Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2012-2013

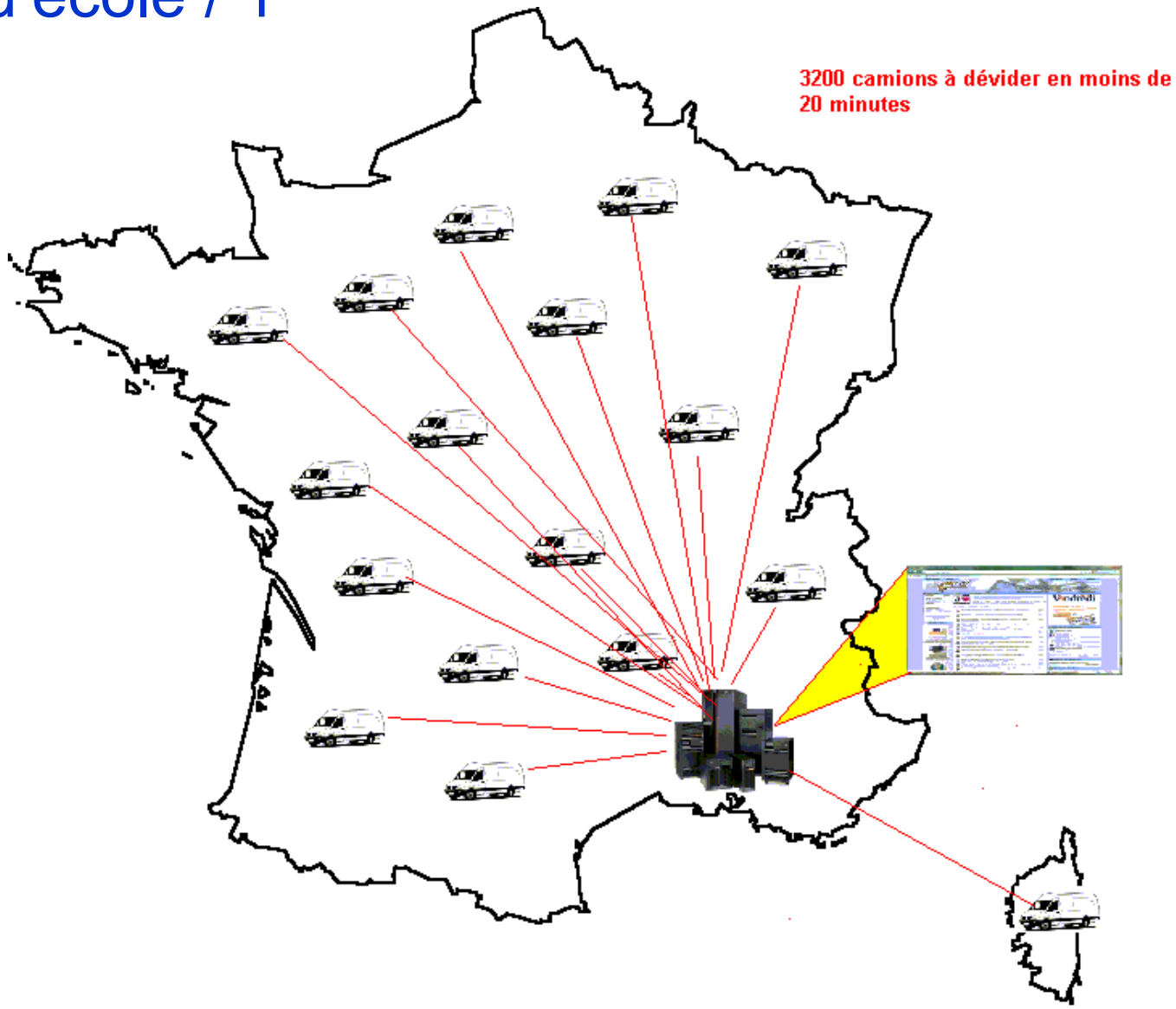
8 et 9 avril 2013 – IBM Client Center Paris, Bois-Colombes

S27 – Optimiser les ressources grâce aux travaux asynchrones : Mise en œuvre des DATAQ

