



# CICS/TS

## Programmation des Applications

(Avril 2017)

**(Avril 2017)**

# SOMMAIRE

<b>SOMMAIRE</b>	<b>1</b>
<b>INTRODUCTION</b>	<b>7</b>
<b>OBJECTIFS DU CHAPITRE</b>	<b>8</b>
<b>HISTORIQUE DE CICS</b>	<b>9</b>
<b>PRÉSENTATION DE CICS</b>	<b>10</b>
<b>LE MONITEUR TP CICS</b>	<b>11</b>
<b>UN PROGRAMME CICS</b>	<b>15</b>
<b>ORGANISATION DE CICS</b>	<b>16</b>
<b>OBJECTIFS DU CHAPITRE</b>	<b>17</b>
<b>CICS ET SYSTEME D'EXPLOITATION</b>	<b>18</b>
<b>LES FICHIERS de CICS/TS</b>	<b>19</b>
<b>JOB D'INITIALISATION</b>	<b>20</b>
<b>.COMPOSANTS D'UNE REGION CICS</b>	<b>20</b>
<b>COMPOSANTS D'UNE REGION CICS</b>	<b>21</b>
<b>LA RÉGION CICS/TS</b>	<b>24</b>
<b>LES DOMAINES DE CICS/TS V.3.3.0</b>	<b>25</b>
<b>LES NOUVEAUX DOMAINES EN CICS/TS 1.3</b>	<b>26</b>
<b>LES NOUVEAUX DOMAINES EN CICS/TS 3.1</b>	<b>27</b>
<b>LA REGION CICS</b>	<b>28</b>
<b>OBJECTIFS DU CHAPITRE</b>	<b>29</b>
<b>MODULARITÉ DE CICS</b>	<b>30</b>
<b>DÉROULEMENT D'UNE TRANSACTION CICS</b>	<b>32</b>
<b>LA TRANSACTION CICS</b>	<b>33</b>
<b>Le KCP ou DS DOMAINE</b>	<b>34</b>
<b>LA PCT (Program Control Table)</b>	<b>35</b>
<b>LE PROGRAMME CICS</b>	<b>36</b>
<b>RÉGION CICS : tâches simultanées</b>	<b>37</b>
<b>LA RÉGION CICS</b>	<b>39</b>
<b>TCP (Terminal Control Program)</b>	<b>40</b>
<b>LA GESTION DE LA MÉMOIRE</b>	<b>41</b>
<b>LE PCP (Application Domaine)</b>	<b>42</b>
<b>PPT (Processing Program Table)</b>	<b>43</b>
<b>TACHE CICS : LE KCP</b>	<b>44</b>
<b>TRANSACTION / TACHE</b>	<b>45</b>
<b>RÉENTRANCE (MULTI-THREADING)</b>	<b>46</b>

FCP (File Control Program) _____	47
FIN D'UNE TACHE _____	48
Schéma récapitulatif du démarrage d'une tâche _____	49
OBJECTIFS DU CHAPITRE _____	51
LE TERMINAL 3270 _____	52
LE TCP (Terminal Control Program) _____	55
LE TERMINAL 3270 et BMS _____	57
Basic Mapping Support (BMS) _____	58
LA MAP PHYSIQUE _____	59
LA MAP SYMBOLIQUE _____	60
DÉFINITION D'UN ÉCRAN _____	61
DÉFINITION D'UN MAPSET DFHMSD _____	62
FIN D'UN MAPSET DFHMSD _____	70
EXEMPLE DE MAP _____	71
CRÉATION DE MAPS _____	73
MAP SYMBOLIQUE Génération _____	74
MAP SYMBOLIQUE Exemple _____	77
MODIFICATION DE L'ATTRIBUT _____	78
<b>LA PROGRAMMATION _____</b>	<b>79</b>
OBJECTIFS DU CHAPITRE _____	80
H.L.P.I _____	81
LE PRÉCOMPILATEUR DFHECP1\$ _____	82
ÉTAPES POUR LA CRÉATION D'UN LOAD _____	83
LE BLOC EIB _____	85
CONTENU DU BLOC EIB _____	86
LE BLOC EIB _____	87
Caractéristiques d'un programme CICS _____	88
SYNTAXE DES COMMANDES CICS _____	89
EXEMPLE _____	90
LES COMMANDES HANDLE _____	91
LES CONDITIONS EXCEPTIONNELLES _____	92
L'OPTION RESP _____	95
RESTRICTIONS COBOL _____	97
EMISSION DE DONNÉES _____	100
RECEPTION DE DONNÉES _____	101
EXEMPLE _____	102
SEND MAP _____	103
RECEIVE MAP _____	106

COMMANDE HANDLE AID	108
<b>LA GESTION DES PROGRAMMES</b>	<b>110</b>
OBJECTIFS DU CHAPITRE	111
LES FONCTIONS DU KCP et PCP	112
LES NIVEAUX LOGIQUES	113
COMMANDE LINK	114
COMMANDE XCTL	115
COMMANDE RETURN	116
EXEMPLE	117
TRANSFERT DE DONNEES	118
COMMANDE ABEND	124
LE MODE CONVERSATIONNEL	125
LE MODE PSEUDO-CONVERSATIONNEL	126
VUE GLOBALE DU PROGRAMME	130
Vue détaillée de la partie 22000-bbbbb	131
RÈGLES	132
NOMS DE PARAGRAPHES	133
OPTIONS DIVERSES	134
STRUCTURE DE LA COMMAREA	136
exemple de squelette	137
Gestion de la COMMAREA	141
Changement de taille de la COMMAREA	143
Utilisation de l'option RESP	145
Utilisation des noms conditions	147
Utilisation de NOHANDLE	148
RESP2	149
asktime	151
FORMATTIME	152
START	156
RETRIEVE	158
CONDITIONS EXCEPTIONNELLES	159
<b>LA GESTION DES FICHIERS</b>	<b>160</b>
OBJECTIFS DU CHAPITRE	161
LA GESTION DES FICHIERS	162
LES COMMANDES D'E/S	163
LE READ	164
EXEMPLE	167
LE WRITE	168

LE READ for UPDATE	169
LE REWRITE	170
EXEMPLE	171
LE DELETE	172
EXEMPLES	174
UNLOCK	175
LE BROWSING	176
LE STARTBR	177
LE READNEXT	178
EXEMPLE	179
LE READPREVIEW	180
LE ENDBR	181
LE RESETBR	182
LES CONDITIONS EXCEPTIONNELLES	183
LES CONDITIONS EXCEPTIONNELLES	184
<b>LES DONNES TEMPORAIRES</b>	<b>185</b>
OBJECTIFS DU CHAPITRE	186
LES DONNÉES TEMPORAIRES	187
UTILISATION DES DONNÉES TEMPORAIRES	189
WRITEQ TS	190
EXEMPLE	192
READQ TS	193
EXEMPLES	194
DELETEQ TS	195
LES CONDITIONS EXCEPTIONNELLES	196
<b>LES DONNEES TRANSITOIRES</b>	<b>197</b>
OBJECTIFS DU CHAPITRE	198
LES DONNÉES TRANSITOIRES	199
DESTINATION EXTRA PARTITION	200
DÉCLENCHEMENT AUTOMATIQUE DE TACHE	204
EXEMPLE	205
WRITEQ TD	207
READQ TD	208
DELETEQ TD	209
LES CONDITIONS EXCEPTIONNELLES	210
L'ENVIRONNEMENT CLIENT/SERVEUR	212
UTILISATION DU CURSLOC	219
DFHBMSCA	220

<b>LES ATTRIBUTS</b>	<b>226</b>
<b>INTRODUCTION</b>	<b>230</b>
<b>LISTE DES SIGLES</b>	<b>231</b>
<b>liste des sigles</b>	<b>233</b>

# INTRODUCTION

## OBJECTIFS DU CHAPITRE

---

- Rappels
  - Historique de CICS
  - Le temps réel
  - Système DB/DC
- Présenter CICS, moniteur TP
- Exécution d'un programme à CICS



# HISTORIQUE DE CICS

---

C.I.C.S : Customer Information Control System

- CICS a été développé dans les années 70 par IBM après IMS
- Les versions ont évolué depuis la version 1.0 en passant par la V 1.7, V2.1.2 puis CICS/TS (V 3.1.0...3.3.0, 4.1.0 puis 5.1.0 appelé :

***Cics Transaction Serveur For OS/390***

***CICS TS 1.3 puis TS 2.2 (2002), CICS/TS 3.1 (2005)  
CICS/TS 3.2 (2006) ; CICS/TS 3.3 (2007)***

- CICS existe sur différentes plates-formes:
  - CICS/OS2
  - CICS/400
  - CICS/6000
  - CICS/9000
  - CICS sur SUN, DIGITAL etc ...

# PRÉSENTATION DE CICS

---

- CICS est un moniteur TP orienté Data Communication (DC)
- Il permet le traitement **temps réel**
- CICS/TS utilise :
  - Les méthodes d'accès réseau VTAM
  - Les méthodes d'accès fichier VSAM, DL1, DB2
  - Les langages COBOL, Assembleur et PL1, C et C++
  - Le Command Level comme interface de programmation

## LE MONITEUR TP CICS

---

En temps réel :

- les données sont saisies sur un terminal
- Il y a traitement unitaire de l'information
- le traitement est immédiat
- l'utilisateur peut envoyer ou recevoir des données
- les ressources sont partagées entre utilisateurs.

Il faut donc un interface pour :

- garder la trace du terminal
- accepter une donnée dès qu'elle est validée
- faire appel au programme adéquat,
- contrôler le partage des ressources.

## LE MONITEUR TP CICS

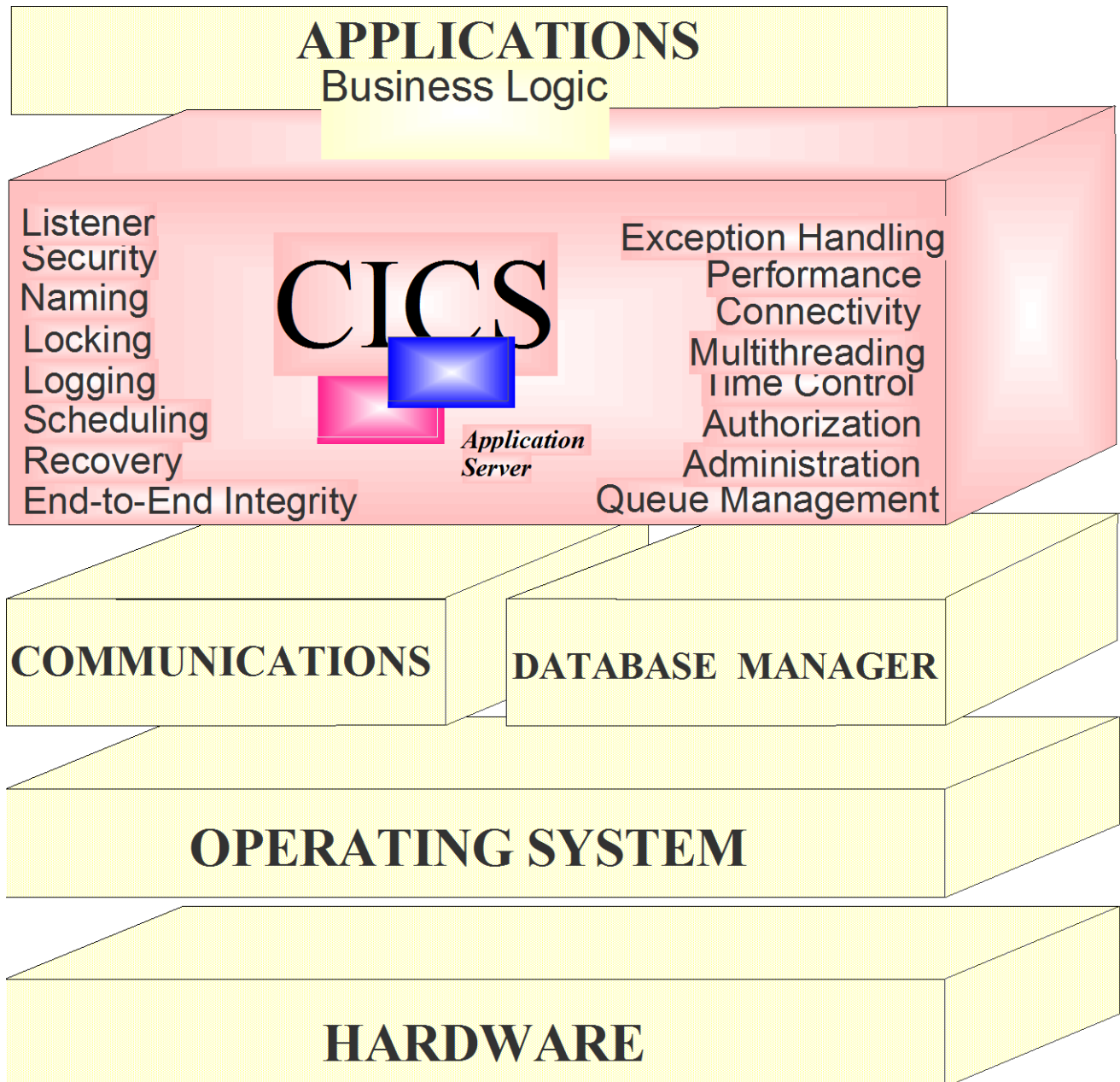
---

Permet de :

- recevoir une information à partir d'un terminal
- transférer cette information à un programme d'application
- gérer les demandes d'accès aux données
- transmettre au terminal la réponse du programme
- permettre le travail simultané de plusieurs terminaux avec :
  - le même programme sollicité plusieurs fois
  - des programmes différents accédant aux mêmes ou différentes ressources

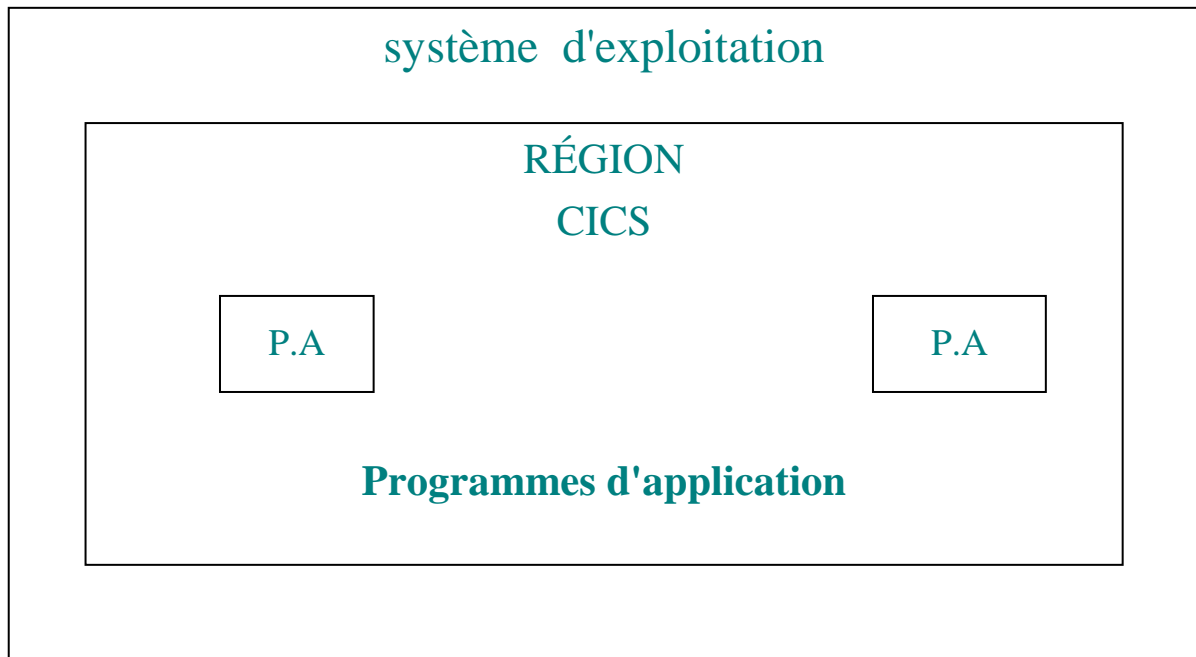
# LE MONITEUR TP CICS

---



# SYSTÈME/CICS/PROGRAMME

---



- CICS s'exécute sous le contrôle du système d'exploitation.
- CICS sert d'interface entre les applicatifs qui s'exécutent dans sa région et le système d'exploitation
- Un programme CICS est sous programme de CICS.

## UN PROGRAMME CICS

---

Un programme CICS n'est pas maître de son environnement.

- Les fichiers ne sont pas déclarés dans le programme.
- Toutes les demandes d'entrée sortie passent par CICS.
- Un programme d'application CICS ne pourra travailler que dans un **environnement prédéfini**.

# ORGANISATION DE CICS



## OBJECTIFS DU CHAPITRE

---

Positionnement de CICS dans un système d'exploitation

Contenu d'une région CICS/TS

# CICS ET SYSTEME D'EXPLOITATION

---

CICS s'exécute dans sa propre région (espace adresse)

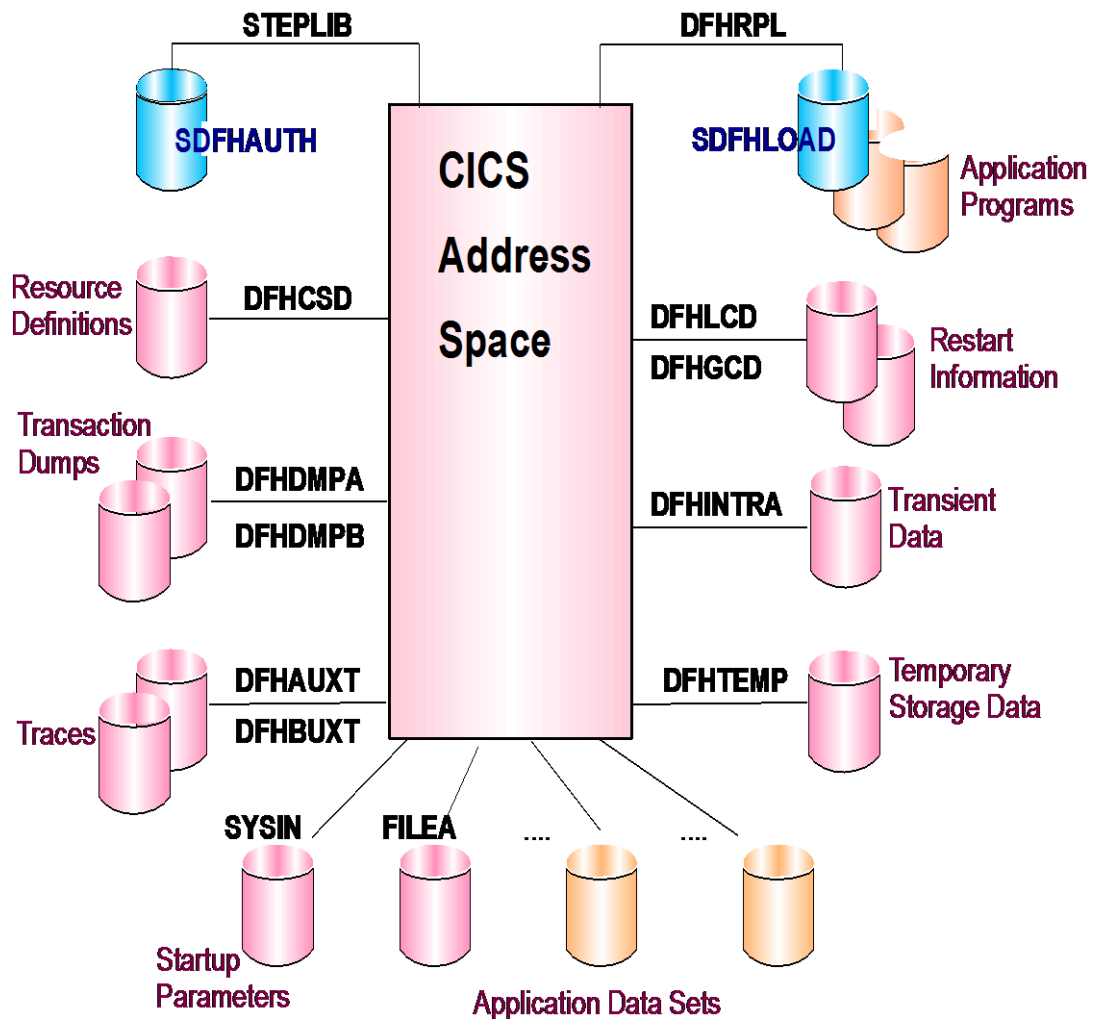
Pour le système d'exploitation, CICS n'est qu'un job batch à priorité élevée.

