

Migrating RPG Programs to EGL

The tool-supported migration of existing RPG programs to EGL is now possible with the Migration Tools 400 EGL. This has the advantage of making it possible to reuse a large proportion of the existing application logic in EGL. Apart from the simple coexistence or integration of existing RPG programs with newly developed EGL programs, most of the existing program logic can be transferred to EGL with this approach, thus considerably prolonging the lifecycle of an application.

This makes sense in the following situations:

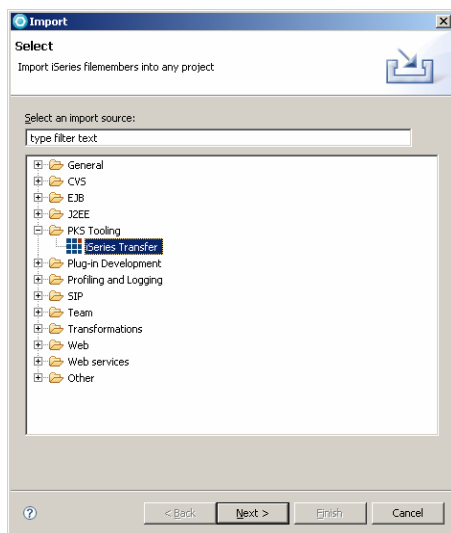
- You want to extend the lifecycle of the application.
- Existing application logic will be broken down into business processes or transferred to a service-oriented architecture.
- The RPG development is to be replaced by more modern development.
- Existing apps should be made platform-independent.
- Serviceability should be improved through modularization.
- Operational workflow is to be optimized through the possibilities of a GUI.

To simplify migration, PKS has developed “Migration Tools 400 EGL” with the support of IBM’s EGL lab. These tools make migrating existing RPG programs to EGL as easy as possible.

In this paper, we’ll be taking a simplified step-by-step look at how migration works.

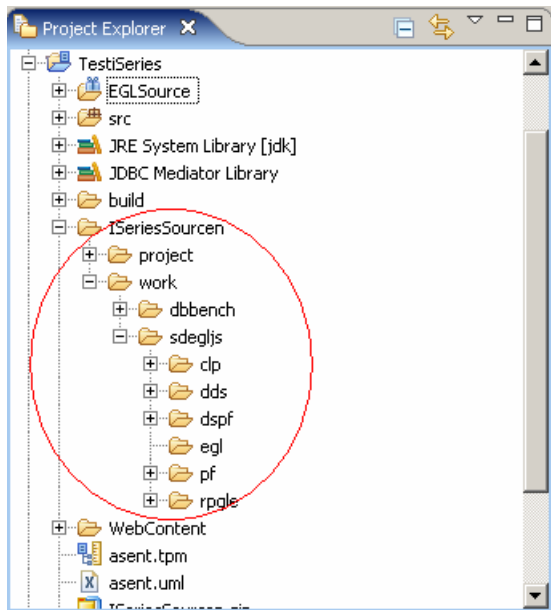
1. Transfer the existing sources from iSeries to the Rational development environment. In doing so, the RPGLE sources will be needed as well as all of the PF, LF, DSPF and PRTF sources including reference files.

Using the Import Wizard, the respective sources are transferred from the System i members to a suitable EGL project structure.



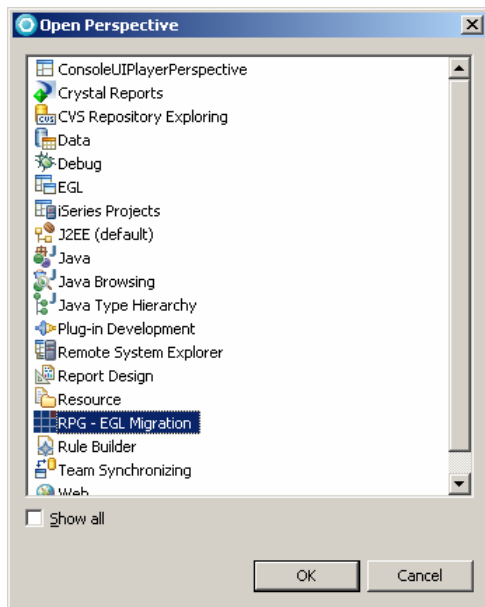
(Image: PKS plugin for transferring sources from iSeries to Rational, part 1)