Mardi 9 avril – 15h15-16h45

Pierre-Louis BERTHOIN / Nathanaël BONNET – GAIA Mini Systèmes

Cas d'école / 1



Cas d'école / 2

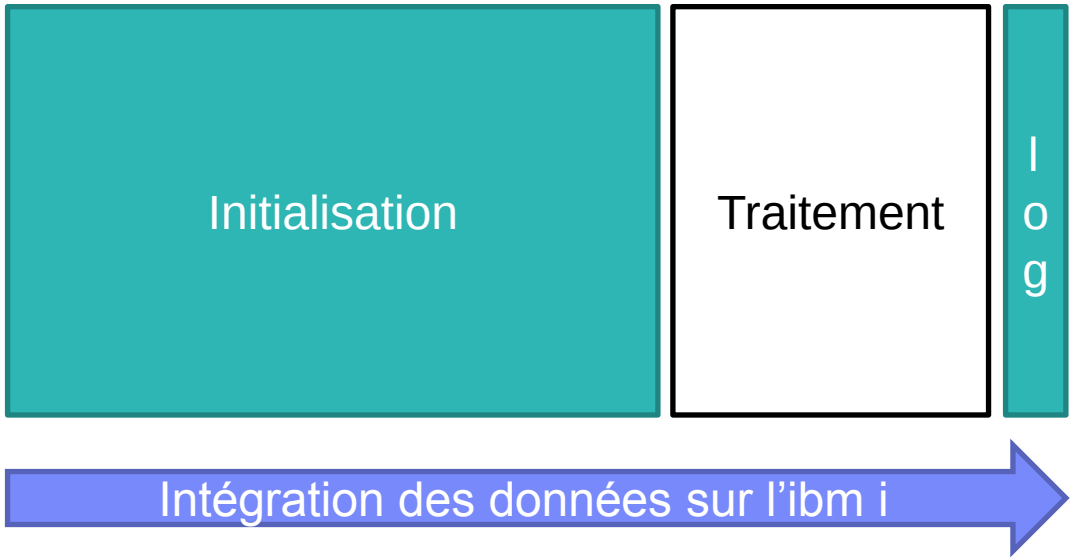
- Cahier des charges
  - 3200 véhicules à dévider avant 8 heures et avant 9 heures
  - sur une plage de 20 minutes
  - pour que le positionnement des colis apparaisse sur le site web.
- Chaque colis traité génère et envoie
  - 1 petit fichier 10 ko environ

Cas d'école / 3

- Après un mois de mise en production
  - on n'arrive pas à tout intégrer dans le lapse de temps prévu
- Après une analyse du process de rapatriement de données
  - (Terminal Symbol) ➔concentrateur (GSM)
  - Concentrateur ➔ IBM i (SDSL)
  - Intégration BD sur IBM i
- On s'aperçoit que le point d'engorgement est le dernier
  - Intégration BD sur IBM i

