life as it evolves into the real application.

Two types of components can be used in the prototype:

- Type 1. The parts of existing applications that are isolated and packaged for repurposing. RAMP's isolation engine takes care of displaying only the screens from heritage applications where users perform actual work, while hiding all the intermediate navigation screens. The RAMP isolation technology is non-intrusive, which means existing programs do not have to be altered. RAMP works with any 5250 application, whether it is home-grown or purchased, RPG or COBOL. The focus during this step is to quickly repurpose all the valuable parts of your applications into the framework and deploy a first version.
- Type 2. New function that can access data directly from one or many databases or can integrate with other applications through technologies such as Web Services, APIs, and message queues. When all the components are added to the framework, the prototype becomes the finished product.

Companies that have multiple software applications and packages running on one or more servers can consolidate these applications (including both batch and interactive tasks) into a composite application, which dramatically increases user productivity. Developers can build applications that give a consistent user interface to multiple applications and deploy this by using Windows or the web. With one set of RAMP navigations for both Windows and web users, training and deployment is simplified.

Modernized RAMP applications can be delivered as a web browser or Windows rich-client application that runs on IBM i servers.

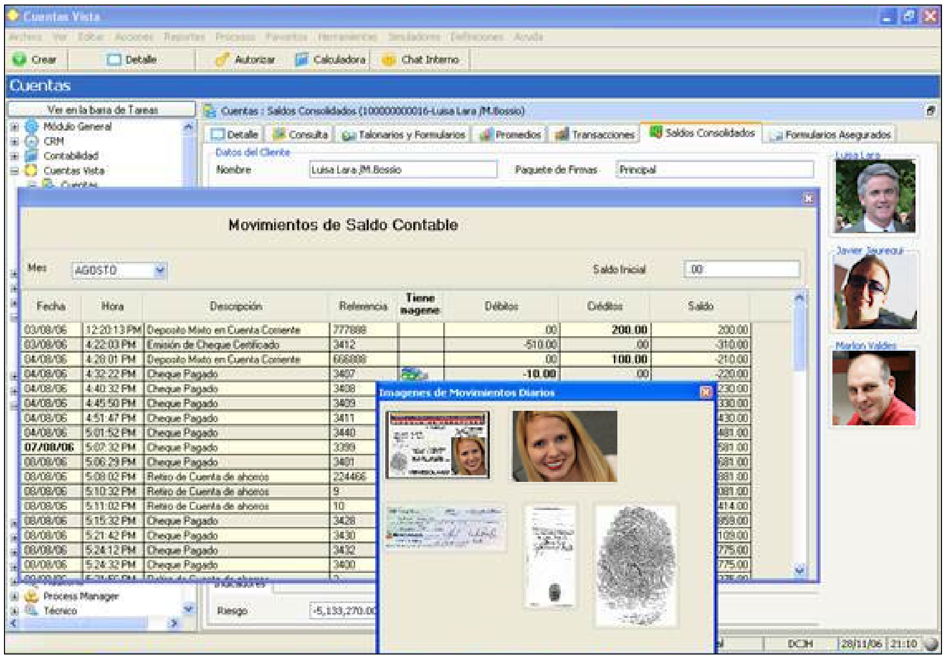
Step 3: Ongoing reengineering and enhancements

RAMP allows users to incrementally reengineer and extend existing applications. Companies can periodically reassess their heritage applications, prioritize which parts need the most attention, and then replace repurposed heritage components with new components.

This incremental approach greatly reduces risk and deployment time. Companies can address the shortcomings of their heritage applications by replacing them a part at a time with brand new enhanced components or adding functions. The result is a blend of existing applications and new programs that are working side-by-side, displayed in a single web browser or Windows rich-client application.

By replacing all of the repurposed components with new platform-independent LANSA components, companies can effectively reengineer their entire application and severe the ties to heritage applications. This severing is an added benefit for companies that want to end packaged application maintenance costs. A fully reengineered RAMP solution can be deployed to IBM i, Windows, or Linux servers.

The final step of RAMP can support complex presentation and multiple languages (see the examples that follow). See Figure 1-105.



Nombre: Luisa Lara JM.Bossio
Paquete de Firmas: Principal

Movimientos de Saldo Contable

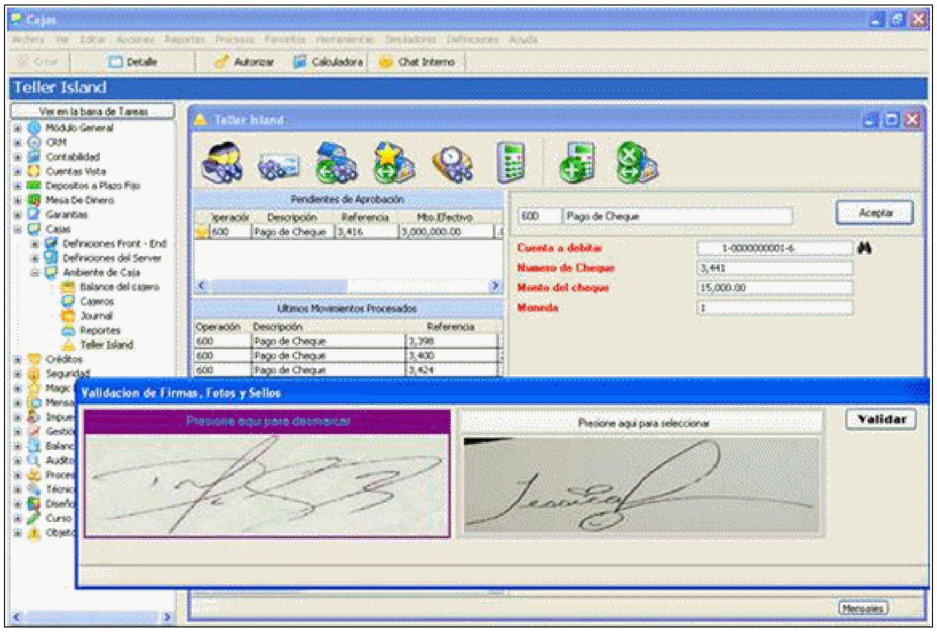
Mes: AGOSTO
Saldo Inicial: .00

Fecha	Hora	Descripción	Referencia	Tiene Imagene	Débitos	Créditos	Saldo
03/08/06	12:20:13 PM	Deposito Mixto en Cuenta Corriente	777888		.00	200.00	200.00
03/08/06	4:22:03 PM	Emisión de Cheque Certificado	3412		-510.00	.00	-310.00
04/08/06	4:28:01 PM	Deposito Mixto en Cuenta Corriente	666888		.00	100.00	-210.00
04/08/06	4:32:22 PM	Cheque Pagado	3407		-10.00	.00	-220.00
04/08/06	4:40:32 PM	Cheque Pagado	3408				230.00
04/08/06	4:45:50 PM	Cheque Pagado	3409				330.00
04/08/06	4:51:47 PM	Cheque Pagado	3411				430.00
04/08/06	5:01:52 PM	Cheque Pagado	3440				481.00
07/08/06	5:07:32 PM	Cheque Pagado	3399				581.00
08/08/06	5:06:29 PM	Cheque Pagado	3401				681.00
08/08/06	5:08:02 PM	Retiro de Cuenta de ahorros	224466				881.00
08/08/06	5:10:32 PM	Retiro de Cuenta de ahorros	9				081.00
08/08/06	5:11:02 PM	Retiro de Cuenta de ahorros	10				414.00
08/08/06	5:15:32 PM	Cheque Pagado	3428				859.00
08/08/06	5:21:42 PM	Cheque Pagado	3430				109.00
08/08/06	5:24:12 PM	Cheque Pagado	3432				775.00
08/08/06	5:24:32 PM	Cheque Pagado	3400				775.00

Riesgo: -5,133,270.00

Figure 1-105 RAMP images

Figure 1-106 shows RAMP signatures.



Pendientes de Aprobación

Operación	Descripción	Referencia	Mto.Efectivo
600	Pago de Cheque	3,416	3,000,000.00

600 Pago de Cheque

Cuenta a debitar: 1-000000001-6
Numero de Cheque: 3,441
Monto del cheque: 15,000.00
Moneda: 1

Ultimos Movimientos Procesados

Operación	Descripción	Referencia
600	Pago de Cheque	3,398
600	Pago de Cheque	3,400
600	Pago de Cheque	3,424

Figure 1-106 RAMP signatures

How does RAMP work?

The RAMP framework has a familiar navigation tree and tab style user interface, and is used to rapidly prototype, build, and deploy applications with web browser or Windows rich-client interfaces. RAMP uses the framework as its foundation to provide a “plug-in” architecture that enables both new and heritage components to seamlessly run side-by-side and give multiple deployment options. Because the RAMP technology to isolate and repurpose heritage applications within the framework is non-intrusive, the overall modernization process is drastically sped up and existing 5250 applications continue to work as they did before.

The simplest way to explain how the RAMP isolation technology works is by visualizing the steps that are required for users to complete a business transaction using existing 5250 applications. Typically, users must navigate through menus and select options by using action codes and function keys to arrive at the screens where they perform the work that is needed to complete a transaction. See Figure 1-107.

RAMP's isolation technology automates navigation through menus and screens

Figure 1-107 RAMP isolation technology

The RAMP wizard-based isolation engine analyzes and records the navigation within a heritage application to determine whether each window is a "navigation" or a "destination" component. A "destination" is the final step of an application process where users work with data. All the steps before the destination are categorized as "navigations" and RAMP hides them from the user. Hiding the cumbersome navigation of multiple menus is a great benefit to productivity.

After a destination is identified, its navigation path and key stroke requirements are packaged for repurposing and can be added into the framework. The framework includes areas for new search and menu components that allow users to navigate to destinations in a much more productive manner. For example, a user might go through five navigation steps to look up customer information (the first destination) and then go through another three navigation steps to view a specific order for that customer (another destination). With RAMP, that 10-step process of eight navigations and two destinations can be consolidated into two steps.

However, RAMP goes much further. Companies can contemplate reengineering an application process that requires several destinations to be run sequentially to complete a complex business transaction. RAMP allows companies to create a single component that collects all the data elements that are required for these destinations. It can then pass the associated data elements to each destination in the correct order for the business transaction. See Figure 1-108.

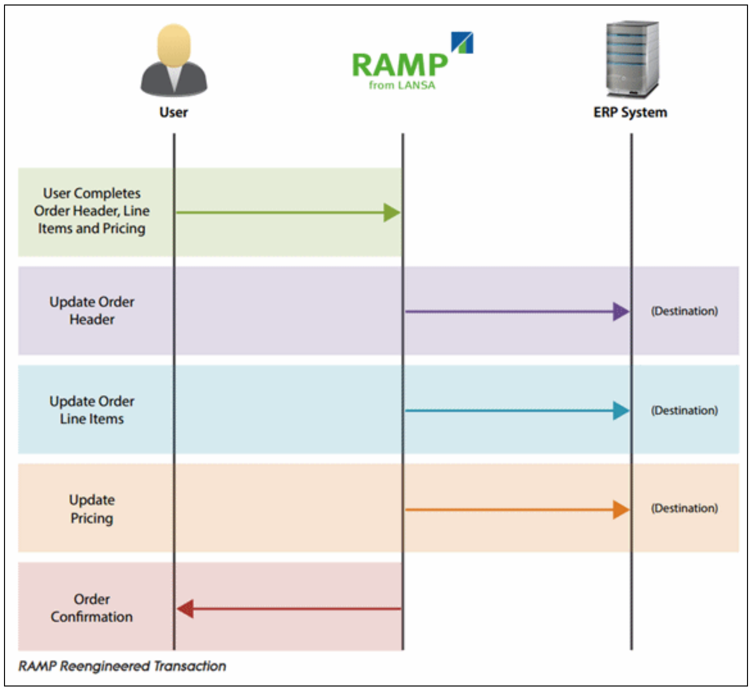


Figure 1-108 RAMP re-engineered transaction

After a user enters all the data into a single window, RAMP runs each destination sequentially by sending the appropriate data elements to the appropriate destination, processing that destination, and moving on to the next destination. The cycle continues until all the steps required to process the transaction are completed.

Because RAMP allows you to mix-and-match existing 5250 screens and batch jobs into one component, users can run multiple programs with a single mouse click. This is a significant improvement over taking many minutes to complete the sequence.

In addition, as many companies use multiple applications (including ERP, inventory, and financial), consolidating destinations from multiple software packages into a single object, and then using the RAMP isolation engine to hide the navigation between the destinations can significantly increase user productivity.

By consolidating applications, streamlining business processes, and making users more productive, the RAMP isolation technology acts as a bridge between where business processes are today and where companies want them to be in the future.

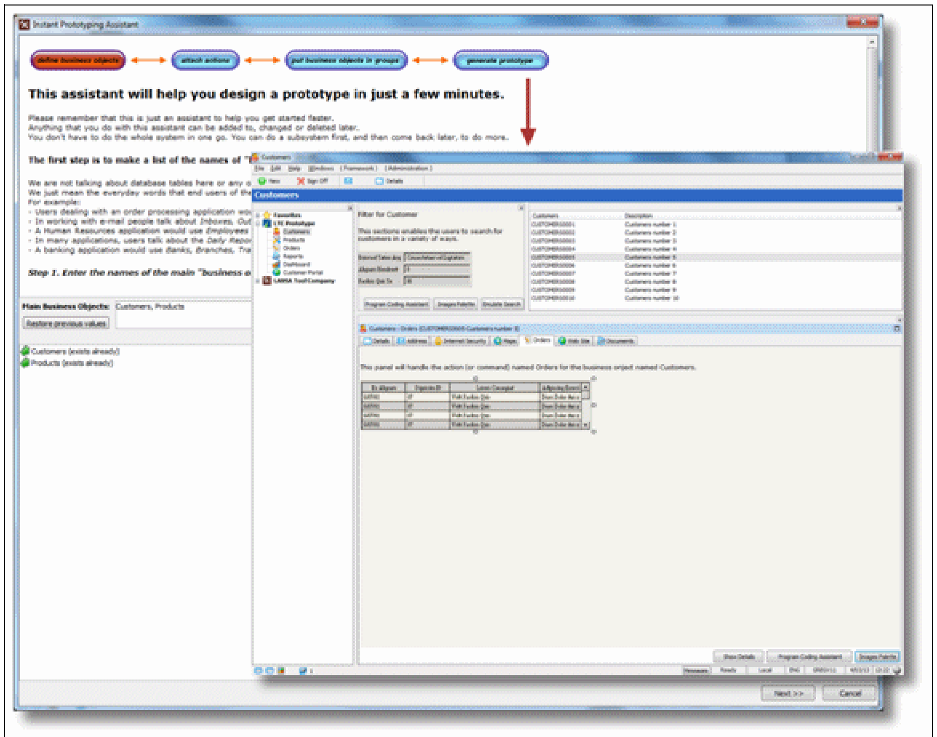
Defining the application framework

The RAMP framework is essentially an application skeleton that provides the overall aesthetics, infrastructure, and behavior of the new application. The biggest benefit of using an application framework is that all the application's underpinnings (including menu structures, search panels, search results panels, work panels, the intercommunication between the panels, user- and role-based security, and user customization features) are already in place.

Because the application's infrastructure is provided by RAMP, developers do not waste time “being artists” or writing the “plumbing code” that is required.

The first step towards defining a new application is starting the LANSA Instant Prototyping Assistant, which creates the initial prototype of a new application. The prototype is completed without writing a single line of code by using the Instant Prototyping Assistant. The assistant is a wizard that walks a business analyst or expert user through a question and answer session to model the new application. Using simple terms, you supply your company's application requirements to the Instant Prototyping Assistant, which in turn feeds those results to the framework for generation of the prototype.

The prototyped application evolves into the final application as developers populate it with working components. The RAMP working prototype enables companies to share the application with their users to get feedback during the initial stages of development. This feedback, early in the development cycle, helps ensure a more successful design and positive user acceptance. Because the prototyping process is fast and easy to use, companies can quickly and iteratively incorporate user requirements into the prototype to help flesh out the best design for their new applications. See Figure 1-109.



This assistant will help you design a prototype in just a few minutes.

Please remember that this is just an assistant to help you get started faster.
Anything that you do with this assistant can be added to, changed or deleted later.
You don't have to do the whole system in one go. You can do a subsystem first, and then come back later, to do more.

The first step is to make a list of the names of "

We are not talking about database tables here or any o
We just mean the everyday words that end users of the
For example:
- Users dealing with an order processing application wo
- In working with e-mail people talk about *Inboxes, Out*
- A Human Resources application would use *Employees*
- In many applications, users talk about the *Daily Repor*
- A banking application would use *Banks, Branches, Tra*

Step 1. Enter the names of the main "business o

Main Business Objects: Customers, Products

Restore previous values

Customers (exists already)
Products (exists already)

Filter for Customer

This sections enables the users to search for customers in a variety of ways.

This panel will handle the action (or command) named Orders for the business object named Customers.

The Instant Prototyping Assistant guides you through the steps needed to build an executable prototype of your new business application.

Figure 1-109 RAMP Instant Prototyping Assistant

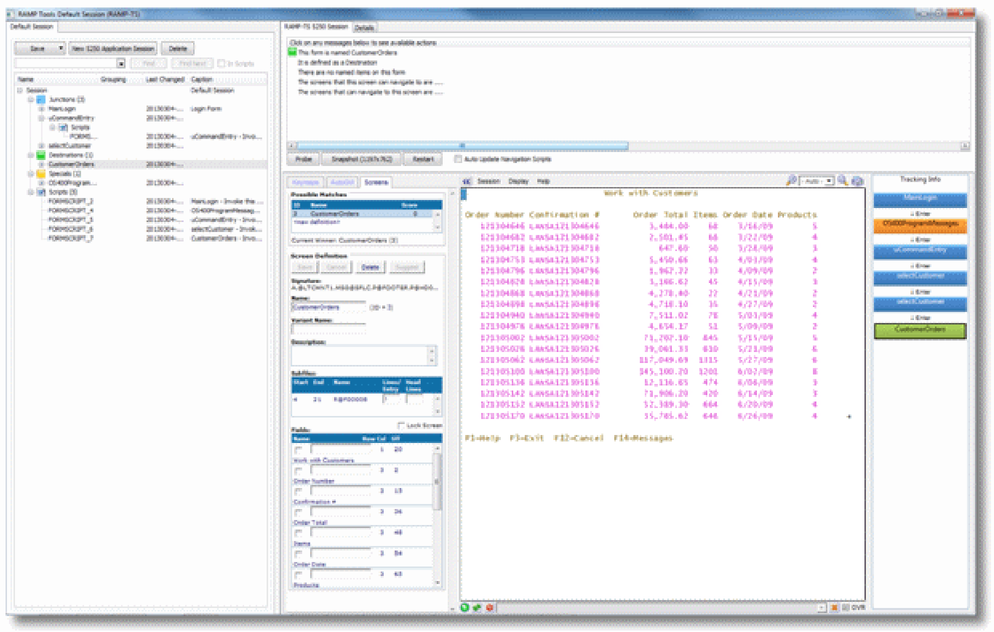
Populating the application framework

When the prototyping and design phase is complete, the next step is to populate the framework with working components. Components are items such as search panels, search result panels, isolated older destinations, or new functions. RAMP provides wizards to quickly generate most of these components for you. As components are added into the framework, the transition from working prototype to the finished application begins.

A typical application might have over 2,000 screens of which only 300 - 400 might be classified as destinations. Moving destinations into RAMP and hiding all the navigation from the user is not a manual process. The RAMP wizard-based isolation engine does it for you.

From the prototype, you start the Application Navigation Assistant to record every menu option, sub file, function key, and key stroke required by a typical user to navigate to the application's destinations. When a destination is captured, its entire process is packaged into a component that can be added into the framework.

The Application Navigation Assistant enables companies to streamline the navigation of their IBM i applications, controlling how heritage applications fit into new applications. A company's heritage application navigation does not control the way that the new application is designed. See Figure 1-110.



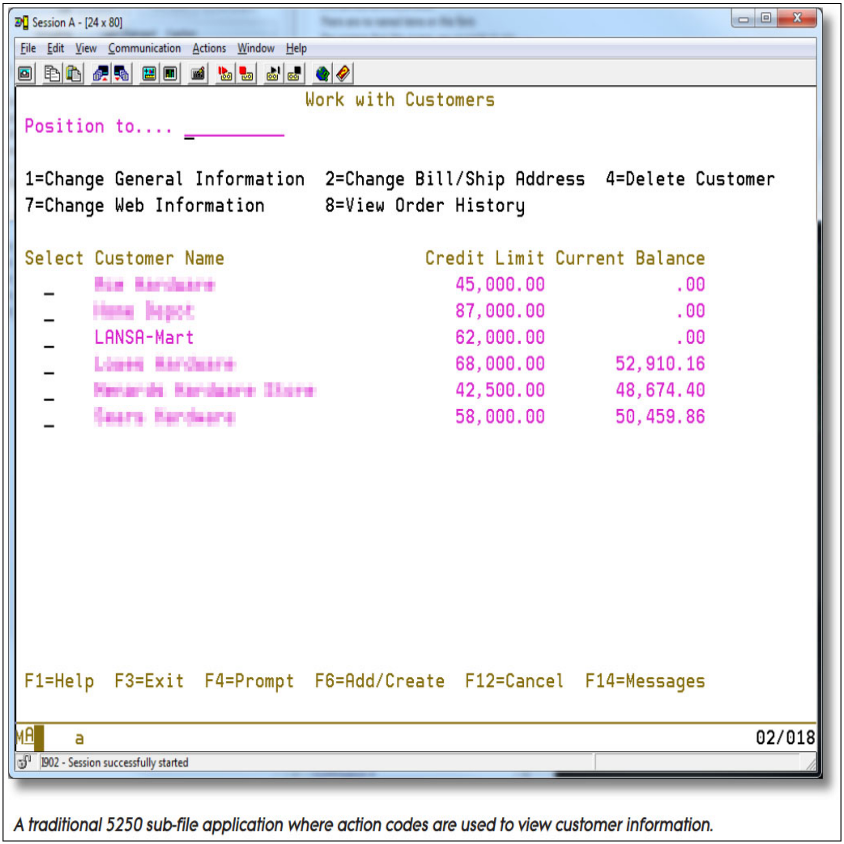
```
Click on any messages below to see available actions
This form is named CustomerOrders
It is defined as a Destination
There are no named items on this form
The screens that this screen can navigate to are ....
The screens that can navigate to this screen are ....
```

```
                              Work with Customers

Order Number Confirmation #       Order Total Items Order Date Products
    121304646 LANSA121304646         3,484.00    68    3/16/09         5
    121304682 LANSA121304682         2,501.45    68    3/22/09         4
    121304718 LANSA121304718           647.60    50    3/28/09         3
    121304753 LANSA121304753         5,450.66    63    4/03/09         4
    121304796 LANSA121304796         1,967.22    33    4/09/09         2
    121304828 LANSA121304828         1,166.62    45    4/15/09         3
    121304868 LANSA121304868         4,278.40    22    4/21/09         2
    121304898 LANSA121304898         4,718.10    16    4/27/09         2
    121304940 LANSA121304940         7,511.02    78    5/03/09         4
    121304978 LANSA121304978         4,654.17    51    5/09/09         2
    121305002 LANSA121305002        71,207.10   845    5/15/09         5
    121305026 LANSA121305026        39,061.33   610    5/21/09         6
    121305062 LANSA121305062       117,049.69  1115    5/27/09         6
    121305100 LANSA121305100       145,300.70  1201    6/02/09         8
    121305136 LANSA121305136        12,116.65   474    6/08/09         3
    121305142 LANSA121305142        71,906.20   420    6/14/09         3
    121305152 LANSA121305152        52,389.30   664    6/20/09         4
    121305170 LANSA121305170        35,785.62   646    6/26/09         4

F1=Help  F3=Exit  F12=Cancel  F14=Messages
```

The Application Navigation Assistant is used to explore and map the navigation paths through your existing 5250 applications.

Figure 1-110 Application Navigation Assistant

Figure 1-111 shows a typical IBM i customer application where customers are presented in subfile format and users must type in action codes to work with certain sets of customer information. For example, users enter the action code “1” to work with customer details, action code “2” to work with address information, action code “7” to work with Internet-related data, and action code “8” to work with customer orders.



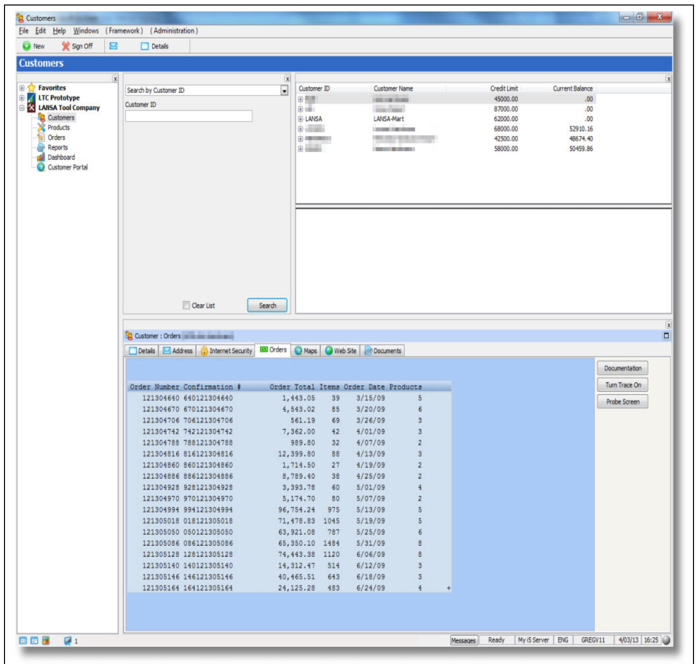
```
                          Work with Customers
Position to.... __________

1=Change General Information  2=Change Bill/Ship Address  4=Delete Customer
7=Change Web Information      8=View Order History

Select Customer Name                      Credit Limit Current Balance
  _                                          45,000.00             .00
  _                                          87,000.00             .00
  _    LANSA-Mart                            62,000.00             .00
  _                                          68,000.00       52,910.16
  _                                          42,500.00       48,674.40
  _                                          58,000.00       50,459.86

F1=Help  F3=Exit  F4=Prompt  F6=Add/Create  F12=Cancel  F14=Messages
MA    a                                                          02/018
1902 - Session successfully started
```

A traditional 5250 sub-file application where action codes are used to view customer information.

Figure 1-111 Traditional 5250 subfile application

With traditional refacing, the new version still behaves the same as the heritage application because the application flow has not changed. Users still must go through the same laborious sequence of steps to work with customer information. This approach neither improves user productivity nor makes accessing information any simpler.

In Figure 1-112, RAMP replaces the subfile navigation with a new search component and all the destinations reachable by using action codes are now in tabs. The application framework takes care of linking everything together. The customer details destination (action code “1”) is on the first tab, the address destination (action code “2”) on second tab, the Internet data destination (action code “7”) is on the third tab, and the customer order destination (action code “8”) is on the fourth tab.



Customer ID	Customer Name	Credit Limit	Current Balance
[redacted]	[redacted]	45000.00	.00
[redacted]	[redacted]	87000.00	.00
LANSA	LANSA-Mart	62000.00	.00
[redacted]	[redacted]	68000.00	52910.16
[redacted]	[redacted]	42500.00	48674.40
[redacted]	[redacted]	58000.00	50459.86

```
Order Number Confirmation #        Order Total Items Order Date Products
   121304640 640121304640             1,443.05    39    3/15/09        5
   121304670 670121304670             4,543.02    85    3/20/09        6
   121304706 706121304706               561.19    69    3/26/09        3
   121304742 742121304742             7,362.00    42    4/01/09        3
   121304788 788121304788               989.80    32    4/07/09        2
   121304816 816121304816            12,399.80    88    4/13/09        3
   121304860 860121304860             1,714.50    27    4/19/09        2
   121304886 886121304886             8,789.40    38    4/25/09        2
   121304928 928121304928             3,393.78    60    5/01/09        4
   121304970 970121304970             5,174.70    80    5/07/09        2
   121304994 994121304994            96,754.24   975    5/13/09        5
   121305018 018121305018            71,478.83  1045    5/19/09        5
   121305050 050121305050            63,921.08   787    5/25/09        6
   121305086 086121305086            65,350.10  1484    5/31/09        8
   121305128 128121305128            74,443.38  1120    6/06/09        8
   121305140 140121305140            14,312.47   514    6/12/09        3
   121305146 146121305146            40,465.51   643    6/18/09        3
   121305164 164121305164            24,125.28   483    6/24/09        4     +
```

The RAMP version of the legacy 5250 application used to view customer order information.

Figure 1-112 Revised application using RAMP with search component and tabs

With RAMP, moving from viewing customer details to customer orders is literally one mouse click away. When users move from tab to tab, behind the scenes RAMP emulates all the keystrokes that are required to navigate from one destination to another. For example, to move from the customer details tab to the customer orders tab, behind the scenes RAMP exits the customer details destination, returns to the subfile navigation, enters an action code of “8” (orders) for that customer, and then displays the customer order destination in the Orders tab. The ease of navigation to disparate sets of customer information on tab sheets boosts user productivity.

The powerful features that are built into RAMP allow companies to mix-and-match existing application screens and batch jobs with brand new components that run on IBM i, Windows, or Linux servers in a web browser or as Windows rich-client applications. Companies can web-enable parts of their application to provide a self-service portal, consume and publish web services, or deliver new function to meet business requirements, all within the same framework that serves as a platform for full modernization.

When the framework is populated, your application contains a mixture of new components and heritage destinations and is ready for deployment. Deciding on the best ratio of heritage destinations and new components depends on time, resources, and the state of your heritage application. If time is short, rapidly deploying a first release with a high ratio of heritage destinations makes sense.

The framework is designed for maximum compatibility with new technologies, so components can interact with data on multiple servers, access information from multiple databases, and integrate with technologies like web services, .NET, and Java.

Ongoing reengineering and enhancements

In the same iterative fashion as the prototyping phase, RAMP allows companies to reengineer their applications and add enhancements over time. RAMP flexibility allows companies to make incremental changes to their applications and deploy them at their own pace. After your first RAMP deployment, you have the opportunity to analyze the current state of your applications, assign a priority for the requirements, and continue enhancing the applications.

New components can be used in two ways: to replace an existing destination with an enhanced version that adds more value, or to add a function. With this architecture, developers can create brand new components that work alongside the heritage components (destinations) that are running inside the framework. IBM i organizations can reengineer the application to take full advantage of the powerful LANSA tools and incorporate new technologies and techniques, like Web Services, SOA and XML.

In Figure 1-113, the customer order destination is replaced with a new component where users can view multiple orders in a tree view control for better visibility of the same information. Tabs have also been added to integrate new technologies and function that is not present in the heritage application. In this case, a third-party logistics software package is running on a Windows server to view a customer's routing information and a browser control to view a customer's website.

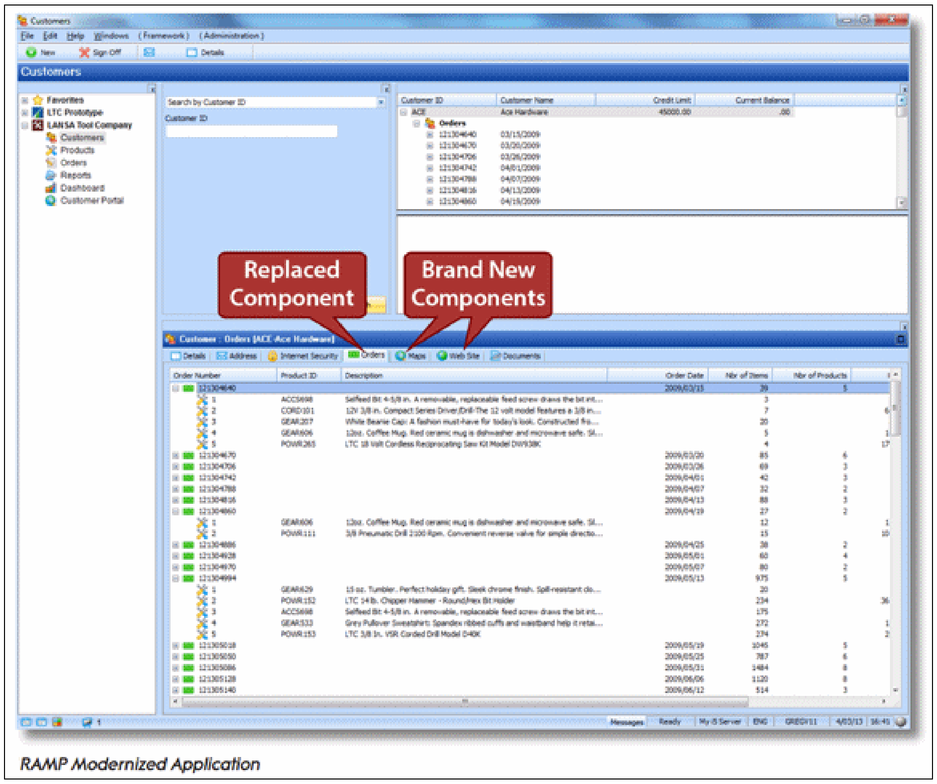


Figure 1-113 RAMP modernized application

If application independence is your final goal, the full range of RAMP possibilities can help you deliver your applications across multiple interfaces and multiple platforms without having to master many more languages.

More information and resources

More product information, data sheets, and customer case studies about RAMP from LANSA can be found at:

http://www.lansa.com/ramp

1.8.3 Visual LANSA

What is Visual LANSA?

Visual LANSA provides a comprehensive Windows-based integrated development environment that is dedicated to delivering high-quality commercial applications for the IBM i, while also providing support for cross-platform deployment. The Visual LANSA platform independent Meta Data Repository and high-level language (HLL) programming language allows you to develop web, mobile (web or native), Windows rich-client, 5250 and server-based applications from a single code base. See Figure 1-114.

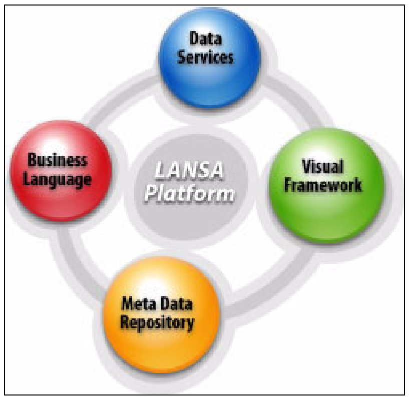


Figure 1-114 Visual LANSA platform

Developers can deploy applications to IBM i, Linux, Windows, and Wireless devices. All these applications can be built with one skill set from within the same visual IDE.

Developers receive all the productivity benefits a Windows based integrated development environment brings: point-and-click to select fields and files, cut-and-paste to rapidly edit and debug, and a powerful forms painter to design graphical applications.

When designing the environment LANSA environment, LANSA follows these principles:

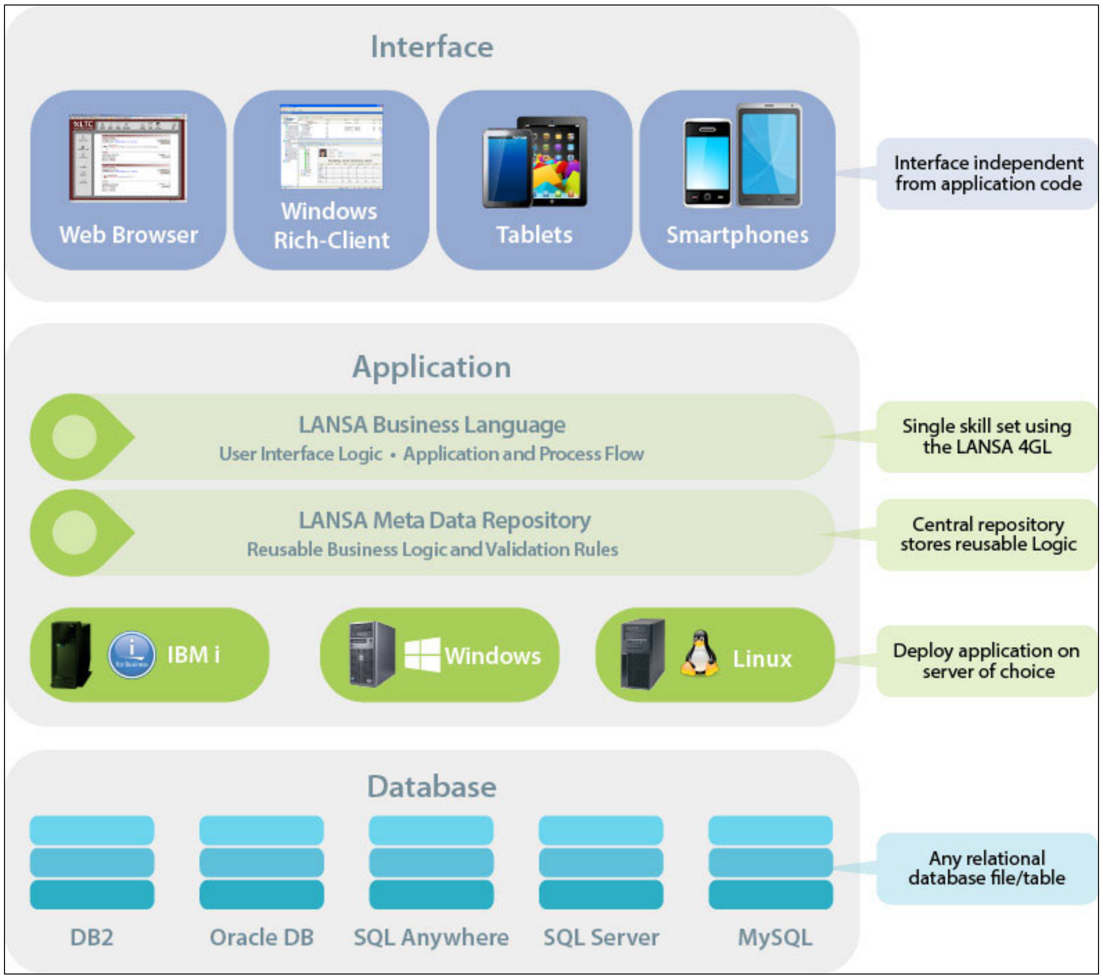
- Making people hardcode things that change is not wise.
- Binding applications to one technology stack is too restrictive.
- Nobody has the luxury of starting with a clean slate.
- Code that can change without tracking becomes impossible to maintain.
- Never define a function or rule more than once: reuse should be the norm.
- Do not expect that one user interface style suits everyone

The LANSA platform is based on four key elements: a business-focused HLL, a metadata repository, an independent data service layer (DSL), and a visual framework.

High-level language (HLL)

To be able to deliver cross-platform, cross-database and multi-interface applications, LANSA developed a specific business-focused language that is called RDML, an acronym for Rapid

Development and Maintenance Language. RDML is regarded as a fourth-generation language (4GL). By knowing this high-level language, it is possible to create many different kinds of software that runs on virtually any combination of hardware. See Figure 1-115.

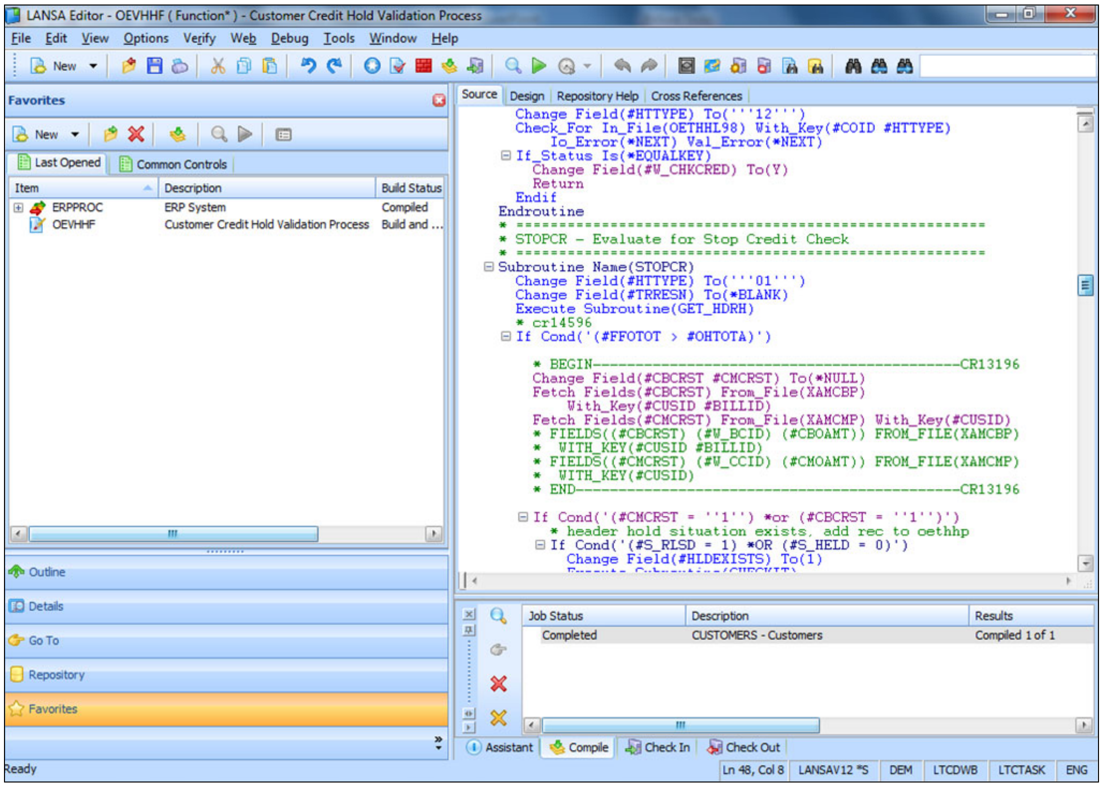


Interface
Web Browser
Windows Rich-Client
Tablets
Smartphones
Interface independent from application code
Application
LANSA Business Language
User Interface Logic • Application and Process Flow
Single skill set using the LANSA 4GL
LANSA Meta Data Repository
Reusable Business Logic and Validation Rules
Central repository stores reusable Logic
IBM i
Windows
Linux
Deploy application on server of choice
Database
DB2
Oracle DB
SQL Anywhere
SQL Server
MySQL
Any relational database file/table

Figure 1-115 Creating multiple types of software that run on multiple types of hardware

A 4GL is a high-level language or environment that is designed with a specific programming purpose in mind, the development of business-oriented, database-centric systems. In the evolution of computing, the 4GL followed the 3GL in an upward trend toward higher abstraction and statement power.

Because RDML is oriented toward a specific task, that of building applications, it is an easy language to learn and delivers significant productivity benefits to the developer. For example, just one line of RDML code generates 30 lines of 3GL code. See Figure 1-116.



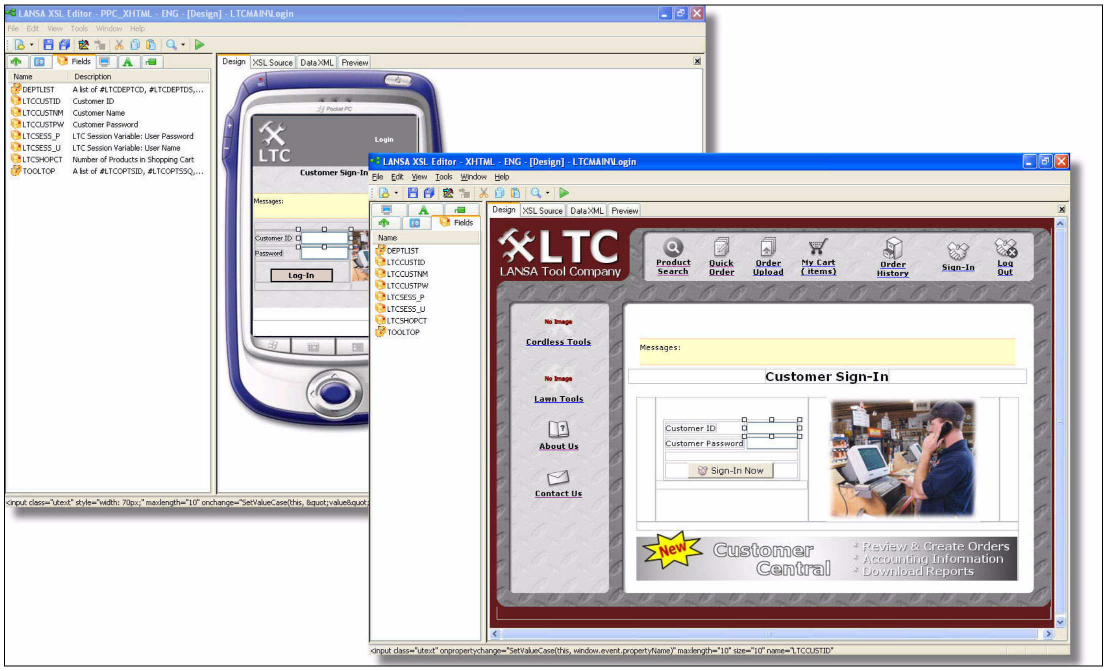
Item	Description	Build Status
ERPPROC	ERP System	Compiled
OEVHHF	Customer Credit Hold Validation Process	Build and ...

```
    Change Field(#HTTYPE) To('''12''')
    Check_For In_File(OETHHL98) With_Key(#COID #HTTYPE)
        Io_Error(*NEXT) Val_Error(*NEXT)
    If_Status Is(*EQUALKEY)
      Change Field(#W_CHKCRED) To(Y)
      Return
    Endif
  Endroutine
  * ====================================================
  * STOPCR - Evaluate for Stop Credit Check
  * ====================================================
  Subroutine Name(STOPCR)
    Change Field(#HTTYPE) To('''01''')
    Change Field(#TRRESN) To(*BLANK)
    Execute Subroutine(GET_HDRH)
    * cr14596
    If Cond('(#FFOTOT > #OHTOTA)')

      * BEGIN-------------------------------------------CR13196
      Change Field(#CBCRST #CMCRST) To(*NULL)
      Fetch Fields(#CBCRST) From_File(XAMCBP)
          With_Key(#CUSID #BILLID)
      Fetch Fields(#CMCRST) From_File(XAMCMP) With_Key(#CUSID)
      * FIELDS((#CBCRST) (#W_BCID) (#CBOAMT)) FROM_FILE(XAMCBP)
      *  WITH_KEY(#CUSID #BILLID)
      * FIELDS((#CMCRST) (#W_CCID) (#CMOAMT)) FROM_FILE(XAMCMP)
      *  WITH_KEY(#CUSID)
      * END---------------------------------------------CR13196

      If Cond('(#CMCRST = ''1'') *or (#CBCRST = ''1'')')
        * header hold situation exists, add rec to oethhp
        If Cond('(#S_RLSD = 1) *OR (#S_HELD = 0)')
          Change Field(#HLDEXISTS) To(1)
```

Job Status	Description	Results
Completed	CUSTOMERS - Customers	Compiled 1 of 1

Figure 1-116   One line of RDML code generates 30 lines of 3GL code

The source editor in the visual IDE can take on web, mobile, Windows rich-client, 5250, server-based applications, wireless, and modernization projects. Developers can create multiple window interfaces from one set of source code. See Figure 1-117.



Name	Description
DEPTLIST	A list of #LTCDEPTCD, #LTCDEPTDS,...
LTCCUSTID	Customer ID
LTCCUSTNM	Customer Name
LTCCUSTPW	Customer Password
LTCSESS_P	LTC Session Variable: User Password
LTCSESS_U	LTC Session Variable: User Name
LTCSHOPCT	Number of Products in Shopping Cart
TOOLTOP	A list of #LTCOPTSID, #LTCOPTSSQ,...

```
<input class="utext" style="width: 70px;" maxlength="10" onchange="SetValueCase(this, &quot;value&quot;
```

```
<input class="utext" onpropertychange="SetValueCase(this, window.event.propertyName)" maxlength="10" size="10" name="LTCCUSTID"
```

Figure 1-117 Screen painting multiple window interfaces from one set of source code

Meta Data Repository and Data Services Layer

LANSA provides tools, collectively called the LANSA Repository, that describe, store, and deploy data definitions and their related business rules. A fundamental tenet of the LANSA repository-based architecture is to keep applications separated from the databases upon which they depend. This configuration is vital for data integrity. To accomplish the separation of business logic from application logic, LANSA allows data definitions, business rules, and algorithms to be stored and centralized in a single metadata repository.

In every system, a set of rules control how an application's programs are allowed to create, read, update, or delete data. These rules exist to maintain the quality and consistency of the data, and to protect data integrity in a multi-user, transactional environment. Failure to follow the rules can lead to anything from bad data values in a report to a total system crash. See Figure 1-118.

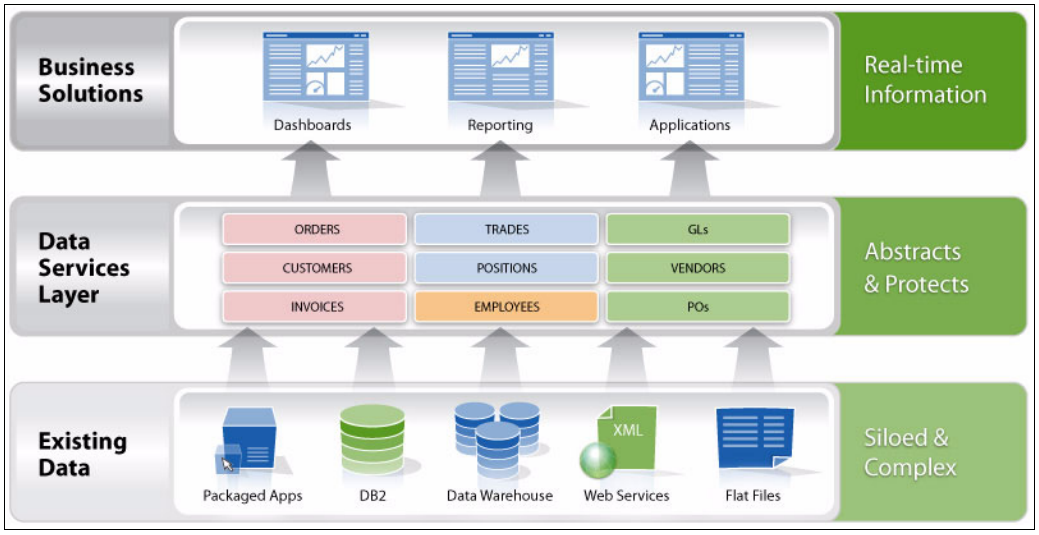
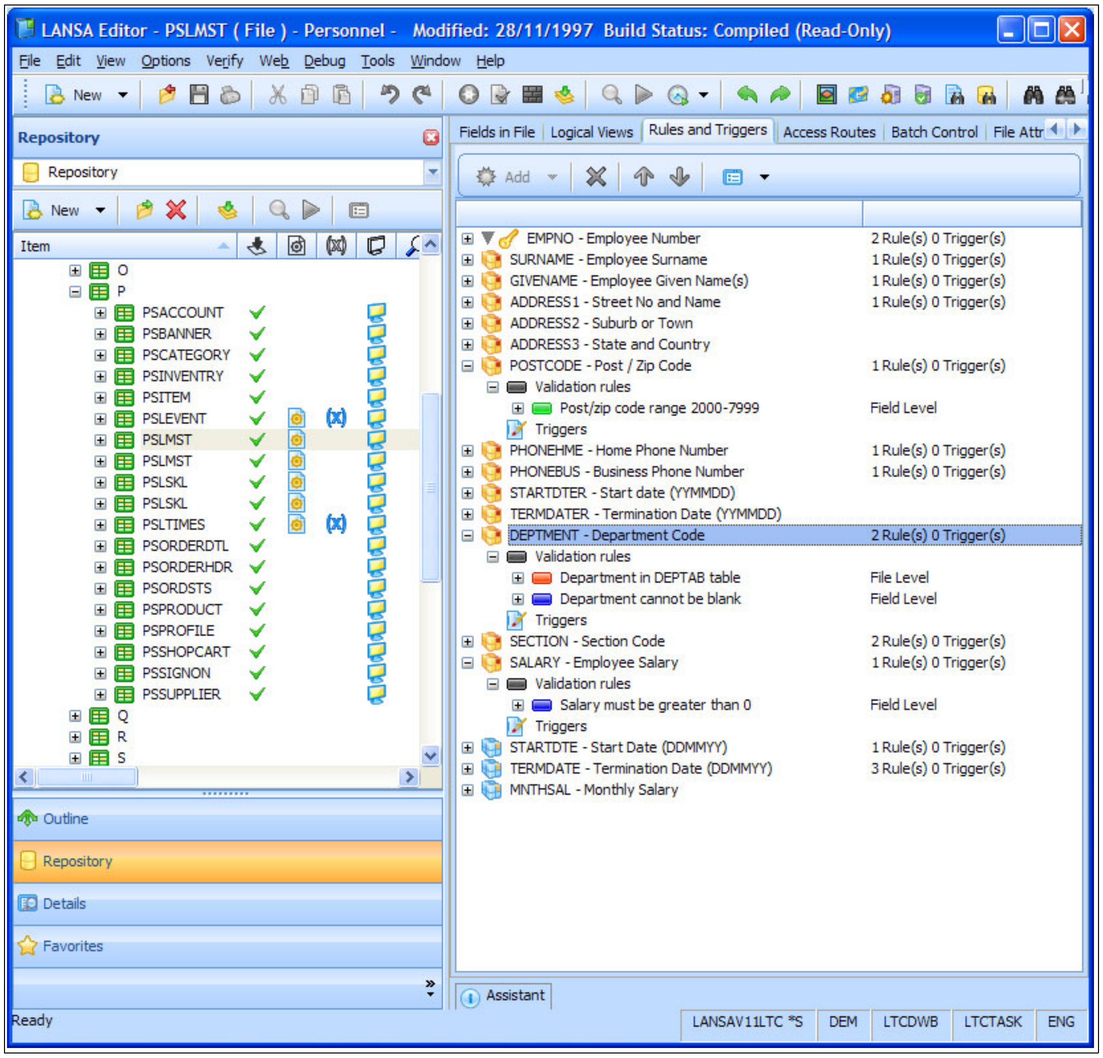


Figure 1-118 LANSA independent Data Services layer

Managing the data is easier when the data definitions and business rules are centrally defined outside the program code. If any definitions or rules need to be changed, developers must make that change in only one place. Second, make sure that any program or utility that accesses the data uses the most recent definitions and business rules. In other words, to protect your data, you want the definitions and rules to be centrally deployed, without exception.

From one window in the visual IDE, developers can work with field and file definitions, validations, relationships, and components that are stored in the metadata repository, in addition to graphical data modeling for databases. See Figure 1-119.



