IBM Power Systems – IBM i – IBM Rational

# Introduction à Rational Open Access: RPG Edition

## *(RPG Open Access / RPG OA)*

Philippe Bourgeois – IBM France

pbourgeois@fr.ibm.com

# Plan de la présentation

- Quelques mots sur les offres Rational for Power Systems 2010

- Introduction à RPG Open Access
  - Qu'est-ce que Rational Open Access: RPG Edition ?
  - Principes de fonctionnement
  - Comment est construit un « handler » ?
  - Exemples simples d'applications Open Access

- Quelques mots sur les handlers d'IBM Lab Services

- Annexe - Quelques mots sur le handler de la société looksoftware

# Quelques mots sur les offres Rational for Power Systems 2010

- RD Power - Rational Developer for Power Systems
  - Développement d'applications pour IBM Power Systems
    - RPG and COBOL Development Tools for i
    - COBOL Development Tools for AIX
    - C/C++ Development Tools for AIX
    - C/C++ Development Tools for Linux on Power

- RTC Power – Rational Team Concert for Power Systems
  - Contrôle des sources / versionning, gestion des demandes de changement, travail en équipe, gouvernance des développements

- Rational Open Access: RPG Edition
  - Ouverture du langage RPG

# Rational Open Access: RPG Edition

## *Introduction*

# Qu'est-ce que RPG Open Access ?

- RPG Open Access fournit aux développeurs RPG le moyen d'utiliser le modèle simple et bien connu d'E/S du RPG (codes-opération READ, WRITE, CHAIN, etc.) pour accéder à des ressources et des unités qui ne sont pas directement supportées en RPG :
  - Fichiers XML, CSV, TXT, etc.
  - Services Web
  - Bases de données externes
  - Fichiers base de données cryptés
  - Navigateurs (browsers)
  - Unités mobiles
  - …

# RPG Open Access – Principe de fonctionnement

- Une application Open Access a trois composantes :
  - 1) Un programme RPG qui utilise les codes-opération d'E/S classiques du RPG (read, write, chain…) sur un fichier déclaré en « open-access »
    - Pour déclarer un fichier en « open access », on indiquera le mot-clé *handler* au niveau de ce fichier avec comme attribut le nom d'un programme ou d'une procédure ILE, que l'on appellera le « handler »
    - La présence de ce mot-clé *handler* permettra d'indiquer que lorsqu'une opération d'E/S sera réalisée sur ce fichier, ce ne sont pas les routines de l'OS qui seront appelées mais le programme handler
  - 2) Un programme « handler »
    - Qui sera appelé à chaque opération d'E/S sur le fichier déclaré en « open access » au point 1
    - Récupérera un buffer d'informations venant du programme RPG
    - Puis, à partir de ce buffer d'informations, communiquera avec une ressource, une unité ou un programme cible de « rendu »

# RPG Open Access – Principe de fonctionnement

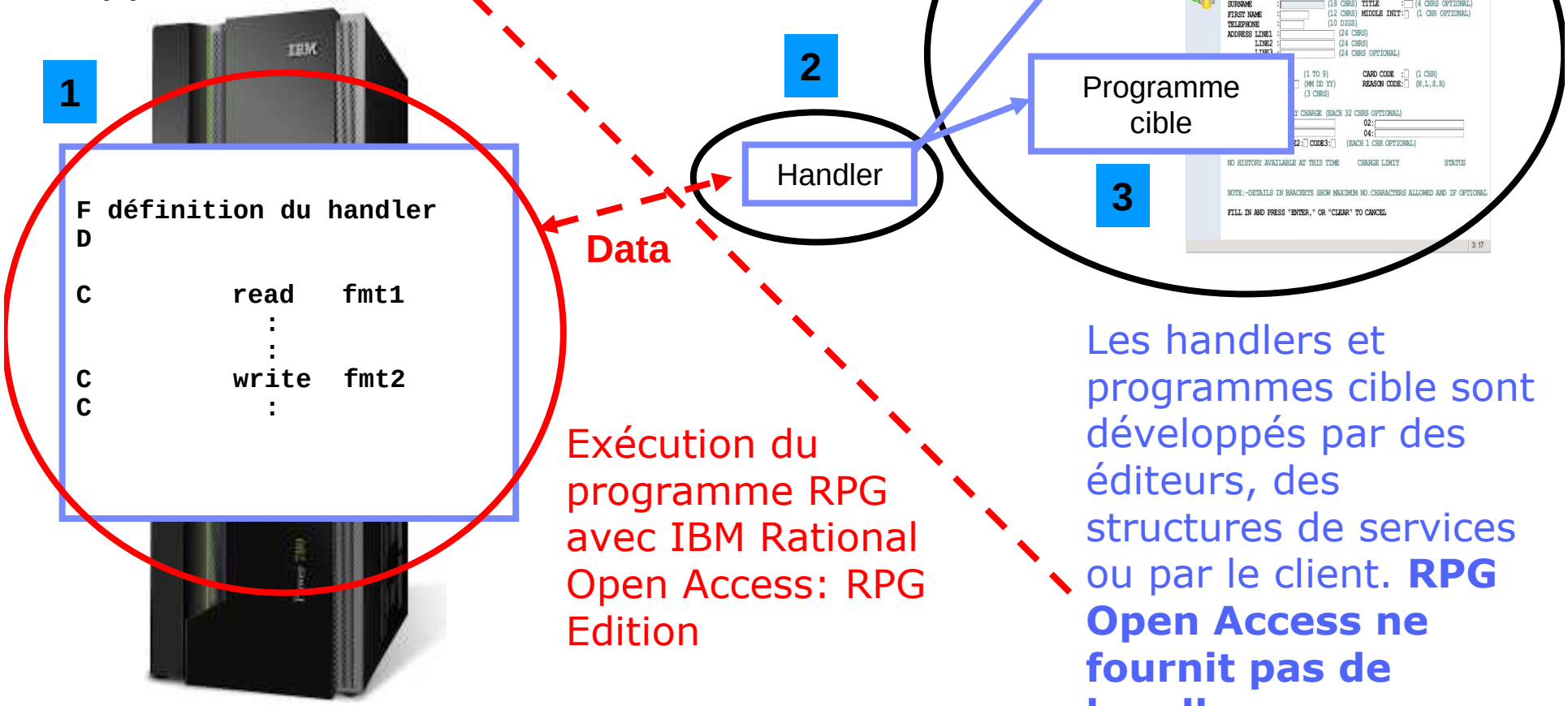
- 3) Une ressource, unité ou un programme cible de rendu
  - Le handler peut communiquer directement avec la ressource (par exemple un fichier dans l'IFS)
  - Ou bien faire appel à un programme cible de rendu (rendering program) qui sera spécifique à l'unité (Web, unité mobile…)

- Le produit Rational Open Access: RPG Edition permet de faire le lien entre les composantes 1 et 2

- Il est nécessaire à l'exécution de l'application

# RPG Open Access - Principes

Applications RPG



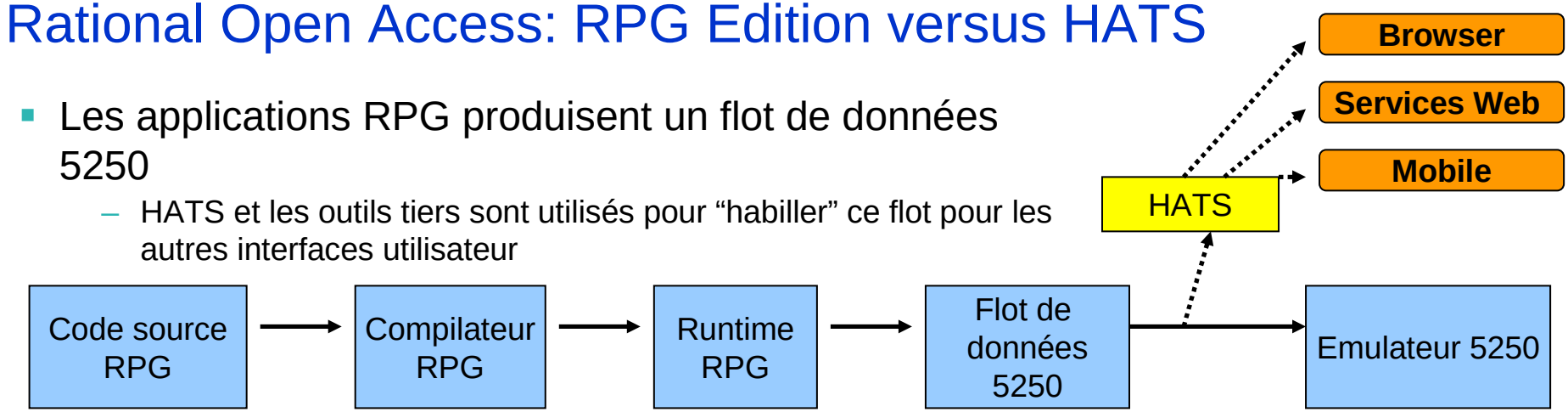

Exécution du programme RPG avec IBM Rational Open Access: RPG Edition

Les handlers et programmes cible sont développés par des éditeurs, des structures de services ou par le client. **RPG Open Access ne fournit pas de handlers**

**Le développeur continue à développer en RPG**
**Il fait appel, de façon transparente, aux procédures du handler**

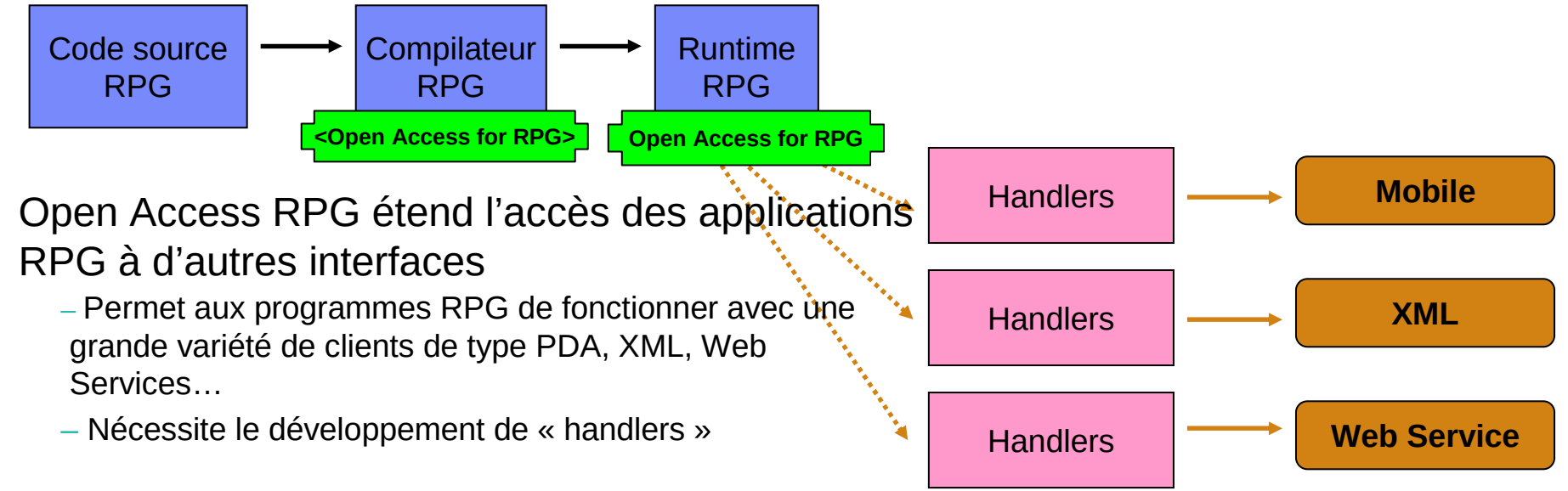
# Rational Open Access: RPG Edition versus HATS

- Les applications RPG produisent un flot de données 5250
  - HATS et les outils tiers sont utilisés pour “habiller” ce flot pour les autres interfaces utilisateur




*Applications RPG traditionnelles*

*Applications RPG avec Rational Open Access: RPG Edition*




- Open Access RPG étend l’accès des applications RPG à d’autres interfaces
  - Permet aux programmes RPG de fonctionner avec une grande variété de clients de type PDA, XML, Web Services…
  - Nécessite le développement de « handlers »

# RPG Open Access – Prérequis - Packaging

- IBM i 7.1 ou 6.1
- En développement :
  - Le mot-clé *handler* n'est pas supporté dans SEU mais uniquement dans RD Power
- En IBM i 6.1, la compilation du programme RPG qui contient le mot-clé handler nécessite une PTF :
  - SI39483 sur le 5761-WDS
- A l'exécution :
  - En IBM i 6.1 la PTF SI39480 sur le 5761-SS1 est nécessaire
  - Le produit 5733-OAR (Rational Open Access: RPG Edition) est nécessaire
    - Facturé au groupe logiciel
      - P05 : 464 €
      - P10 : 927 €
      - P20 : 2318 €
      - P30 à P60 : 4637 €

# RPG Open Access – Les handlers disponibles

- Les handlers des éditeurs en 2010
  - IBM Lab Services
    - 3 handlers : FFM (File Format Messaging), Services Web et sockets
  - Looksoftware
    - Interfaces client riche, client léger et client mobile
  - Profound Logic
    - Interface client riche Web
  - VAI
    - Interface client riche Web (pour leur ERP uniquement)
  - Rocket Seagull Software
    - Services Web, SOA

# RPG Open Access – Quelques détails

- **1** Le programme RPG
  - On indique quels sont les fichiers qui sont en « open access » en spécifiant le mot-clé HANDLER en spécification F
  - Le mot-clé HANDLER permet d'indiquer quel est le nom du programme ou du programme de service handler :
    - HANDLER(nom du programme ou du programme de service)
    - Le nom peut être défini en dur ou sous forme de variable
  - Exemples :

```
Ffichier1 if    e  k  disk     handler('bib1/pgm1')
Ffichier2 cf    e     workstn  handler('srvpgm1(proc1)')
```

  - Le fichier peut être de type DISK, WORKSTN, PRINTER et il est généralement en corrélation avec le type d'unité cible :
    - Interface utilisateur → fichier WORKSTN
    - Fichier Excel → fichier PRINTER
    - Service Web → Fichier DISK ... / …
  - Le code qui suit est inchangé : on utilise les mêmes codes-opération (WRITE, READ, EXFMT…)

# RPG Open Access – Quelques détails

- **2** Le handler
  - C'est un programme ou un programme de service ILE
  - Il peut être écrit dans tout langage ILE (RPG, COBOL, CL, C, C++)
  - Il gérera toutes les opérations d'E/S sur les fichiers concernés (READ, WRITE, CHAIN, EXFMT…)
  - Les données qui sont passées du/vers le programme RPG final et le handler sont passées sous la forme d'une structure de données
  - Cette structure de données est définie dans le membre source QRNOPENACC dans les fichiers source QRPGLESRC, QCBLLESRC et H de la bibliothèque QOAR
  - Il est possible de passer au handler des données autres que celles liées aux opérations d'E/S. Il suffit d'indiquer le nom d'une structure de données comme second paramètre (optionnel) du mot-clé handler :

```
Ffichier1 if  e  k  disk    handler('bib1/pgm1':ds1)
```