**z/OS**

VTAM			
B A T C H	D  B  2	C I C S	C I C S

# LES FICHIERS DE CICS/TS

---



## JOB D'INITIALISATION

---

Le job d'initialisation de CICS (DFHSIP) :

- effectue l'ouverture des fichiers système
- établit la communication avec les terminaux
- charge des programmes du noyau
- etc ...

### Exemple de JCL

```
//CICSPROD JOB CLASS=W,TIME=1440
.
.
//CICS      EXEC PGM=DFHSIP,PARM=(DSNCRCTZX,'SIT=01,...')
//STEPLIB   DD  DSN=CICS630.SDFHLOAD,DISP=SHR
.
//DFHRPL    DD  DSN=CICS630.SDFHLOAD,DISP=SHR
//          DD  ...
//DFHRSD    DD  DSN=CICS630.SDJ.DFHRSD,DISP=SHR
//DFHTEMP   DD  DSN=CICS630.SDJ.DFHTEMP,DISP=SHR
//DFHLCD    DD  DSN=CICS630.SDJ.DFHLCD,DISP=SHR
//DFHGCD    DD  DSN=CICS630.SDJ.DFHGCD,DISP=SHR
```

■

# COMPOSANTS D'UNE REGION CICS

---

## Zoom sur une région CICS/MVS



# COMPOSANTS D'UNE REGION CICS

---

Le NOYAU comprend :

Les modules de gestion CICS :

- des terminaux
- des mémoires
- des tâches
- des programmes
- des fichiers

...

ainsi que leurs tables associées.

La CSA (Common System Area) contient l'adresse des différents modules et tables CICS.

La CWA (Common Work Area) est une extension de la CSA dont la durée de vie est la même que celle de la région CICS

## COMPOSANTS D'UNE REGION CICS

---

La partie RESIDENT :

contient les programmes déclarés résidents (associés aux transactions fréquemment utilisées ou prioritaires),

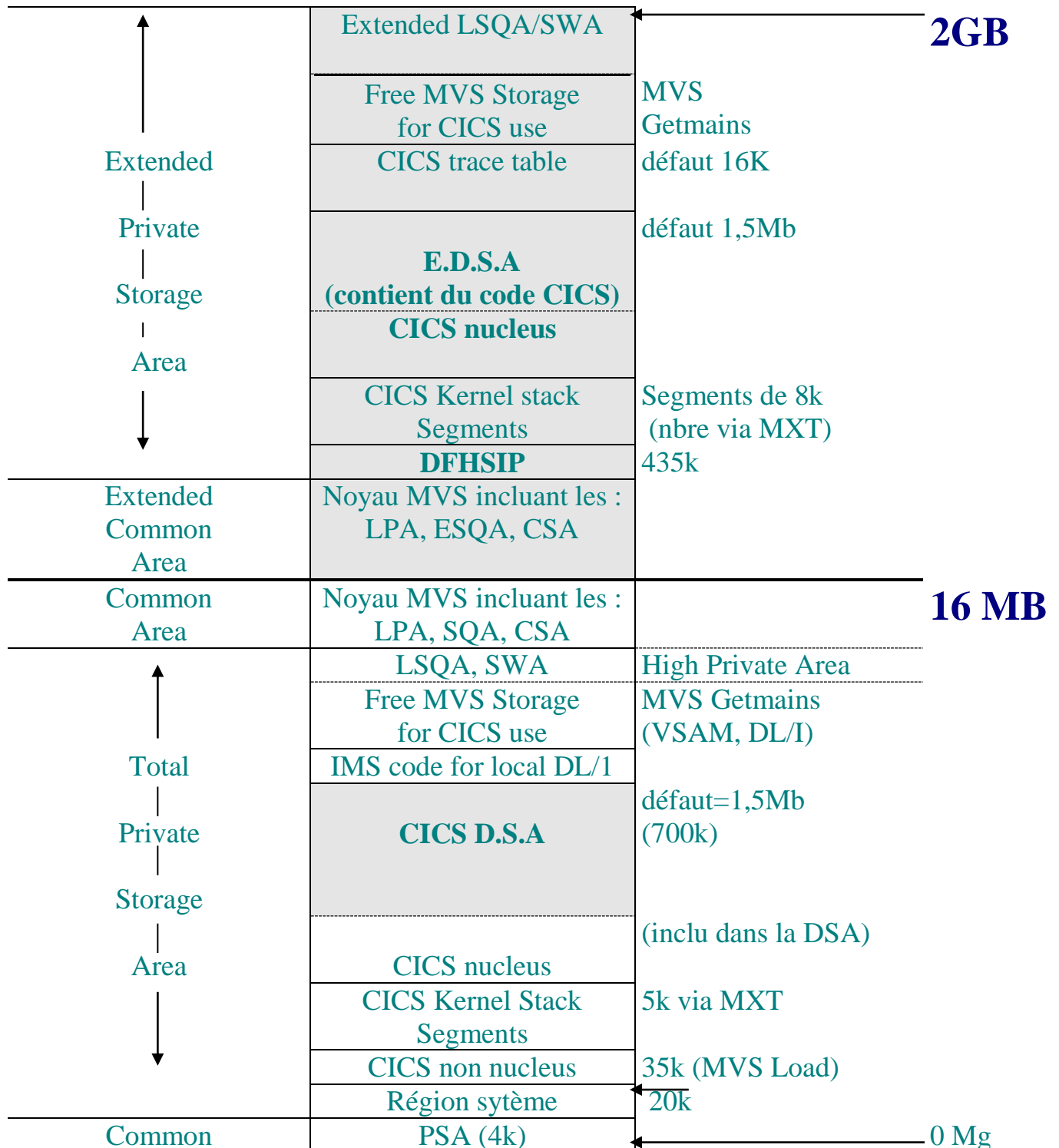
La DSA (Dynamic Storage Area) :

Mémoire disponible qu'utilise CICS pour l'allocation dynamique de zones mémoire.

Les programmes d'applications s'exécutent dans la DSA

Il existe 8 types de DSA dont la taille est paramétrée dans la System Initialization Table (SIT)

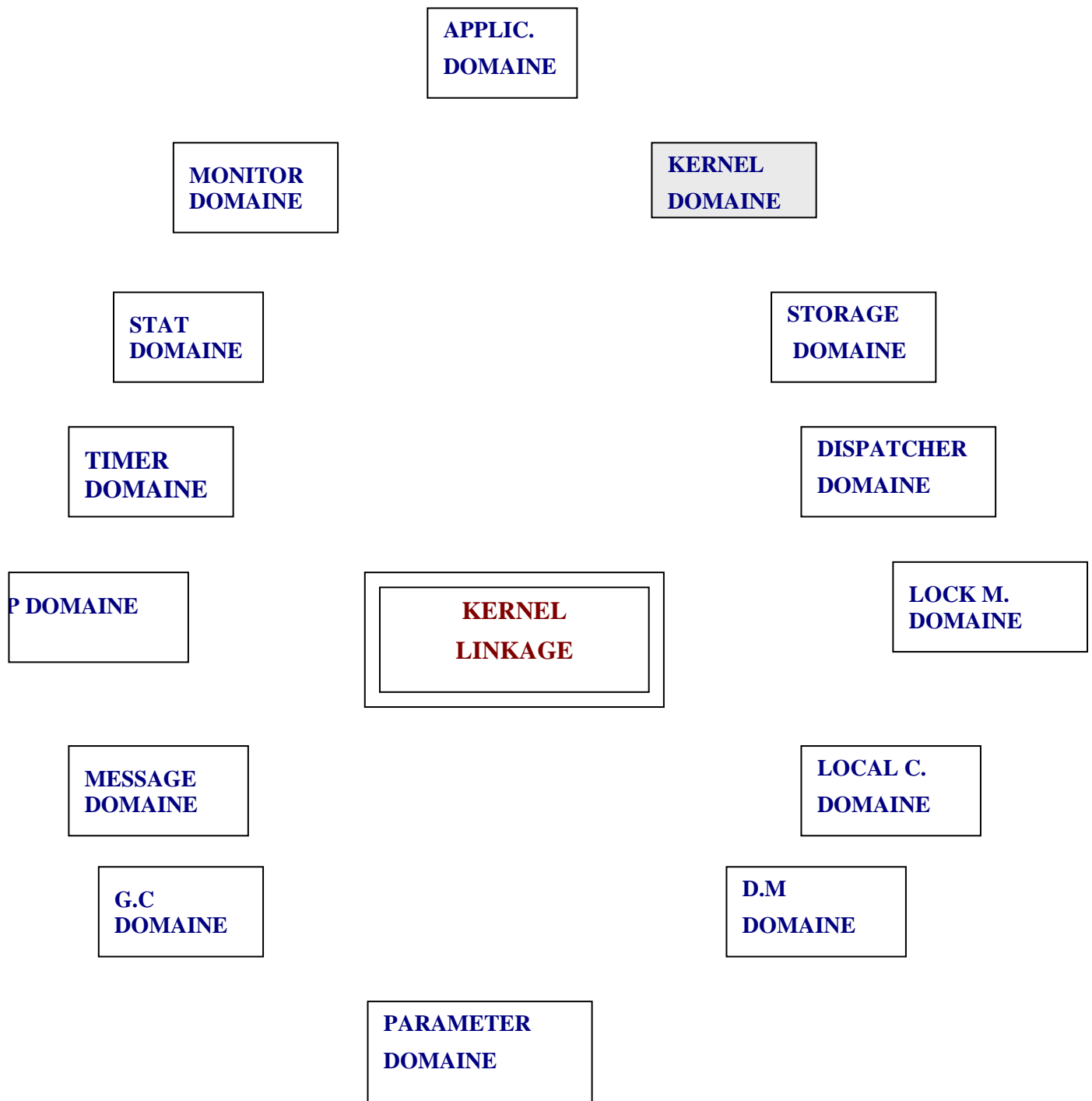
# LA RÉGION CICS/TS





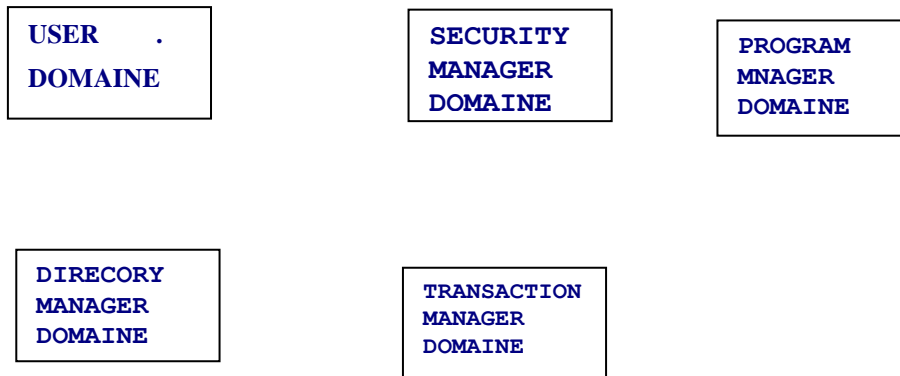
# LES DOMAINES DE CICS/TS V.3.3.0

---

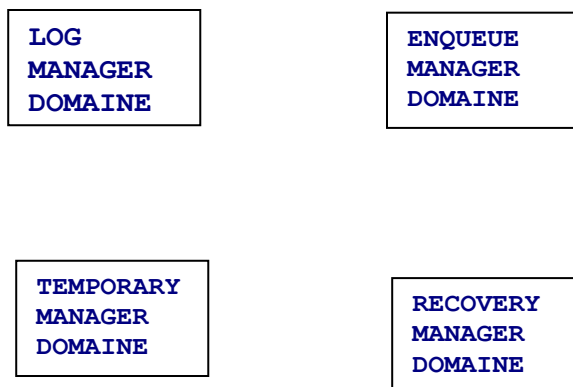


# LES NOUVEAUX DOMAINES EN CICS/TS

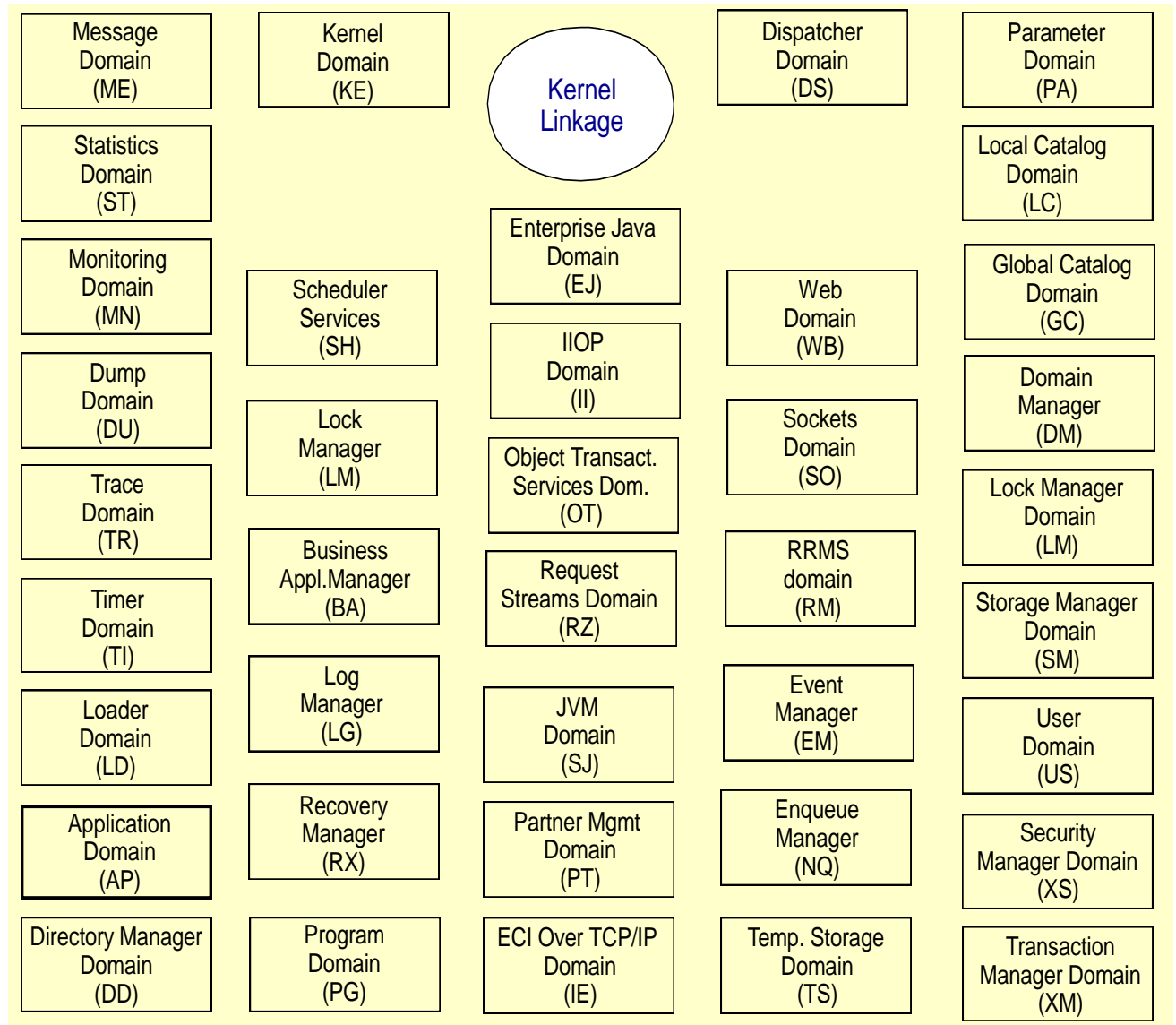
---



- 4 nouveaux domaines en CICS TS



# LES NOUVEAUX DOMAINES EN CICS/TS



# LA RÉGION CICS

## OBJECTIFS DU CHAPITRE

---

Montrer la modularité de CICS (programmes et tables)

Expliquer le rôle des différents modules CICS

Comprendre le déroulement d'une transaction CICS

Définir les notions de tâche et réentrance

*Pour des raisons pédagogiques les schémas utilisés seront ceux de la région CICS/MVS et non de CICS/TS*

# MODULARITÉ DE CICS

---

Une région CICS est composée de :

1. Modules de gestion qui exécutent les commandes CICS communiquent entre eux communiquent avec le système
2. Tables systèmes qui permettent de définir l'environnement CICS sont utilisées par les modules de gestion CICS
3. Zones mémoire :  
Allouées dynamiquement par CICS utilisées par CICS et par les programmes d'application

## MODULARITÉ DE CICS

---

Les différents modules CICS important pour le déroulement d'une transaction sont les suivants :

TCP (TCT/TCTTE)

SCP (SM en CICS/TS)