The result of the source transfer will look like this in the Project Explorer:

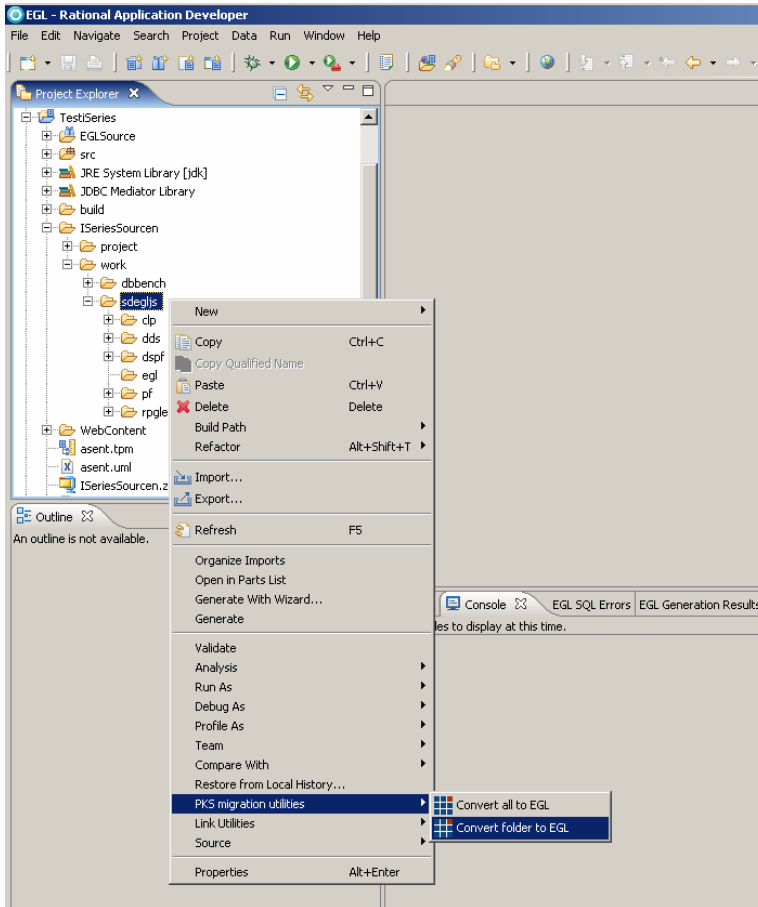


All sources will be transferred to a standardized project structure. An individual directory will be created here for every library, as well as corresponding subdirectories for the various types of sources. Additional project information such as migration protocols is stored in the “Project” subdirectory.

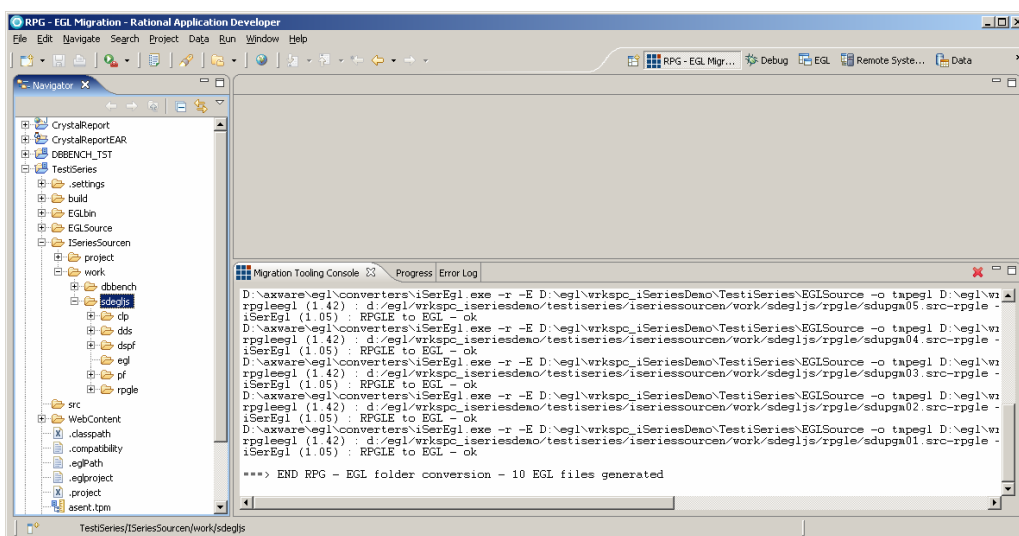
2. Now the perspective changes to “RPG – EGL Migration.” You’ll find all of the tools needed for migration in this perspective:



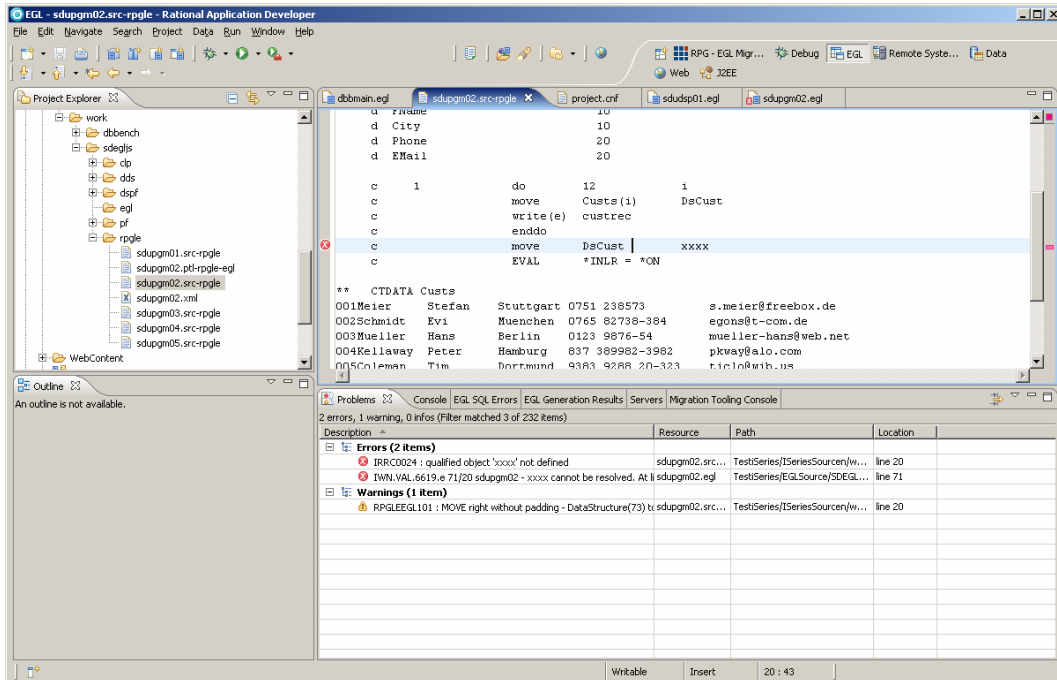
- After changing the perspective, the corresponding tools are available. Migration can now start. It can be done source-for-source or as a full-scale migration for a complete project or subproject.



While migration is running, all of the completed steps will be shown in the Migration Tooling Console. Once migration is completed, the console responds with a message about the number of migrated programs.