O
P
PSACCOUNT
PSBANNER
PSCATEGORY
PSINVENTRY
PSITEM
PSLEVENT
PSLMST
PSLMST
PSLSKL
PSLSKL
PSLTIMES
PSORDERDTL
PSORDERHDR
PSORDSTS
PSPRODUCT
PSPROFILE
PSSHOPCART
PSSIGNON
PSSUPPLIER
Q
R
S

EMPNO - Employee Number	2 Rule(s) 0 Trigger(s)
SURNAME - Employee Surname	1 Rule(s) 0 Trigger(s)
GIVENAME - Employee Given Name(s)	1 Rule(s) 0 Trigger(s)
ADDRESS1 - Street No and Name	1 Rule(s) 0 Trigger(s)
ADDRESS2 - Suburb or Town	
ADDRESS3 - State and Country	
POSTCODE - Post / Zip Code	1 Rule(s) 0 Trigger(s)
Validation rules	
Post/zip code range 2000-7999	Field Level
Triggers	
PHONEHME - Home Phone Number	1 Rule(s) 0 Trigger(s)
PHONEBUS - Business Phone Number	1 Rule(s) 0 Trigger(s)
STARTDTER - Start date (YYMMDD)	
TERMDATER - Termination Date (YYMMDD)	
DEPTMENT - Department Code	2 Rule(s) 0 Trigger(s)
Validation rules	
Department in DEPTAB table	File Level
Department cannot be blank	Field Level
Triggers	
SECTION - Section Code	2 Rule(s) 0 Trigger(s)
SALARY - Employee Salary	1 Rule(s) 0 Trigger(s)
Validation rules	
Salary must be greater than 0	Field Level
Triggers	
STARTDTE - Start Date (DDMMYY)	1 Rule(s) 0 Trigger(s)
TERMDATE - Termination Date (DDMMYY)	3 Rule(s) 0 Trigger(s)
MNTHSAL - Monthly Salary	

Figure 1-119 Visual IDE window

Describing the data and rules

The Meta Data Repository allows you to describe data items ("fields" in IBM i terminology) and their business rules, plus tables ("files" in IBM i terminology) and their business rules.

Data item definitions might include:

- Name. A variable that programs can refer to
- Labels. For the user interfaces
- Data type. Strings and numbers
- Formats. For example, dates in various representations
- Validation rules. For example, the data item must not be blank
- Actions. Actions that need to happen under certain conditions
- Help text. For a better understanding

A table definition might include these items:

- A list of the data items in the table. Data items can be physical or derived. Derived means that their value is based on the value of other data items from the same table or from other tables.
- Validation rules that are specific to the table context. For example, a validation for a customer table might be that you are not allowed to delete a customer if invoices or orders exist for that customer.
- Actions specific to the table context. For example, if a customer record is updated in this table, update customer information in other tables as well (a typical CRM scenario).
- Indexes and relationships with other tables.

Developers describe what the objects (data, tables, and rules) are in the repository rather than coding them to create them. LANSA stores the descriptions in database tables in the repository as shown in Figure 1-120.

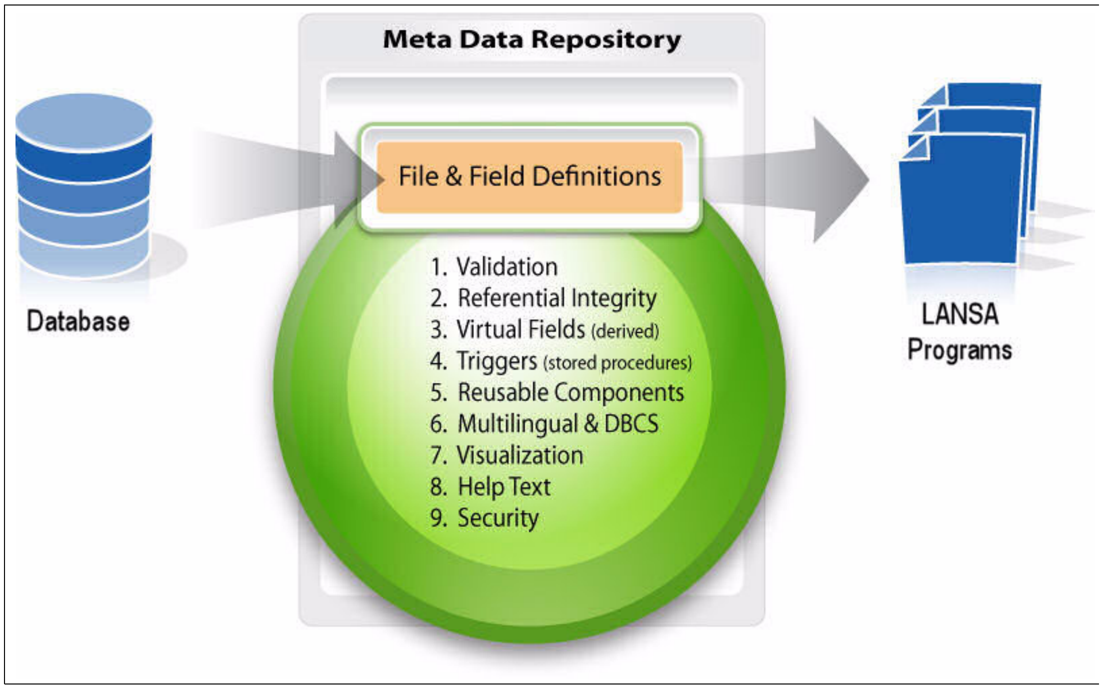


Figure 1-120 LANSA Data Management Services protects data from programs developed with LANSA

The Repository typically contains thousands of definitions and rules, depending on the size of the application. That is really where the logic is or should be defined, so the applications that use the data do not need copies of the definitions and rules.

Deploying the definitions

From the data, table, and business rule descriptions that are stored in the repository, LANSA generates an executable program to manage access to the data. This might be a compiled C#, C or RPG program, depending on the platform. This executable program is a component of the LANSA Data Management Services (DMS).

The DMS provides independence for the data from the applications that use it and provides independence from the database management systems in which the data is stored. This independence means that when you want to move your application to another platform, you just move and deploy the repository definitions to the other platform. This feature provides complete cross-platform capabilities for your solution.

Enforcing the rules to all applications

The LANSA DMS has always been available to programs that are developed with LANSA and programs that use LANSA Open for .NET to access data.

You can enforce the LANSA DMS routines that deploy the definitions and business rules (potentially thousands) to other applications and utilities by using LANSA enforcement triggers. Deployed as triggers at the database level, LANSA enforcement triggers can provide the same level of protection to any program or utility that accesses data that is described in the LANSA Repository.

A few Enforcement Triggers might activate hundreds of LANSA DMS defined rules and validations (Figure 1-121).

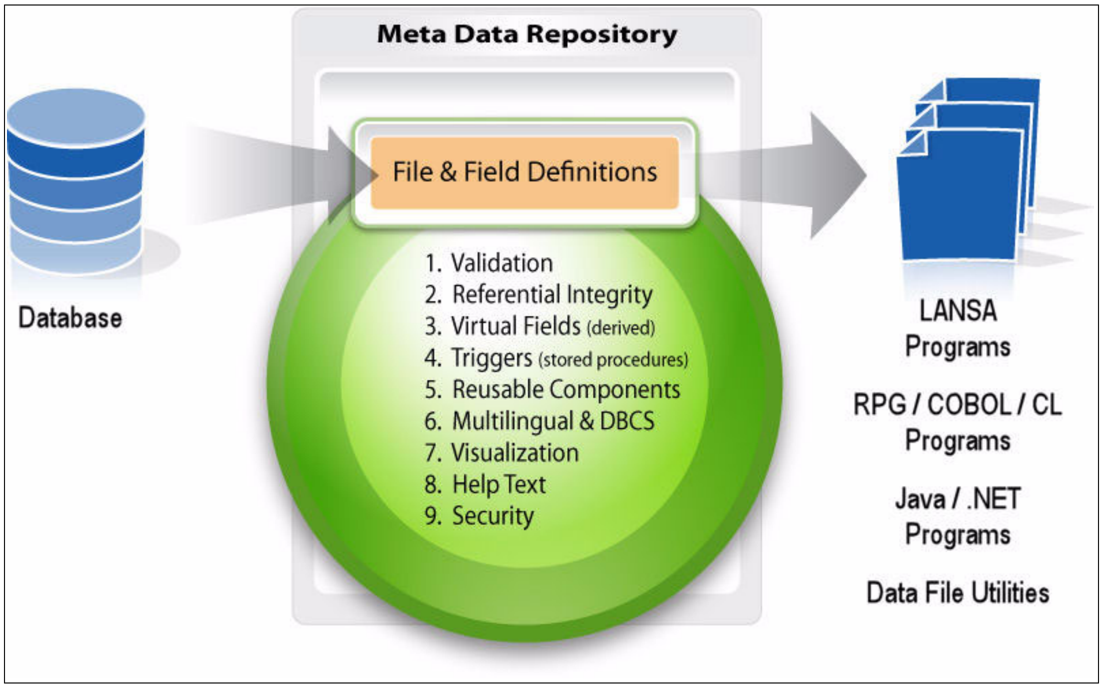


Figure 1-121 Meta Data Repository

What about existing data sets?

You can import the definitions of any existing data set into the LANSA Repository and then optionally further enhance the definitions by using the LANSA Repository tools. This process of making a file known to LANSA does not affect the file itself, nor does it involve any duplication of data.

Many companies use the Repository on top of a packaged solution. It allows them, for example, to define more field names that are easier to use, add formula/derived fields, and define more rules and actions, without affecting the packaged solution itself.

LANSA Data Management Services provide developers with these benefits:

- A single point of protection for business data but also universal coverage. Whether by mistake or intentionally, no program or utility can corrupt your data or cause referential integrity problems.
- Reduced maintenance costs. With LANSA, you change the definition once, rebuild the Data Management Service, and deploy it. There is no need to change a class and repair the repercussions. There is no need to change a /COPY (copybook) and recompile every program that uses the copied code.

- Consistency. The business rules that are associated with a data set are in one place, so when the rules change you need to maintain it only in this one place. All programs that access the data set through the LANSA DMS automatically use the same changed rule.
- Business level definitions. When the rules are described at a business level, instead of being coded in a particular programming language, maintenance is easier. The LANSA tools use data abstraction to remove details that are specific to program language, database, and platform deployment.
- Cross platform capabilities. It is easy to generate the Data Management Services for another platform from the same Repository definitions.

Visual LANSA Framework

The Visual LANSA Framework is part of Visual LANSA. It is a design framework that minimizes the effort that is required for business-focused developers and designers to create graphical, robust web and Windows applications, or a combination of both.

The Visual LANSA Framework allows developers to prototype commercial applications rapidly without coding. It automatically generates high-quality Windows and web applications with a consistent look and behavior (Figure 1-122).

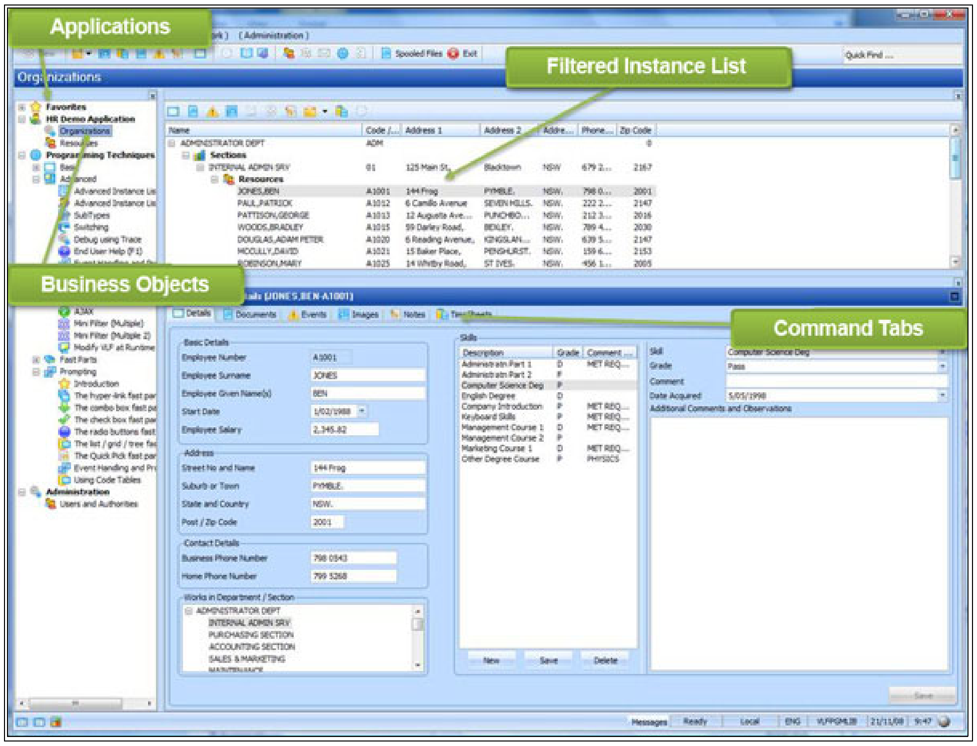
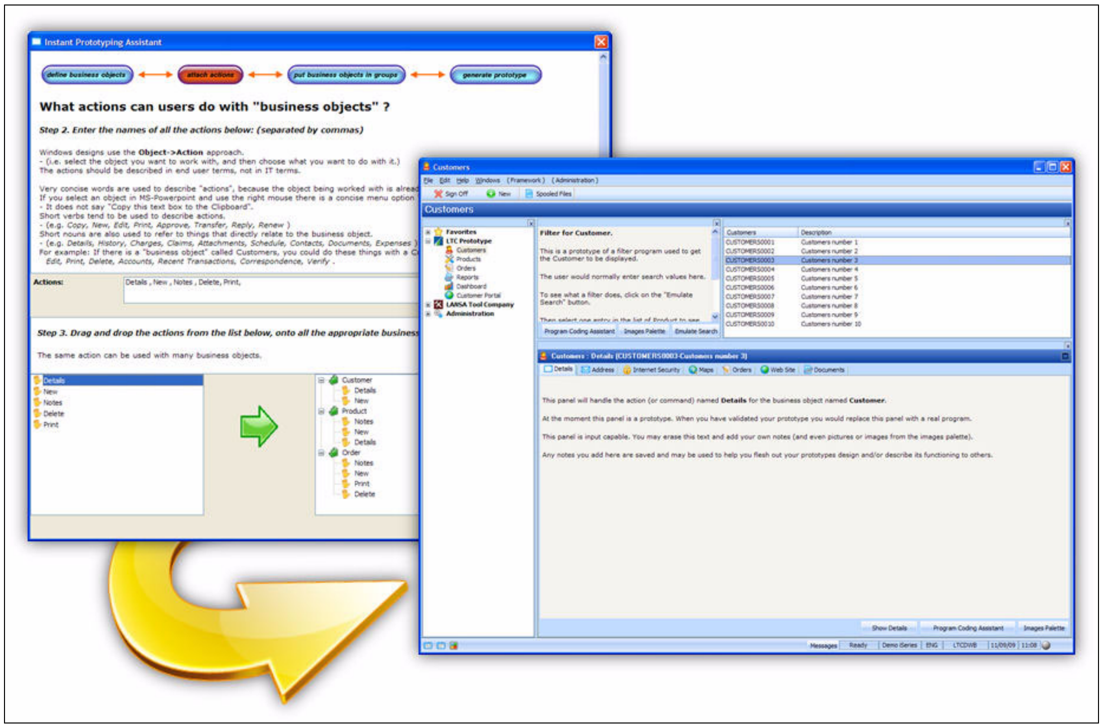


Figure 1-122 An example of a Visual LANSA Framework Windows rich-client desktop application window

You can prototype the application and user interface design at the same time and use the same design for both Windows and web deployment. Software prototyping minimizes risk and maximizes success. Prototypes are a simple and effective way to manage expectations and uncover hidden or new business requirements. The prototype is not thrown away but converts into a real application. The snap-in architecture of the Visual LANSA Framework facilitates the gradual conversion of the prototype into the final application (Figure 1-123).



What actions can users do with "business objects" ?

Step 2. Enter the names of all the actions below: (separated by commas)

Windows designs use the **Object->Action** approach.
- (i.e. select the object you want to work with, and then choose what you want to do with it.)
The actions should be described in end user terms, not in IT terms.

Very concise words are used to describe "actions", because the object being worked with is alread
If you select an object in MS-Powerpoint and use the right mouse there is a concise menu option
- It does not say "Copy this text box to the Clipboard".
Short verbs tend to be used to describe actions.
- (e.g. *Copy, New, Edit, Print, Approve, Transfer, Reply, Renew*)
Short nouns are also used to refer to things that directly relate to the business object.
- (e.g. *Details, History, Charges, Claims, Attachments, Schedule, Contacts, Documents, Expenses*)
For example: If there is a "business object" called Customers, you could do these things with a C
Edit, Print, Delete, Accounts, Recent Transactions, Correspondence, Verify .

Step 3. Drag and drop the actions from the list below, onto all the appropriate business

The same action can be used with many business objects.

Filter for Customer.

This is a prototype of a filter program used to get the Customer to be displayed.

The user would normally enter search values here.

To see what a filter does, click on the "Emulate Search" button.

Customers	Description
CUSTOMERS0001	Customers number 1
CUSTOMERS0002	Customers number 2
CUSTOMERS0003	Customers number 3
CUSTOMERS0004	Customers number 4
CUSTOMERS0005	Customers number 5
CUSTOMERS0006	Customers number 6
CUSTOMERS0007	Customers number 7
CUSTOMERS0008	Customers number 8
CUSTOMERS0009	Customers number 9
CUSTOMERS0010	Customers number 10

This panel will handle the action (or command) named **Details** for the business object named **Customer**.

At the moment this panel is a prototype. When you have validated your prototype you would replace this panel with a real program.

This panel is input capable. You may erase this text and add your own notes (and even pictures or images from the images palette).

Any notes you add here are saved and may be used to help you flesh out your prototypes design and/or describe its functioning to others.

Figure 1-123 Gradual conversion of the prototype into the final application

Prototypes can be defined in minutes. The prototype shows how the completed application appears and behaves. The users can see the application before a single line of code is written. If the application is browser-based, designers can email a URL so that users can run the prototype.

How the application framework operates

The application framework uses a familiar “work-with” or model view controller (MVC) paradigm to define applications, and is structured around business objects such as customers, products, and orders. The application framework also facilitates software prototyping, design, test, and implementation phases of the software development lifecycle.

LANSA supplies components with the application framework, including user management, authority management, server management, and common code management. Complete the application by replacing the prototype parts with real parts that reflect real business logic. The component structure of the applications simplifies ongoing maintenance and enhancement.

Visual LANSA Framework applications consist of components for business object navigation, search tools (or filters), search results, and business object details. Business object details, including properties and actions, are displayed in one or more tabs.

Each application, for example, Human Resources or ERP, has its own icon in the navigation pane. Selecting an application icon opens its business objects. In a Human Resources application, the employee is an example of a business object.

A business object filter is the tool for users to select a subset of information for a business object, such as using filters to search for employees in a department or a list of unpaid invoices. The search results pane presents a list of the business object instances that match the search criteria.

Command handlers, such as show product details or delete a product, are used to run specific actions against the selected instance of the business object. Typically, a command handler appears in the user interface as a tab.

Figure 1-124 shows creating applications using Visual LANSA and Visual LANSA Framework.

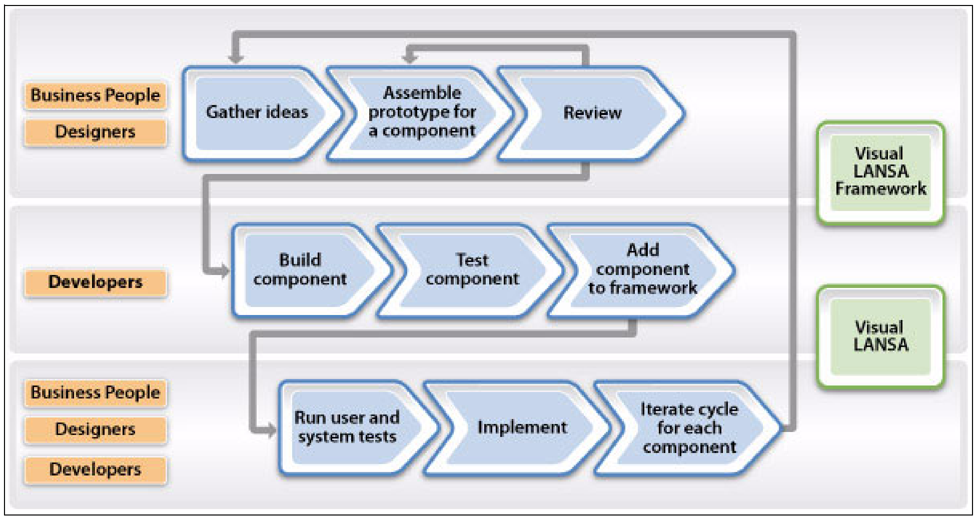
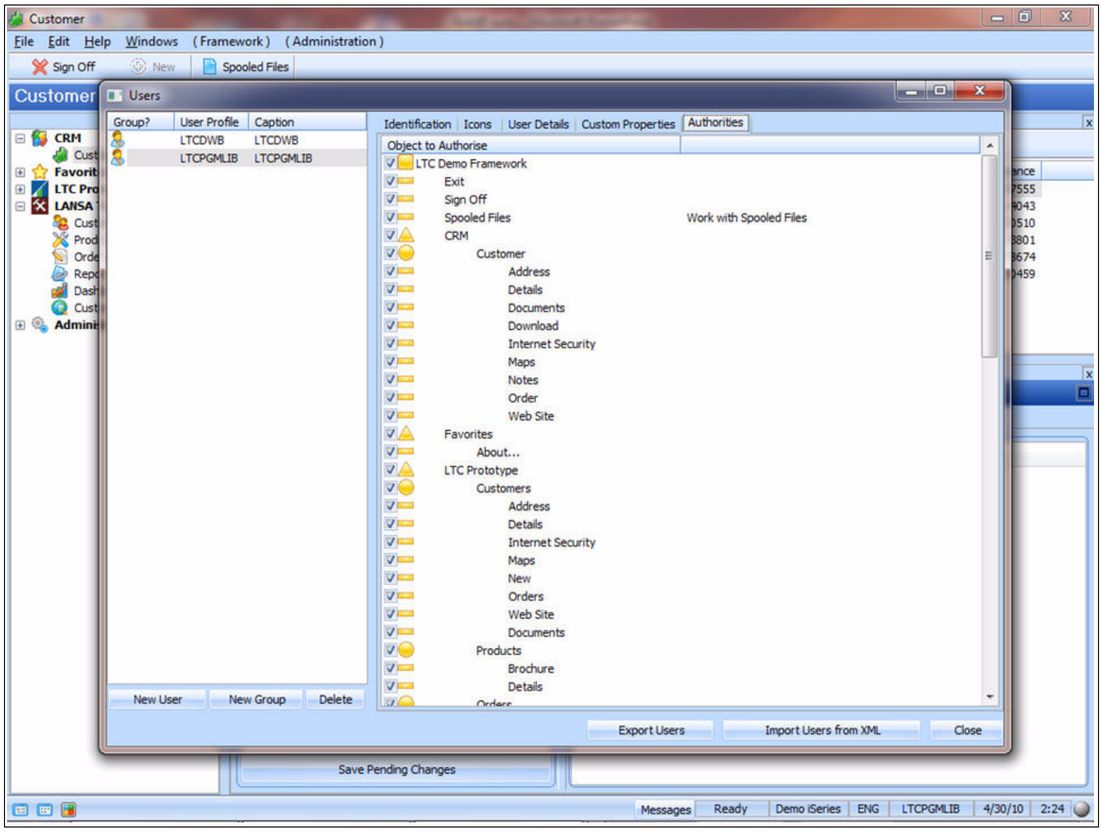


Figure 1-124 Assembling applications from components by using Visual LANSA and the Visual LANSA Framework

Flexible security options

The framework has a convenient, optional security system that supports user profiles, passwords, and authorities to objects within the framework. This system offers a moderate level of security that can interface with your own security system if needed (Figure 1-125).



Group?	User Profile	Caption
	LTCDWB	LTCDWB
	LTCPGMLIB	LTCPGMLIB

Object to Authorise	
✓ LTC Demo Framework	
✓ Exit	
✓ Sign Off	
✓ Spooled Files	Work with Spooled Files
✓ CRM	
✓ Customer	
✓ Address	
✓ Details	
✓ Documents	
✓ Download	
✓ Internet Security	
✓ Maps	
✓ Notes	
✓ Order	
✓ Web Site	
✓ Favorites	
✓ About...	
✓ LTC Prototype	
✓ Customers	
✓ Address	
✓ Details	
✓ Internet Security	
✓ Maps	
✓ New	
✓ Orders	
✓ Web Site	
✓ Documents	
✓ Products	
✓ Brochure	
✓ Details	

Figure 1-125 Built-in security in the Visual LANSA Framework

Multiple deployment options

Deploy the applications as Windows rich client on the desktop for expert users or browser-based with advanced Windows-like function. Create Ajax web applications by using the framework to achieve optimal web performance with functionality close to that of a Windows rich-client application.

The web applications are zero-client deployment and provide ubiquitous access from devices that are running Chrome, Firefox, Safari or Internet Explorer including netbooks, notebooks, and mobile devices such as iPads and Android tablets (Figure 1-126).

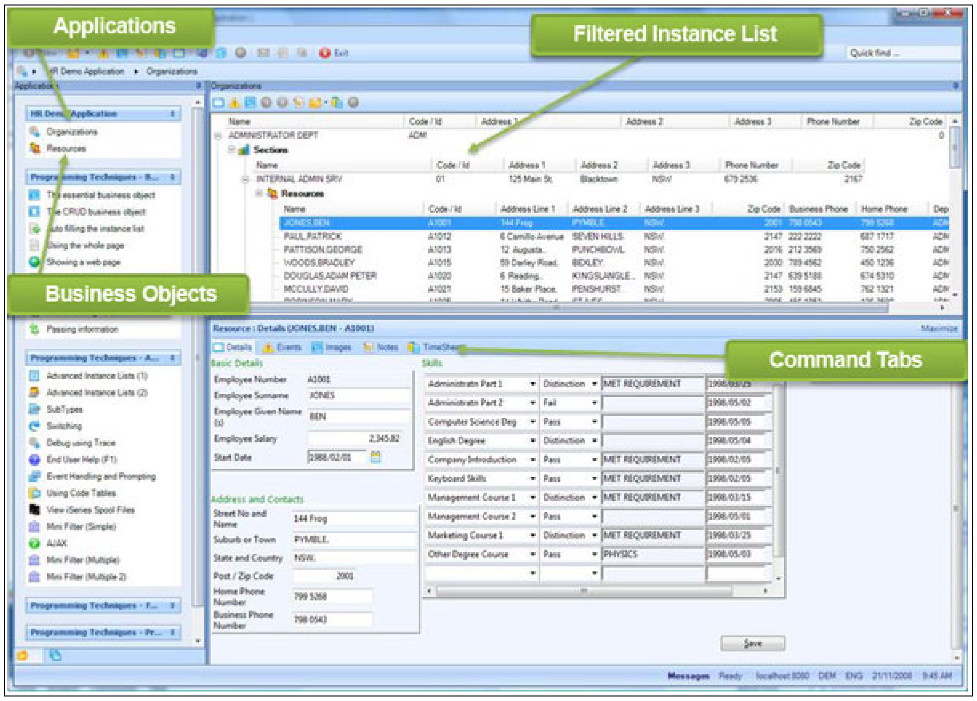


Figure 1-126   Example of a Visual LANSA Framework web browser application window

Additional information and resources

Additional product information, data sheets, and customer case studies on Visual LANSA can be found at:

`http://www.lansa.com/products/visual-lansa-ide.htm`

1.9 IBS

This section describes IBS XT.

1.9.1 What is XT?

Introduction

Xross platform technology is the new platform for IBS Enterprise software. It includes everything that you need to develop and deploy advanced enterprise applications.

XT consists of the following components:

- Server. The local installation of the dispatcher to the backend
- Client. The instance of the XT Client
- Studio. The development environment

Concept of communication

XT includes Server, Client, and Studio. The application, in this case IBS Enterprise, is stored in a different location on one or several servers, including the database or databases. See Figure 1-127.

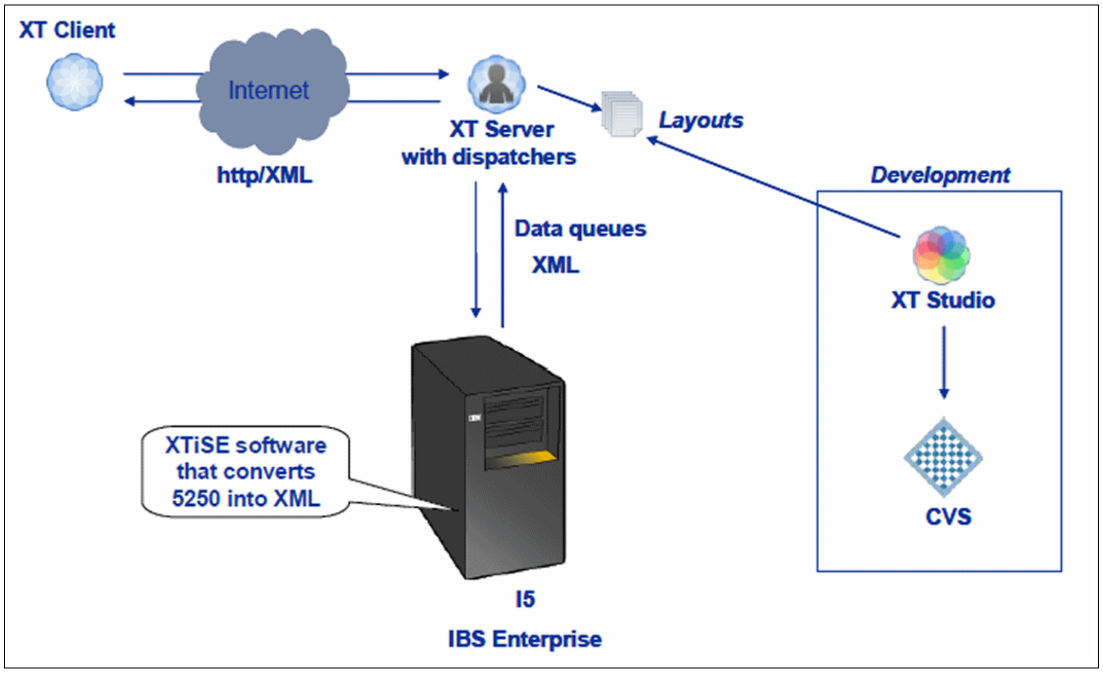


Figure 1-127 XT communication overview

The communication between the client and the application is achieved through the Dispatcher. The Dispatcher controls the clients and provides the form layouts as XML files.

In the future, IBS will work on other platforms such as Windows, UNIX, and Linux.

Development

Layouts, that is, XML files, are created or modified in the Studio, which is also connected to a version control system, CVS. Today business logic is created in RPG and then the display files are converted into XML from the Studio. To be able to make the conversion, an XT command on I5 must first be run on the newly created RPG program. The software for the XT command is called XTiSE. The Studio also includes export function for exporting layouts (XML files) to the Server, where all layouts are stored.

Server

The Server is a web server that is used as a service container. This server should be a singleton instance at the customer site in a production environment as clients communicate with this server. The server also holds a database for layout translations. All configurations for the server are done in the Administration.

Dispatcher

Services that are managed by the server are currently other instances of web servers, called dispatchers. A dispatcher is what the client logs on to, using different ports. Data is passed and received by the clients from this dispatcher, mainly in XML format. The dispatcher is connected either to an RPG-based Business System on the IBM i or to a multi-platform business system. The server can hold several dispatchers, and is therefore capable of handling several business systems at once.

Administration

Administration is the web-based administration of the server and its services. Here dispatchers are started, stopped, and configured. Attachments and the translation database are also managed from here.

Client

The Client is a thin client, that is, it does not contain any business logic at all. It is based on the Eclipse framework, and has these features, among others:

- Multiple sessions can be run, that is, several programs can be run at the same time.
- Quick links, that is, a window to where the user can drag program links from the menu.
- Connections to Excel and PDF.
- Flexible attachment handling.
- Graphical Query Builder for building SQL statements.

Figure 1-128 shows the start page of the Client.

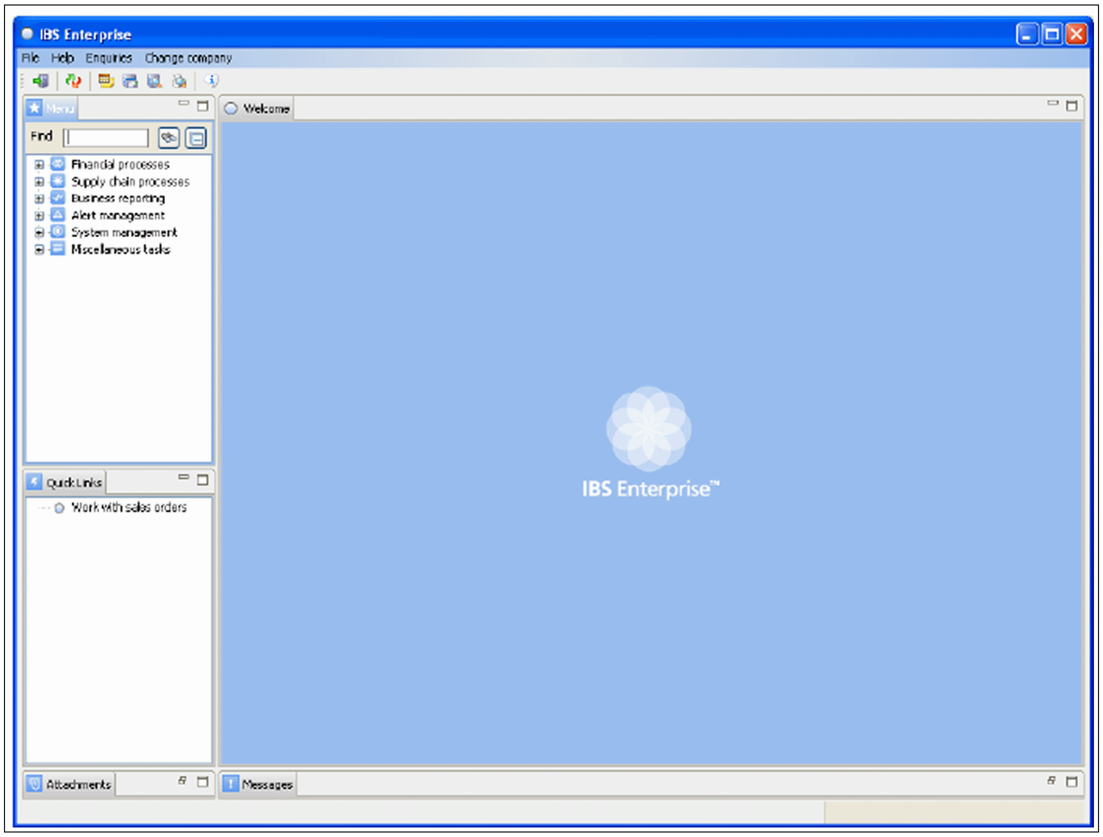


Figure 1-128 Start page of the Client

Studio

The Studio is a development tool, which today has today two different development modes: visual and XML (eXtensible Markup Language). Hence it is the place where XML layouts (UI designs) are created. In the future, the Studio will also have the modes RPG and Java.It will be the place where you change business logic as well. The Studio is a complete workbench that is based on the Eclipse framework where you have all your development needs concentrated in one common place.

The Studio includes a tool for obtaining display files from an System i system and converting them into XML files, and a translation tool for XML files to be translated from English into other languages.

CVS is the version control system that is used together with the Studio to synchronize the work with the team.

Figure 1-129 shows the XT workbench window.

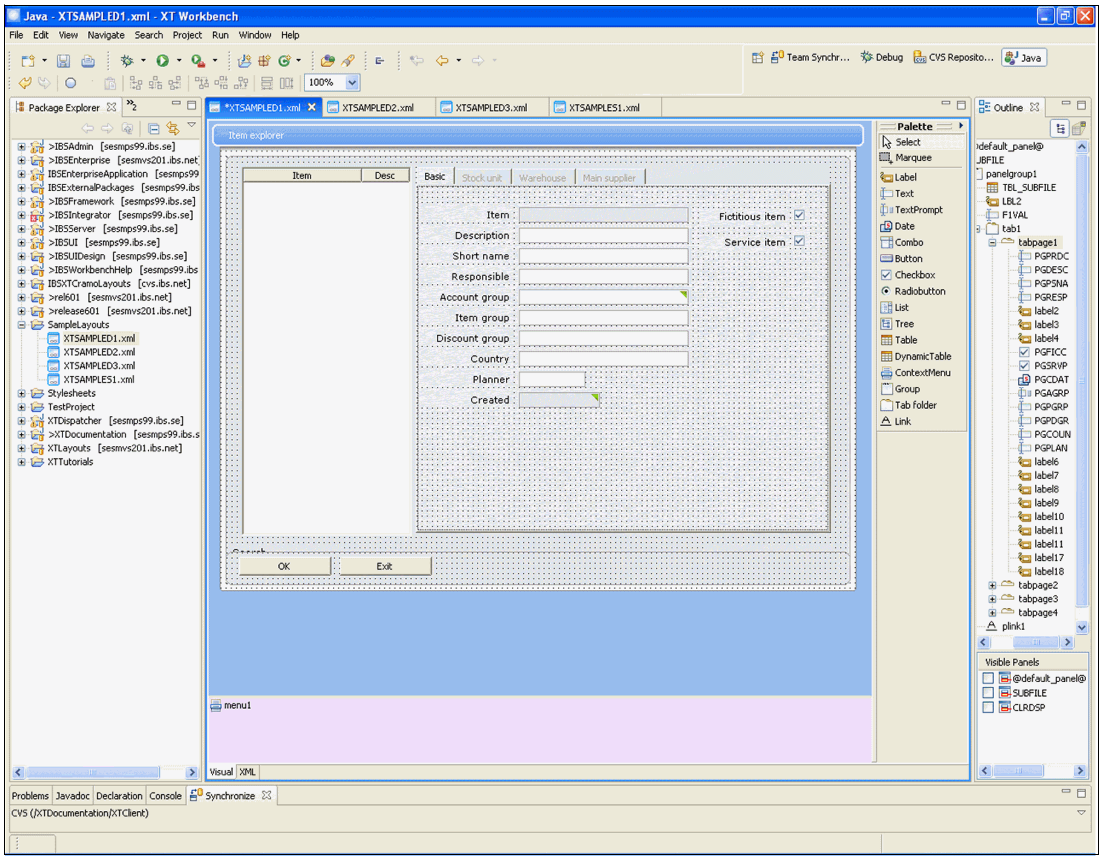


Figure 1-129 XT workbench

XTiSE

XTiSE is a program that enables the IBS Enterprise application to work with the aXT Platform. It removes the 5250 stream and transfers the data as XML on a data queue to the Server instead. Therefore, after the IBS Enterprise hash been installed, XTiSE must be run to modify the RPG code to work with the T platform. When a customer-specific modification is made, XTiSE must also be used to XT enable that RPG program.

2

Database modernization tools

This chapter describes database modernization tools. It includes the following sections:

- Database Modernization: long-term value with Adsero Optima (AO) Foundation
- Database Modernization: long-term value with Adsero Optima (AO) Roadmap
- Automated IBM i database and application modernization by using Fresche Legacy X-Analysis
- Xcase for i

2.1 Database Modernization: long-term value with Adsero Optima (AO) Foundation

This section describes Adsero Optima (AO) Foundation.

2.1.1 Your database structures: The key to your future

As discussed in 2.2, “Database Modernization: long-term value with Adsero Optima (AO) Roadmap” on page 149, a fundamental assertion of AO is that most of the value in a heritage application is encapsulated in the use of the underlying database structures and components, the coded relationship constructs, and the validation rules that are sprinkled throughout the application logic. Therefore, the first step in a long-term modernization approach is to achieve the immediate migration to DDL, and to then use this platform as the foundation for *all* other modernization going forward.

AO Foundation helps with the migration to DDL. The migration is invisible to the application (even if the source code is not available). The application then uses the new DDL-defined database as a foundation to gradually modernize the heritage applications. The result is a significant improvement to the application. This result improves the fundamental database design because it allows the implementation of proper relational structures, referential constraints, and database-enforced validations. Entity relationships are gradually moved out of the code and into the DB2 database engine. All validation rules are also gradually moved into the new DDL-defined database.

Based on significant achievements and experience in this area, AO Foundation (AOF) uses the following high-level database modernization process:

1. In phase 1, AOF converts DDS to native DDL defined constructs without changing any LVLIDs on the new database definitions. This process ensures absolute transparency to the application logic. AOF allows the conversion of a single database file, multiple selected files, one library, multiple selected libraries, or an entire system at a time to DDL. Note that to maintain LVLIDs, *none* of the logical files are converted in this phase. On completion, all constructs are new, and have improved performance characteristics.
2. AOF supports both native and surrogate migrations, with the preference for a native migration. The reason is that the objective is to gradually use the newly defined DDL constructs as foundation to move entity relationships and data validations into the new DDL defined database. Surrogates can cause problems during this extraction of relationships and validations from code into DB2. Surrogates are used, in general, only where technical requirements indicate their use.
3. A three-way parallel process can be followed:
   a. Extract all metadata from the old database into a new consolidated data dictionary and start a process of removing all duplicates and inconsistencies in the metadata. These duplicates and inconsistencies are a factor of the age of the systems (most applications have been in existence for more than two decades) and the maintenance processes that are used. It is not a reflection on developer resources. The result is improved and consistent use of metadata attributes throughout the system, with considerable integrity improvements.
   b. Use AOF to identify unique keys in the database constructs and to roll this back into the new DDL-defined tables. This process allows the implementation of a TRUE relational database model, which is necessary to implement constraints. In heritage applications and historic coding practices, unique keys were normally enforced by using logical files (In early S/38 days keys on physical files was against recommendations), or through

   application logic. Note that AOF helps drive these constructs into DB2, with no change to LVLIDs.

   c. Depending on coding practices and installation standards, use field and file descriptions as initial SQL “aliases” for long file (table) and field (column) descriptions. This process exposes more sensible, descriptive table and column names to users. This step alone improves the perception of your application. The database appears modern to executives and users with more descriptive names compared to the usual 6 - 10 character abbreviated “techno-speak” that is commonly used. AOF can automate this process completely and transparently, with no impact to LVLIDs. Alternatively, manual editing of the descriptions can be enabled to include descriptions that are more meaningful as part of a metadata enrichment process, which AOF facilitates.

4. Some of the preceding actions might affect LVLIDs, but AO provides a comprehensive repository function with extensive cross-reference capabilities, down to metadata element level, to enable full management and implementation capability.
5. After AOF is used to provide a true relational database model and removes all duplication and conflicts, the new database model is used as a foundation to start driving extra inferred relationships and rules into the database. This process includes extracting validation rules from the application logic and implementing those rules as triggers. In addition, it introduces new I/O servers (which deliver results sets to high-level application functions, gradually phasing out record level access in favor of result sets that are produced for consumption system wide), and implements system wide “Enterprise Services”.

The entire focus and philosophy of the AO Roadmap, technology, and product portfolio is designed to facilitate a modernization process with these characteristics:

- Low risk
- Gradual
- Nondisruptive
- Transparent
- Iterative

This focus and philosophy assist software developers in modernizing their systems from the inside out. Most of this modernization can be achieved during normal routine maintenance, and after (or during) the initial database modernization process.

The AO roadmap and products do not force any specific sequence, allowing the choice of preferred practices and to achieve rapid results for any business.

For more information, see the following video, which describes the process succinctly:

http://www.adsero-optima.com/nutshell-msp

2.1.2 Modern application architecture

In the AO approach, IBM i applications have been “broken” from an application architecture perspective, ever since the introduction of the ILE programming model and the massive advances in the database engine. Therefore, any long-term modernization must start at the database engine layer.

As part of any modernization project, you need to implement modern multitier application architecture, with separation between database, the UI, and business logic. Most programmers refer to this model as the MVC model, which depicts Model-View-Controller, as shown in Figure 2-1.

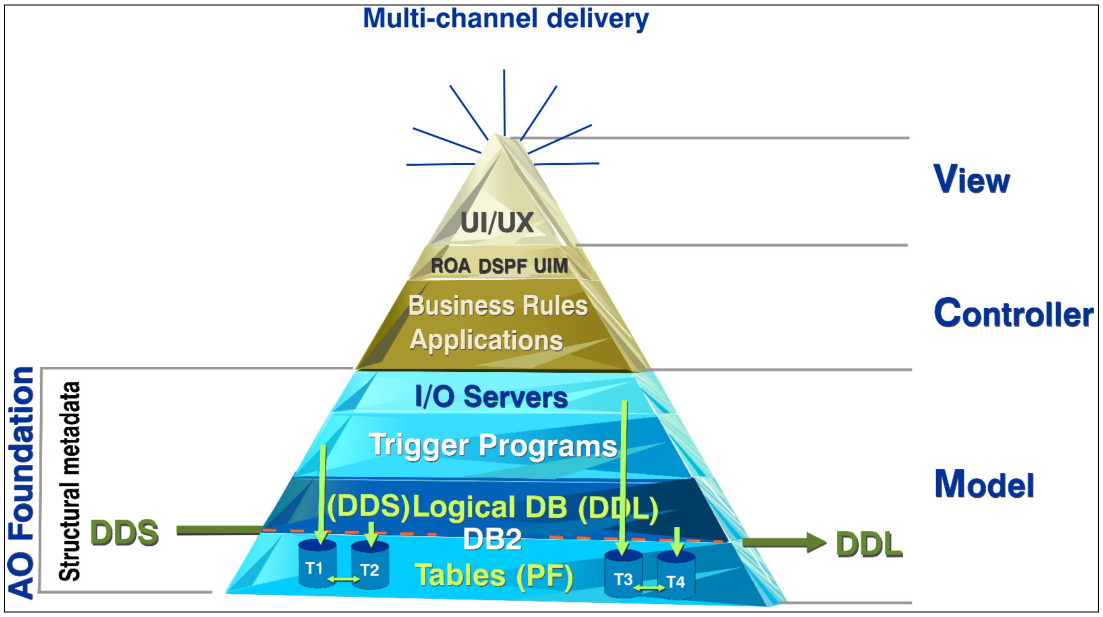


Figure 2-1 MVC model

As most heritage applications remain large monolithic constructs, with database, application logic, and UI (display file) interspersed, maintenance can be excessively burdensome and error prone. This limitation causes most agility constraints, as changes often break other functions elsewhere in the application.

These monoliths must be re-engineered in such a way as to facilitate the separation of the database, UI, and business logic, in a low risk, gradual, nondisruptive process, to remove this fundamental constraint. The database is the key to achieve this restructure.

The objective of any modernization project is to deliver a modern application architecture that provides clear separation of function and recovery of the fundamental business value in your heritage application. It must preserve the competitive advantage that is hidden in your application code.

A truly modern IBM i based application has the architectural components and structures that are shown in Figure 2-2.

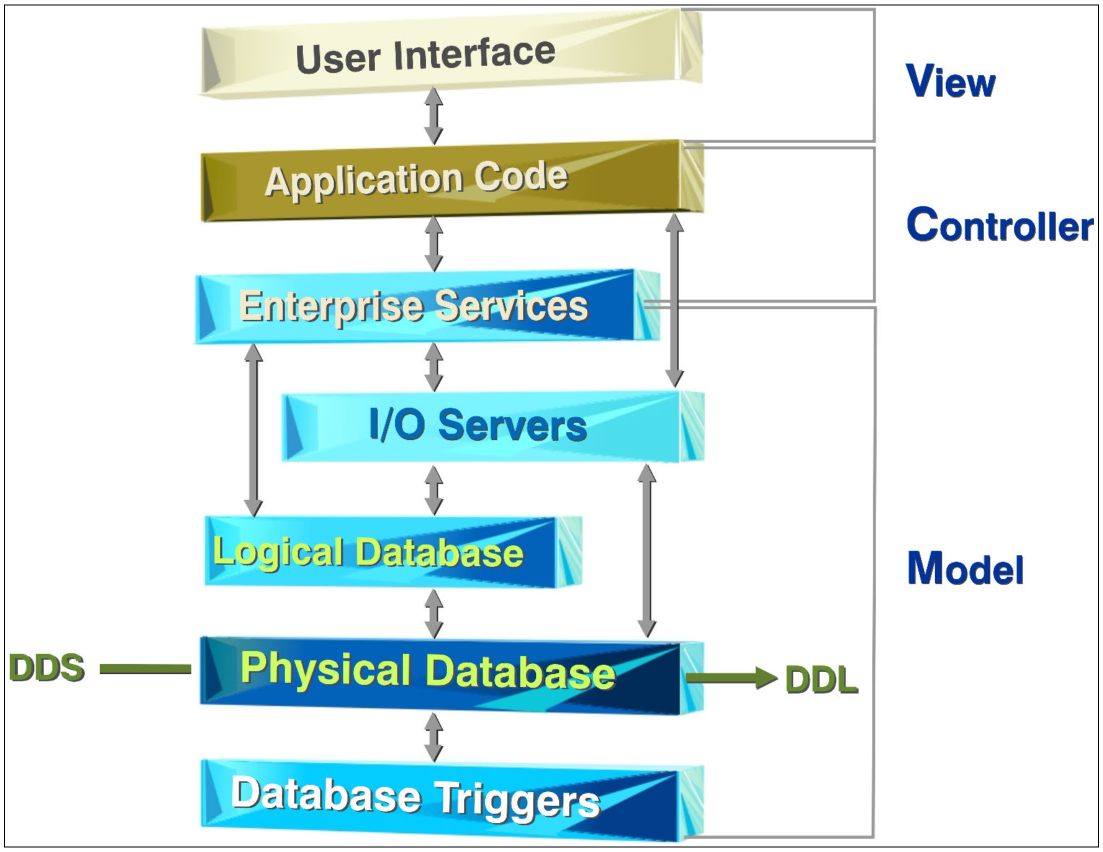


Figure 2-2 Architectural components and structures of a modern IBM i application

Implementing this architecture provides structures and components that achieve dramatic improvements in application integrity and flexibility, and produce dramatic increases in agility. It provides mechanisms for recovering the true competitive advantage encapsulated within the heritage code.

Of particular importance is the gradual transition away from record level access (RLA) or Native I/O, in favor of introducing I/O Servers that produce result sets for consumption/processing by your RPG or other high-level language (HLL) programs. The long-term objective is to get the database engine to run as much data set pre-processing as possible, as it delivers result sets far more efficiently and effectively than can be achieved in any HLL program.

Another goal is delivering system and application-wide generic services as *SRVPGMs, to be available as components anywhere in the system, similar to the ESB concept in the SOA world.

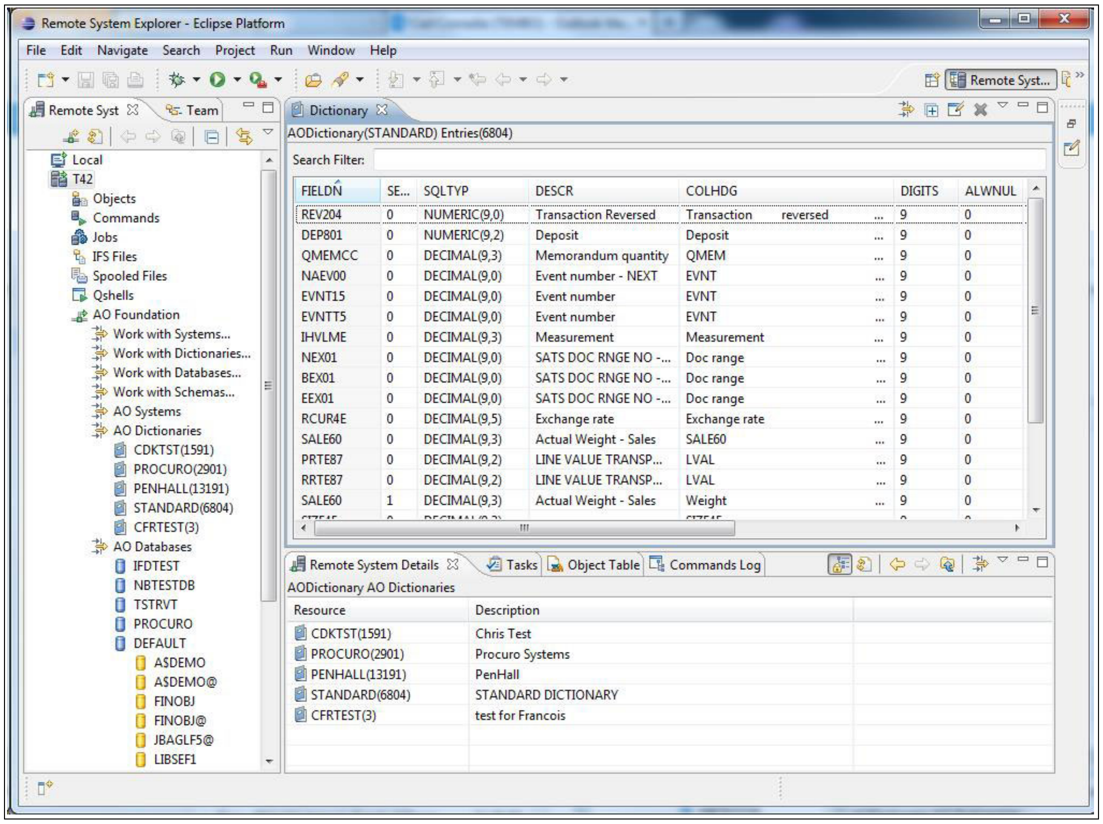
2.1.3 Using AO Foundation to achieve long-term modernization

To achieve the highlighted application architecture, complete these steps:

1. Moving the base database definitions from DDS to DDL with no LVLID changes. During this initial step, problematic or non-supported constructs are usually excluded. The power of DB2 on IBM i allows the use of any combination of old and new constructs, with few to no negative consequences.

2. Automatic synchronization (AO Replay) of your database content between your DDS production database and your new DDL database. This synchronization allows for seamless, nondisruptive switching between old DDS and new DDL production databases.
3. The generation of a new data dictionary, which becomes your new central repository for all metadata and structural metadata. This dictionary is used as to consolidate and sanitize your metadata, removing especially inconsistent attributes and removing all duplication. This new data dictionary also becomes the repository of all enrichment activities on the metadata (standard fonts, colors, icons, URLs, other behavior). It can be automatically used in future iterations of UIs, printing functions, and so on. This new repository eventually replaces your Field Reference Files (FRFs).
4. The implementation of unique keys on tables, using existing logical database constructs, to generate a true relational database at the DB2 layer. This configuration allows the generation of “automatic” constraints between entities, improving database structures and integrity significantly.
5. The exposure of long file and field names (table and column names) as aliases at a DB2 layer level. Doing so immediately improves the perceptions and experience of your users, in the way the database presents itself to business intelligence (BI) and other analytical tools.
6. Using these new DDL definitions to gradually drive data validations down into the database engine.
7. Separation of all UI, by using Rational Open Access for RPG (ROA) and a few of the top UI tool vendors. AO supports a number of these tools. By using ROA, many of the constraints imposed by the 5250 datastream effectively disappear, providing an improved UI/UX of your heritage application.
8. Using the preceding constructs to serve as foundation to gradually recover and encapsulate those components in your systems causing agility constraints or that represent competitive advantage “hidden” in your business logic.