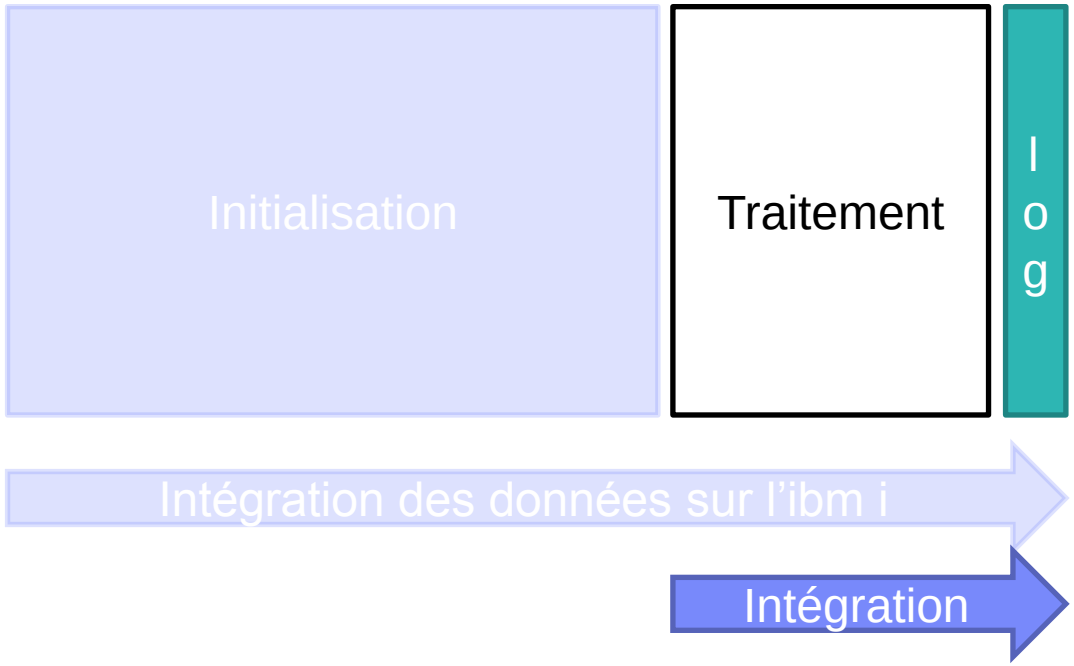
Cas d'école / 4

- Analyse du travail d'intégration
  - Représentation pour traiter 1 fichier



Cas d'école / 5

- Le traitement représentait moins 37 % de la CPU
  - Et du temps de traitement
- Piste de travail
  - Diminuer (voir supprimer) le temps d'initialisation du travail



Problématique

- Savoir lancer des traitements
  - des tâches à temps de CPU assez faibles de manière asynchrone
    - sans retour d'information immédiate
- Faciliter l'exploitation de la machine

Rappel

- Sur l'IBM i : 3 typologies de travaux
  - Les interactifs
    - Dialogue Homme machine qui va du signon au signoff
  - Les batchs
    - Des travaux consommateurs en ressources et qui ne nécessitent pas d'interaction
  - Les Deamons
    - Des travaux serveurs venant du monde unix qui attendent des connexions clientes (Web, etc…)

Les travaux asynchrones / 1

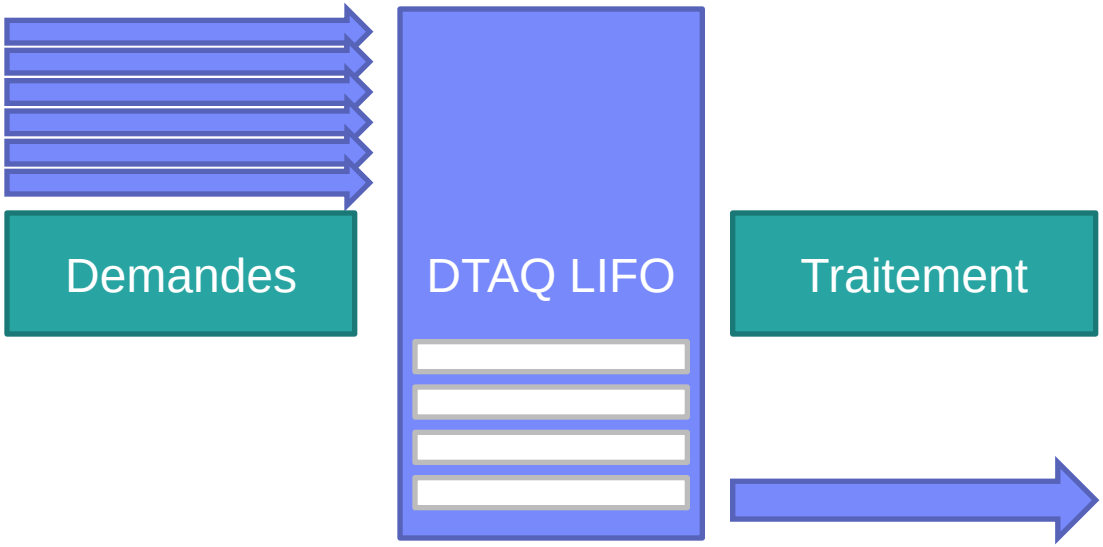
- C’est le meilleur des 3 mondes
  - un travail batch qui va
    - recevoir des demandes
    - les traiter de manière séquentielle
    - proche d’un démon Unix
- Plusieurs solutions sur l’IBM i, dont
  - Les fichiers à fin de fichier retardée
  - Les files de messages (MSGQ) avec un programme de veille
  - Les files de données (DATAQ)
    - la seule méthode créée à cet effet

Les travaux asynchrones / 2

- C'est également la philosophie d'un logiciel qui s'appelle MQSERIES
  - permet d'envoyer et de traiter des messages entre plusieurs systèmes
- C'est une solution complètement Packagée
  - Gestion des queues
  - Acheminement des messages
  - Administration de la solution
- Des solutions de « Queuing » existent
  - sur chaque plateforme
    - MSMQ pour Microsoft par exemple
  - dans de nombreux logiciels serveurs
    - Apache …

Les travaux asynchrones / 3

- Donc c'est un process dit asynchrone qui attend et traite des demandes envoyées
- La solution la plus adaptée est la DTAQ