# Fichiers en Open Access versus fichiers de type SPECIAL

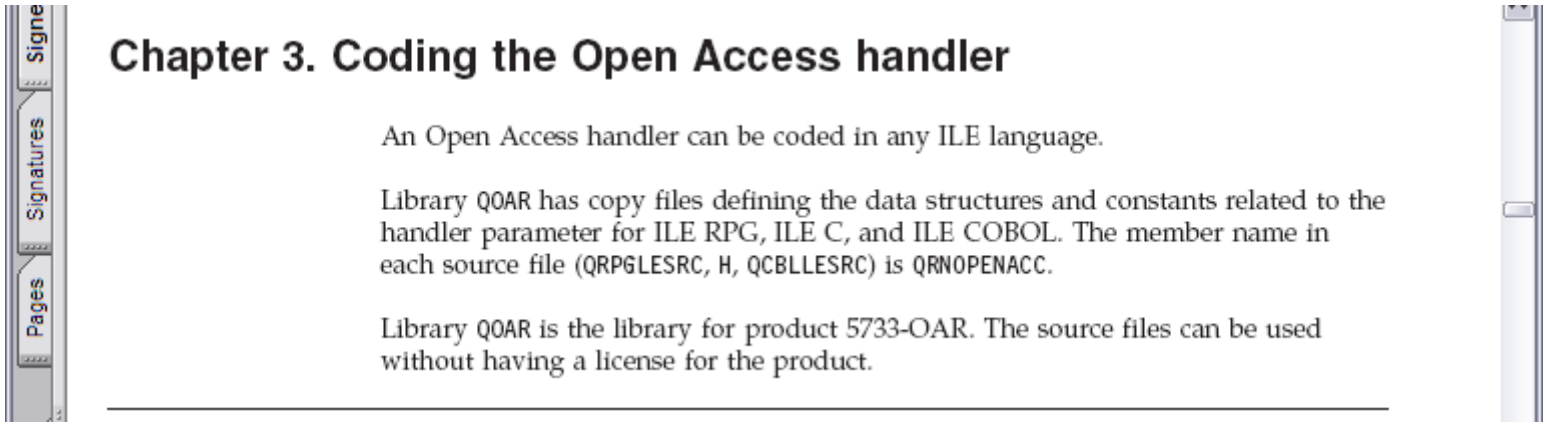
- Le principe est le même : appel d'un programme externe pour gérer les opérations d'E/S sur le fichier

- Différences
  - Les seules opérations possibles sur un fichier SPECIAL sont celles valables pour un fichier séquentiel (OPEN, READ, WRITE, UPDATE, DELETE, CLOSE, FEOD). Un fichier « Open Access » peut être utilisé pour tout type d'unité (donc codes-opération CHAIN, READE, SETLL… possibles)
  - Avec un fichier SPECIAL, peu d'informations sont passées au handler (type d'opération et enregistrements). Avec un fichier Open Access, des informations comme le noms du fichier et des formats, le nom et le type des zones, etc. sont passées au handler
  - Réciproquement, en retour, avec un fichier SPECIAL, il n'est possible de récupérer que le code retour. Avec un fichier Open Access, il est possible de récupérer le code status RPG, le RRN, la touche de fonction utilisée, etc.
  - Avec un fichier SPECIAL, le handler est obligatoirement un programme. Avec un fichier Open Access, le handler peut être une procédure

# RPG Open Access – Comment coder le handler ?

- Documentation





# RPG Open Access – La DS passée au handler – 1

- Se nomme QrnOpenAccess_T
- Contient :
  - Le type d'opération réalisée sur le fichier (READ, WRITE…)
    - Récupérée sous forme d'une valeur numérique dans la zone *rpgOperation*
    - Des constantes associées à ces valeurs numériques simplifient la lisibilité

QRNOPENACC.RPGLE

```
Ligne 56      Colonne 1     Replacer
      ....DName++++++++++++++++++++++++++++++++++++++++
000056     D QrnOperation_OPEN...
000057     D                 C                   1
000058     D QrnOperation_POSITION_START...
000059     D                 C                   2
000060     D QrnOperation_POSITION_END...
000061     D                 C                   3
000062     D QrnOperation_READ...
000063     D                 C                   4
000064     D QrnOperation_READC...
000065     D                 C                   5
000066     D QrnOperation_READE...
000067     D                 C                   6
000068     D QrnOperation_READP...
000069     D                 C                   7
000070     D QrnOperation_READPE...
```

# RPG Open Access – La DS passée au handler – 2

- Des zones récupérées par le handler sur le contexte de l'opération
  - Nom des bibliothèque, fichier et membre
  - Est-ce que le fichier est sur clé, est décrit en externe,etc. ?
  - *keyedFile, externalFile, compileFile, recordLevels,* etc…

- Des zones indiquant le « résultat » de l'opération et qui sont alimentées par le handler avant de retourner au programme RPG
  - *rpgStatus* : code d'erreur (0 ou entre 1000 et 2000)
  - *eof* : fin de fichier (sur opération READ)
  - *found* : enregistrement trouvé (sur opération CHAIN, SETLL, SETGT)
  - *equal, rrn, functionKey, deviceFeedback/ioFeedback/openFeedback* (INFDS), etc.

```
/* O   F1-F24, PRINT,ROLLUP etc   */
/*     See QrnFunctionKey_*       */
/*     - Unknown values cause an  */
/*       RPG exception            */
D   functionKey...
D                                 3U 0
```

```
     Colonne 1      Replacer
..DName++++++++++++++++++++++++++++++++++++++
 D QrnFunctionKey_None...
 D                   C                   0
 D QrnFunctionKey_01...
 D                   C                   1
 D QrnFunctionKey_02...
 D                   C                   2
 D QrnFunctionKey_03...
 D                   C                   3
```

# RPG Open Access – La DS passée au handler – 3

- Un pointeur (*stateInfo*) vers l'adresse d'une structure de données dont l'état sera maintenu lors de tous les appels au handler par le programme RPG
- Des pointeurs (*inputBuffer* et *outputBuffer*) vers des structures de données représentant les buffers d'entrée et sortie du fichier ou du format

```
D                                 *
/* O   Input buffer               */
/*     NULL if not used by opcode */
/*     - Length is given by       */
/*       inputBufferLen           */
D   inputBuffer...
```

```
D                                 *
/* I   Output buffer              */
/*     NULL if not used by opcode */
/*     - Length is given by       */
/*       outputBufferLen          */
D   outputBuffer...
D                                 *
```

- Un pointeur (*namesValues*) vers une structure de données (*QrnNamesValues_T*) contenant la liste des noms, types et valeurs des zones du format

```
/* I/O Alternate version of       */
/*     I/O buffer information     */
D   namesValues...
```

```
D QrnNamesValues_T...
D                 DS                  QUALIFIED TEMPLATE ALIGN
D   num                           10I 0
D   field                             LIKEDS(QrnNameValue_T)
D                                     DIM(32767)
```

```
D QrnNameValue_T...
D                 DS                  QUALIFIED TE
/* I   Name from external file    */
D   externalName...
D                                 10A
/* I   Data type of field         */
/*     See QrnDatatype_*          */
D   dataType...
D                                 3U 0
/* I   Defined length of numeric  */
/*      - Decimal: total digits   */
```

# RPG Open Access – La DS passée au handler – 4

- Il existe donc deux façons d'alimenter / récupérer les données :
  - Via des « record buffers »
  - Via un tableau « Nom / valeur » (nom de la zone / valeur de la zone)

- On choisit en positionnant la valeur *useNamesValues* à *ON ("0") ou à *ON ("1") :
  - *OFF : record buffers (valeur par défaut)
  - *ON : tableau « Nom / Valeur »
    - Ce tableau permet en fait de récupérer :
      - Nom
      - Type
      - Longueur
      - Nombre de décimales
      - Format (date/heure)
      - Séparateur (date/heure)
      - Accepte la valeur indéfinie ?
      - Contient la valeur indéfinie ?
      - CCSID
      - Valeur
      - etc.

# RPG Open Access – La DS passée au handler – Doc

## Chapter 5. Handler parameter

The same parameter is passed to the handler for all operations to the file.

The parameter is a data structure with several types of subfields:

## The subfields of the main parameter structure

The subfields of the main parameter structure

The name of the main parameter structure is QrnOpenAccess_T.

*Table 1. Subfields of QrnOpenAccess_T*

| Subfield | Type | Set by | Used by |
|---|---|---|---|
| *structLen* | UINT4 | RPG | Handler |
| *parameterFormat* | CHAR(8) | RPG | Handler |
| *userArea* | Pointer [1] | RPG | Handler and RPG programmer |
| *stateInfo* | Pointer [3] | Handler | Handler |
| *recordLevels* [4] | Pointer [1] | RPG | Handler |
| *inputBuffer* | Pointer [1, 2] | Handler | RPG |
| *inputNullMap* [4] | Pointer [1, 2] | Handler | RPG |
| *outputBuffer* | Pointer [1, 2] | Handler | RPG |
| *outputNullMap* [4] | Pointer [1, 2] | Handler | RPG |

## Descriptions of the subfields

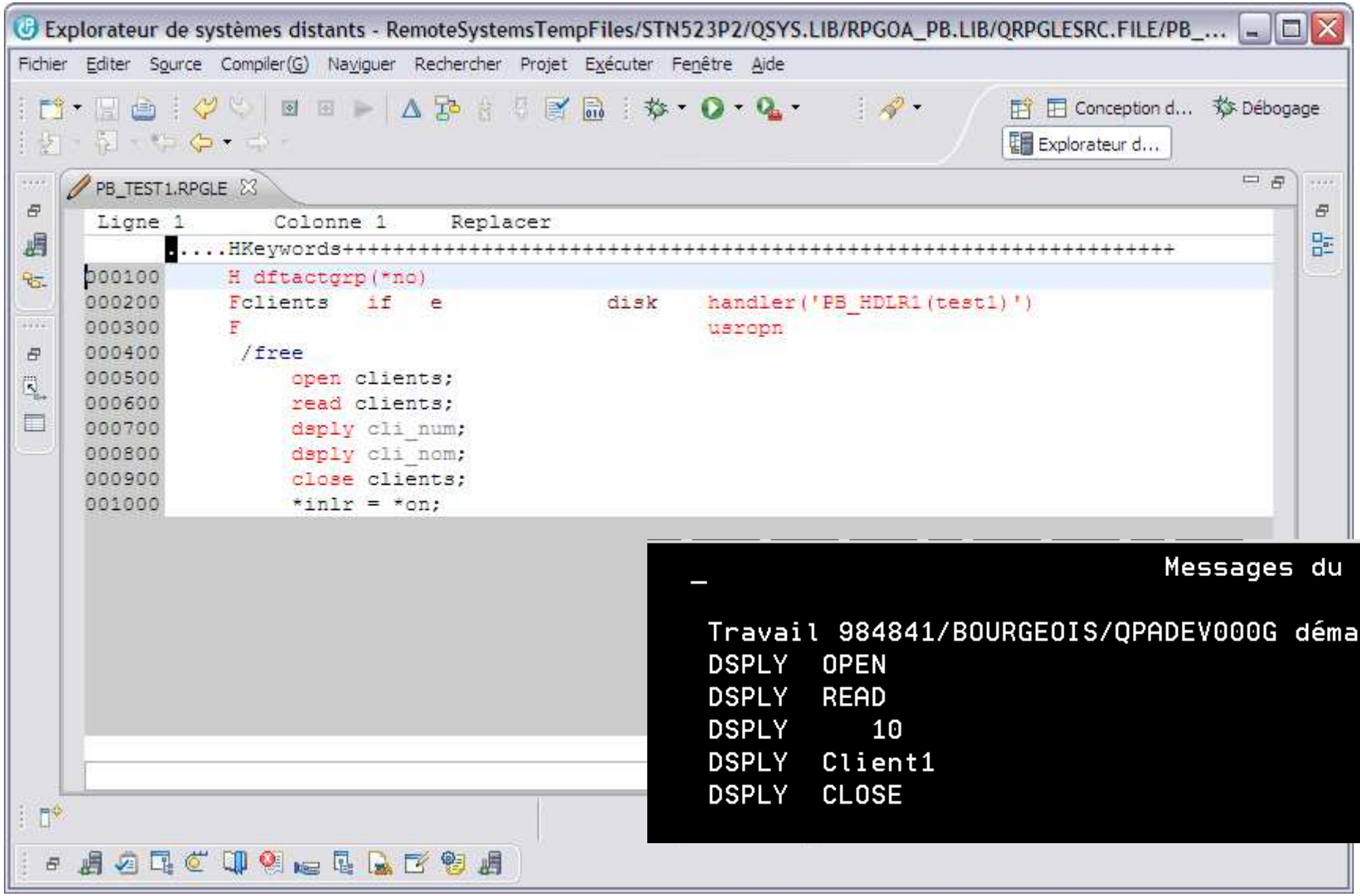
*blocked*
'1' if the file is defined to be blocked in the

**Note:** This value is provided for informatic
does not work with blocks of record, but th
to control whether it block the records from
dealing with.

*commit*
'1' if the file should be opened under comm

*compileFile*
The library and file that the RPG compiler
description of an externally-described file.

# RPG Open Access – 1er exemple – Programme RPG



# RPG Open Access – 1er exemple – Programme handler

PB_HDLR1.RPGLE

```
Ligne 1       Colonne 1     Replacer
      ....HKeywords+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000100     H nomain
000200      /copy qoar/qrpglesrc,qrnopenacc
000300
000400     D test1           pr                  extproc('test1')
000500     D   ds_oa                             likeds(QrnOpenAccess_T)
000600
000700     P test1           b                   export
000800     D                 pi
000900     D   ds_oa                             likeds(QrnOpenAccess_T)
001000     D
001100     D client_ds     e ds                  extname('CLIENTS') qualified
001200     D                                     based(p_client_ds)
001300      /free
001400         if ds_oa.rpgOperation = QrnOperation_OPEN;
001500          dsply 'OPEN';
001600
001700         elseif ds_oa.rpgOperation = QrnOperation_CLOSE;
001800          dsply 'CLOSE';
001900
002000         elseif ds_oa.rpgOperation = QrnOperation_READ;
002100          dsply 'READ';
002200          p_client_ds = ds_oa.inputBuffer;
002300          client_ds.cli_num = 10;
002400          client_ds.cli_nom = 'Client1';
002500
002600         else;
002700            ds_oa.rpgStatus = 1299;
002800         endif;
002900      /end-free
003000     P test1           e
```

# RPG Open Access – 2nd exemple – Programme RPG

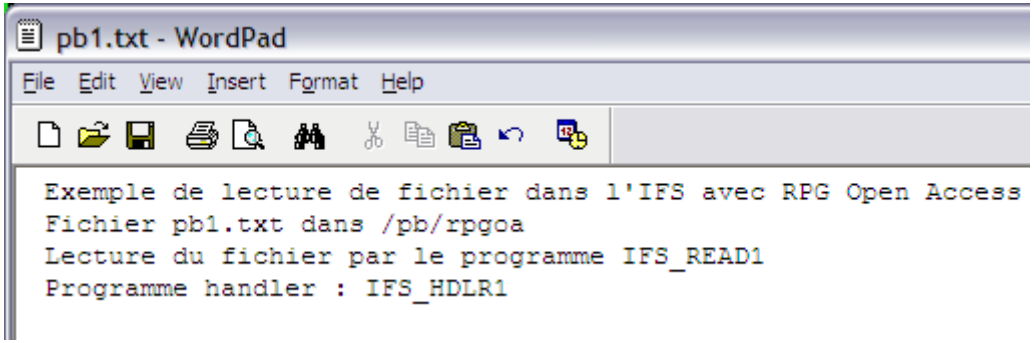
IFS_READ1.RPGLE

```
Ligne 1       Colonne 1     Replacer
      .........1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
000100       * Rational Open Access: RPG Edition
000200       * Programme de lecture d'un fichier dans l'IFS - Utilisation d'un handler
000300       * Le fichier IFS_F comporte une zone alpha nommée line de longueur 32740
000400       * Le programme lit les lignes du fichier de l'IFS comme s'il lisait les
000500       * lignes du fichier DISK et en imprime les 300 premiers caractères
000600
000700     H dftactgrp(*no)
000800
000900     FIFS_F     if   e             disk    handler('IFS_HDLR1(read_ifs)')
001000     F                                     usropn
001100     FQSYSPRT   o    f  300        printer
001200
001300     D ds_ifs          ds            300
001400      /free
001500         monitor;
001600           open ifs_f;
001700           read ifs_f;
001800           dow not %eof;
001900             ds_ifs = line;
002000             write qsysprt ds_ifs;
002100             read ifs_f;
002200           enddo;
002300         on-error 1217;
002400           dsply 'Erreur lors de l''ouverture/fermeture du fichier IFS';
002500         on-error 1299;
002600           dsply 'Tentative d''opération non supportée sur fichier IFS';
002700         endmon;
002800          *inlr = *on;
002900      /end-free
```

IFS_READ1.RPGLE | IFS_F.PF

```
Ligne 1       Colonne 1     Replacer
      .....+A*..1....+....2....+....3....+....4....+....5....+..
000100       * Ce fichier ne contient pas de données
000200       * Il est utilisé comme interface pour le handler
000300     A          R IFS_REC
000400     A            LINE        32740A         VARLEN
```