Figure 2-3 shows the current beta version of the Eclipse-based Remote System Explorer (RSE) plug-in, which ensures that AO can be used by any of your RDP or Rational Developer for i enabled programmers. Additionally it is also integrated with IBM Data Studio to improve graphical database modeling.



AODictionary(STANDARD) Entries(6804)

FIELDN	SE...	SQLTYP	DESCR	COLHDG	DIGITS	ALWNUL
REV204	0	NUMERIC(9,0)	Transaction Reversed	Transaction reversed ...	9	0
DEP801	0	NUMERIC(9,2)	Deposit	Deposit ...	9	0
QMEMCC	0	DECIMAL(9,3)	Memorandum quantity	QMEM ...	9	0
NAEV00	0	DECIMAL(9,0)	Event number - NEXT	EVNT ...	9	0
EVNT15	0	DECIMAL(9,0)	Event number	EVNT ...	9	0
EVNTT5	0	DECIMAL(9,0)	Event number	EVNT ...	9	0
IHVLME	0	DECIMAL(9,3)	Measurement	Measurement ...	9	0
NEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range ...	9	0
BEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range ...	9	0
EEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range ...	9	0
RCUR4E	0	DECIMAL(9,5)	Exchange rate	Exchange rate ...	9	0
SALE60	0	DECIMAL(9,3)	Actual Weight - Sales	SALE60 ...	9	0
PRTE87	0	DECIMAL(9,2)	LINE VALUE TRANSP...	LVAL ...	9	0
RRTE87	0	DECIMAL(9,2)	LINE VALUE TRANSP...	LVAL ...	9	0
SALE60	1	DECIMAL(9,3)	Actual Weight - Sales	Weight ...	9	0

AODictionary AO Dictionaries

Resource	Description
CDKTST(1591)	Chris Test
PROCURO(2901)	Procuro Systems
PENHALL(13191)	PenHall
STANDARD(6804)	STANDARD DICTIONARY
CFRTEST(3)	test for Francois

Figure 2-3 Screen capture of the Eclipse-based Remote System Explorer (RSE) plug-in

Due to the immense function-rich Eclipse/RDP/Rational Developer for i interface, the balance of the figures in this document use AO UIM screens. UIM (the same architectural construct that is used for most of the IBM i operating system UI) based screens are used to highlight only those functions that are relevant for this section.

This architecture means that AO uses I/O Servers, Enterprise Services and *SRVPGMs that provide the result sets processed by both your UIM panels and the Java-based Eclipse interface.

This design requires you top recode entity relationships and validation rules that *already exist* in you RPG programs, in their new C#, JAVA, PHP, and other HLL environments. It is easier (and provides dramatic integrity improvements) o expose existing relationships and validations, by rolling this back into DB2. By using this process properly, all heritage and new applications can use the same enterprise services and I/O Servers to use result sets from the database.

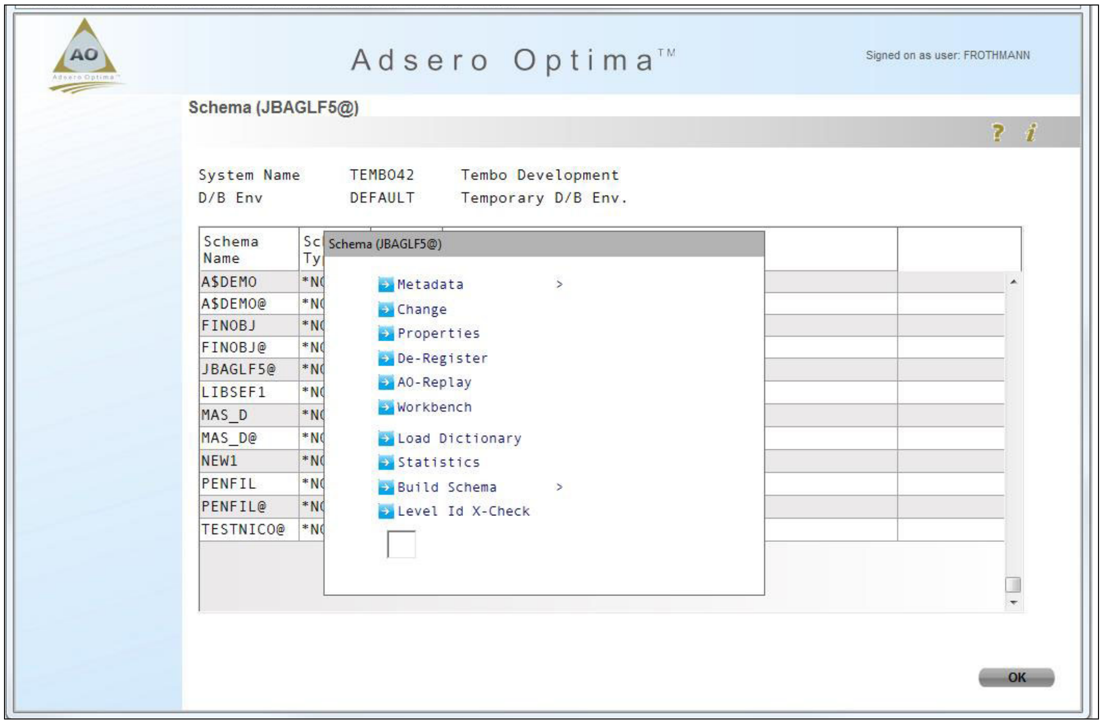
2.1.4 AO Foundation DDS to DDL migration

As indicated in the “AO in a Nutshell” animation, the structural metadata is extracted directly from the internal DB2 object structures, which means that the change can be done from DDS to DDL without source code. All actions are carried out on AO internal definitions and *never* against any DDS production versions.

For more information, see:

http://www.adsero-optima.com/nutshell-msp

Figure 2-4 shows the AO management interface.


Adsero Optima™

Signed on as user: FROTHMANN

Schema (JBAGLF5@)

```
System Name     TEMB042     Tembo Development
D/B Env         DEFAULT     Temporary D/B Env.
```

```
Schema      Sc
Name        Ty
A$DEMO      *N
A$DEMO@     *N
FINOBJ      *N
FINOBJ@     *N
JBAGLF5@    *N
LIBSEF1     *N
MAS_D       *N
MAS_D@      *N
NEW1        *N
PENFIL      *N
PENFIL@     *N
TESTNICO@   *N
```

Figure 2-4 AO management interface

One of the fundamental design considerations of AO is that it provides a single management interface for *all* development and maintenance functions. It facilitates the introduction of a true database scientist (DBS), database engineer (DBE), or database architect (DBA) type function, dependent upon the scale of the environment involved. Up to 80% of your application functions in the average online commercial transaction processing application today, are (or should be) implemented within the database engine layer.

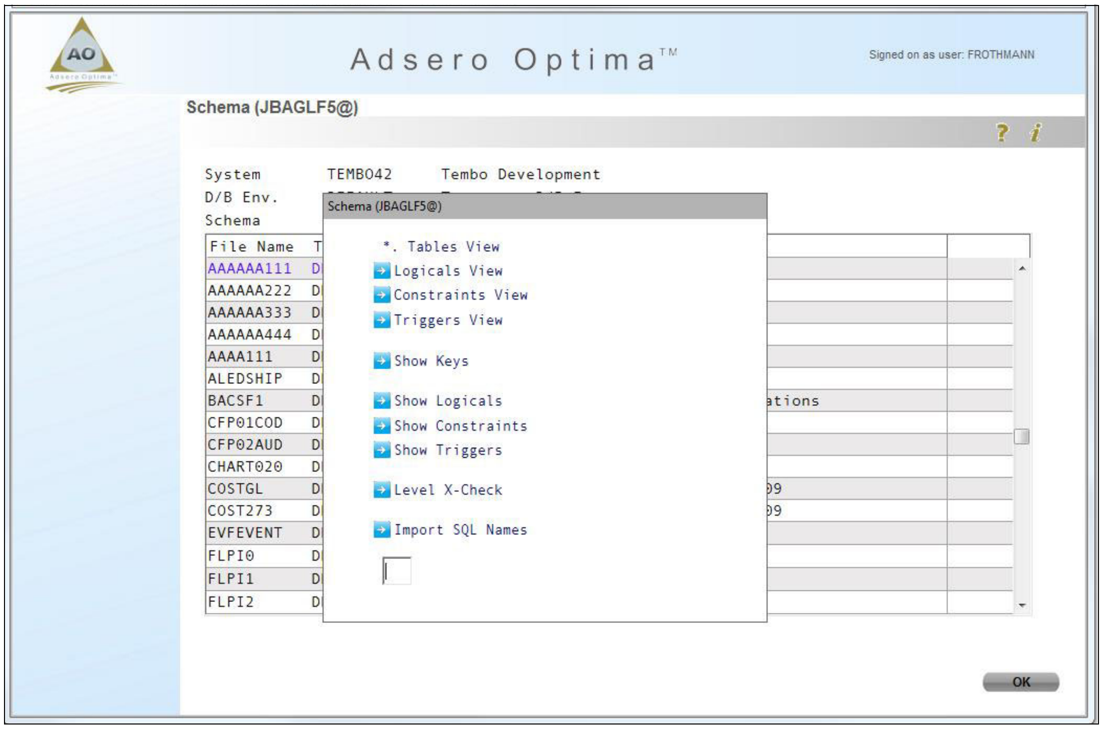
AOF continuously tracks the database “health” and statistics to allow the DBS/DBE/DBA to focus on the areas and components that deliver the most significant improvements as you modernize your heritage application portfolio. The most significant functions include AO Replay that facilitates the conversion of the old (DDS) production database to the new DDL database. This migration can be achieved while running in production, as AO ensures that the databases are synchronized.

The development team and the developer resources use the AOF Database Workbench function as they continue to modernize and improve heritage applications. This is a powerful and function rich feature within AOF.

The Statistics and Level ID Cross Check functions allows your DB staff to continuously measure progress of the modernization project and what the potential impact is.

2.1.5 AO Foundation: Database Workbench

The focus of the AOF Database Workbench function is to provide a single view of every component of the database. This view includes a logical perspective (all LF, views, and indexes), a separate perspective for constraints, keys, and triggers in addition to functions to display all database components on a single view. See Figure 2-5.



```
System       TEMB042      Tembo Development
D/B Env.
Schema
```

Schema (JBAGLF5@)

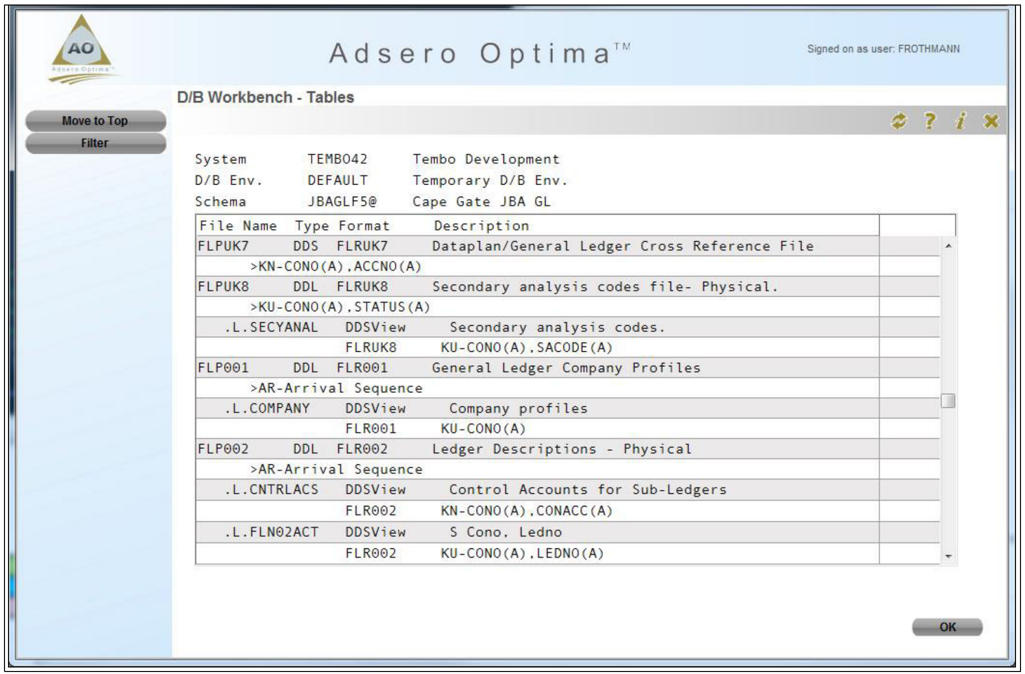
```
 *. Tables View
 Logicals View
 Constraints View
 Triggers View

 Show Keys

 Show Logicals
 Show Constraints
 Show Triggers

 Level X-Check

 Import SQL Names
```

File Name	T
AAAAAA111	D
AAAAAA222	D
AAAAAA333	D
AAAAAA444	D
AAAA111	D
ALEDSHIP	D
BACSF1	D
CFP01COD	D
CFP02AUD	D
CHART020	D
COSTGL	D
COST273	D
EVFEVENT	D
FLPI0	D
FLPI1	D
FLPI2	D

Figure 2-5 AOF Database Workbench

The next example shows how this function can be used to display all logical database constructs. This function can be used to identify logical file definitions to determine the most suitable candidate to use as a unique key. The key can be implemented at the new DDL Table level for use in generating constraints and relationship definitions.

AOF has powerful tools that are focused on making the development team more efficient and accurate when it is working with the heritage database constructs and components. The objective is to provide all the required information within a few keystrokes or mouse clicks to help make an informed, scientific decision and then implement this back into your new database with maximum efficiency. AOF also highlights the impact of that change throughout the database. See Figure 2-6.



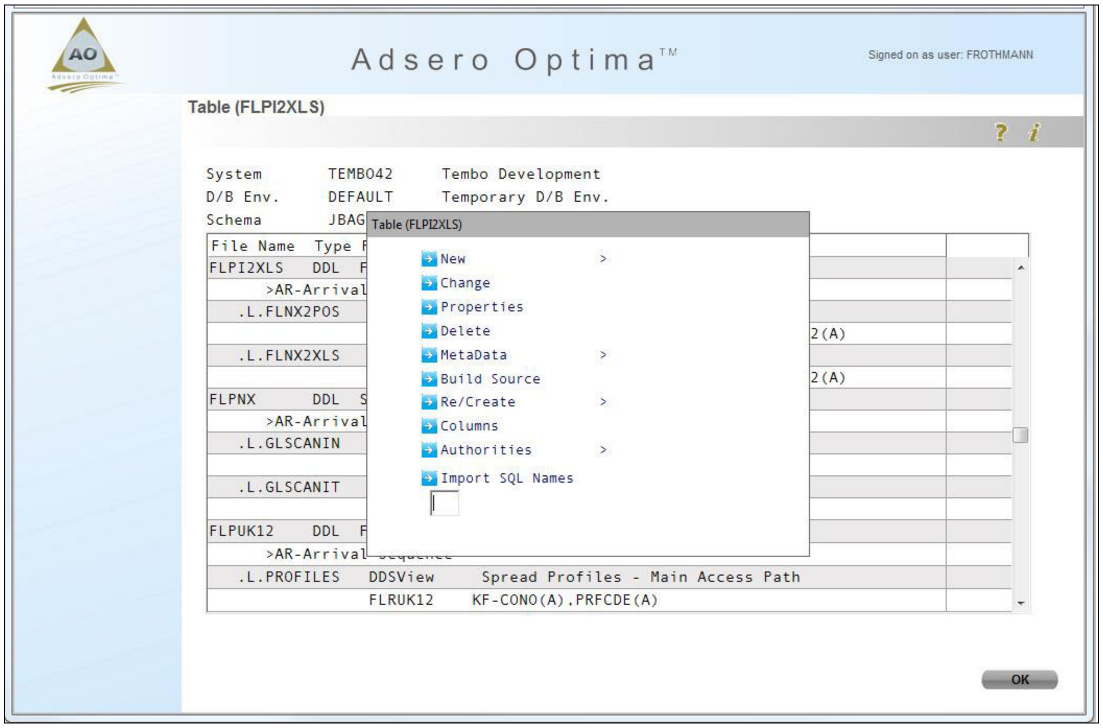
```
System        TEMB042     Tembo Development
D/B Env.      DEFAULT     Temporary D/B Env.
Schema        JBAGLF5@    Cape Gate JBA GL
File Name  Type Format     Description
FLPUK7      DDS  FLRUK7     Dataplan/General Ledger Cross Reference File
      >KN-CONO(A),ACCNO(A)
FLPUK8      DDL  FLRUK8     Secondary analysis codes file- Physical.
      >KU-CONO(A),STATUS(A)
   .L.SECYANAL   DDSView     Secondary analysis codes.
                 FLRUK8     KU-CONO(A),SACODE(A)
FLP001      DDL  FLR001     General Ledger Company Profiles
      >AR-Arrival Sequence
   .L.COMPANY    DDSView     Company profiles
                 FLR001     KU-CONO(A)
FLP002      DDL  FLR002     Ledger Descriptions - Physical
      >AR-Arrival Sequence
   .L.CNTRLACS   DDSView     Control Accounts for Sub-Ledgers
                 FLR002     KN-CONO(A),CONACC(A)
   .L.FLN02ACT   DDSView     S Cono, Ledno
                 FLR002     KU-CONO(A),LEDNO(A)
```

Figure 2-6 View of a heritage application database

Figure 2-6 shows a typical view of a heritage application database that is midway through a database modernization project. The database that is shown has been in production since April 2012 with no disruption or downtime to the users of this system (which is a 24 x 7 manufacturing installation). The remaining DDS physical file on this panel is regenerated, and functions similar to the DDL generated objects. This PF was regenerated from the extracted metadata and structural metadata with, in this case, a page size of 64 K, a standard parameterized setting in AO. AO retained it as DDS, while allowing the customer development staff to research the best option to implement a unique key on this file.

As indicated in the next panel, one of the objectives of AOF is to provide a single management and maintenance platform for your entire database, with a focus on bringing heritage database constructs and components into the new millennium. The product generates any source, which means it can also be used to educate heritage staff in the use and syntax of DDL, or for generating source code for integration into change and configuration management products to move these changes into production.

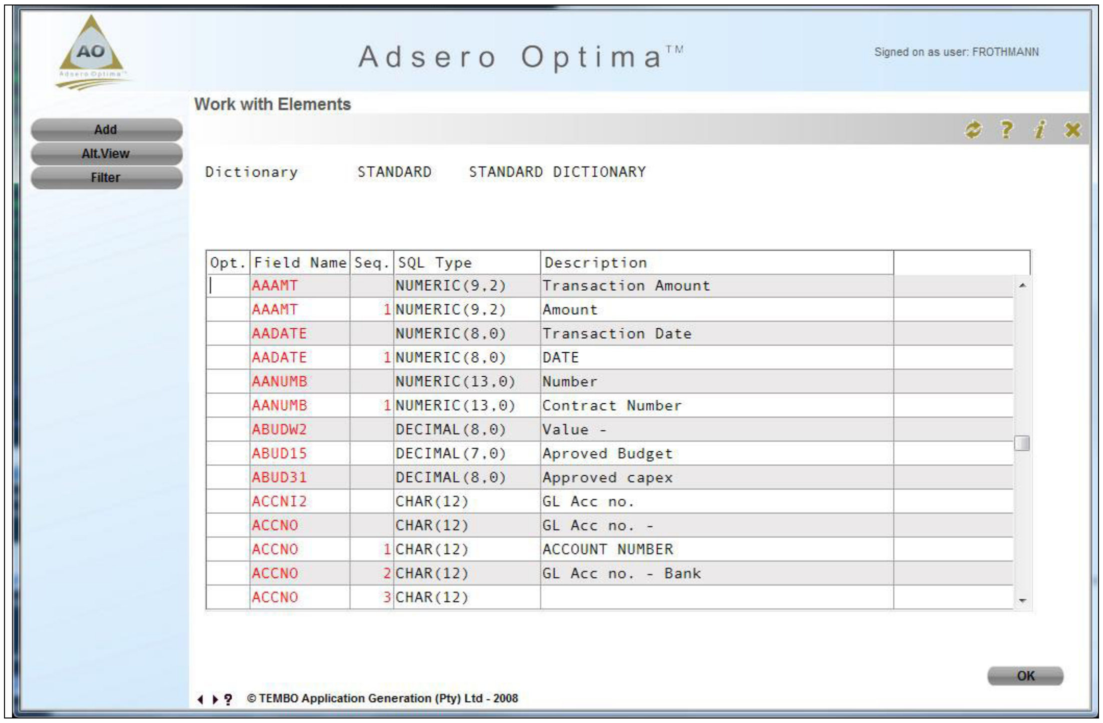
Figure 2-7 shows a single management and maintenance platform.



```
System      TEMB042     Tembo Development
D/B Env.    DEFAULT     Temporary D/B Env.
Schema      JBAG

File Name   Type F
FLPI2XLS     DDL   F
     >AR-Arrival
  .L.FLNX2POS
                                                      2(A)
  .L.FLNX2XLS
                                                      2(A)
FLPNX        DDL   S
     >AR-Arrival
  .L.GLSCANIN

  .L.GLSCANIT

FLPUK12      DDL   F
     >AR-Arrival
  .L.PROFILES   DDSView      Spread Profiles - Main Access Path
                FLRUK12    KF-CONO(A).PRFCDE(A)
```

Figure 2-7 A single management and maintenance platform

Figure 2-8 shows the data dictionary.


Adsero Optima™
Signed on as user: FROTHMANN
Work with Elements
Add
Alt.View
Filter
OK
© TEMBO Application Generation (Pty) Ltd - 2008

Dictionary STANDARD STANDARD DICTIONARY

Opt.	Field Name	Seq.	SQL Type	Description
	AAAMT		NUMERIC(9,2)	Transaction Amount
	AAAMT	1	NUMERIC(9,2)	Amount
	AADATE		NUMERIC(8,0)	Transaction Date
	AADATE	1	NUMERIC(8,0)	DATE
	AANUMB		NUMERIC(13,0)	Number
	AANUMB	1	NUMERIC(13,0)	Contract Number
	ABUDW2		DECIMAL(8,0)	Value -
	ABUD15		DECIMAL(7,0)	Aproved Budget
	ABUD31		DECIMAL(8,0)	Approved capex
	ACCNI2		CHAR(12)	GL Acc no.
	ACCNO		CHAR(12)	GL Acc no. -
	ACCNO	1	CHAR(12)	ACCOUNT NUMBER
	ACCNO	2	CHAR(12)	GL Acc no. - Bank
	ACCNO	3	CHAR(12)	

Figure 2-8 Function provided in the new data dictionary

Figure 2-8 on page 147 provides a glimpse of the function that is available in the new consolidated data dictionary. This function and tools allow you to remove all metadata duplication, by highlighting the representative master definition of that element. This process can then be used to consistently roll this master definition back into the production database, removing all duplication and inconsistencies in the metadata. The amount of duplication and inconsistencies in your metadata might be surprising, but it is due to the age of the systems and the methodologies that were used to create them.

For example, more than 50 instances of a particular metadata element were found in a customer database, with quite a few different attributes. This is in no way unique, as many systems are 20 years or older with many programmers maintaining the system during the elapsed period.

As indicated earlier in this section, during review and consolidation of your metadata and data dictionary, enrich your metadata because this process improves the quality of your system and what can be achieved with your metadata.

2.1.6 Conclusion

Massive value remains in heritage applications and compelling business justification exists to modernize heritage application assets. This process must start at the database layer, because approximately 80% of heritage functions belong in the database engine in the modern commercial application development paradigm.

Adsero Optima achieves this goal by:

1. Facilitating the (mostly automated) migration from DDS defined database structures, to native DDL defined database structures as an immediate initial step. This process becomes the absolute foundation for ANY and ALL subsequent modernization.
2. Consultation to determine a modernization strategy and plan, and to highlight future development potential.
3. Consolidating and sanitizing metadata and structural metadata because an immense amount of duplication and inconsistencies are integrated within your applications and the underlying database due to decades of neglect.
4. Introducing referential constraints into the database, with fundamental knowledge recovered from your applications. This process immediately starts to improve data integrity by orders of magnitude.
5. Removing user interface constraints (strategic partnerships) to facilitate multi-channel delivery of application function to any current or new UI delivery channel.
6. Recovering validation rules from the code base, facilitating the introduction of these validation rules as triggers directly into the database. This provides a process that facilitates consolidation of all validation rules, improve data integrity and agility, and achieve single instances (as opposed to multiple instances) of the validation rules. Additionally these rules are implemented and used consistently throughout the system, regardless from where this is accessed.
7. Providing a gradual recovery and modernization of true business unique logic, this represents the underlying competitive advantage encapsulated by your heritage application.

2.1.7 Supported platforms

This solution is supported by platforms with the following features:

- The AO Roadmap can be applied on *any* release of IBM i, or earlier releases of OS/400® and i5/OS, although some manual work must be done.
- AO Foundation requires IBM i V6R1M0 as minimum.

2.2 Database Modernization: long-term value with Adsero Optima (AO) Roadmap

This section describes Adsero Optima (AO) Roadmap.

2.2.1 Advantages of AO Roadmap

AO Roadmap has these advantages:

- You only have to maintain 10-15% of your lines of code, yet provide the same function to your business users? And in parallel, deliver improved data quality, data integrity, and performance.
- Your code becomes more manageable, maintainable, and modern as a natural consequence of this process.
- You can achieve these benefits for a fraction of the cost of replacing your applications or doing the effort manually.

2.2.2 Background

Understanding the AO approach requires acknowledgment of these fundamental observations that are taken from many modernization projects:

- The enormous amount of time, effort, money, and intellectual property that is invested in your IBM i, i5/OS, and OS/400 based applications.
- The function rendered by these applications provides a significant competitive advantage to the business.
- The value that is encapsulated in the heritage application, in addition to your database structures and the underlying rules and relationships that have evolved over the decades.
- The validation rules and database relationships that are critical for application extensibility and development of new applications, modules, and functions.

Because of the preceding three fundamental observations, AO focuses on providing a realistic, low-risk, nondisruptive, gradual process to unlock the inherent value, and massive investment that is made in your accumulated customer and business data. This focus includes the investment that is made in developing the business rules and exposing this knowledge and the rules for reuse by the business.

Based on the preceding assertions, it becomes imperative to analyze why business users consider their LOB (line-of-business) applications as “legacy”. It is important to analyze this perspective more acutely than usual, which superficially suggests that the user interface/user experience (UI/UX) is the biggest inhibitor and cause for user anxiety.

Although UI/UX is an important contributor to legacy perception, it is not the main reason why business users become disillusioned with older applications.

Close analysis indicates the primary reason why installations move away from the platform shows that the common thread is a “lack of agility”, which surfaces quickly. A lack of agility usually presents itself with business users, who are frustrated with the slow response to IT changes and the ability to use new business opportunities. IT is working exceedingly hard, but are fighting a losing battle due to immense maintenance backlogs.

A look at maintenance operations rapidly highlights the factors that cause this excessive maintenance burden and lack of agility. The applications were developed in the 1980s and early to middle 1990s, and during this the time frame, the bulk of IBM i based ISVs and applications first came into being.

Contributing to this situation, IBM successfully protects and insulates the IBM i installed base from underlying changes in the IBM i system hardware and operating system. The platform is so reliable and forgiving that applications developed in the 1980s keep running, despite massive advances in the hardware and software constructs.

Most heritage IBM i applications are characterized by huge structured monolithic programs that have been developed over the years. This development strategy eventually created complex system environments, massive maintenance backlogs, and frustrated users. Within these monoliths, 80% of the lines of code deal with database relationships, also known as entity relationships, and database validations.

The monolithic programs allowed little or no separation of function, which made maintenance increasingly problematic. Every program that manipulated the database had (or at least was supposed to have, to ensure integrity) the same 80% of code is repeated. This process led to an enormous amount of duplication, which was the only option in the 1980s and early 1990s.

Changing database relationships, adding fields, or changing validation rules meant finding every instance of that “rule” being used throughout the entire system and updating it. This is a complex, time-consuming, and potentially error prone process, which inhibits agility.

Heritage applications encapsulate the competitive advantage of business, but critical questions need to be answered to make informed business decisions about the future.

2.2.3 Determining whether your application is worth modernizing

If you are considering modernization, you need to know whether your heritage applications are of value to the business. Do your applications provide significant value when considering these questions:

1. The business success globally of companies that are using the platform?
2. The competitive edge encapsulated in the heritage application?
3. The total cost of ownership that is being delivered to clients, suppliers, and business partners?
4. The cost and risk of the viable alternatives?

If not, it does not make sense to modernize them, and you can either retire or redevelop them.

Based on experience and analysis of many installations that have moved off the platform, operational constraints and a lack of agility are consistently highlighted as the most significant factors. However, many organizations often regret replacing the IBM i applications due to a critical loss of required functionality. In addition, companies soon realize a dramatic decrease in availability and reliability were compromised because of the platform change.

If there is a significant value in your heritage applications and you determine that this value needs to be retained, you need to modernize those applications to a modern database environment such as DB2 SQL. To begin, a careful analysis of the critical operational and strategic constraints needs to be identified:

1. Does the application provide a competitive advantage, such as a unique order entry process, special stock allocation and stock management algorithms? List these items in your environment.
2. Can your current business processes be improved?
3. Does the system allow this improvement?
4. What is inhibiting service delivery models for clients, suppliers, and business partners?
5. Document the cost structure to deliver the applications.
6. What is the functional fit of your current application? As a rule of thumb, off the shelf applications deliver 60 - 75% function.
7. Are the constraints that you have experienced due to a lack of function or service delivery?

2.2.4 How much should you modernize?

Be careful of the trap of “analysis paralysis”. You do not have to modernize your entire application, as many people suggest. Your objective is to deliver maximum value to the business. These important considerations can help you determine how to extract maximum value:

1. Identify the 20% of application function that generates 80% of all transactions.
2. Perform a careful analysis of your maintenance or change requests because this highlights application functions and programs that demand higher levels of maintenance that cause most frustration.

Start with the preceding two elements and you can soon identify where the greatest return on investment (ROI) can be achieved. In parallel, you can gain confidence with the methodology and approach.

With this approach, about 50 - 60% of your application is normally modernized. The balance does not provide you with sufficient value (ROI) to invest the effort. Additionally, implement a standard policy and criteria that guides development staff to consider modernizing the code that is involved of most maintenance requests, as a standard. This process allows you to gradually modernize your system while performing routine maintenance. A coherent modernization strategy and plan should form part of the consultative process.

2.2.5 What options exist to best achieve the wanted results?

Often, the only lasting modernization approach must start at the database definition level.

Any other adjustments or maneuvers might be tactical at best, providing only a brief respite. In the long term, these adjustments do not remove the barriers to a permanent solution. This is key for unlocking and reclaiming your heritage.

It is imperative to get your database definitions from the old DDS definitions into DDL. This step is the first and foundational requirement to start long lasting modernization. AO then facilitates moving the bulk of your relationships and validations gradually out of your application logic into the database engine. You want DB2 to do all the “heavy lifting” work for you, allowing you to focus on delivering innovative business solutions and logic.

This opens a considerable amount of value and benefit, which can be achieved with these characteristics:

- Minimum disruption to your users
- The lowest risk
- Greatest ease
- Fastest speed possible
- Delivering a significant return on investment

2.2.6 The business benefits that can be achieved by modernization

Cape Gate decided to modernize rather than replace their IBM i applications resulting in costs that equated to just 18% of the total hardware, software replacement, and services costs of a replacement solution. The company rearchitected a single function as a trial to test the AO approach in their commercial application and reduced the runtime on a long running job from more than 20 hours every month down to 20 minutes.

In addition, embracing change, implementing DB modernization provides the following extra benefits:

- Use the latest DB2 SQL advancements
- Achieve exceptional cost savings:
  - 15% to 18% of expected replacement costs
  - 10% to 20% to perform a manual modernization if possible. Larger installations cannot implement a manual modernization, with some installations requiring on average a project timeline of almost 19 man-years.
- Gain access to a large young technical resource pool for future systems maintenance
- Benefit from reinvigorated developers and user resources
- Significant potential workload savings
- A modern new database structure and methodology for on-going application development

Tip: Your database structures are the *key* to your future applications and modernizing your heritage.

Figure 2-9 shows the long-term modernization process.

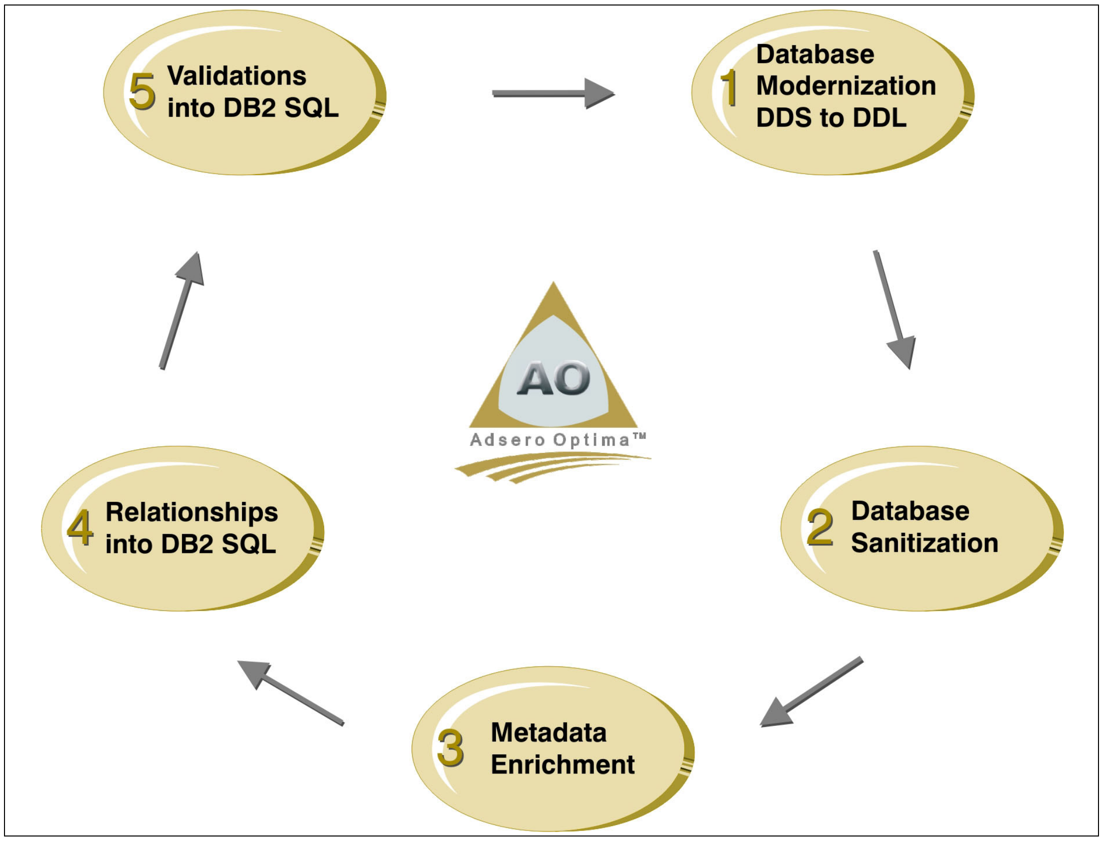

1 Database Modernization DDS to DDL
2 Database Sanitization
3 Metadata Enrichment
4 Relationships into DB2 SQL
5 Validations into DB2 SQL
AO
Adsero Optima™

Figure 2-9 Steps for a long-term modernization process

The first step in a long-term modernization process is to convert the database schema from DDS to DDL, unlocking a host of extra functions in the SQE interface. This process can be achieved transparently to your heritage applications without the requirement of recompiling any programs. A few caveats exist: see the AO Inspector output.

This process can be achieved with or without the need for surrogate logical files (AO supports both native and surrogating as viable migration methods). Surrogate logical files masquerade the change to the underlying database, allowing older systems to access the new database files without the need for recompilation. Surrogates are beneficial when new applications are being implemented, and older applications remain unchanged. However, if you aim to use the competitive advantage and value of your older applications, approximately 80 per cent of the lines of code (all lines of code that implements validations and enforcing data relationships) currently in your heritage applications eventually ends up in the database engine. By eliminating surrogate local files, you end up with a far more efficient approach to enabling long-term modernization.

In most cases, the use of surrogates is unnecessary and arguably detrimental to future modernization and development efforts. AO uses unique technology to do the DDS to DDL conversion.

The aim of this exercise is to build DDL from the cloned DDS structural metadata without changing level IDs in any way. This is a significant requirement when you are migrating and upgrading DDS to DDL in phase 1, which facilitates an easy, nondisruptive, low-risk process that is entirely transparent to heritage applications. The following animation provides a more concise description of what AO does:

http://www.adsero-optima.com/nutshell-msp

Additionally the following tool called AO Inspector is available at no additional cost to assist you in analyzing your database and highlighting all the considerations during your modernization project:

http://www.adsero-optima.com/content/adsero-optima-inspector

2.2.7 AO Inspector

The objective of AO Inspector (AOI) is to provide factual information about the size and complexity of your schemas and databases, which assists during your planning process by highlighting specific information from your databases. This information is used to define tactical solutions, and provides the basis for your modernization strategy.

AOI also highlights the state of your database, providing rich statistics and insight into your underlying database structures in a single place. Of particular importance are statistics that allow you to gradually convert database structures to structures that conform to a true relational database model.

AOI Output for scoping effort and to highlight areas that require attention includes the following information, which is based on your selection criteria:

- The total count of DDS defined physical files and DDL defined tables
- The total count of DDS defined logical files and DDL defined views and indexes
- The total count of all database constructs

As a general rule, an experienced database scientist can manually generate DDL constructs from DDS by using standard tools (iNavigator or the APIs) at an average rate of 2 hours per physical file and all its associated logical files, excluding the migration of the data.

Additionally, AOI extracts and presents all the following information:

- Unsupported DDS functions in DDL (PF):
  - Keyed, Not Unique
  - Program Described File
  - Multiple Members
  - Alternate Collating Sequence
  - Zero Members (usually FRF)
  - Keywords
- Unsupported in DDL (LF):
  - Derived Fields
- Physicals or Tables (Issues) (individual counts and lists for both DDS and DDL):
  - Arrival sequence
  - Level Check = *NO
  - *PUBLIC not *EXCLUDE

  - Re-Use Deleted Records
  - Not Journaled
  - *NULL Key Fields
- Constraint Definitions (individual counts and lists for both DDS and DDL):
  - Total Constraints
  - Primary Keys
  - Unique Constraints
  - Check Constraints
  - Referential Constraints
- *BEFORE Triggers Definitions (individual counts and lists for both DDS and DDL):
  - Total *BEFORE
  - *BEFORE/*INSERT
  - *BEFORE/*DELETE
  - *BEFORE/*UPDATE
- *AFTER Triggers Definitions (individual counts and lists for both DDS and DDL):
  - Total *AFTER
  - *AFTER/*INSERT
  - *AFTER/*DELETE
  - *AFTER/*UPDATE
  - *AFTER/*READ
- DDS Logicals (total counts and Select/Omit at LF level):
  - Views
  - Surrogate Views
  - Joins
  - Multi-Formats
- DDL Logicals (total counts and Select/Omit at LF level):
  - EV Indexes
  - BR Indexes
  - Views

Additionally, AO also retrieves program observability statistics to highlight potential considerations. Part of the modernization effort is to move to the latest release of the operating system, with IBM i V6R1M0 as the bare minimum when complete.

Of particular interest are two statistics that are retrieved (see the preceding items). Many of the express and highlight design and coding practices are from the 1980s and early 1990s. This coding was before the major DB2 database enhancements were made on the platform. For more information about how AO facilitates implementing a true relational database from heritage database constructs, see 2.1, “Database Modernization: long-term value with Adsero Optima (AO) Foundation” on page 138.

Although a relational database engine has been available on IBM i since the announcement of the platform, most installations and ISVs did not use it as such. The reason for this is application age and the elapsed time during which enhancements for DB2 for IBM i were made (25+ years). Added to these factors is the maturity and acceptance gained by relational database technology globally for IBM i and its predecessors.

As a result, the database engine made little or no enforcement of database validation rules. However, it is implemented in application logic at every instance whenever that database file is used (that is the theory of what is supposed to happen). The same applies to database relationships. This theory means that relational database is achieved by both DB2 and by the application logic. This situation is the biggest cause for agility constraints.

If you use the latest capabilities of DB2 on IBM i, you can achieve these goals relatively quickly:

1. Reduce the code base of your application by as much as 85% over time.
2. Improve your agility, allowing you to compete head-on with your competitors and provide a significant platform for true application innovation.
3. Use your competitive advantage and unlock the enormous value that is hidden in your heritage application's intellectual capital.

2.2.8 Conclusion

Based on extensive experience, great value remains in heritage applications and that compelling business justification exists to modernize heritage application assets. This modernization process must start at the database layer because approximately 80% of heritage functions belong in the database engine in the modern commercial application development paradigm.

Adsero Optima achieves this goal through these methods:

1. Facilitating the (mostly automated) migration from DDS defined database structures, to native DDL defined database structures as an immediate initial step. This process becomes the absolute foundation for *all* subsequent modernization.
2. Consultation to determine a modernization strategy and plan, and to highlight future development potential.
3. Consolidating and sanitizing metadata and structural metadata, as an immense amount of duplication and inconsistencies might be integrated within your applications and the underlying database due to decades of neglect.
4. Introducing referential constraints into the database, with fundamental knowledge recovered from your applications. This process immediately starts to improve data integrity by orders of magnitude.
5. Removing user interface constraints (strategic partnerships) to facilitate multi-channel delivery of application function to any current or new UI delivery channel.
6. Recovering validation rules from the code base, facilitating the introduction of these validation rules as triggers directly into the database. This step provides a process that facilitates consolidation of all validation rules, improves data integrity and agility, and achieves single instances (as opposed to multiple instances) of the validation rules. Additionally these rules are implemented and used consistently throughout the system, regardless from where it is accessed.
7. Providing a gradual recovery and modernization of true business unique logic, it represents the underlying competitive advantage encapsulated by your heritage application.

2.2.9 Supported platforms

This solution is supported by platforms with the following features:

- The AO Roadmap can be applied on *any* release of IBM i, or earlier releases of OS/400 and i5/OS, although some manual work must be done.
- AO Inspector requires IBM i V5R4M0 at a minimum.
- AO Foundation requires IBM i V6R1M0 at a minimum.

2.3 Automated IBM i database and application modernization by using Fresche Legacy X-Analysis

This section describes automated database and application modernization using X Analysis, an IBM i application analysis, management, and modernization suite of tools by Fresche Legacy.

IBM i systems have been developed over decades and contain vast amounts of valuable information and data. However, companies need to modernize to take advantage of real-time access to critical information, improve user experiences, implement better security, optimize code bases, and adopt new ways of sharing and using the information and data that is stored in the IBM i.

A modern IBM i database has much more than just DDS as its source: it has detailed information about its use, description, and architecture that is explicitly embedded in the Data Definition Language (DDL) itself. IBM i applications traditionally already have much of this detail, but it is spread around the application code in RPG, COBOL or CA 2E (Synon).

To modernize, you must harvest all these valuable application and data-related artifacts. Typically, companies tackle this process manually, which places a huge demand on existing resources, often impacting ongoing maintenance and development. X-Analysis automatically gathers and derives the entire application map of architecture, business rules, and constraints from the application and database code. It then consolidates this information back to the database design and uses the output to feed the automated modernization of the database to a rich, DDL variant. This process creates the database and copies the data into the modernized database without the need to recompile existing programs.

X-Analysis also allows users to minimize risk and maximize productivity during database and application modernization projects through graphical impact analysis, structure charts, data flow diagrams, modeling, problem analysis, metrics analysis, and reporting features. The use of pseudo code, interactive diagrams, and editing capabilities empower even non-IBM i experts to collaborate and contribute in these mission-critical modernization projects.

The database modernization that is provided by X-Analysis includes data quality analysis, test data extraction, sensitive data masking, and regression testing automation. It also performs field expansion and Unicode conversion automation where design changes are required as part of the database modernization. See Figure 2-10.

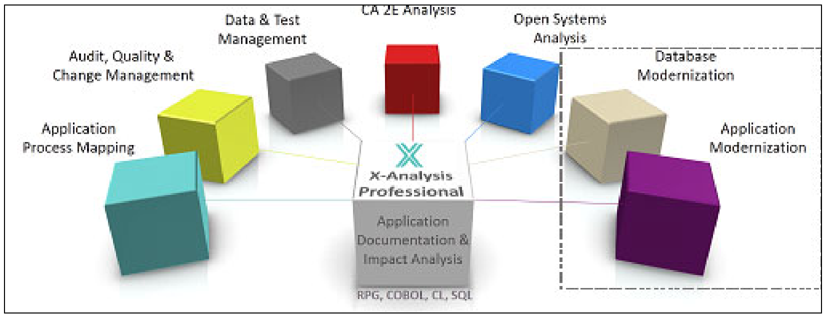

CA 2E Analysis
Data & Test Management
Open Systems Analysis
Audit, Quality & Change Management
Database Modernization
Application Process Mapping
Application Modernization
X-Analysis Professional
Application Documentation & Impact Analysis
RPG, COBOL, CL, SQL

Figure 2-10 Database and application modernization with X-Analysis

2.3.1 Did you know?

Earlier versions of the DB2 database did not offer the ability to describe relationships between tables such as foreign key constraints. As a result, relationships were enforced in the program logic. Relationships can be as simple as foreign keys or embedded deeply within the application logic. An example of this complexity is where a transaction type code held on a control table might form part of a transaction number field in a historical table. In this case, there is an implicit relationship between these two fields.

The effort companies go through to identify information like this can extend over many years. X-Analysis can extract the same information with accuracy in just a few hours. X-Analysis uses algorithms that have been built over 25 years to search and map the patterns of logic inside RPG, COBOL, and CA 2E (Synon) systems that contain business rules and database relational constraints. In addition, the automation means that the logical data models can be kept in sync with ongoing changes to the system.

After it is explicitly defined, the relational model or logical data model can be reused in a number of scenarios such as understanding application architecture, modernizing databases, business intelligence, test data management, and even ERP upgrades.

2.3.2 Business value

The Fresche Legacy database modernization offerings are structured to help provide the following business benefits:

- Reduced effort and cost by using automated modernization tools
- Accurate project planning, scheduling, and constant visibility of project scope
- Reduced risk of human error through automation and quality output
- Reduced overall impact on IT resourcing required to modernize
- Improved security and performance

Key technology features

The key features that are embedded in Fresche Legacy's suite of database modernization tools include these:

- Application and database metrics for planning and scheduling
- Automated project scoping and de-scoping through application area mapping
- Automated long table and column name creation
- Automated business rule and foreign key constraint extraction from RPG/COBOL/CA 2E
- Modernization problem analysis reporting
- Interactive graphical entity relationship diagrams of derived constraints and business rules
- Automated field expansion and Unicode conversion through RPG code
- Test data extraction, masking, and regression test automation
- Impact analysis and cross referencing through multiple levels
- Intelligent reuse of implicit logical models and rules
- Improved data security with test data masking

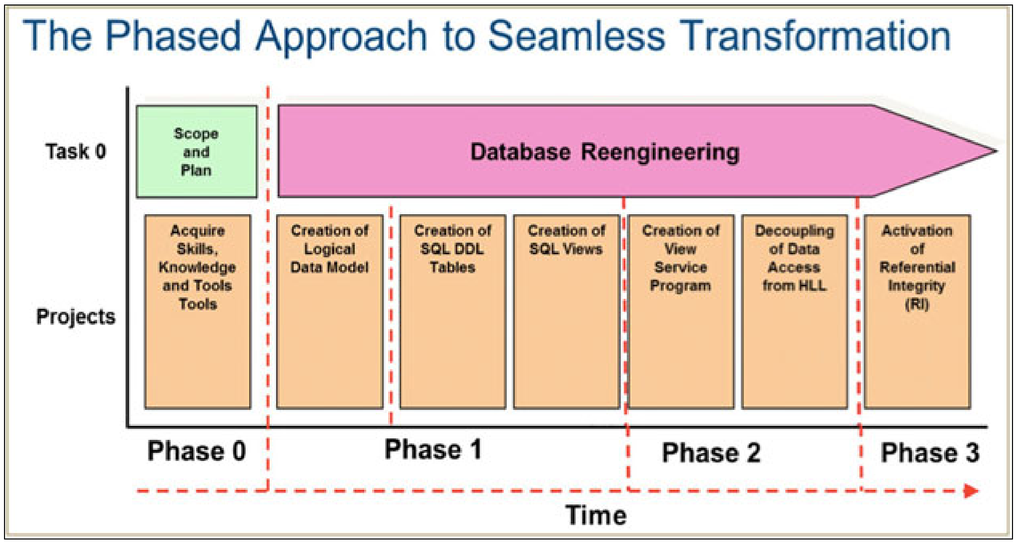
2.3.3 Solution overview

For most IBM i application databases, the database source code has little if any information about the logical data model. The only reliable resource for database modernization, the logical data model, is embedded in millions of lines of application code of varying styles that is written over decades. Accurately deriving an explicit model for database modernization manually might take years. X-Analysis analyzes the entire code base automatically to derive the logical data model and output as an explicit logical and physical model. The complete logical and physical model in X-Analysis is then used to automate the modernization process.

Any system modernization initiative starts naturally with the database. One of the primary reasons for this is that in a business application, 80% of its architecture and domain fit is determined by the architecture of the data model. Therefore, having an explicitly defined logical model that can be used both programmatically and manually by developers is a key requirement that all modern applications have. By moving much of the descriptive and referential information back into the database, new programs do not have to contain code for these conditions, producing cleaner code that is purely focused on transactional functions.

A modern database that has reusable logic that is embedded in it also provides opportunities for further application modernization. Development in modern languages or development of new UI device types can benefit from database modernization because coding effort and errors can be reduced. Changes can be implemented quicker and more consistently. For more information about these benefits, see *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

Fresche Legacy's tooling and processes follow a staged approach to modernization of the database as shown in Figure 2-11 and as described in *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.



The Phased Approach to Seamless Transformation
Task 0
Projects
Scope and Plan
Database Reengineering
Acquire Skills, Knowledge and Tools Tools
Creation of Logical Data Model
Creation of SQL DDL Tables
Creation of SQL Views
Creation of View Service Program
Decoupling of Data Access from HLL
Activation of Referential Integrity (RI)
Phase 0
Phase 1
Phase 2
Phase 3
Time

Figure 2-11   Overview of the database modernization phased approach

Following the phased approach to seamless transformation outlined in the *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185, this solution guide is divided into these sections:

- Planning and scoping
- Automated creation of the logical data model
- Automated creation of SQL DDL tables and views
- Creation of view service programs
- Decoupling of data access from high-level languages
- Activation of referential integrity
- Data access layer modernization

Planning and scoping

This solution guide is primarily about database modernization. However, the features and methodologies that are used for database modernization are equally relevant to broader application modernization initiatives.

Underestimation of complexity is often cited as the main culprit in scope creep. In its CHAOS report, the Standish Group reports that on average, 60% of IT projects fail, are late, or suffer from budget overruns. X-Analysis automatically provides compete and detailed project scope with corresponding detailed complexity data.

Building an accurate modernization plan

Heritage IBM i applications tend to be large and full of unexpected complexities from decades of various development techniques, skills, people, and compilers. For many IBM i modernization projects, companies have an idea of where they want to end up, but the challenge is often how to get there and even where to start. X-Analysis simplifies the startup process by providing high-level analysis from any entry point in a system and an ability to provide specific metrics, designs, and code architectures in a simple, graphical manner.

In modernization planning, the challenge is to create a tangible, referenceable modernization plan that has little to no impact on the business while it is being carried out, and that results in added value. The plan must take into account any unique technical characteristics of the current application. These characteristics vary widely even within a subsetted project scope. X-Analysis provides relief from this complex task as it identifies all application characteristics and highlights the impacts of change and avoiding the knock-on effect.

Analyzing applications can be an exhaustive, costly, and time-consuming manual task. The primary reason for the difficulty of this effort is the variation of older designs and coding styles and the difficulty quantifying the analysis results in a way that can be reused in effort estimations. X Analysis automation is robust for any style of code or design. The following are some examples of the potential problems in modernization analysis that are alleviated by X-Analysis:

- Disruptive heritage technologies in project scope: GOTOs, GSORTS, O-Specs
- Select omits in logical files
- Joint logical files
- Multi-format files
- Multi-member files
- The inferred relationship between program variables that trace back to different database fields, including the passing of these parameters through a call stack
- Generic program variable calls used instead of direct program calls
- File updates and I/O through a generic program that is driven by either program hardcoding or directives that are stored in a data file
- Dramatically varying program or window complexities across project scope

All source change management tools have some form of basic cross-referencing. This form is driven by object reference information that is extracted from the object metadata or observable information. None have the level of information that is required to build a detailed and complete database modernization plan.

X-Analysis contains automated modernization features that reuse extracted design metadata, while simultaneously using this data to provide the details required to build a comprehensive and accurate plan.

Discovery service with X-Analysis

Correctly planning a project is critical to minimize the risk of overspending. This requirement is true of IBM i modernization projects, where new stakeholders often have little knowledge of these large, complex systems.

Fresche Legacy provides a Discovery Service that combines the powerful features of X-Analysis with many years of successful modernization project planning to help build the strategy and plan that aligns with the organization's modernization goals. Discovery Services can be constrained to a concise technical analysis and scoping exercise, or can facilitate the engagement of all key project stakeholders driving to a thorough and rationalized plan.

Figure 2-12 depicts a high-level view of a typical Discovery Service that can help ensure a successful modernization project.