Remarques / 1

- Il existe peu de commandes systèmes pour les gérer
  - CRTDTAQ
  - DLTDTAQ
  - WRKDTAQ
- L'utilisation de DTAQ se fait donc par APIs
  - pour écrire et lire à l'intérieur
  - des APIs natives de l'os
    - mais également sur Windows et sur Unix ou JAVA
- Nous allons voir les APIs de base de l'OS

Remarques / 2

- Vous allez donc créer une ou plusieurs DTAQ
  - Le plus souvent le nom de la DTAQ est en paramètre
- Créer ou modifier un programme qui va venir empiler dans la DTAQ
- Créer un programme d'écoute
  - En CLLE ou en RPGLE (CBLLE, CLE, CPPLE …)
  - C'est lui le nerf de la guerre

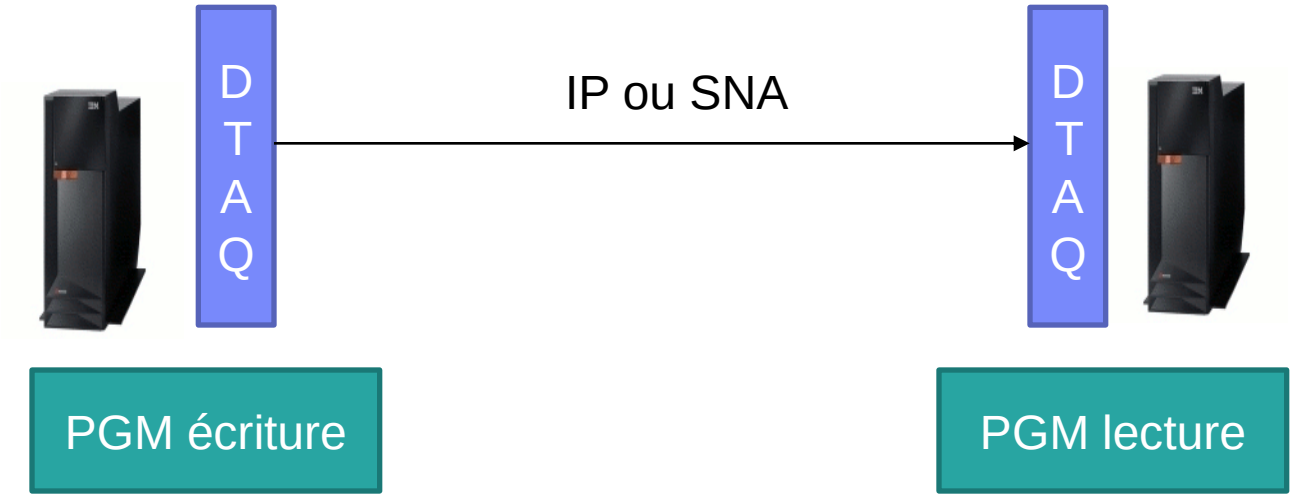
Création d'une DTAQ – CRTDTAQ

```
File d'attente de données  . . .  DTAQ          > MADTAQ
  Biblio . . . . . .                            >   MALIB
Type . . . . . . . . . . . . . .  TYPE            *STD
Longueur maximale de poste . . .  MAXLEN        > 156
Forcer sur mémoire secondaire  .  FORCE           *NO
Séquence . . . . . . . . . . . .  SEQ             *FIFO
Longueur de clé  . . . . . . . .  KEYLEN
Inclure ID émetteur  . . . . . .  SENDERID        *NO
Taille de la file d'attente:      SIZE
  Nombre maximal de postes . . .                > *MAX2GB
  Nombre initial de postes . . .                  16
Récupération automatique . . . .  AUTORCL        > *YES
```

- MAXLEN Longueur maxi d'un poste 64512
- SIZE Taille maximum en nombre de poste
- SENDERID Inclure ou pas le profil émetteur
- AUTORCL *YES = compression automatique

Création d'une DTAQ – CRTDTAQ

- On ne parlera pas des DTAQ REMOTE
  - le principe est le même



Lecture / Ecriture

- 2 jeux d’ APIs permettent de lire et d'écrire
- Les APIs historiques
  - QSNDDTAQ
    - Ecriture dans la DTAQ
  - QRCVDTAQ
    - Lecture dans la DTAQ
      - supprime le poste dans la DTAQ
  - QCLRDTAQ
    - Mise à blanc de la DTAQ
- Les APIs ajoutées
  - QMHQRDQD
    - Extraction de la description
  - QMHRDQM
    - Lecture dans la DTAQ sans suppression