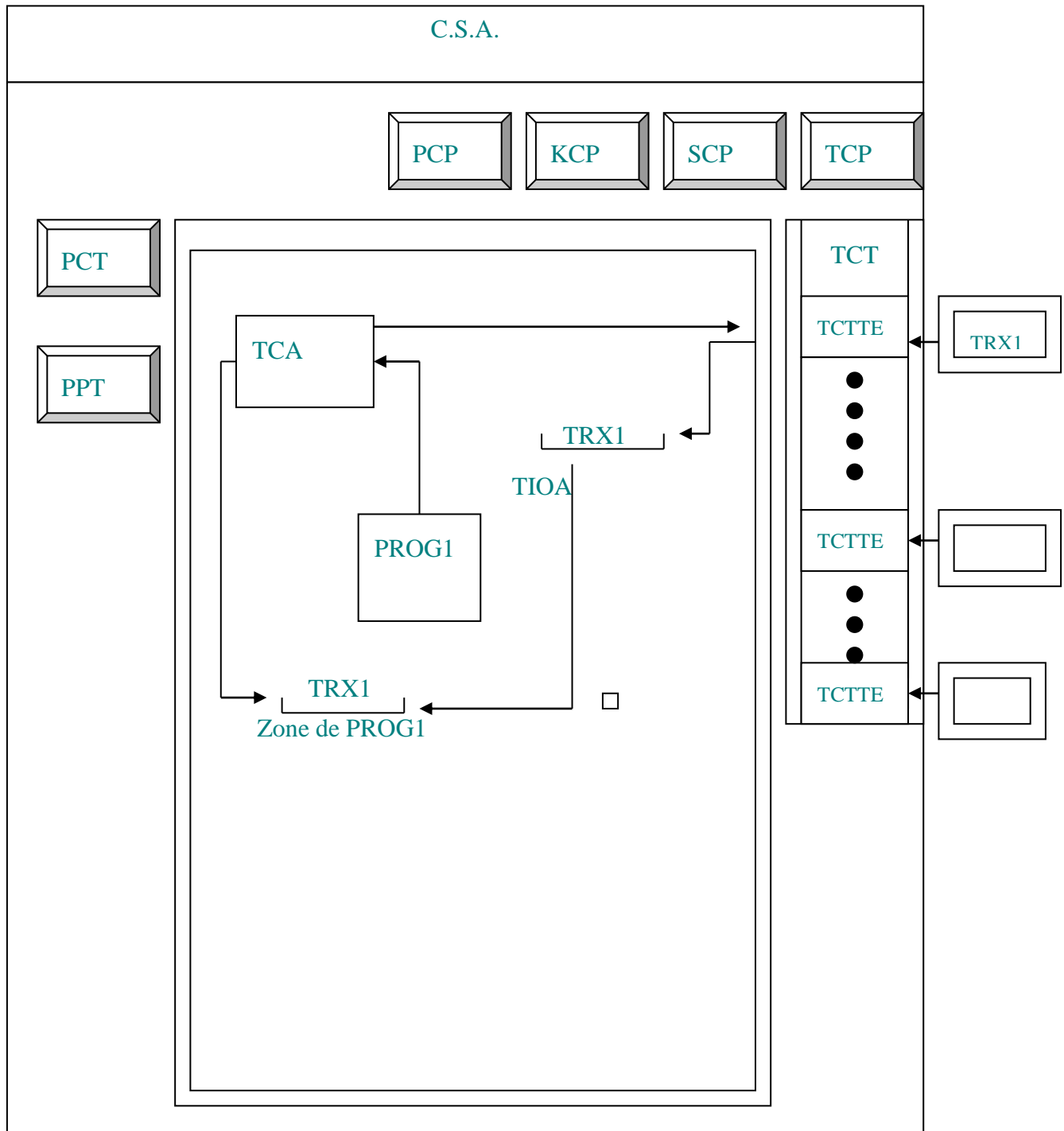
KCP (DS en CICS/TS) PCT/**TXD**

PCP (AP en CICS/TS) PPT/**PDT**

FCP/ FCT

*Cf. fin de chapitre pour plus de détails*

# DÉROULEMENT D'UNE TRANSACTION CICS





# LA TRANSACTION CICS

---

Une transaction CICS est :

identifiée par un code transaction de 4 caractères défini dans la PCT

un programme exécutée sous le contrôle de CICS pouvant lui même faire appel à d'autres programmes

## LE KCP OU DS DOMAINE

---

Le KCP (tasK Control Program)

contrôle le code transaction,  
crée la tâche (TCA) associée à la transaction,  
relie la tâche au terminal émetteur.

La tâche CICS est l'unité interne de traitement d'une transaction; c'est l'ensemble des éléments capables de réaliser pour le compte d'un terminal l'exécution d'une transaction.

Une tâche est matérialisée par un bloc de contrôle qui lui est propre : la TCA (Task Control Area), elle a un numéro unique.

La TCA peut avoir une extension (La TWA)

## LA PCT (PROGRAM CONTROL TABLE)

---

La PCT contient :

les codes transaction

le nom du premier programme associé pour chaque code transaction.

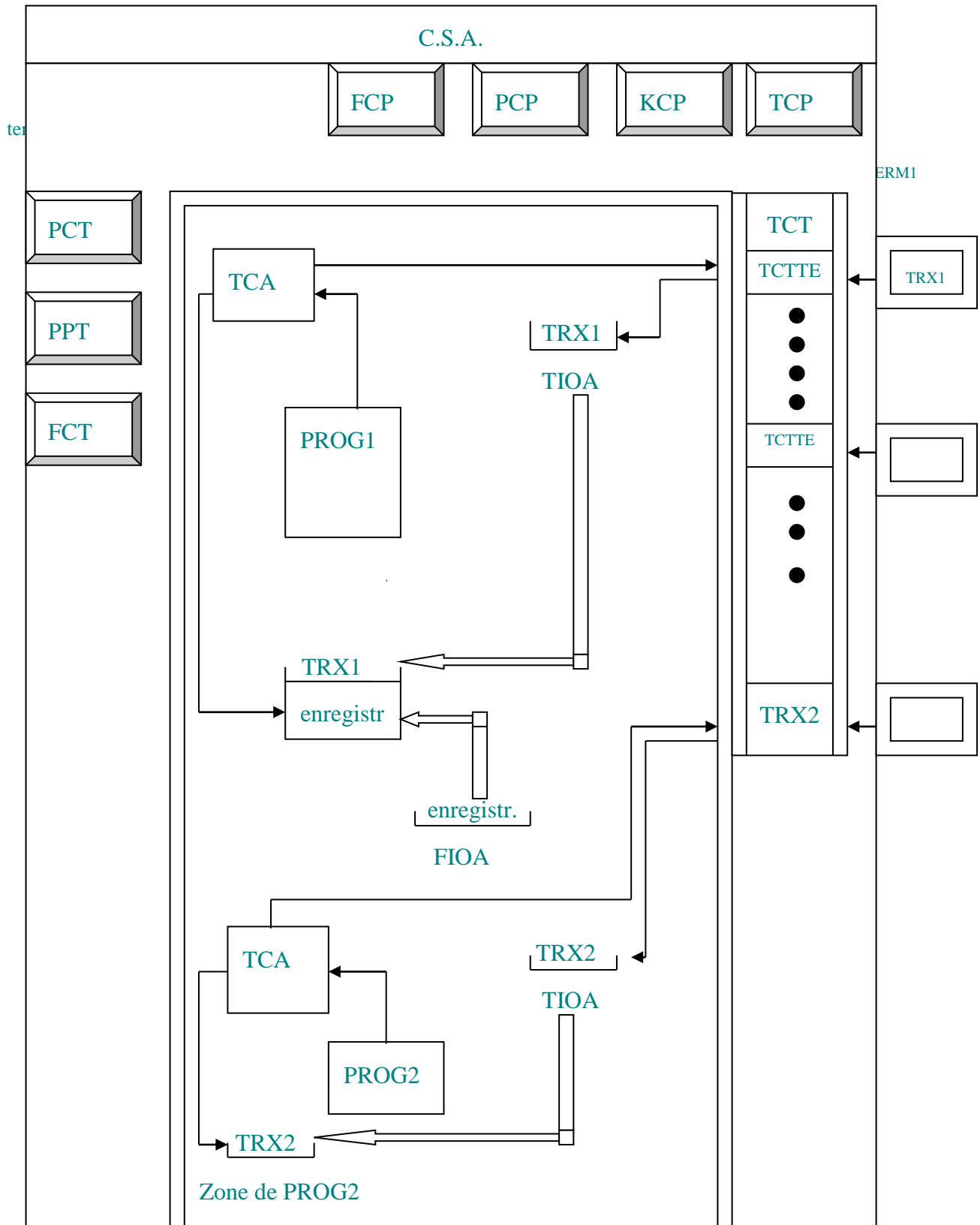
## LE PROGRAMME CICS

---

Un programme CICS est :

- un sous programme du système CICS,
- partagé éventuellement entre plusieurs tâches,
- obligé de passer par CICS pour toutes les communications avec leur environnement,
- un poste de la table PPT.

# RÉGION CICS : TACHES SIMULTANÉES



## RÉGION CICS : TACHES SIMULTANEEES

---

La tâche associée à la transaction TRX1 pour le terminal TERM1 est en attente sur une ressource, une autre tâche peut démarrer.

Un opérateur saisit le code transaction TRX2.

Le TCP effectue le POLLING

Le code transaction est récupéré dans la TIOA.

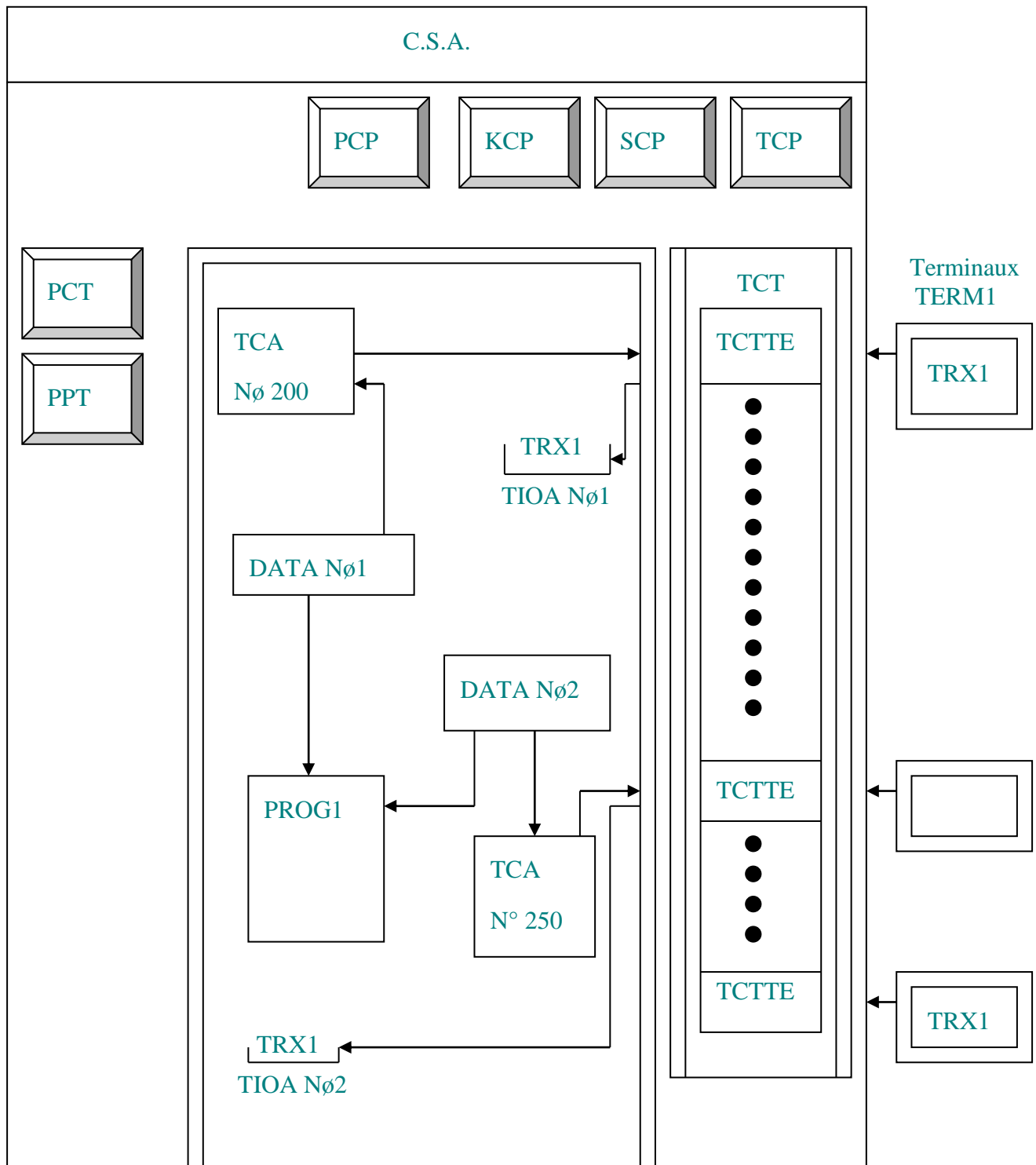
Sa définition est vérifiée par le KCP dans la PCT.

Une TCA est créée (tâche).

Le PCP charge le programme PROG2.

Le PROG2 commence à s'exécuter.

# LA RÉGION CICS



## TCP (TERMINAL CONTROL PROGRAM)

---

Le gestionnaire des terminaux TCP :

- gère les lignes et les terminaux,
- contrôle les communications avec les terminaux à l'aide de la TCT (polling, addressing),
- crée en mémoire une zone réceptrice TIOA, relie la TCTTE émettrice à la TIOA.

Terminaux CICS TCT

Un terminal peut dialoguer avec CICS s'il est défini dans la table TCT (Terminal Control Table).

La TCT, pour chaque terminal CICS, comporte un poste appelé TCTTE, grâce auquel elle :

- le référence par un nom,
- en décrit les caractéristiques,
- indique son état actuel (mise sous tension),
- précise si une transaction est en cours.



# LA GESTION DE LA MÉMOIRE

---

A l'aide de son module SCP (Storage Control Program)  
**Storage Manager**

CICS gère dynamiquement la mémoire:

par acquisition au moment voulu,

par libération dès que possible des zones

## LE PCP (APPLICATION DOMAINE)

---

PCP : Program Control Program

- localise le programme applicatif
- le charge si nécessaire
- incrémente le compteur de la PPT
- passe le contrôle au programme.

Le programme d'application :

- traite le message (MAP) envoyé par le terminal
- demande à CICS de le transférer dans une zone de travail définie par le programme en WSS.

## PPT (PROCESSING PROGRAM TABLE)

---

La PPT contient :

- le nom des programmes,
- leur emplacement en mémoire,
- leur emplacement sur disque,
- le langage,
- les compteurs d'utilisation.

### **Dispatcher Domain**

Lorsqu'un programme est en attente :

- le KCP reprend le contrôle,
- cherche un autre traitement à exécuter;
- le TCP est prêt à interroger les terminaux.

## TRANSACTION / TACHE

---

Une transaction CICS est une unité logique de traitement identifiée par :

- le code transaction

- le premier programme à exécuter.

Plusieurs terminaux peuvent solliciter la même transaction. CICS identifie le lien entre une transaction et un terminal en créant une **tâche**.

Une tâche correspond à l'ensemble des ressources allouées à un terminal pour le traitement d'une transaction.

CICS contrôle le travail simultané de plusieurs terminaux, donc de plusieurs tâches.

On appelle cela le multi tasking de CICS.

## RÉENTRANCE (MULTI-THREADING)

---

Plusieurs tâches peuvent demander à exécuter le même programme.

Une seule copie du programme est chargée en mémoire.

Les zones de travail sont dupliquées pour chaque tâche.

Une tâche s'interrompt sur chaque ordre CICS pour "passer la main" à CICS qui choisit une autre tâche à exécuter.

# FCP (FILE CONTROL PROGRAM)

---

Le FCP :

gère l'accès aux fichiers.

place l'enregistrement du fichier dans une zone de travail du programme (à la demande du programme d'application).

Un fichier CICS est défini dans la FCT

La FCT : File Control Table

On y définit :

- le DDname et DSname du fichier VSAM
- la méthode d'accès,
- la taille des buffers (VSAM),
- les caractéristiques du fichier
- les autorisations d'accès au fichier

## FIN D'UNE TACHE

---

Les zones mémoires allouées sont libérées à l'exception de la TIOA

La réponse destinée au terminal est conservée dans la TIOA jusqu'à réception.

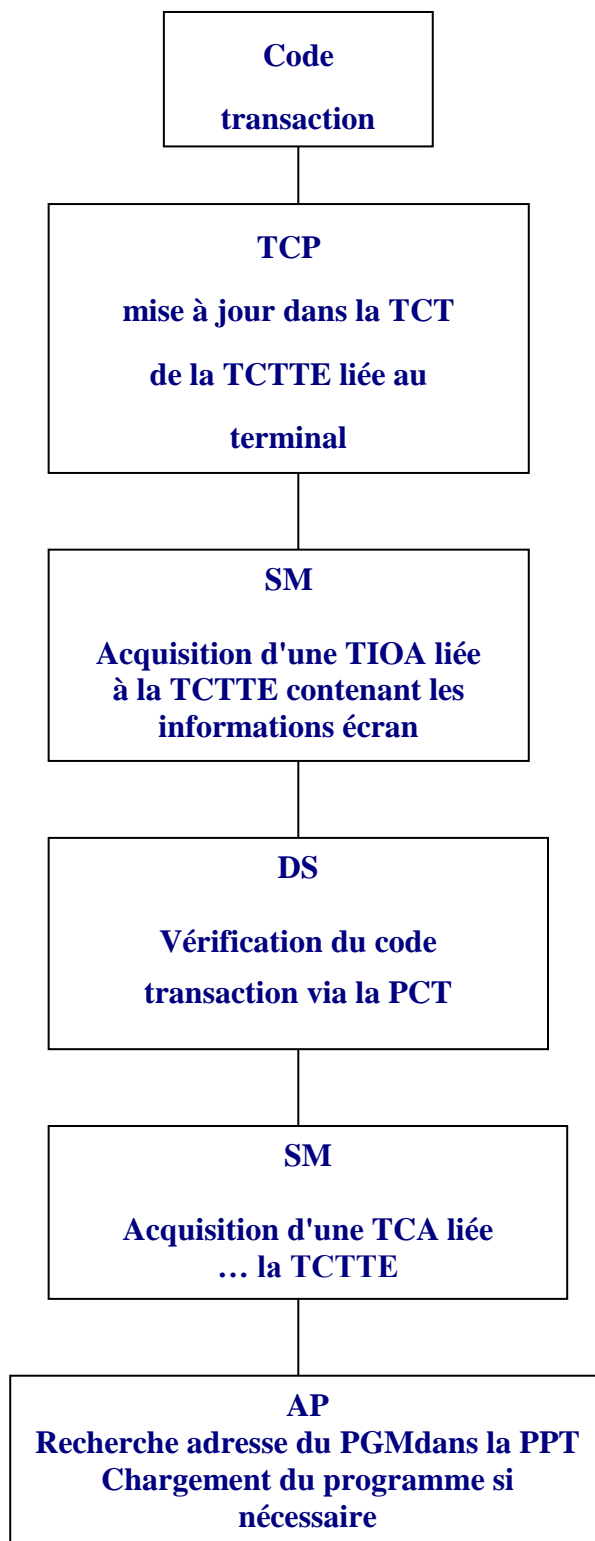
La TIOA est alors libérée.