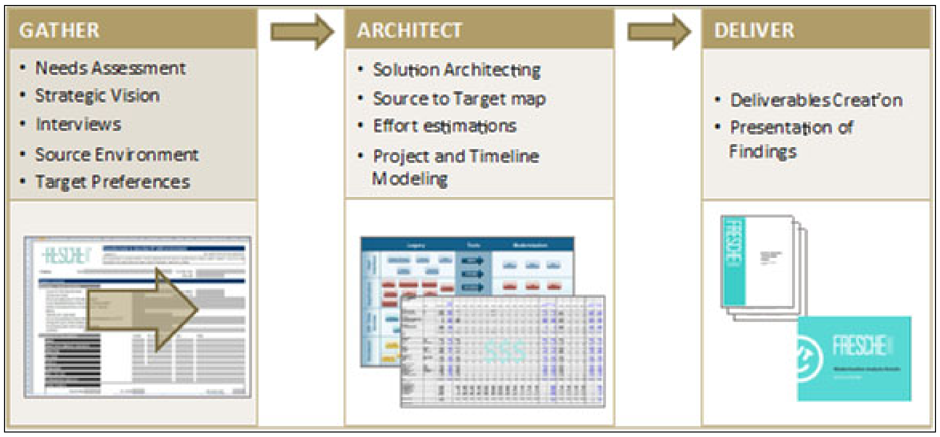


Figure 2-12 Discovery Service with X-Analysis

The following is a general framework of how to accurately scope and plan a modernization project using X-Analysis. This framework is also used as a key component in most Discovery Service engagements.

1. Map to application objects with X-Analysis using application areas
2. Review metrics per application area
3. Review problem analysis per application area
4. Map interface and integration issues
5. Create a test plan
6. Define and scope modernization plan per application area

1. Divide the system into application areas

Business stakeholders have ownership responsibility for their respective system functions, and so provide funding for modernization projects. By automatically mapping stakeholders' key functional and data requirements to the objects that relate to these business functions, an inventoried level of ownership and scope can be visualized and managed. This collection of objects is referred to as an application area.

Figure 2-13 shows application areas in X-Analysis.

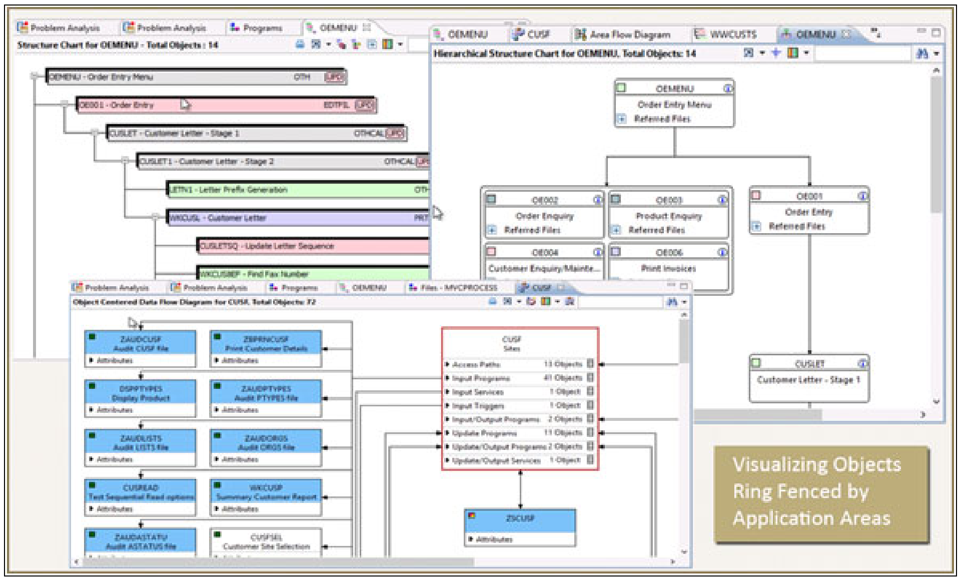


Figure 2-13 Application areas in X-Analysis

Application areas can further be subdivided into subapplication areas, such as for different technology types. This division can be useful for planning a modernization project because splitting into technology application areas improves the accuracy of the metrics.

2. Review metrics per application area

Instead of scoping and planning projects at an entire project level, common metrics at an application area level can help outline which parts are complex. This process gives modernization project managers a first look at estimates of future bottlenecks. Common metrics for estimating program complexity include lines of code, cyclomatic complexity, Halstead volume, maintainability index (MI), and the number of files and displays that are used by a program.

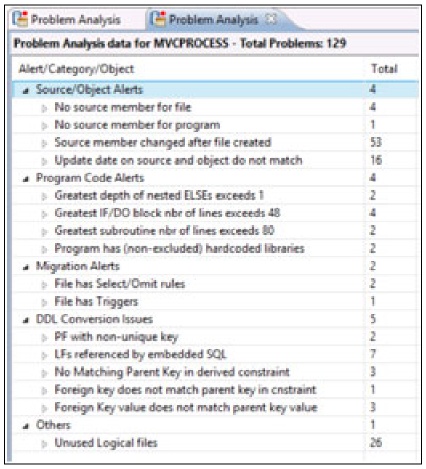
Display metrics are also an important factor to consider. Refacing projects are often underestimated in complexity, and therefore, budget and time. In database modernization projects, complex screens might require much more testing resources than are allocated and so bottlenecks occur. X-Analysis measures display complexity using these display metrics: number of display file formats, database files count, database fields, work fields, and function keys.

Review problem analysis per application area

One of the key knowledge bases for successful modernization projects is a detailed software problems list. Modernization projects can uncover all sorts of code deficiencies.

Problem analysis is integrated with X-Analysis, allowing project managers to examine each object and its specific issue to determine how to solve each problem.

Proactive problem analysis as part of modernization project planning helps avoid unnecessary project delays and costly overruns. See Figure 2-14.



Problem Analysis data for MVCPROCESS - Total Problems: 129

Alert/Category/Object	Total
Source/Object Alerts	4
No source member for file	4
No source member for program	1
Source member changed after file created	53
Update date on source and object do not match	16
Program Code Alerts	4
Greatest depth of nested ELSEs exceeds 1	2
Greatest IF/DO block nbr of lines exceeds 48	4
Greatest subroutine nbr of lines exceeds 80	2
Program has (non-excluded) hardcoded libraries	2
Migration Alerts	2
File has Select/Omit rules	2
File has Triggers	1
DDL Conversion Issues	5
PF with non-unique key	2
LFs referenced by embedded SQL	7
No Matching Parent Key in derived constraint	3
Foreign key does not match parent key in cnstraint	1
Foreign Key value does not match parent key value	3
Others	1
Unused Logical files	26

Figure 2-14 Database Problem Analysis

4. Map interface and integration issues

Because the data dependency relationships between project areas are recorded, a project manager knows which parts of a project depend on other parts. In this way, implementation and test plans can be more clearly defined for both data and interactive calls. For example, if you are migrating data to DDL, you must understand data dependencies by application area to allow for a phased or staged approach. The Application Area diagram in X-Analysis shows where application areas of your modernization project depend on other application areas (Figure 2-15).

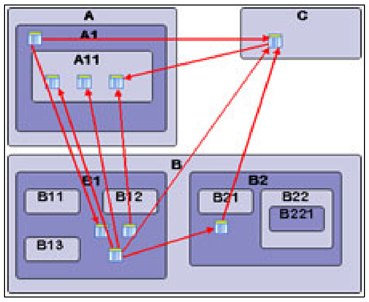


Figure 2-15 Application area diagram

5. Create a test plan

Testing is generally the most labor-intensive, underestimated, and poorly planned part of a modernization project. Fresche Legacy addresses these areas that are often overlooked during testing of modernization projects:

- Lack of properly defined test data. X-Analysis creates coherent cut down databases for testing
- Sensitive data is often required for testing. X-Analysis provides automation of intelligent data masking.
- No detailed regression testing. X-Analysis provides “before and after” test data comparison management and reports
- A lack of test cases that are driven by business rules coverage. X-Analysis provides key code data coverage along with display field-to-business rule mapping reports.

Test methodologies for modernization projects are not the same as the methodologies that are used for software development. Fresche Legacy can provide organizations with test methodology templates and processes and automated tools to ensure success and mitigate risk.

6. Define and scope modernization plan per application area:=

With all the analysis data reviewed and available to use in calculating timelines and building project schedules, the task remains to build a project plan. The fundamental difference with Fresche Legacy's approach is that plans are based on application forensics, and objective and documented metrics. Each company's modernization journey is unique and organizations can reach out to benefit from the numerous years of experience Fresche Legacy has in planning and running modernization projects.

2.3.4 Automated creation of the logical data model

The logical data model of an IBM i application is extracted automatically by X-Analysis. X-Analysis has evolved over 25 years to search and map the patterns of logic inside RPG, COBOL, and CA 2E (Synon) systems that implement business rules and manage the database relational constraints.

This automation can shave man-years off manual efforts to derive the same information, and the logical model can also be kept in sync with ongoing changes to the system.

The relational model of an enterprise application is a powerful piece of information and potentially valuable asset to the organization. The term ‘model’ refers to the foreign key or relational model, not just the physical model of the database. After it is explicitly defined, the relational model or architecture of the database can be reused in a number of scenarios:

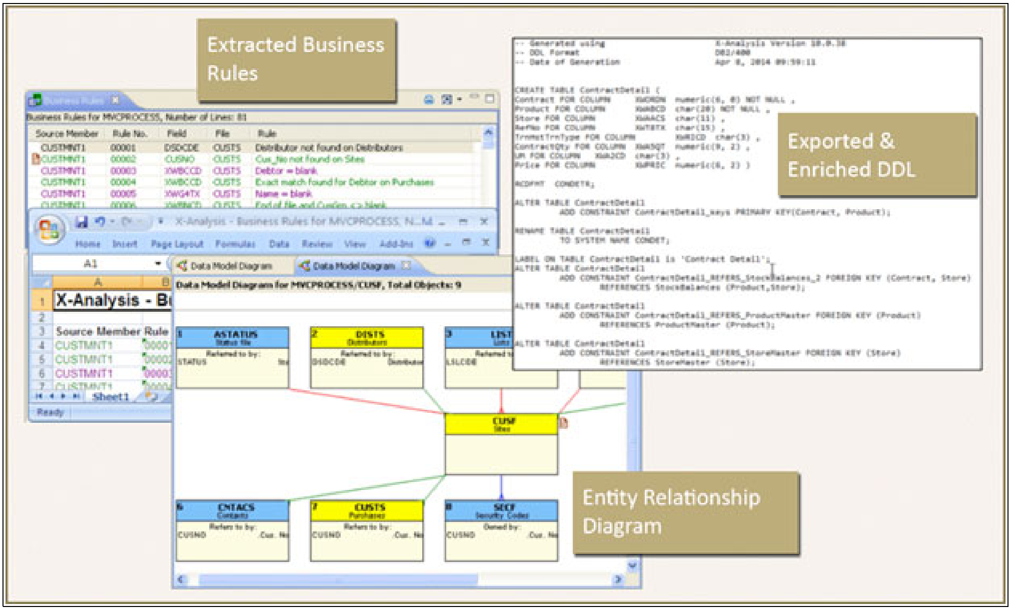
- Application architecture analysis
- Data quality referential integrity testing
- Test data extraction
- Sensitive test data masking and aging
- BI applications and data warehouse creation
- Data mapping for ERP upgrades
- Object relational maps creation
- Java access to existing databases

Deriving business rules and database constraints

X-Analysis analyzes patterns of code inside existing RPG, COBOL, or SYNON applications. The pre-built cross-reference repository of X-Analysis that is used by the pattern searching algorithms helps trace a field as it is used in a business rule or as part of data model constraint logic. After all the business rules, constraints and key-maps are derived from the application code, a data dictionary is stored in the X Analysis DB2 repository on the IBM i. This dictionary can then be displayed as entity relationship diagrams, foreign key maps, access path diagrams, or exported into Visio, Word, PDF, Excel, DDL, or even UML and XML. X-Analysis builds long names for the fields/columns by reusing the field text of the files concatenated with underscores. This process is used during the DDL generation phase of the automated modernization, and for creating SQL IO modules and ORM maps in Hibernate.

The X-Analysis repository can also be used to build custom analysis and modernization automation tools. Details of this process can be found in chapter 6 of *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

You can consolidate the derived logical model and business rules by physical file or table, field, program, display, or even by application area. In X-Analysis, the business rules that are related to the logical data model are indexed and color-coded to differentiate them from exportable or non-exportable rules. See Figure 2-16.


Extracted Business Rules
Exported & Enriched DDL
Entity Relationship Diagram
Business Rules for MVCPROCESS, Number of Lines: 81
X-Analysis - Business Rules for MVCPROCESS, N...M
Home Insert Page Layout Formulas Data Review View Add-Ins
Data Model Diagram
Data Model Diagram for MVCPROCESS/CUSF, Total Objects: 9
X-Analysis - B
ASTATUS
DISTS
LIST
CUSF
CNTACS
CUSTS
SECF
Sheet1
Ready

Source Member	Rule No.	Field	File	Rule
CUSTMNT1	00001	DSDCDE	CUSTS	Distributor not found on Distributors
CUSTMNT1	00002	CUSNO	CUSTS	Cus_No not found on Sites
CUSTMNT1	00003	XWBCCD	CUSTS	Debtor = blank
CUSTMNT1	00004	XWBCCD	CUSTS	Exact match found for Debtor on Purchases
CUSTMNT1	00005	XWG4TX	CUSTS	Name = blank
CUSTMNT1	00006	XWBNCD	CUSTS	End of file and CusGrp <> blank

```
-- Generated using                    X-Analysis Version 10.0.38
-- DDL Format                         DB2/400
-- Date of Generation                 Apr 8, 2014 09:59:11

CREATE TABLE ContractDetail (
Contract FOR COLUMN      XWORDN  numeric(6, 0) NOT NULL ,
Product FOR COLUMN       XWABCD  char(20) NOT NULL ,
Store FOR COLUMN         XWAACS  char(11) ,
RefNo FOR COLUMN         XWT8TX  char(15) ,
TrnHstTrnType FOR COLUMN         XWRICD  char(3) ,
ContractQty FOR COLUMN   XWA5QT  numeric(9, 2) ,
UM FOR COLUMN   XWA2CD  char(3) ,
Price FOR COLUMN         XWPRIC  numeric(6, 2) )

RCDFMT  CONDETR;

ALTER TABLE ContractDetail
        ADD CONSTRAINT ContractDetail_keys PRIMARY KEY(Contract, Product);

RENAME TABLE ContractDetail
        TO SYSTEM NAME CONDET;

LABEL ON TABLE ContractDetail is 'Contract Detail';
ALTER TABLE ContractDetail
        ADD CONSTRAINT ContractDetail_REFERS_StockBalances_2 FOREIGN KEY (Contract, Store)
                REFERENCES StockBalances (Product,Store);

ALTER TABLE ContractDetail
        ADD CONSTRAINT ContractDetail_REFERS_ProductMaster FOREIGN KEY (Product)
                REFERENCES ProductMaster (Product);

ALTER TABLE ContractDetail
        ADD CONSTRAINT ContractDetail_REFERS_StoreMaster FOREIGN KEY (Store)
                REFERENCES StoreMaster (Store);
```

Figure 2-16 Views in the X-Analysis repository (see the descriptions that follow)

The *Extracted Business Rules* shows rules that are mapped to an application area that is being exported to Microsoft Excel. Rules that are color-coded green are related specifically to the Logical Data Model

The *Entity Relationship Diagram* shows an interactive data Model Diagram centered on the Physical File CUSF. The colored lines denote the relationship, and the blue headers show PFs that share foreign keys with CUSF, but are outside the application area or scope of modernization.

Exported and Enriched DDL shows DDL that is exported directly from X-Analysis. Note the long field names and table constraints created by X-Analysis during the extraction phase.

Automated creation of SQL DDL tables and views

The DDS → DDL tool in X-Analysis converts DDS-defined physical and logical files into SQL DDL tables, views, and indexes. As part of the DDL generation, the tool uses additional information from the database objects to automatically provide:

- Long column names

  Values that are specified on either the COLHDG or TEXT keywords are used to generate a valid long column name in the DDL.

  Original DDS:

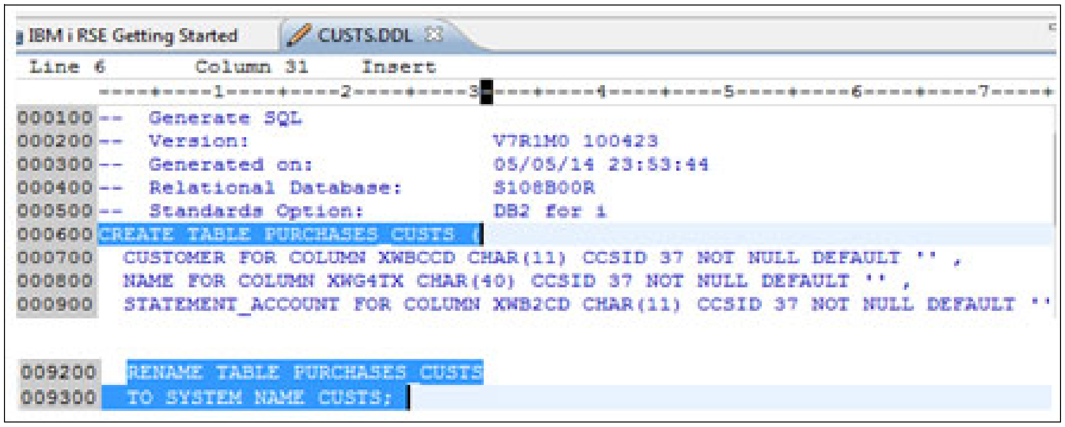
  ```
  A     XWBCCD11ATEXT('Customer')
  A               COLHDG('Customer' 'Number')
  ```

  Generated DDL:

  ```
  CUSTOMER FOR COLUMN XWBCCD CHAR(11) CCSID 37 NOT NULL DEFAULT '' ,
  ```

- Long table names

  Any text description that is associated with the database object is used to generate a valid long table name (while still preserving the original short name). See Figure 2-17.



```
Line 6          Column 31       Insert
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
--  Generate SQL
--  Version:                    V7R1M0 100423
--  Generated on:               05/05/14 23:53:44
--  Relational Database:        S108B00R
--  Standards Option:           DB2 for i
CREATE TABLE PURCHASES_CUSTS (
  CUSTOMER FOR COLUMN XWBCCD CHAR(11) CCSID 37 NOT NULL DEFAULT '' ,
  NAME FOR COLUMN XWG4TX CHAR(40) CCSID 37 NOT NULL DEFAULT '' ,
  STATEMENT_ACCOUNT FOR COLUMN XWB2CD CHAR(11) CCSID 37 NOT NULL DEFAULT ''

   RENAME TABLE PURCHASES_CUSTS
   TO SYSTEM NAME CUSTS;
```

Figure 2-17 Long table names

The DDS → DDL tool is designed to produce DDL-defined database objects, which maintain the same format level identifier as their original DDS-defined counterparts. This process ensures that existing applications operate normally without the need for program recompilation.

The pros and cons of foreign key constraints

Using the information within the X-Analysis data model, the DDS → DDL tool also establishes a list of candidate constraints for the new DDL-defined database. These constraints reflect primary → foreign key relationships which, although not defined at the database level, are enforced by the user's application code. Deploying these new constraints provides these advantages:

- Migration of the logic that enforces the constraint from the software application and into the database itself
- Less application code is required because the database itself now manages the constraint
- Greater portability of the database to another platform, with the constraints included

Deploying the new constraints has these downsides:

- If the application allows for the creation of child records before parents are created, the programs might fail if foreign key constraints are enabled between those two tables. Fresche Legacy provides a tool-assisted service to identify and resolve all such instances.
- Effort that is involved in identifying all instances in the application code where the constraint is enforced. This effort can be dramatically reduced with the X-Analysis constraint rules analysis feature. This feature identifies all instances in the code where referential integrity rules are implemented in the application code.
- Removal from the application code of all the code sections that enforce the constraints. Fresche Legacy provides a tool-assisted service to remove this code from existing programs, building automation tools from the instances that are discovered and indexed during the constraints analysis described previously.

2.3.5 Creation of view service programs

X-Analysis also allows you to automatically create, retrieve, update, and delete service programs for each table within a database. These service programs contain exported procedures for each I/O routine (create, read, update, delete, read rows).

The code that is required to implement each service program is complicated to develop manually and takes a significant amount of time. The automatic generation of the service programs alleviates these issues and delivers consistent results. Each procedure is implemented by using embedded SQL in ILE RPG, an example of which is shown in Example 2-1.

Example 2-1 Embedded SQL in ILE RPG

```
(select xwe4nb, xwdldt, xwc8dt, row_number() over (order by xwbccd)
as cte_rrn from trnhst)
select min(cte_rrn)
from trnhst ae join cte_row_number using (xwe4nb, xwdldt, xwc8dt)
where xwbccd = ?
group by xwbccd
order by xwbccd
```

Stateless pagination is implemented within the service program procedures, specifically on requests to return the first, next, and previous set of rows. Optimistic locking is also performed on update and delete requests.

The exported create, retrieve, update, and delete service programs are automatically deployed as web services within the Integrated Web Services Server on the IBM i, and are thus available to clients on multiple platforms and languages.

2.3.6 Data access layer modernization

This section describes the data access layer modernization.

Reuse business rules in data access layer

X-Analysis automatically extracts, cross-references, and stores business rules that are implemented in your applications. Business rules are consolidated back to files and fields, and matched with similar or exact matches across the application. X-Analysis can export these consolidated rules as XML for reuse with rules engines or other CASE tools. Fresche Legacy provides extra solutions to generate executable business rule entity programs or classes for use with IBM Business Process Management (BPM) or service-oriented architecture (SOA) modernization projects. This approach includes these key benefits. among others:

- Facilitates analysis and reuse of proven business logic
- Makes business logic more accessible to users and business analysts
- Lowers the cost of developing and modifying business logic
- Externalizes rules so they can be shared by multiple applications
- Improves architecture by further separating UI, database, and business logic layers

Automated Java I/O and hibernate generation

One of the biggest challenges that faces Java developers who build add-ons to existing IBM i applications is the mnemonic field names such as FLDRCT or MARTCS. Another issue is how files relate to each other to do validation, or bring in related field data such as text descriptors for use in GUI drop downs.

X-Analysis uses the derived logical and mapped physical model in its repository to generate a complete ORM as a Hibernate I/O abstraction layer. It includes the constraints and long field names that are derived by X-Analysis, and so provides a more meaningful and controlled access to the DB2 for i database for Java developers.

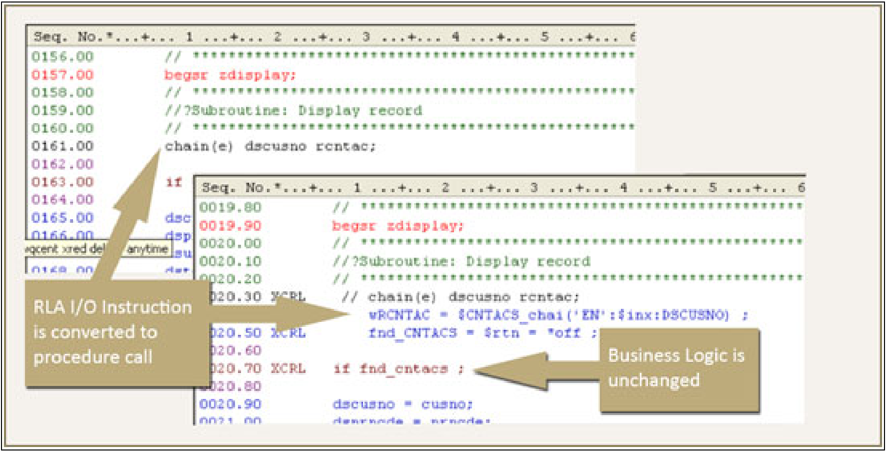
2.3.7 Decoupling of data access from high-level languages

Decoupling the database of a complex heritage RPG application from the programs is a labor intensive and difficult task. The main reason for this complexity is the distributed RLA statements such as READ, UPDATE, and SETLL proliferated throughout hundreds or thousands of programs. Fresche Legacy tools automatically externalize RPG RLA I/O in RPG programs into external CRUD-type procedures. The process simultaneously converts all RLA statements into CALLPs in the existing programs. Major database changes can then be made without the need to change heritage code or recompile and test vast tracks of the system. I/O can also be optimized and used with web services and new modern UIs, again with no impact on heritage code.

Fresche Legacy's tool-assisted service includes the following key features:

- All RLA is automatically analyzed and changed
- Fully automated and produces all requisite objects and sources
- Binds modules into new program objects
- Multiple template modes, which can all be easily tailored to any specific needs
- “Freeze” mode allows heritage code to continue “as is”, regardless of subsequent database changes
- Can be run as big-bang or incrementally
- Integrates with X-Analysis and SCMs

See Figure 2-18.



```
Seq. No.*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6
0156.00        // ***********************************************
0157.00        begsr zdisplay;
0158.00        // ***********************************************
0159.00        //?Subroutine: Display record
0160.00        // ***********************************************
0161.00        chain(e) dscusno rcntac;
0162.00
0163.00        if
0164.00
0165.00        dsc
0166.00        dsp
```

```
Seq. No.*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6
0019.80        // ***********************************************
0019.90        begsr zdisplay;
0020.00        // ***********************************************
0020.10        //?Subroutine: Display record
0020.20        // ***********************************************
0020.30 XCRL    // chain(e) dscusno rcntac;
                  wRCNTAC = $CNTACS_chai('EN':$inx:DSCUSNO) ;
0020.50 XCRL      fnd_CNTACS = $rtn = *off ;
0020.60
0020.70 XCRL   if fnd_cntacs ;
0020.80
0020.90        dscusno = cusno;
```

RLA I/O Instruction is converted to procedure call
Business Logic is unchanged

Figure 2-18 Automated Changes Made for RLA Decoupling

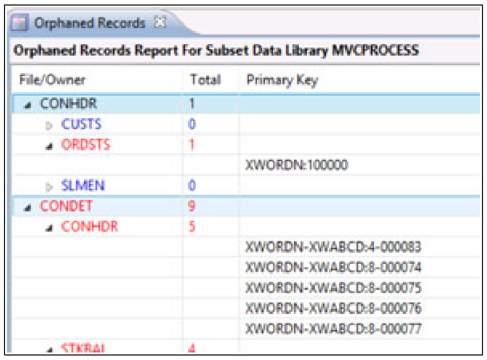
2.3.8 Activation of referential integrity

This section describes activation of referential integrity.

Automated data integrity validations

X-Analysis uses the derived logical data model to establish where orphaned records exist in the application database, so data quality can be verified before activating referential integrity in the database. The product examines each data record in turn to see whether its referential integrity relationships are satisfied.

Each integrity relationship that is breached is separately reported in a comprehensive audit report produced by the product (Figure 2-19).



Orphaned Records

Orphaned Records Report For Subset Data Library MVCPROCESS

File/Owner	Total	Primary Key
CONHDR	1	
CUSTS	0	
ORDSTS	1	
		XWORDN:100000
SLMEN	0	
CONDET	9	
CONHDR	5	
		XWORDN-XWABCD:4-000083
		XWORDN-XWABCD:8-000074
		XWORDN-XWABCD:8-000075
		XWORDN-XWABCD:8-000076
		XWORDN-XWABCD:8-000077
STKBAL	4	

Figure 2-19 Integrity relationship report

2.3.9 Supported platforms and software

This section lists the supported platforms and software.

- Operating System: X-Analysis Suite connects to IBM i running OS/400 V5R4 (and later)
- Languages: RPG III and later; RPG Free; COBOL; Java; PHP; PowerBuilder; C#; VB6; VB.NET; ASP
- Software: Rational Developer Suite 9.5
- Packages:
  - JDE 7.1, 8.3, 9.1, 9.2. 9.3
  - Fiserv
  - Infor LX, Infor XA

2.3.10 More information about X-Analysis

Contact Fresche Legacy at `info@freschelegacy.com` for more information about any X-Analysis module:

- X-Analysis Professional. Automated application documentation and impact analysis for RPG, COBOL, and CA 2E (Synon).
- Application Process Mapping. Business rule and relational data model extraction and documentation of application processes and flows.
- Audit, Quality and Change Management. Auditing of core application functionality, including design, quality, and complexity, to identify and change problematic areas in the application.

- Data and Test Management. Analysis of data quality; data archiving, data subsetting, and data masking. Test data automation and management.
- CA 2E Analysis. Everything that is required to analyze and document CA 2E (Synon) applications.
- Open Systems Analysis. Cross-referencing and documentation of Java, C#, PHP, VB/VB.NET, and PowerBuilder.
- Database Modernization. Automated conversion of DDS to DDL, including creation of constraints, long field names, and views.
- Application Modernization. RPG, COBOL, and CA 2E (Synon) automatically converted to Java.

2.3.11 Related information

- Modernizing and Improving the Maintainability of RPG Applications Using X-Analysis Version 5.6

  http://www.redbooks.ibm.com/redpieces/abstracts/redp4046.html
- Modernize IBM i Applications from the Database up to the User Interface and Everything in Between

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248185.html

More technical resources for X-Analysis are available at:

http://www.databorough.com/resources.html

2.3.12 Fresche Legacy contact information

info@freschelegacy.com

1-800-361-6782

2.4 Xcase for i

The database is the foundation of IBM i applications, and so is the first consideration for modernization. As most databases still rely on outdated database techniques, the ROI that can be achieved by modernizing the database is considerable.

The simultaneous modernization of the database and the applications is a complex and risky process. To facilitate this, IBM developed a method that allows re-engineering the database from DDS to SQL without having to modify or even recompile the existing applications. When the database is modernized, the applications can be modified with the benefit of the modernized database.

Based on the IBM method, Xcase for i is a set of tools that completely automates this complex task. In addition, Xcase for i automatically discovers and implements implicit relationships that are hidden in the database in a non-obstructive and flexible way. The modernized database can then be managed in a state-of-the-art graphical database modeling environment.

This section describes the Xcase for i suite of tools developed by Resolution. It allows you to modernize your database thoroughly and efficiently.

2.4.1 Contents

Most DB2 for i databases are defined by using the outdated DDS implementation of DB2 for i instead of the world standard SQL language. In addition, these databases do not contain declared relationships. As a result, they are not documented and are difficult to understand. They might also have serious data integrity issues. The Xcase for i suite of modernization tools overcomes these shortcomings by re-engineering heritage databases into modern and well-documented SQL defined databases. This process preserves the integrity of their data and allows you to manage them in a state-of-the-art graphical database modeling environment.

2.4.2 Considerations for older DB2 for i databases

DB2 for i is one of the most powerful, feature-rich, high performance database engines available. Nonetheless, most database implementations use only a small part of its tremendous power and functionality. Lacking openness to modern tools and new developers, they are perceived as "proprietary."

Most DB2 for i databases are not modern

Here are only a few of the reasons why most DB2 for i databases are not modern:

- DDS defined instead of SQL/DDL defined

  SQL, the de facto world standard database language, is the strategic choice of IBM for DB2 for i. Although there have not been any new DDS developments over the past 10 years, SQL has been continuously and significantly enhanced.

- Cryptic names

  A typical DB2 for i database has tens of thousands of fields with names limited to 10 characters that often contain prefixes or suffixes, which make them difficult to understand.

- Missing identities and auditing columns

  Identities that uniquely identify a record on a table are not supported by DDS, and auditing columns that indicate by whom and when a record was inserted or modified in a table are often missing.

- No declared relationships

  Relationships only exist implicitly and are not documented in most DB2 for i databases. This lack of documentation makes the database difficult to understand and bars the use of modern tools, which rely on declared relationships.

- Integrity issues

  Most DB2 for i databases have severe data integrity issues because relationships are not declared, and the database engine therefore cannot enforce integrity.

- Logical files instead of views

  Most DB2 for i databases still rely on logical files to view the data instead of the much more powerful and flexible SQL Views.

- Heritage dates instead of true dates

  Although the Date type has been available in DDS for more than 15 years, date information is still often stored in inadequate heritage formats. These formats allow invalid dates to be stored in the database and considerably slow down reporting on date-related information.

- Not managed by modern database modeling tools

  Database modeling tools are widely used in the database world. Still, most DB2 for i databases are managed by tools that do not qualify as modeling tools because they directly modify the database instead of a model.

Database modernization does not require modifying existing applications

A methodology developed by IBM and implemented by Modernize DB allows re-engineering to SQL without requiring the modification of the existing applications. This methodology also allows adding identities, auditing columns, and long, meaningful names to the database objects.

In addition, Relate DB allows the automatic identification of the implicit relationships that are hidden in a database, thus providing a documented database. Relate DB identifies integrity issues in the data and traces the applications that are causing them. Flexible implementation methods avoid the reoccurrence of these errors by allowing the database engine to automatically prevent them, regardless of the application that is used to update the database. The discovered relationships are also used to efficiently transform the heritage dates into true dates and to automatically generate complex join clauses, which greatly facilitate the creation of SQL Views. The modernized database can then be managed in a state-of-the-art graphical database modeling environment by using Evolve DB.

2.4.3 Solution overview

The Xcase for i suite is composed of four independent modules:

- Modernize DB. Allows re-engineering a database from DDS to SQL while preserving compatibility with existing applications.
- Relate DB. Allows automatically discovering and flexibly implementing the implicit relationships that exist in the database. Relate DB can work with a DDS or SQL defined database.
- Evolve DB. Allows creating and maintaining SQL databases by using a mature and sophisticated graphical modeling tool that automatically generates all the necessary DDL and SQL code.
- Viewer DB. Allows sharing dynamic database documentation with the development team and with power users.

2.4.4 Business value

The Xcase for i suite provides the following business value:

- SQL is the strategic choice of IBM for DB2 for i. For more than 10 years, there have not been new developments on DDS, which is considered to be “stabilized.” SQL defined databases, the de-facto world standard database definition language, provides access to continuous enhancements and new functions that are not available on DDS.
- Although it is increasingly difficult to find developers with DDS skills, there are many skilled SQL developers in the labor market.
- Moving to SQL opens the door to important performance gains.
- Modernizing the database provides access to modern technologies and tools that often require SQL databases with declared relationships.

- An undocumented database represents a high risk and higher costs to the organization. A well-documented database facilitates communication among developers and with users. It speeds up development while avoiding critical errors.
- A well-documented database is essential for the successful integration of new developers, analysts, and power users.
- When the database engine does not enforce integrity, the database is prone to errors and the result is unreliable, poor-quality data.
- The Xcase for i suite of tools allows the full modernization of the database in a safe and controlled manner at a fraction of the time and cost of a manual approach.

2.4.5 Solution architecture

This section covers the solution architecture, including Modernize DB and the discovery process.

Modernize DB

Modernize DB, which allows re-engineering a database from DDS to SQL, is based on a methodology developed by IBM that was published in *Modernizing IBM eServer iSeries Application Data Access - A Roadmap Cornerstone*, SG24-6393. The key point of this methodology is to re-engineer the database to SQL without needing to modify or recompile existing applications. This process is accomplished by using the “LF Surrogate” technique.

The LF Surrogate

For each physical file that is re-engineered into an SQL table, a logical file called an LF Surrogate is generated. Existing applications address the Surrogate LF that replaces the Physical File and continue to function as before, *except* that they write and read data from the new SQL table.

For this process to happen, the LF Surrogate must have the following characteristics:

- The same system name as the original PF
- The same columns as the original PF
- The same key words as the original PF
- The same access path as the original PF
- The same Format Level Identifier as the original PF
- It points to the NEW SQL table

In some cases, you can directly use the new SQL table without using the LF Surrogate. Although this method is also supported by Modernize DB, it is not the recommended approach. The use of an LF Surrogate does not add any constraints and does not have any negative impact on performance. However, not using an LF Surrogate often requires you to modify your applications, particularly if you add new columns to the SQL table, such as identity and auditing columns, or if the PF is keyed or has keywords that are not supported by SQL.

Figure 2-20 illustrates how the LF Surrogate is used to preserve compatibility with the existing applications.

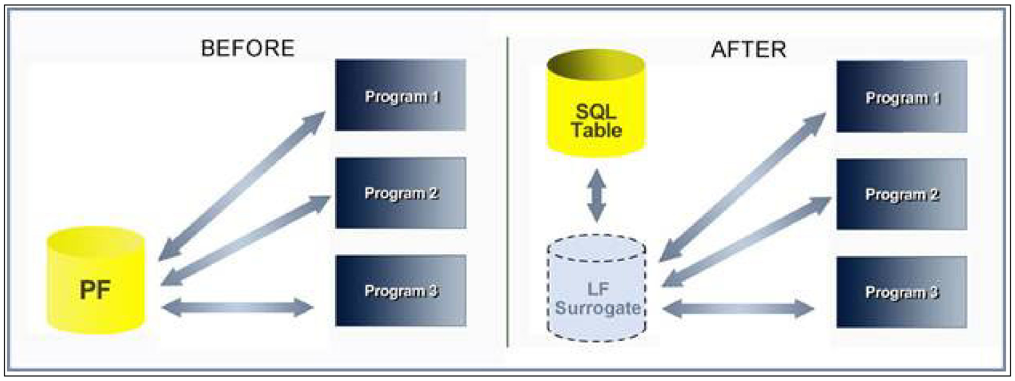


Figure 2-20 The LF surrogate

Modernization process overview

Modernize DB completely automates the modernization process, which consists of the following steps:

- Analysis

  The physical files that can be modernized are identified.

  In general, physical files that are work files do not need to be modernized.

  Physical files that have multiple members cannot be modernized. However, this constraint applies only if the physical file holds multiple members and not if it was defined as a multi-member file, but only holds a single member.

- Setting names

  Long names for the objects in the modernized database are set. You can use meaningful long names for tables and columns. The names can be automatically set by using customizable templates and the resulting names can be individually overridden, if necessary.

- Setting extra columns

  The extra columns to be added to the new SQL tables are set. The possible columns are Identity and auditing columns (when and by whom a record was added or modified). These new columns are managed by the database engine and, so you do not have to modify or recompile your existing applications.

- Creating the SQL tables

  Although IBM i Navigator provides a utility that generates the SQL table “equivalent” of a DDS file, many more specifications must be taken into account to generate a correct SQL Table. These specifications include the long names of the columns, the ownership of the object, the table's authorizations, the extra columns, the indexes of the table that are derived from the keys in the PF and the associated LF, the journaling options, compilation parameters, and many more. Modernize DB generates the SQL table, taking into account all the preceding parameters, and stores the source code in a source file.

- Creating the LF surrogates

  Modernize DB automatically generates and compiles the LF surrogate and checks that the Format Level Identifier, Access Path, and Key Words precisely match those of the original PF. In the rare cases that this matching does not occur, Modernize DB identifies the programs that use the original PF and the cause of the discrepancy.

- Checking the dData in the original physical files

  In DDS defined files, the database engine checks that the data is the valid type when reading the data instead of when writing the data, as is done with SQL defined tables. This method provides a performance advantage to SQL because, on average, the database engine reads data 20 times more than it writes data. In addition, there might be incorrect data in a DDS defined table, such as a '$' character in a numeric field. Modernize DB checks the data and provides detailed information to fix the problem.

- Copying the data from the original physical files into the SQL Tables

  Upon ensuring that no invalid data exists in the original DDS tables, the data can be copied into the new SQL tables. Modernize DB optimizes this process to minimize the database downtime. The modernization process can be conducted gradually because it is not necessary to modernize all the tables at once. The database engine provides full support for hybrid systems that are composed of DDS and SQL defined tables.

- Transforming the existing logical files

  The existing logical files need to be preserved in the new database so that the applications continue working as before, but they must undergo two transformations:

  a. They need to point to the new SQL tables.

  b. The key words that are implicitly inherited from the underlying PFs must now be explicit because the transformed LF points to SQL tables, which do not support keywords.

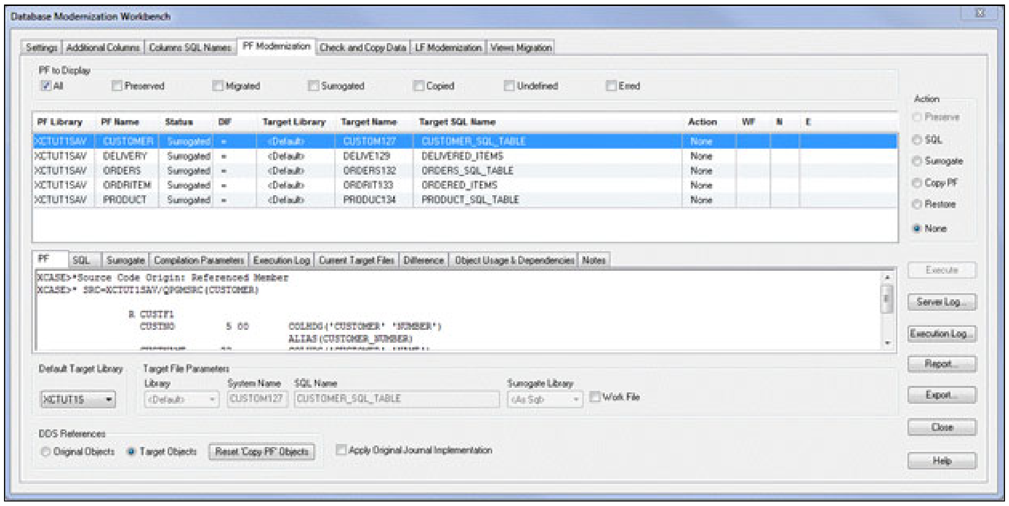
  As for the LF Surrogate, in the rare cases in which incompatibilities between the original and transformed LF are found, Modernize DB provides a detailed report of the differences and the usage of the LF in the applications.

  Modernize DB also optionally generates an SQL View that corresponds to the transformed LF (but does not replace it).

- Transforming the Existing SQL Views and Materialized Query Tables

  If the database contains SQL Views or Materialized Query Tables based on DDS defined tables, which are modernized to SQL, they must be transformed so that they point to the new SQL defined tables.

Figure 2-21 shows the workbench.



PF to Display: [✓ All] Preserved Migrated Surrogated Copied Undefined Erred

PF Library	PF Name	Status	Dif	Target Library	Target Name	Target SQL Name	Action	WF	N	E
XCTUT15AV	CUSTOMER	Surrogated	=	<Default>	CUSTOM127	CUSTOMER_SQL_TABLE	None			
XCTUT15AV	DELIVERY	Surrogated	=	<Default>	DELIVE129	DELIVERED_ITEMS	None			
XCTUT15AV	ORDERS	Surrogated	=	<Default>	ORDERS132	ORDERS_SQL_TABLE	None			
XCTUT15AV	ORDRITEM	Surrogated	=	<Default>	ORDRIT133	ORDERED_ITEMS	None			
XCTUT15AV	PRODUCT	Surrogated	=	<Default>	PRODUC134	PRODUCT_SQL_TABLE	None			

Action: Preserve SQL Surrogate Copy PF Restore [✓ None]

```
XCASE>*Source Code Origin: Referenced Member
XCASE>* SRC=XCTUT15AV/QPGMSRC(CUSTOMER)

             R CUSTF1
               CUSTNO         5 0O       COLHDG('CUSTOMER' 'NUMBER')
                                         ALIAS(CUSTOMER_NUMBER)
```

Default Target Library: XCTUT15

Target File Parameters: Library: <Default> System Name: CUSTOM127 SQL Name: CUSTOMER_SQL_TABLE Surrogate Library: <As Sql> Work File

DDS References: Original Objects [✓ Target Objects] Apply Original Journal Implementation

Figure 2-21 The Modernize DB database modernization workbench

Relate DB

Most DB2 for i databases do not have declared relationships. This lack of declaration does not mean that relationships do not exist, but that they exist only implicitly. As a result, the databases are not documented and are difficult to understand. This lack of documentation puts one of the most important assets of the company, the database, at high risk.

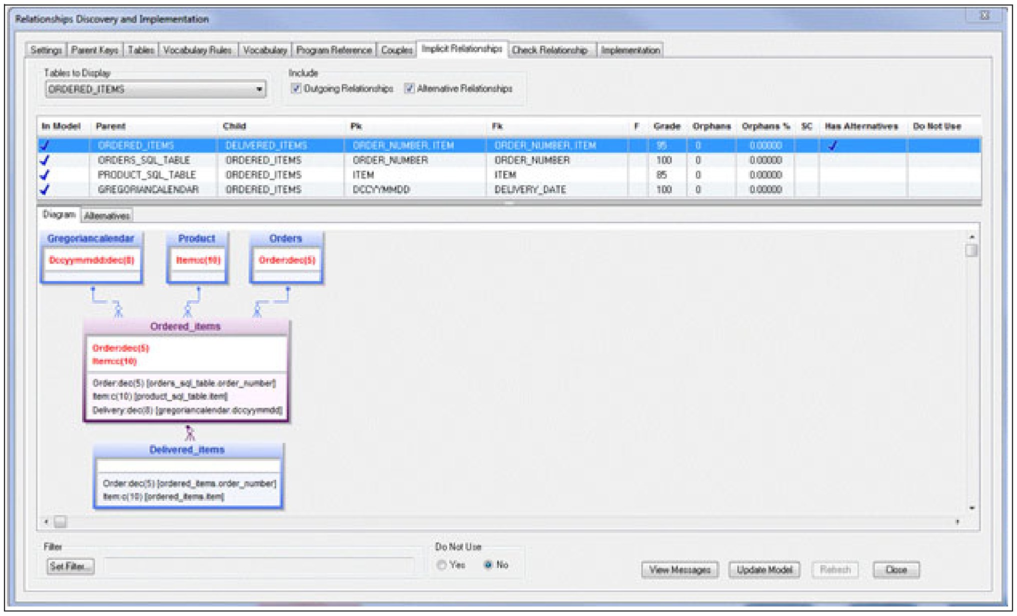
Almost all databases that rely on programs to ensure referential integrity instead of the database engine can have severe data integrity issues. The lack of declared relationships also restricts the use of modern database tools, such as reporting or test data generators, which require declared relationships.

This situation might be unique in the database world. Most DB2 for i workshops do not address the problem of undocumented databases because devising a solution represents a huge task that requires considerable investment in time, in addition to comprehensive database knowledge.

Relate DB is an expert system that not only automatically discovers the implicit relationships that are hidden in the database, but also provides flexible implementation methods that allow declaring relationships in the database without needing to modify the existing applications.

Discovery process

In a typical DB2 for i database of a thousand tables, billions of possible relationships exist. Relate DB is a Data Driven Expert System that uses all available metadata (field specifications, keys, joins, REFFLD, program cross-references …) semantic data (names, prefixes, suffixes, text, headings …) and physical data (existence of duplicates, existence of orphans) to automatically discover the implicit relationships. The algorithm for this process, which has been refined over the years, is able to suggest the correct relationships so that all you need to do is to validate the propositions that are made by Relate DB. The propositions are fully explained and can be viewed in entity relationship diagrams. In addition, the discovery parameters are configurable and Relate DB supports an incremental discovery approach. Relate DB reduces the discovery time from infinite (because it is never done manually) to a matter of days. See Figure 2-22.


Relationships Discovery and Implementation
Settings | Parent Keys | Tables | Vocabulary Rules | Vocabulary | Program Reference | Couples | Implicit Relationships | Check Relationship | Implementation
Diagram | Alternatives
Set Filter...
View Messages
Update Model
Refresh
Close

Tables to Display: ORDERED_ITEMS

Include: [✓ Outgoing Relationships] [✓ Alternative Relationships]

In Model	Parent	Child	Pk	Fk	F	Grade	Orphans	Orphans %	SC	Has Alternatives	Do Not Use
✓	ORDERED_ITEMS	DELIVERED_ITEMS	ORDER_NUMBER, ITEM	ORDER_NUMBER, ITEM		95	0	0.00000		✓	
✓	ORDERS_SQL_TABLE	ORDERED_ITEMS	ORDER_NUMBER	ORDER_NUMBER		100	0	0.00000			
✓	PRODUCT_SQL_TABLE	ORDERED_ITEMS	ITEM	ITEM		85	0	0.00000			
✓	GREGORIANCALENDAR	ORDERED_ITEMS	DCCYYMMDD	DELIVERY_DATE		100	0	0.00000			

Gregoriancalendar
Dccyymmdd:dec(8)

Product
Item:c(10)

Orders
Order:dec(5)

Ordered_items
Order:dec(5)
Item:c(10)
Order:dec(5) [orders_sql_table.order_number]
Item:c(10) [product_sql_table.item]
Delivery:dec(8) [gregoriancalendar.dccyymmdd]

Delivered_items
Order:dec(5) [ordered_items.order_number]
Item:c(10) [ordered_items.item]

Filter

Do Not Use: Yes [✓ No]

Figure 2-22 The Relate DB relationship discovery and implementation workbench

Relationships from the calendar table

Although the Date type has been available on DDS for more than 15 years, date information is still often stored in inadequate heritage formats. These formats allow invalid dates to be stored in the database and that considerably slow down reporting on date-related information.

To solve this problem, Relate DB creates a calendar table with a record for each day for 100 years, both before and after the current date. Each record is uniquely identified by a date field that also contains all the possible variations of heritage dates, in addition to date-related information such as the day of the week. You can even customize it and add new columns that identify, for example, the fiscal year or a work day for your company. Relate DB discovers relationships between the heritage dates in your database and the dates that are stored in the calendar table. The relationships allow retrieving a true date and date-related information much more efficiently than calculated expressions, in addition to detecting invalid dates in your database.

Discovery benefits

When the relationships are discovered and validated, and the database is documented, you can produce entity relationship diagrams by using the sophisticated and comprehensive graphical features of Xcase. The benefits of providing a deep understanding of the database structure to the development team and the power users include promoting constructive and productive conversations about the current and future design of the database, and accelerating new developments. Moreover, by using the relationships to automatically and optimally create the complex joins between distant tables, Xcase allows you to easily create SQL Views, and to generate compact but coherent and comprehensive test data, obfuscating all confidential information.

Implementation benefits

By implementing the discovered relationships in the database, further benefits can be obtained. For example, when relationships are declared in the database engine, they can be used by external tools, such as reporting tools, or used by the database engine itself to speed up access to related information.

Another obvious advantage of implementing relationships in the database is to allow the database engine to manage Referential Integrity. This approach qualifies as a preferred practice because of the following benefits:

- Referential integrity is safely and optimally implemented by the database engine.
- Development is faster because the referential integrity code does not have to be repeated in each application because it is now handled by the database engine itself.
- It is possible to reduce the application code and the associated maintenance time by removing the referential Integrity code that is no longer required.
- The integrity of the database is preserved, regardless of the application that modifies it.

Implementation issues

When you are attempting to implement the discovered relationships in the database, a number of problems might arise:

- Referential Integrity is violated by the data in the database.

  When Referential Integrity has not been set and controlled by the database engine for many years, there are integrity errors, such as an order pointing to a nonexistent customer. Relate DB detects these problems and allows you to manually or programmatically fix the corrupted data by generating SQL statements to retrieve the offending records.

- Existing applications create integrity issues in the database.

  When the database contains data integrity errors, it is likely that some applications are creating those errors. Relate DB allows you to identify those applications by writing the trace of the application that caused the error in a message queue.

- Implicit relationships cannot be declared in SQL.

  It might not always be possible to formally declare an implicit relationship because the strict rules that must be enforced are not always respected by the Implicit Relationship such as:

  – The Parent Key and the Foreign Key fields must have the same type, length, and CCSID. These requirements are not always the case in an implicit relationship.

  – When the value of a Foreign Key is not known, SQL expects it to be NULL, not “Pseudo NULL” (0 or empty character), as it often is in heritage databases.

- Sometimes a program writes the Foreign Key in the Child Table and later creates a corresponding Parent Key value in the Parent Table. This temporary integrity violation is not tolerated by the database engine.

Relate DB also proposes a solution for these situations by defining an alternative relationship, which is functionally equivalent to the Implicit Relationship because it links the children to the same parent, but can be declared. It can also be better controlled than a regular relationship by allowing, for example, using a “Pseudo Null” instead of a Null value when the value of the foreign key is not known.

Implementation strategy

To handle the implementation issues that are previously described, Relate DB offers four different implementation methods that can be applied individually for each discovered relationship. In addition, you can set whether to trace integrity errors for a relationship. This method allows recording the trace of the application that caused a database integrity violation into a message queue without creating an error.

The four implementation methods are:

- None

  The relationship is not declared in the database, but can be used for documentation purposes in the Xcase Entity Relationship Diagrams to create SQL Views, and to coherently populate a Test Database by using Xcase.

- Disabled

  The relationship is declared in the database and is visible to external tools, but disabled so that the Database Engine does not take any action when its integrity is violated.

- Declared

  The relationship is declared in the database and the database enforces integrity according to the actions that are supported by the database engine that you set.

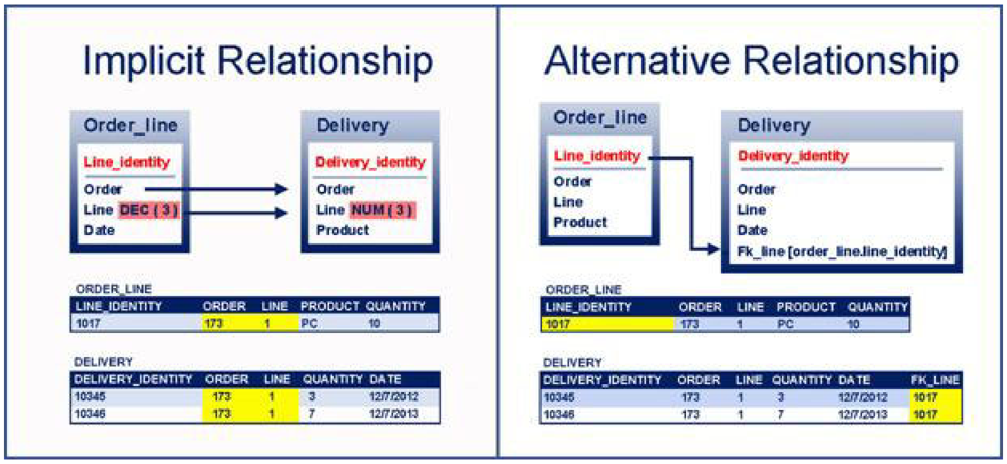
- Alternative relationship

  When a functionally equivalent relationship is declared in the database, a non-composite primary key must exist in the parent table, and a new foreign key field that points to the appropriate parent must be defined in the child table. The new foreign key field is automatically updated by triggers generated by Relate DB so your applications do not have to be modified. The advantages of the Alternative Relationship are that it can be declared even when the Implicit Relationship cannot, and it offers more flexibility than a “regular” relationship.

For example, you can set that no error occurs when the foreign key is a “Pseudo Null” (0 or empty character) even if no such value exists in the parent.

The triggers add some processor burden when you are writing in the database, but when reading from the database, the performance improves because the relationship relies on a single key. Because the database reads 20 times more often than it writes, overall performance improves.

Figure 2-23 shows an implicit relationship that cannot be declared because the types of the parent key and the foreign key are not identical. The alternative relationship on the right relates an identity of the parent table to a new foreign key field in the child table. The implicit and the alternative relationships are functionally equivalent because they link the child tables to the same parent. The new foreign key field is defined by Xcase and is maintained by Xcase generated triggers.