# RPG Open Access – 2nd exemple – Résultat




# RPG Open Access – 2nd exemple – Programme handler - 1

IFS_READ1.RPGLE | IFS_F.PF | IFS_HDLR1.RPGLE

Ligne 1 Colonne 1 Replacer

```
     .........1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
000100       * Programme handler pour lecture d'un fichier dans l'IFS
000200       * Dans cet exemple, le nom du fichier est défini en dur
000300       * Mais il pourrait être passé en paramètre (par la zone userArea de la DS)
000400       * Hypothèses :
000500       *  - le fichier IFS existe
000600       *  - les lignes du fichier ne dépassent pas 32740 caractères
000700
000800      H nomain
000900       /copy qoar/qrpglesrc,qrnopenacc
001000       /copy qsysinc/qrpglesrc,ifs
001100
001200      D fichier_ifs     c                   '/pb/rpgoa/pb1.txt'
001300
001400      D buffer_t      e ds                  extname('IFS_F') template
001500       * (IFS_F est le fichier DISK utilisé dans le programme RPG)
001600
001700      D state_t         ds                  qualified template
001800      D   code                        10i 0
001900       *---------------------------------------------------------------------
002000       * Handler appelé par le programme RPG
002100      D read_ifs        pr                  extproc('read_ifs')
002200      D   ds_oa                             likeds(QrnOpenAccess_T)
002300       *---------------------------------------------------------------------
002400       * Procédures de gestion des fichiers dans l'IFS
002500      D openFile        pr            10i 0
002600      D   path                     5000a   varying const
002700
002800      D closeFile       pr
002900      D   status                    10i 0 value
003000
```

# RPG Open Access – 2nd exemple – Programme handler - 2

IFS_READ1.RPGLE | IFS_F.PF | IFS_HDLR1.RPGLE

Ligne 31 Colonne 1 Replacer

```
     .....DName++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++
003100     D readFile        pr                n
003200     D   status                       10i 0 value
003300     D   buffer                             likeds(buffer_t)
003400      *---------------------------------------------------------
003500      * Handler appelé par le programme RPG
003600     P read_ifs        b                    export
003700     D                 pi
003800     D   ds_oa                              likeds(QrnOpenAccess_T)
003900
004000     D state           ds                   likeds(state_t)
004100     D                                      based(p_state)
004200     D
004300     D buffer          ds                   likeds(buffer_t)
004400     D                                      based(p_buffer)
004500      /free
004600         p_state = ds_oa.stateInfo;
004700
004800          // Ouverture du fichier dans l'IFS
004900         if ds_oa.rpgOperation = QrnOperation_OPEN;
005000            p_state = %alloc(%size(state));
005100            clear state;
005200            ds_oa.stateInfo = p_state;
005300            state.code = openFile(fichier_ifs);
005400            if state.code < 0;
005500               ds_oa.rpgStatus = 1217; // erreur d'ouverture
005600            endif;
005700
005800          // Lecture du fichier dans l'IFS
005900         elseif ds_oa.rpgOperation = QrnOperation_READ;
006000            p_buffer = ds_oa.inputBuffer;
006100            ds_oa.eof = readFile (state.code : buffer);
```

# RPG Open Access – 2nd exemple – Programme handler - 3

IFS_READ1.RPGLE | IFS_F.PF | IFS_HDLR1.RPGLE

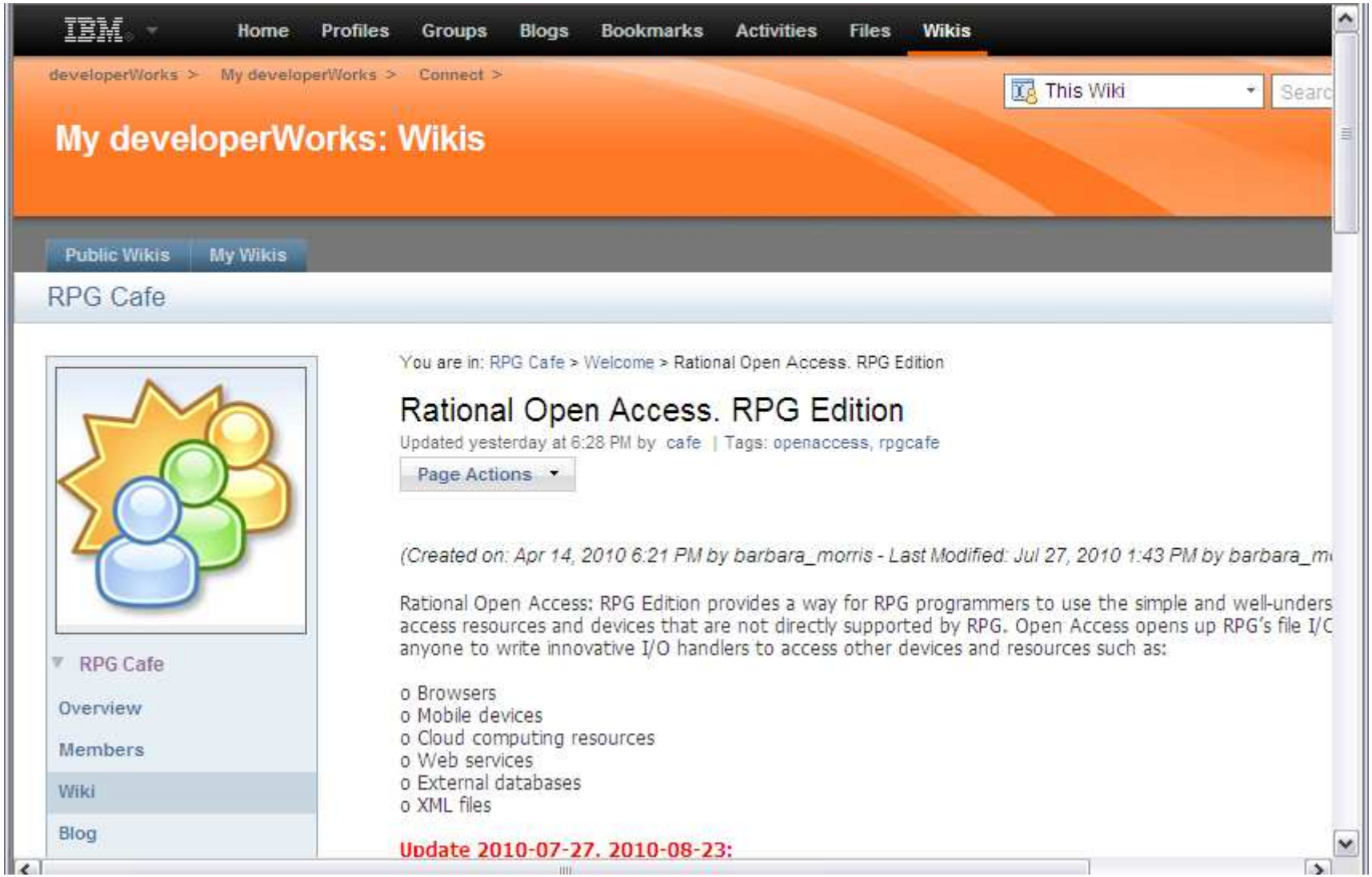
Ligne 63 Colonne 1 Replacer

```
006300           // Fermeture du fichier IFS
006400          elseif ds_oa.rpgOperation = QrnOperation_CLOSE;
006500             closeFile (state.code);
006600             state.code = -1;
006700             dealloc(n) p_state;
006800             ds_oa.stateInfo = *null;
006900
007000           // Si autre opération : erreur
007100          else;
007200             ds_oa.rpgStatus = 1299;
007300          endif;
007400       /end-free
007500      P read_ifs        e
007600
007700       *-------------------------------------------------------------------------
007800       * Procédure d'ouverture d'un fichier dans l'IFS
007900      P openFile        b
008000      D                 pi            10i 0
008100      D   path                       5000a   varying const
008200       /free
008300          // Appel de la procédure open (voir qsysinc/qrpglesrc,ifs)
008400          return open(path : O_RDONLY + O_TEXTDATA);
008500       /end-free
008600      P openFile        e
```

# RPG Open Access – 2nd exemple – Programme handler - 4

```
008800      * Procédure de fermeture d'un fichier dans l'IFS
008900     P closeFile       b
009000     D                 pi
009100     D   status                      10i 0 value
009200     D rc              s             10i 0
009300      /free
009400         // Appel de la procédure close (voir qsysinc/qrpglesrc,ifs)
009500         rc = close(status);
009600      /end-free
009700     P closeFile       e
009800      *---------------------------------------------------------------
009900      * Procédure de lecture d'un fichier dans l'IFS
010000     P readFile        b
010100     D                 pi              n
010200     D   status                      10i 0 value
010300     D   buffer                            likeds(buffer_t)
010400
010500     D numBytes        s             10i 0
010600     D z1a             s              1a
010700     D CR              c                   x'0A'
010800     D LF              c                   x'25'
010900      /free
011000         buffer.line = '';
011100         // Appel de la procédure read (voir qsysinc/qrpglesrc,ifs)
011200         numBytes = read(status : %addr(z1a) : %size(z1a));
011300         if numBytes = 0;
011400            return '1';  // fin de fichier atteinte
011500         endif;
011600         dow (numBytes > 0) and (z1a <> CR) and (z1a <> LF);
011700            buffer.line += z1a;
011800            numBytes = read(status : %addr(z1a) : %size(z1a));
011900         enddo;
012000         return '0';  // fin de fichier non atteinte
012100      /end-free
012200     P readFile        e
```

# RPG Open Access – Coding des handlers - Pour en savoir plus



# RPG Open Access – Coding des handlers - Pour en savoir plus

IBM i AIX MAINFRAME POWER NEWSLETTERS

IBM Systems magazine

SUBSCRIBE
CURRENT ISSUE
ARCHIVE

IBM i
ALL EDITIONS

ADMINISTRATOR | DEVELOPER | TRENDS | TIPS & TECHNIQUES | CASE STUDIES | STORAGE | PRODUCT NEWS | ENDPGM

Developer > RPG

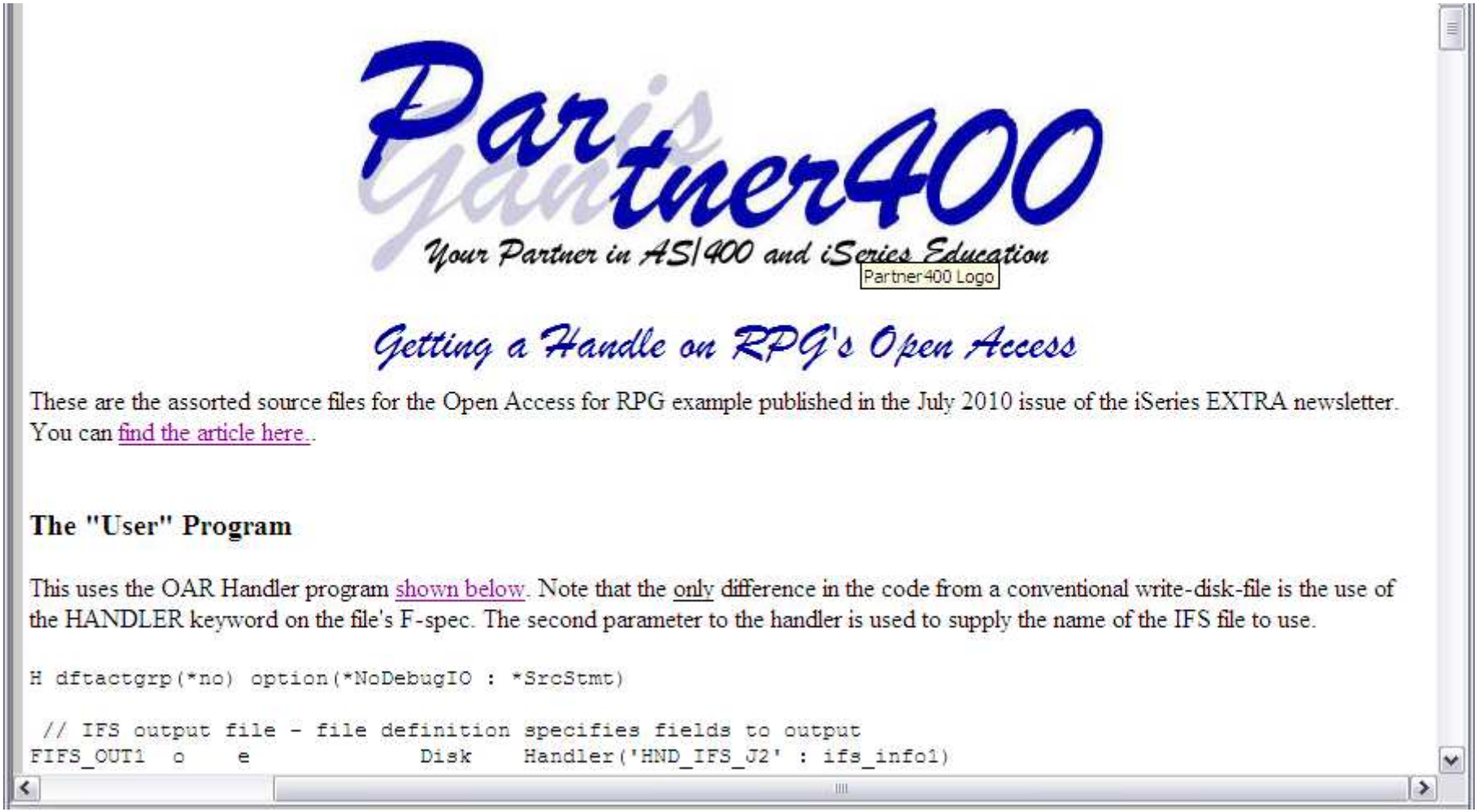
e-Newsletter Exclusive

## Getting a Handle on RPG's Open Access

Discover how simple it is to create a generic handler

July 2010 | by Jon Paris and Susan Gantner

# RPG Open Access – Coding des handlers - Pour en savoir plus



Partner400 Logo

Your Partner in AS/400 and iSeries Education

## Getting a Handle on RPG's Open Access

These are the assorted source files for the Open Access for RPG example published in the July 2010 issue of the iSeries EXTRA newsletter. You can find the article here..

### The "User" Program

This uses the OAR Handler program shown below. Note that the only difference in the code from a conventional write-disk-file is the use of the HANDLER keyword on the file's F-spec. The second parameter to the handler is used to supply the name of the IFS file to use.

```
H dftactgrp(*no) option(*NoDebugIO : *SrcStmt)

 // IFS output file - file definition specifies fields to output
FIFS_OUT1  o    e             Disk    Handler('HND_IFS_J2' : ifs_info1)
```

# RPG Open Access – Coding des handlers - Pour en savoir plus

VOLUBIS

CONSEIL ET FORMATION SUR I5/OS ET OS/400

VOLUBIS Devenez un expert

INFOS DU MOIS | LE MOIS DERNIER | PAUSE-CAFE | TESTS | LIENS | AF400 | SPLF2 | FORUM | CONTACTS

À LA UNE
Informations du mois / Nouveautés

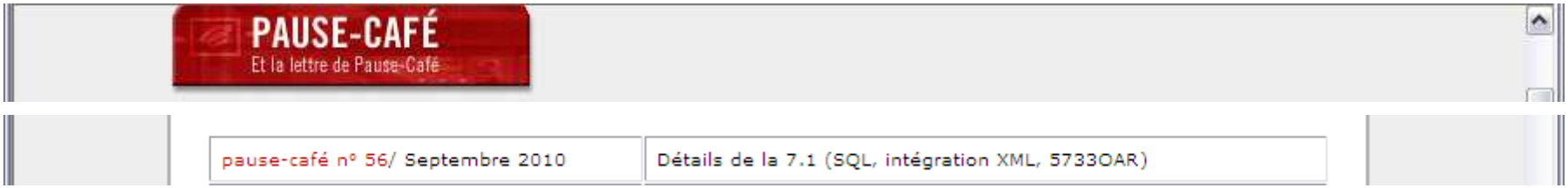
**Rational Open Access, RPG edition (5733OAR)**

Ce produit permet d'associer à une déclaration de fichier, une routine (écrite ou achetée) qui sera utilisée à la place des routines standard lors des entrées/sorties.