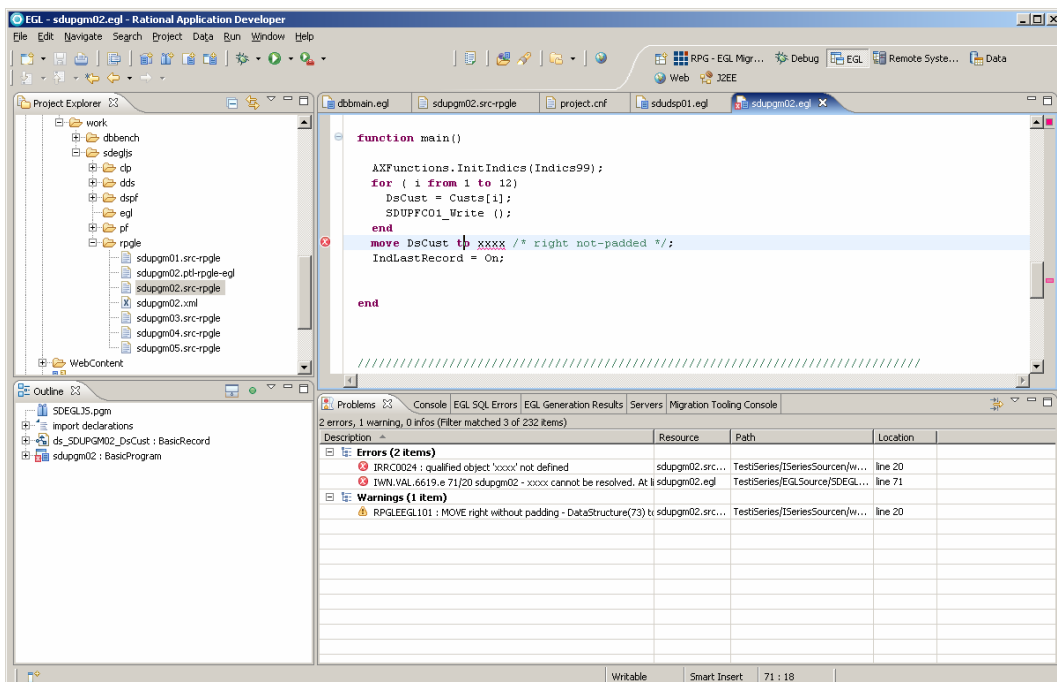


In order to see what happens when an error occurs while migrating programs, we will simulate a problem with a MOVE command in the program SDUPGM02. The command MOVE DSCUST XXXX works with a non-declared field XXXX.



In the Problems Console, you can see a complete list of all problems found in the RPG program. By clicking one of the problems listed here, the editor will jump to exactly that erroneous line. The error can then be fixed directly in RPG before remigrating the program.

Alternatively, the problem can also be viewed in the EGL perspective and corrected directly within the generated EGL source code. To the right of the scrollbar you can see the corresponding color code for warnings (yellow) or errors (red).



4. All database calls in EGL are neatly encapsulated in EGL libraries. The required access modules are automatically generated in the corresponding PF and LF sources. For each PF and LF, an individual module is created. The known CRUD functions (Create Read Update Delete) are available in an extended form within the access module. The module works internally with SQL and is therefore database and platform-independent.

List of the database access functions:

- function Create
- function Chain
- function Read
- function ReadEqual
- function ReadPrior
- function ReadPriorEqual
- function SetGreaterThan
- function SetLowerLimit
- function Write
- function UpdateRcd
- function DeleteRcd
- function DeleteCurrent

In addition, so-called data items and the corresponding SQL record are created for the database fields:

```
dataItem di_SDUPFC01_CUSTREC_CITY      char(10) end
dataItem di_SDUPFC01_CUSTREC_EMAIL    char(20) end
...

record sql_SDUPFC01_CUSTREC type SQLRecord
{
  tablenames = [{"SDUPFC01"}],
  keyItems = [ID]
}
ID          di_sql_SDUPFC01_CUSTREC_ID;
LNAME      di_SDUPFC01_CUSTREC_LNAME;
FNAME      di_SDUPFC01_CUSTREC_FNAME;
CITY       di_SDUPFC01_CUSTREC_CITY;
PHONE      di_SDUPFC01_CUSTREC_PHONE;
EMAIL      di_SDUPFC01_CUSTREC_EMAIL;
end
```

5. For the soft copies, a corresponding access module will be created from the DSPF sources. These access modules are based on a PKS library for depicting DSPF-based displays via XML on the web or in Windows. Through the use of this library, the effort needed to make the switch will be kept to a minimum and all newly developed EGL programs with native JSF/AJAX technology are easy to integrate.

List of the display access functions:

- function Write Head
- function Write Tail
- function Exfmt Ctl
- function Read Tail
- function ReadC Tail

Generally speaking, these display access modules can also be replaced by any other UI functionality.

6. In the current version, PRTF and CL programs are not migrated. For printing programs, you should check if a replacement with EGL reporting programs (e.g. BIRT or Crystal Reports) makes sense. EGL v7.1 will have enhanced migration tools with a converter for PRTF to EGL TUI forms.

The migration of CL programs is only necessary if the application is needed in a platform-independent version. This constitutes a special case that is usually dealt with according to the specific project.