Programme d'écriture

- C'est un simple stockage dans une file sans aucune malice
- Il faut simplement formater le message
  - terme employé dans les process asynchrones pour indiquer la structuration des données

Ecriture – QSNDDTAQ

```
 // Prototypage des APIs en RPGLE

D SndDtaQ         PR                  EXTPGM('QSNDDTAQ')
D DataQueueNam                  10A   Const
D DataQueueLib                  10A   Const
D DataLength                     5P 0 Const
D DataBuffer                 32767A   Const Options(*varsize)
 * Optional parameter group (Keyed DTAQ)
D KeyLength                      3P 0 Const Options(*nopass)
D KeyBuffer                    256A   Const Options(*nopass : *varsize)
 * Optional parameter group (Asynchronous) / DTAQ DDM
D AsyncReq                      10A   Const Options(*nopass )
 * Optional parameter group (Journal Entry)
D DataFrmJrnE                   10A   Const Options(*nopass )

 * ...


 // Le 1er groupe de paramètres ne s'utilise qu'avec une DTAQ à clé
/free
  SnDtaq( 'MADTAQ' : 'MALIB' : 11 : 'Mon message' ) ;
/end-free
```

Le programme de lecture

- C’est un programme qui devra boucler en attendant une information dans la file
- Pour ce faire il faudra indiquer une valeur d’attente négative dans l’API QRCVTAQ

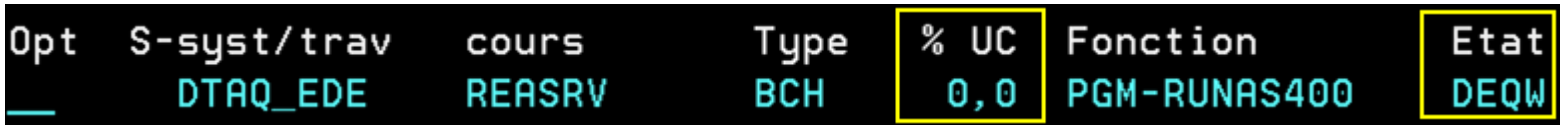
- Attention
  - par défaut l’information dépilée est effacée de la file
  - Si cette perte est préjudiciable il faudra
    - Sauvegarder l’info dans une DTAARA
    - Ou utiliser les APIs
      - QMHRDQM (lecture sans suppression)
      - QRCVDTAQ (lecture avec suppression)

Le programme de lecture

- Il ne faut pas oublier de prévoir la fin du traitement
  - Par arrêt brutal du traitement
    - à proscrire
  - Par un poste d'arrêt placé dans la file
    - '*FIN' par exemple
  - Sur une DTAQ avec clé
    - On peut se servir de la clé comme d'une priorité dans l'ordre de traitement des messages
      - un poste d'arrêt qui n'attendrai pas la fin de l'exécution des postes qui précèdent

Le programme de lecture

- Certains process suppriment et recréent la DTAQ
  - c'est historique
  - aujourd'hui on peut indiquer à la création de la compresser paramètre AUTORCL à *YES
- Job en attente de lecture de DTAQ



Opt	S-syst/trav	cours	Type	% UC	Fonction	Etat
__	DTAQ_EDE	REASRV	BCH	0,0	PGM-RUNAS400	DEQW

  - Attente passive
    - Pas de consommation de CPU !
  - DEQW
    - L'unité d'exécution initiale du travail attend la fin de l'opération de retrait d'une file d'attente.

QRCVDTAQ

- Prototypage des APIs en RPGLE

```
D RcvDtaQ          PR                  EXTPGM('QRCVDTAQ')
D DataQueueNam                   10A   Const
D DataQueueLib                   10A   Const
D DataLength                      5P 0
D DataBuffer                  32767A   Options(*Varsize)
D WaitTime                        5P 0 Const
 * Optional parameter group 1 (Keyed DTAQ)
D KeyOrder                        2A   Const Options(*Nopass)
D KeyLength                       3P 0 Const Options(*Nopass)
D KeyBuffer                     256A   Options(*Nopass : *Varsize)
D SndLength                       3P 0 Const Options(*Nopass)
D SndBuffer                      44A   Options(*Nopass)
 * Optional parameter group 2
D RemoveMsg                      10A   Const Options(*Nopass : *Omit)
D RcvSize                         5P 0 Const Options(*Nopass : *Omit)
D Error                       32767A   Options(*Nopass : *Varsize)
```