Voici ce que dit l'annonce *"Simplifies the development of transaction processing applications for mobile devices and web services when developing with Rational Developer for Power. Removes the requirement for the 5250 data stream"*

Pour faire fonctionner ce module (facturable) il faut les PTF **SI39480** et **SI39912**

Voyez dès à présent nos premiers tests sur le sujet !



PAUSE-CAFÉ
Et la lettre de Pause-Café

pause-café n° 56/ Septembre 2010 | Détails de la 7.1 (SQL, intégration XML, 5733OAR)

# Rational Open Access: RPG Edition

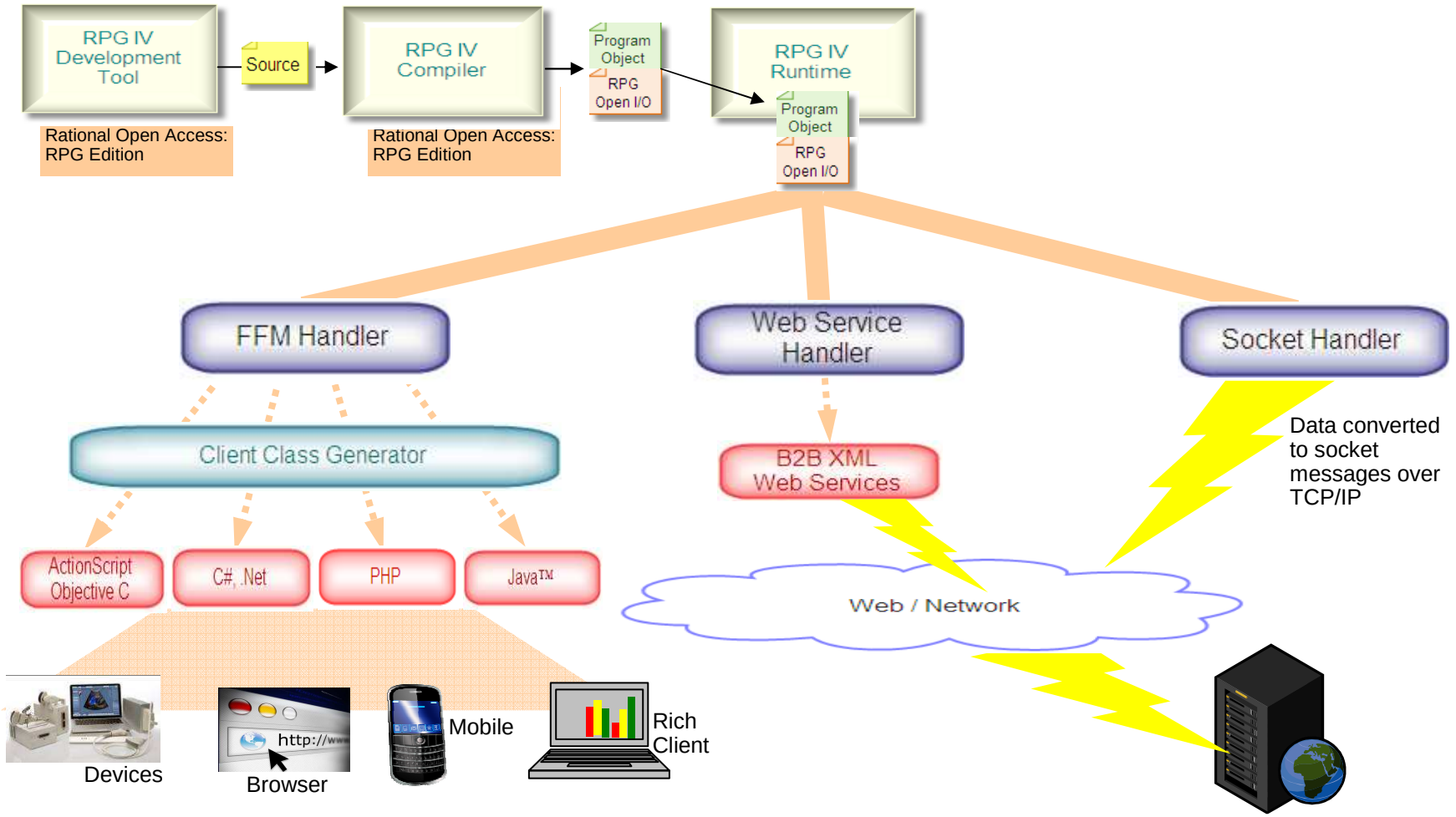
## *Quelques mots sur les handlers de IBM Lab Services*

# IBM Lab Services et RPG Open Access

- Les missions / prestations de IBM Lab Services
  - Objectif : Faciliter l'adoption des technologies et des produits IBM
  - Prestations : de service (étude d'architecture, implémentation de solutions, développement, étude de performances…)
  - IBM La Gaude, à côté de Nice
  - Compétences techniques de haut niveau, en relation avec les labs

- IBM Lab Services a développé 3 handlers RPG Open Access :
  - FFM (File Format Messaging) handler
    - Interfaçage d'applications RPG avec des applications Java, PHP, C#
  - Web Services handler
    - Connexion d'applications RPG à des interfaces B2B (appel de Services Web externes, intégration avec des applications externes, etc.)
  - Sockets handler
    - Serveur de sockets TCP/IP

# Trois handlers pour interfacer les applications RPG



# RPG OA – Quelques mots sur le handler FFM

- Objectif
  - Faciliter la communication (conversations/transactions) entre des programmes RPG et des programmes Java, PHP, C#, etc.

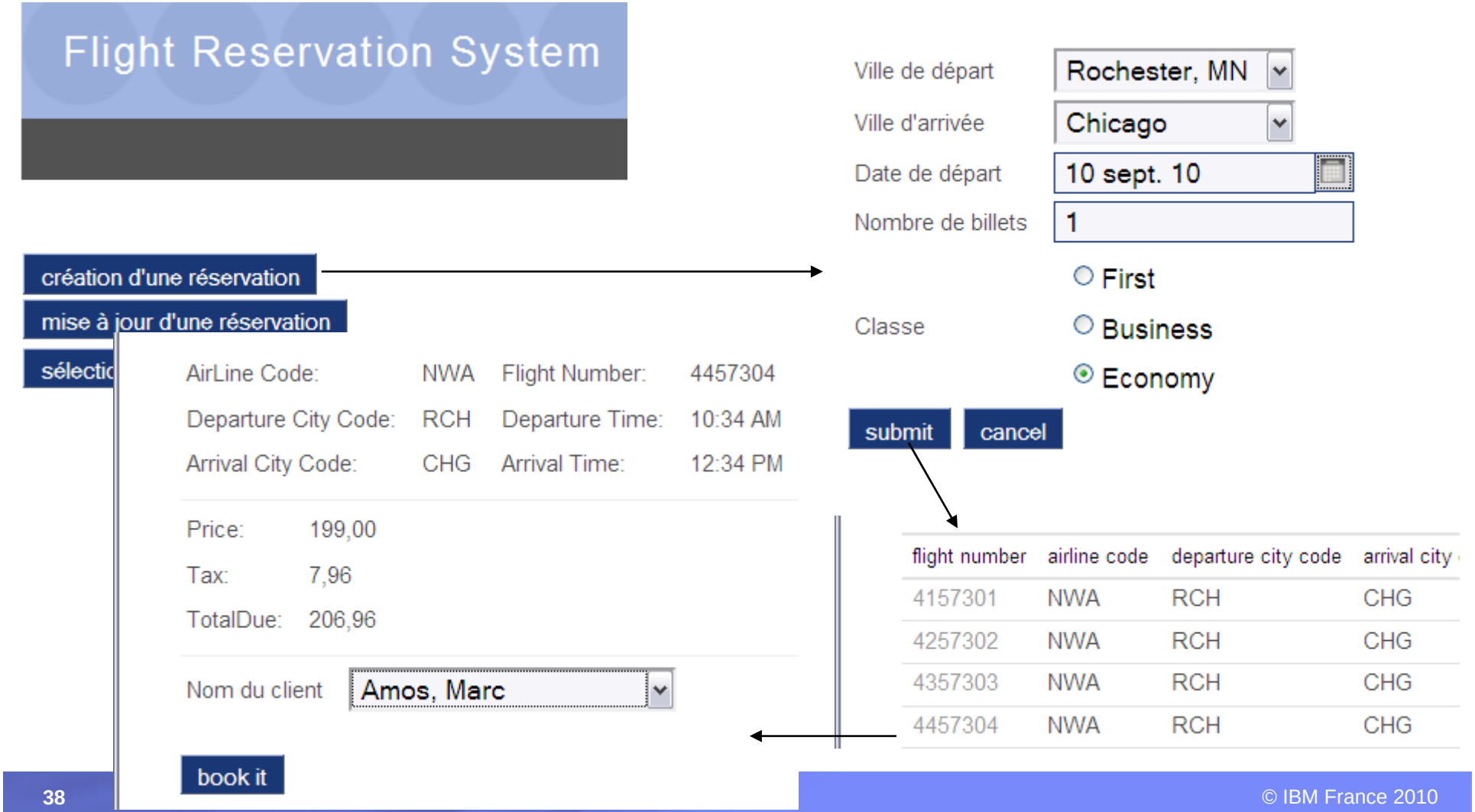
- Principe
  - Le handler FFM est un framework/toolkit permettant de définir des transactions client-serveur basées sur des messages
    - Le modèle recommandé est de définir des transactions « stateless » (événementiel, pools de connexion)
    - Les messages sont définis à partir de fichiers IBM i (fichiers écran, fichiers ICF, etc.)
    - Côté client, le toolkit permet de générer des classes (Java, PHP, C#) à partir des définitions des formats de ces fichiers
    - Côté serveur, le programme RPG utilise des ordres READ et WRITE

# RPG OA – Quelques mots sur le handler FFM

- Le handler FFM :
  - Ne permet pas de générer automatiquement des interfaces Web, Web 2.0, mobile, etc.
  - Nécessite des compétences RPG pour développer la partie serveur et des compétences Java, PHP ou C# pour développer la partie client
  - A pour objectif d'aider les développeurs à mettre en place une architecture client-serveur basée sur des messages :
    - Le toolkit inclut des classes de base pour gérer la communication entre le client et le serveur (connexion, transformation des données, conversations, etc.)
    - Le toolkit permet de générer des classes à partir des formats de fichiers IBM i
      - La conversion des données est gérée par le framework
    - Côté serveur il suffira de développer un programme RPG « dispatcher »
      - Récupération du nom du format et exécution de l'opération associée

# RPG OA – Le handler FFM – Exemple

- Application IBM FLGHT400 – Réservation de vols



# RPG OA – Le handler FFM – Exemple - Quelques détails

- Exemple de fichier requête
  *Fichier RQINFO*

```
A          R RQFLGHTINF
A            FLGHTNUM        7

A          R RQCITYNM
A            INITIALS        3
A            FROMTO          1

A          R RQCOMPPRC
A            BASEPRC         3
A            SERVCLSS        1
A            TICKETS         3  0

A          R RQFFLGHTDT
A            FROM           16
A            TO             16
A            MDY             8

A          R RQFFRMCITS
A            CITYNAME       16
A            LISTTYPE        1
A            COUNTREQ       10  0
```

- Exemple de fichier réponse
  Fichier *RSFLGHTINF*

```
A          R FLGHTINFR
A            AIRLINE         3
A            FLIGHT          7
A            FLIGHTDOW       2
A            DEPCITY         3
A            ARRCITY         3
A            DEPTIME         8
A            ARRTIME         8
A            PRICE           3
```

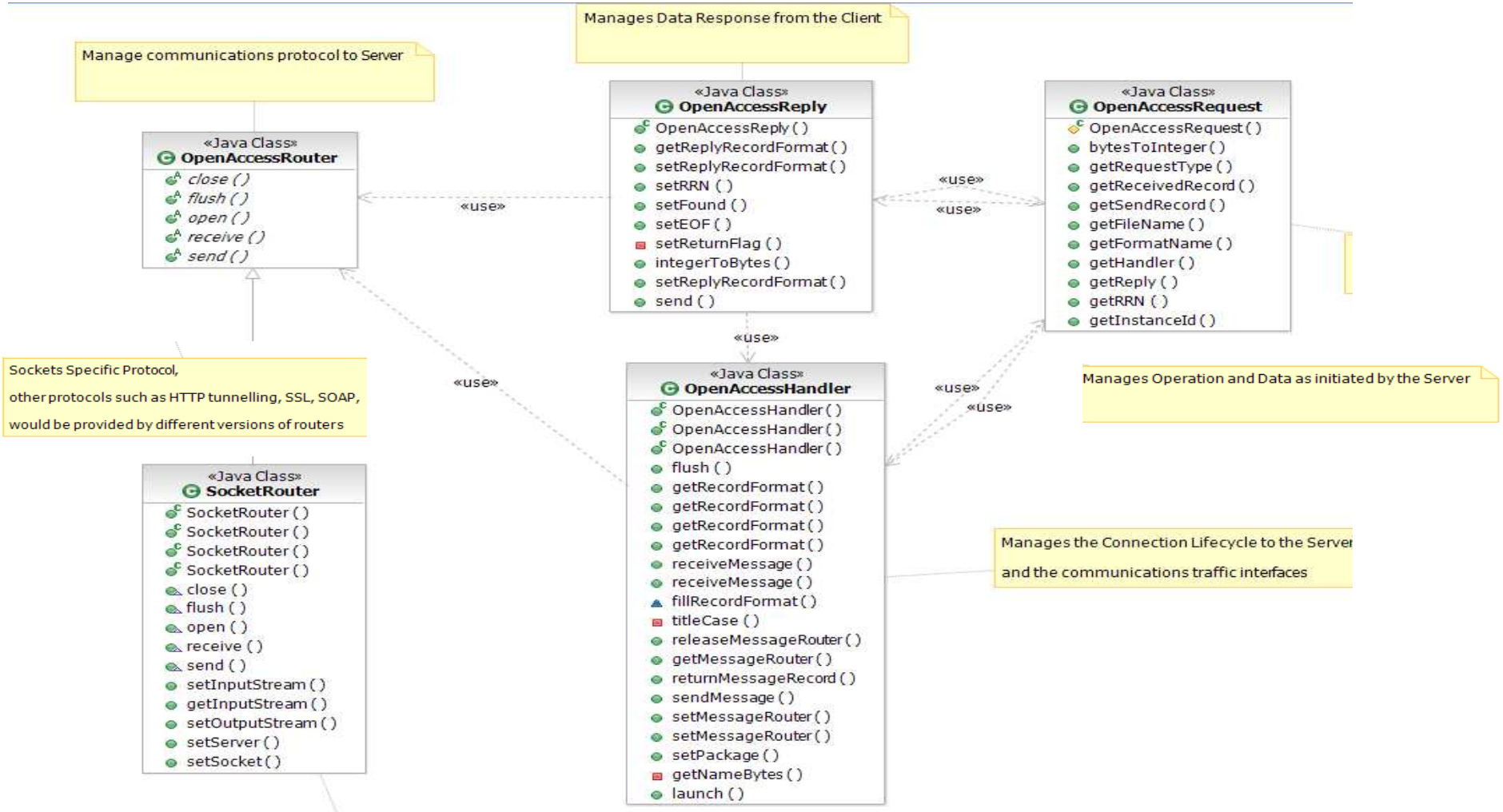
# RPG OA – Le handler FFM – Exemple - Quelques détails

- Côté client – Génération des classes à partir des fichiers IBM i
  - Alimentation des données en entrée à envoyer au READ RPG
  - Récupération des données en sortie envoyées par le WRITE RPG
  - Utilisées en conjonction avec les classes de base

```
                    Generate Open Access Class (GENCLSOA)

Type choices, press Enter.

Files  . . . . . .                          Name, generic*
  Library  . . . . . .      *LIBL           Name, *LIBL, *CURLIB
Package or namespace . . . . . .

Common package or namespace  . .   *LANGDFT

Path . . . . . .                   '.'

Generated source language  . . .   *JAVA      *JAVA, *CSHARP, *PHP
```