7. Now let's have a look at the generated EGL code in comparison with the old RPG code.

Example 1

RPGLE:

```

D cCMD          s          10
D cOutput       s          1024

C              EVAL      HEADER = 'PKS - Demo - iSeries RPGLE to EGL'
C              EVAL      + ' Ravensburg 31.10.2007'

C              DOW       *IN12 = *OFF AND *IN03 = *OFF

C              EVAL      SELECTION = ' '

C              EXFMT     FORMAT01

C              EVAL      *IN50 = *OFF
C              EVAL      *IN51 = *OFF
C
C              SELECT
C              WHEN      SELECTION = '1'
C              EVAL      *IN51 = *ON
C              WHEN      SELECTION = '2'
C              CALL      'SDUPGM03'
C              WHEN      SELECTION = '3'
C              EXSR      NEWFUNC
C              WHEN      SELECTION = '6'
C              CALL      'SDUREFO'
C              WHEN      SELECTION = '9'
C              CALL      'SDEGLCLEAR'
C              CALL      'SDUPGM02'
C              OTHER
C              EVAL      *IN50 = *ON
C              ENDSL
C
C              ENDDO
C
C              EVAL      *INLR = *ON
C
C              NewFunc   begin
C              EVAL      *IN50 = *ON
C              EVAL      cCMD = 'WINDOW'
C              EVAL      cURL = 'http://js-laptop:9080' +
C              '/TestiSeriesWeb/displayAll.faces'
C              EVAL      cOutput = 'CID=0'
C              call      'EGLCJSP'
C              parm      cCMD
C              parm      cURL
C              parm      cOutput
C              end

```

EGL:

```

function main()
    AXFunctions.InitIndics(Indics99);
    SDUSP01.Call ("OPEN","");
    HEADER = "PKS - Demo - iSeries RPGLE to EGL" + " Ravensburg 31.10.2007";
    while ( Ind[12] == Off && Ind[3] == Off )
        SELECTION = " ";
        SDUSP01.Call ("EXFMT","FORMAT01");
        Ind[50] = Off;
        Ind[51] = Off;
        case
        when ( SELECTION == "1" )
            Ind[51] = On;
        when ( SELECTION == "2" )
            call "SDUPGM03";
        when ( SELECTION == "3" )
            NEWFUNC ();
        when ( SELECTION == "6" )
            call "SDUREFO";
        when ( SELECTION == "9" )
            call "SDEGLCLEAR";
            call "SDUPGM02";
        otherwise
            Ind[50] = On;
        end
    end
    IndLastRecord = On;
end

function NewFunc ( )
    //              EVAL      *IN50 = *ON
    cCMD = "WINDOW";
    cURL = "http://js-laptop:9080" + "/TestiSeriesWeb/displayAll.faces";
    cOutput = "CID=0";
    call "EGLCJSP" ( cCMD, cURL, cOutput );
end

```

This simple example reveals that the migrated RPG programs are easy for RPG developers to understand at this stage, even in EGL. Particular attention was paid to good serviceability and the structure of the generated code in order to simplify any future reengineering tasks that might need to be done.

For non-RPG developers in particular, the programs' comprehensibility has been greatly improved.

RPGLE:

- Not easily readable for non-RPGL developers due to RPG-specific keywords
- Procedural setup
- Database and display operation indicators difficult to read (for untrained RPGLE programmers)
- Beginning and end of loop rather difficult to read
- IF queries (multi-nested) are difficult to read

EGL:

- Easy to read even for non-RPG developers thanks to "standard" keywords
- Procedural setup due to use of encapsulated functions for DSPF and the database
- Database or display operation indicators are subsequently supplied per function. The name of the function reflects the meaning of the indicator.
- The beginning and end of loops – and the loop constraint – are easily recognizable thanks to indentation and color coding
- IF queries including constraints are easier to read