La TCA qui relie l'ensemble des ressources pour une transaction et un terminal est libérée.



# SCHEMA RECAPITULATIF DU DEMARRAGE D'UNE TACHE

---



# **BASIC MAPPING SUPPORT**

---

## **(B.M.S)**

## OBJECTIFS DU CHAPITRE

---

- Présenter le terminal 3270
- Préciser le rôle de BMS (Basic Mapping Support)
- Codification de la définition d'une Map :  
définition de mapset, de map, de champs
- Commandes permettant l'envoi et la réception  
de MAP

## LE TERMINAL 3270

---

Le clavier est composé de 3 types de touches :

- Saisie
- Contrôle d'entrée
- Permettant de transmettre les données :
  - ENTER transmet son code et les données modifiées
  - PF1 à PF24 transmettent leur code et les données modifiées
  - PA1 et PA2 ne transmettent que leur code
  - CLEAR qui transmet son code et efface l'écran

Une zone, ou champs, d'écran est toujours composée de deux éléments :

- un octet dit **attribut**
- la zone elle même

# LE TERMINAL 3270

---

Contenu de l'octet attribut :

Signification des bits

x	x	u/p	A/N	Affich	res	MDT	
0	1	2	3	4	5	6	7

Bits	signification
0	système
1	système, toujours à 1
2	“00”non protégé alphanum.
3	“01”non protégé numérique
	“10”protégé
	“11”protégé autoskip
4	“00”affichage normal non détectable
5	“01”affichage normal détect.
	“10”double brillance
	“11”non affichable (dark)
6	Toujours à 0
7	Réputé modifié ou non : MDT on si 1 MDT off si 0

# LE TCP (TERMINAL CONTROL PROGRAM)

---

## Rappels :

- Il assure la gestion des lignes et des terminaux (utilisation de la TCT).
- Il pourvoit au transfert de données entre un programme applicatif et un terminal via la TIOA (Terminal Input Output Area).

La TIOA peut être utilisée en

- mode LOCATE emploi direct du contenu de la TIOA au moyen d'un pointeur du programme
- mode MOVE transfert du contenu de la TIOA dans une zone de travail propre au programme

## LE TCP (TERMINAL CONTROL PROGRAM)

---

- Peut gérer deux types de message :
  - sans format (hors BMS) : l'utilisateur peut saisir les données n'importe où sur l'écran
  - avec format (avec BMS) : l'utilisateur saisit les données dans des champs prédéfinis
- Des commandes associées à chaque type existent, nous nous intéresserons principalement aux commandes liées à BMS.



### Rôle de BMS lors de l'émission d'un message

**T.C.P**

**B.M.S**

### Rôle de BMS lors de la réception d'un message

**T.C.P**

**B.M.S**

BMS, interface entre le TCP et le programme, rend le programme indépendant :

- des caractéristiques des terminaux
- du dessin d'écran.
- Un MAPSET est composé de une ou plusieurs MAPS
- Une MAP est composée d'un ensemble de champs

C'est l'unité d'échange entre un programme et un terminal.

- le champ est la donnée élémentaire d'un écran

Un mapset est la fusion d'une map physique et d'une map logique

## LA MAP PHYSIQUE

---

- Est défini dans la PPT.
- Sert à BMS afin de formater les données en entrée et en sortie
- Est assemblée et link editée, contient des informations sur chaque champ
- Appartient à un mapset qui
  - Se trouve dans une bibliothèque des load modules
  - Est chargé par BMS au moment de l'exécution

A toute map physique est associée une map symbolique.

## LA MAP SYMBOLIQUE

---

- Contient la description logique des champs et zones associées
- doit être ajoutée dans le source du programme (COPY)
- Les données sont traitées à l'intérieur de champs séparés.
- Chaque donnée est précédée de trois caractères pour les écrans monochromes.

Exemple : Un champ CHP1 d'une map STAG sera subdivisé par BMS de la manière suivante :

\* Map symbolique en entrée

01 STAGI.

05 CHP1L (longueur de la zone)

05 CHP1F (flag de saisie)

05 CHP1I (zone de saisie)

\* Map symbolique en sortie

01 STAGO REDEFINES STAGI.

05 CHP1L (longueur de la zone)

05 CHP1A (octet attribut)

05 CHP1O (zone à afficher)

## DÉFINITION D'UN ÉCRAN

---

La définition s'effectue à l'aide de macro instructions assembleur :

- DFHMSD  
caractéristiques d'un MAPSET
- DFHMDI  
caractéristiques d'une MAP
- DFHMDF  
caractéristiques d'un CHAMP

les mêmes macro instructions servent à créer les maps  
symboliques et physiques  
(à un paramètre près ...)

Format général d'une macro instruction :

Colonne : 1	10	16	72
LabelX	MacroX	paramètre1,paramètre2,.. paramètres...	...., *

## DÉFINITION D'UN MAPSET DFHMSD

---

```
LABEL DFHMSD TYPE={DSECT/MAP/FINAL}  
                [,MODE={IN/OUT/INOUT}]  
                [,LANG={ASM/COBOL/PLI}]  
                [,CTRL={FREEKB,FRSET}]  
                [,STORAGE=AUTO]  
                [,TIOAPFX=(NO/YES)]  
                [,TERM=type]  
                [,EXTTAT=YES]
```

## DÉFINITION D'UN MAPSET DFHMSD

---

- LABEL : nom du mapset.
- TYPE : fonction de la macro.
  - TYPE=DSECT crée une map symbolique.
  - TYPE=MAP produit une map physique.
  - TYPE=FINAL détermine la fin du mapset.
- MODE : utilisation du mapset
  - IN pour des messages en entrée
  - OUT pour des messages en sortie
  - INOUT pour des messages d'entrée/sortie

## DÉFINITION D'UN MAPSET DFHMSD

---

- LANG : langage dans lequel sera générée la map symbolique
- CTRL : paramètre associé à un mapset défini OUT ou INOUT
  - FREEKB déverrouillage du clavier une fois la map affichée
  - FRSET restauration des attributs des champs comme non modifiés
- STORAGE = AUTO zones mémoire séparées pour les maps symboliques si plusieurs maps par mapset
- TIOAPFX = YES
  - permet de transmettre au programme le préfixe de la TIOA (12 octets)
- TERM : type de terminal avec lequel on veut travailler (par défaut 3270)



## DÉFINITION D'UNE MAP DFHMDI

---

```
[LABEL] DFHMDI
      [COLUMN=numéro]
      [,LINE=numéro]
      [,SIZE=(ligne,colonne)]
      [,CTRL=(FREEKB,FRSET)]
      MAPATTS=(COLOR,HILIGHT),
      DSATTS=(COLOR,HILIGHT), CURSLOC=YES
```

## DÉFINITION D'UNE MAP DFHMDI

---

- LABEL : nom de la MAP
- COLUMN et LINE : coordonnées du point origine de la map
- SIZE : taille de la map en nombre de lignes et colonnes
- Les paramètres identiques à ceux de la macro instruction DFHMSD ont la même signification ; s'ils sont codés, ils priment, pour cette map sur ceux de la macro DFHMSD.

## DÉFINITION D'UN CHAMP DFHMDF

---

```
[LABEL] DFHMDF
      [POS=(ligne,colonne)]
      [,ATTRB=
      ([{ ASKIP PROT UNPROT[,NUM]}],
      {BRT NORM DRK}][,FSET][,IC)]
      [,INITIAL='constante']
      [,LENGTH=longueur]
      [,PICIN='picture']
      [,PICOUT='picture']
      [,JUSTIFY=([{LEFT RIGTH}]
      [, {BLANK ZERO}])]
      [,OCCURS=n]
      [,GRPNAME=nom]
      [,COLOR=nom]
```

## DÉFINITION D'UN CHAMP DFHMDF

---

- LABEL : nom du champ, si on veut pouvoir le référencer par programme
- POS : position de l'attribut du champ.
- ATTRB : caractéristiques de la zone
  - ASKIP : zone protégée et saut du curseur automatique
  - PROT : zone protégée
  - UNPROT : zone non protégée
  - NUM : zone numérique
  - BRT : affichage en double brillance
  - NORM : affichage en brillance normale
  - DRK : pas d'affichage
  - FSET : champ toujours considéré comme modifié (MDT ON)
  - IC : positionnement initial du curseur

## DÉFINITION D'UN CHAMP DFHMDF

---

- INITIAL : valeur initiale du champ en sortie
- LENGTH : longueur du champ en octets (1 à 256), attribut exclu
- PICOUT : pour mapset OUT ou INOUT, définition du format d'affichage à l'écran (masque d'édition)
- JUSTIFY :
  - LEFT : donnée en entrée cadrée à gauche
  - RIGHT : donnée en entrée cadrée à droite
  - BLANK : padding à blanc
  - ZERO : padding à zéro
  - OCCURS : Le champ est dupliquée n fois.

### EXEMPLES :

**Z1     xxx xxx xxx xxx xxx xxx**

**Z1     DFHMDF ..... LENGTH=3,OCCURS=6**

**Z2     20032006**

**JJ     DFHMDF POS=(1,c),LENGTH=2,GRPNAME=DATE,ATTRB=NUM**

**MM     DFHMDF POS=(1,c),LENGTH=2,GRPNAME=DATE**

**SS     DFHMDF POS=(1,c),LENGTH=2,GRPNAME=DATE**

**AA     DFHMDF POS=(1,c),LENGTH=2,GRPNAME=DATE**

## FIN D'UN MAPSET DFHMSD

---

[LABEL] DFHMSD TYPE=FINAL

Obligatoirement la dernière macro instruction à fournir pour un mapset

# EXEMPLE DE MAP

---

1234567890123456789012345678901234567890123456789012345678901234567890

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

## INTERROGATION DU FICHIER STAGIAIRE

NOM : \_\_\_\_\_ nom du stagiaire à saisir

NUMERO : xxxxx no de compte à saisir

SOLDE : xxxx,xx (alimenté par le programme lorsque le nom  
et le N° de compte sont valides)

### MESSAGE :

ENTER : VALIDATION

PF3 : RETOUR MENU PF1 : HELP

PF12 : FIN DE TRANSACTION/

# EXEMPLE DE MAP

1	9	16	72
<b>MAPSET</b>	<b>DFHMSD</b>	TYPE=MAP,MODE=INOUT,LANG=COBOL, CTRL=(FREEKB,FRSET),STORAGE=AUTO,TIOPFX=YES	*
*			
<b>MAP</b>	<b>DFHMDI</b>	SIZE=(24,80),CTRL=(FREEKB,FRSET),MAPATTS=(COLOR,HILIGHT), DSATTS=(COLOR,HILIGHT), <b>CURSLOC=YES</b>	*
	<b>DFHMDF</b>	POS=(1,17) ,ATTRB=ASKIP,LENGTH=34, INITIAL='INTERROGATION DU FICHIER STAGIAIRES'	*
*			
	DFHMDF	POS=(4,1),ATTRB=ASKIP,LENGTH=08, INITIAL='NOM :'	
<b>NOM</b>	DFHMDF	POS=(4,10),ATTRB=(UNPROT,IC),LENGTH=20, HILIGHT=UNDERLINE	
*			
	DFHMDF	POS=(4,31),ATTRB=ASKIP,LENGTH=1	
	DFHMDF	POS=(11,1),ATTRB=ASKIP,LENGTH=08, INITIAL='NUMERO : '	
<b>NUMBER</b>	NUMBER	POS=(11,10),ATTRB=(UNPROT,NUM),LENGTH=05	
	DFHMDF	POS=(11,16),ATTRB=ASKIP,LENGTH=1	
	DFHMDF	POS=(17,1),ATTRB=ASKIP,LENGTH=08, INITIAL='SOLDE :'	
<b>SOLDE</b>	DFHMDF	POS=(17,10),ATTRB=ASKIP,LENGTH=07,PICOUT='ZZZ9,99'	
	DFHMDF	POS=(23,1), ATTRB=ASKIP,LENGTH=09, INITIAL='MESSAGE :'	
<b>MESSAGE</b>	DFHMDF	POS=(24,01),ATTRB=(ASKIP,BRT),LENGTH=79,COLOR=YELLOW	
	<b>DFHMSD</b>	<b>TYPE=FINAL</b>	
	<b>END</b>		



## CRÉATION DE MAPS

---

- Après son codage, la map doit être soumise à un utilitaire pour que soient créées une map physique et une map symbolique.
- On spécifiera

**TYPE = MAP** pour la map physique

**TYPE = DSECT** pour la map symbolique

**TYPE = &SYSPARM** si l'utilitaire est prévu pour traiter une même source pour l'un et l'autre type

## MAP SYMBOLIQUE GENERATION

---

- Chaque macro DFHMDI d'un mapset donne lieu à la création d'une map symbolique.
- Les maps symboliques doivent être insérées par COPY dans le programme d'application
  - en WORKING STORAGE SECTION  
si mode MOVE utilisé
  - en LINKAGE SECTION  
si mode LOCATE utilisé

## MAP SYMBOLIQUE (GENERATION)

---

Chaque macro DFHMDI implique la génération des niveaux  
01 suivants :

- 01 CHPI si mapset en MODE=IN
- 01 CHPO si mapset en MODE=OUT
- 01 CHPI et 01 CHPO

si mapset en MODE=INOUT

( CHP étant le label de la macro instruction DFHMDI )

La zone CHPI va contenir toutes les informations de la map  
en entrée.

La zone CHPO va contenir toutes les informations de la map  
en sortie.

## MAP SYMBOLIQUE GENERATION

---

Pour chaque macro DFHMDF ayant un nom associé,  
CHAMP par ex., on aura

- si le champ est en entrée
  - CHAMPL : longueur **réelle** saisie pour le champ
  - CHAMPF : flag de saisie
  - CHAMPI : contenu en entrée  
(n octets, n = valeur du **paramètre LENGTH** de la macro)
- si le champ est en sortie
  - FILLER : non utilisé
  - CHAMPA : attribut
  - CHAMPO : contenu en sortie  
( n octets, n = valeur du paramètre LENGTH de la macro )

## MAP SYMBOLIQUE EXEMPLE

---

### COPY MAP1.

#### 01 MAP1I. ←

05 FILLER PIC X(12). préfixe TIOA

05 NOML PIC S9(04) COMP.

05 NOMF PIC X.

05 NOMI PIC X(20).

05 NUMEROL PIC S9(04) COMP.

05 NUMEROF PIC X.

05 NUMEROIPIC X(05).

05 SOLDEL PIC S9(04) COMP.

05 SOLDEF PIC X.

05 SOLDEI PIC X(07).

05 MESSAGEL PIC S9(04) COMP.

05 MESSAGEF PIC X.

05 MESSAGEI PIC X(79).

#### 01 MAP1O REDEFINES MAP1I

05 FILLER PIC X(12). préfixe TIOA

05 FILLER PIC XX.

05 NOMA PIC X.

05 NOMO PIC X(20).

05 FILLER PIC XX.

05 NUMEROA PIC X.

05 NUMEROO PIC X(05).

05 FILLER PIC XX.

05 SOLDEA PIC X.

05 SOLDEO PIC **ZZZ9,99**.

05 FILLER PIC XX.

05 MESSAGEA PIC X.

05 MESSAGEO PIC X(79).

## MODIFICATION DE L'ATTRIBUT

---

Liste de ses valeurs potentielles mise à notre disposition par  
COPY **DFHBMSCA** (Cf en annexe)

Nom symbolique	Attribut
DFHBMASB	askip,double brillance
DFHBMASF	askip, MDT on
DFHBMASK	askip
DFHBMBRY	double brillance
DFHBMDAR	zone invisible
DFHBMFSE	MDT on
DFHBMPRF	protégé, MDT on
DFHBMPRO	protégé
DFHBMUNN	non protégé, numérique
DFHBMUNP	non protégé
DFHBMEOF	effacement du champ
etc ...	

Exemple d'utilisation :

- MOVE DFHBMBRY TO champA

Permet de rendre le NOM brillant !

- IF chamF = DFHBMEOF

pour tester si l'utilisateur a effacé le champ qu'il avait saisi

# LA PROGRAMMATION

## OBJECTIFS DU CHAPITRE

---

- Développer un programme CICS,
- Description du contenu du bloc EIB
- Gestion des conditions exceptionnelles
- Comprendre le mode pseudo conversationnel
- Exemple de programme CICS



## HLPI : High Level Programming Interface

### Syntaxe d'une commande CICS

```
EXEC CICS .....  
      .....  
      .....  
END-EXEC.
```

### Les commandes CICS :

- Permettent de faire des requêtes à CICS,

## LE PRÉCOMPILATEUR DFHECP1\$

---

Le traducteur de commandes transforme les  
"EXEC . . . . . END-EXEC" en instructions COBOL.

Chaque EXEC est remplacé dans le source par

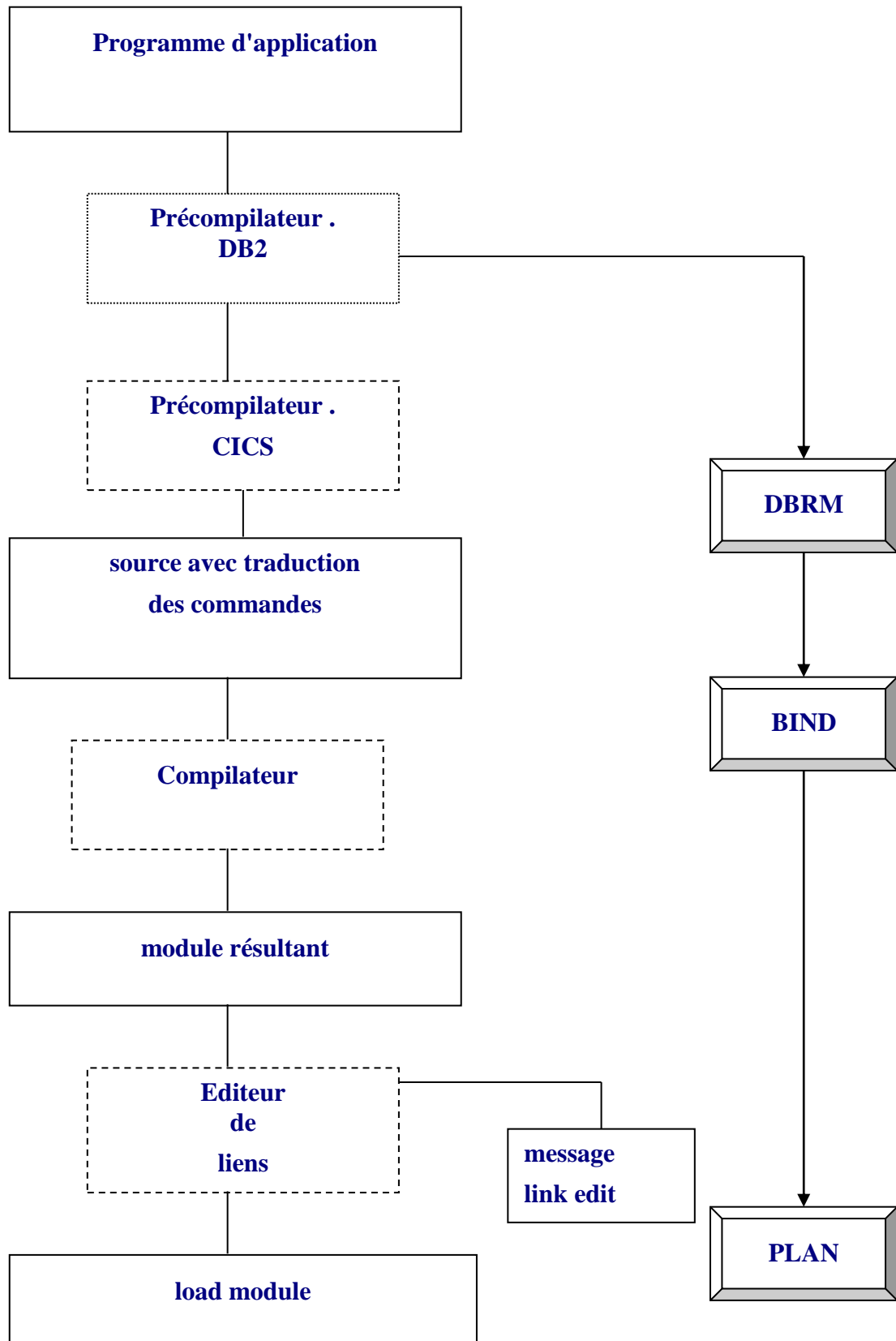
1. un CALL à une routine de l'interface un ou plusieurs  
MOVE
2. HLPI.