# RPG OA – Le handler FFM – Exemple - Quelques détails

- Côté client – Les classes de base fournies avec le handler



# RPG OA – Le handler FFM – Exemple - Quelques détails

- Côté serveur - Le programme RPG dispatcher – 1/2

F400DISP.RPGLE

Ligne 12 Colonne 79 Replacer

```
     .........1....+....2....+....3....+....4....+....5....+....6....+....7...
000203      * The request file is a multi-format file that contains separate
000204      * formats for each request type
000205      *
000206     fRQINFO    if   e             workstn handler(OAHANDLER)
000207     f                                     infds(iofeedback)
000208      *
000209      * The response files each contain one format (although they could
000210      * be combined)
000211      *
000600     fRSCITYINFOcf   e             workstn handler(OAHANDLER)
000601     fRSCITYNM  cf   e             workstn handler(OAHANDLER)
000800     fRSCUSTINF cf   e             workstn handler(OAHANDLER)
000801     fRSCUSTNM  cf   e             workstn handler(OAHANDLER)
001000     fRSFLGHTINFcf   e             workstn handler(OAHANDLER)
001001     fRSGETCNBR cf   e             workstn handler(OAHANDLER)
001002     fRSORDSUMM cf   e             workstn handler(OAHANDLER)
001003     fRSPRICEINFcf   e             workstn handler(OAHANDLER)
001004     fRSRESINFO cf   e             workstn handler(OAHANDLER)
001005     fRSRESFLGHTcf   e             workstn handler(OAHANDLER)
001006     fRSUPDORD  cf   e             workstn handler(OAHANDLER)
001200      *
001201      * All the files use the Lab Services handler
001202      *
001301     d OAHANDLER       c                   'QZRDOARPG/QZRDOASRV(HANDLER)'
001302      *
```

# RPG OA – Le handler FFM – Exemple - Quelques détails

- Côté serveur - Le programme RPG dispatcher – 2/2

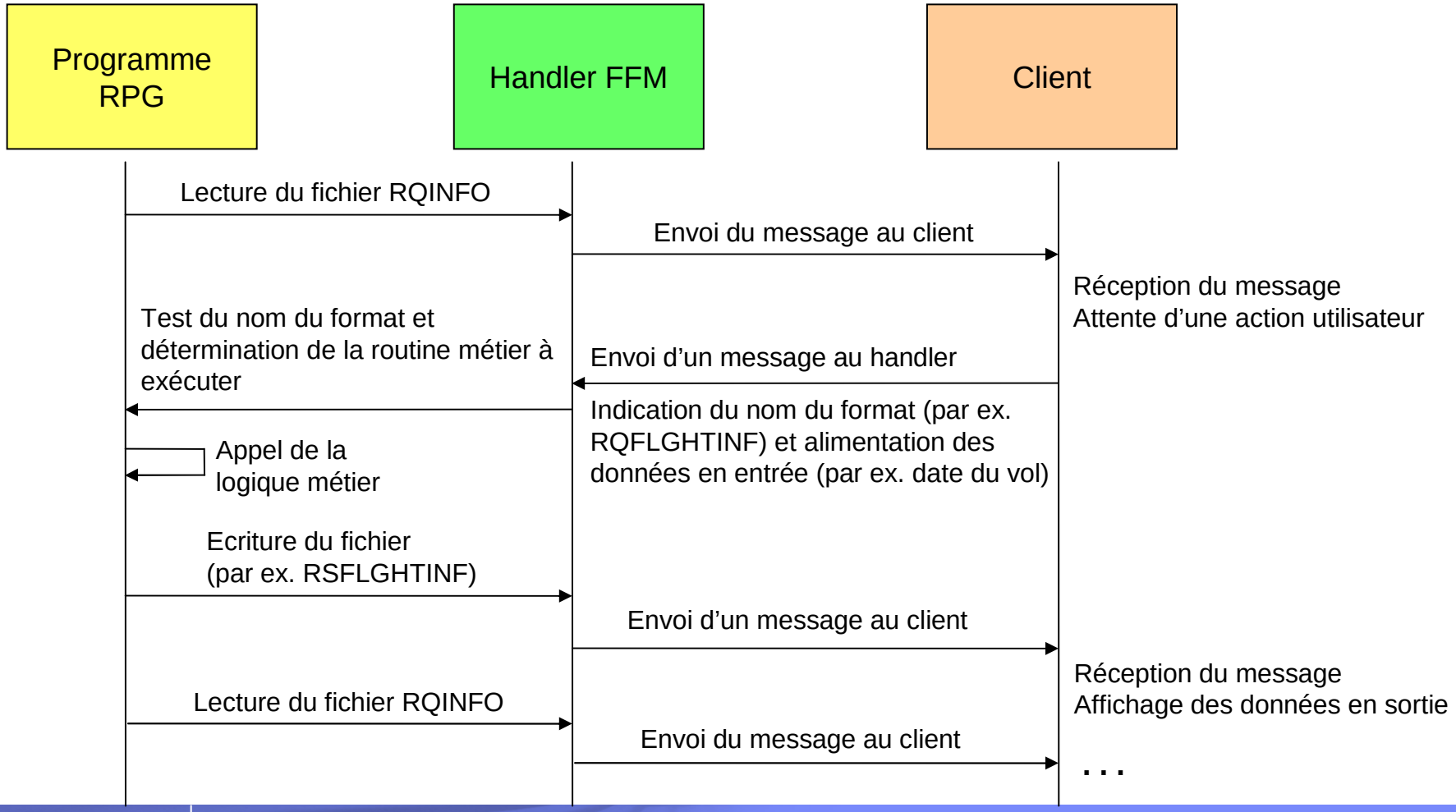
```
  Colonne 5     Replacer
...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
dou %error;                                  // Exit on error
  read(e) rqinfo;                            // Read from request file
  if not %error;
    select;
      when iofeedback.formatname = 'RQFLGHTINF'; // Flight Information
        GetFlightInfo(FLGHTNUM:FlightArray(1));
        WrtFlightInfo(FlightArray(1));
      when iofeedback.formatname = 'RQCITYNM';   // City Name
        GetCityName(INITIALS:FROMTO:CITYNAME);
        write(e) citynmr;
      when iofeedback.formatname = 'RQCOMPPRC';  // Calculate Price
        ComputePrice(BASEPRC:SERVCLSS:TICKETS:COMPPRICE:TAX:TOTAL);
        write(e) priceinfr;
      when iofeedback.formatname = 'RQFFLGHTDT'; // Fligh
        FindFlights(FROM:TO:MDY:count:FlightArray);
        for i = 1 to count;
          WrtFlightInfo(FlightArray(i));
        endfor;
      when iofeedback.formatname = 'RQFFRMCITS'; // Find
        FindFromCities(CITYNAME:LISTTYPE:
                       Limit(COUNTREQ:%elem(CityArray)):
                       count:CityArray);
        for i = 1 to count;
          WrtCityInfo(CityArray(i));
        endfor;
      when iofeedback.formatname = 'RQFINDCUST'; // Find Customers
        FindCustomers(CUSTNAME:LISTTYPE:
```

```
     ********************************************
p GetFlightInfo   b                   export
     ********************************************
d GetFlightInfo   pi
d  FlightNumber                  7    const
d  Flight                             likeds(FlightInfo)
d  FlightKey      s              9s 0
 /free
  FlightKey = %dec(FlightNumber:7:0);
  chain(e) FlightKey FlghtI;
  if %found;
    Flight.Airline = AIRLNM;
    Flight.Flight = %char(FLGHTN);
```