ORDER_LINE

LINE_IDENTITY	ORDER	LINE	PRODUCT	QUANTITY
1017	173	1	PC	10

DELIVERY

DELIVERY_IDENTITY	ORDER	LINE	QUANTITY	DATE
10345	173	1	3	12/7/2012
10346	173	1	7	12/7/2013

ORDER_LINE

LINE_IDENTITY	ORDER	LINE	PRODUCT	QUANTITY
1017	173	1	PC	10

DELIVERY

DELIVERY_IDENTITY	ORDER	LINE	QUANTITY	DATE	FK_LINE
10345	173	1	3	12/7/2012	1017
10346	173	1	7	12/7/2013	1017

Figure 2-23 An implicit relationship that cannot be declared

2.4.6 Evolve DB

When your database is modernized, it must be modified as your business needs change.

Evolve DB is a state-of-the-art database modeling solution that has been continuously enhanced over the last 20 years, and is used daily by more than 3,000 customers in 50 countries. Unlike other database modeling tools, it offers full support for all the attributes of DB2 for i objects.

Two-way synchronization process

Evolve DB allows you to reverse engineer an existing database into a database model or to create a database, add database objects and modify existing ones in a rich graphical environment. The two-way synchronization process allows you to compare what you have in the database with what you have in the model.

Evolve DB displays all the differences that are found at the attribute level. You can decide which modifications should be ported to the database or to the model itself. This decision allows you to run an impact analysis without directly modifying the database. Then Evolve DB generates all the SQL code that is required to synchronize the database with the model automatically. You do not have to write a single line of code.

The generated code takes into account not only your direct modifications to the model, but also their consequences. For example, modifying a primary key or a field name might cause the modification of foreign keys (perhaps in cascade), which might also require the modification of indexes, fields, views, and stored procedures. The resulting code is available as an SQL script and as source code that is stored in a source file.

Graphical features

Because databases can contain thousands of tables, it is important to be able to decompose them into manageable, multiple diagrams. Every aspect of the form and content of the diagram is configurable. Evolve DB even supports embedded diagrams, or diagrams within diagrams, for clear and easy representation of the relationships between subject areas. Sophisticated automatic layout of a diagram is also supported.

Customizable data dictionary

The browsers that allow you to inspect database object attributes at the database or the specific object level are completely customizable. You can also add you own user-defined attributes for documentation or code generation purposes.

Database code generation

The Xcase scripting language allows you to generate database-related code, such as XML schemas and stored procedures, and to export metadata into text files. The scripting language contains object-oriented methods that allow easy access to all the metadata in the model, including your own user-defined attributes.

Version control

Models can be archived, compared with new versions, and selectively modified as the result of the comparison. You can also merge two models and selectively update each one.

Navigating and editing physical data

Evolve DB uses the relationships in the model to allow you to relationally navigate the physical data (from parent to child, from child to grandchild, and so on) and to edit it with integrity (automatic look-ups on foreign keys and enumerated types).

Checking data integrity

When you want to set a new business rule in the database, such as a unique constraint, a relationship, or a field constraint, Evolve DB tells you if the existing data conforms to this rule. If it does not, it allows you to fix the faulty data by displaying it in an editable browser or by producing an SQL statement to retrieve it and programmatically fix it.

Powerful SQL view generator

Evolve DB allows you to create SQL Views by selecting its columns from related tables or other views and graphically selecting the path (set of relationships) that lead from one table to another. The automatically generated code includes all the necessary joins and is dynamically reevaluated when compared with the current view in the database.

Support for REFFLD

When a database is modernized to SQL, the REFFLD field keyword (field reference) is not supported. Evolve DB preserves this information in the model in the form of “Pseudo Domains” so that when a field is modified, it is (optionally) cascaded to all the fields that reference it.

Figure 2-24 displays the Xcase visual workspace.

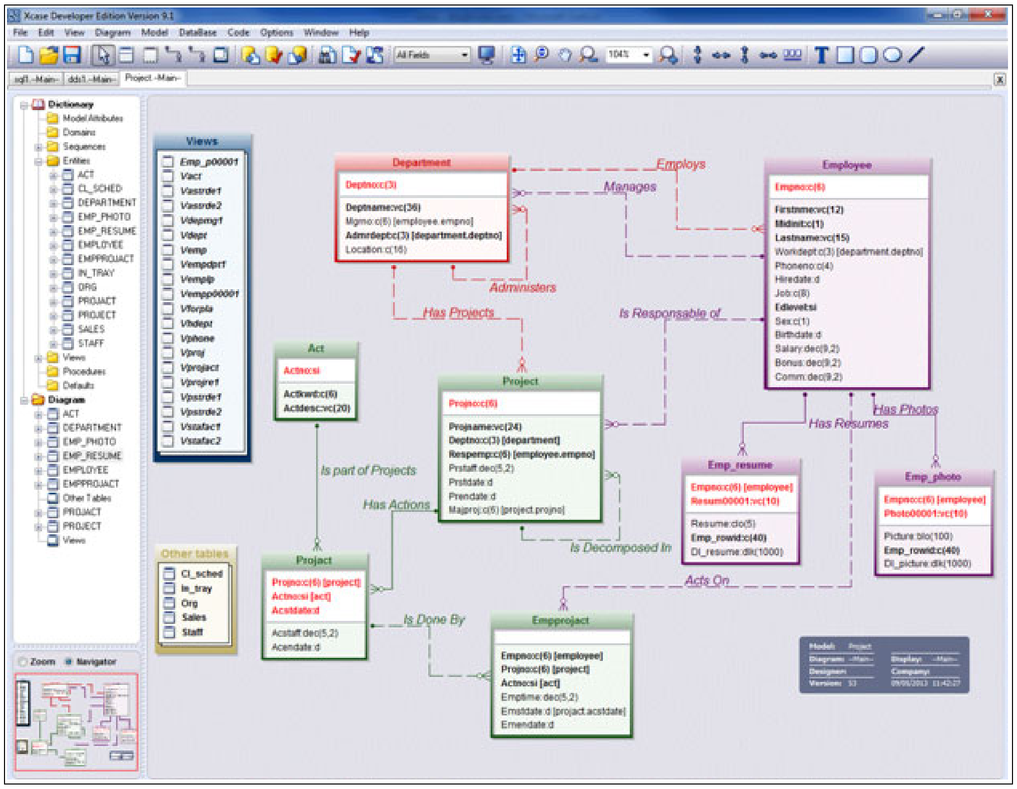


Figure 2-24 A picture is worth a thousand words

2.4.7 Supported platforms

Xcase for i has two installation components. One is on the server and the other is on the PC. The OS requirements are:

- On the server, V5.4 and later
- On the PC, any Windows platform

2.4.8 Ordering information

Contact `xcasefori@xcase.com`.

2.4.9 Related information

For more information, see:

`http://www.xcasefori.com`

3

Security

This chapter describes security solutions for modernizing your IBM i applications.It includes the following section:

- Using SkyView Policy Minder in application development

3.1 Using SkyView Policy Minder in application development

As security becomes more of an issue, and as compliance requirements are more strictly enforced, the security settings (ownership, *PUBLIC authority authorization list assignments, program adoption, and so on) of the applications that access that data become more important. SkyView Policy Minder is a security administration product for IBM i that aids in the configuration management of application objects and development, quality assurance (QA), and production environments.

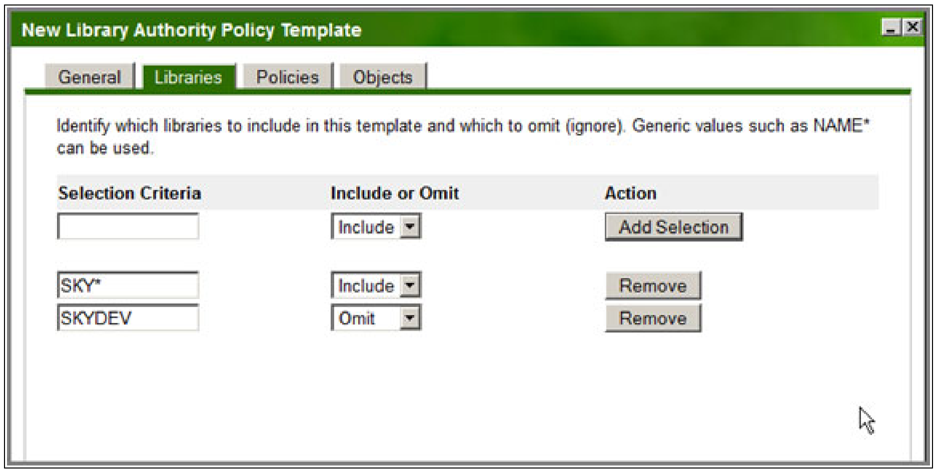
With Policy Minder you perform these tasks:

- Define the security-related settings for libraries (and the objects in libraries) and directories (subdirectories and objects in them)
- Set the ownership of application objects to different profiles as the objects are promoted through the development environments (development, QA, Prod, and so on)
- Use FixIt to set ownership and authorities to their defined values. This task can be done periodically or at specific times such as after a restore or as part of the application's build process. For example, SkyView Partners uses Policy Minder to set its Risk Assessor and Policy Minder application objects to be owned by SKYVIEWOWN, *PUBLIC authority gets set to *EXCLUDE, and all private authorities are removed. In addition, it has one program that must adopt authority so that is also configured.
- Integrate the Fix immediate commands into your change management process to provide more granular security configuration settings. Examples include securing files with a different authorization than the default autl or securing programs, data areas, and files that work with encryption differently than the rest of the application. For example, SkyView Partners has clients that use these FIX commands as part of their change management process. They have a different authorization list for each library. As objects are promoted through their process, their authorities are set by using these FIX commands. Provide the template where the object is defined and the rest is done for you (more information about templates is provided later in this section).
- Take a snap-shot of library and object settings such as ownership, creation authority, object auditing, *PUBLIC authority, private authorities, authorization list assignment, adopted authority settings, and so on. Then, if necessary, run a program that sets everything back to the snap-shot values. For example, some clients want to take a snapshot of their current configuration in case they must back out of an upgrade (typically required as part of their change control process).

Policy Minder provides both a traditional “green screen” and a browser-based GUI interface. Consider exactly how you can use Policy Minder to aid in application development.

First, look at how you can configure Policy Minder to work with your application objects in libraries. To set up your application in Policy Minder, create a library template. In this library template, specify the library or libraries that you want to work with. It is easy to work with multiple versions of your application by using wildcards. You can also omit libraries in case you want to omit certain versions such as Development or Test. When a check is run on the template that is being defined, it pulls in the SKYPROD, SKYQA, and SKYUAT libraries, but not the SKYDEV library. You can also use the positional wildcard '?' in the name. For example, SKY???LIB.

Figure 3-1 shows the tab where you specify the libraries.

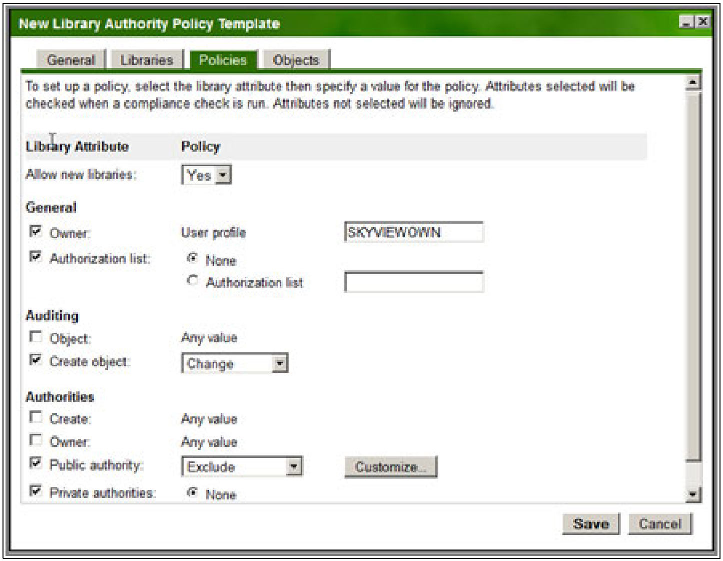

New Library Authority Policy Template

Identify which libraries to include in this template and which to omit (ignore). Generic values such as NAME* can be used.

Selection Criteria	Include or Omit	Action
	Include	Add Selection
SKY*	Include	Remove
SKYDEV	Omit	Remove

Figure 3-1 New Library Authority Policy Template Libraries tab

Next, you can define the security-relevant attributes of the libraries. Click the Policy tab. In the example shown in Figure 3-2, the library is owned by SKYVIEWOWN. It is not secured by an authorization list, create object auditing is set to *CHANGE, *PUBLIC authority is set to *EXCLUDE, and there are no private authorities. Not all attributes must be specified. You can choose which attributes make sense for your application's requirements.

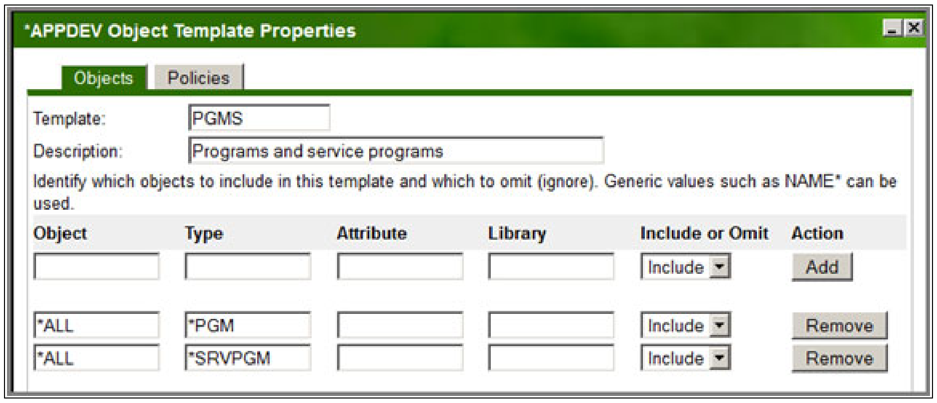

New Library Authority Policy Template

To set up a policy, select the library attribute then specify a value for the policy. Attributes selected will be checked when a compliance check is run. Attributes not selected will be ignored.

Library Attribute	Policy	
Allow new libraries:	Yes	
General		
☑ Owner:	User profile	SKYVIEWOWN
☑ Authorization list:	● None	
	○ Authorization list	
Auditing		
☐ Object:	Any value	
☑ Create object:	Change	
Authorities		
☐ Create:	Any value	
☐ Owner:	Any value	
☑ Public authority:	Exclude	Customize...
☑ Private authorities:	● None	

Figure 3-2 New Library Authority Policy Template Policies tab

Click the Objects tab to create one or more object templates. These templates define the requirements for the objects in the libraries (Figure 3-3).


*APPDEV Object Template Properties

Objects | Policies

Template: PGMS

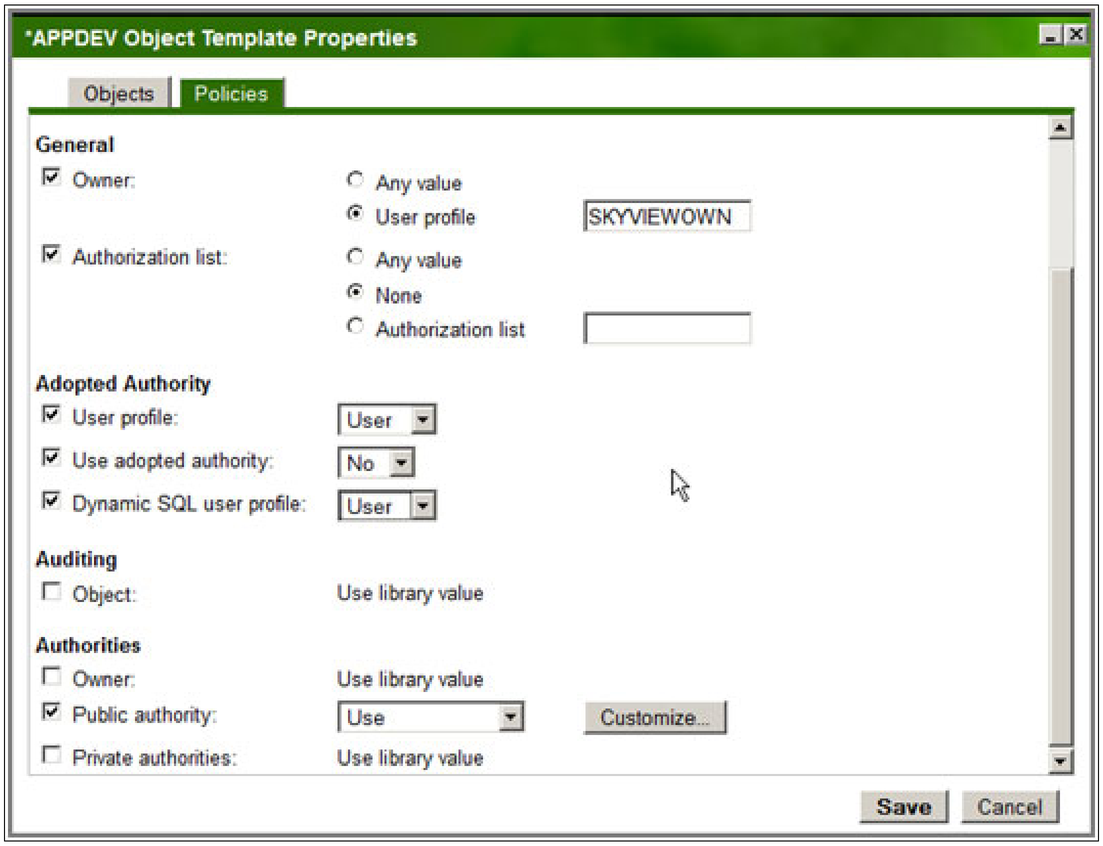
Description: Programs and service programs

Identify which objects to include in this template and which to omit (ignore). Generic values such as NAME* can be used.

Object	Type	Attribute	Library	Include or Omit	Action
				Include	Add
*ALL	*PGM			Include	Remove
*ALL	*SRVPGM			Include	Remove

Figure 3-3 Object Template Properties Objects tab

The example in Figure 3-4 shows how to check the security settings for programs and service programs. Make sure that they are owned by SKYVIEWOWN, not secured with an authorization list, do not adopt authority, and have *PUBLIC *USE.


*APPDEV Object Template Properties

Objects | Policies

General

- [✓] Owner: ( ) Any value (●) User profile SKYVIEWOWN
- [✓] Authorization list: ( ) Any value (●) None ( ) Authorization list

Adopted Authority

- [✓] User profile: User
- [✓] Use adopted authority: No
- [✓] Dynamic SQL user profile: User

Auditing

- [ ] Object: Use library value

Authorities

- [ ] Owner: Use library value
- [✓] Public authority: Use Customize...
- [ ] Private authorities: Use library value

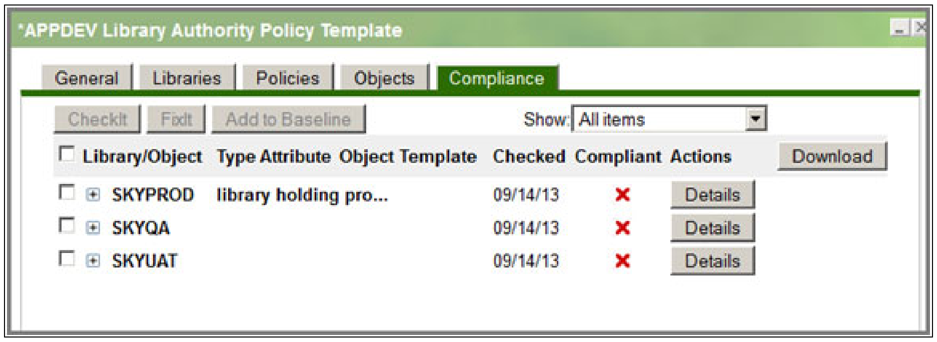
Save Cancel

Figure 3-4 Object Template Properties Policies tab

If you have specific programs that need more power and those programs must be owned by QSECOFR and adopt authority (for example, programs that do encryption operations), you can create an object template with those specifications.

You might also want to create other object templates. Some examples include an object template for Physical and Logical files and setting the attribute that requires them to be journaled. You might want to create a "catchall" template that includes all of the rest of the objects in the application libraries to set the ownership correctly. There is no limit to the number of object templates you create. This flexibility allows you to be as granular or specific as you want to be.

Now you are ready to run a check. The check identifies any setting that does not match the configuration that you have defined (Figure 3-5).



Show: All items

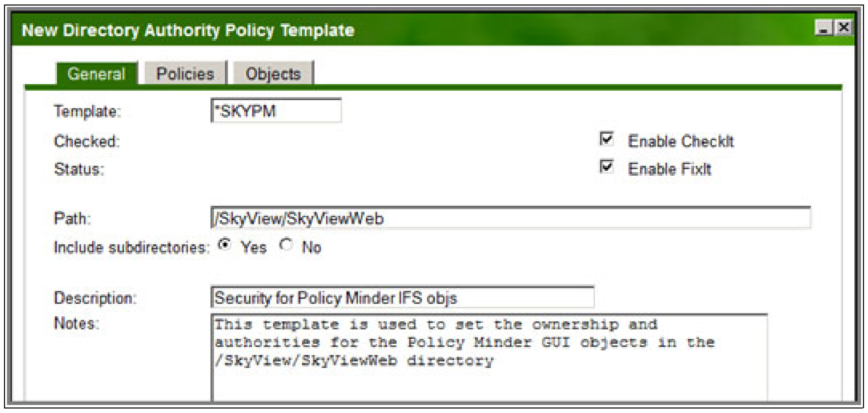
Library/Object	Type	Attribute	Object Template	Checked	Compliant	Actions
SKYPROD	library	holding pro...		09/14/13	✗	Details
SKYQA				09/14/13	✗	Details
SKYUAT				09/14/13	✗	Details

Figure 3-5 LIbrary Authority Policy Template Compliance tab

When you click **Details**, you can see the library attributes are not set correctly. Although you can change these attributes yourself, you can much more easily get the correct settings by running FixIt to change the values for you. Both the check and FixIt commands can be run either interactively or in batch, and they can be scheduled to run at specific times with a job scheduler.

After you create your basic library template, you can copy the template and change the libraries that are included in it. This is helpful if you need to apply the same settings, but to a different set of libraries. It is also helpful if you have a different naming convention for different environments such as QA and PROD.

Now consider how you can use Policy Minder to set the security attributes (ownership and authorities) for your application objects in directories. If you are writing a web-based application, you probably have your web objects in the integrated file system (IFS). This example uses the template that is used to set the values for the Policy Minder GUI objects (Figure 3-6).

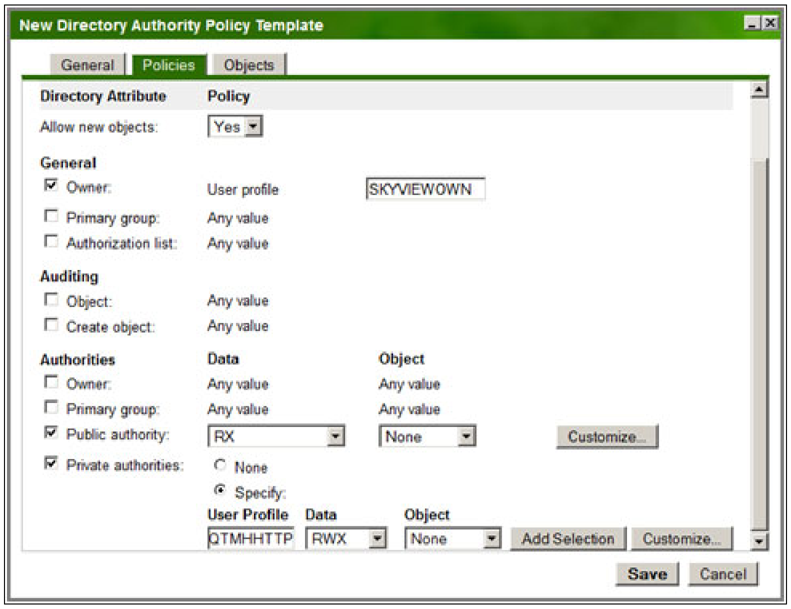


Template: *SKYPM
Checked:
Status:
[✓] Enable CheckIt
[✓] Enable FixIt
Path: /SkyView/SkyViewWeb
Include subdirectories: [✓ Yes] No
Description: Security for Policy Minder IFS objs
Notes:

```
This template is used to set the ownership and
authorities for the Policy Minder GUI objects in the
/SkyView/SkyViewWeb directory
```

Figure 3-6 New Directory Authority Policy Template General tab

As shown in these displays, the `/SkyView/SkyViewWeb` directory and all subdirectories are owned by SKYVIEWOWN and are *PUBLIC DTAAUT(*RW) OBJAUT(*NONE) with QHTTPSVR having a private authority of DTAAUT(*RWX) OBJAUT(*NONE) as shown in Figure 3-7.



Directory Attribute	Policy		
Allow new objects:	Yes		
General			
✓ Owner:	User profile	SKYVIEWOWN	
☐ Primary group:	Any value		
☐ Authorization list:	Any value		
Auditing			
☐ Object:	Any value		
☐ Create object:	Any value		
Authorities	**Data**	**Object**	
☐ Owner:	Any value	Any value	
☐ Primary group:	Any value	Any value	
✓ Public authority:	RX	None	
✓ Private authorities:	○ None / ● Specify:		
	User Profile	**Data**	**Object**
	QTMHHTTP	RWX	None

Figure 3-7 New Directory Authority Policy Template Policies tab

Like library templates, you can create one or more object templates, defining the security requirements of the objects in the directories and subdirectories. You can run a check to find the objects that do not meet your requirements, then run FixIt to set them correctly.

For AIX or Linux development, Policy Minder OPEN is available to do the same tasks as Policy Minder for IBM i.

For more information, contact SkyView Partners.

info@skyviewpartners.com

http://www.skyviewpartners.com

425-458-4975

4

Tools for understanding and modernizing RPG and COBOL

This chapter contains descriptions of the following software:

- ARCAD Software
- Surround Technologies: Accelerator development solutions

4.1 ARCAD Software

ARCAD Software makes applications useful for these tasks:

- Mass source transformation and conversion using ARCAD-Transformer
- Automated application analysis and retro-documentation by using the ARCAD-Observer

4.1.1 Mass source transformation and conversion using ARCAD-Transformer

Certain modernization, and even routine maintenance projects can involve mechanical, mass changes that affect a huge volume of source code. Typical examples are expanding the size of a field, converting an application to Unicode format, and converting to a new language syntax.   These isolated changes invariably have ripple effects, and modifying the impacted code is time-consuming, tedious, and error-prone for development teams, constituting a major cost in IT.

ARCAD-Transformer is a set of tools that are designed to automate the transformation of native IBM i applications. It covers three areas that are commonly encountered in modernization:

- Transformation of certain entities in the database and their related source code (for example, field expansion, field type conversion, and addition of an entity such as a company ID in relation to an existing one)
- Transformation of an application to be full Unicode compliant
- Transformation of RPGLE code into the newer RPG Free Format

This section introduces the ARCAD-Transformer tools, including their features, benefits, and architecture. This information is intended for IT professionals who need to understand the business value of these solutions.

For more information about application modernization, see *Modernize IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

ARCAD-Transformer mass source code transformation

With the explosion in data volume and big data comes the problem of “codes” (IDs) becoming too small. Indeed most application modernization projects on IBM i involve modifications to the database structure itself, such as increases in field size or change of field type. These mass changes have a widespread impact across the entire application, requiring painstaking updates to each related field, and each source line manipulating those fields.

The ARCAD-Transformer solution automates both the impact analysis and source modification phases that are required after a database structure change, reducing the manual effort in mass transformation projects by as much as 50 - 100%. ARCAD-Transformer identifies the source lines that are affected by the increased field size, and, when validated by the developer, automatically modifies the affected source lines themselves.

Precisely the same level of automation is provided for a change in field type. This feature is useful in database modernization, when upgrading to a more modern or standard field type (such as integer, big integer, and varchar). Also, mass conversion to a standard type opens the application for communication with other databases. A typical example is the standard SQL “date” type. Further, conversion to new types makes it easier to use standard reporting tools that are available on the market.

Specifically for database modernization, ARCAD-Transformer supports conversion from DDS to DDL (SQL). Here ARCAD offers a progressive modernization approach, compiling the associated RPG programs in addition to data. The objective is to modularize the source code towards a multitier application that modernizes RPG assets instead of discarding them, while improving overall performance by using Java and SQL.

ARCAD-Transformer Unicode migration to Unicode

Similarly, as applications are extended to the web and made usable worldwide, support for multiple character sets and double-byte character set (DBCS) in particular is making mass migration to Unicode an increasingly common requirement. ARCAD-Transformer Unicode automates the transformation of IBM i application components into Unicode, including alphanumeric database fields and the business rules themselves.

ARCAD-Transformer RPG (ARCAD-Converter) conversion from RPGLE to Free Format RPG

Another important factor in application modernization is the conversion of heritage source code to RPG Free Format to improve the modularity, maintainability, and reusability of the application in the long term. Use of Free Format syntax enhances the readability of source code. It also paves the way for new non-native developers, even Java, C# or C developers, to participate in IBM i development, facilitating mobility and communication between teams.

By automatically converting RPGLE source code into RPG Free Format, the ARCAD-Transformer RPG solution brings a major productivity boost at the start of an application modernization project. ARCAD-Transformer RPG can be used alone or as part of the ARCAD Pack for Rational, and started either from the command line or from Rational Developer for i.

History of ARCAD-Transformer

The ARCAD impact analysis and mass code conversion technology is the result of 20+ years of experience in the field of IBM i application analysis, multiple Year 2000 and euro conversion projects, and more recently Unicode migration projects.

As proof of technology, the source code of the entire ARCAD solution range has been mass converted to Unicode, using the company's own ARCAD-Transformer Unicode product. The Version 9 of the ARCAD solution range that is installed at customer sites since early 2012 is full Unicode due to the automation features that are provided.

ARCAD-Transformer can be used with the ARCAD change management solutions to manage concurrent changes. This enables ARCAD-Transformer to check out and apply changes at the last possible moment to reduce the number of changes that are made in parallel.

Business value

ARCAD-Transformer brings huge productivity gains in mass-transformation projects, typically exceeding 70%. Further, by maximizing automation in all phases and eliminating human error, the completeness of the transformation and conversion is ensured, and the resulting source code is easier to maintain.

The high level of automation means that ARCAD-Transformer typically brings a return on investment from the first project, while contributing to application reliability and reducing the risk of regression.

Solution overview

This section provides an overview of the solution.

ARCAD-Transformer mass source code transformation

For a change at the database level such as increase in field size or change in field type, the impact on related fields (such as working fields) can often mean that more than 80% of programs need to be changed.

ARCAD-Transformer automates the program changes themselves by starting with a field to be changed, and finding its relationship to other fields in other files. From this list of related fields, it finds all the relationships for those fields throughout your system, and automatically propagates that change throughout the source code within your application.

Typically ARCAD-Transformer reduces the cost of such mass transformation projects by a factor of four, automating the following tasks:

- Search *all* fields in the database based on predefined, configurable criteria
- Exhaustively search all processes by “propagating” through all fields that are derived from initial database fields
- Modify or automatically insert source code lines that are impacted
- Recompile all components and their dependencies
- Automatically retrieve data during the transfer to production phase

ARCAD-Transformer supports all native IBM i languages such as RPG (from III through Free Format), COBOL OPM or ILE, and CLP.

A mass transformation project with ARCAD-Transformer includes the following steps:

- Auditing impacted fields

  To get started, use ARCAD processes to automatically generate a complete central repository for your applications. These processes analyze your libraries and source code to derive metadata that describes all the inter-relationships between your application components. ARCAD-Transformer then uses this repository to reference all the information flows or links in your application.

  ARCAD-Transformer then helps you to develop a “source list” of the database fields (PF-TABLE SQL) that contain the entity that is concerned (for example, product code).

- Running the propagation engines

  Starting from this list of database fields, ARCAD-Transformer does a complete field propagation, by using the logical files, programs, procedures and commands, up to the DSPF or PRTF.

  To do so, ARCAD-Transformer has three “engines” that must be run successively in batch mode. The entry point for these engines is the list of fields in the database that you select for change. The first and simplest engine handles the modifications in the sources of physical files (for example, field additions, changes in size). The second engine is more extensive because it searches for all the related fields in your programs, in three distinct phases:

  – Phase 1 determines the list of programs that are affected by modification of the selected fields.

  – Phase 2 searches for the connection between the initial database fields and the different work fields in a step-by-step manner. This phase is run iteratively to obtain the wanted result.

  - Phase 3 generates source modification suggestions by using a specific engine to modify the DDS of screen and print files.

  This propagation examines all types of instruction (in any of the following languages: RPG, RPGLE, CBL, CLP, CLLE, SQL, OPNQRYF, CMD) to propagate the entity from field to field (or from field section to field section). It also detects job fields, parameters, parts of the structure, and fillers. It provides support for internal file descriptions (I/O specifications).

  At the end of this propagation, warning or error reports are generated. After a guided analysis of these errors, ARCAD-Transformer corrects the list of initial fields and sets some limiters. The propagation process is then resubmitted in batch. After a few iterations, a detailed, precise impact result is obtained, down to the source line level.

- Applying modifications to source

  Before applying modifications to the source, ARCAD-Transformer automatically "checks out" the impacted components. The source lines are then flagged for modification. The only remaining work for the developer is to check the flagging and accept or reject the flag for modification. The components are now under change control.

- Recompiling components

  The final phase is recompilation of the application. Because of the application knowledge stored in the repository, an optimized recompilation list is generated that takes into account the dependencies between components and ensures a complete build.

- Generating data conversion programs

  ARCAD-Transformer is also able to generate data conversion programs to convert to the new database structure. After specifying the retrieval rules, it generates one program per impacted file.

ARCAD-Transformer Unicode conversion to Unicode

Universal language support in an application requires all alphanumeric fields in the database and all management rules to be converted into Unicode.

However, there are some limitations to the "all Unicode" approach. For example, the EBCDIC (SBCS/DBCS) encodings must be conserved in certain cases, as the following features, among others, do not support Unicode, or support it incompletely:

- DSPF/PRTF
- CLP/CLLE variables
- Using commands
- API calls from the OS

Further, it is sometimes necessary to enlarge certain fields to allow DCBS characters to be accepted in them (minimum length = 4).

ARCAD-Transformer is designed to take these exceptions into account, with these objectives:

- Conserving maintained source code that is still high maintainable (similar to the original source code)
- Inserting the necessary conversions upon compilation
- Optimizing conversions of literals.

The ARCAD-Transformer Unicode solution involves two distinct steps:

1. Transformation of applications into Unicode

   This step involves a one-shot transformation of the application components into Unicode by using ARCAD-Transformer, including these tasks:

   – Assistance in selecting the fields that are concerned (one of the following):
     - Only from database fields
     - All alphanumeric fields

   – Propagation through multiple iterations
     - Warning reports that flag cases that require sample manual adaptations
     - Redefining numeric fields into alphabetical fields that become Unicode
     - Transition of the indicator parameter to Unicode
     - Using instructions that do not support Unicode
     - Detecting literals with variant characters or characters with no equivalent in certain languages

   – Transforming source code into Unicode
     - Changing database fields into Unicode UCS2 (Graphic CCSID 13488)
     - Changing RPGLE fields into type C, or expanding structures (certain parts of which become Unicode)
     - Declaring fields in Specification D instead of Specification C
     - Doubling lengths in CLP/CLLE fields
     - Converting operation code that does not support Unicode into operation code that does allow it
     - Changing literals into Unicode constants

   – Generating the retrieval program (between the old SBCS files and the new Unicode files).

2. ARCAD Pre-compiler for Unicode:

   The ARCAD-Transformer Unicode solution has the overriding goal of ensuring that converted source code is no more complex than the original source. Therefore, the converted code that is modified during the transition to Unicode has no more lines than the original source. Also, the conversions to SBCS fields that are required for certain features are not present in it.

   To achieve this goal, the compilation of this source code (for example, of type RPGLE,) passes first through the ARCAD Pre-compiler for Unicode, to ensure these tasks occur:

   – Checks on certain instructions (for example, using variant characters or characters that do not exist in every country)

   – Pre-display/post-entry conversions for DSPF/PRTF fields

   – Unicode support in CLP/CLLE (for literals, concatenations, command calls, and so on) by inserting all of the necessary conversions

   – In RPGLE, changes of the literals into Unicode constants to optimize processing time

   – Insertion of pre-/post non-Unicode third-party program call conversions (other application or OS API).

   A second piece of source code is thus generated (for example, of type RPGLE2) and it is this code that is compiled.

If you develop using the LPEX editor in Rational Developer for i, pre-compilation errors (ARCAD) or compilation errors (IBM) are conveniently displayed along with the corresponding source line.

ARCAD change management handles these two kinds of source code transparently, as they correspond to a single object. You modify only the maintained source code. However, the generated source code is retained and regenerated as many times as you want.

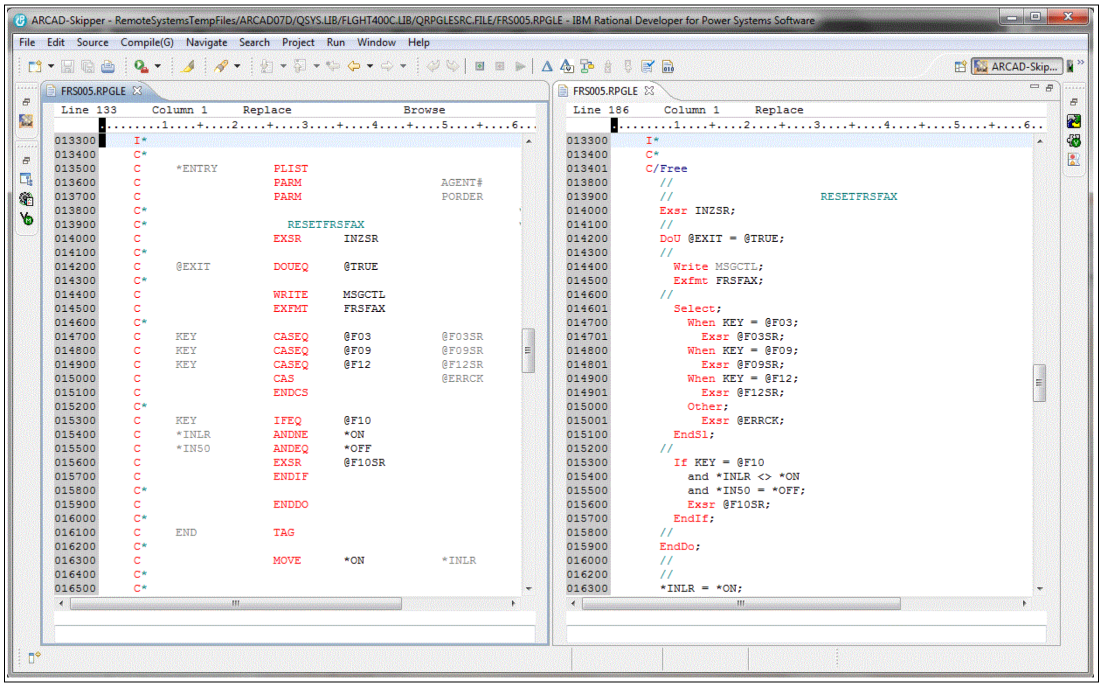
An ARCAD plug-in to the IBM Debugger allows you to view/modify CLP fields as though they were Unicode type.

ARCAD-Transformer RPG (ARCAD-Converter) Automated conversion from RPGLE to Free Format RPG

For any application modernization project, employ the latest RPG language syntax to improve the modularity, maintainability, and reusability of the application in the long term. The move to Free Format enhances the readability of source code and paves the way for new non-native developers to join the development team. Even Java, C#, or C developers can work in Free Format RPG code.

To facilitate the move to Free Format, the ARCAD-Transformer RPG is a solution that can be started from Rational Developer for i or the command line to automatically convert the RPGLE source code.

The objective of ARCAD-Transformer RPG is to automatically convert all C specifications into free syntax. Whereas this process is straightforward for certain operation codes (for example, EVAL and ADD), others are more complex (for example, MOVE, MOVEL). Indeed, the choice of instruction to generate often depends on the types, lengths, and dimensions of fields that are used in Factor 1, Factor 2, or results factor. For this reason, ARCAD-Transformer RPG starts by a complete cross-reference analysis down to the field level to check the characteristics of each field in the program (Figure 4-1).


ARCAD-Skipper - RemoteSystemsTempFiles/ARCAD07D/QSYS.LIB/FLGHT400C.LIB/QRPGLESRC.FILE/FRS005.RPGLE - IBM Rational Developer for Power Systems Software
File Edit Source Compile(G) Navigate Search Project Run Window Help
ARCAD-Skip...
FRS005.RPGLE
FRS005.RPGLE

```
Line 133     Column 1      Replace                  Browse
 .........1....+....2....+....3....+....4....+....5....+....6...
     I*
     C*
     C     *ENTRY        PLIST
     C                   PARM                    AGENT#
     C                   PARM                    PORDER
     C*
     C*                    RESETFRSFAX
     C                   EXSR      INZSR
     C*
     C     @EXIT         DOUEQ     @TRUE
     C*
     C                   WRITE     MSGCTL
     C                   EXFMT     FRSFAX
     C*
     C     KEY           CASEQ     @F03          @F03SR
     C     KEY           CASEQ     @F09          @F09SR
     C     KEY           CASEQ     @F12          @F12SR
     C                   CAS                     @ERRCK
     C                   ENDCS
     C*
     C     KEY           IFEQ      @F10
     C     *INLR         ANDNE     *ON
     C     *IN50         ANDEQ     *OFF
     C                   EXSR      @F10SR
     C                   ENDIF
     C*
     C                   ENDDO
     C*
     C     END           TAG
     C*
     C                   MOVE      *ON           *INLR
     C*
     C*
```

```
Line 186     Column 1      Replace
 .........1....+....2....+....3....+....4....+....5....+....6..
     I*
     C*
     C/Free
       //
       //                        RESETFRSFAX
       Exsr INZSR;
       //
       DoU @EXIT = @TRUE;
       //
         Write MSGCTL;
         Exfmt FRSFAX;
       //
         Select;
           When KEY = @F03;
             Exsr @F03SR;
           When KEY = @F09;
             Exsr @F09SR;
           When KEY = @F12;
             Exsr @F12SR;
           Other;
             Exsr @ERRCK;
         EndSl;
       //
         If KEY = @F10
           and *INLR <> *ON
           and *IN50 = *OFF;
           Exsr @F10SR;
         EndIf;
       //
       EndDo;
       //
       //
       *INLR = *ON;
```

Figure 4-1 Comparison of RPGLE and its Free Format equivalent generated by ARCAD-Converter

The converted source can usually be compiled directly, with user intervention required only for the handling of %Found and %Equal on SCAN, CHECK, CHECKR, and LOOKUP.

Solution architecture

ARCAD-Transformer is structured as a series of processes to be started iteratively in batch. Processes use the ARCAD repository of application metadata that is generated automatically from a list of application libraries (Figure 4-2).

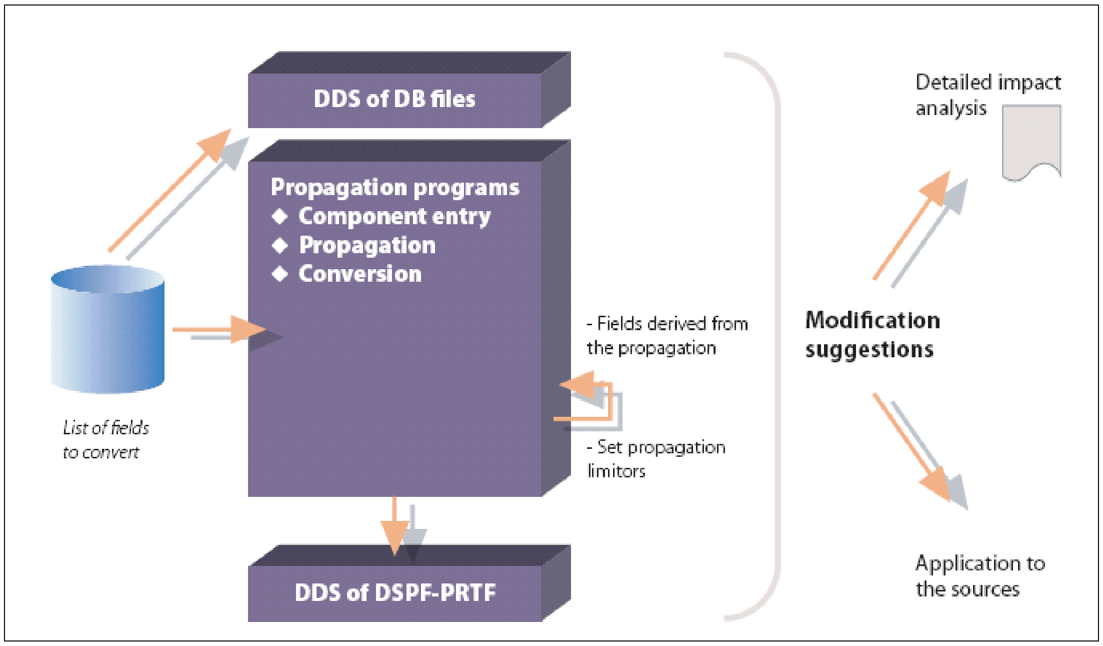


Figure 4-2 Impact analysis with propagation from field to field, and source modification with ARCAD-Transformer

Usage scenarios

ARCAD-Transformer can be used to automate the mass transformation of sources in the following scenarios.

Increasing the size of a field

When you are expanding the size of a database field, source modifications are made automatically for these objects:

- All database files
- All programs, for example:
  - Length of declared fields
  - Positions that are shifted in the DSs
  - Modification of the values in %SUBST instructions
- All the DSPF/PRTFs. In the PRTFs, adjoining fields are "pushed" if necessary to provide the available space (or else the literals are cropped).

Change in field type

The same level of automation is achieved when you are changing field type, such as when you are changing a customer code from numeric to alphanumeric.

Standardization in storage of an entity

The same method applies when you are standardizing the way that an entity is stored. For example, in the case where an entity is sometimes managed in P(3.0) and sometimes in S(5.0), you might want to change it to I(10.0). ARCAD-Transformer is able to automate this transition across your entire code base.

Adding an entity in relation to another entity

Another example is when you require an entity to be inserted wherever an entity is used, such as adding a subcategory code when category code is present. ARCAD-Transformer automatically modifies source code by adding a line for the new entity, when the fields examined have these characteristics:

- Declared
- Included in parameters
- Used as a key
- Assigned
- Compared

Source code is modified by putting obsolete fields into comments (or deleting them entirely). If a field contains only the entity that is examined, the line is deleted. However, if the field that is examined is stored in larger structures, then these structures are collapsed (by shifting the next fields).

Conversion to Unicode

As described previously, the transformation of application data and source code to Unicode can be automated by using the ARCAD-Transformer Unicode extension.

Supported platforms

ARCAD-Transformer and ARCAD-Transformer RPG (ARCAD-Converter) are supported by platforms with the following features:

- IBM i operating software V5R4 and later
- x86 64-bit systems with a minimum of 4 GB memory
- Minimum 40 GB of disk storage

Ordering information

ARCAD-Transformer licenses and support can be purchased from ARCAD Software. Contact your regional representatives at sales-eu@arcadsoftware.com, sales-us@arcadsoftware.com, or sales-asia@arcadsoftware.com.

Related information

For more information, see the following links:

- ARCAD-Transformer RPG - automated conversion of RPGLE to Free Format RPG

  http://www.arcadsoftware.com/resource-items/arcad-transformer-rpg/

- Expanding field size with ARCAD-Transformer

  http://www.arcadsoftware.com/products/arcad-transformer-ibm-i-refactoring-tools/arcad-transformer-field/

- ARCAD-Transformer Unicode - automated conversion to Unicode

  http://www.arcadsoftware.com/products/arcad-transformer-ibm-i-refactoring-tools/arcad-transformer-for-unicode/

4.1.2 Automated application analysis and retro-documentation by using the ARCAD-Observer

Software development teams can often spend as much as 80% of their time trying to understand the internal architecture of an application before they modify it. If you are involved in application maintenance or modernization, you need an efficient way to find structural and dependency information within complex applications and have instant access to up-to-date technical documentation. The solution to this business problem is application analysis software that is technology-independent and focuses on productivity and ease-of-use.

ARCAD-Observer is an Eclipse-based solution to reveal application architecture and rules with a set of graphical tools. All software and data components that are associated with an application are analyzed, whether they are IBM i based or are on another platform, such as Windows, AIX, Linux. ARCAD-Observer can be used to inspect an application and discover interrelationships between components, and also automatically generate offline technical documentation in HTML format. The solution enables developers to sift through a vast quantity of diverse information and rapidly understand an application before making a software change.

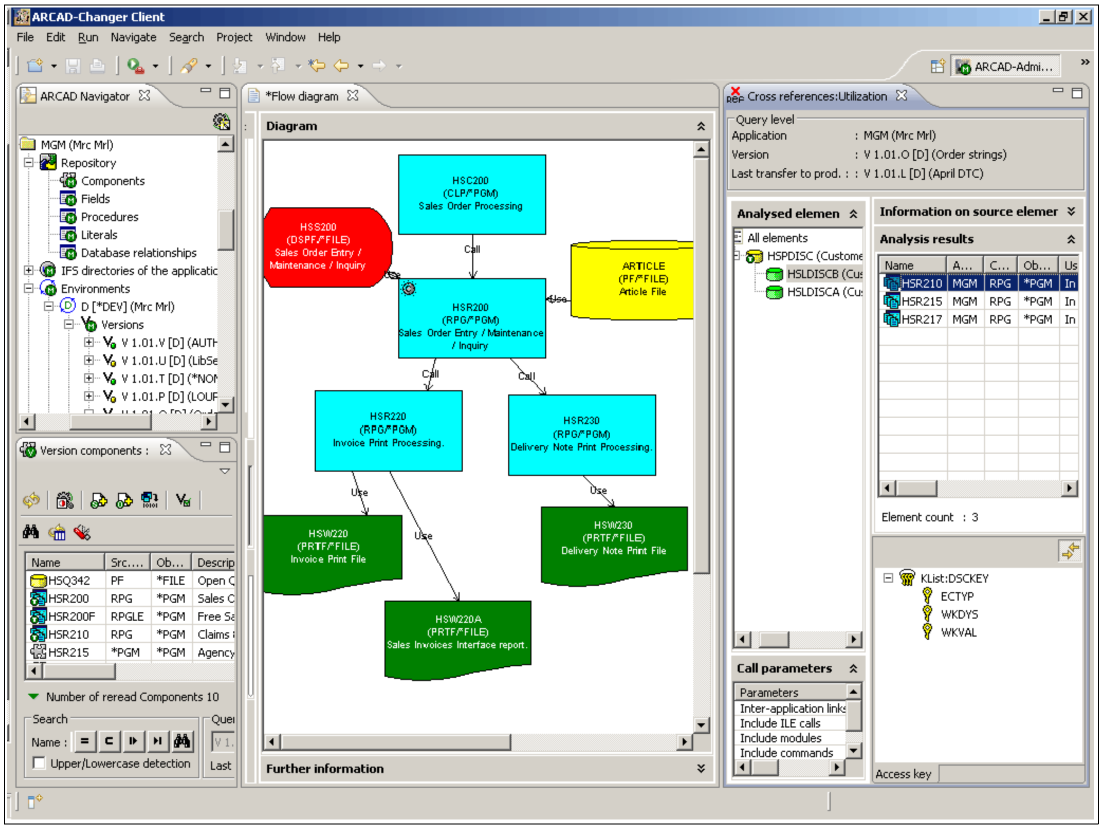
This section introduces ARCAD-Observer including its features, benefits, and architecture. This information is intended for IT professionals who need to understand the business value of ARCAD-Observer.

For more information about the topic of application modernization, see *Modernize IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

Contents

ARCAD-Observer provides application intelligence for maintaining and transferring knowledge of existing systems. Whether the context is application maintenance or modernization, the basic needs are the same: to instantly locate structural and dependency information and browse technical documentation that is relevant and up to date.

Figure 4-3 shows using ARCAD-Observer ti generate a workflow diagram.


ARCAD-Changer Client
File Edit Run Navigate Search Project Window Help
ARCAD Navigator
*Flow diagram
Cross references:Utilization
Diagram
HSC200 (CLP/*PGM) Sales Order Processing
HSS200 (DSPF/*FILE) Sales Order Entry / Maintenance / Inquiry
ARTICLE (PF/*FILE) Article File
HSR200 (RPG/*PGM) Sales Order Entry / Maintenance / Inquiry
HSR220 (RPG/*PGM) Invoice Print Processing.
HSR230 (RPG/*PGM) Delivery Note Print Processing.
HSW220 (PRTF/*FILE) Invoice Print File
HSW230 (PRTF/*FILE) Delivery Note Print File
HSW220A (PRTF/*FILE) Sales Invoices Interface report.
Call
Use
Further information
MGM (Mrc Mrl)
Repository
Components
Fields
Procedures
Literals
Database relationships
IFS directories of the applicatio
Environments
D [*DEV] (Mrc Mrl)
Versions
V 1.01.V [D] (AUTH
V 1.01.U [D] (LibSe
V 1.01.T [D] (*NON
V 1.01.P [D] (LOUF
Version components :
Number of reread Components 10
Search
Name :
Upper/Lowercase detection
Analysed elemen
All elements
HSPDISC (Custome
HSLDISCB (Cus
HSLDISCA (Cus
Information on source elemer
Analysis results
Element count : 3
KList:DSCKEY
ECTYP
WKDYS
WKVAL
Call parameters
Parameters
Inter-application links
Include ILE calls
Include modules
Include commands
Access key

Query level

Application	: MGM (Mrc Mrl)
Version	: V 1.01.O [D] (Order strings)
Last transfer to prod. :	: V 1.01.L [D] (April DTC)

Name	Src....	Ob...	Descrip
HSQ342	PF	*FILE	Open C
HSR200	RPG	*PGM	Sales C
HSR200F	RPGLE	*PGM	Free Sa
HSR210	RPG	*PGM	Claims
HSR215	*PGM	*PGM	Agency

Name	A...	C...	Ob...	Us
HSR210	MGM	RPG	*PGM	In
HSR215	MGM	RPG	*PGM	In
HSR217	MGM	RPG	*PGM	In

Figure 4-3 Use of ARCAD-Observer to generate a workflow diagram and do impact analysis

ARCAD-Observer facilitates the management of technical knowledge. For new or veteran developers, it offers an easy alternative to reading through reams of source code. ARCAD-Observer can be used to graphically navigate through application architecture, analyze impacts and relationships across multiple platforms and languages, and to automatically generate documentation. New staff members are productive more rapidly, application lifetime is extended, and regulatory obligations are fulfilled. In the case of Application Management, ARCAD-Observer brings transparency to workload estimates and, for the customer, a technical grasp over its applications.