Exemple CLP

- Lecture dans une DTAQ

```
/* Variables de travail */
            DCL         VAR(&util) TYPE(*CHAR) LEN(10) VALUE(' ')
            DCL         VAR(&msg) TYPE(*CHAR) LEN(100) VALUE(' ')
/* -- paramètres pour QNSDDTAQ */
            /* Requis */
            DCL         VAR(&DTAQNAM) TYPE(*CHAR) LEN(10) VALUE('MADATQ')
            DCL         VAR(&DTAQLIB) TYPE(*CHAR) LEN(10) VALUE('MABIB')
            DCL         VAR(&DATALEN) TYPE(*DEC) LEN(5 0) VALUE(0)
            DCL         VAR(&DATA) TYPE(*CHAR) LEN(256) VALUE(' ')
            DCL         VAR(&WAIT) TYPE(*DEC) LEN(5 0) VALUE(-1) /* attente */
            /* Groupe optionnel 1 */
            DCL         VAR(&KEYORDR) TYPE(*CHAR) LEN(2) VALUE(' ')
               /* Pas de clé */
            DCL         VAR(&KEYDTALEN) TYPE(*DEC) LEN(3 0) VALUE(0)
            DCL         VAR(&KEYDTA) TYPE(*CHAR) LEN(1) VALUE(' ')
            DCL         VAR(&SNDRIDLEN) TYPE(*DEC) LEN(3 0) VALUE(44)
            DCL         VAR(&SNDRID) TYPE(*CHAR) LEN(44) VALUE(' ')
            /* Groupe optionnel 2 */
            DCL         VAR(&RMVMSG) TYPE(*CHAR) LEN(10) VALUE('*YES      ')
            DCL         VAR(&DTARCVLEN) TYPE(*DEC) LEN(5 0) VALUE(256)
            DCL         VAR(&ERRORCODE) TYPE(*CHAR) LEN(256) VALUE(' ')
/* -- découpage ERRORCODE */
            DCL         VAR(&MSGID) TYPE(*CHAR) LEN(7)
```

Les DTAQ (5/7)

Lecture d'une DTAQ

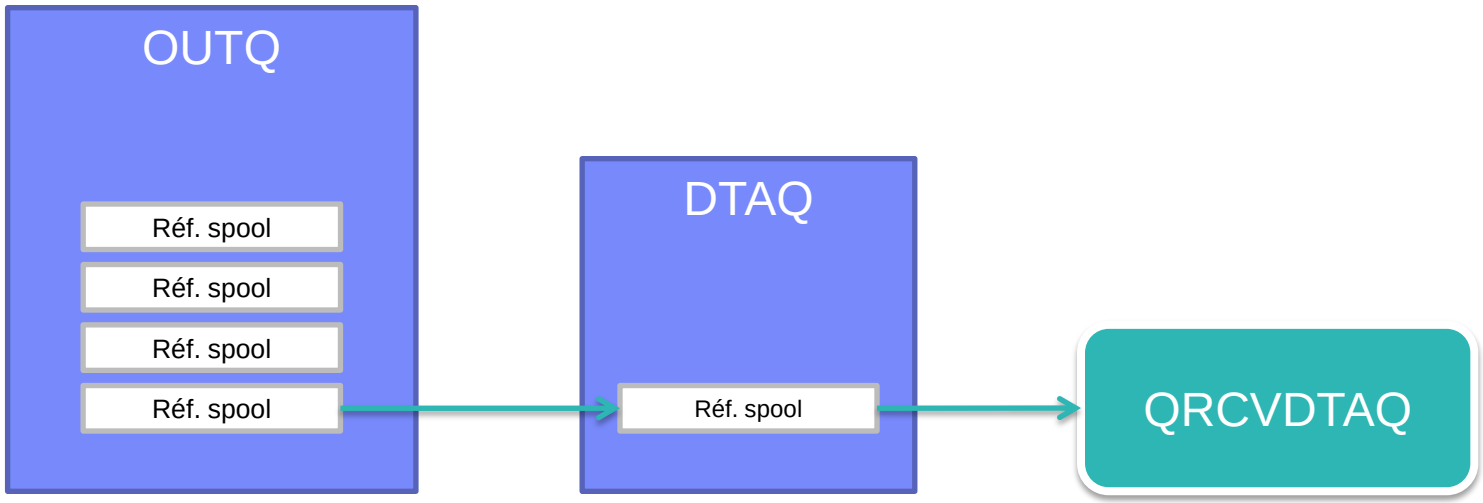
```
 /* Traitement */
BOUCLE :
CHGVAR  %bin(&ERRORCODE 1 4) 256
CALL        PGM(QRCVDTAQ) PARM(&DTAQNAM &DTAQLIB &DATALEN &DATA &WAIT &KEYORDR +
            &KEYDTALEN &KEYDTA &SNDRIDLEN &SNDRID &RMVMSG &DTARCVLEN &ERRORCODE )
/* Test anomalie */
IF          COND(%BIN(&ERRORCODE 5 4) *GT 0) THEN(DO)
    CHGVAR      &msgid %sst(&errorcode 9 7)
    CHGVAR      VAR(&MSG) VALUE('Anomalie à la lecture : ' !!  &MSGID)
ENDDO
ELSE        CMD(DO)
/* extraction message */
    CHGVAR      &msg VALUE( 'Message : ' !! %sst( &data 1  &datalen))
/* extraction senderid */
    CHGVAR      VAR(&MSG) VALUE(&MSG *tcat '. SenderID : ' !! %SST(&SNDRID 9 10) +
                !! '/' !! %SST(&SNDRID 19 10) !! '/' !! %SST(&SNDRID 29 6) !! +
                ' Profil en cours : ' !! %SST(&SNDRID 35 10))
    SNDMSG MSG(&msg) TOUSR(berthoin)
ENDDO

IF    COND(&data 1 4 ) *NE '*FIN')  THEN(DO)
       /* Votre traitement ici  *
      GOTO BOUCLE
ENDDO

ENDPGM
```