… / …

# RPG OA – Le handler FFM – Exemple - Quelques détails

- Schéma général des flux



# Annexe

## Rational Open Access: RPG Edition

*Le handler de **look**software*

# **Look**software et **itheis**

Modernisation d’applications
depuis 1995

Pionnier de
l’architecture
dynamique

Réutilisation est
notre niche

3000 clients et
partenaires dans
plus de 50 pays

1 Million d’utilisateurs connectés

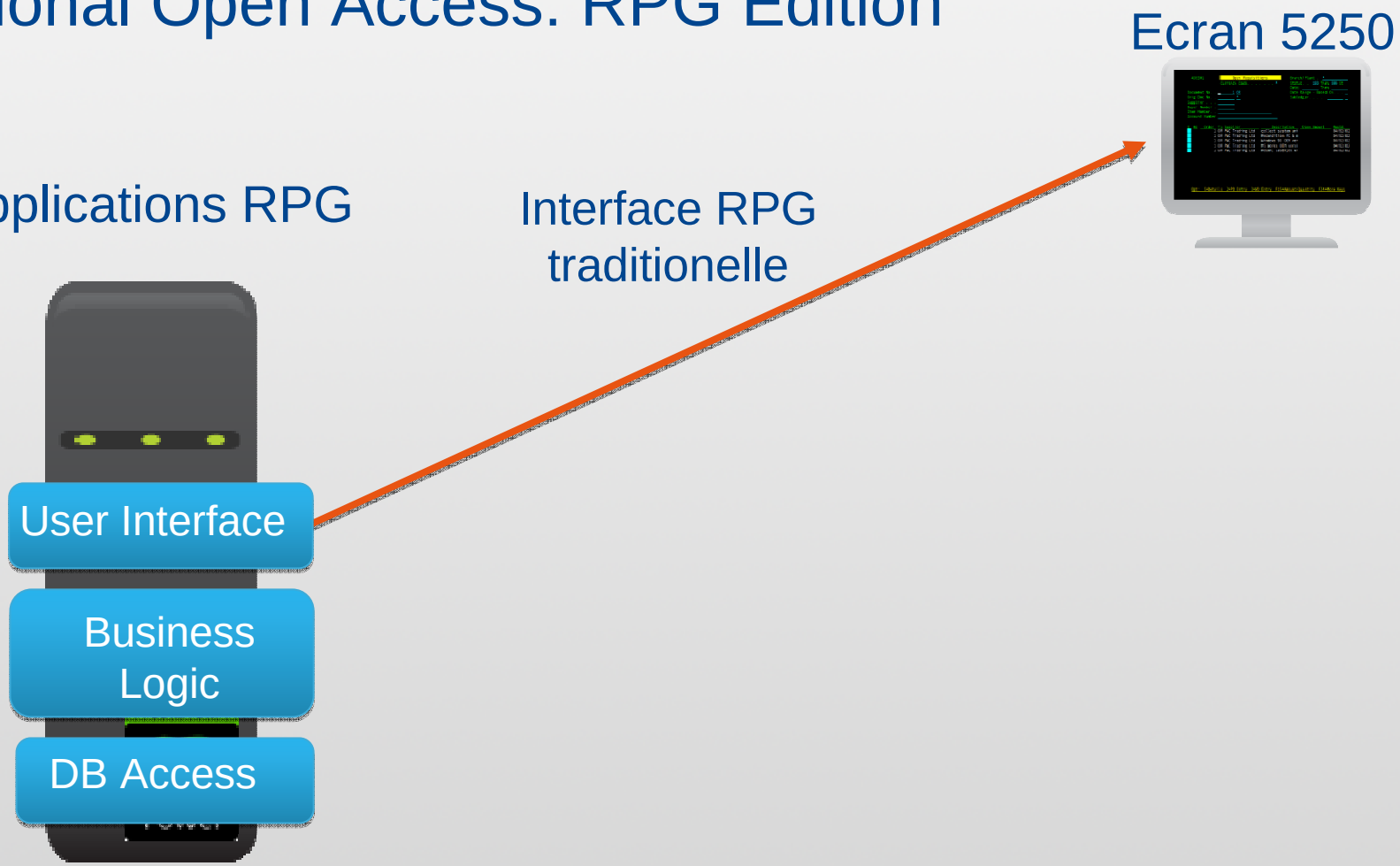
# **look**software – Les solutions


**new**look

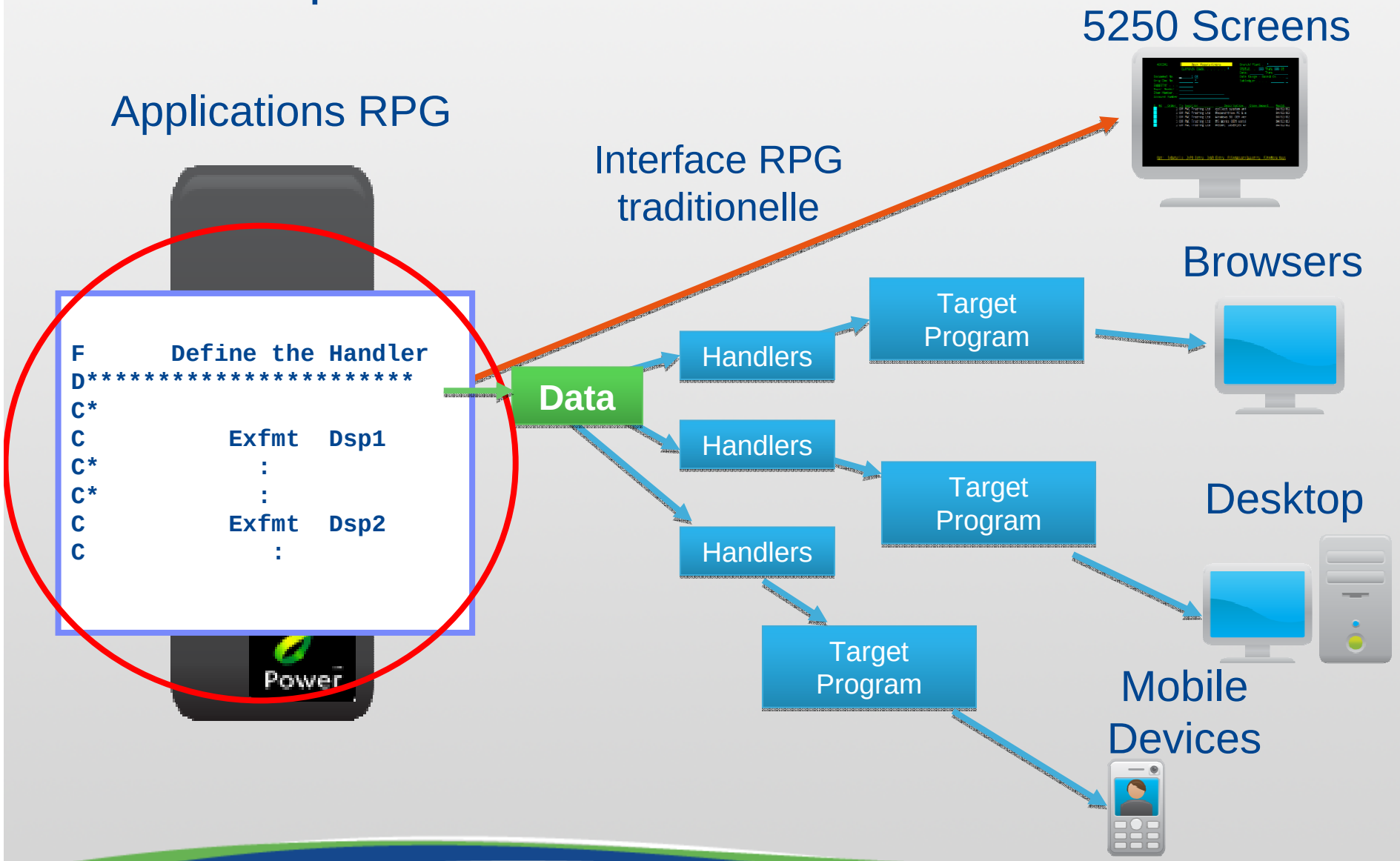
**soa**rchitect

RPG **look**server for Open Access RPG OA

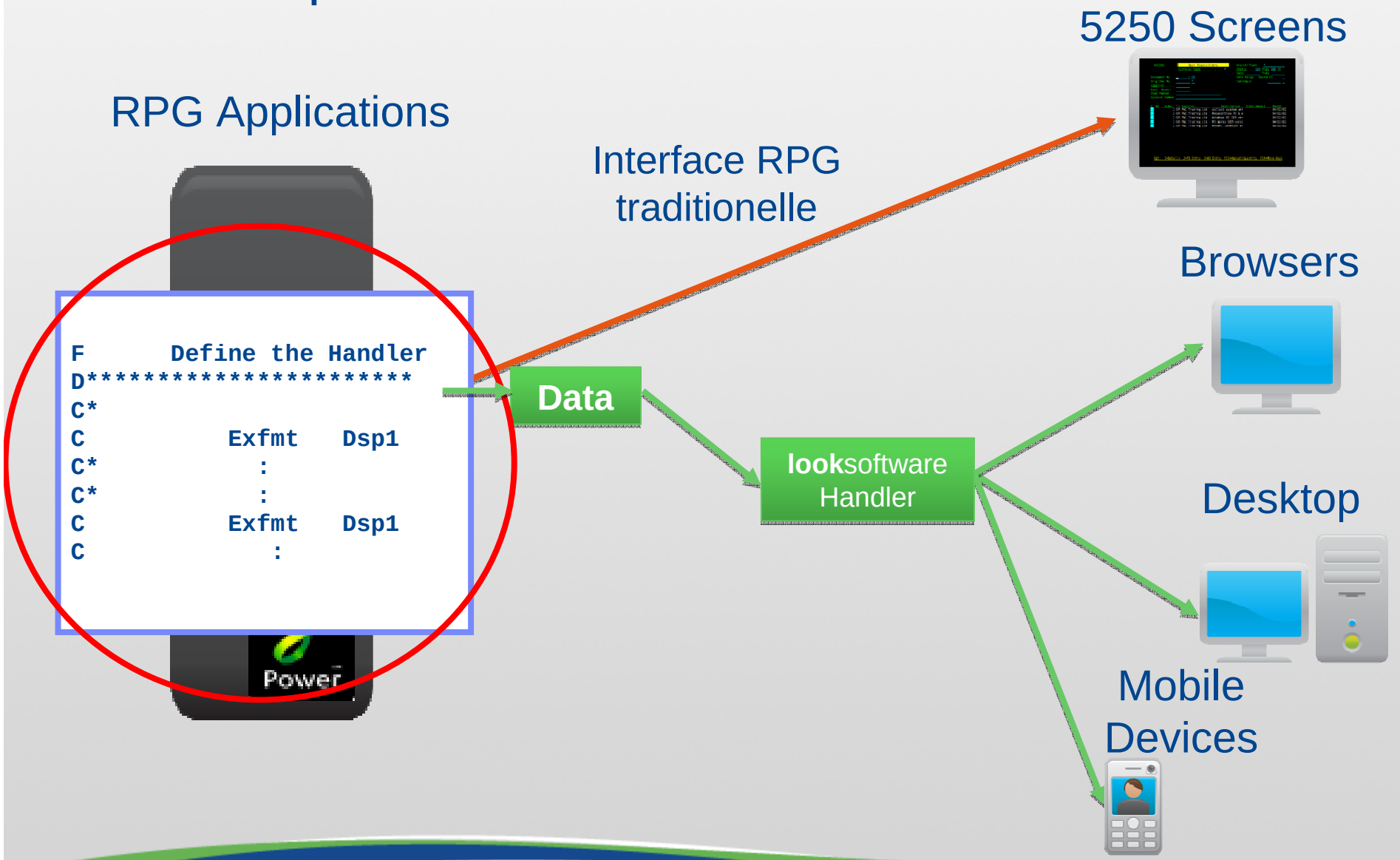
# Rational Open Access: RPG Edition



# Rational Open Access: RPG Edition



# Rational Open Access: RPG Edition



# Avant Open Access

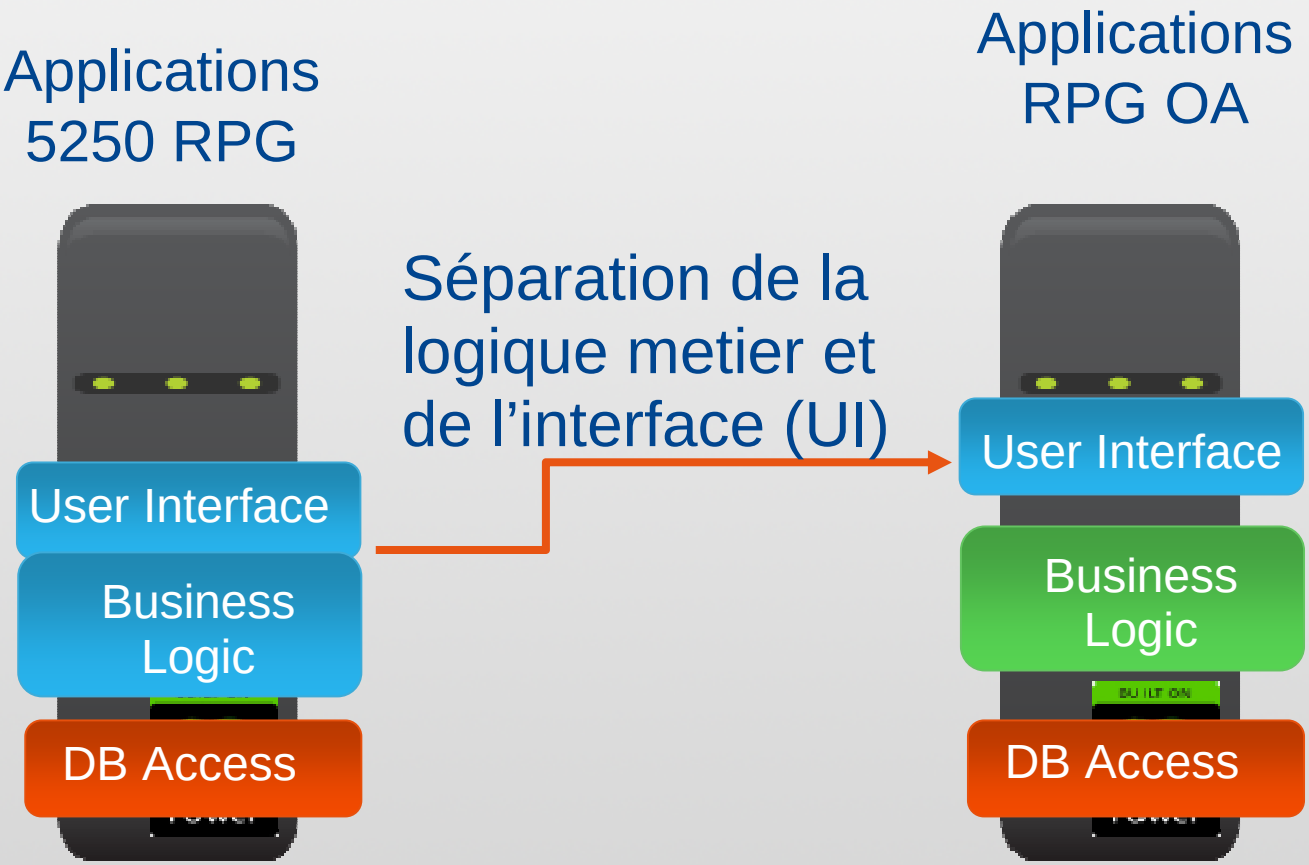
- Datastream 5250

- Données supplémentaires recueillies manuellement avec DDM / RPC ou webservice autour du 5250

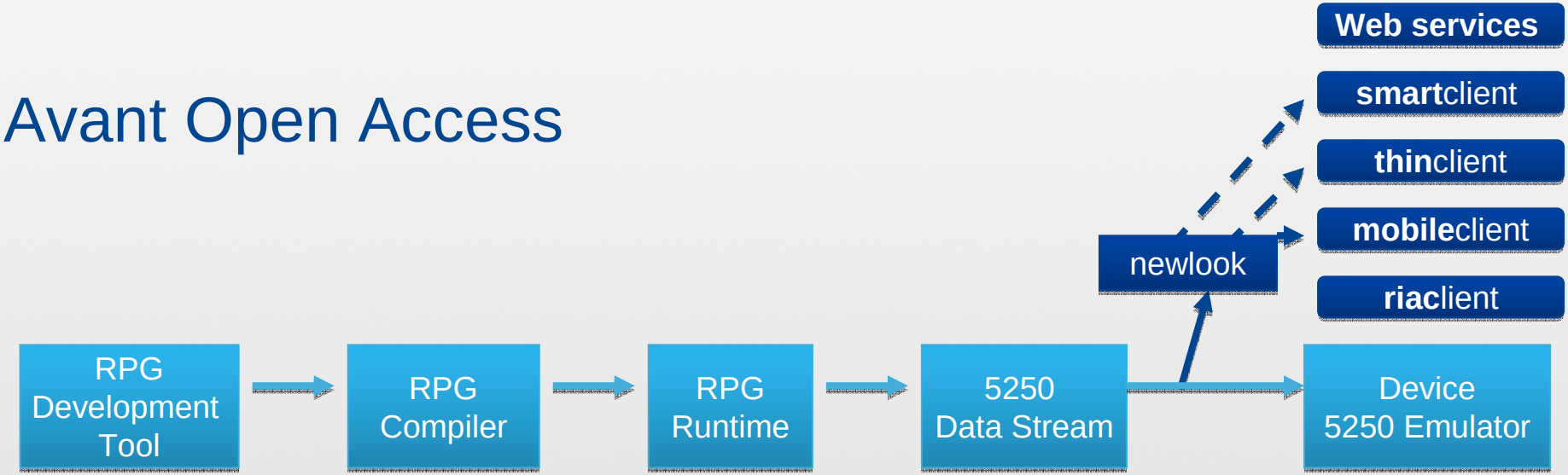
# Avec Open Access

- Display file entier
- Mémoire / buffer
- Sous-fichier entier
- Toutes les touches de fonctions
- Toutes les valeurs de liste
- DDM / RPC / WS toujours disponibles

# Véritable architecture multi-tiers

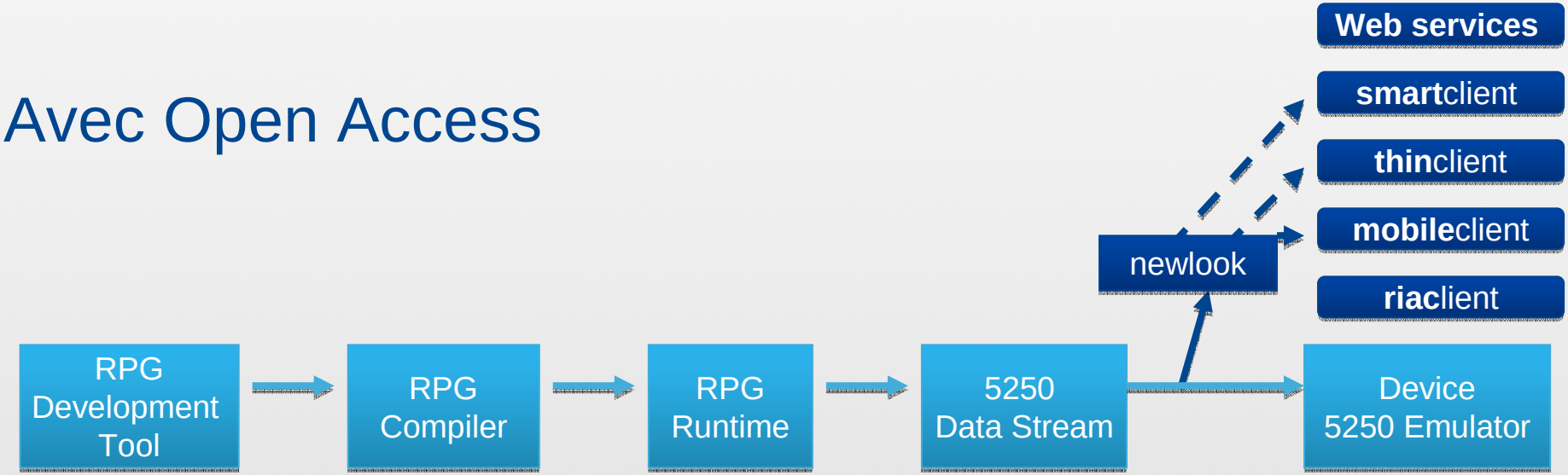


# Avant Open Access

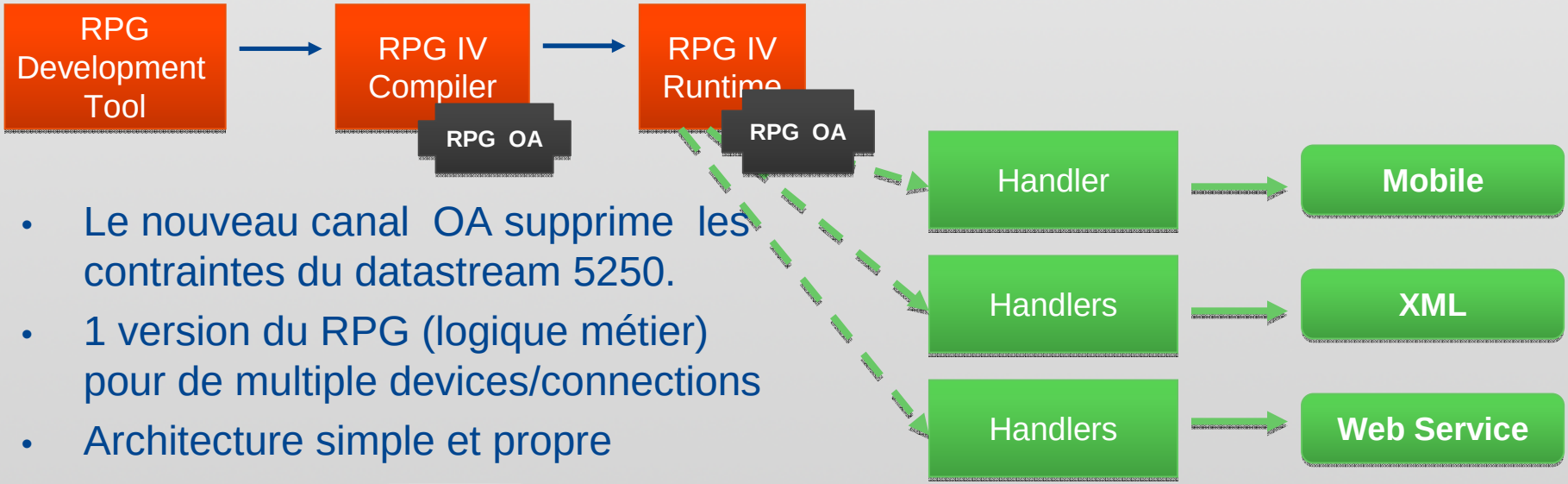



- Contraintes 5250 datastream
  - 24 lignes X 80 colonnes
  - Aucun type de donnée riche
    - Images etc..
- Nouvelles exigences
  - Ecrire nouveaux RPG et 5250 et revamper?
- Une architecture moderne est nécessaire pour soutenir les technologies / dispositifs émergents