Considerations

Analysis of the underlying architecture of a business application is usually the first step in a modernization project so that organizations can keep the investment they have made in their existing system. Composite teams, including new hires, are often assigned to design new web interfaces to heritage applications, and understanding a system that has grown over many years can be a daunting task. Manually created documentation is often non-existent or out-of-date. Further, manual impact analysis is costly and time-consuming, leading to imprecision in project estimates, resource needs, and technical complexity.

Application modernization requires a reassessment of user needs coupled with an analysis of the existing system to determine answers to these questions:

- Are current user needs still covered?
- Are the applications maintainable?
- Will the applications still be maintainable in five years?
- Can application code be reused in a modernization project?

Automated analyses can give an accurate indication of code quality and complexity, pinpointing the areas to be replaced or reworked. Analysis tools identify, archive, and clean up redundant code, improving the productivity of any mass source change. Regressions at later stages in the project are avoided by highlighting duplicate or mismatching objects.

Many in-house developed and therefore specific business applications hold unique competitive advantages for the organization that no standard software package can fulfill. An automated retro-documentation tool can facilitate code comprehension and reveal valuable business logic and features to ensure that they are preserved in a re-engineered or customized system.

Business value

ARCAD-Observer provides IT and project managers with a productivity tool to analyze software and file dependencies. It accurately determines project scope and improves software estimations while it generates a component list of artifacts that are needed for the automation and integrity of the build operation.

By speeding code comprehension and automating impact analysis, ARCAD-Observer reduces the learning curve for development teams, and enables accurate preparation of code and files for project stream builds. Further, by facilitating understanding of software and dependencies, ARCAD-Observer can increase accuracy and speed of project development, realizing gains of 15-20%.

Solution overview

This section provides an overview of the solution.

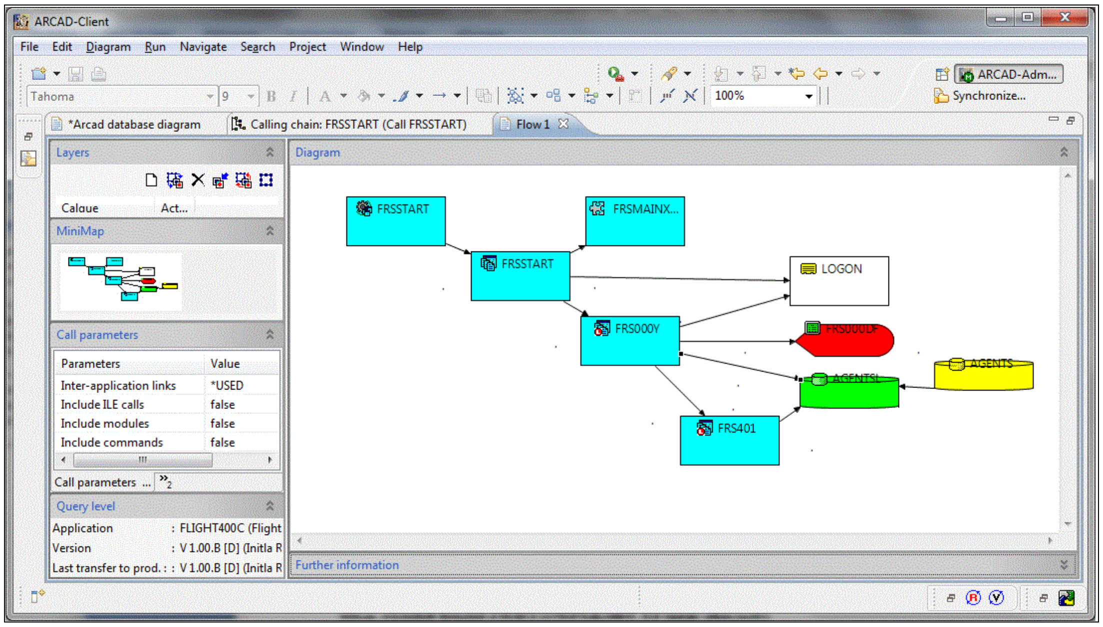
Application mining

ARCAD-Observer is an efficient tool for impact analysis, referencing all dependencies between applications and components down to the field and source line level, whatever the language used (for example, COBOL, CL, SQL, RPG III through Free Format, and ILE). In addition, all cross-platform dependencies between IBM i components (programs, files, fields,) and other non-native components (such as Java, PHP, and C#) are traced and visible directly from the developer workbench.

With ARCAD-Observer, you can analyze the information flow across processes and quickly identify business rules that are contained in the code. Its built-in diagram editor displays information as graphics. You can easily enhance these diagrams and include them in documentation.

ARCAD-Observer diagrams show how files relate to each other, how a set of programs accesses a group of files, and how modules in an information system interact.

For example, the workflow diagram in Figure 4-4 combines file usage with program call information and actual values of call parameters.

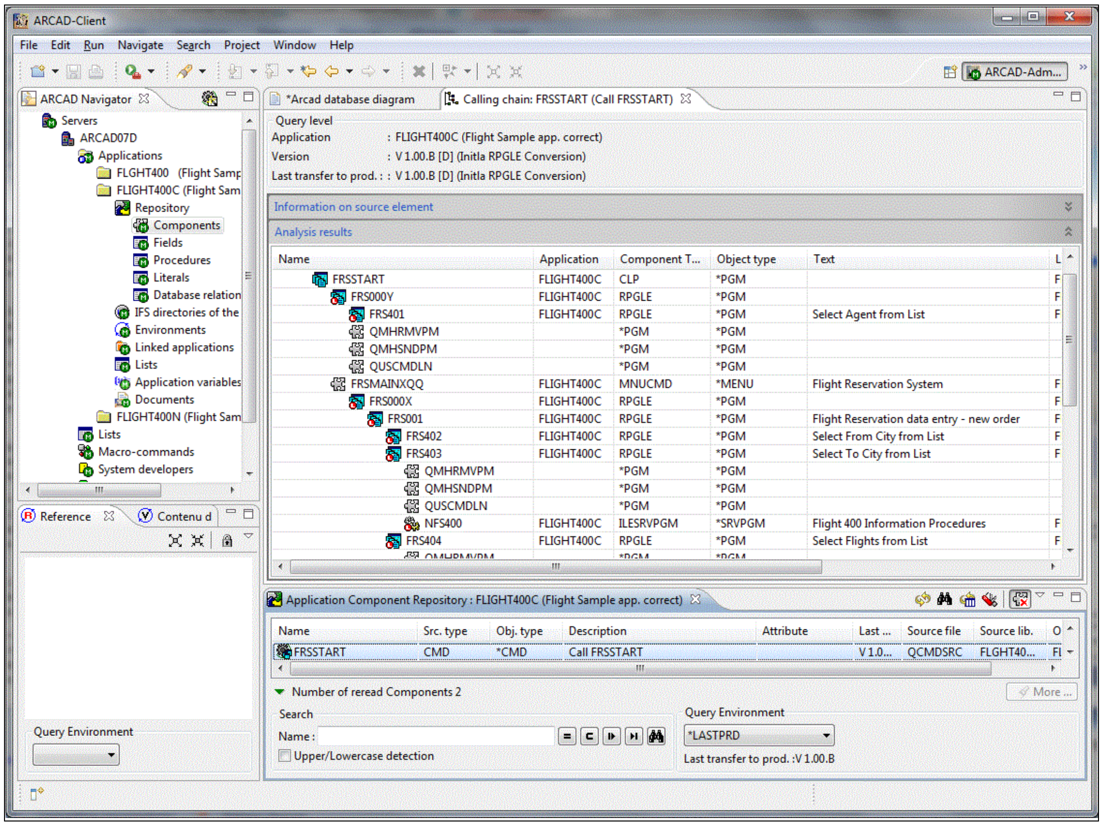


Parameters	Value
Inter-application links	*USED
Include ILE calls	false
Include modules	false
Include commands	false

Application : FLIGHT400C (Flight
Version : V 1.00.B [D] (Initla R
Last transfer to prod. : : V 1.00.B [D] (Initla R

Figure 4-4 Workflow diagram that is generated from ARCAD-Observer with call parameter values

ARCAD-Observer shows where a procedure is used, and where a field is updated, with a precise source line view. The full program and procedure calling chain can be displayed at any point in the interface, together with parameters that are passed (Figure 4-5).



Query level
Application : FLIGHT400C (Flight Sample app. correct)
Version : V 1.00.B [D] (Initla RPGLE Conversion)
Last transfer to prod. : : V 1.00.B [D] (Initla RPGLE Conversion)

Name	Application	Component T...	Object type	Text	L
FRSSTART	FLIGHT400C	CLP	*PGM		F
FRS000Y	FLIGHT400C	RPGLE	*PGM		F
FRS401	FLIGHT400C	RPGLE	*PGM	Select Agent from List	F
QMHRMVPM		*PGM	*PGM		
QMHSNDPM		*PGM	*PGM		
QUSCMDLN		*PGM	*PGM		
FRSMAINXQQ	FLIGHT400C	MNUCMD	*MENU	Flight Reservation System	F
FRS000X	FLIGHT400C	RPGLE	*PGM		F
FRS001	FLIGHT400C	RPGLE	*PGM	Flight Reservation data entry - new order	F
FRS402	FLIGHT400C	RPGLE	*PGM	Select From City from List	F
FRS403	FLIGHT400C	RPGLE	*PGM	Select To City from List	F
QMHRMVPM		*PGM	*PGM		
QMHSNDPM		*PGM	*PGM		
QUSCMDLN		*PGM	*PGM		
NFS400	FLIGHT400C	ILESRVPGM	*SRVPGM	Flight 400 Information Procedures	F
FRS404	FLIGHT400C	RPGLE	*PGM	Select Flights from List	F

Application Component Repository : FLIGHT400C (Flight Sample app. correct)

Name	Src. type	Obj. type	Description	Attribute	Last ...	Source file	Source lib.	O
FRSSTART	CMD	*CMD	Call FRSSTART		V 1.0...	QCMDSRC	FLGHT40...	Fl

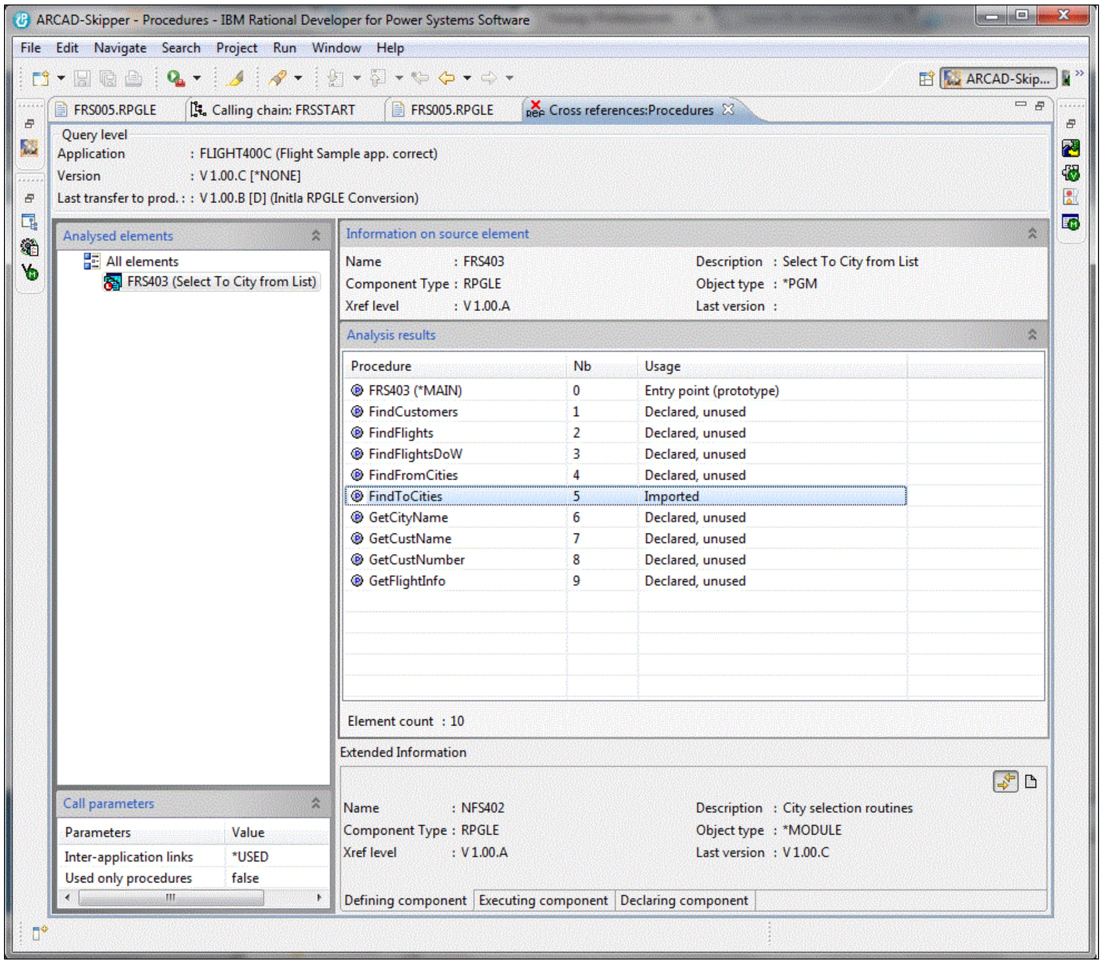
Number of reread Components 2

Search
Name :
Upper/Lowercase detection

Query Environment
*LASTPRD
Last transfer to prod. :V 1.00.B

Figure 4-5 Automatically generated diagrams from ARCAD-Observer (example: Calling Chain)

In line with modern programming practices on IBM i, ARCAD-Observer provides management of technologies such as ILE and SQL. Users are shielded from any additional complexity and can easily visualize how ILE programs are structured (Figure 4-6) and how files are accessed in embedded SQL.


ARCAD-Skipper - Procedures - IBM Rational Developer for Power Systems Software
File Edit Navigate Search Project Run Window Help
ARCAD-Skip...
FRS005.RPGLE
Calling chain: FRSSTART
FRS005.RPGLE
Cross references:Procedures
Analysed elements
All elements
FRS403 (Select To City from List)
Information on source element
Analysis results
Call parameters
Extended Information
Defining component
Executing component
Declaring component

Query level

Application	: FLIGHT400C (Flight Sample app. correct)
Version	: V 1.00.C [*NONE]
Last transfer to prod. :	: V 1.00.B [D] (Initla RPGLE Conversion)

Name	: FRS403	Description	: Select To City from List
Component Type	: RPGLE	Object type	: *PGM
Xref level	: V 1.00.A	Last version	:

Procedure	Nb	Usage
FRS403 (*MAIN)	0	Entry point (prototype)
FindCustomers	1	Declared, unused
FindFlights	2	Declared, unused
FindFlightsDoW	3	Declared, unused
FindFromCities	4	Declared, unused
FindToCities	5	Imported
GetCityName	6	Declared, unused
GetCustName	7	Declared, unused
GetCustNumber	8	Declared, unused
GetFlightInfo	9	Declared, unused

Element count : 10

Parameters	Value
Inter-application links	*USED
Used only procedures	false

Name	: NFS402	Description	: City selection routines
Component Type	: RPGLE	Object type	: *MODULE
Xref level	: V 1.00.A	Last version	: V 1.00.C

Figure 4-6 View of ILE procedure usage from ARCAD-Observer

ARCAD-Observer supports multi-level views. That is, the viewpoint for analysis and diagramming can be switched between “version” levels to ensure accurate reporting on the production environment, test environment, or a particular development branch.

The following video demonstrates the simple steps that are required when you are developing for IBM i to analyze dependencies by using ARCAD-Observer from the ARCAD Pack for Rational together with IBM Rational Team Concert™ and Rational Developer for i:

`https://jazz.net/library/video/1319`

Macroscopic views

When you have thousands of components in your information system, you need a tool to provide a global or “big picture”. For this requirement, ARCAD-Observer provides “macroscopic views”. With this feature, you can easily subdivide your applications into different functional domains, associate your components to those domains, and then view all dependencies at a functional level.

Users obtain an overall "map" of an information system and can visualize dependencies between the different domains (Figure 4-7).

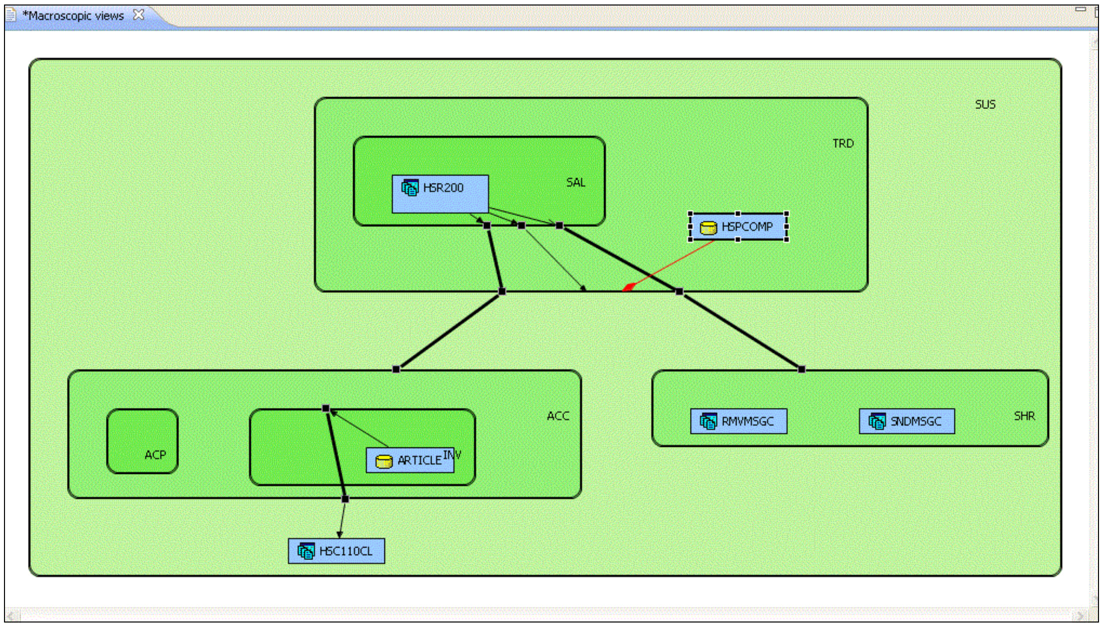


Figure 4-7 Automatically generated diagrams from ARCAD-Observer (Macroscopic/Functional View)

Isolation of business logic

ARCAD-Observer is valuable when you are modernizing heritage systems for use on the web. The solution analyzes application libraries and source code to uncover critical business processes and interdependencies. It separates and documents critical business logic so that the gathered knowledge or metadata can be reused in new design and development efforts.

Large, monolithic applications can be analyzed to reveal the lines of "like" source that can be combined into a method, see the code that is no longer callable, and understand visually how various other aspects of the code interact together. For re-engineering purposes, business rules are easy to identify and code areas can be selected for provision as web services. Source code can be more easily re-engineered into small inter-connectable, reusable procedures.

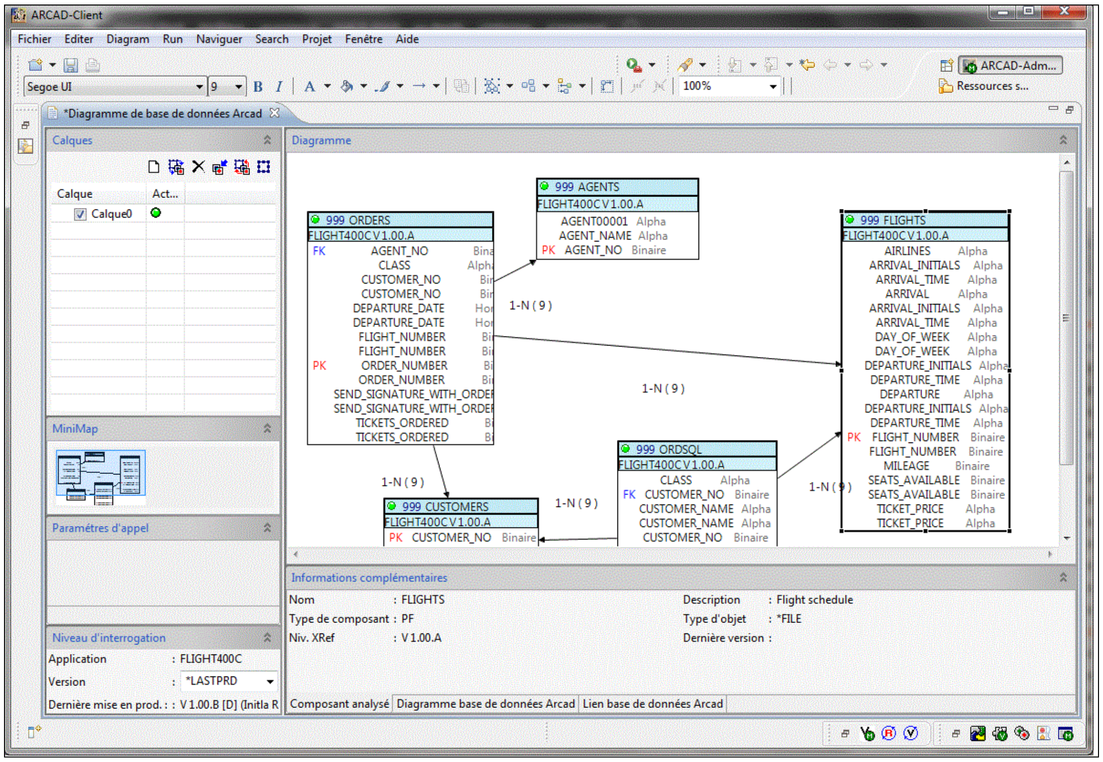
ARCAD-Observer also helps evolving and developing new applications around an existing system. It rapidly builds a global view of your application that can be navigated interactively to help understand heritage applications quickly and efficiently.

Database re-engineering

ARCAD-Observer is able to reverse-engineer the database relationship model that underlies an application, particularly in the case of flat files, generating SQL (DDL) if required. To do so, it uses a system of rules to analyze the program source code and identify complex database relations.

Multiple data exploration techniques are used to discover the database relationship model, adapting equally well to heritage applications without any declared keys/constraints and also to new database models. ARCAD-Observer uses name syntax, type and length information, and reference file constraints during the analysis process. A powerful discovery engine identifies foreign keys by parsing programs and propagating field content over the program calling chain until another file or table is reached.

Figure 4-8 shows an example of the analysis process.



Nom : FLIGHTS
Type de composant : PF
Niv. XRef : V 1.00.A
Description : Flight schedule
Type d'objet : *FILE
Dernière version :

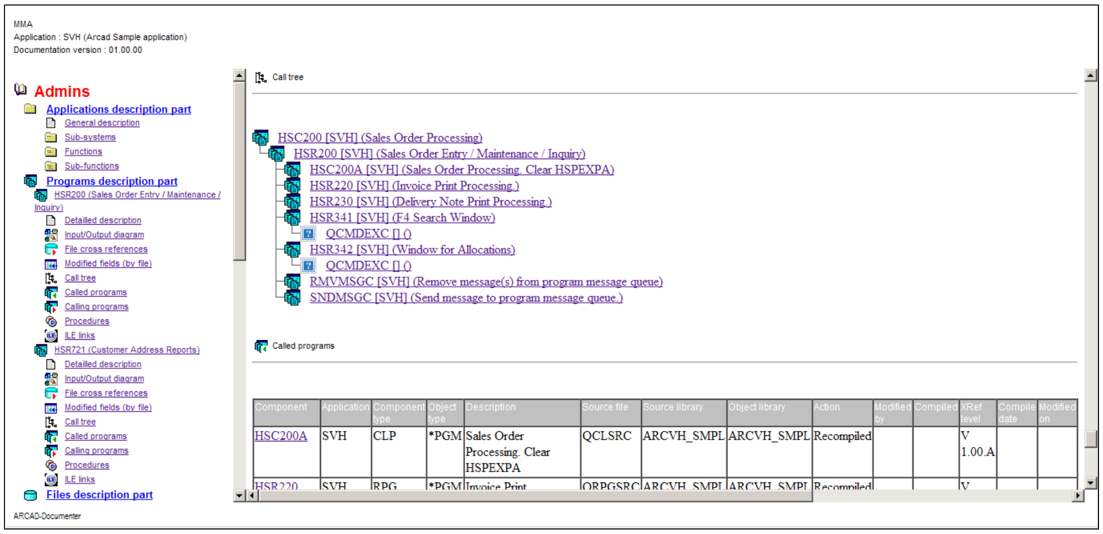
Application : FLIGHT400C
Version : *LASTPRD
Dernière mise en prod. : : V 1.00.B [D] (Initla R

Figure 4-8 ARCAD-Observer during the analysis process

Reverse-engineering of database relationship model with ARCAD-Observer

Access to and distribution of knowledge on the existing system is also critical in any modernization project. If you discover that your application documentation is obsolete, or even non-existent, ARCAD-Observer offers an immediate solution that fulfills regulatory requirements (Sarbanes-Oxley, FDA, ISO 9000, and so on).

ARCAD-Observer automatically generates technical documentation for an application, using chapter definitions that are easily customized for the target audience (for example, development teams, auditors). You can enhance the automatically generated documentation by adding text or customizing diagrams using the documentation editor (Figure 4-9). The standard export format is HTML for ready distribution to anyone with a browser. An update procedure can be defined to ensure that documentation is automatically kept up to date with application changes.


MMA
Application : SVH (Arcad Sample application)
Documentation version : 01.00.00

- Admins
  - Applications description part
    - General description
    - Sub-systems
    - Functions
    - Sub-functions
  - Programs description part
    - HSR200 (Sales Order Entry / Maintenance / Inquiry)
      - Detailled description
      - Input/Output diagram
      - File cross references
      - Modified fields (by file)
      - Call tree
      - Called programs
      - Calling programs
      - Procedures
      - ILE links
    - HSR721 (Customer Address Reports)
      - Detailled description
      - Input/Output diagram
      - File cross references
      - Modified fields (by file)
      - Call tree
      - Called programs
      - Calling programs
      - Procedures
      - ILE links
  - Files description part

Call tree

- HSC200 [SVH] (Sales Order Processing)
  - HSR200 [SVH] (Sales Order Entry / Maintenance / Inquiry)
    - HSC200A [SVH] (Sales Order Processing. Clear HSPEXPA)
    - HSR220 [SVH] (Invoice Print Processing.)
    - HSR230 [SVH] (Delivery Note Print Processing.)
    - HSR341 [SVH] (F4 Search Window)
      - QCMDEXC [] ()
    - HSR342 [SVH] (Window for Allocations)
      - QCMDEXC [] ()
    - RMVMSGC [SVH] (Remove message(s) from program message queue)
    - SNDMSGC [SVH] (Send message to program message queue.)

Called programs

Component	Application	Component type	Object type	Description	Source file	Source library	Object library	Action	Modified by	Compiled	XRef level	Compile date	Modified on
HSC200A	SVH	CLP	*PGM	Sales Order Processing. Clear HSPEXPA	QCLSRC	ARCVH_SMPL	ARCVH_SMPL	Recompiled			V 1.00.A		
HSR220	SVH	RPG	*PGM	Invoice Print	QRPGSRC	ARCVH_SMPL	ARCVH_SMPL	Recompiled			V		

ARCAD-Documenter

Figure 4-9 Auto-generation of technical application documentation, in HTML format as a website

Solution architecture

ARCAD-Observer uses application metadata or knowledge that is stored in the ARCAD-Open Repository. After the product is installed, it needs an application definition to start its analysis. An application is defined as a set of libraries for the IBM i server side, or a set of directories for the open systems. An ARCAD process then runs in batch to find and record all the dependencies between components and fields down to the source line level. It is also possible to run a prior audit on your applications by using the ARCAD-Audit solution to clean up, archive, and remove obsolete components from the start of the project.

ARCAD-Observer is able to analyze all components and artifacts in the information system whatever their host platform, such as RPG, COBOL, Java, C#, and PHP. It offers a single, common tool for managing application knowledge, a standard export format for wide and rapid distribution of documentation, and a shared point of communication between teams. See Figure 4-10.

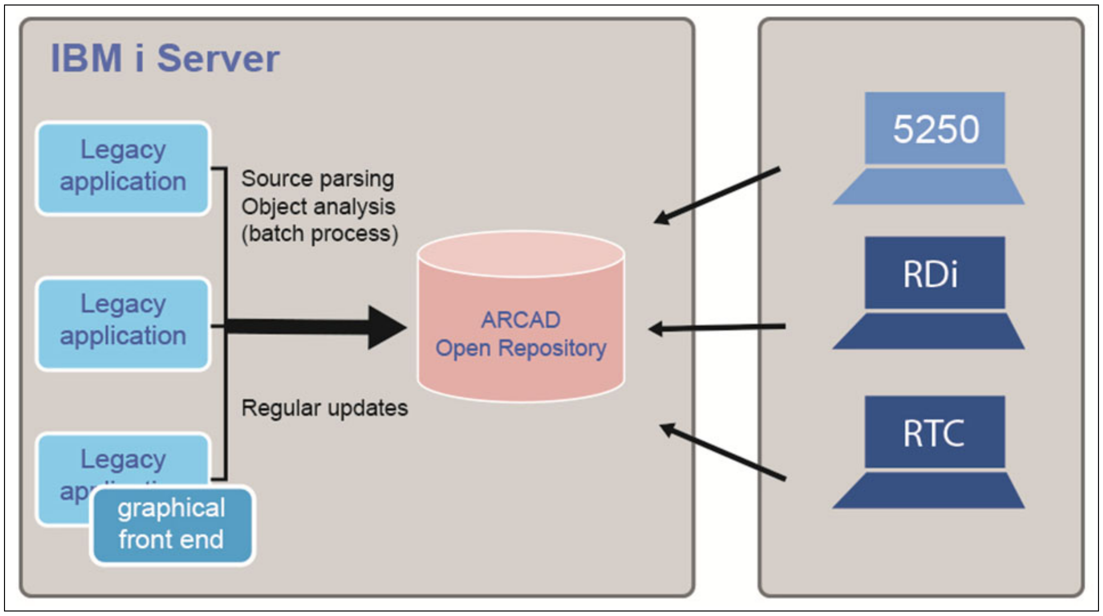


Figure 4-10 Architecture of ARCAD-Observer solution, with multiple interface options

Usage scenarios

As a general-purpose and multi-platform solution for application analysis and management of application knowledge, ARCAD-Observer can be used to improve productivity and accuracy in various projects:

- Analysis and audit of the information system before application modernization
- Workload and cost estimates for software change projects
- Knowledge transfer on an information system
- Ramp-up of new development staff
- Rapid and precise impact analysis during software change
- Test fixes
- Detection of bugs and areas of code complexity
- Eased development of new (application or user) interfaces
- Development of web services,
- Restructuring of applications to n-tier or modular architecture,
- Rearchitecting of applications to a service-oriented architecture (SOA),
- Migration of applications to new technologies/platforms

Supported platforms

This solution is supported by platforms with the following features:

- IBM i operating software V5R4 and later
- x86 64-bit systems with a minimum of 4 GB memory
- Minimum 40 GB of disk storage

Ordering information

ARCAD-Observer licenses and support can be purchased from ARCAD Software. Contact your regional representatives at sales-eu@arcadsoftware.com, msales-us@arcadsoftware.com, or sales-asia@arcadsoftware.com.

ARCAD-Observer is also available through IBM Passport Advantage®.

Table 4-1 shows the ordering information.

Table 4-1 IBM Ordering part number and feature code

Program name	PID number	Part number
ARCAD Pack for Rational per Observer Authorized User	5725-L13	D12EXLL
IBM Elite Support for ARCAD Pack for Rational per Observer Authorized User	5725-L24	D12F1LL
ARCAD Pack for Rational per Observer Floating User	5725-L13	D12EZLL
IBM Elite Support for ARCAD Pack for Rational per Observer Floating User	5725-L24	D0ZJMLL

Related information

For more information, see the following website:

http://www.arcadsoftware.com/products/arcad-observer-application-analysis/

4.2 Surround Technologies: Accelerator development solutions

Accelerator is a software development solution to quickly provide people with high quality, highly functional software they love to use.

It is estimated that over 50% of total IT spending is used for ongoing software operations and maintenance of existing solutions and for companies with heritage systems. When most of a company's IT budget is dedicated to application maintenance, change requests, and general system operations, IT has no capacity to deliver innovative business solutions.

This situation is the problem that the Accelerator Development Solutions from Surround Technologies solve.

Too often companies look at software modernization as distinctly different from software development itself. This process can lead them down the wrong path of selecting tactical and wasteful temporary or one-off type solutions or strategic risky, large rip and replace type solutions. Both have a high level of failure and neither address the root of the problem.

Accelerator is different than other development solutions because it is not just one part of the solution or the other. It is both, but it is even more than that.

Whether you must modernize monolithic heritage applications or deliver new software, this section describes how Surround's “RADx3 Accelerated Software Development” process and

“Accelerator Development Solutions” enables application developers to keep up with the business so the business can stay ahead of the competition.

Surround Technologies substantially accelerates the creation, modernization, delivery, and maintenance of high-quality, state-of-the-art software.

Accelerator is the only comprehensive modernization solution that can address both immediate tactical needs and strategic long-term modernization requirements with no limitations or proprietary lock-in. With Accelerator, you modernize your existing IBM i or z/OS systems and provide an evolutionary roadmap to a fully modernized solution through low risk, short, incremental releases.

Because Accelerator can provide significant function with standard generation and can use existing heritage applications without any changes, it is ideal for customers with large custom applications and customized heritage third-party applications.

Also, Accelerator helps with much more than modernization. It is a rapid application development solution with Windows, web, and mobile user experiences in an n-tier agile architecture with no limitations or proprietary lock-in. Whether modernizing heritage applications or creating new function, with Accelerator your developers can develop faster, with better results, and more function (Figure 4-11).

Figure 4-11 Modernized applications for Windows, web, and mobile

4.2.1 Business value

Accelerator is a comprehensive development solution that substantially accelerates the creation, modernization, delivery, and maintenance of high-quality, state-of-the-art software for companies of all sizes and in any industry.

Whether modernizing heritage applications or creating new functions, with Accelerator your developers can develop faster, with better results, and more function. They can deliver the software that you need at a lower cost and in less time.

Accelerator can help with these tasks:

- Write new software.
- Rewrite existing software.
- Extend existing software.
- Reuse and integrate with existing programs.
- Reuse and transform existing 5250/3270 “green screen” user interfaces.
- Rapidly deliver enterprise Windows, web, and mobile applications.
- Evolve heritage systems to new modern applications at your pace.
- Combine new, older, and other systems into one seamless unified user experience.
- Stop building more heritage applications and accumulating even more technical debt.
- Advance developer skills, learning the most modern technologies quickly.

- Use preferred practices for software design and architecture.
- Have superior application performance and reliability.
- Eliminate your change request backlog.

Top reasons CIOs select Accelerator

Surround Technologies surveyed customers about why they choose Accelerator for their development projects and boiled the results down to the following top 10 list:

1. Reduce software release times
2. Improve response times to change requests
3. Fast ongoing ROI and lower software TCO
4. No limitations and no proprietary lock-ins
5. Minimal risk and business disruption
6. Simplify the reuse of your existing heritage systems
7. Quickly adapt to changing technologies.
8. Increase software adoption and user productivity
9. Embedded Application Management, Monitoring, and Control
10. Decreased project backlog and improve delivery capacity

4.2.2 Solution overview

The Surround Technologies' mission is to substantially accelerate the creation, modernization, delivery, and maintenance of high quality, state-of-the-art software. This process is referred to as "Accelerated Software Development".

However, Surround Technologies did not set out to create a developer productivity tool. It set out to accelerate its own software development. It takes into account the entire Software Development Life cycle from beginning to end to make the development process predictable, fast, continual, iterative, and repeatable across many developers. It identifies patterns, builds out architecture from those patterns, and formulates frameworks from that architecture. It abstracts as much as possible into higher-level classes and automates what cannot be abstracted. It simplifies and defines the development process wherever possible. It supports servers and data sources anywhere they might be from the cloud to an offline device. Most importantly, everything it does follows open standards, never limits the developer's capabilities, and never locks them into any company.

Accelerated Software Development provides these benefits:

- The developers can focus on business development and not on the underlying technologies
- Reduction of the guessing, costly development errors, and overall total cost of the software ownership through enforced patterns and standards
- The foundational agile architecture helps keep pace with business and the ever changing technology shifts
- Capitalize on the software that is already built

To do this, Surround Technologies created the RADx3 Foundational Principles for software development and the Accelerator Software Development Solution to support those principals. With Accelerator, you do not need to compromise on time, quality, or scope. Surround Technologies calls it faster, better, and more. You get all 3 and deliver the greatest possible value for the ROI-based budget (Figure 4-12).

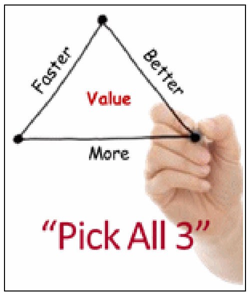


Figure 4-12 Time, quality, and scope

RADx3 foundational principles

Accelerator is based on the Surround Technologies RADx3 Foundational Principles for software development that drives faster development with better quality and more function. RADx3 is the comprehensive ability to build and grow agile systems productively.

It provides rapid application development. Surround Technologies calls it RAD1. Accelerator also delivers a robust architectural design, RAD2, which is a powerful n-tier SOA. You get rich, well-thought-out user experiences, accessible data integration, and interoperability that is called rich agile data (RAD3) (Figure 4-13).

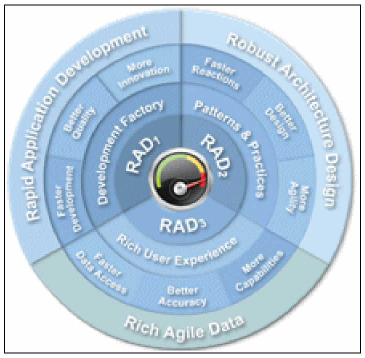


Figure 4-13 RADx3

RAD1: Rapid application development

Software development has always been mostly focused on the speed of delivery. In traditional development, that speed comes at the price of reduced IT control, inflexible software design, reduced function, poor code documentation, scattered code practices, and a lack of application monitoring and performance tuning. Essentially, quality and function are sacrificed for speed.

Therefore, productivity solutions for software engineers are a good idea, and so rapid application development (RAD) tools have been around for decades. However, these solutions are usually restrictive or incomplete. The results that you get are underwhelming and you spend more time to overcome limitations and dead ends.

To get true accelerated software development, you must be able to speed up development while you are delivering high-quality, robust software without sacrificing any of the necessary functions. This is Surround Technology's rapid application development, RAD1, foundational principle: To deliver software faster, with better quality and more function.

RAD2: Robust architectural design

Architecture is the great enabler of agile enterprise software.

Accelerator developed solutions garner power and agility from the Accelerator architecture. Because of it, the software that is built can connect to, integrate, and interoperate with most things. The software architecture is the primary reason that Accelerator-based solutions can easily adapt to the ever-changing business landscape and requirements. It helps create software that gets out of the user's way and enables them to productively do their daily tasks.

The software architecture enables developers to do more by allowing them to focus on business function, not the underlying technology. With Accelerator's agile architecture and consistent generated code, you get software that can be productively enhanced and adapted to changing business and technology needs over decades of life with a low total cost of ownership.

Figure 4-14 shows a high-level architecture that is accepted in the industry as one of the best and necessary approaches to software architecture. It is the basis of the Accelerator architecture. The figure provides a representation of the basic layers or tiers of the technology. It shows that they are separated but have communication between them. Accelerator applications are fully service enabled and the services can be both data and presentation. Also, there are framework services available across all layers. With Accelerator, you can provide your system-specific services and access to the underlying framework services, such as system configuration, properties, and security.

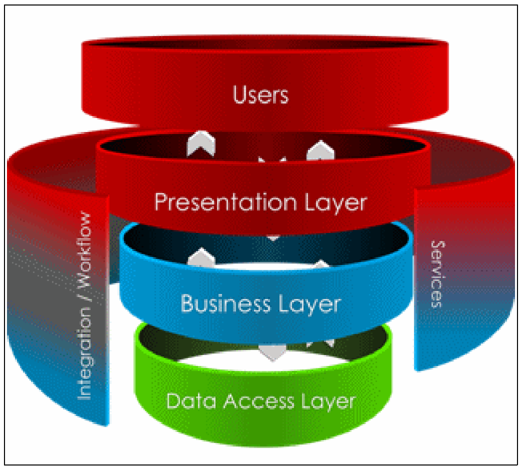


Figure 4-14 Industry accepted high-level architecture

Figure 4-15 provides more detail about the architecture details that are used with Accelerator. At the presentation level, it shows how users can access the system from multiple channels, with an integrated composite desktop application, from web browsers, and mobile applications. At the data level (the bottom of the diagram), it shows the architectural support for data access to sources that can include various databases, server platforms, third-party packages, and multiple clouds, in addition to access other systems and information through interfaces and service providers. The left of the diagram, the gray section, shows the different functions that are provided by the Accelerator services where the dynamic agility of the system is derived through operational metadata. The right side shows the services interface.

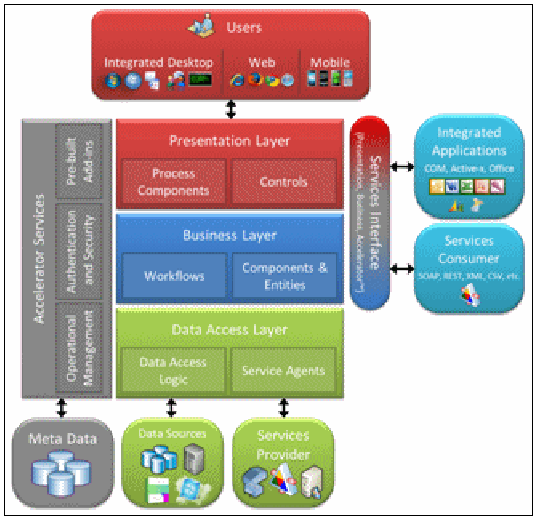


Figure 4-15 Accelerator high-level software architecture

The services interface provides secure access to system information and the ability to use the system's presentation layer. Web services are available in SOAP, REST, XML, CSV, and virtually any format that is necessary. The presentation services can be used within Microsoft Office products like Microsoft Outlook, Word, Excel, and PowerPoint in addition to MS Dynamics and SharePoint. It can also be used by third-party applications and included in web pages.

The Accelerator architecture is a loosely coupled, modular, dynamic (data driven), single underlying (standard, unified), scalable, high performing, n-tier, distributive, service-oriented architecture.

Figure 4-16 shows that Accelerator architecture provides a greater level of detail of the architecture. It shows the different components and how they relate to the various levels.

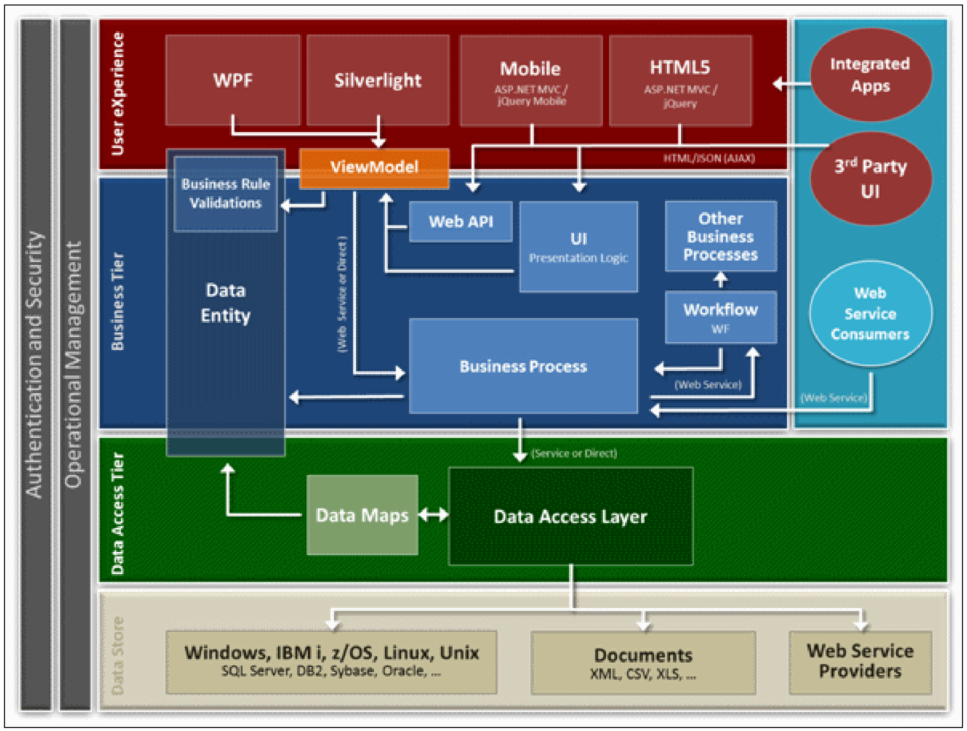
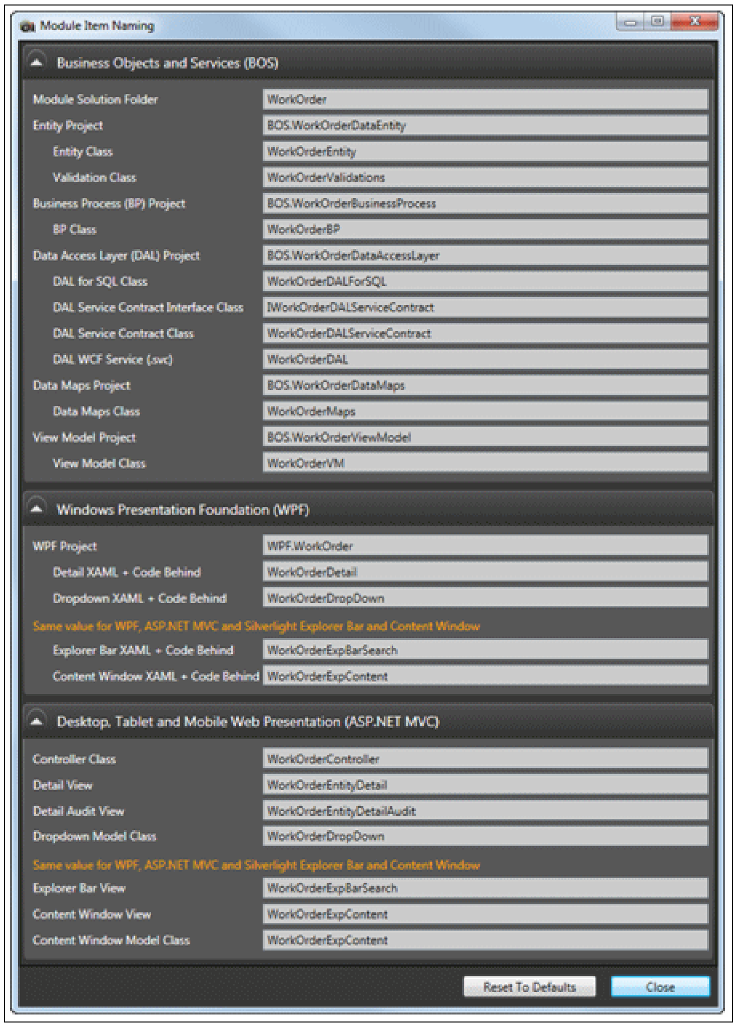


Figure 4-16 Accelerator architecture

The Accelerator Module Wizard generates all the components necessary, which significantly decreases development time and increases consistency and adherence to standards. Figure 4-17 shows the various items that can be generated for each module.



Business Objects and Services (BOS)

Module Solution Folder	WorkOrder
Entity Project	BOS.WorkOrderDataEntity
Entity Class	WorkOrderEntity
Validation Class	WorkOrderValidations
Business Process (BP) Project	BOS.WorkOrderBusinessProcess
BP Class	WorkOrderBP
Data Access Layer (DAL) Project	BOS.WorkOrderDataAccessLayer
DAL for SQL Class	WorkOrderDALForSQL
DAL Service Contract Interface Class	IWorkOrderDALServiceContract
DAL Service Contract Class	WorkOrderDALServiceContract
DAL WCF Service (.svc)	WorkOrderDAL
Data Maps Project	BOS.WorkOrderDataMaps
Data Maps Class	WorkOrderMaps
View Model Project	BOS.WorkOrderViewModel
View Model Class	WorkOrderVM

Windows Presentation Foundation (WPF)

WPF Project	WPF.WorkOrder
Detail XAML + Code Behind	WorkOrderDetail
Dropdown XAML + Code Behind	WorkOrderDropDown
Same value for WPF, ASP.NET MVC and Silverlight Explorer Bar and Content Window	
Explorer Bar XAML + Code Behind	WorkOrderExpBarSearch
Content Window XAML + Code Behind	WorkOrderExpContent

Desktop, Tablet and Mobile Web Presentation (ASP.NET MVC)

Controller Class	WorkOrderController
Detail View	WorkOrderEntityDetail
Detail Audit View	WorkOrderEntityDetailAudit
Dropdown Model Class	WorkOrderDropDown
Same value for WPF, ASP.NET MVC and Silverlight Explorer Bar and Content Window	
Explorer Bar View	WorkOrderExpBarSearch
Content Window View	WorkOrderExpContent
Content Window Model Class	WorkOrderExpContent

Figure 4-17 Module item naming

RAD3: Rich agile data

With Accelerator, you have your choice of user experiences, devices, interfaces, and platforms.

Superior user experience

Accelerator generates a comprehensive and superior User eXperience (UX), incorporating the top eight keys to a productive UX: Learnability, memorability, findability, discoverability, efficiency, accuracy, multitasking, and subjective user experience.

Many different presentation frameworks can be used such as Line of Business, Product Catalog, Wizard, Dashboard, Task Pane, Touch Screen, and Advanced Information Visualizations. Complete skin and theme support is also built into the interfaces, with several pre-built skins available.

Windows desktop applications are generated by using Windows Presentation Foundation, WPF. Web desktop and mobile interfaces are generated by using a single adaptive ASP.NET

MVC HTML5 design. Native mobile application can use Phone Gap or Xamarin to use the web interfaces or use the native Objective C for Apple or Java for Android (Figure 4-18).

Figure 4-18 Accelerator interfaces for Windows, web, and mobile

Application interfaces

There are no limits to how you can use your application interfaces. Most customers use the Accelerator user experience, but sometimes there are business needs to go beyond that. With Accelerator, you can create in part or in its entirety your own user experience with full integration to the user management, security, and other Accelerator Services. You can even enable other third-party applications to access the services for seamless integration with them. All of which can be monitored, managed, and controlled through the Accelerator Maintenance and Interface consoles.

Flexible interactions

Rich data is more than the user experience. It is also delivering information through integration and interfaces to other software and systems. With Accelerator, you can share data between applications on various platforms by using EDI, XML, CSV, web services and other interfaces, introduce new applications and technologies, and replace batch-style processing with real-time communication, all more efficiently and at a lower cost. You can quickly and securely build custom reports by using any reporting tool that can use XML including MS Reporting, MS Office (Word, Excel), Crystal, Altova, and so on.

Built-in service adapters provide the necessary interfaces to integrate with other applications, business partners, and customers. It is also possible to interoperate, or plug in, parts of your system into other software such as MS Office, SharePoint, and reporting solutions.

As with all of Accelerator, the back-end operational management, security, and authentication are built in.

Accelerator development solutions

Accelerator is a comprehensive end-to-end software development solution that drives faster development with better quality and more function. There are many frameworks with modern enterprise architecture, code generation wizards, software development tools, and a repeatable set of methodologies and practices that allow developers to continually deliver next-generation advanced agile software for their business.

The capabilities and high quality of the software, the speed in which it is delivered, and the ability for it to adapt to business changes all with no limitations and no lock-in is makes Accelerator different from other solutions.

The Accelerator Launchpad provides quick access to the various accelerator wizards and tools, all of which are open and highly configurable to adapt Accelerator to develop the way you want it to (Figure 4-19).

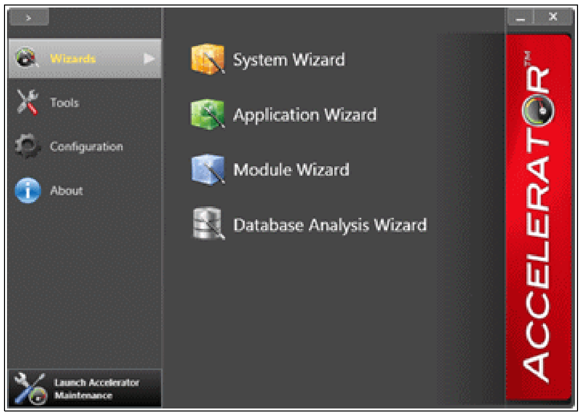
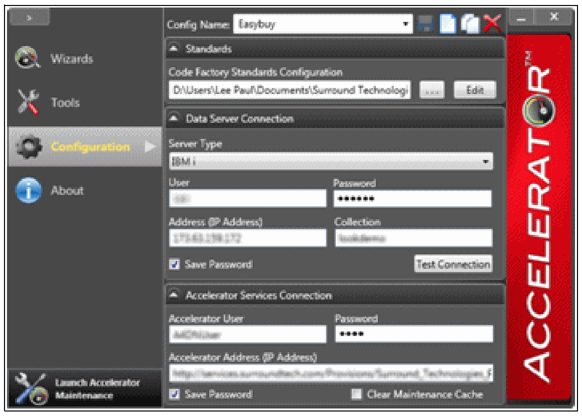


Figure 4-19 The Accelerator launchpad

Figure 4-20 shows the Accelerator configuration.



Config Name: Easybuy

Standards

Code Factory Standards Configuration: D:\Users\Lee Paul\Documents\Surround Technologi ... Edit

Data Server Connection

Server Type: IBM i

User: [redacted] Password: ••••••

Address (IP Address): [redacted] Collection: [redacted]

[✓] Save Password Test Connection

Accelerator Services Connection

Accelerator User: [redacted] Password: ••••

Accelerator Address (IP Address): [redacted]

[✓] Save Password [ ] Clear Maintenance Cache

Figure 4-20 Accelerator configuration

What makes Accelerator different is its completeness, which consists of four major parts plus the combined sum of those parts as a whole.

Core architecture and software

Accelerator provides the most important and critical part of an application, the core architecture and software. These components form the essential part of your software and define how it operates, performs, and adapts to support the needs of your business.

For many software projects, very little planning is put into how the software is designed and built. And, when it is, it is seldom done correctly the first or even second time. To do this process correctly takes experienced developers who know the potential pitfalls. They know how the decisions made now affect development throughout the project and throughout the life of the project.