Example 2

```

C          DOW      *IN12 = *OFF AND *IN03 = *OFF
C          IF      Sfiload = '1'
C          *EXT   ClearSfil
C          *EXT   LoadSfil
C          *ENDIF
C          write    head
C          write    tail
C          *EXT    ctl

C          DOW      *IN90 <> *ON
C          *READC  SFL
C          *IF     *IN90 <> *ON
C          ***** SFL Option 2
C          IF      OPT = '2'
C          call    'SDUPGM04'
C          parm   id
C          eval   Sfiload = '1'
C          *ENDIF
C          ***** SFL Option 5
C          IF      OPT = '5'
C          call    'SDUPGM05'
C          parm   id
C          *ENDIF
C          ***** SFL Option 6 Call JSP
C          IF      OPT = '6'
C          MOVE    ID          cID
C          EVAL   cCMD = ''
C          EVAL   cURL = 'http://js-laptop:9080' +
C                      '/Test1SeriesWeb/ChangeRecord.faces'
C          EVAL   cOutput = 'CID'+cID
C          call    'EGLCJSP'
C          parm   cCMD
C          parm   cURL
C          parm   cOutput
C          eval   Sfiload = '1'
C          *ENDIF
C          ***** SFL Option 7 Call JSP
C          IF      OPT = '7'
C          MOVE    ID          cID
C          EVAL   cCMD = ''
C          EVAL   cURL = 'http://js-laptop:9000' +
C                      '/Test1SeriesWeb/DisplayRecord.faces'
C          EVAL   cOutput = 'CID'+cID
  
```

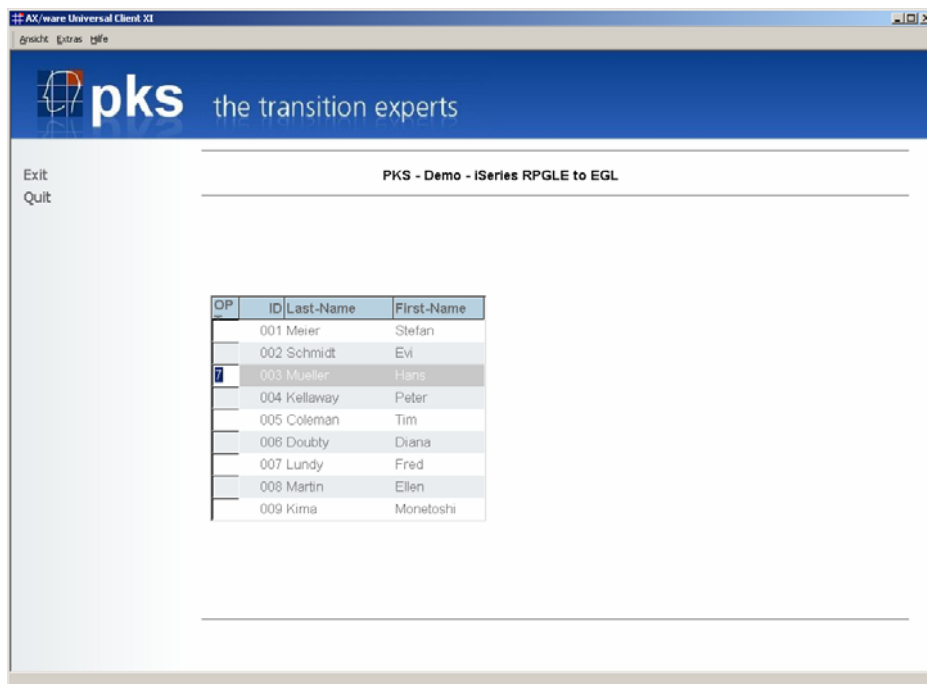
```

while ( Ind[12] == Off && Ind[03] == Off )
if ( Sfiload == "1" )
ClearSfil();
LoadSfil();
end
SDUPSPO3_Call ("WRITE","HEAD");
SDUPSPO3_Call ("WRITE","TAIL");
SDUPSPO3_Call ("EXTNT","CTL");

while ( Ind[90] != On )
SDUPSPO3_Call ("READC","SFL");
Ind[90] = axinfds.Check_KOV(SDUPSPO3_INFD3);
if ( Ind[90] != On )
/* ** SFL Option 2
if ( OPT == "2" )
call "SDUPGM04" ( ID );
Sfiload = "1";
end
/* ** SFL Option 5
if ( OPT == "5" )
call "SDUPGM05" ( ID );
end
/* ** SFL Option 6 Call JSP
if ( OPT == "6" )
cID = ID;
cCMD = " ";
cURL = "http://js-laptop:9080" + "/Test1SeriesWeb/ChangeRecoo";
cOutput = "CID=" + cID;
call "EGLCJSP" ( cCMD, cURL, cOutput );
Sfiload = "1";
end
/* ** SFL Option 7 Call JSP
if ( OPT == "7" )
cID = ID;
cCMD = " ";
cURL = "http://js-laptop:9000" + "/Test1SeriesWeb/DisplayRec";
cOutput = "CID=" + cID;
call "EGLCJSP" ( cCMD, cURL, cOutput );
Sfiload = "1";
end
if ( OPT == "8" )
call "EARNWS";
end
  