# Avec Open Access

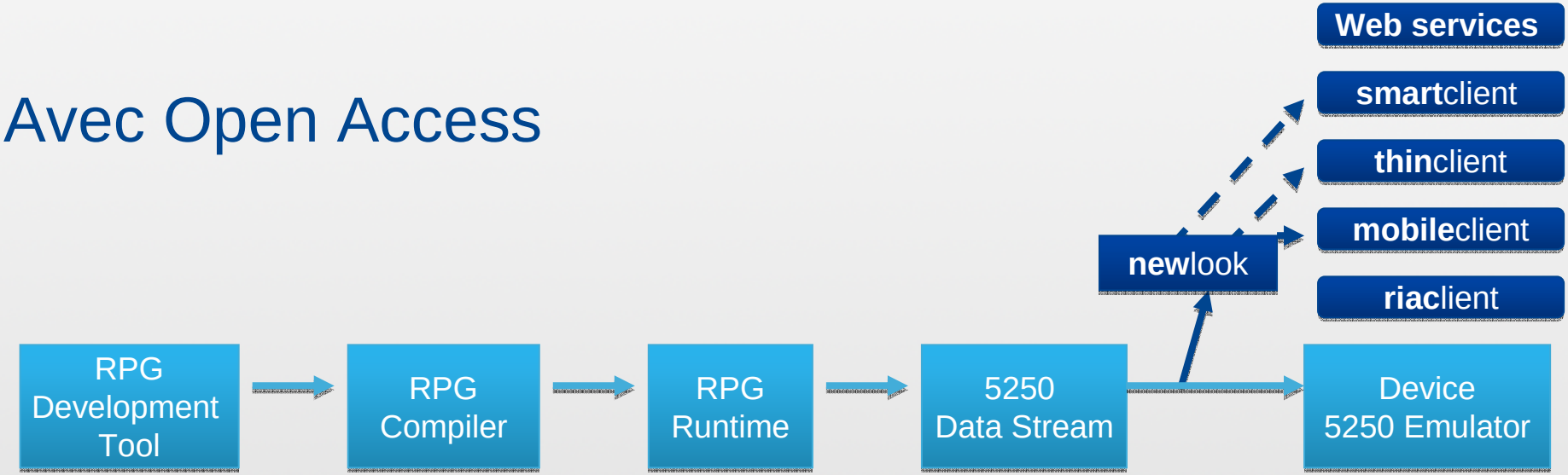



# Le canal natif Open Access

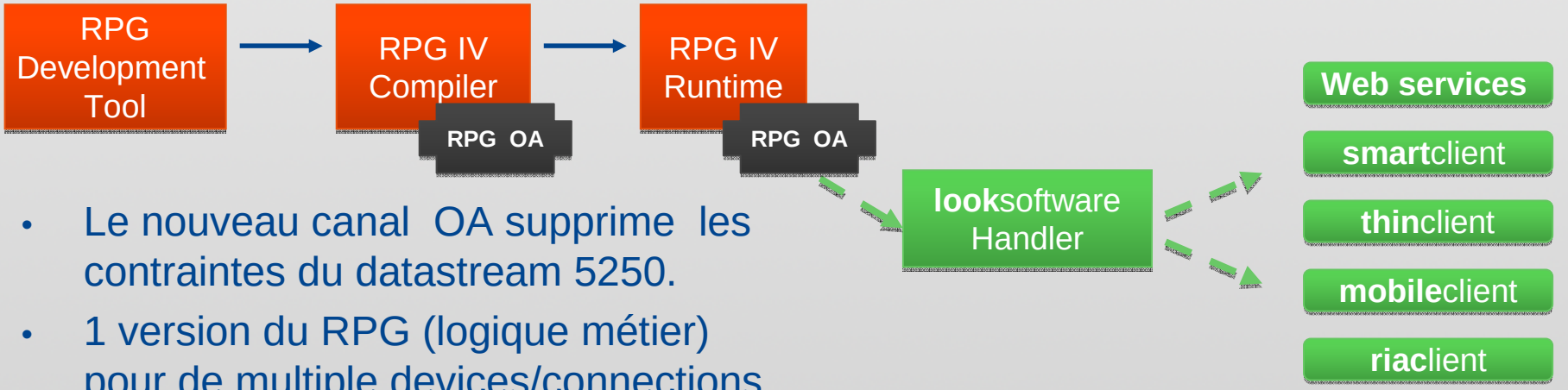



- Le nouveau canal OA supprime les contraintes du datastream 5250.
- 1 version du RPG (logique métier) pour de multiple devices/connections
- Architecture simple et propre

# Avec Open Access

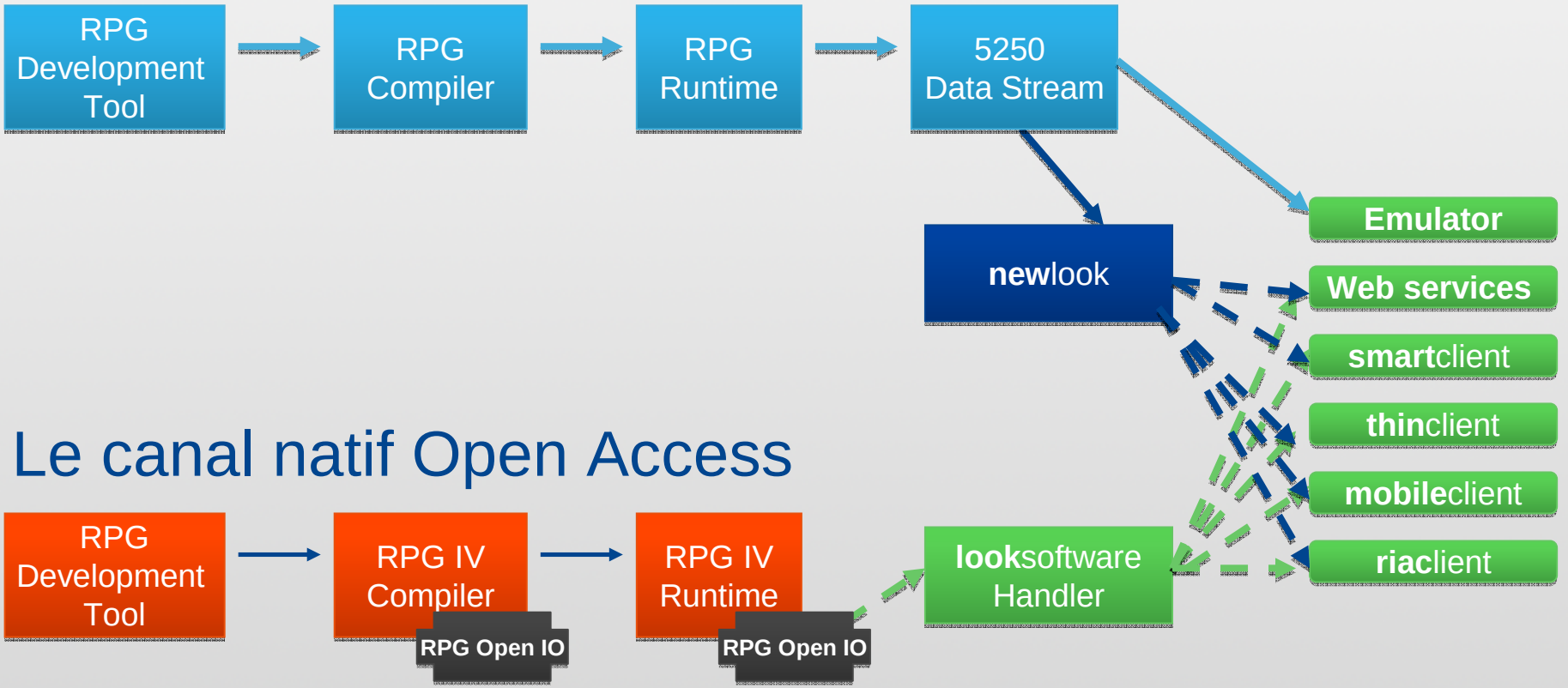



# Le canal natif Open Access




- Le nouveau canal OA supprime les contraintes du datastream 5250.
- 1 version du RPG (logique métier) pour de multiple devices/connections
- Architecture simple et propre

# Avec Open Access




# Le canal natif Open Access

# Pourquoi ROA est une avancée majeure ?

- Support RPG natif pour les nouveaux devices des technologies d'aujourdhui et de demain
- Supprime les contraintes du 5250
- RPG existants deviennent multi-tiers




# Pourquoi ROA est une avancée majeure ?

- Futur effectif pour le RPG
- L'architecture du Handler est ouverte
  - N'est pas liée à des technologies d'ISV
- Facile à implémeter
  - Logique metier en RPG
  - Le handler s'occupe de la "plomberie" (connection/device)

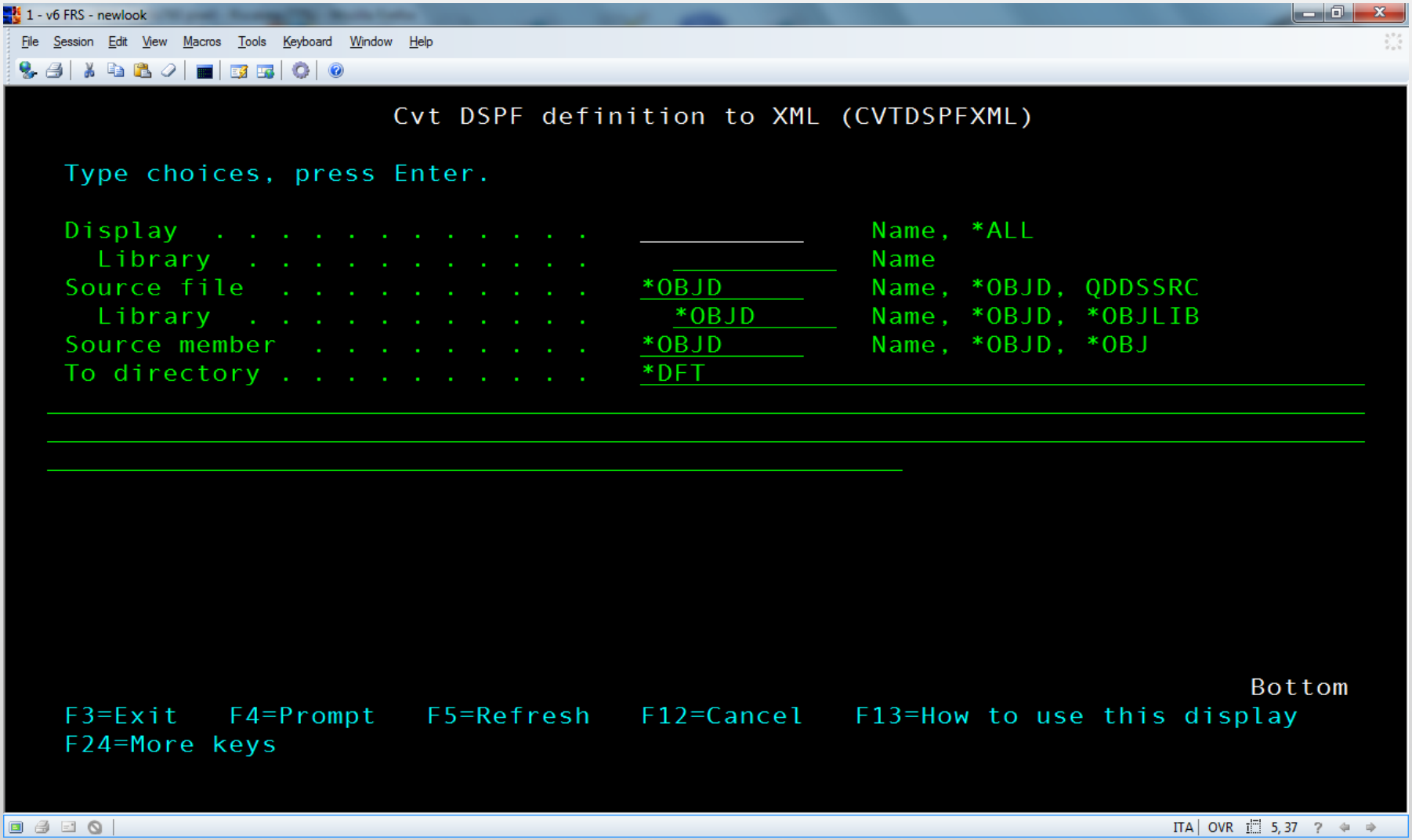



# Que doit faire le programmeur ?

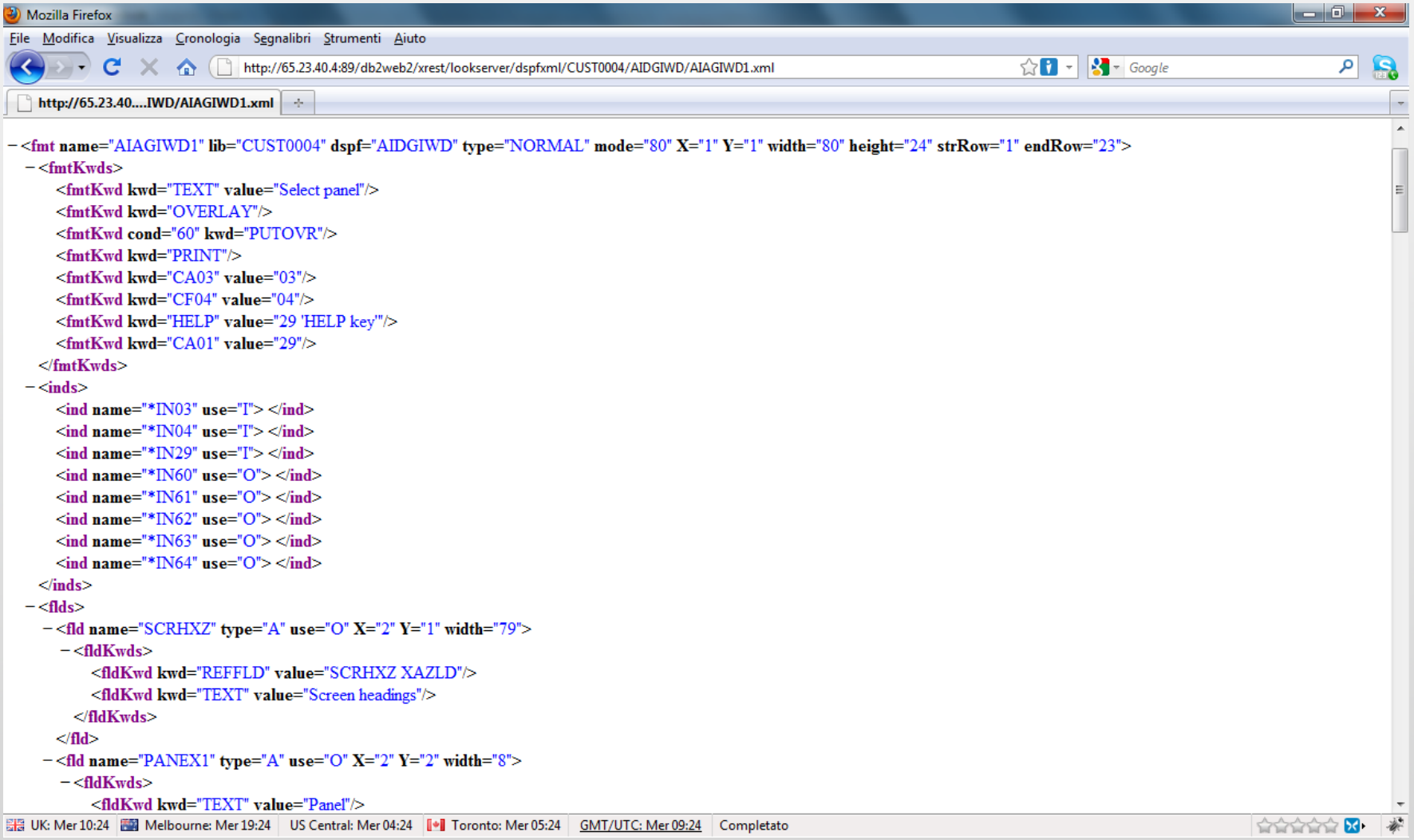
Looksoftware met a disposition 2 commandes :