Even when it is done correctly, the underlying architectural code is typically as much as 80% of the development for most software projects. It is the most costly part of software development by a significant margin, and it provides little business-related function. The users cannot see much of it, do not know that it exists, and do not care about it. With the Accelerator

Core Architecture and Software, the hardest, most important, and most expensive part of your software development is done correctly.

Accelerator developed solutions garner power and agility from the visionary core architecture that the Surround team has created within Accelerator. It acts as a technology abstraction layer that enables developers to do more. It does so by allowing them to focus on business function instead of the complexity of the underlying technology, with systems that behave in a homogeneous, unified, consistent, and mature manner.

To relate to the building metaphor, the Accelerator core is the footing, foundation, framing, plumbing, electrical, and communications. It can save you development time and related costs now and for many years to come. It can also minimize future maintenance costs and delivery time by providing robust software and eliminating technical debt that is incurred from rushed unplanned development.

Software factories and development accelerators

The Surround Technologies software factories and development accelerators are used to generate complete systems or individual applications, modules, and other various parts through easy-to-use blueprints, wizards, and development acceleration components. It generates the custom code that is specific to your database, provides 100% of your create, read, update, delete, and other base function, and allows developers to expand and enrich it for business-specific function.

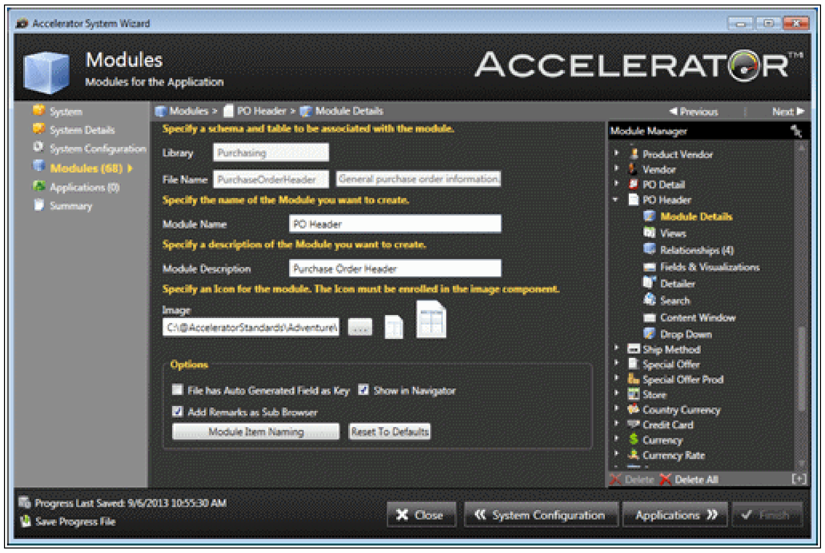
With Accelerator, the entire comprehensive base software is created for you. This base software is fully functional, near production ready, software that is generated without any coding.

Open Generation Technology (OGT)

The Surround Technologies acceleration tools are also open by using the powerful OGT. This technology allows the development team to configure their own development standards, define their own source in the generation templates, and specify generation rules and logic that is based on their database and software needs. With it, you can continually make your software generation smarter, more powerful, and specific to your business needs.

Software generation wizards

Accelerator wizards walk the developers through the process of data storage selection, module relationships, and presentation definition. The wizards then use various frameworks and code generators to produce fully functioning, high quality, state-of-the-art software with open, well-formed, and documented code (Figure 4-21).



Specify a schema and table to be associated with the module.

Library: Purchasing

File Name: PurchaseOrderHeader — General purchase order information.

Specify the name of the Module you want to create.

Module Name: PO Header

Specify a description of the Module you want to create.

Module Description: Purchase Order Header

Specify an Icon for the module. The Icon must be enrolled in the image component.

Image: C:\@AcceleratorStandards\Adventure\

Options

☐ File has Auto Generated Field as Key ☑ Show in Navigator

☑ Add Remarks as Sub Browser

Progress Last Saved: 9/6/2013 10:55:30 AM

Figure 4-21 Accelerator module definition

Customizable software code starts

The Accelerator loosely coupled modular architecture provides capabilities for other Accelerator-based or -wrapped software. It comes with a set of common cross-industry software that is already built and ready to be plugged into your systems.

It provides functions such as document management; date, time, and user-stamped remarks on anything in your system; and creating custom tables for reference, properties, and codes. These functions can also be customized or further enhanced to meet your specific needs.

Code Enhancer

Code Enhancer compares database and generated programs to analyze inconsistencies, find errors, and optionally update or fix the code. It analyzes added, changed, or removed fields and differences in types, lengths, and names. It updates existing generated code that is based on rejecting or accepting inconsistencies and can even update code that has already been customized by a developer.

Combining the core and the accelerators

With the combination of the core architecture and software with the software factories and development accelerators you get these items:

- Windows, web, and mobile user experiences
- Business Objects and Services (BOS)
- Line of Business CRUDx
- E-commerce product catalog
- Business wizards
- Operational and business intelligence dashboards
- Customizable task panes
- Efficient system navigation
- Multitasking enablement and management
- Advanced informational visualizations

- Theme and skinning
- Various data controls and visualizations
- Robust drop-down selections
- Powerful search and selection
- Quick adds
- Documents and media upload, download, and integration
- Microsoft Office (Word, Excel, Outlook) integration
- Google and Bing mapping integration

The Accelerator core architecture and software is the most important and valuable aspect of the solution. It is the cornerstone to what is called the advanced agile software. It is software that can adapt to the ever-changing business landscape and requirements. It is software that enables users to productively do their daily tasks. It is software that is built to last and built for change. The advanced agile software expected lifetime is measured in decades as it changes in response to new, unforeseen requirements.

Development process and methodologies

Accelerator provides rapid application development, RAD1, through advance automation techniques that are used in an agile process.

Figure 4-22 shows the Accelerator iterative rapid application development process. It illustrates the high-level process that developers go through to deliver software by using Accelerator. The left, gray section uses wizards and development tools to produce systems with little to no coding necessary. The gold section is where developers can customize the application.

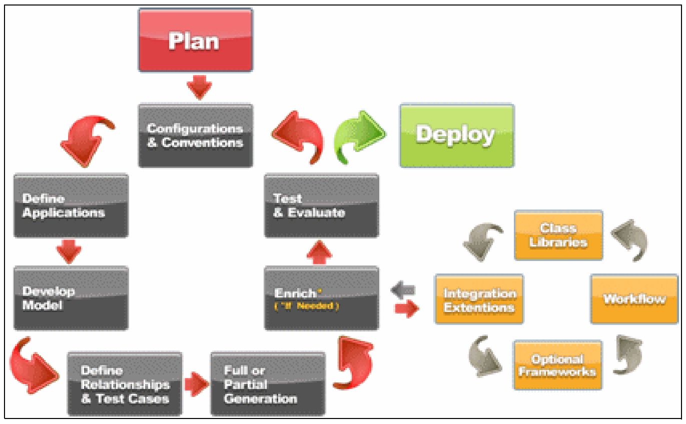


Figure 4-22 The Accelerator iterative rapid application development process

The typical development process consists of the following steps:

1. Plan and estimate. Accelerator's consistent, repeatable, iterative development process simplifies defining project deliverables and setting expectations. Accelerator eliminates technology challenges and provides comprehensive system function through fast software generation that enables architects and developers to focus on the business deliverables. This same generation can be used to create fully functional prototype to involve users and stakeholders early on in the project. Also, you can use the Accelerator analysis tools and predictable development activities to accurately size and estimate projects, minimizing missed tasks and unwanted surprises.
2. Generation. With Accelerator, the comprehensive base software is created for you. Accelerator wizards walk the developers through the process of data storage selection, module relationships, and presentation definition. The wizards then use various

frameworks and code generators to produce fully functioning, high quality, state-of-the-art software with open, well-formed, and documented code.

3. Enrich, extend, and integrate. Using the fully functional generated software as the foundational basis, developers can choose to unleash their genius and creativity to enrich and extend it further. Because there are no limitations, you can take the system as far as your creativity can take you and what your business needs are. Add dashboards, productivity task panes, configurable system options, document management, workflow wizards, advanced information visualizations, and much more. Use more Accelerator frameworks, code generators, tools, and pre-built add-ins to accelerate that process too.
4. Renew IBM i and zSeries Interfaces. With the renew option, developers can masterfully integrate existing IBM i 5250 and System z 3270 host screens into new more advanced composite software, repurposing the full function. Screen navigation and processes are done automatically and silently through background transactions. The business function host screens are presented in a graphically enhanced, dynamically derived version of the underlying 5250 or 3270 panel. The default dynamic refacing engine provides the graphical look for most screens without any intervention, and developers can enrich further to create highly customized graphical forms. Developers can orchestrate and reface the host screens into encapsulated modern snap-in business function and get application integration that combines every system you have into one seamless user experience.
5. Test. Focus testing on your system customizations. The core architecture and base generated system function can be extensively tested by Surround Technologies. Implement Test Driven Development (TDD) to further automate the testing and validation process.
6. Deploy. Use Accelerator tools to simplify deployments and integrate with other tools to establish a complete continuous integration practice.
7. Monitor, maintain, and enhance. Secure, monitor, control, and maintain your systems through the Accelerator Maintenance System (also built by using Accelerator) and continue to enhance it further using various Accelerator tools.
8. Repeat. Create new modules and add them to your existing systems or create complete systems with new modules that can also be shared with the other systems.

Figure 4-23 shows the business value that Accelerator provides throughout the full life of the software.

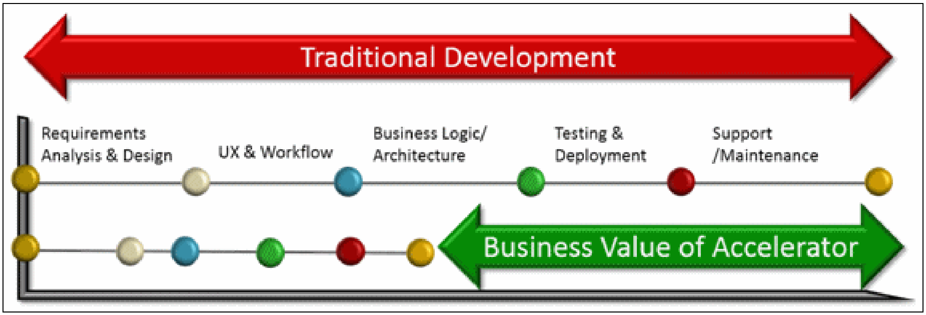

Traditional Development
Requirements Analysis & Design
UX & Workflow
Business Logic/ Architecture
Testing & Deployment
Support /Maintenance
Business Value of Accelerator

Figure 4-23 Accelerator productivity gains through the software development lifecycle

The *Accelerator Server* ties everything together.

The *Accelerator Core Run Time* pulls all the development and system metadata together to provide a high-performing, unified user experience in Windows, web, and mobile applications.

Accelerator services securely exposes all of the Accelerator configuration, operation, and administrative services to other applications where and when it is needed.

Operation Management and Administration is a complete system that is developed by using Accelerator itself. It provides full security, system management and administration, and system monitoring, control, and performance tuning. These administrative functions can also be plugged in and integrated with your own system, pulling in your information and providing even greater operational control.

Software Configuration services provide system, application, and module level information that is used by the presentation layer to visualize the system and deliver the dynamic, adaptive, robust user experience. You can configure system preferences and options: System navigation including menu navigation, favorite and recent modules and tabbed navigation, system and user options, skin and theme settings, logging, and user information.

4.2.3 Modernization

To Surround Technologies, modernization is only a part of software development in which the same RADx3 approach is applied to deliver the same faster and better results. Viewing software modernization as different from software development is a mistake and results in two possible approaches:

1. Purely tactical. Some solutions treat the symptoms while ignoring the root causes of the problem. They are targeted at solving just one or perhaps a couple of the top eight areas that define modern applications. This often results in adding even more heritage code, increasing technical debt and putting greater drag on your development. That is, more technology and code that slows down development and increases your development backlog. See Figure 4-24.

Figure 4-24 Purely tactical solutions

2. Purely strategic solutions. These solutions are “rip and replace” style solutions where you either purchase a commercial off the shelf (COTS) package or set out to completely rewrite your systems. Both approaches are tremendously expensive, have high risks, and a high level of failure. Both approaches also take a long time to implement, are taxing on the team to support current operations while implement new software, and are based on business needs that have changed since the start of the project. They slow down development and increase your development backlog.

You can change how you develop software with a process that evolves and adapts your heritage systems at your pace that is based on business needs. You can evolve and adapt with Accelerator.

Accelerator is the only comprehensive modernization solution that can address both immediate tactical needs and strategic long-term modernization requirements with no limitations or proprietary lock-in. With Accelerator, you modernize your existing IBM i or z/OS systems and provide an evolutionary roadmap to a fully modernized solution through low risk, short, incremental releases.

Because Accelerator can provide significant function with standard generation and can use older applications without any changes to it, it is ideal for customers with large custom applications and customized third-party applications.

Also, Accelerator helps with much more than just modernization. It is also the only rapid application development solutions with Windows, web, and mobile user experiences in an n-tier agile architecture with no limitations or proprietary lock-in.

With Accelerator, you surround your existing technology with new, modern technology. Then, based on business need, you use as is, modernize, or replace what you need when you need, retire unnecessary functions, or build new software when and where needed. You can do it all as part of a modern software development process that decreases your project backlog and keeps up with the needs of the business (Figure 4-25).

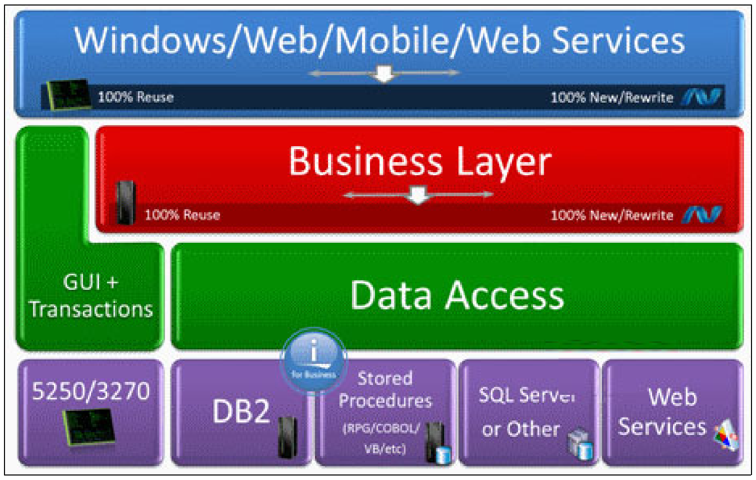


Figure 4-25 Accelerator architectural layers

Figure 4-26 shows a modernization process.

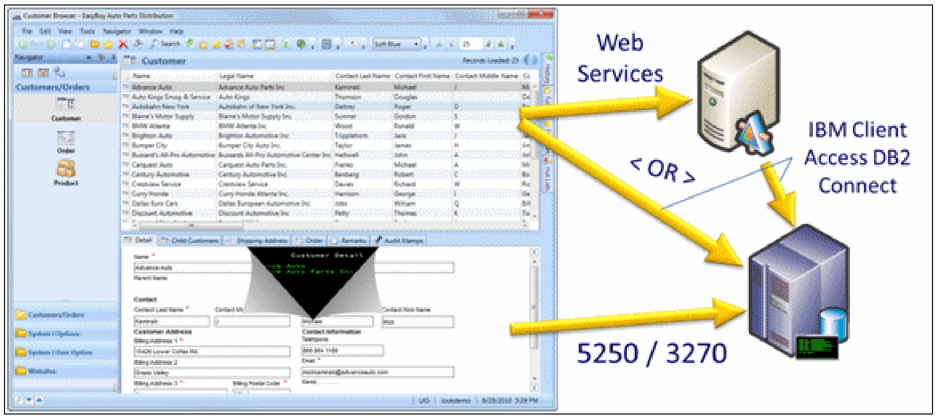


Figure 4-26 Modernization process

4.2.4 Usage scenarios

With Accelerator, there are numerous options from all new software to various levels of existing software reuse to deliver Windows, web, and mobile user interfaces in addition to providing web services all in a well-designed system and development process. There is no limitation and no lock-in to what you can produce, and you can build on top of your current systems without increasing your technical debt.

Create brand new software that is based on your new, enhanced, or current database, reuse your existing 5250 screens and RPG/COBOL code with all the benefits of the new modern architecture and interfaces, or both (Figure 4-27).

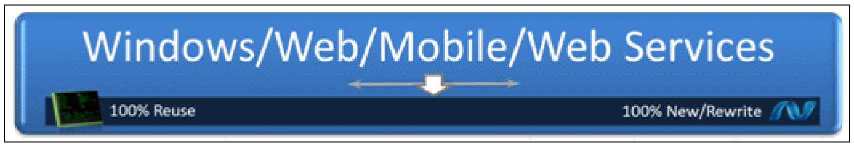


Figure 4-27 Reusing existing screens

Customers have taken the following paths:

1. Wrote new software first to facilitate an immediate need, then started modernizing their existing software with a mixture of new software, rewritten software, and transformed 5250 UI software.
2. Delivered a new modernized solution with little coding by using built-in generation and plugging in transformed green-screen UIs. Then, they added new functions such as dashboards and task wizards. They also started rewriting software to replace the transformed green-screen UI over time as they needed to add new function or provide greater productivity.
3. Rewrote a portion of their older software and kept their existing database and some back-end programs. After seeing how fast it was and the quality of what they could produce, they decided to rewrite it all, reusing only the database.
4. After seeing how fast they could produce brand new software and knowing what a large functional gap they had with their current database in key areas of the business, they focused their up-front time on designing a new database and generating prototypes to involve their users. Then, they built replacement software for that portion and modernized the large backend management system with either built-in generated programs or plugged in transformed green-screen UIs.

The following are typical uses of Accelerator:

- System maintenance with generation that is available for immediate use
- Prototyping with generation that is available for immediate use
- Write new or rewrite existing software
- Reuse current programs
- Transform existing 5250/3270 “green screen” user interface
  - Transactional reuse. Reuse transformed transactional screens.
  - Background transaction worker. Reuse screens in the background as a single transaction.
  - Background transaction batch. Reuse screens in the background as a batch job.
  - Specialized function, destination module. Reuse transformed work-with and transactional screens.
  - Split screen.
- Improve your ongoing current software development.
- Web services and integration
- Desktop applications
- Web applications
- Mobile applications

- Dashboards
- Task panes
- Task wizards
- Website plug-ins and integration
- Advanced information visualization
- Options
- Properties and codes
- Themes, skins, and branding

4.2.5 Supported platforms

IBM i system requirements:

- IBM i V5R4 or later. V6.1 or later is preferable.

Windows server requirements:

- This requirement depends on the size of the system that is running, the number of users, and what layers of the application are running on it. Essentially, the recommendation is the same as for any .NET developed system.
- Windows Server 2008 R2 SP1 Standard, X64 or later
- A minimum of dual core (as fast as you can), 32 GB RAM, and a 500 GB 7200 RPM hard disk. The specific requirements depend on the system that is being developed and the number of users who are accessing the system.
- IBM i Access for Windows .NET data provider (the license is a no-cost part of IBM i Access for Windows)

Development PC requirements:

- This is standard Visual Studio .NET development. Use recommendations for Visual Studio. As with all development, more power means more developer productivity.
- Windows XP or later. Windows 7 or later is preferable. 32/64 bit compatible.
- Development software
- Visual Studio 2010 SP1 Professional (or other editions) or later. Visual Studio 2012 or later is preferable.
- IBM i Access for Windows .NET data provider (License is a no-cost part of IBM i Access for Windows)

4.2.6 Ordering information

Accelerator Development Solutions is available through Surround Technologies and their authorized partners.

For more information and sales, contact Surround at:

Phone: (239) 405-8427

Email: `sales@surroundtech.com`

Web: `http://www.surroundtech.com/contactus/contactus.aspx`

Address:

9480 Corkscrew Palms Circle

Suite 4

Estero, FL 33928

4.2.7 Related information

For more information refer to the following websites:

- Surround Technologies website:

  http://www.surroundtech.com
- Accelerator Product Page:

  http://www.surroundtech.com/Accelerator
- IBM i modernization information site:

  http://www.imodernization.com

5

Source control and project management

This chapter describes source control and project management software. It includes the following sections:

- Collaborative lifecycle management for IBM i with ARCAD Pack for Rational and ARCAD-Verifier
- Solution architecture
- Usage scenarios
- Related information

5.1 Collaborative lifecycle management for IBM i with ARCAD Pack for Rational and ARCAD-Verifier

Thanks to the longevity of IBM i, any application development or modernization tooling requires specific support for its unique technical environment, capable of analyzing 20 - 30 year old heritage code and variants up to the latest object-oriented languages. The ARCAD Pack for Rational encapsulates this detailed IBM i knowledge into a set of solutions that are integrated with the IBM Rational offering for collaborative lifecycle management. These solutions complete the automation of dependency analysis, integration build, multi-platform deployment, and rollback.

Integrated with ARCAD Pack for Rational, ARCAD-Verifier automates regression test and unit test phases on IBM i, facilitating Continuous Integration and Test.

This section introduces both ARCAD Pack for Rational and ARCAD-Verifier, including their features, benefits, and architecture. This information is intended for IT professionals who need to understand the business value of these solutions.

For more information about the application modernization, see *Modernize IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185.

5.1.1 Contents

Tighter budgets, stiffer competition, and regulatory constraints are pressing IT departments to modernize not only applications but also their methods and tools. Heterogeneous web and mobile components have made application architectures more complex, spanning multiple systems, often with a heritage business application at the core. This mix of diverse cultures in IT shows the limitations of classic methodologies and makes efficient team communication and collaboration paramount.

Massive software changes must be delivered with increasing frequency and to higher levels of quality. These changes require modernization efforts across both development teams and IT operations to become more proficient in rolling changes into production, with a closer cooperation between the two. Therefore, both agile methods and DevOps are central to the modernization process, emphasizing pragmatism, collaboration, accountability, and a need for speed.

The overall IBM Rational offering for collaborative lifecycle management (built on the open IBM Jazz™ platform) provides a highly scalable and versatile solution that supports formal planning, agile team collaboration, and the DevOps lifecycle.

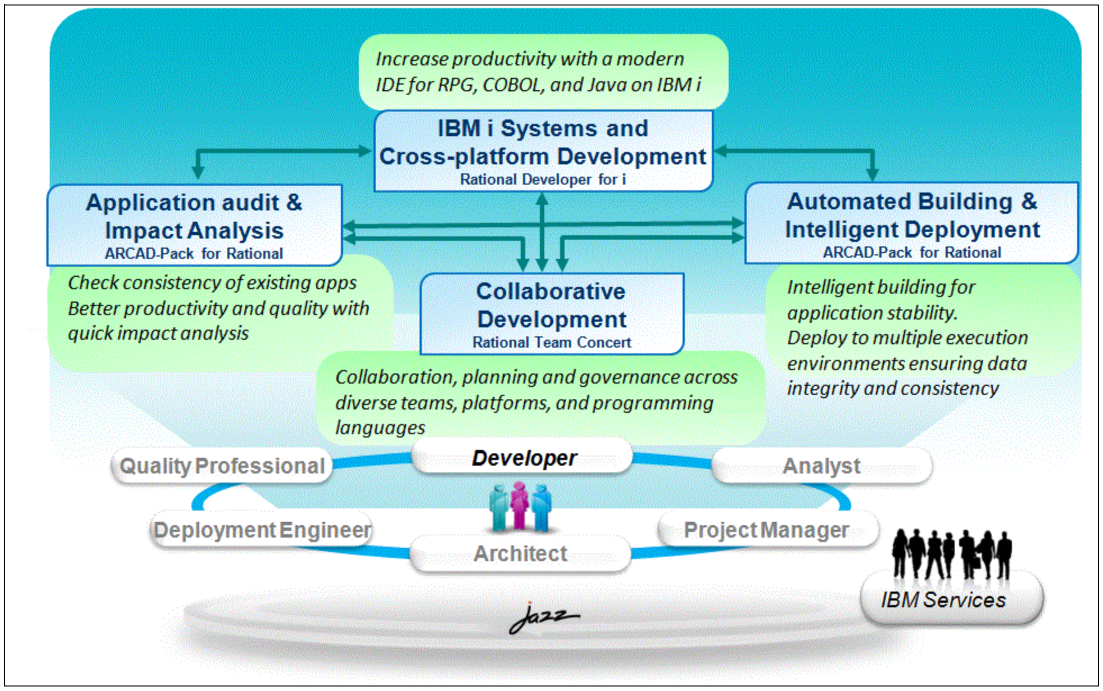
IBM Rational Team Concert is the application lifecycle management component of this collaborative lifecycle management offering. It is a common, collaborative software delivery environment to govern application development in multiple technologies, including IBM i, System z, and distributed environments such as Java and Microsoft .NET. By unifying technology cultures under the same toolset, Rational Team Concert simplifies project management and helps break down the silos that inhibit efficiency and change.

Integrated with Rational Team Concert for i, the ARCAD Pack for Rational provides the deep IBM i functionality that is essential for managing the many specialized technology variants on the platform. In particular, it completes the automation of dependency analysis, integration build, and deployment. This automation simplifies the change process and shortens time to delivery. It supports all IBM i languages and environments from 30-year-old heritage applications in COBOL or RPG III through to Free Form ILE and newer object-oriented code.

The combination of Rational Team Concert, Rational Developer for i (RDi), and ARCAD Pack for Rational contributes to breaking the divide between technologies and gives IBM i shops an enterprise capability for accelerated and continuous software delivery of high-quality software to customers and users.

With ARCAD Pack for Rational, teams that are working on multitier and application modernization projects are managed in a consistent way. The workflow and actions such as check-out, edit, compile, and check-in of changes, and association of changes with tasks or defects, are similar whether the source is RPG, COBOL, CL, EGL, C#, PHP or Java.

Figure 5-1 shows a functional view of the solution, which is a modern, productive work environment that is based on the standard Eclipse IDE. It makes IBM i easily accessible to all developers, from recent graduates to platform veterans.



Increase productivity with a modern IDE for RPG, COBOL, and Java on IBM i
IBM i Systems and Cross-platform Development
Rational Developer for i
Application audit & Impact Analysis
ARCAD-Pack for Rational
Automated Building & Intelligent Deployment
ARCAD-Pack for Rational
Check consistency of existing apps
Better productivity and quality with quick impact analysis
Collaborative Development
Rational Team Concert
Intelligent building for application stability.
Deploy to multiple execution environments ensuring data integrity and consistency
Collaboration, planning and governance across diverse teams, platforms, and programming languages
Quality Professional
Developer
Analyst
Deployment Engineer
Project Manager
Architect
IBM Services
Jazz

Figure 5-1 IBM integrated solution for Power Systems development: RDi, Rational Team Concert, and ARCAD Pack for Rational

5.1.2 Did you know?

IBM i has a highly specialized technical environment due in part to its longevity and the multiple language variants that must be supported. This specificity can isolate development teams and discourage new developers from moving to the platform. For Rational Team Concert to successfully federate web and IBM i developers into a common environment, a rich layer of IBM i-specific technology is needed to maximize the automation of routine tasks.

ARCAD Software has been building industry-leading IBM i development tools for over 20 years. This expertise encapsulated in the ARCAD Pack for Rational advances the pace of enterprise modernization while helping IBM i development organizations to maximize the return on their investments in Rational software and the IBM i platform.

5.1.3 Business value

Validated by IBM as Ready for Rational Software, ARCAD Pack for Rational is an industry-leading lifecycle management suite that extends Rational Team Concert and Rational Developer for IBM i and is designed to help deliver solutions for modernizing the IBM i world.

ARCAD Pack for Rational empowers IT staff with modern collaborative tools for agile and traditional development across all platforms and roles, which are tuned specifically to take advantage of IBM i technology.

ARCAD Pack for Rational focuses on five key functional areas, targeting productivity, reliability, and traceability across development and operations:

- Improved development intelligence with specialized analysis of IBM i application code and dependencies, maintained automatically in a multi-platform repository
- Automated conversion of RPGLE source code into Free Format RPG, easing the learning curve for new developers
- Accelerated code comprehension and impact analysis, with Eclipse-based visualization of application structure and dependencies
- Automated build of IBM i applications, using dependency knowledge for optimal sequencing of recompilations
- Automated and synchronous deployment of applications to multiple platform types, with complete traceability and rollback on error

Application modernization is an obligatory transition for all companies to retain their competitive standing and use the business value that is invested in development. Modernization projects uncover the valuable logic in your applications while providing the opportunities that present and future technologies offer. Many of the tasks that are involved can be automated. Highly specific tools such as ARCAD Pack for Rational rely on a repository of metadata or knowledge that is generated directly from the application for reference.

ARCAD solutions help automate the following modernization tasks, improving productivity and accuracy in the process:

- Managing multi-platform environments
- Moving to a new user interface
- Modernizing source code and database
- Moving to a modular architecture
- Creating web applications
- Expanding field size
- Conversion to Unicode
- Sharing skills

5.1.4 ARCAD-Audit: IBM i code audit and restructuring

ARCAD-Audit is used to normalize and sanitize heritage application libraries ready for a migration to Rational Team Concert and ARCAD Pack for Rational. It analyzes the contents of IBM i libraries, IFS directories, and source code to populate a metadata repository with knowledge of the inter-relationships between programs, files and databases, fields, and source lines.

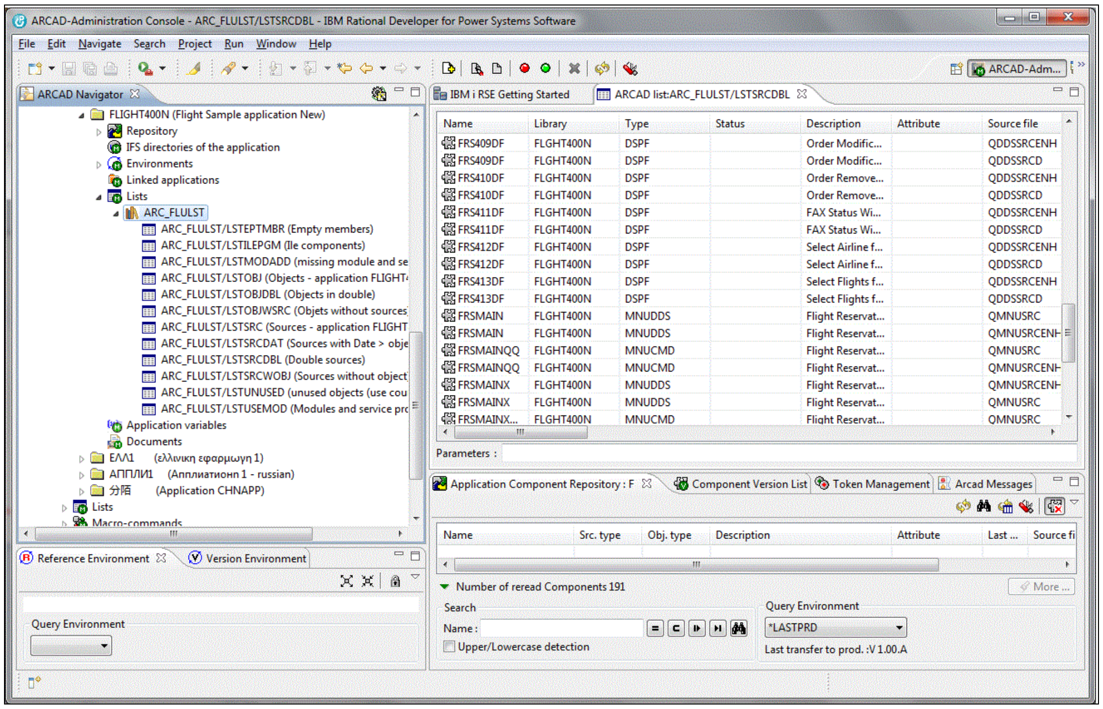
In particular, ARCAD-Audit solution is the first step in any application modernization project on IBM i. It helps use value from an existing application, supporting both the IBM i file system and integrated file system (IFS) on Windows, UNIX and Linux.

This analysis reveals any problems that might cause regressions in the future:

- Multiple occurrences of the same source
- Unused objects
- Source without objects (either object losses or unnecessary sources)
- Objects without source (this is a problem if you want to modify a related component)
- Source with a date later than the object (that is, components whose executed object is not at the same level as the source)

A rapid clean-up feature securely compares, archives, and deletes obsolete components and traces the actions that are taken.

The stored application knowledge is then used throughout the modernization project by the other modules in the ARCAD Pack for Rational, such as ARCAD-Observer, ARCAD-Builder, and ARCAD-Deliver (Figure 5-2).


ARCAD-Administration Console - ARC_FLULST/LSTSRCDBL - IBM Rational Developer for Power Systems Software
File Edit Navigate Search Project Run Window Help
ARCAD-Adm...
ARCAD Navigator
IBM i RSE Getting Started
ARCAD list:ARC_FLULST/LSTSRCDBL
Application Component Repository : F
Component Version List
Token Management
Arcad Messages
Reference Environment
Version Environment
More ...

FLIGHT400N (Flight Sample application New)
- Repository
- IFS directories of the application
- Environments
- Linked applications
- Lists
  - ARC_FLULST
    - ARC_FLULST/LSTEPTMBR (Empty members)
    - ARC_FLULST/LSTILEPGM (Ile components)
    - ARC_FLULST/LSTMODADD (missing module and se
    - ARC_FLULST/LSTOBJ (Objects - application FLIGHT
    - ARC_FLULST/LSTOBJDBL (Objects in double)
    - ARC_FLULST/LSTOBJWSRC (Objets without sources
    - ARC_FLULST/LSTSRC (Sources - application FLIGHT
    - ARC_FLULST/LSTSRCDAT (Sources with Date > obje
    - ARC_FLULST/LSTSRCDBL (Double sources)
    - ARC_FLULST/LSTSRCWOBJ (Sources without object
    - ARC_FLULST/LSTUNUSED (unused objects (use cou
    - ARC_FLULST/LSTUSEMOD (Modules and service pro
- Application variables
- Documents

ΕΛΛ1 (ελλινικη εφαρμωγη 1)
АППЛИ1 (Апплиатионн 1 - russian)
分陌 (Application CHNAPP)
Lists
Macro-commands

Name	Library	Type	Status	Description	Attribute	Source file
FRS409DF	FLGHT400N	DSPF		Order Modific...		QDDSSRCENH
FRS409DF	FLGHT400N	DSPF		Order Modific...		QDDSSRCD
FRS410DF	FLGHT400N	DSPF		Order Remove...		QDDSSRCENH
FRS410DF	FLGHT400N	DSPF		Order Remove...		QDDSSRCD
FRS411DF	FLGHT400N	DSPF		FAX Status Wi...		QDDSSRCENH
FRS411DF	FLGHT400N	DSPF		FAX Status Wi...		QDDSSRCD
FRS412DF	FLGHT400N	DSPF		Select Airline f...		QDDSSRCENH
FRS412DF	FLGHT400N	DSPF		Select Airline f...		QDDSSRCD
FRS413DF	FLGHT400N	DSPF		Select Flights f...		QDDSSRCENH
FRS413DF	FLGHT400N	DSPF		Select Flights f...		QDDSSRCD
FRSMAIN	FLGHT400N	MNUDDS		Flight Reservat...		QMNUSRC
FRSMAIN	FLGHT400N	MNUDDS		Flight Reservat...		QMNUSRCENH
FRSMAINQQ	FLGHT400N	MNUCMD		Flight Reservat...		QMNUSRC
FRSMAINQQ	FLGHT400N	MNUCMD		Flight Reservat...		QMNUSRCENH
FRSMAINX	FLGHT400N	MNUDDS		Flight Reservat...		QMNUSRCENH
FRSMAINX	FLGHT400N	MNUDDS		Flight Reservat...		QMNUSRC
FRSMAINX...	FLGHT400N	MNUCMD		Flight Reservat...		QMNUSRC

Parameters :

Name	Src. type	Obj. type	Description	Attribute	Last ...	Source fi

Number of reread Components 191

Search

Name :

Upper/Lowercase detection

Query Environment

*LASTPRD

Last transfer to prod. :V 1.00.A

Query Environment

Figure 5-2 List of application anomalies revealed by ARCAD-Audit

This video demonstrates the simple steps that are required to audit and clean software applications on the IBM i platform by using the ARCAD Pack for Rational before you load the software into Rational Team Concert:

https://jazz.net/library/video/1316

5.1.5 ARCAD-Observer: application analysis and retro-documentation

ARCAD-Observer provides features to describe and document existing applications.

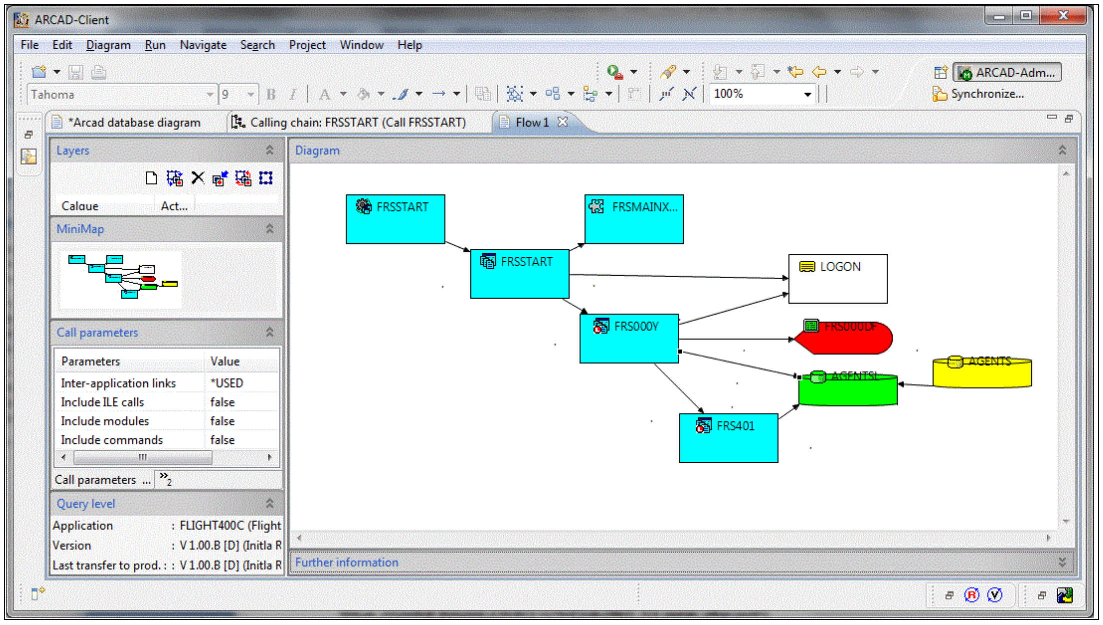
Application mining

ARCAD-Observer is an Eclipse-based application mining solution to reveal architecture and rules with a set of graphical tools. All software and data components that are associated with an application are analyzed, whether they are IBM i-based or are located on another platform, such as Windows, UNIX, or Linux. ARCAD-Observer can be used to inspect an application and discover interrelationships between components, and also automatically generate offline technical documentation in HTML format. The solution enables developers to sift through a vast quantity of diverse information and understand an application in seconds before making a software change.

ARCAD-Observer provides detailed impact analysis. This analysis references all dependencies between applications and components down to the field and source line level, whatever the language used (for example, COBOL, CL, SQL, and RPG III through Free Format and ILE). In addition, all cross-platform dependencies between IBM i components (programs, files, fields) and other non-native components (such as Java, PHP and C#) are traced and visible directly from the developer workbench.

With ARCAD-Observer, you can analyze the information flow across processes and quickly identify business rules that are contained in the code. Its built-in diagram editor displays information as graphics. You can easily enhance these diagrams and include them in your documentation.

ARCAD-Observer diagrams show how files relate to each other, how a set of programs accesses a group of files, and how modules in an information system interact. For example, the workflow diagram in Figure 5-3 combines file usage with program call information and actual values of call parameters.

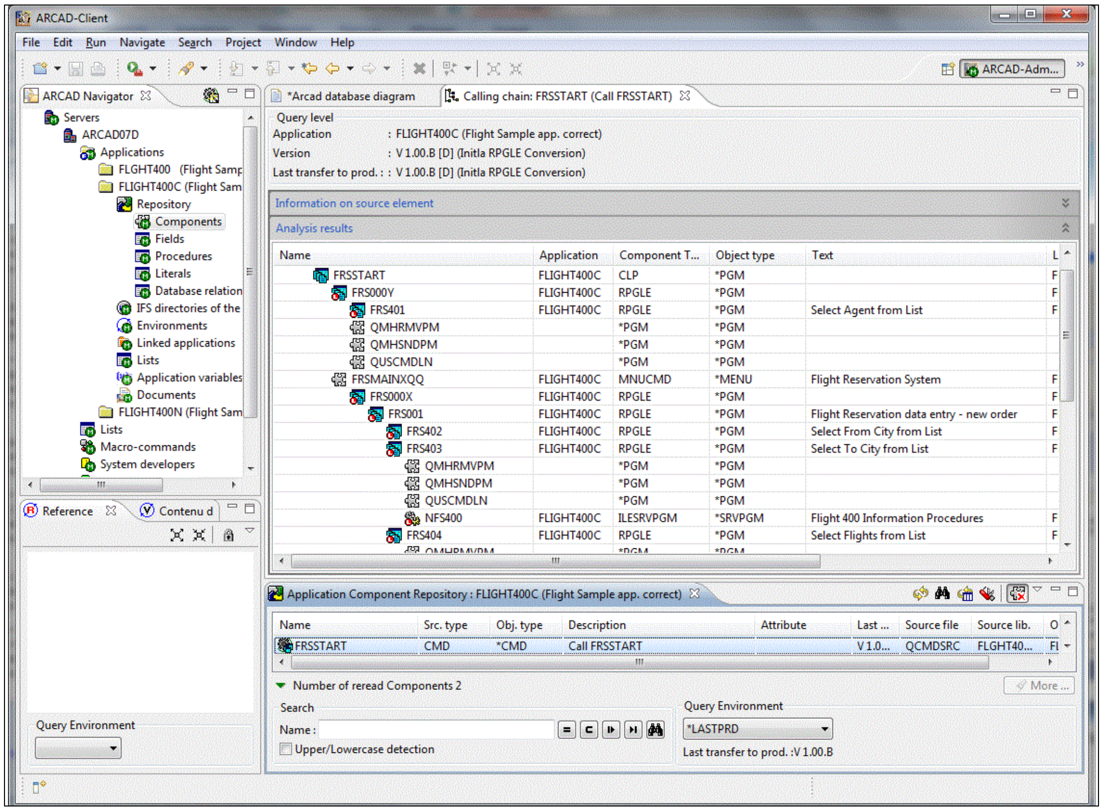


Parameters	Value
Inter-application links	*USED
Include ILE calls	false
Include modules	false
Include commands	false

Application : FLIGHT400C (Flight
Version : V 1.00.B [D] (Initla R
Last transfer to prod. : : V 1.00.B [D] (Initla R

Figure 5-3   Workflow diagram that is generated from ARCAD-Observer with call parameter values

ARCAD-Observer shows where a procedure is used, and where a field is updated, with a precise source line view. The full program and procedure calling chain can be displayed at any point in the interface, together with parameters that are passed (Figure 5-4).



Query level
Application : FLIGHT400C (Flight Sample app. correct)
Version : V 1.00.B [D] (Initla RPGLE Conversion)
Last transfer to prod. : : V 1.00.B [D] (Initla RPGLE Conversion)

Name	Application	Component T...	Object type	Text	L
FRSSTART	FLIGHT400C	CLP	*PGM		F
FRS000Y	FLIGHT400C	RPGLE	*PGM		F
FRS401	FLIGHT400C	RPGLE	*PGM	Select Agent from List	F
QMHRMVPM		*PGM	*PGM		
QMHSNDPM		*PGM	*PGM		
QUSCMDLN		*PGM	*PGM		
FRSMAINXQQ	FLIGHT400C	MNUCMD	*MENU	Flight Reservation System	F
FRS000X	FLIGHT400C	RPGLE	*PGM		F
FRS001	FLIGHT400C	RPGLE	*PGM	Flight Reservation data entry - new order	F
FRS402	FLIGHT400C	RPGLE	*PGM	Select From City from List	F
FRS403	FLIGHT400C	RPGLE	*PGM	Select To City from List	F
QMHRMVPM		*PGM	*PGM		
QMHSNDPM		*PGM	*PGM		
QUSCMDLN		*PGM	*PGM		
NFS400	FLIGHT400C	ILESRVPGM	*SRVPGM	Flight 400 Information Procedures	F
FRS404	FLIGHT400C	RPGLE	*PGM	Select Flights from List	F

Name	Src. type	Obj. type	Description	Attribute	Last ...	Source file	Source lib.	O
FRSSTART	CMD	*CMD	Call FRSSTART		V 1.0...	QCMDSRC	FLGHT40...	Fl

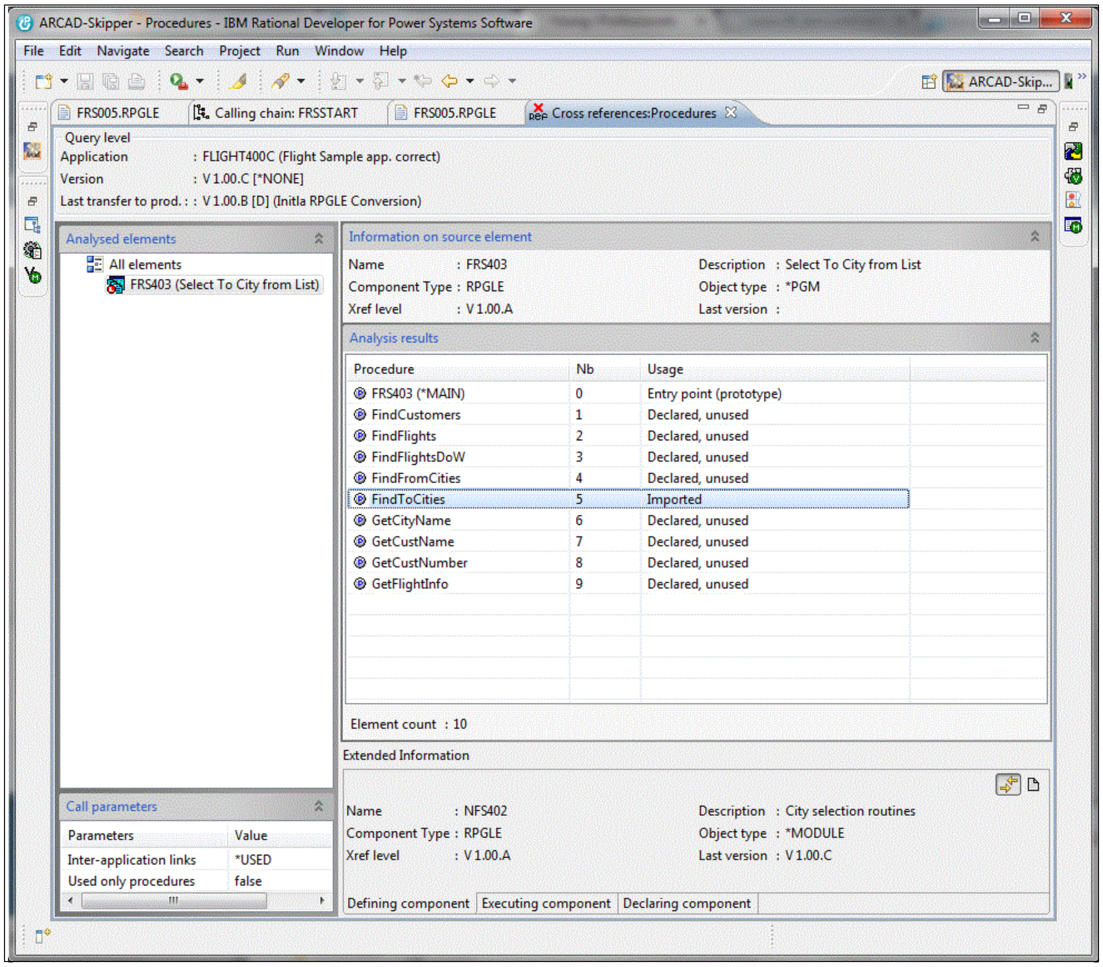
Number of reread Components 2

Search
Name :
Upper/Lowercase detection

Query Environment
*LASTPRD
Last transfer to prod. :V 1.00.B

Figure 5-4 Automatically generated diagrams from ARCAD-Observer (Example: Calling Chain)

In line with modern programming practices on IBM i, ARCAD-Observer provides powerful and transparent management of technologies such as ILE and SQL. Users are shielded from the complexity and can easily visualize how ILE programs are structured (Figure 5-5) and how files are accessed in embedded SQL.


ARCAD-Skipper - Procedures - IBM Rational Developer for Power Systems Software
File Edit Navigate Search Project Run Window Help
ARCAD-Skip...
FRS005.RPGLE
Calling chain: FRSSTART
FRS005.RPGLE
Cross references:Procedures
Analysed elements
All elements
FRS403 (Select To City from List)
Information on source element
Analysis results
Call parameters
Extended Information
Defining component
Executing component
Declaring component

Query level

Application	: FLIGHT400C (Flight Sample app. correct)
Version	: V 1.00.C [*NONE]
Last transfer to prod. :	: V 1.00.B [D] (Initla RPGLE Conversion)

Name	: FRS403	Description	: Select To City from List
Component Type	: RPGLE	Object type	: *PGM
Xref level	: V 1.00.A	Last version	:

Procedure	Nb	Usage
FRS403 (*MAIN)	0	Entry point (prototype)
FindCustomers	1	Declared, unused
FindFlights	2	Declared, unused
FindFlightsDoW	3	Declared, unused
FindFromCities	4	Declared, unused
FindToCities	5	Imported
GetCityName	6	Declared, unused
GetCustName	7	Declared, unused
GetCustNumber	8	Declared, unused
GetFlightInfo	9	Declared, unused

Element count : 10

Parameters	Value
Inter-application links	*USED
Used only procedures	false

Name	: NFS402	Description	: City selection routines
Component Type	: RPGLE	Object type	: *MODULE
Xref level	: V 1.00.A	Last version	: V 1.00.C

Figure 5-5 View of ILE procedure usage from ARCAD-Observer

ARCAD-Observer supports multi-level views. That is, the viewpoint for analysis and diagramming can be switched between version levels to ensure accurate reporting on the production environment, test environment, or a particular development branch.

Macroscopic views

When you have thousands of components in your information system, you need a tool to provide a global or "big" picture. For this requirement, ARCAD-Observer provides macroscopic views. With this feature, you can easily subdivide your applications into different functional domains, associate your components to those domains, and then view all dependencies at a functional level. Users obtain an overall map of an information system and can visualize dependencies between the different domains (Figure 5-6).

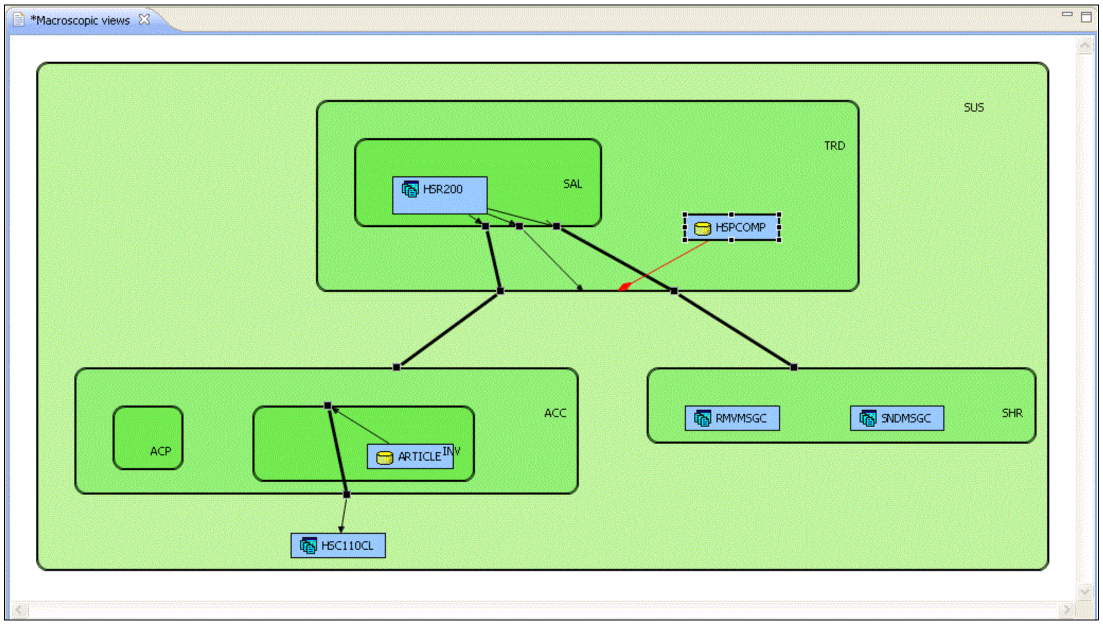


Figure 5-6 Automatically generated diagrams from ARCAD-Observer (Example: Macroscopic and Functional View)

Isolation of business logic

ARCAD-Observer is valuable when you are modernizing heritage systems for use on the web. The solution analyzes application libraries and source code to uncover critical business processes and interdependencies. It separates and documents critical business logic so that the gathered knowledge or metadata can be reused in new design and development efforts.