```

In this example, you can see some more complex display operations with subfiles. Each of the red blocks shows the functional correspondent in RPG and EGL.

- The migrated program can be started after compilation. In our example, native code for i5 OS has been generated and directly compiled from the EGL source. The performance is roughly comparable to that of RPG. The soft copies result from the PKS library in our example.



Through the SDUPGM01 program, the SDUPGM03 program is launched.

The library uses XML descriptions for the display, which are automatically generated from the DSPF source. Transformation of the green screen into a web-based user interface is done through additional rules and keywords. Special programming knowledge of HTML, JavaScript, JSF or AJAX is not required for this purpose.

DSPF-based displays migrated this way can be seamlessly integrated in the newly developed displays in EGL. To demonstrate this, we've chosen a set with the option "7" in a subfile, which will take us right into the newly coded EGL program (this program will not be looked at in more depth here).

When switching programs, the transition is made between a DSPF-based display and a JSF-based EGL "native" display. What exactly happens when launching the program?

Through the API a JSP page can be loaded with a simple call. At the same time, data can also be transferred to and from the JSP page.

Excerpt from the EGL program SDUPGM03 for the subfile option "7:"

```
// *** SFL Option 7 Call JSP
if ( OPT == "7" )
  cID = ID;
  cCMD = " ";
  cURL = "http://js-laptop:9080" +
        "/TestiSeriesWeb/DisplayRecord.faces";
  cOutput = "CID=" + cID;
  call "EGLCJSP" ( cCMD, cURL, cOutput );
End
```

The API from the PKS library supports three different display variants:

- in a small frame (Image 1); full control goes to the JSP PGM; EGL iSeries native PGM waits for this/these JSP page/s to finish
- entire frame (Image 2); full control goes to JSP PGM; EGL iSeries native PGM waits for this/these JSP page/s to finish
- external window (Image 3); work can continue to be done directly in the application; separate window for the report evaluation

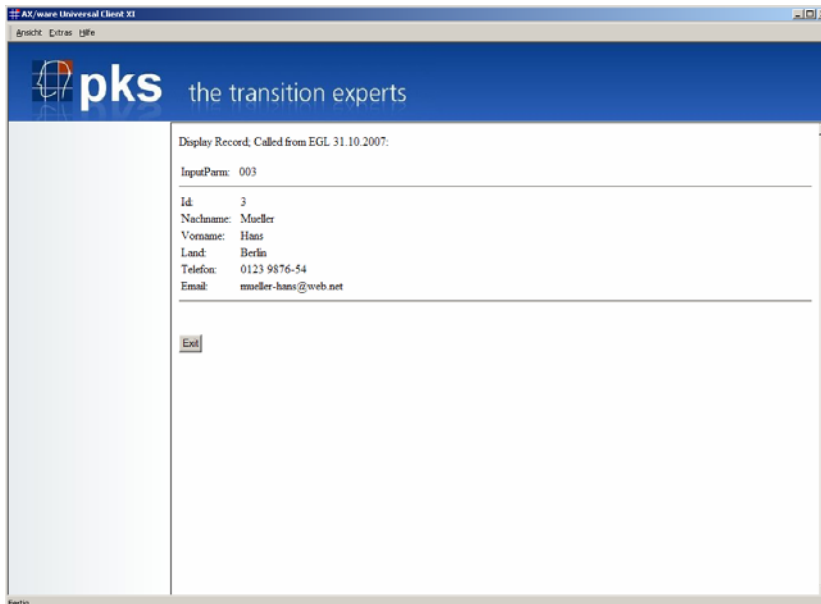


Image 1: loading a JSP page of an EGL native program via the EGLCJSP API

With a bit of programming in EGL, a program can be created for e.g. accessing the latest weather report

on the Internet through web services. This can be integrated in the migrated RPG program. With RPG itself, a problem like this would hardly be able to be solved.