Le source est aussi complété par :

- l'ajout du bloc EIB en tête de Linkage Section,
- la déclaration des paramètres  
(bloc DFHEIVAR) utilisés par les CALL à l'interface  
HLPI.

# ÉTAPES POUR LA CRÉATION D'UN LOAD

---





## LE BLOC EIB

---

### EIB : Execute Interface Block

- Le bloc EIB créé au début de la tâche est adressé automatiquement par CICS
- CICS rafraîchit automatiquement le bloc EIB qui contient des informations exploitable par le programme d'application

## CONTENU DU BLOC EIB

---

déplacement	champs	longueur
0	EIBTIME	4
4	EIBDATE	4
8	EIBTRNID	4
12	EIBTASKN	4
16	EIBTRMID	4
20	EIBRSVD1	2
22	EIBCPOSN	2
24	EIBCALEN	2
26	EIBAID	1
27	EIBFN	2
29	EIBRCODE	6
35	EIBDS	8
43	EIBREQID	8
51	EIBRSRCE	8
59	EIBSYNC	1
60	EIBFREE	1
61	EIBRECV	1
Etc ...		
...	EIBRESP	4
...	EIBRESP2	4

# LE BLOC EIB

---

## Principaux champs utilisés

<b>EIBAID</b>	: contient la touche de fonction sollicitée lors de la dernière intervention ( PIC X )
<b>EIBCALEN</b>	: contient la longueur de la zone de communication passée par le programme appelant ( PIC S9(4) COMP )
<b>EIBRCODE</b>	: contient le code retour CICS après chaque exécution d'une commande ( PIC X(6) )
<b>EIBCPOSN</b>	: contient la position du curseur à l'écran lors de la dernière saisie ( PIC S9(4) COMP )
<b>EIBDATE</b>	: contient la date à laquelle la tâche a été activée ( PIC S9(7) COMP 3 )
<b>EIBTIME</b>	: contient l'heure à laquelle la tâche a été activée ( PIC S9(7) COMP 3 )
<b>EIBDS</b>	: contient le nom logique du dernier fichier manipulé par le programme ( PIC X(8) )
<b>EIBTRMID</b>	: contient le nom logique du terminal associé à la tâche ( PIC X(4) )
<b>EIBTRNID</b>	: contient le nom du code transaction associé à la tâche ( PIC X(4) )
<b>EIBRESP</b>	: contient le code retour CICS après chaque exécution d'une commande ( PIC S9(8) COMP ) et permet une programmation structurée.

*Remarque : Tous les champs du bloc EIB ont par défaut la valeur zéro*

# CARACTERISTIQUES D'UN PROGRAMME CICS

---

Un programme CICS Command Level est

- écrit en assembleur, COBOL, PL1 ou C, C++
- réentrant CICS.

Pour cela CICS

- Duplique dynamiquement la DATA DIVISION en autant d'exemplaires qu'il y a de tâches afin d'en assurer la réentrance.
- Partage le code du programme entre les tâches.



## SYNTAXE DES COMMANDES CICS

---

Le format des commandes CICS, en COBOL, est le suivant :

```
{EXECUTE/EXEC}  CICS
      fonction
      [option]
      [option [(argument)]
      [ ...] ...  ]
END-EXEC.
```

- La "fonction" indique l'opération adressée à l'un des modules de CICS (exemple SEND).
- Les options permettent de préciser la demande.
- Certaines options peuvent avoir des paramètres.

# EXEMPLE

---

## *WORKING-STORAGE SECTION.*

```
01  ZONE-DE-TRAVAIL.  
    05 FICHER      PIC X(08) VALUE 'STAGIAIR'.  
    05 LONG        PIC S9(04) COMP VALUE 200.  
    05 NUM-STAG    PIC X(04) VALUE '1200'.  
    05 LONG-CLE    PIC 9(04) COMP VALUE 5.  
01  ENREG-STAG     PIC X(200).
```

## *PROCEDURE DIVISION.*

```
    EXEC CICS READ  
        FILE (FICHER) OU FILE ('STAGIAIR')  
        INTO (ENREG STAG)  
        LENGTH (LONG)  
        RIDFLD (NUM STAG)  
        KEYLENGTH (LONG CLE) OU KEYLENGTH (4)  
    END-EXEC
```

*Type des arguments utilisés :*

*FILE* : name  
*INTO* : data area  
*LENGTH* : data area  
*RIDFLD* : data area  
*KEYLENGTH* : data value

# LES COMMANDES HANDLE

---

Permettent de gérer

- Les conditions exceptionnelles
- Les touches de fonction
- Les ABENDs

Elles ne sont actives que pour le programme dans lequel elles ont été codés à L'exception de HANDLE ABEND

## LES CONDITIONS EXCEPTIONNELLES

---

Une condition exceptionnelle est une situation particulière susceptible d'apparaître pendant l'exécution d'une commande CICS.

Exemple : Ecriture d'un enregistrement existant

# LES CONDITIONS EXCEPTIONNELLES

---

La commande **HANDLE CONDITION** permet de prévoir un traitement adapté aux exceptions pouvant survenir lors de l'exécution d'un programme.

Syntaxe :

```
EXEC CICS HANDLE CONDITION
    condition1 (label1)
    condition2 (label2)
    condition3
    condition4 (label4)
    :
    :
    ERROR (labelx)
END-EXEC.
```

- au maximum 12 conditions dans une même **HANDLE**
- label étiquette de débranchement(**GO TO**)
- condition mot clé identifiant une condition exceptionnelle
- **ERROR** n'importe quel autre cas d'anomalie que ceux explicitement spécifiés dans l'instruction

## LES CONDITIONS EXCEPTIONNELLES

---

Remarque : *L'option NOHANDLE au sein d'une commande entraîne toujours la continuation en séquence du programme, ce qui veut dire qu'il faudra la gérer*

Une condition de la commande HANDLE prend le pas sur la précédente et n'est valide que dans le programme où elle se trouve.

- Si survient une condition exceptionnelle dont l'interception n'est pas prévue par un HANDLE CONDITION, on aura exécution de l'action par défaut de CICS, un ABEND le plus souvent.

## L'OPTION RESP

---

Afin de palier à l'inconvénient de la HANDLE CONDITION(débranchement par GO TO) IBM a mis en place l'option RESP qui devra faire partie de la commande CICS.

Lorsqu'elle est utilisée, elle a priorité sur la HANDLE CONDITION.

La gestion des erreurs devra être géré en séquence dans le programme.

### EXEMPLE :

```
WORKING-STORAGE SECTION.  
77 C-R          PIC S9(8) COMP.  
01 ENGT         PIC X(200).
```

```
PROCEDURE DIVISION.
```

```
.....  
    EXEC CICS READ FILE('STAGIAIR') INTO (ENGT)  
                .....  
                RESP (C-R)  
    END-EXEC.  
  
    IF C-R = DFHRESP(NOTFND) PERFORM ENREG-NON-TROUVE  
    ELSE  
        IF C-R = DFHRESP(LENGERR) PERFORM ERREUR-LONG  
        ELSE  
            IF C-R NOT = DFHRESP(NORMAL) PERFORM AUTRE-ERR  
            END-IF  
        END-IF  
    END-IF  
    EVALUATE C-R  
        WHEN DFHRESP(NOTFND) PERFORM ENREG-NON-TROUVE  
        WHEN DFHRESP(LENGERR) PERFORM ERREUR-LONG  
    -----
```

END-EVALUATE



# RESTRICTIONS COBOL

---

## Un programme CICS COBOL

- a une ENVIRONMENT DIVISION vide,
- a une DATA DIVISION sans FILE SECTION.

Les instructions d'entrée sortie (OPEN, CLOSE, READ, WRITE) sont interdites.

Les traitements suivants sont impossibles:

- SEGMENTATION,
- SORT,
- TRACE,
- REPORT WRITER

*\*Il est possible de faire des Display dans un programme CICS. Ils pourront être visualisés dans le job CICS (dans CEEMSG)*

## RESTRICTIONS COBOL

---

Les clauses ou les instructions suivantes sont interdites :

- EXHIBIT,
- OCCURS DEPENDING ON en WSS  
mais est possible en LINKAGE SECTION,
- SIGN IS SEPARATE,

## RESTRICTIONS COBOL

---

Les options suivantes du compilateur COBOL/VS ne peuvent être utilisées :

- COUNT,
- DYNAM,
- ENDJOB,
- FLOW,
- STATE,
- SYMDUMP,
- SYST,
- TEST

## EMISSION DE DONNÉES

---

```
EXEC CICS SEND  
      FROM (data area)  
      LENGTH (data value)  
      [ERASE]  
END-EXEC
```

Commande qui permet d'envoyer des données à un terminal

### FROM

Data area : variable définie en working storage section contenant le message à envoyer

### LENGTH :

Data value : zone contenant la longueur du message à transmettre

### ERASE :

Option permettant d'effacer l'écran avant affichage

## RECEPTION DE DONNÉES

---

```
EXEC CICS RECEIVE  
      INTO (data area)  
      LENGTH (data value)  
END-EXEC
```

Commande qui permet de lire des données à partir d'un terminal

INTO :

Data area : variable définie en working storage.

LENGTH

Data value : zone contenant la longueur du message à transmettre

# EXAMPLE

---

## IDENTIFICATION DIVISION.

.  
.  
.

## WORKING-STORAGE SECTION.

77 LONG                PIC S9(4) COMP VALUE 70.  
01 MESSAGE            PIC X(70).

.  
.  
.

## PROCEDURE DIVISION.

.  
.

MOVE 'FIN DE TRANSACTION' TO MESSAGE  
EXEC CICS SEND            FROM (MESSAGE)  
                          LENGTH (LONG)  
                          ERASE

END-EXEC

.  
.

## SEND MAP

---

```
EXEC CICS
  SEND MAP (name)
    [FROM (data area)]
    [LENGTH (data value)]
    [DATAONLY  MAPONLY]
    [MAPSET (name)]
    [CURSOR [(data value)]
    [ALARM] [ERASE]
    [ERASEAUP]
    [FREEKB] [FRSET]
  END-EXEC
```

*Cette commande permet l'affichage d'une map*

## SEND MAP

---

- MAP : Nom de la MAP
- FROM : Nom de la zone contenant les données à envoyer
- LENGTH : Longueur de cette zone (utile si on ne veut pas transmettre tous les champs de la map)
- DATAONLY : Seules les données variables sont envoyées
- MAPONLY : Affichage du cadre constant, sans les données variables (par exemple pour un premier affichage du fond d'écran)
- MAPSET : Nom du mapset

*Si nom de la zone émettrice = nom de map suffixé par O  
inutile de préciser FROM ( ...*

*Si nom du mapset = nom de la map  
inutile de préciser MAPSET.*



## SEND MAP

---

*Options permettant de remplacer les paramètres équivalents spécifiés par les macros BMS :*

- **CURSOR** : Indication de la position du curseur après l'affichage de la map.

Si aucune valeur n'est indiquée, alors le curseur sera placé sur le champ ayant sa longueur égale à 1 .

- **ALARM** :Signal sonore
- **ERASE** : effacement de l'écran et positionnement du curseur
- **ERASEAUP** : effacement du contenu des champs non protégés
- **FREEKB** : Libération du clavier
- **FRSET**: Restauration des bits MDT à OFF

## RECEIVE MAP

---

Pour la réception de données formatées par BMS :

```
EXEC CICS
  RECEIVE MAP (name)
  [SET(pointer ref) INTO(data area)]
  [MAPSET (name)]
  [ASIS]
END-EXEC
```

Cette commande permet de récupérer des données en provenance d'un terminal dans une map symbolique.

## RECEIVE MAP

---

- MAP nom de la map dans le mapset
- SET adresse des données à recevoir (mode LOCATE)
- INTO zone où doit être transféré le contenu de la map symbolique (mode MOVE)
- MAPSET nom du mapset dans la PPT

*Si nom de la zone réceptrice = nom de map suffixé par I  
inutile de préciser INTO ( ...*

*Si nom du mapset = nom de la map  
inutile de préciser MAPSET.*

## COMMANDE HANDLE AID

---

```
EXEC CICS
  HANDLE AID
    [touche1 [(Paragraphe1)]
    [touche2 [(Paragraphe2)]
    .....
    [touche12[(Paragraphe12)]]
  END-EXEC
```

Permet un débranchement(GO TO) au label selon la touche saisie par l'utilisateur (à coder avant un RECEIVE)

- PA1, PA2, PA3
- PF1 à PF24
- CLEAR, ENTER
- ANYKEY : toutes les touches non spécifiées sauf ENTER et CLEAR

## COMMANDE HANDLE AID

---

### Exemple

```
.  
PROCEDURE DIVISION.  
  
  .  
  EXEC CICS HANDLE AID  
    PF1 (PAR1)  
    PF2 (PAR2)  
    PF3 (FIN)  
    ANYKEY (INVALIDE)  
  END EXEC.  
  
  EXEC CICS RECEIVE ... INTO ...  
  END EXEC.  
  
  .  
  .  
  
  PAR1.  
  .  
  
  PAR2.  
  .  
  
  INVALIDE.  
  
  .  
  FIN.
```

### Condition exceptionnelle MAPFAIL

*Se réalise lorsque aucune donnée n'a été récupérée  
lors d'un RECEIVE*

# **LA GESTION DES PROGRAMMES**

## **(LE P.C.P)**

## OBJECTIFS DU CHAPITRE

---

- Rappeler les fonctions du PCP
- Décrire la notion de niveau logique en CICS
- Comprendre la logique d'enchaînement de programme
- Comprendre ce qu'est le PSEUDO CONVERSATIONNEL
- Étudier la syntaxe des commandes adéquates
  - XCTL
  - LINK
  - RETURN
- Présenter les 3 modes de travail CICS
- Donner des éléments de réflexion concernant le développement d'applications CICS

# LES FONCTIONS DU KCP ET PCP

---

## Rappels

- Lors de la demande d'exécution d'une transaction, le KCP (gestionnaire des tâches) *vérifie la validité du code transaction* crée une tâche
- Le contrôle est ensuite donné au PCP *pour qu'il charge le programme* à l'aide des informations contenues dans la *PPT* dont chaque poste a le format suivant :

Nom du load module					
--------------------------	--	--	--	--	--



## LES NIVEAUX LOGIQUES

---

Il existe des commandes CICS permettant de gérer l'enchaînement des programmes ainsi que la gestion de niveaux logiques.

- Appel d'un programme avec intention de retour au programme appelant

<b>Passage au niveau inférieur (LINK)</b>
---

- Appel d'un programme sans retour prévu au programme appelant

<b>Maintien du niveau (XCTL)</b>
----------------------------------

- Retour au programme appelant

<b>Passage au niveau supérieur (RETURN)</b>
---

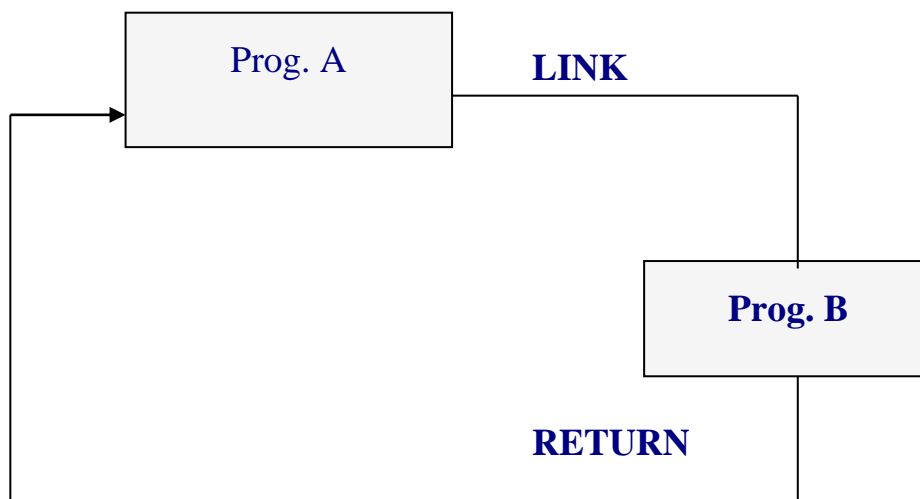
*CICS se situe au niveau logique le plus élevé.*

## COMMANDE LINK

---

EXEC CICS	<b>LINK</b> PROGRAM	(name)	END-EXEC
-----------	---------------------	--------	----------

- Appel à un programme de niveau logique inférieur
- Le contrôle est passé au programme spécifié entre parenthèses.
- Le programme appelant suspend son exécution.
- Le contrôle est rendu à l'appelant à la rencontre de la commande RETURN dans le programme appelé.

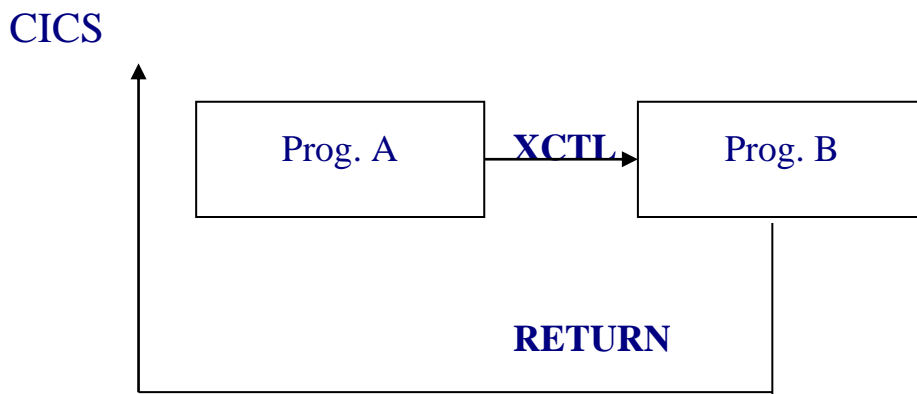


## COMMANDE XCTL

---

```
EXEC CICS XCTL PROGRAM (name) END-EXEC
```

- Transfert à un autre programme de même niveau
- Le contrôle est passé au programme spécifié entre parenthèses.
- Le programme appelant est terminé mais la tâche est toujours active : les ressources restent allouées.