- Commande CVTDSPFXML
- Commande CVTRPGROA

# Conversion des display file en XML

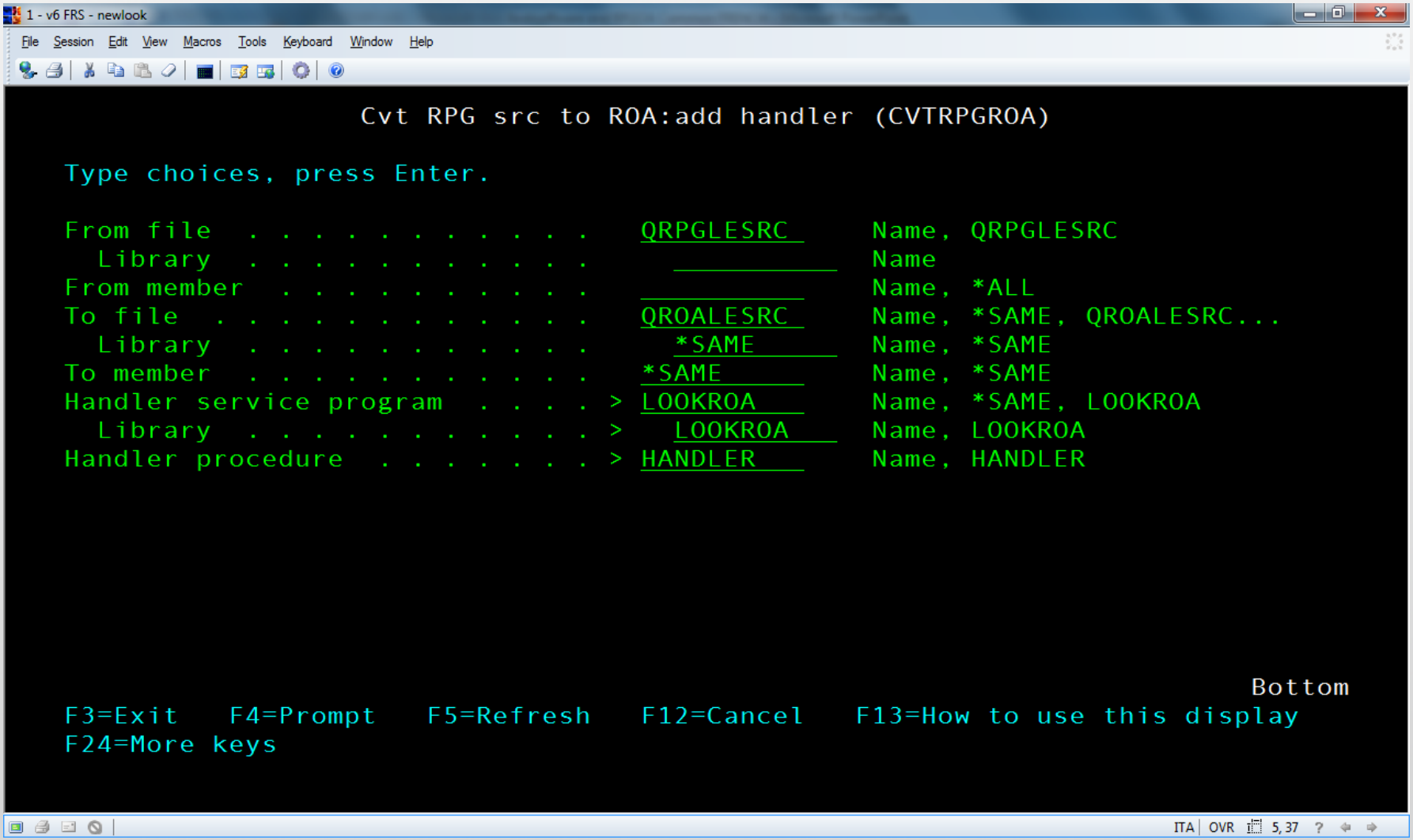


# Conversion des display file en XML

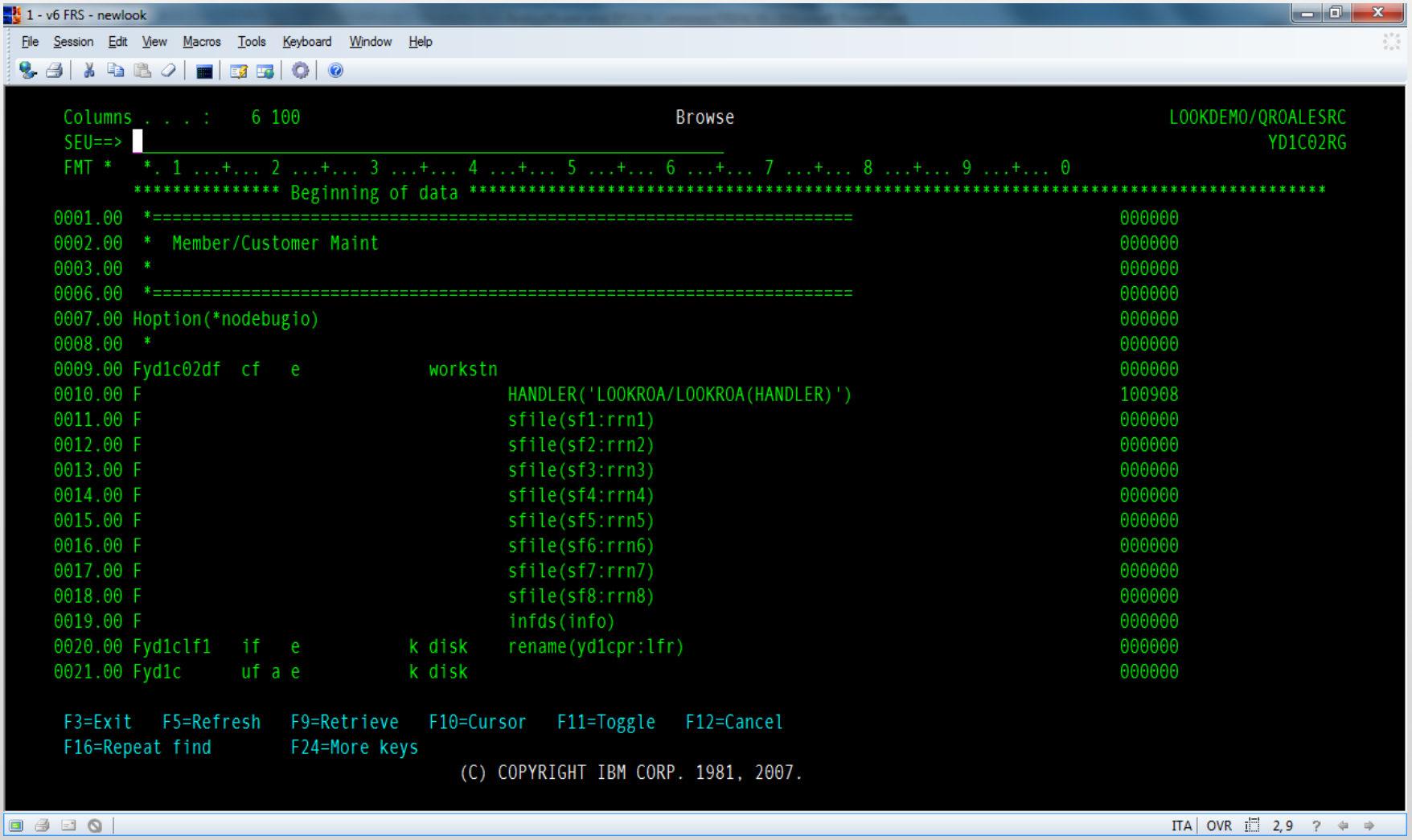



```
- <fmt name="AIAGIWD1" lib="CUST0004" dspf="AIDGIWD" type="NORMAL" mode="80" X="1" Y="1" width="80" height="24" strRow="1" endRow="23">
  - <fmtKwds>
      <fmtKwd kwd="TEXT" value="Select panel"/>
      <fmtKwd kwd="OVERLAY"/>
      <fmtKwd cond="60" kwd="PUTOVR"/>
      <fmtKwd kwd="PRINT"/>
      <fmtKwd kwd="CA03" value="03"/>
      <fmtKwd kwd="CF04" value="04"/>
      <fmtKwd kwd="HELP" value="29 'HELP key'"/>
      <fmtKwd kwd="CA01" value="29"/>
    </fmtKwds>
  - <inds>
      <ind name="*IN03" use="I"> </ind>
      <ind name="*IN04" use="I"> </ind>
      <ind name="*IN29" use="I"> </ind>
      <ind name="*IN60" use="O"> </ind>
      <ind name="*IN61" use="O"> </ind>
      <ind name="*IN62" use="O"> </ind>
      <ind name="*IN63" use="O"> </ind>
      <ind name="*IN64" use="O"> </ind>
    </inds>
  - <flds>
    - <fld name="SCRHXZ" type="A" use="O" X="2" Y="1" width="79">
      - <fldKwds>
          <fldKwd kwd="REFFLD" value="SCRHXZ XAZLD"/>
          <fldKwd kwd="TEXT" value="Screen headings"/>
        </fldKwds>
      </fld>
    - <fld name="PANEX1" type="A" use="O" X="2" Y="2" width="8">
      - <fldKwds>
          <fldKwd kwd="TEXT" value="Panel"/>
```

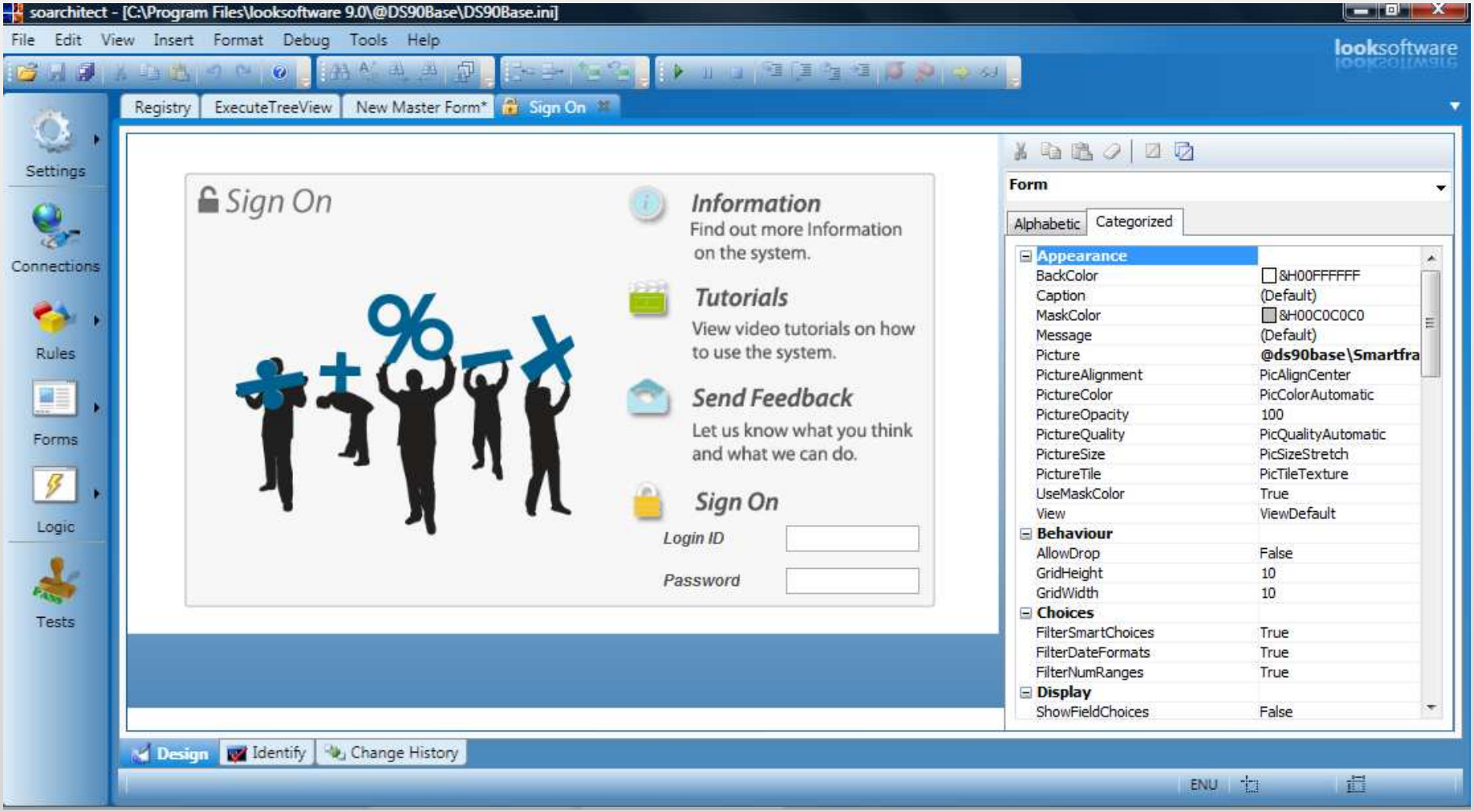
# Conversion des RPG en ROA



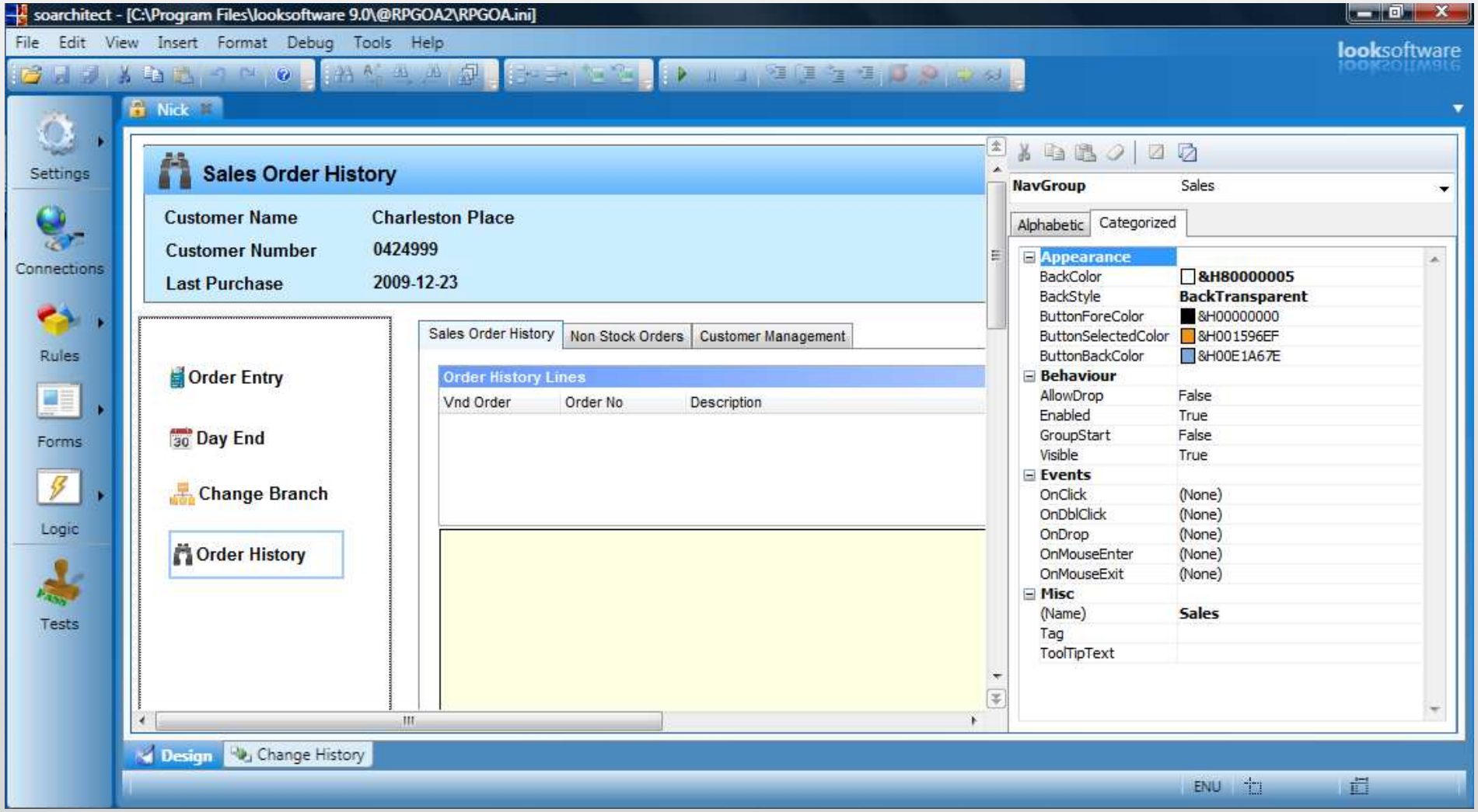
# Conversion des RPG en ROA



# Le designer WYSIWYG…



# … en remplacement de SDA



# Pourquoi utiliser le handler de **look**software?

- Solution globale et complète
  - Modernisation du non OA à OA
  - Nouveau + ancien dans la même interfac (UI) et même session
  - Ecrans système + écrans applicatifs dans la même session
- Maximise le potentiel du code natif
  - Utilise RPG natif et DDS
- Handler générique multi-canal
  - Plus qu'un simple browser
    - Rich Windows client, RIA, thin, mobile,
    - Et Service Web
- Integration solide
  - Front-end, back-end, + nouveau

looksoftware et

# IBM Open Access for RPG

Contact :


**itheis**

**Pascal BLANDIN**

pascal.blandin@itheis.com

**06 15 16 39 95**