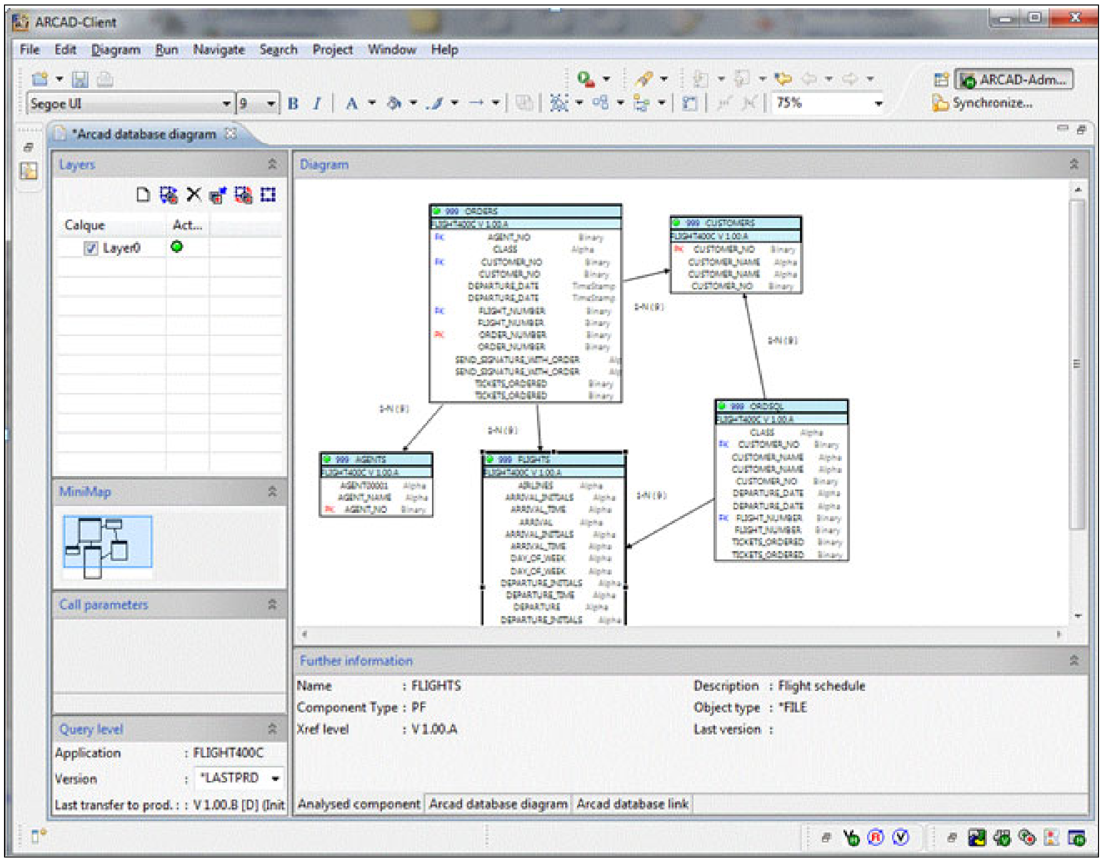
Large, monolithic applications can be analyzed to reveal the lines of similar source that can be combined into a method, see the code that is no longer callable, and visually understand how various other aspects of the code interact together. For re-engineering purposes, business rules are easy to identify and code areas can be selected for provision as web services. Source code can be more easily re-engineered into small inter-connectable, reusable procedures.

ARCAD-Observer also helps evolve and develop new applications around an existing system. It rapidly builds a global view of your application that can be navigated interactively to help understand heritage applications quickly and efficiently.

Database re-engineering

ARCAD-Observer is able to reverse-engineer the database relationship model that underlies an application, even in the case of flat files, generating SQL (DDL) if required. To do so, it uses a system of rules to analyze the program source code and identify complex database relations.

Multiple data exploration techniques are used to discover the database relationship model, adapting equally well to heritage applications without any declared keys and constraints and also to new database models. ARCAD-Observer uses name syntax, type and length information, and reference file constraints during the analysis process. A powerful discovery engine identifies foreign keys by parsing programs and propagating field content over the program calling chain until another file or table is reached (Figure 5-7).



Application : FLIGHT400C
Version : *LASTPRD
Last transfer to prod. : : V 1.00.B [D] (Init

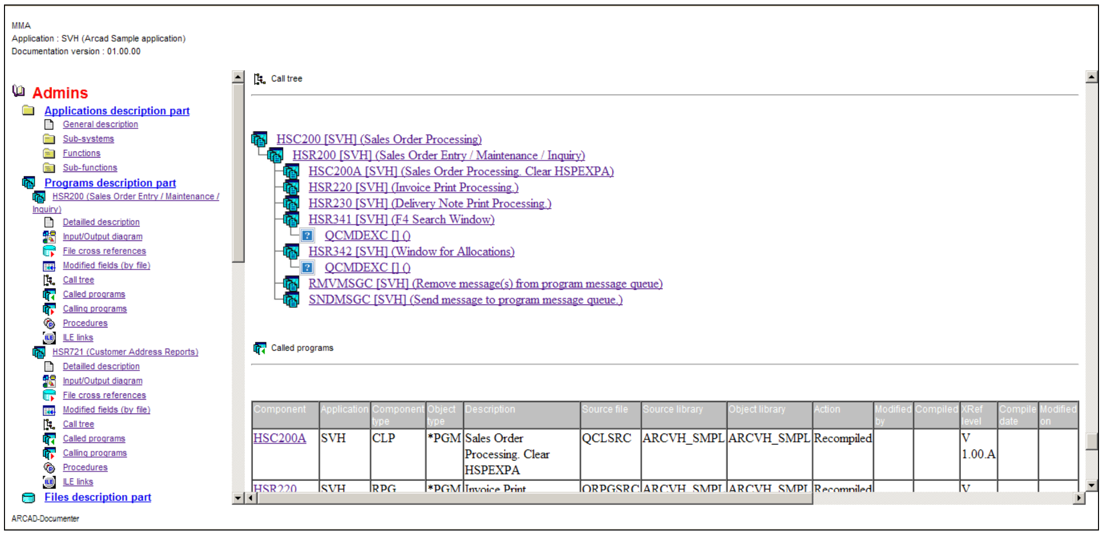
Name : FLIGHTS
Component Type : PF
Xref level : V 1.00.A
Description : Flight schedule
Object type : *FILE
Last version :

Figure 5-7 Reverse-engineering of database relationship model with ARCAD-Observer

Retro-documentation of applications

Access to and distribution of knowledge on the existing system is also critical in any modernization project. If you discover that your application documentation is obsolete or even non-existent, ARCAD-Observer offers an immediate solution that fulfills regulatory requirements (Sarbanes-Oxley, FDA, ISO 9000, and so on).

ARCAD-Observer automatically generates technical documentation for an application, according to chapter definitions that are easily customized for the target audience (for example, development teams and auditors). You can enhance the automatically generated documentation by adding text or customizing diagrams by using the documentation editor (Figure 5-8). The standard export format is HTML for ready distribution to anyone with a browser. An update procedure can be defined to ensure that documentation is automatically kept up to date with application changes.


MMA
Application : SVH (Arcad Sample application)
Documentation version : 01.00.00

Admins
- **Applications description part**
  - General description
  - Sub-systems
  - Functions
  - Sub-functions
- **Programs description part**
  - HSR200 (Sales Order Entry / Maintenance / Inquiry)
    - Detailled description
    - Input/Output diagram
    - File cross references
    - Modified fields (by file)
    - Call tree
    - Called programs
    - Calling programs
    - Procedures
    - ILE links
  - HSR721 (Customer Address Reports)
    - Detailled description
    - Input/Output diagram
    - File cross references
    - Modified fields (by file)
    - Call tree
    - Called programs
    - Calling programs
    - Procedures
    - ILE links
- **Files description part**

ARCAD-Documenter

Call tree

```
HSC200 [SVH] (Sales Order Processing)
  HSR200 [SVH] (Sales Order Entry / Maintenance / Inquiry)
    HSC200A [SVH] (Sales Order Processing. Clear HSPEXPA)
    HSR220 [SVH] (Invoice Print Processing.)
    HSR230 [SVH] (Delivery Note Print Processing.)
    HSR341 [SVH] (F4 Search Window)
      QCMDEXC [] ()
    HSR342 [SVH] (Window for Allocations)
      QCMDEXC [] ()
    RMVMSGC [SVH] (Remove message(s) from program message queue)
    SNDMSGC [SVH] (Send message to program message queue.)
```

Called programs

Component	Application	Component type	Object type	Description	Source file	Source library	Object library	Action	Modified by	Compiled	XRef level	Compile date	Modified on
HSC200A	SVH	CLP	*PGM	Sales Order Processing. Clear HSPEXPA	QCLSRC	ARCVH_SMPL	ARCVH_SMPL	Recompiled			V 1.00.A		
HSR220	SVH	RPG	*PGM	Invoice Print	QRPGSRC	ARCVH_SMPL	ARCVH_SMPL	Recompiled			V		

Figure 5-8 Auto-generation of technical application documentation, in HTML format as website

This video demonstrates the simple steps that are required when you are developing for IBM i to analyze dependencies by using the ARCAD Pack for Rational together with Rational Team Concert and Rational Developer for i:

https://jazz.net/library/video/1319

5.1.6 ARCAD-Builder: automated integration build

ARCAD-Builder reduces the effort that is required to recompile a full application by automating the build of any type of IBM i component. It uses the same dependency knowledge as ARCAD-Observer to optimize the build sequence and ensure the integrity of the executable code.

Its primary benefit in application modernization is in advanced support for newer and specialized technologies such as ILE and SQL. ARCAD-Builder fully automates recompiling of dependent components in a single operation, shielding complexity from the developer.

ARCAD-Builder starts by creating a set of self-consistent objects that comprise an application release. It takes into account all the dependencies among components, including /COPY, inclusions of file definitions (data description specifications or DDS), files, file programs, service programs, and data. It also manages changes in database structure and optimizes the reconstruction of indexes.

ARCAD-Builder provides full automation, repeatability, and traceability during these tasks:

- Management of compilation specifics
- Automatic handling of pre- and post-compilation commands
- Preservation of attributes, rights, and object ownership
- Automatic save and restore of data into a new file structure (database recovery)

- Optimal sequencing of recompilations of dependent components
- Automatic compilation of SQL-based objects and their dependencies
- Builds of DDL sources
- Full support for specifics of ILE compilations
- Traceability at the process and object levels
- Invocation directly from Rational Team Concert as part of the build phase

Figure 5-9 shows the build option and automatic dependency recompilation provided by ARCAD-Builder.

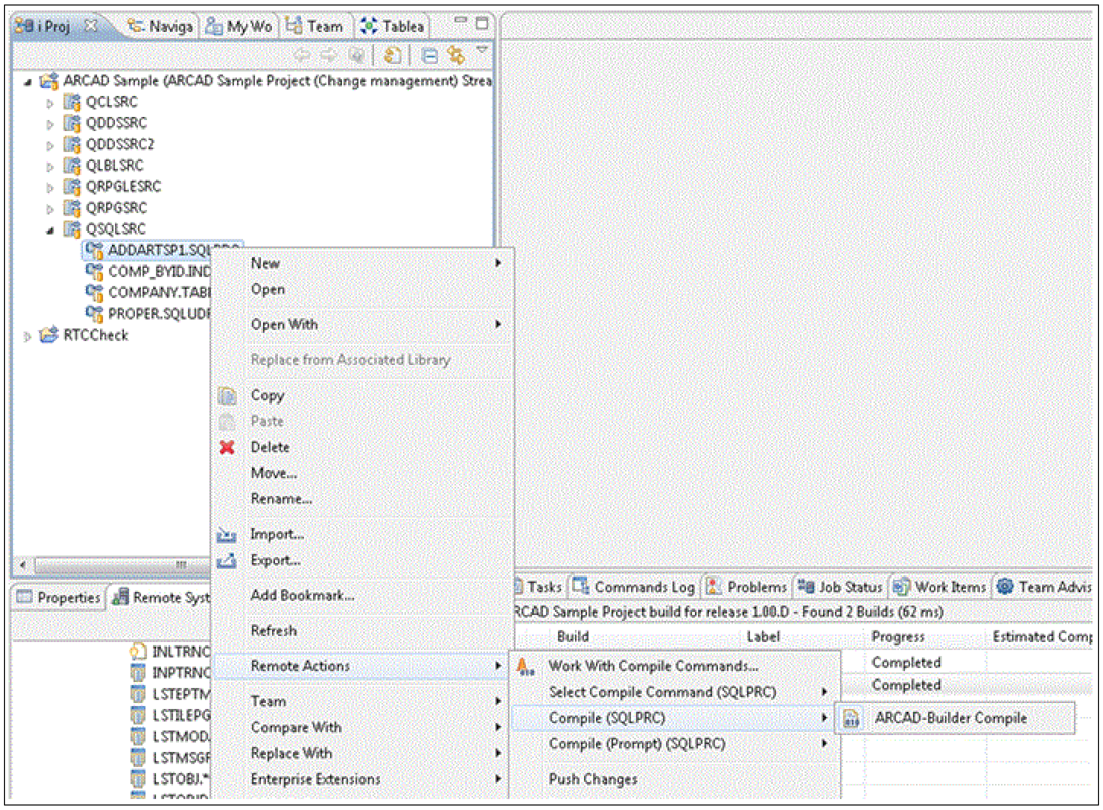


Figure 5-9 Build option and automatic dependency recompilation from ARCAD-Builder

ARCAD-Builder allows developers to view and modify attributes and specific commands directly in the source code. This information is stored as comments. This process is achieved without any special options to save developer time and reduce the risk of errors.

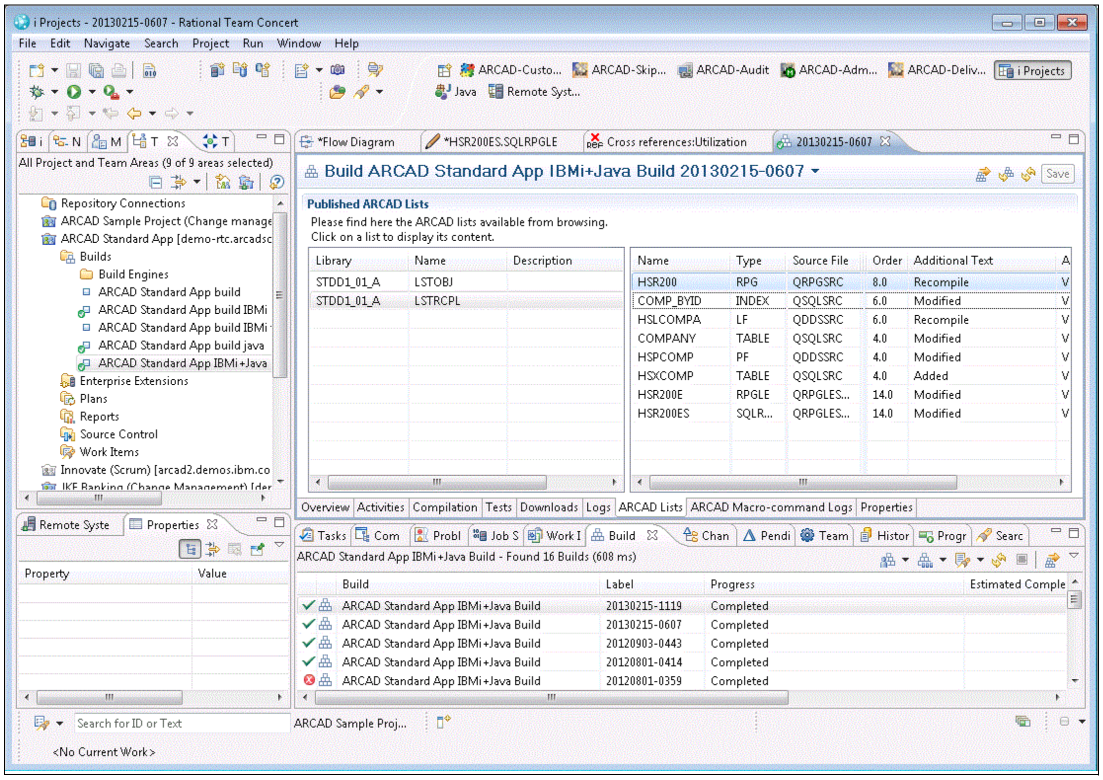
ARCAD-Builder automatically handles the intricacies of ILE, including:

- Recompilation of programs and service programs by using a module
- Dynamic signature management
- Dynamic ordering of sets of programs when procedures are used.

For developers who are using SQL, referential constraints also require files to be compiled in a precise order. Because of its detailed knowledge of SQL dependencies (including tables, views, and indexes down to the field level), ARCAD-Builder can fully automate the recompilation. It can build of all SQL-based objects and DDL sources, taking the burden off the developer.

ARCAD-Builder is generally used in Rational Team Concert build configurations to reduce the duration of the build phase and improve its accuracy. The solution is also well-adapted for integration with Jenkins and BuildForge. This is important in an agile and continuous integration (CI) environment where integration builds are frequent (ideally after each check-in operation) to test and detect errors as early as possible in the development cycle.

Figure 5-10 shows the automated sequencing and dependency in ARCAD-Builder.



Build ARCAD Standard App IBMi+Java Build 20130215-0607

Published ARCAD Lists

Please find here the ARCAD lists available from browsing.
Click on a list to display its content.

Library	Name	Description
STDD1_01_A	LSTOBJ	
STDD1_01_A	LSTRCPL	

Name	Type	Source File	Order	Additional Text	A
HSR200	RPG	QRPGSRC	8.0	Recompile	V
COMP_BYID	INDEX	QSQLSRC	6.0	Modified	V
HSLCOMPA	LF	QDDSSRC	6.0	Recompile	V
COMPANY	TABLE	QSQLSRC	4.0	Modified	V
HSPCOMP	PF	QDDSSRC	4.0	Modified	V
HSXCOMP	TABLE	QSQLSRC	4.0	Added	V
HSR200E	RPGLE	QRPGLES...	14.0	Modified	V
HSR200ES	SQLR...	QRPGLES...	14.0	Modified	V

ARCAD Standard App IBMi+Java Build - Found 16 Builds (608 ms)

	Build	Label	Progress	Estimated Comple
✓	ARCAD Standard App IBMi+Java Build	20130215-1119	Completed	
✓	ARCAD Standard App IBMi+Java Build	20130215-0607	Completed	
✓	ARCAD Standard App IBMi+Java Build	20120903-0443	Completed	
✓	ARCAD Standard App IBMi+Java Build	20120801-0414	Completed	
⊗	ARCAD Standard App IBMi+Java Build	20120801-0359	Completed	

Figure 5-10 Automated sequencing and dependency build of artifacts in ARCAD-Builder

This video demonstrates the simple steps that are required when developing for IBM i to check the effects of a change and run a build that includes all dependencies using the ARCAD Pack for Rational together with Rational Team Concert and Rational Developer for i:

`https://jazz.net/library/video/1317`

5.1.7 ARCAD-Deliver: Automated, multi-platform Deployment and Rollback

Full automation of application deployment requires the coordinated, secure, synchronous transfer of objects for all technologies and platforms. Most importantly, it requires use of the same methods, rules, and products to deploy (or roll back) all your applications, whatever the target system.

ARCAD-Deliver, the final release management component of ARCAD Pack for Rational, meets these requirements by synchronizing application deployment across multiple platform types (UNIX, AIX, Linux, Windows, and IBM i) within a single process. Deployment is managed and traced from a central console, with automatic rollback on error in case of failure on any target platform (Figure 5-11).

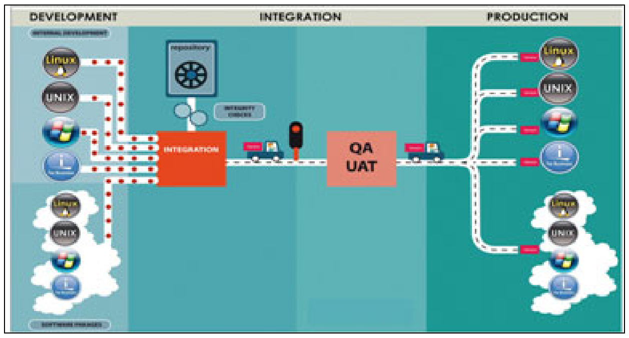


Figure 5-11 Channeling of multi-platform artifacts into a single ARCAD-Deliver process, for delivery to target platforms

This open, multi-platform support makes ARCAD-Deliver a practical choice in application modernization projects where the integrity of an application in production depends on diverse, inter-dependent artifacts such as RPG, CL, SQL, Java, .NET, and PHP being deployed simultaneously, securely and at the correct version level.

ARCAD-Deliver reduces complexity in IBM i environments by automatically running repository-based integrity checks on multi-platform artifacts to detect consistency errors before deployment. For example, it ensures that all dependent artifacts are present in the release and have been successfully built.

The transfer protocol used can be FTP, Secure FTP, SFTP (over SSH), or the ARCAD-internal SSL-secured protocol.

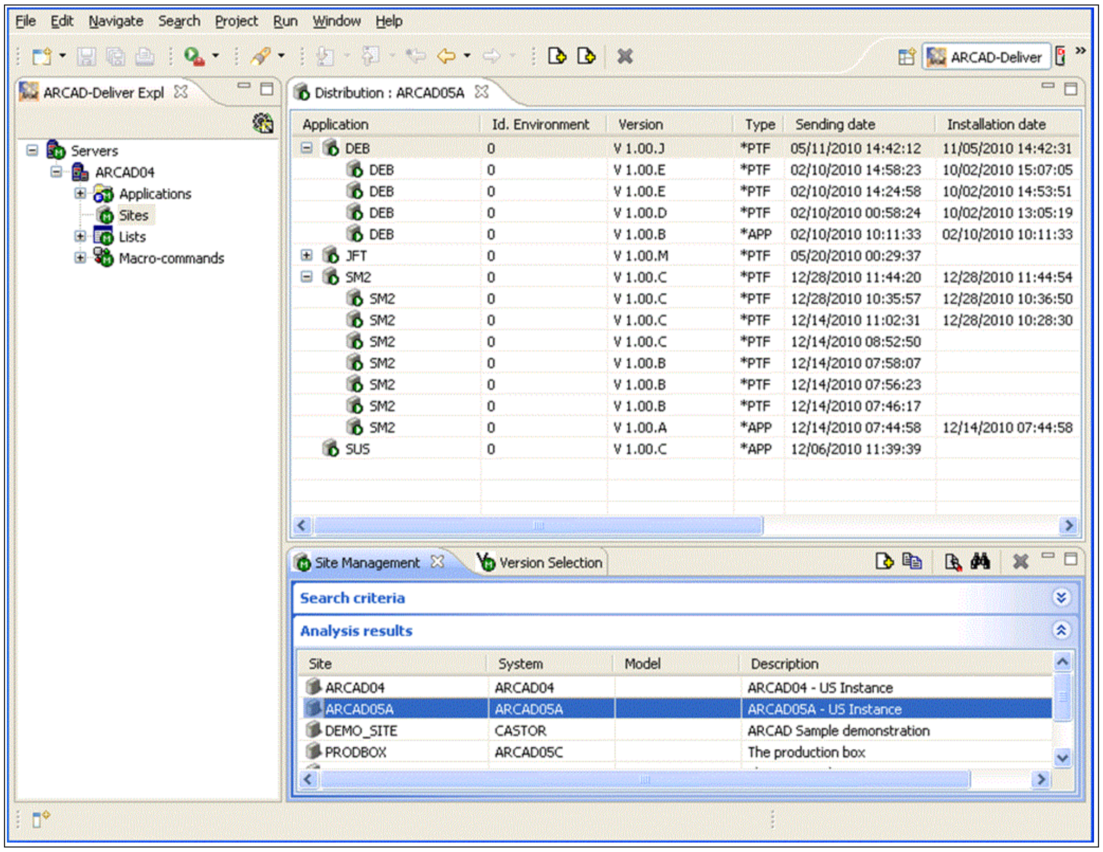
ARCAD-Deliver has an open design using the portable standard ANT as process manager. The delivery agent that runs ANT scripts can therefore be used with multiple technologies such as Java, web servers, application servers, Windows, Linux, UNIX, and IBM i servers. It is easy to customize according to local installation requirements on the target system.

ARCAD-Deliver automates packaging, transfer, and even remote installation (such as on Tomcat/WebSphere Application Server) from a central console with history and rollback on error. The product is bundled with a library of predefined ANT scripts (or Jython in the case of WebSphere) that can be used as received or customized.

ARCAD-Deliver facilitates coordination between Development and Operations, by automating these tasks:

- Site and system management
- Environment management
- Application version monitoring
- Deployment process monitoring
- Traceability at the application and environment levels
- Traceability at the deployed source code/objects level
- Customized deployment process support (at the version level)
- Database deployment optimization
- Overall (multiplatform) deployment transactions
- Deployment transaction rollback

Traceability is maintained at multiple levels to facilitate transparency and tracking of changes. For example, independently of the system log, specific logs are kept for applications, environments, sites, sources and objects, and versions and processes (Figure 5-12).



Application	Id. Environment	Version	Type	Sending date	Installation date
DEB	0	V 1.00.J	*PTF	05/11/2010 14:42:12	11/05/2010 14:42:31
DEB	0	V 1.00.E	*PTF	02/10/2010 14:58:23	10/02/2010 15:07:05
DEB	0	V 1.00.E	*PTF	02/10/2010 14:24:58	10/02/2010 14:53:51
DEB	0	V 1.00.D	*PTF	02/10/2010 00:58:24	10/02/2010 13:05:19
DEB	0	V 1.00.B	*APP	02/10/2010 10:11:33	02/10/2010 10:11:33
JFT	0	V 1.00.M	*PTF	05/20/2010 00:29:37	
SM2	0	V 1.00.C	*PTF	12/28/2010 11:44:20	12/28/2010 11:44:54
SM2	0	V 1.00.C	*PTF	12/28/2010 10:35:57	12/28/2010 10:36:50
SM2	0	V 1.00.C	*PTF	12/14/2010 11:02:31	12/28/2010 10:28:30
SM2	0	V 1.00.C	*PTF	12/14/2010 08:52:50	
SM2	0	V 1.00.B	*PTF	12/14/2010 07:58:07	
SM2	0	V 1.00.B	*PTF	12/14/2010 07:56:23	
SM2	0	V 1.00.B	*PTF	12/14/2010 07:46:17	
SM2	0	V 1.00.A	*APP	12/14/2010 07:44:58	12/14/2010 07:44:58
SUS	0	V 1.00.C	*APP	12/06/2010 11:39:39	

Site	System	Model	Description
ARCAD04	ARCAD04		ARCAD04 - US Instance
ARCAD05A	ARCAD05A		ARCAD05A - US Instance
DEMO_SITE	CASTOR		ARCAD Sample demonstration
PRODBOX	ARCAD05C		The production box

Figure 5-12 Transfer history log at the environment level, from ARCAD-Deliver

The open design of ARCAD-Deliver means that it can be used with any (or several) third-party source code control systems. Although a full migration to a common application lifecycle management solution such as Rational Team Concert is a worthy goal, it can be a long and often incremental process. The process can leave certain teams on established point solutions such as SubVersion (SVN) or Visual Source Safe. Here ARCAD-Deliver manages the entire downstream phase, pulling the artifacts to deploy from each individual source management tool, and running a cross-platform integrity check before deploying all artifacts synchronously to the target environment.

To optimize the handing of database modifications, ARCAD-Deliver supports a transfer in optimized mode by using ALTER or CHGPF to minimize downtime in production. It manages dynamic constraints, and referential integrity constraints are efficiently handled as a single unit by using a global view of the transfer.

In an agile approach, ARCAD-Deliver ensures continuous deployment by deploying successful builds automatically. For example, when a build is successful on a developer's workstation, changes can be automatically deployed to the project integration environment. This process can also trigger an automated deployment into another environment, and so on. Therefore, a developer's code is continuously integrated with the rest of the team's code, ready for automated testing (with ARCAD-Verifier, for example), and reducing the time to customer feedback.

This video demonstrates the simple steps that are required to do an intelligent deployment of software to the IBM i platform by using the ARCAD Pack for Rational together with Rational Team Concert:

https://jazz.net/library/video/1318

5.1.8 ARCAD Pack for Rational CASE and 4GL support: Manage CASE and 4GL environments alongside other languages

The optional ARCAD Pack for Rational CASE and 4GL support integrates CASE and 4GL environments into the standard ARCAD Pack for Rational and Rational Team Concert solutions. For example, specific support is provided for CA 2E (Synon), LANSA, Adelia, and JD Edwards (JDE).

To facilitate the adoption by developers using CASE and 4GL tools, ARCAD and Rational functionality is tightly integrated into the tool-specific development environments. Links to Rational Team Concert Work Items are managed automatically for easy planning and approval.

Enterprise Generation Language (EGL) does not have its own repository and does not require this module.

5.1.9 ARCAD-Deliver While Active Promoter (WAP): Transfer to Production of high volume data

The optional While Active Promoter (WAP) module enhances ARCAD-Deliver to secure the transfer to production of high volume data.

As part of a modernization process, and to unlock the growing potential of big data and data analytics (big data analytics), organizations need to employ exceptional technologies to handle the collection, transport, processing, and storage of data, with volume a constantly moving target.

With the advent of mobile devices, users are constantly connected and have grown to expect constant availability of applications. Therefore, the time that is allowed for operational tasks, and in particular the transfers to production of new application versions, is considerably reduced. Application downtime that results from a transfer of high volume data files and index regeneration is no longer acceptable.

ARCAD-WAP is designed to resolve this problem. A replication procedure based on High Availability (HA) technology enables updates to the structure of large files (which can otherwise last several hours or days), during normal application operation. This way, large files can be upgraded without hindering the users.

The solution runs a data copy from the source environment towards the replication area until the files are fully “synchronized”. This phase can last several hours or even days, depending on the volume of data to replicate. It is run during normal application operation and will stay active until the next phase to ensure that the files remain synchronized. A short switchover operation then transfers all objects from the new version to production by placing them into the target environment in a traditional manner. It then switches the files from the replication area to the target environment by a simple move operation (Figure 5-13).

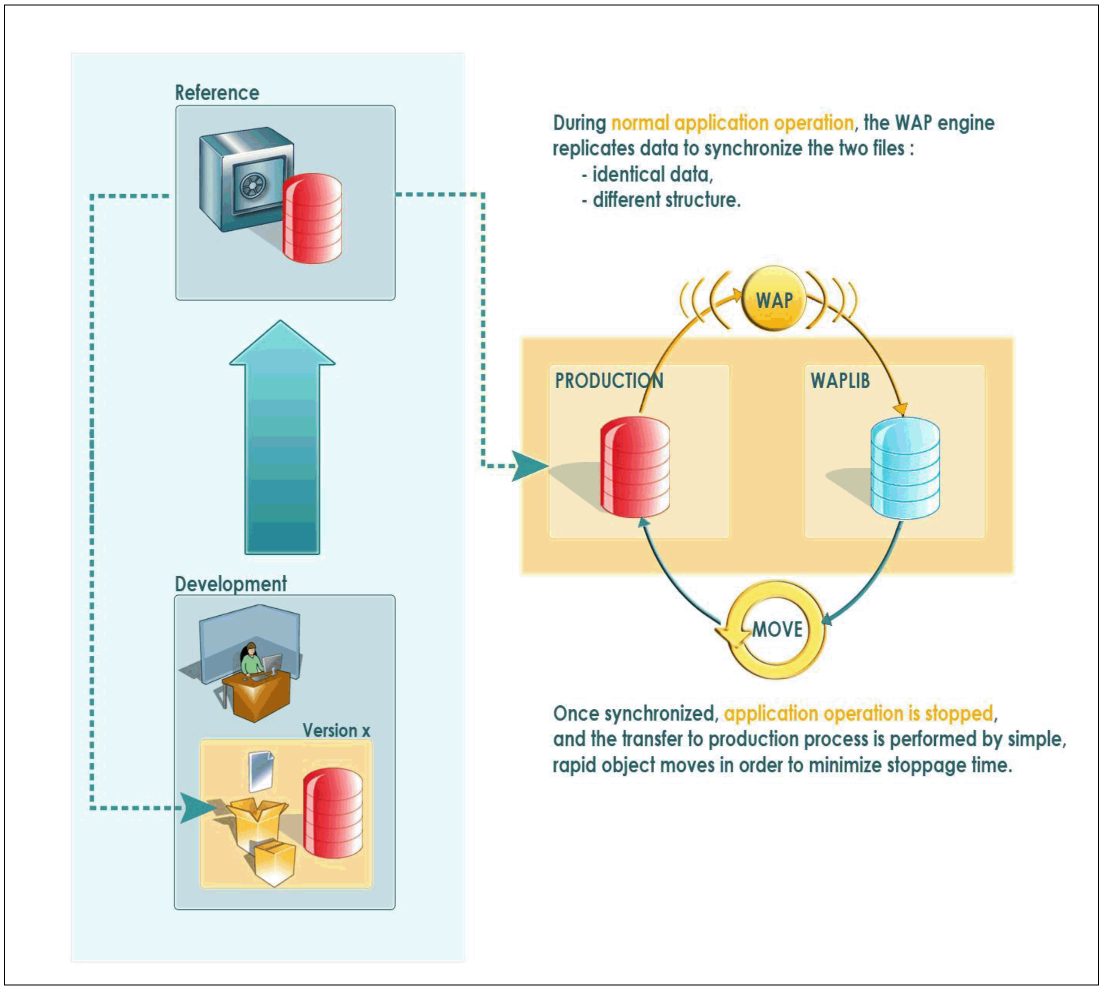

Reference
During normal application operation, the WAP engine replicates data to synchronize the two files :
- identical data,
- different structure.
WAP
PRODUCTION
WAPLIB
MOVE
Development
Version x
Once synchronized, application operation is stopped, and the transfer to production process is performed by simple, rapid object moves in order to minimize stoppage time.

Figure 5-13 File replication, synchronization, and transfer to production of high-volume data with ARCAD-Deliver WAP

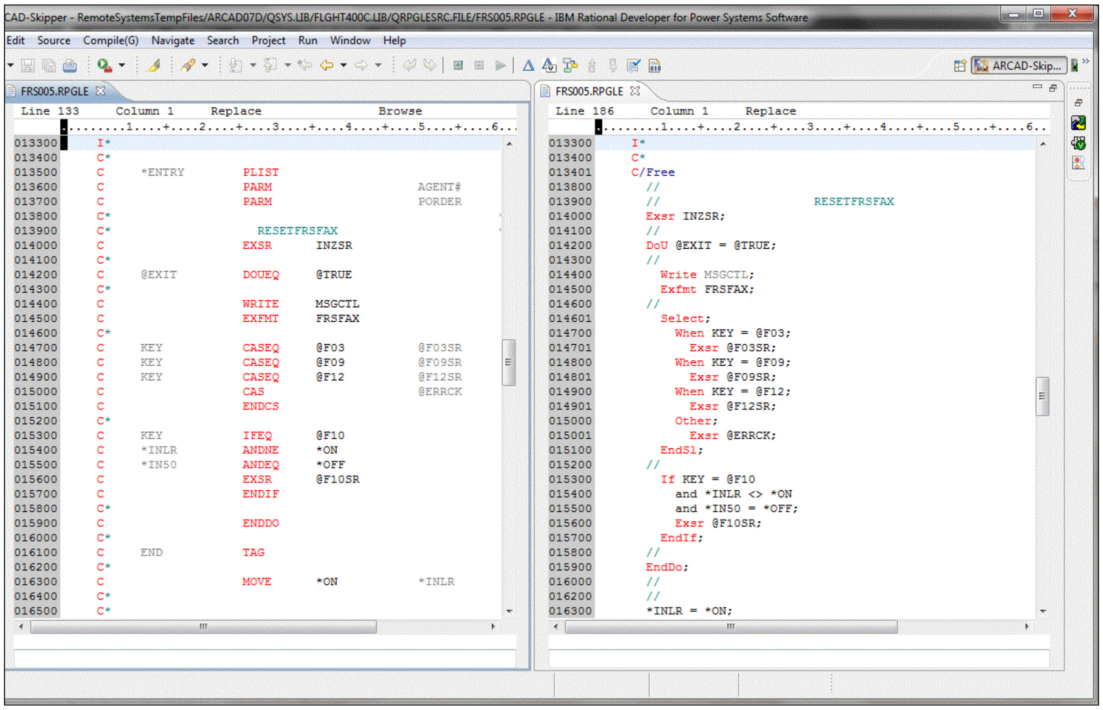
To fully automate the process, particular high-volume files can be designated for ARCAD-WAP processing during the transfer to production (or test/pre-production) phases.

5.1.10 ARCAD-Converter (ARCAD-Transformer RPG): Automated conversion from RPGLE to Free Format RPG

It goes without saying that any application modernization project should employ the latest RPG language syntax to improve the modularity, maintainability, and reusability of the application in the long term. The move to Free Format enhances the readability of source code and paves the way for new non-native developers to join the development team. Even Java, C#, or C developers can work in Free Format RPG code.

To facilitate the move to Free Format, the ARCAD-Converter is a solution that can be started from RDi or the command line to automatically convert the RPGLE source code.

The objective of ARCAD-Converter is to automatically convert all specifications into free syntax. Whereas this is straightforward for certain operation codes (for example, EVAL and ADD), others are significantly more complex (for example, MOVE and MOVEL). Indeed, the choice of instruction to generate often depends on the types, lengths, and dimensions of fields used in Factor 1, Factor 2, or results factor. For this reason, ARCAD-Converter starts by a complete cross-reference analysis down to the field level, to check the characteristics of each field in the program (Figure 5-14).


CAD-Skipper - RemoteSystemsTempFiles/ARCAD07D/QSYS.LIB/FLGHT400C.LIB/QRPGLESRC.FILE/FRS005.RPGLE - IBM Rational Developer for Power Systems Software
Edit Source Compile(G) Navigate Search Project Run Window Help
ARCAD-Skip...
FRS005.RPGLE
FRS005.RPGLE

```
Line 133      Column 1      Replace                    Browse
 .........1....+....2....+....3....+....4....+....5....+....6...
     I*
     C*
     C     *ENTRY        PLIST
     C                   PARM                    AGENT#
     C                   PARM                    PORDER
     C*
     C*                    RESETFRSFAX
     C                   EXSR      INZSR
     C*
     C     @EXIT         DOUEQ     @TRUE
     C*
     C                   WRITE     MSGCTL
     C                   EXFMT     FRSFAX
     C*
     C     KEY           CASEQ     @F03          @F03SR
     C     KEY           CASEQ     @F09          @F09SR
     C     KEY           CASEQ     @F12          @F12SR
     C                   CAS                     @ERRCK
     C                   ENDCS
     C*
     C     KEY           IFEQ      @F10
     C     *INLR         ANDNE     *ON
     C     *IN50         ANDEQ     *OFF
     C                   EXSR      @F10SR
     C                   ENDIF
     C*
     C                   ENDDO
     C*
     C     END           TAG
     C*
     C                   MOVE      *ON           *INLR
     C*
     C*
```

```
Line 186      Column 1      Replace
 .........1....+....2....+....3....+....4....+....5....+....6..
     I*
     C*
     C/Free
       //
       //                      RESETFRSFAX
       Exsr INZSR;
       //
       DoU @EXIT = @TRUE;
       //
         Write MSGCTL;
         Exfmt FRSFAX;
       //
         Select;
           When KEY = @F03;
             Exsr @F03SR;
           When KEY = @F09;
             Exsr @F09SR;
           When KEY = @F12;
             Exsr @F12SR;
           Other;
             Exsr @ERRCK;
         EndSl;
       //
         If KEY = @F10
           and *INLR <> *ON
           and *IN50 = *OFF;
           Exsr @F10SR;
         EndIf;
       //
       EndDo;
       //
       //
       *INLR = *ON;
```

Figure 5-14 Comparison of RPGLE and its Free Format equivalent generated by ARCAD-Transformer RPG (ARCAD-Converter)

The converted source can usually be compiled directly, with user intervention required only for the handling of %Found and %Equal on SCAN, CHECK, CHECKR, and LOOKUP.

5.1.11 ARCAD-Verifier: automated regression testing

ARCAD-Verifier is an optional module that is integrated with ARCAD Pack for Rational, further increasing productivity and accuracy in the development lifecycle by automating the unit and regression test phases. It provides an easy-to-use test scenario player to record user keystrokes and detect all manipulations of data. After they are recorded, test scenarios can be automatically replayed in any order to detect differences in user interface, database, and spool files. Test results can be navigated or viewed as reports in PDF format.

One of the principal characteristics of ARCAD-Verifier is its deep difference detection at the data level. ARCAD-Verifier can compare test runs down to the field level, and even to parts of fields. This granularity gives a high level of flexibility as to the type of application user interface used to record the test, with any mix of 5250, web, and client and server interfaces supported. It even allows meaningful comparison of tests runs that are run from different user interfaces, highlighting any differences in the data written. For both unit and regression testing, all test data is managed transparently, ensuring that a test replay uses precisely the same data as the previous run. ARCAD-Verifier simulates 5250 emulators in memory so that both interactive and batch tests can be scheduled to run outside working hours.

In an agile environment, testing is not considered a separate phase, but rather a part of software development along with coding. ARCAD-Verifier is well-adapted for CI, automating testing throughout the project lifecycle. Both unit and full system testing are supported to reveal problems quickly and early.

Automated testing with ARCAD-Verifier reduces overall testing effort and enhances delivery speed. Full process automation from the software change through to the presentation of test results improves software quality and shortens delivery cycles.

5.1.12 Support for multiple user interfaces: web, client and server, and 5250

The principal advantage of ARCAD-Verifier in an application modernization project lies in its ability to record and replay test scenarios from any application interface:

- Client/Server interface: Windows programs, .NET, Java, Delphi, C++, LANSA
- Web interface: Ajax, Flash, JSP/HTML/Java
- 5250 interface
- Batch processing
- Interconnection supported with all technology types such as MQ Series

5.1.13 Transparent management of data

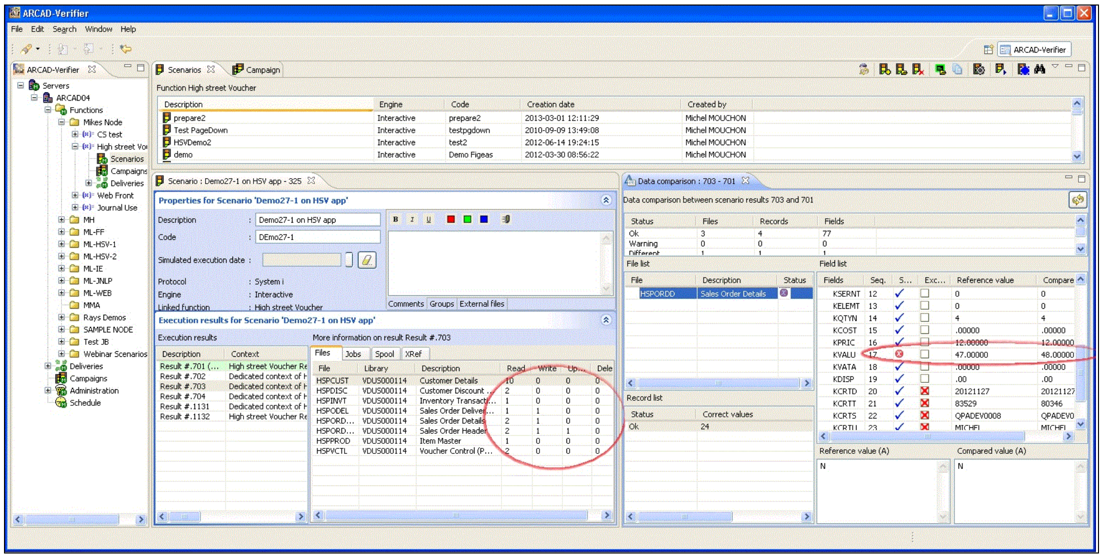
ARCAD-Verifier provides unique features for the management of test data:

- Detection of differences at the database level, meaning that the solution works for both batch and interactive jobs.
- Automatic reinitialization of test data, by using a data repository, to ensure that you are testing program changes only.

For application modernization, test scenarios already recorded in 5250 remain valuable for detecting regressions even in a hybrid application with web-front end, as they are able to detect differences early at the database level:

- Files, records, and file fields that have been changed
- File fields with modified lengths
- File fields added into the version

Figure 5-15 shows an example of test results.



Function High street Voucher

Description	Engine	Code	Creation date	Created by
prepare2	Interactive	prepare2	2013-03-01 12:11:29	Michel MOUCHON
Test PageDown	Interactive	testpgdown	2010-09-09 13:49:08	Michel MOUCHON
HSVDemo2	Interactive	test2	2012-06-14 19:24:15	Michel MOUCHON
demo	Interactive	Demo Figeas	2012-03-30 08:56:22	Michel MOUCHON

Properties for Scenario 'Demo27-1 on HSV app'

Description	: Demo27-1 on HSV app
Code	: DEmo27-1
Simulated execution date	:
Protocol	: System i
Engine	: Interactive
Linked function	: High street Voucher

Execution results for Scenario 'Demo27-1 on HSV app'

Execution results

Description	Context
Result #.701 (…	High street Voucher Re
Result #.702	Dedicated context of H
Result #.703	Dedicated context of H
Result #.704	Dedicated context of H
Result #.1131	Dedicated context of H
Result #.1132	High street Voucher Re

More information on result Result #.703

File	Library	Description	Read	Write	Up…	Dele
HSPCUST	VDUS000114	Customer Details	10	0	0	0
HSPDISC	VDUS000114	Customer Discount …	2	0	0	0
HSPINVT	VDUS000114	Inventory Transacti…	1	0	0	0
HSPODEL	VDUS000114	Sales Order Deliver…	1	1	0	0
HSPORD…	VDUS000114	Sales Order Details	2	1	0	0
HSPORD…	VDUS000114	Sales Order Header	2	1	1	0
HSPPROD	VDUS000114	Item Master	1	0	0	0
HSPVCTL	VDUS000114	Voucher Control (P…	2	0	0	0

Data comparison between scenario results 703 and 701

Status	Files	Records	Fields
Ok	3	4	77
Warning	0	0	0
Different	1	1	1

File list

File	Description	Status
HSPORDD	Sales Order Details	●

Record list

Status	Correct values
Ok	24

Field list

Fields	Seq.	S…	Exc…	Reference value	Compare
KSERNT	12	✓	☐	0	0
KELEMT	13	✓	☐	0	0
KQTYN	14	✓	☐	4	4
KCOST	15	✓	☐	.00000	.00000
KPRIC	16	✓	☐	12.00000	12.00000
KVALU	17	⊗	☐	47.00000	48.00000
KVATA	18	✓	☐	.00000	.00000
KDISP	19	✓	☐	.00	.00
KCRTD	20	✓	☒	20121127	20121127
KCRTT	21	✓	☒	83529	80346
KCRTS	22	✓	☒	QPADEV0008	QPADEV0
KCRTU	23	✓	☒	MICHEL	MICHEL

Reference value (A)	Compared value (A)
N	N

Figure 5-15 Test results viewed interactively from ARCAD-Verifier, with data comparison view

5.1.14 Test scenario management

After a test scenario is recorded, it can then be validated by replaying the scenario to verify its consistency. It then becomes the “reference”, and will be used later to detect possible regressions.

The validation phase supports these items:

- Verification of test scenario execution
- Exclusion of certain window fields that should not be compared (time, device ID)
- Exclusion of certain database fields
- Exclusion of certain spool file lines

Following the replay of a test scenario, any differences are highlighted for user validation. Expected or insignificant differences can be accepted to make this replay the new reference. ARCAD-Verifier can also simulate a change in the Job date without affecting the underlying system itself.

Scenarios can be readily maintained by inserting breakpoints and re-recording or adding specific areas of a scenario.

5.1.15 Automatic generation of test campaigns

With ARCAD-Verifier, test campaigns can be as frequent as needed because the tool automatically generates, runs, and reports the campaigns, starting from the list of artifacts changed. That is, ARCAD-Verifier uses built-in cross-references between scenarios and programs to automatically identify which scenarios must be replayed to regression test your latest program changes.

5.1.16 Integration with Rational Quality Manager

Although ARCAD-Verifier can be used as a stand-alone application, it is also integrated with Rational Quality Manager. ARCAD-Verifier test scenarios can be included within Rational Quality Manager Test Cases and Test Suites, allowing ARCAD-Verifier users to benefit from the web-based collaborative and test planning features offered by Rational Quality Manager.

5.2 Solution architecture

ARCAD Pack for Rational is an Eclipse-based solution, extending Rational Team Concert and RDi. It populates and maintains an internal metadata repository by analyzing source code from multiple platforms that include IBM i, Windows, UNIX, and Linux.

ARCAD Pack for Rational provides the deep IBM i functionality that is needed to support the many specialized technology variants on the platform. It completes the automation of dependency analysis, integration build and deployment, simplifying the change process and shortening time to delivery (Figure 5-16).

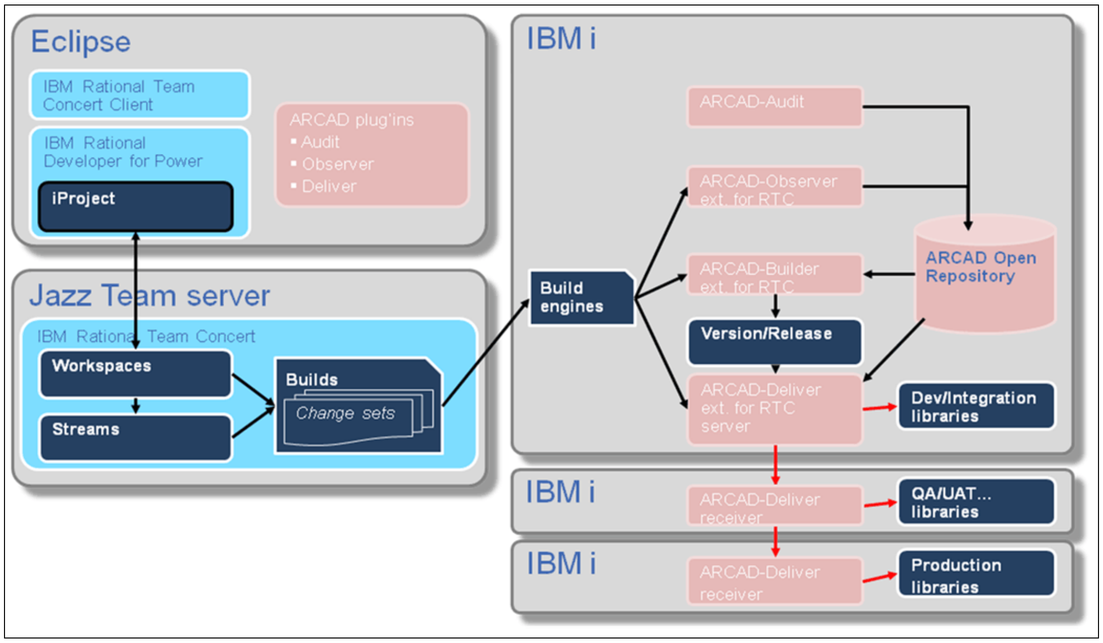


Figure 5-16 ARCAD Pack for Rational: Solution Architecture diagram

5.3 Usage scenarios

ARCAD Pack for Rational advances the pace of enterprise modernization within the IBM i development community, enabling organizations to maximize the return on their investment in application development.

For example, the ARCAD Pack for Rational is used in these tasks:

- Improving your team's development intelligence with specialized analysis of the IBM i application code and dependencies, maintained automatically in a dedicated repository with the integration of ARCAD-Observer and Rational Team Concert

- Speeding code comprehension and change analysis with the extension of the Rational Developer IDE by ARCAD-Observer's Eclipse-based user interface for visualization of application structure and dependencies
- Automating and improving the integration build of IBM i applications by incorporating the ARCAD-Builder tool in the Rational Team Concert build configuration. ARCAD-Builder uses the ARCAD Open Repository to determine all dependencies and steps that are required for an accurate, repeatable integration build ready for deployment.
- Orchestrating the automated deployment of your applications to multiple test or production platforms simultaneously with the integration of ARCAD-Deliver and Rational Team Concert. ARCAD-Deliver includes security-rich features, including system-wide traceability and the ability to roll back on error for deployment operations. ARCAD-Deliver supports multiple target platform types that include IBM i, Windows and Linux, offering a single, synchronized process for all platforms in the information system.
- Automating the conversion of RPG sources to Free Format RPG thus bridging the gap between web and older development cultures and freeing movement of staff between teams.

Teams that are embarking on application modernization face a strategic choice from open systems or Microsoft frameworks, GUI modernization, web enablement tools, and older back-office components that are too critical to change. For the sake of maintainability and to capitalize on the multiple interfaces offered by web and mobile technologies, many organizations modularize their IBM i applications into a multitier and model view controller (MVC) architecture.

Languages such as Java or PHP are commonly used at the presentation layer, RPGLE for the model or business logic, and SQL for data management. The outcome is a rich mix of inter-dependent environments co-existing in the same information system. ARCAD Pack for Rational is able to handle the specifics of each technology and provide a unified user interface that is acceptable and familiar to the development communities involved. It manages the heterogeneous components through the development cycle up to delivery on various different target platforms, and has an open and modular architecture that allows rapid interfacing with existing tools.

5.3.1 Supported platforms

This solution is supported by platforms with the following features:

- IBM i operating system V5R4 and later
- x86 64-bit systems with a minimum of 4 GB of memory
- Minimum 40 GB of disk storage

5.3.2 Ordering information

ARCAD Pack for Rational and ARCAD-Verifier licenses and support can be purchased from ARCAD Software. Contact your regional representatives at `sales-eu@arcadsoftware.com`, `sales-us@arcadsoftware.com`, or `sales-asia@arcadsoftware.com`.

ARCAD Pack for Rational is available through IBM Passport Advantage.

Table 5-1 shows the ordering information.

Table 5-1 IBM Ordering part number and feature code

Program name	**PID number**
ARCAD Pack for Rational per Observer Authorized User	5725-L13
IBM Elite Support for ARCAD Pack for Rational	5725-L24

5.4 Related information

- EMA Research Report, *DevOps for a New Millennium: A Lifecycle Perspective Supporting Business Growth in an Altered Economy* (Q2, 2013).
- “Modernize your IBM i projects with Rational Software and ARCAD-Rational Power Pack,” *IBM Software*, April 2013.
- Simon Webb, “Agility Matters,” *IBM Systems Magazine*, May 2013.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Modernize IBM i Applications from the Database up to the User Interface and Everything in Between*, SG24-8185

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ADSERO Optima

  http://www.adsero-optima.com
- ARCAD Software

  http://www.arcadsoftware.com
- ASNA Mobile RPG

  http://asna.com/us/products/mobile-rpg/
- ASNA Wings

  http://asna.com/us/products/wings/
- BCD Presto

  http://www.bcdsoftware.com/iseries400solutions/presto/
- BCD WebSmart

  http://www.bcdsoftware.com/iseries400solutions/websmart/
- CNX Valence

  http://www.cnxcorp.com/valence
- Fresche Legacy

  http://www.freschelegacy.com/
- IBS XT Platform

  http://ibs-xt-platform.software.informer.com/

- LANSA LongRange

  http://www.longrangemobile.com
- LANSA RAMP

  http://www.lansa.com/ramp
- LANSA Visual LANSA

  http://www.lansa.com/products/visual-lansa-ide.htm
- looksoftware newlook

  http://www.looksoftware.com/products/newlook.aspx
- looksoftware openlook

  http://www.looksoftware.com/products/openlook.aspx
- Profound Logic Software

  http://www.profoundlogic.com
- Rocket LegaSuite

  http://www.rocketsoftware.com/brand/rocket-legasuite
- SkyView Partners

  http://www.skyviewpartners.com
- Surround Technologies

  http://www.surroundtech.com
- SystemObjects

  http://www.systemobjects.com/
- Xcase for i

  http://www.xcasefori.com

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Tools and Solutions for Modernizing Your IBM i Applications

Discover application modernization tools

Create mobile, web, and client solutions

Manage security using SkyView Policy Minder

This IBM Redpaper publication is a companion to the Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between, SG24-8185. It describes independent software vendor (ISV) and Business Partner tools that can be used to modernize your IBM i applications. It includes the following types of tools:

- Mobile, web, and client solutions
- Database modernization tools
- Security
- Tools for understanding and modernizing RPG and COBOL

REDP-5095-00