- LINK cède provisoirement le contrôle.
- XCTL cède définitivement le contrôle.

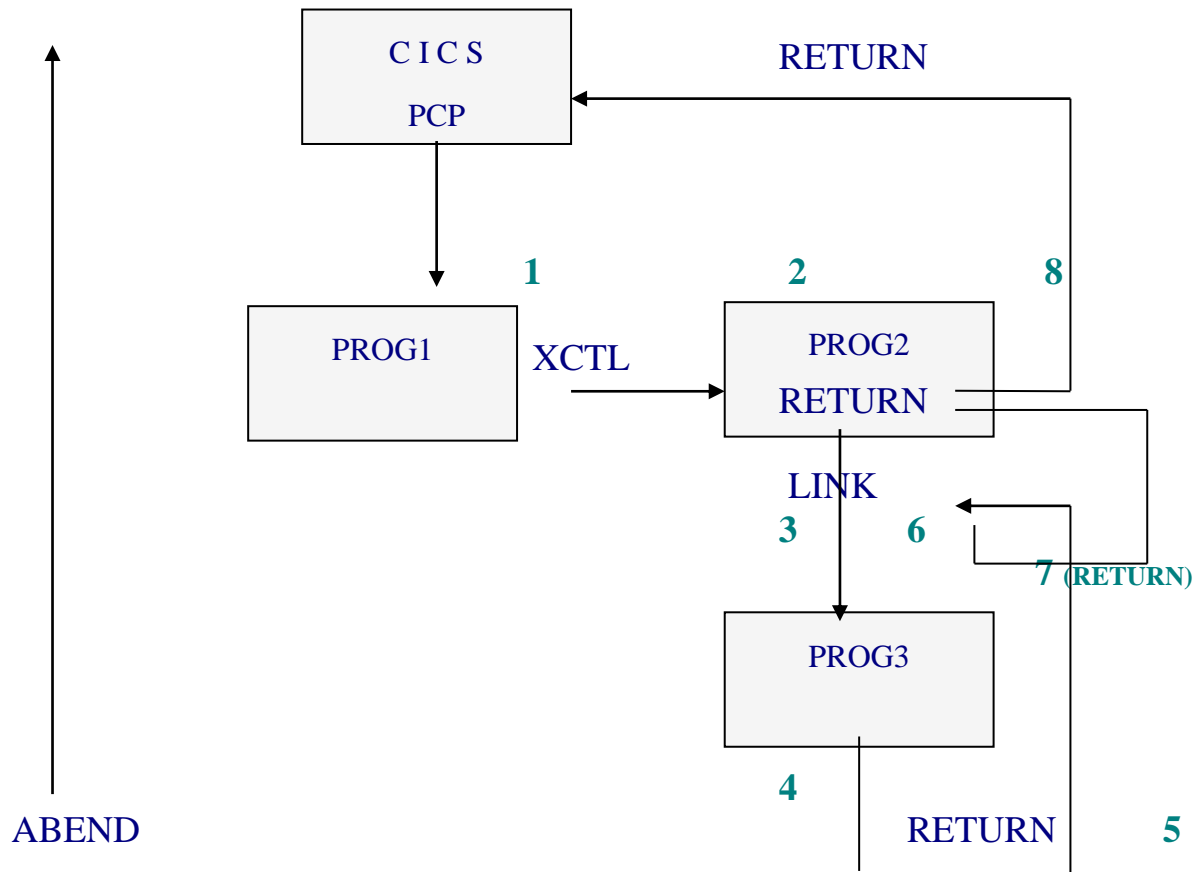
# COMMANDE RETURN

---

EXEC CICS	<b>RETURN</b>	END-EXEC
-----------	---------------	----------

- Termine un programme
- Rend le contrôle
  - soit au programme de niveau logique immédiatement supérieur
  - soit à CICS et la tâche se termine :
    - les ressources sont libérées,
    - un point de synchronisation est établi.

# EXEMPLE



## TRANSFERT DE DONNEES

---

Il est possible de transférer des données d'un programme vers un autre programme (ou vers lui même), ou d'une tâche à la suivante. Cela s'effectue à l'aide de l'option COMMAREA des commandes LINK, RETURN et XCTL.

```
EXEC CICS
  LINK PROGRAM (name)
    [COMMAREA (data area)]
    [LENGTH (data value)]
    [INPUTMSG (data area)
     [INPUTMSGLEN (data value)]]
END-EXEC
```

## TRANSFERT DE DONNEES

---

```
EXEC CICS
  XCTL PROGRAM (name)
    [COMMAREA (data area)]
    [LENGTH (data value)]
      [INPUTMSG (data area)
        [INPUTMSGLEN (data value)]]
END-EXEC
```

- COMMAREA Nom de la zone de travail contenant les informations à transférer.
- LENGTH Longueur de la COMMAREA.
- INPUTMSG Nom de zone à transférer à un programme d'une même tâche émettant un RECEIVE.

## TRANSFERT DE DONNEES

---

- Le programme appelant définit la zone de communication en WSS.
- Le programme appelé récupère le contenu de la zone passée sous le nom (imposé) DFHCOMMAREA placé en Linkage Section.
- La longueur de la COMMAREA récupérée est connue dans le bloc EIB par le paramètre EIBCALEN
- La valeur du paramètre EIBCALEN
  - est égale à zéro si aucune donnée n'est transmise,
  - est différente de zéro si le programme appelant passe des données.



# TRANSFERT DE DONNEES

---

## Exemple :

\*\*\*\*\* PROGRAMME APPELANT \*\*\*\*\*

IDENTIFICATION DIVISION.

PROGRAM ID. PROG1.

.  
.  
.

WORKING-STORAGE SECTION.

01 LG-COMMAREA PIC S9(04) COMP VALUE 4.

01 W-COMMAREA.

05 DONNEE PIC X(04).

.  
.

PROCEDURE DIVISION.

IF .....

MOVE 'TRX1' TO DONNEE

ELSE

MOVE 'TRX2' TO DONNEE

END-IF

EXEC CICS LINK PROGRAM ('PROG2')

COMMAREA (W-COMMAREA)

LENGTH (LG-COMMAREA)

END-EXEC.

.

\*\*\*\*\* PROGRAMME APPELE \*\*\*\*\*

IDENTIFICATION DIVISION.

PROGRAM ID. PROG2.

.  
..

LINKAGE SECTION.

01 DFHCOMMAREA.

05 FONC PIC X(04).

PROCEDURE DIVISION.

IF EIBCALEN = 0 PERFORM FIN END IF

EVALUATE FONC

WHEN 'TRX1' PERFORM TAIT1

WHEN 'TRX2' PERFORM TAIT2

WHEN OTHER PERFORM ERREUR

END-EVALUATE

## COMMANDES LOAD / RELEASE

---

```
EXEC CICS
  LOAD PROGRAM (name)
              SET (ptr ref)
              [LENGTH (data area)]
              [HOLD]
END-EXEC
```

- Charge un programme sans lui passer le contrôle.
- Généralement, il s'agit d'une table.
- PROGRAM identifie la table à charger. Celle ci doit résider dans une bibliothèque de load modules et donc être référencée dans la table ... .
- SET nomme le pointeur contenant l'adresse où CICS a chargé la table.
- LENGTH limite la longueur des données chargées.

## COMMANDES LOAD / RELEASE

---

- HOLD concerne la place mémoire occupée par la table.
- S'il est omis, la place est automatiquement libérée en fin de tâche.

S'il est codé, la table reste dans la région jusqu'à la rencontre de la commande

```
EXEC CICS RELEASE PROGRAM (name) END-EXEC
```

## COMMANDE ABEND

---

EXEC CICS <b>ABEND</b> ABCODE (NAME) END-EXEC
---

- Abandon volontaire de la tâche.
- Provoque un dump si le paramètre ABCODE (name) est spécifié; le nom permet d'identifier le dump.
- Un message à l'écran signale l'ABEND.

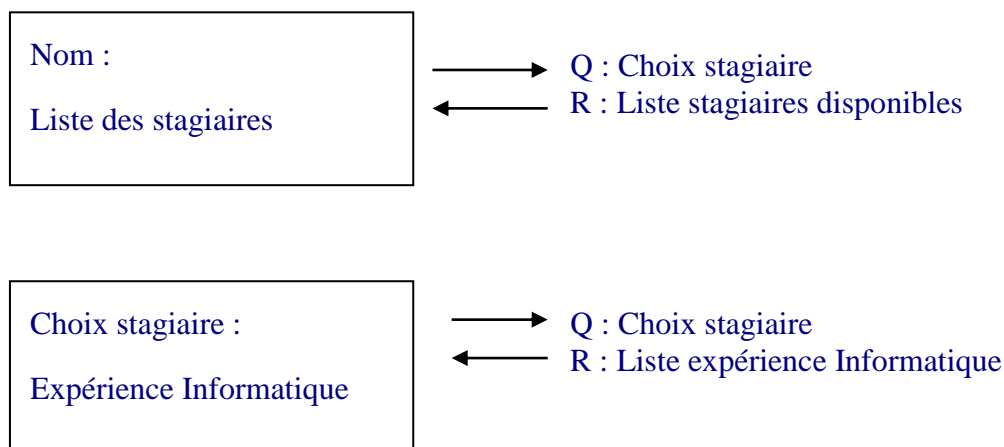
# LE MODE CONVERSATIONNEL

---

Ce mode définit un dialogue de type interactif. Une question est en attente d'une réponse qui sera génératrice d'une autre question et ainsi de suite : il s'agit donc d'une succession de Questions Réponses.

## Exemple

### *Consultation d'un stagiaire*



## LE MODE PSEUDO-CONVERSATIONNEL

---

Problème : il faut limiter le temps de d'allocation des ressources.

Pour cela, il faudrait rendre le contrôle à CICS (avec libération de la tâche), pendant que s'effectue la saisie.

La solution consiste à boucler sur la transaction de la façon suivante :

Afficher une grille de saisie.

Rendre le contrôle à CICS qui conserve en mémoire le nom du terminal, le nom de la transaction suivante et le contexte (commarea).

Ainsi la saisie utilisateur est effectuée alors que la tâche est inactive.

Recevoir les données et les traiter par la transaction prédéfinie.

## LE MODE PSEUDO-CONVERSATIONNEL

---

En pseudo-conversationnel, il existe une tâche par couple Question-Réponse.

Les ressources ne sont pas mobilisées lors de l'attente de la saisie par l'utilisateur.

Exemple

## LE MODE PSEUDO-CONVERSATIONNEL

---

```
EXEC CICS
  RETURN TRANSID (name)
  [COMMAREA (data-area) ]
  [LENGTH (data-value)]
  [IMMEDIATE
  [INPUTMSG (data-area)
  [INPUTMSGLEN (data-value) ]]]
END-EXEC
```

**IMMEDIATE** : La prochaine transaction démarre immédiatement.

Option ne pouvant être mise que dans le programme de niveau supérieur, redonnant la main à CICS.

**INPUTMSG** : N'est valide qu'avec IMMEDIATE.  
Si IMMEDIATE est omis, il sera pris par défaut.



## LES CONDITIONS EXCEPTIONNELLES

---

- INVREQ :

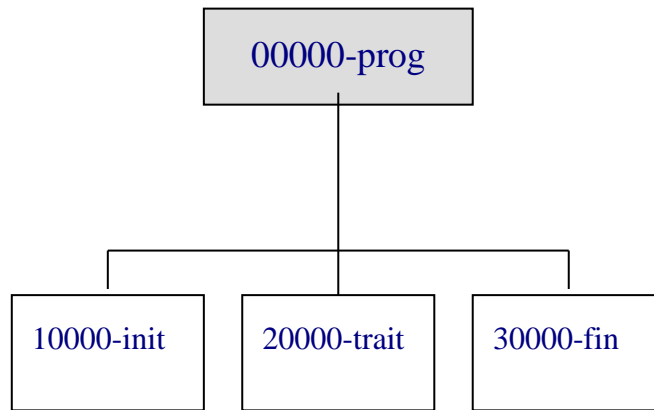
Commande RETURN avec COMMAREA dans un programme qui n'est pas de plus haut niveau logique ou commande RETURN avec TRANSID dans une tâche non associée à un terminal.

- PGMIDERR :

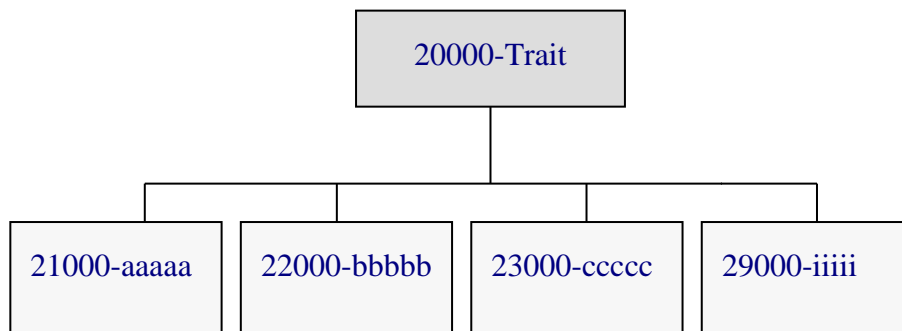
Nom de programme non répertorié dans la PPT ou non présent dans la bibliothèque des load modules.

# VUE GLOBALE DU PROGRAMME

---

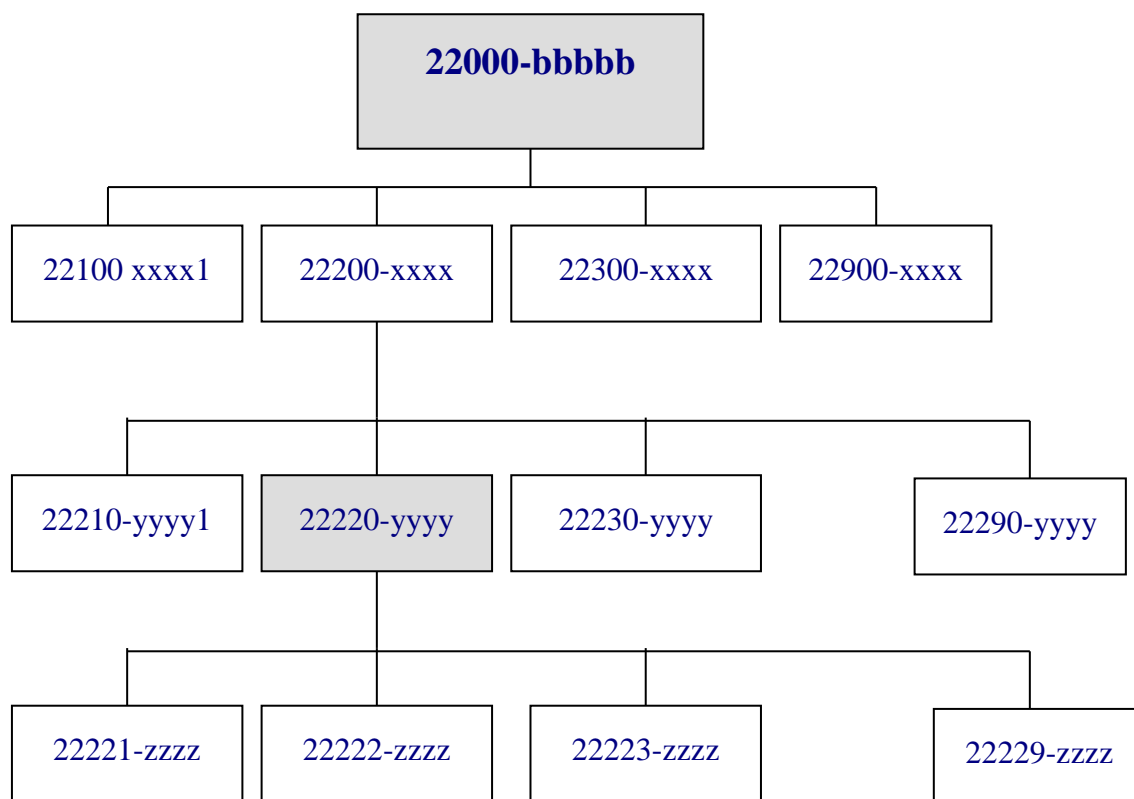


## Vue de la partie 20000 trait



## VUE DETAILLEE DE LA PARTIE 22000-BBBBBB

---



## RÈGLES

---

- Un paragraphe par traitement.

Ce dernier ne pourra faire appel qu'à des paragraphes de niveau inférieur suivant une norme bien définie.

Un paragraphe 20000 ne peut appeler (PERFORM) que les paragraphes préfixés par "2" en incrémentant de 1 le premier "0" de gauche.

Exemple : *Dans le paragraphe 20000 on ne devra faire des "PERFORM" qu'aux paragraphes 21000, 22000 23000 etc ... 29000.*

*Dans le paragraphe 22000 on ne devra faire des "PERFORM" qu'aux paragraphes 22100, 22200 22300 etc ... 22900.*

*Dans le paragraphe 22200 on ne devra faire des "PERFORM" qu'aux paragraphes 22210, 22220 22230 ... 22290 etc.*

## NOMS DE PARAGRAPHES

---

Il est important qu'un numéro de paragraphe soit accompagné d'un nom explicitant le traitement de ce dernier.

### Exemple

- 10000-INIT : Initialisation du programme
- 20000-TRAIT : Plan du traitement global du programme
- 30000-FIN : Les traitements de fin de programme
- 40000-VSAM : Traitement des requêtes VSAM
- 50000-DB2 : Traitement des requêtes DB2
- 60000-TS : Traitement des TS
- 70000-TD : Traitement des TD
- 80000-APPEL : Traitement des LINK, XCTL, CALL ou RETURN
- 90000-MESSAGES : Traitement des messages

Avantage en maintenance : *Lorsqu'un problème survient on sait dans quelle partie du programme aller chercher sans avoir à le dérouler entièrement*

## OPTIONS DIVERSES

---

Les options suivantes doivent être respectées.

- Utiliser l'option RESP plutôt que "l'EXEC CICS HANDLE CONDITION" facilitant ainsi la programmation structurée.
- Utiliser l'option FRSET (MDT OFF) dans la définition des mapsets afin de limiter les transmissions lors d'un RECEIVE.

**Note :** Lors d'un RECEIVE, seuls les champs ayant le bit MDT à 1 sont transmis (attention aux maps de mise à jour).

## OPTIONS DIVERSES

---

Tout programme utilisant une TS auxiliaire (le défaut CICS) devra la détruire. Pour cela le Delete pourra se faire à deux niveaux : en début et fin de programme en tenant compte de la logique applicative.