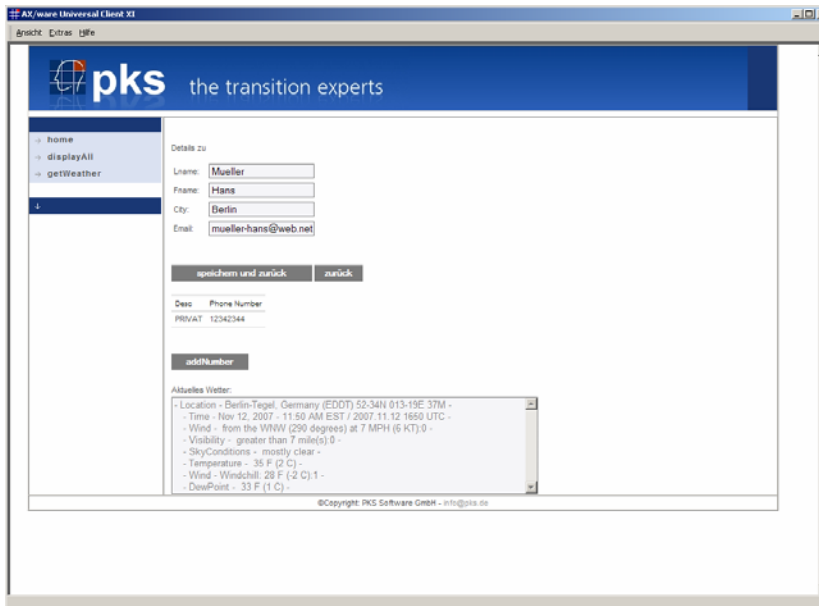


Image 2: loading a JSP page of an EGL native program via EGLCJSP API Use of the entire window and integration in an external web service.

Reporting problems can also easily be solved with EGL by integrating reporting tools like Crystal Reports:

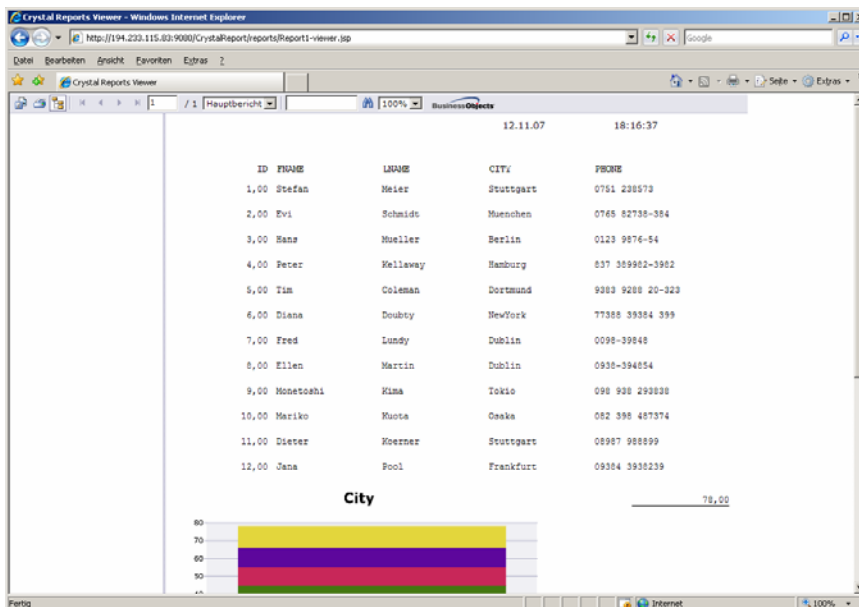


Image 3: loading the Crystal Report view via EGLCJSP API in its own frame/window

As this example shows, EGL provides many useful ways to enhance migrated RPG programs with new functionality. Switching to EGL is an attractive way for RPG programmers to quickly improve existing programs and expand them with new functionality. The transition can be made in small steps and thus provide added value to daily business. RPG/EGL developers can now easily accomplish things that used to call for tedious programming in Java.