Les OUTQs

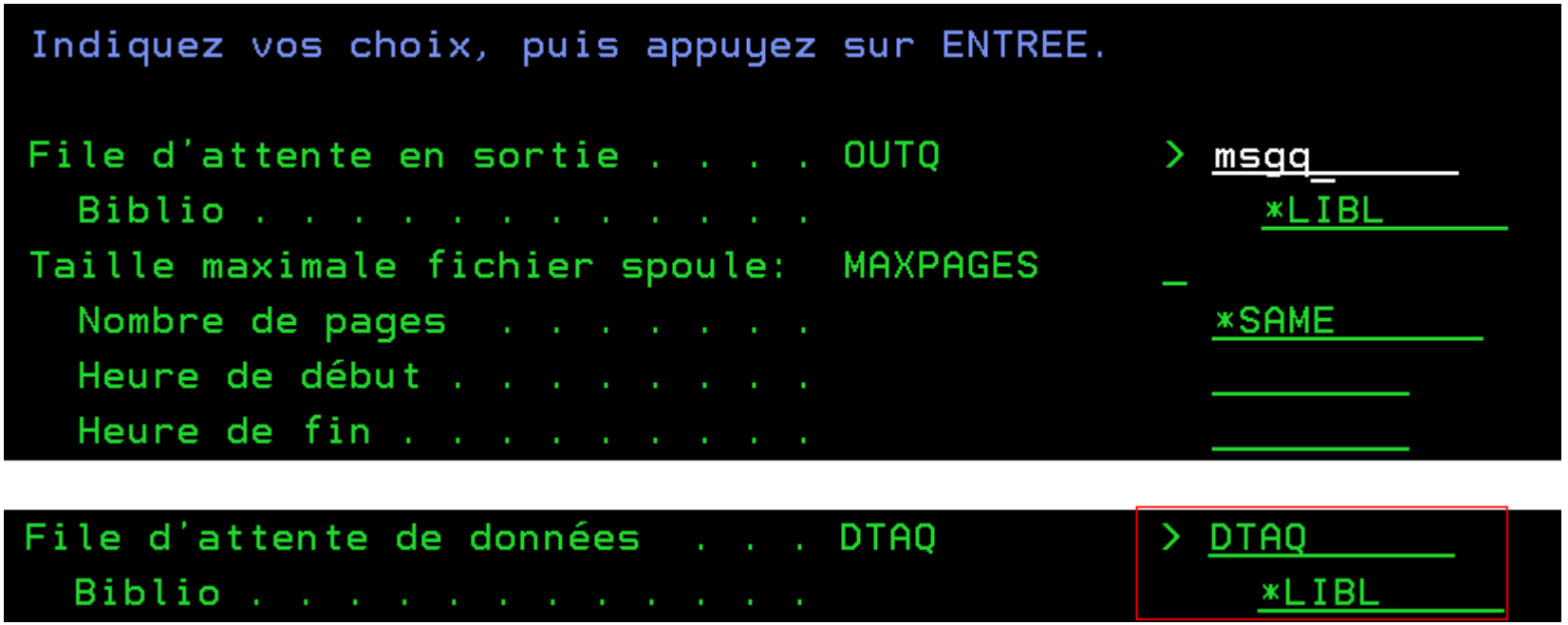
- Une autre utilisation spécifique à l'IBM i est très fréquente pour automatiser le traitement de fichiers spools