Il serait souhaitable de prévoir une "HANDLE ABEND" pour le delete des TS en cas de fin anormale de la transaction.

- La passation de zones entre programmes devra se faire via la COMMAREA et non une TS sauf cas particulier justifié.

## STRUCTURE DE LA COMMAREA

---

La technique de programmation utilisée est basée sur une commarea structurée comme suit :

01 COMMAREA- WS.

05 PROG-PRECEDENT	PIC X(08).
05 PROG-COURANT	PIC X(08).
05 PROG-SUIVANT	PIC X(08).
05 FILLER	PIC X(nn).

La zone PROG-PRECEDENT indique le nom du programme appelant.

La zone PROG-COURANT indique le nom du programme en cours d'exécution.

La zone PROG-SUIVANT indique le nom du programme appelé.

Cette technique permet d'avoir un squelette unique quelque soit le type de traitement à effectuer ainsi qu'une trace du déroulement de l'application permettant ainsi de resimuler des abends utilisateurs.



# EXEMPLE DE SQUELETTE

---

```
*=====*
*  SQUELETTE DE PROGRAMME EN PSEUDO CONVERSATIONNEL  *
*              (ALBERT MEGUIRA)                      *
*  M.A.J : Le 15-03-2009                             *
*=====*

*  I D E N T I F I C A T I O N      D I V I S I O N      *
IDENTIFICATION DIVISION.
PROGRAM-ID. SKELCICS.

*  E N V I R O N M E N T      D I V I S I O N      *
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-390.
OBJECT-COMPUTER. IBM-390.

*          D A T A          D I V I S I O N          *
DATA DIVISION.

WORKING-STORAGE SECTION.
77  LG-COM          PIC S9(4) COMP VALUE +NN.
77  XX              PIC X(NN) .
77  YY              PIC X(NN) .

*  DESCRIPTION DETAILLEE DE LA ZONE DE COMMUNICATION  *
01  ZONE-COMMAREA-WS.
    05 PROG-PRECEDENT      PIC X(8) .
    05 PROG-COURANT       PIC X(8) .
    05 PROG-SUIVANT       PIC X(8) .
    05 FILLER              PIC X(nn) .

*  DESCRIPTION      DE      LA      MAP              *
COPY MAPNCG.

*  ZONE DE MESSAGE TAMPON POUR LE SEND FROM          *
01  MESSAGE1          PIC X(79) .

*  DESCRIPTION      DES      TOUCHES      FONCTIONS  *
COPY DFHAID.

*  DESCRIPTION      DES      ATTRIBUTS              *
COPY DFHBMSCA.

*          L I N K A G E      S E C T I O N          *
LINKAGE SECTION.
01  DFHCOMMAREA.
    05 LK-COMMAREA          PIC X(nn1) .   (nn1 = 24 + nn)
```

# EXEMPLE DE SQUELETTE

---

```
*      P R O C E D U R E      D I V I S I O N      *
      PROCEDURE DIVISION.

      00000-INIT-PROGRAMME.
*-----*
      PERFORM 10000-DEBUT-PROGRAMME
      PERFORM 20000-TRAIT-PROGRAMME
      PERFORM 30000-TRAIT-FIN
      .

      10000-DEBUT-PROGRAMME.
*-----*
      EXEC CICS HANDLE CONDITION ERROR      (70000-ERREUR)
      END-EXEC

*      TEST DE PREMIERE ENTREE DANS LE PROGRAMME
      IF EIBCALEN = 0
          MOVE SPACES TO ZONE-COMMAREA-WS
      ELSE
          MOVE LK-COMMAREA TO ZONE-COMMAREA-WS
      END-IF

      MOVE  PROG-COURANT      TO  PROG-PRECEDENT
      MOVE  'INFNCG'          TO  PROG-COURANT
      .

*                               T R A I T E M E N T                               *
      20000-TRAIT-PROGRAMME.
*-----*
      IF PROG-PRECEDENT NOT = PROG-COURANT
          PERFORM 21000-TRAIT-CHARGEMENT
          PERFORM 22000-TRAIT-ENVOI
      ELSE
          PERFORM 23000-TRAIT-RECEPTION
      END-IF
      .

      30000-TRAIT-FIN.
*-----*
      EXIT PROGRAM.

      21000-TRAIT-CHARGEMENT.
*-----*
*      CHARGEMENT DE LA MAP AVANT AFFICHAGE

*      SI TRAITEMENT PARTICULIER AVANT AFFICHAGE
      PERFORM 21100-TRAIT-SPECIFIQUE.

      21100-TRAIT-SPECIFIQUE.
*-----*
*      .....
*      .....
```

# EXEMPLE DE SQUELETTE

---

## 22000-TRAIT-ENVOI.

\*-----\*

```
EXEC CICS SEND MAP      ('MAPN')
              MAPSET ('MAPNCG')
              ERASE

END-EXEC

MOVE PROG-COURANT      TO    PROG-SUIVANT

EXEC CICS RETURN TRANSID ('TNCG')
              COMMAREA (ZONE-COMMAREA-WS)
              LENGTH    (LG-COM)
              RESP(C-R)
```

END-EXEC

```
IF C-R NOT = DFHRESP(NORMAL)
  PERFORM .
```

## 23000-TRAIT-RECEPTION.

\*-----\*

### EVALUATE EIBRID

```
WHEN DFHENTER
  PERFORM 23100-TRAIT-ENTER
WHEN DFHPF12
  PERFORM 70000-TRAIT-FIN
WHEN OTHER
  PERFORM 711000-ERR-TOUCHE
END-EVALUATE.
```

## 23100-TRAIT-ENTER.

\*-----\*

```
*   TRAITEMENT DE VALIDATION DU MOT DE PASSE
*   SI LES CONDITIONS SONT REMPLIES
*   ==> 23110-PROG-SUIVANT
*
```

```
EXEC CICS RECEIVE MAP      ('MAPN')
              MAPSET ('MAPNCG')
              RESP(C-R)
```

END-EXEC

```
EVALUATE C-R
  WHEN DFHRESP(NORMAL)      CONTINUE
  WHEN DFHRESP(MAPFAIL)    PERFORM 71300-PB-RECEIVE
  WHEN OTHER                PERFORM 71400-AUTRE-ERR
END-EVALUATE
```

```
EVALUATE CHOIX
  WHEN '1' MOVE 'PROG1'    TO    PROG-SUIVANT
  WHEN '2' MOVE 'PROG2'    TO    PROG-SUIVANT
  WHEN '3' MOVE 'PROG3'    TO    PROG-SUIVANT
  WHEN OTHER PERFORM 71200-ERREUR-CHOIX
END-EVALUATE
```

PERFORM 23110-PROG-SUIVANT.

# EXEMPLE DE SQUELETTE

---

```
23110-PROG-SUIVANT.  
*-----*  
EXEC CICS XCTL PROGRAM (PROG-SUIVANT)  
          COMMAREA (ZONE-COMMAREA-WS)  
          LENGTH (LG-COM)  
  
END-EXEC.  
  
70000-TRAIT-FIN.  
*-----*  
MOVE 'FIN DE TRANSACTION ....' TO MESSAGEO  
EXEC CICS SEND FROM(MESSAGEO)  
          LENGTH(LONG)  
          ERASE  
  
END-EXEC  
  
EXEC CICS RETURN END-EXEC.  
  
71100-ERR-TOUCHE.  
*-----*  
MOVE 'TOUCHE INVALIDE ....' TO MESSAGEO  
PERFORM 22000-TRAIT-ENVOI.
```

## GESTION DE LA COMMAREA

---

*Afin de garantir la meilleure sécurité des applications, il est possible de copier la COMMAREA reçue (par RETURN ou LINK) dans une zone de WORKING STORAGE SECTION, et ainsi de pouvoir la manipuler en toute tranquillité.*

*On évite ainsi les possibilités d'écrasement inopiné des zones de mémoire n'appartenant pas au programme à cause d'une définition incorrecte ou d'une erreur de programmation.*

*Pour mouvementer la COMMAREA dans la mémoire de son programme, il suffit de la définir dans une zone de WORKING STORAGE SECTION.*

```
WORKING-STORAGE SECTION.  
01 WS-COMMAREA PIC X(3000).  
....  
LINKAGE SECTION.  
01 DFHCOMMAREA PIC X OCCURS 1 TO 4000 DEPENDING ON EIBCALEN.  
  
PROCEDURE DIVISION.  
....  
    IF EIBCALEN 0  
        MOVE SPACES TO WS-COMMAREA  
    ELSE  
        MOVE DFHCOMMAREA TO WS-COMMAREA  
    END-IF.
```

## GESTION DE LA COMMAREA

---

*Dans cet exemple, la zone de WORKING STORAGE SECTION est appelée WS COMMAREA.*

*Si le programme reçoit une COMMAREA, celle ci est copiée de la LINKAGE SECTION vers la WORKING STORAGE SECTION, sinon, cette zone est initialisée à blanc.*

*Dans cet exemple, la COMMAREA peut faire au maximum 4000 octets. Si la COMMAREA passée est plus courte que la zone réceptrice de WORKING STORAGE SECTION, celle ci sera complétée par des blancs.*

*Lors du retour vers CICS, il est possible de spécifier la zone WS COMMAREA comme étant la COMMAREA, plutôt que d'utiliser la COMMAREA présente dans la LINKAGE SECTION.*

*Ainsi, mis à part le MOVE initial, la COMMAREA en LINKAGE SECTION n'est plus jamais référencée, évitant ainsi une Storage Violation malencontreuse. Bien entendu, ceci ne s'applique pas aux sous programmes qui seront obligés de mettre à jour la COMMAREA définie en LINKAGE SECTION avant le RETURN.*

## CHANGEMENT DE TAILLE DE LA COMMAREA

---

*Supposons qu'un programme reçoive une COMMAREA d'une taille donnée et qu'il doive repasser au programme suivant une COMMAREA de même taille, mais qu'il ait besoin, pendant le dialogue écran, de garder des éléments supplémentaires :*

***comment faire ?***

*Au premier appel, la COMMAREA reçue est mouvementée dans une zone de WORKING STORAGE SECTION.*

*Au RETURN, le programme spécifie la longueur qu'il désire et non pas la longueur qu'il a reçue. La fois d'après, lorsque le programme sera redémarré, la COMMAREA aura la longueur souhaitée, plutôt que zéro ou la longueur initiale. Il est également possible d'utiliser cette information pour savoir si le programme a déjà été appelé ou non.*

*L'exemple suivant montre comment programmer cette modification de longueur :*

# CHANGEMENT DE TAILLE DE LA COMMAREA

---

```
WORKING-STORAGE SECTION.  
01 WS-COMMAREA.  
    01 WS-APPELANT PIC X(300).  
    01 WS-MA-ZONE PIC X(050).  
....  
LINKAGE SECTION.  
01 DFHCOMMAREA PIC X OCCURS 1 TO 4000 DEPENDING ON EIBCALEN.  
  
PROCEDURE DIVISION.  
    MOVE DFHCOMMAREA TO WS-COMMAREA  
    ....  
    IF EIBCALEN = 0  
        entrée directe  
    ELSE  
        IF EIBCALEN = 300  
            première fois  
        ELSE  
            IF EIBCALEN = 350  
                fois suivantes  
            ELSE  
                entrée directe  
            END-IF  
        END-IF  
    END-IF  
  
    PERFORM construction écran  
    PERFORM affichage écran  
    EXEC CICS  
        RETURN TRANSID(transid) COMMAREA(WS COMMAREA) LENGTH(210)  
    END-EXEC.
```

*Ce programme attend une COMMAREA de 300 octets la première fois, ou de 350 octets les autres fois. SI la longueur de la COMMAREA est zéro ou une valeur différente de 300 ou 350, un appel à un traitement spécifique est déclenché. Ce traitement peut, par exemple, faire un XCTL au premier programme de l'application, afin de forcer l'utilisateur à passer par là où il le désire.*



## UTILISATION DE L'OPTION RESP

---

*Auparavant, la seule méthode pour gérer les erreurs dans un ordre CICS était l'HANDLE CONDITION et l'HANDLE ABEND.*

*Ces ordres devaient être passés avant l'exécution de la commande. De plus, ces HANDLE génèrent des GO TO, ce qui n'est pas compatible avec les normes de programmation structurée.*

*Un HANDLE peut être activé pour n'importe quel ordre CICS ultérieur, ce qui peut provoquer des erreurs.*

*Désormais, il est possible d'utiliser l'option RESP (et RESP2) dans les ordres CICS. Au lieu de se débrancher vers un paragraphe précédemment défini par l'HANDLE CONDITION, CICS place le code retour dans une zone de WORKING STORAGE SECTION, définie en PIC S9(8) COMP, nommée par l'option RESP.*

*Le test du ce code retour se fera après la commande; ce dernier est réinitialisé après chaque EXEC CICS.*

*L'option RESP a priorité sur HANDLE CONDITION.*

## UTILISATION DE L'OPTION RESP

---

La commande STARTBR ci dessous montre comment coder une option RESP :

```
WORKING-STORAGE SECTION.
```

```
.....
```

```
01 WS-RESP PIC S9(8) COMP.
```

```
EXEC CICS  
  STARTBR  
  FILE('nom de fichier')  
  RIDFLD(ridfld)  
  KEYLENGTH(n)  
  RESP(WS-RESP)  
END-EXEC.
```

```
EXEC CICS  
  STARTBR  
  FILE('nom de fichier')  
  RIDFLD(ridfld)  
  KEYLENGTH(n)  
  RESP(WS-RESP)  
END-EXEC.
```

```
IF WS-RESP = DFHRESP (NOTFND)  
    PERFORM PAS-TROUVE  
ELSE  
  IF WS-RESP NOT = DFHRESP (NORMAL)  
    PERFORM ABEND-ROUTINE  
  END-IF  
END-IF.
```

## UTILISATION DES NOMS CONDITIONS

---

Il est également possible de coder les DFHRESP (nom) comme les valeurs d'un nom condition COBOL (niveau 88), et de les utiliser dans le programme.

Par exemple, WS RESP peut être codé comme suit :

```
WORKING-STORAGE SECTION.  
.... 01 WS-RESP PIC S9(8) COMP.  
      88 RESP-NORMAL      VALUE DFHRESP (NORMAL) .  
      88 RESP-NOTFND      VALUE DFHRESP (NOTFND) .  
      88 RESP-ENDFILE     VALUE DFHRESP (ENDFILE) .
```

Le STARTBR et les tests qui suivent deviennent :

```
EXEC CICS  
  STARTBR  
  FILE('nom de fichier')  
  RIDFLD(ridfld)  
  KEYLENGTH(n)  
  RESP(WS-RESP)  
END-EXEC.  
  
IF RESP-NOTFND  
  PERFORM PAS-TROUVE  
ELSE  
  IF NOT RESP-NORMAL  
    PERFORM ABEND-ROUTINE  
  END-IF  
END-IF.
```

Les noms utilisés le sont à titre d'exemple. Il est possible de coder autre chose, FIN-DE-FICHER au lieu de RESP- ENDFILE. Ce qui est important, c'est que le programme soit le plus clair possible.

Il est également possible d'avoir plusieurs zones réponse dans un programme, mais on le fera uniquement si la logique le requiert.

## UTILISATION DE NOHANDLE

---

NOHANDLE précise à CICS que l'on va gérer les conditions exceptionnelles, ce qui veut dire que, pour l'ordre CICS comportant l'option NOHANDLE, on ignore toutes les conditions exceptionnelles, et que le programme continue en séquence.

## RESP2

---

Certains ordres CICS peuvent, en plus du champ RESP, renseigner également un champ RESP2, permettant d'avoir une vision plus fine de l'erreur qui s'est produite.

*Par exemple, pour une lecture de fichier, il est possible que CICS réponde par LENGERR (erreur de longueur). RESP2 permet de savoir de quel type d'erreur de longueur il s'agit :*

- RESP2 = 10 : LENGTH ou SET n'ont pas été spécifiés pour un fichier variable,
- RESP2 = 11 : la longueur du fichier est plus grande que la valeur stockée dans LENGTH, CICS modifie cette valeur et exécute quand même l'ordre,
- RESP2 = 13 : une longueur incorrecte est spécifiée pour un enregistrement de longueur fixe.

# **LA GESTION DU TEMPS (I.C.P.)**

## ASKTIME

---

Permet, d'avoir l'heure du jour.

```
EXEC CICS ASKTIME  
      [ABSTIME (data-area) ]  
END-EXEC
```

**ABSTIME** : permet d'obtenir dans une zone de 8 octets binaires l'intervalle de temps écoulé en *millisecondes* depuis le *1<sup>er</sup> janvier 1900*.

# FORMATTIME

---

Permet de formater la date et l'heure

## EXEC CICS **FORMATTIME**

[**ABSTIME** (data-area)]  
[YYDDD (data-area)]  
[YYMMDD (data-area)]  
[YYDDMM (data-area)]  
[**DDMMYY** (data-area)]  
[MMDDYY (data-area)]  
[YYYYDDD (data-area)]  
[YYYYMMDD (data-area)]  
**V4** [YYYYDDMM (data-area)]  
[DDMMYYYYY (data-area)]  
[MMDDYYYYY (data-area)]  
[DATE (data-area)]  
[DATEFORM (data-area)]  
[**DATESEP** (data-area)]  
[DAYCOUNT (data-area)]  
[DAYOFWEEK (data-area)]  
[DAYOFMONTH (data-area)]  
[MONTHOFYEAR (data-area)]  
[YEAR (data-area)]  
[**TIME** (data-area)]  
[**TIMESEP** (data-area)]



END-EXEC

## FORMATTIME

---

DATE : Zone de 8 octets contenant la date reformatée.

DATEFORM : Zone de 6 octets contenant le format de la date.

*(La valeur retournée peut être YYMMDD, DDMMYY, MMDDYY).*

DATESEP : Désigne le caractère utilisé comme séparateur.

DAYCOUNT : Zone de 4 octets contenant le nombre de jours depuis le 01/01/1900.

DAYOFWEEK : Zone de 4 octets contenant le numéro du jour de la semaine (Dimanche = 0, Samedi = 6).

DAYOFMONTH : Zone de 4 octets contenant le numéro du jour dans le mois.

MONTHOFYEAR : Zone de 4 octets contenant le numéro du mois dans l'année.

## FORMATTIME

---

TIME : Zone de 8 octets contenant l'heure.

La présence des séparateurs dépend de l'option TIMESEP.

YEAR : Zone de 4 octets contenant l'année AAAA.

# START

---

Permet à un instant donné, d'initialiser une tâche reliée ou non à un terminal.

```
EXEC CICS START
```

```
    [INTERVAL (hhmmss) | TIME (hhmmss) ]
```

```
    TRANSID (name)
```

```
    [FROM (data-area)
```

```
    LENGTH (data-value) ]
```

```
    [TERMID (name) ]
```

```
END-EXEC
```

**INTERVAL** : Initialisation de la tâche dans un intervalle de temps spécifié.

**TIME** : Initialisation de la tâche à une heure donnée.

**TRANSID** : Code transaction à initialiser.

**FROM** : Permet de passer des information à la tâche.

## START

---

LENGTH : Demi-mot binaire indiquant la longueur de la donnée transmise.