Les OUTQs

- On peut associer une DTAQ à une OUTQ
  - Paramètre DTAQ sur CRTOUTQ ou CHGOUTQ
- Chaque fois qu'un spoule passe à l'état RDY un poste est créé dans la DTAQ
  - si vous avez un job en attente sur celle-ci vous pourrez intervenir sur ce spoule
    - envoyer, archiver, supprimer, etc…
  - Les outils de rewamping sont souvent basés la dessus
- Exemple
  - Un programme de suppression

CHGOUTQ



Exemple / 1

```
PGM parm(&dtaq &lib)

/* Paramètres */
DCL         VAR(&dtaq) TYPE(*char) LEN(10)
DCL         VAR(&lib) TYPE(*char) LEN(10)

/* Variables de travail */
DCL         VAR(&FLDLEN) TYPE(*DEC) LEN(5 0)
DCL         VAR(&FIELD) TYPE(*CHAR) LEN(128)
DCL         VAR(&WAIT) TYPE(*DEC) LEN(5 0) VALUE(-1)
DCL         VAR(&ERR) TYPE(*CHAR) LEN(50)
DCL         VAR(&SPLNBR) TYPE(*DEC) LEN(9 0)
DCL         VAR(&USER) TYPE(*CHAR) LEN(10)
DCL         VAR(&SPLNM) TYPE(*CHAR) LEN(10)
DCL         VAR(&JOBNM) TYPE(*CHAR) LEN(10)
DCL         VAR(&JOBNBR) TYPE(*CHAR) LEN(6)
```

Exemple / 2

```
/* Boucle d'écoute de la DTAQ */
LOOP:
CALL        PGM(QRCVDTAQ) PARM(&dtaq &lib &FLDLEN &FIELD &WAIT)

/* Extraction des infos */
CHGVAR      VAR(&SPLNBR) VALUE(%BIN(&FIELD 49 4))
CHGVAR      VAR(&JOBNM) VALUE(%SST(&FIELD 13 10))
CHGVAR      VAR(&USER) VALUE(%SST(&FIELD 23 10))
CHGVAR      VAR(&JOBNBR) VALUE(%SST(&FIELD 33 6))
CHGVAR      VAR(&SPLNM) VALUE(%SST(&FIELD 39 10))

/* Suppression du spoule */
DLTSPLF FILE(&SPLNM) JOB(&JOBNBR/&USER/&JOBNM) SPLNBR(&SPLNBR)

/* Spoule suivant */
GOTO        CMDLBL(LOOP)

ENDPGM
```

Remarque

- Cet exemple
  - Doit être complété
    - Par un contrôle de l'existence de la DTAQ
    - Par un arrêt du programme d'attente
    - Par les contrôles d'usages
    - Par un verrouillage de la DTAQ qui évite un double lancement du programme d'écoute
    - Etc …

Autres cas

- Contrôler la consommation des ressources
  - JAVA
    - Chaque job utilisant Java se voit associé une JVM
      - Importante consommation de mémoire, temps d'instanciation élevé
    - Un job unique traite les demandes d'appel JAVA qui sont empilées dans la DTAQ
- Contrôle de l'exploitation
  - Démons spécialisés par tâches
  - Et/ou mise en attente de certaines tâches

Le suivi

- Il n'existe aucune solution simple de visualiser le contenu d'une DTAQ
  - 1ère solution : la solution du pauvre
    - DMPOBJ de votre objet
  - 2ème solution : faire un programme qui lit la pile sans supprimer les postes
    - on peut en trouver en « googlant » un peu
- Exploitation
  - Le contenu des DTAQ est sauvegardé
    - SAVOBJ … QDTA(*DTAQ)
    - Depuis la 6.1

Informations utiles

- Ressources
  - http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0
- Forums
  - http://forum.xdocs400.com
  - http://www.developpez.net
  - http://forum.commonfr.org
  - http://www.volubis.fr
  - http://www.iprodeveloper.com

Nous contacter

- Par mail
  - plberthoin@gaia.fr
  - contact@gaia.fr
- Nos sites
  - www.gaia.fr
  - www.know400.fr
  - www.as400.fr