TERMID : Nom du terminal associé à la tâche à démarrer.

# RETRIEVE

---

Permet la lecture des données éventuellement transmises à une tâche.

```
EXEC CICS RETRIEVE
```

```
    INTO (data-area)
```

```
    LENGTH (data-area)
```

```
END-EXEC
```

**INTO** : Zone de travail où sont placées les données transmises.

**LENGTH** : Demi-mot binaire indiquant la longueur de la zone récupérée.

## CONDITIONS EXCEPTIONNELLES

---

ENDDATA : Plus de zone a récupérer par RETRIEVE

ENVDEFERR : Option incohérente entre le START et le RETRIEVE.

TERMIDERR : Terminal spécifié par un START inconnu.

TRANSIDERR : Code transaction inconnu dans la PCT.

IOERR : Erreur d'I/O pendant un RETRIEVE.

# **LA GESTION DES FICHIERS**

## **(LE F.C.P)**



## OBJECTIFS DU CHAPITRE

---

- Rappeler des rôles du FCP et de la FCT
- Présenter les différentes commandes d'entrée/sortie CICS.
- Deux parties seront étudiées dans ce chapitre :

1. lecture et mise à jour de fichier VSAM

2. balayage de fichier VSAM

# LA GESTION DES FICHIERS

---

## Rappels :

- Le FCP (File Control Program) assure l'interface entre les programmes d'application CICS et la méthode d'accès VSAM.
- Les fichiers sont définis (DEFine CLuster) via IDCAMS
- Et sont déclarés dans la FCT afin de pouvoir être utilisés sous CICS
- Le nom logique du fichier (défini dans la FCT) est associé à un nom physique (data set name)
- L'ouverture des fichiers est faite de manière général par CICS dynamiquement au premier accès au fichier

## LES COMMANDES D'E/S

---

- *READ*
- *WRITE*
- *READ for UPDATE*
- *REWRITE*
- *DELETE*
- *UNLOCK*

## LE READ

---

```
EXEC CICS READ FILE (name)
      RIDFLD (data area)
      {SET(pointer ref) INTO(data area)}
      [LENGTH (data area)]
      [KEYLENGTH (data value)]
      [GENERIC]
      [RBA RRN]
      [GTEQ EQUAL](VSAM seul)
      [TOKEN (valeur)]
      [UPDATE]
END-EXEC
```

Lecture en accès direct d'un fichier

*Par défaut, il s'agit d'un fichier VSAM KSDS*

- **FILE**

Nom logique du fichier correspondant à une entrée de la FCT

- **RIDFLD (data area)**

Data-area : contient la clé de l'enregistrement à lire pour un KSDS .

Doit être un mot de 4 octets lorsqu'il s'agit d'un RBA ou RRN

## LE READ

---

- SET

Pointeur permettant d'adresser la zone d'E/S de l'enregistrement défini en Linkage Section

- INTO

Zone d'E/S de l'enregistrement définie en W.S.S

- LENGTH

Demi mot binaire contenant la longueur de la zone d'E/S référencée par INTO

- KEYLENGTH

Demi mot binaire contenant la longueur de la clé complète ou générique Obligatoire si GENERIC

- GENERIC

Indique que l'on accède au fichier à l'aide d'une partie de la clé (spécifiée via l'option KEYLENGTH)

## LE READ

---

- RBA (Relative Byte Address)

Déplacement relatif par rapport au début du fichier pour un ESDS ou numéro relatif d'enregistrement pour un RRDS

- RRN

La zone RIDFLD contient un RRN, en RRDS.

- GTEQ

Lecture du 1er enregistrement dont *clé*  $\geq$  *valeur de RIDFLD*

- EQUAL

- Lecture de l'enregistrement dont *clé* = *valeur de RIDFLD*

# EXAMPLE

---

```
.  
.   
WORKING STORAGE SECTION.  
01 LONG ENGT PIC BINARY VALUE +400.  
01 W ENGT.  
05 CLE ENR PIC 9(05).  
.   
.   
.   
PROCEDURE DIVISION.  
.   
.   
.   
MOVE 01200 TO CLE ENR.  
EXEC CICS READ FILE ('STAGIAIR')  
    RIDFLD (CLE ENR)  
    INTO (W ENGT)  
    LENGTH (LONG ENGT)  
END-EXEC
```

## LE WRITE

---

```
EXEC CICS WRITE FILE (name)
           RIDFLD (data area)
           FROM (data area)
           [LENGTH (data value)]
           [KEYLENGTH (data value)]
           [RBA RRN]
           [MASSINSERT]
END-EXEC
```

### Création d'un enregistrement

- **RIDFLD** (data-area)

Data-area : contient la clé de l'enregistrement à créer pour un KSDS

Doit être un mot de 4 octets lorsqu'il s'agit d'un RBA ou RRN

- **MASSINSERT**

Permet d'insérer plusieurs enregistrements en séquence ascendante

Cette boucle d'écriture est comparable à un chargement initial.

Pendant le **MASSINSERT** le fichier est bloqué; il faudra donc le libérer à l'aide de la commande **UNLOCK**.



## LE READ FOR UPDATE

---

```
EXEC CICS READ FILE (name)
          RIDFLD (data area)
          {INTO(data area) SET(pointer ref)}
          UPDATE
          [TOKEN (valeur)]
END-EXEC
```

- **UPDATE**

**Est obligatoire avant toute mise à jour**

- **TOKEN**

Valeur unique (4 octets binaire) liée à une tâche identifiant le READ for UPDATE. Est lié au REWRITE, DELETE ou UNLOCK.

Possibilité de programmer plusieurs READ UPDATE.

## LE REWRITE

---

```
EXEC CICS REWRITE FILE (name)
              FROM (data area)
              [LENGTH (data value)]
              [TOKEN (valeur)]
(CICS V4)
END-EXEC
```

### Modification d'un enregistrement

- Doit être précédé d'un READ UPDATE
- FILE : Nom du fichier
- Data-area : Nom de W.S.S qui contient l'enregistrement préalablement modifié
- LENGTH :Longueur de l'enregistrement

*Ne jamais modifier le champ clé.*

## EXAMPLE

---

```
.  
WORKING-STORAGE SECTION.  
01 LONG-ENGT PIC S9(04) COMP VALUE 400.  
01 W-ENGT.  
05 CLE-ENR PIC 9(05).  
. . .  
PROCEDURE DIVISION.  
. . .  
    MOVE 01200 TO CLE-ENR.  
    EXEC CICS READ FILE ('STAGIAIR')  
        RIDFLD (CLE-ENR)  
        INTO (W-ENGT)  
        LENGTH (LONG-ENGT)  
        UPDATE  
    END-EXEC  
. . .  
    MODIFICATIONS (SAUF CLE)  
. . .  
    EXEC CICS REWRITE FILE ('STAGIAIR')  
        FROM (W-ENGT)  
        LENGTH (LONG-ENGT)  
    END-EXEC
```

# LE DELETE

---

*Seulement pour KSDS et RRDS*

- Deux cas de figure pour la suppression :
  1. L'enregistrement a été lu avec l'option UPDATE.

```
EXEC CICS DELETE FILE (name) END-EXEC  
(CICS V4) [TOKEN (valeur)]
```

2. L'enregistrement n'a pas été lu.

```
EXEC CICS DELETE FILE (name)  
      [RIDFLD (data area)]  
      [KEYLENGTH (data value)]  
      [GENERIC [NUMREC (data area)]]  
      [RBA RRN]  
END-EXEC
```

## LE DELETE

---

- GENERIC

Générique indiquée Tous les enregistrements ayant la clé sont supprimés.

- NUMREC


Indique le nombre d'enregistrements supprimés.

## EXEMPLES

---

### L'enregistrement est lu préalablement

```
.  
WORKING-STORAGE SECTION.  
01 LONG-ENGT PIC S9(04) COMP VALUE 400.  
01 W-ENGT.  
05 CLE-ENR PIC 9(05).  
..  
PROCEDURE DIVISION.  
..  
MOVE 01200 TO CLE ENR.  
EXEC CICS READ FILE ('STAGIAIR')  
      RIDFLD (CLE-ENR)  
      INTO (W ENGT)  
      LENGTH (LONG ENGT)  
      UPDATE  
END EXEC  
.  
EXEC CICS DELETE FILE ('STAGIAIR') RIDFLD  
END EXEC
```



### L'enregistrement n'est pas lu préalablement

```
.  
WORKING STORAGE SECTION.  
01 LONG ENGT PIC S9(04) COMP VALUE +400.  
01 W ENGT.  
05 CLE-ENR PIC 9(05).  
.  
PROCEDURE DIVISION.  
..  
MOVE 01200 TO CLE-ENR.  
EXEC CICS DELETE FILE ('STAGIAIR')  
      RIDFLD (CLE-ENR)  
END-EXEC
```

# UNLOCK

---

```
EXEC CICS UNLOCK FILE (name) END EXEC  
(CICS V4) [TOKEN (valeur)]
```

- Libère le contrôle exclusif d'un ou de plusieurs enregistrements lus avec l'option UPDATE
- Utilisé lorsque le programme renonce à sa mise à jour ou à sa suppression
- Un READ UPDATE doit toujours être suivi d'un DELETE, REWRITE ou UNLOCK si l'on veut coder un WRITE ou un autre READ UPDATE sur le même fichier.
- Une fin de tâche ou unabend provoque le déverrouillage du fichier

## LE BROWSING

---

Browsing = balayage d'un fichier en consultation

Le balayage se décompose en 4 étapes:

1. STARTBR : Positionnement initial sur un enregistrement du fichier
2. READNEXT : Lecture avant
3. READPREV : Lecture arrière
4. RESETBR : Repositionnement du balayage
5. ENDBR : Arrêt du balayage

*Pendant le balayage les mises à jour sont ne sont pas possibles directement.*

Pour éviter tout problème avec les fichiers, il faudra arrêter le balayage, mettre à jour l'enregistrement, puis redémarrer le balayage.



## LE STARTBR

---

```
EXEC CICS STARTBR FILE (name)
      RIDFLD (data area)
      [KEYLENGTH (data value)
      [GENERIC]]
      [REQID (data value)]
      [RBA RRN]
      [GTEQ EQUAL]
END-EXEC
```

Data-area: contient la clé qui indique le positionnement du balayage

N'effectue pas de lecture.

- REQID : Identifie le browsing, si browsings simultanés (par défaut 0)
- GTEQ : Option interdite dans le cas d'un ESDS

## LE READNEXT

---

```
EXEC CICS READNEXT FILE (name)
          RIDFLD (data area)
          {SET(pointer ref) INTO (data area)}
          [LENGTH (data area)]
          [KEYLENGTH (data value)]
          [REQID (data value)]
          [RBA RRN] (VSAM)
END-EXEC
```

Lecture d'un enregistrement à partir de la position courante dans le fichier (STARTBR ou READNEXT)

Un STARTBR, un READNEXT ou RESETBR doit être effectué préalablement

- **RIDFLD** : Obligatoire. Mis à jour par le FCP qui y place la clé des enregistrements lus successivement. Contient la clé complète (même si STARTBR a été exécuté avec une clé générique)

*Il est possible d'effectuer un saut en avant dans le fichier en modifiant le contenu du RIDFLD*

## EXEMPLE

---

**MOVE '01200' TO CLE-ENR**

**EXEC CICS STARTBR RIDFLD(CLE-ENR)**

(égalité la 1<sup>ère</sup> fois)

**EXEC CICS READNEXT RIDFLD(CLE-ENR)**

.....

**MOVE '03000' TO KEY.**

**EXEC CICS READNEXT RIDFLD(CLE-ENR)**

## LE READPREVIEW

---

```
EXEC CICS READPREV FILE (name)
           {SET(pointer ref) INTO (data area)}
           RIDFLD (data area)
           [LENGTH (data area)]
           [KEYLENGTH (data value)]
           [RBA RRN]
           [REQID (data value)]
END-EXEC
```

Lecture de l'enregistrement précédent à partir de la position courante dans le fichier

- Un STARTBR, un READNEXT ou RESETBR doit être effectué préalablement
- Si une commande READPREV est codée après un READNEXT, le même enregistrement sera restitué;

*Il faut donc doubler les **READNEXT** ou **READPREV** pendant le balayage .*

## LE ENDBR

---

```
EXEC CICS ENDBR FILE (name)
           [REQID (data value)]
END-EXEC
```

- Provoque la fin du balayage
- REQID : Indique le numéro du browsing si plusieurs browsings simultanés

Si cette commande n'est pas explicite, il faudra attendre la fin fin de la tâche afin que le balayage soit fermé implicitement.

## LE RESETBR

---

```
EXEC CICS RESETBR FILE (name)
          RIDFLD (data area)
          [KEYLENGTH (data value)
          [GENERIC]]
          [REQID (data value)]
          [GTEQ EQUAL](VSAM)
          [RBA  RRN]   (VSAM)
END-EXEC
```

Permet de repositionner le balayage à un autre endroit du fichier (RIDFLD indique la nouvelle position) tout en changer le mode de recherche (GTEQ ou EQUAL)

Identique à un ENDBR suivi d'un STARTBR

## LES CONDITIONS EXCEPTIONNELLES

---

- DSIDERR

Fichier inconnu dans la FCT.

- DUPKEY

Lors d'un accès via la clé secondaire, il y a plusieurs clés secondaires répondant à ce critère.

- DUPREC

Enregistrement existant (lors d'un WRITE)

- ENDFILE

Fin de fichier lors d'un balayage (READNEXT ou READPREV)

- ILLOGIC

Si enregistrement non trouvé en cas d'accès avec option RBA

- INVREQ

Demande invalide (commande incorrecte)

## LES CONDITIONS EXCEPTIONNELLES

---

- IOERR

Erreur d'entrée/sortie sur disque

- LENGERR

Erreur de longueur

- NOTFND

Enregistrement non trouvé

- NOSPACE

Plus de place disponible sur le fichier

- NOTOPEN

Le fichier est fermé



# **LES DONNEES TEMPORAIRES**

## **(LE T.S.P)**

## OBJECTIFS DU CHAPITRE

---

- Présenter Le TSP
- Intérêt des données temporaires
- Cas d'utilisation des TS
- Les conditions exceptionnelles

## LES DONNÉES TEMPORAIRES

---

- Elles sont gérées par le TSP ( Temporary Storage Program ).
- Il n'existe pas de table de définition des données temporaires (TS)
- Son allocation et sa libération sont effectués dynamiquement par le programme
- Chaque zone créée est appelé un ITEM
- Un ensemble d'ITEM est un MESSAGE SET
- On peut y stocker des zones de longueur fixe ou variable (Maximum 32K pour une zone)
- Une fin de tâche ne libère pas implicitement la TS
- Une TS peut être exploitée par une tâche autre que celle qui l'a créée
- Une TS est identifiée par un nom symbolique de huit caractères normalisé.
- Les enregistrements de la QUEUE TS peuvent être écrits et lus séquentiellement ou directement (en indiquant le numéro de l'ITEM).

## LES DONNEES TEMPORAIRES

---

- Les données stockées sont réutilisables plusieurs fois
- Elles sont stockées soit:
  - en mémoire principale
  - en mémoire auxiliaire par défaut (fichier VSAM ESDS)

Dès qu'elle n'est plus utile, la QUEUE TS doit être purgée, par programme, afin de libérer la place occupée.

*Une TS ne peut être effacée que dans sa totalité*

## UTILISATION DES DONNÉES TEMPORAIRES

---

- Complément de COMMAREA
- Fichier de travail pour la pagination
- Stock d'enregistrements sélectionnés en vue d'un travail sur une partie de fichier afin de limiter les accès disque
- Source d'impression d'états récapitulatifs par récupération de données issues de divers traitements

## WRITEQ TS

---

```
EXEC CICS WRITEQ TS QUEUE (name)
          FROM (data area)
          [LENGTH (data value)]
          [ITEM (data area)]
          [REWRITE] ← (Option du write)
          [MAIN AUXILIARY]
END-EXEC
```

- **QUEUE**

Indique le nom symbolique du message set ou de la file d'attente.

- **FROM**

Nom symbolique de la zone de travail où se trouve l'item à écrire

- **LENGTH**

Longueur de l'item à écrire sur 2 octets binaire

## WRITEQ TS

---

- ITEM

Demi mot binaire indiquant le numéro de la zone à écrire

- Sans REWRITE, zone où CICS indique le numéro de la donnée écrite
- Avec REWRITE indique le numéro de la donnée qui doit être mise à jour

- MAIN

Stockage en mémoire principale

- AUXILIARY (par défaut)

Stockage en mémoire auxiliaire

## EXAMPLE

---

### WORKING-STORAGE SECTION.

```
01 LONG                PIC S9(04) BINARY.  
01 NUM-ITEM            PIC S9(04) BINARY.  
01 NOM-TS.  
    05 ID-TERM          PIC X(04).  
    05 ID-TRAN         PIC X(04).  
  
01 ZONE-WSS            PIC X(40).  
01 ZENT.  
    05 CODTRAN         PIC X(04).
```

### PROCEDURE DIVISION.

MOVE +40 TO LONG.

MOVE **EIBTRMID** TO ID-TERM.  
MOVE **CODTRAN** TO ID-TRAN.

EXEC CICS **WRITEQ TS**  
 **QUEUE (NOM-TS)**  
 FROM (ZONE-WSS)  
 LENGTH (LONG)  
 ITEM (**NUM-ITEM**)  
END-EXEC.



## READQ TS

---

Permet de lire des données temporaires

```
EXEC CICS READQ TS QUEUE (name)
      {INTO (data area) SET (ptr ref)}
      LENGTH (data area)
      [ITEM (data area)]
      [NEXT]
END-EXEC
```

- INTO

Zone de W.S.S destinée à recevoir les items

- LENGTH

Longueur maximale de l'item à lire

Après lecture, contient sa longueur réelle

- ITEM

Demi mot binaire indiquant le numéro de l'item à lire

- NEXT

Incompatible avec ITEM, permet la lecture séquentielle de la file d'attente

## EXEMPLES

---

- Lecture du premier enregistrement d'un Message Set

```
EXEC CICS READQ TS  
    QUEUE (NOM TS)  
    INTO (ZONE WSS)  
    LENGTH (LONG)  
END-EXEC
```

- Lecture de l'enregistrement de numéro nn.

```
EXEC CICS READQ TS  
    QUEUE (NOM TS)  
    INTO (ZONE WSS)  
    LENGTH (LONG)  
    ITEM (nn)  
END-EXEC
```

## DELETEQ TS

---

```
EXEC CICS DELETEQ TS QUEUE (name) END-EXEC
```

- Supprime la TS dans sa totalité
- Libère la place occupée par la file d'attente spécifiée.
- Il est conseillé de supprimer impérativement les TS inutiles en fin de programme.

## LES CONDITIONS EXCEPTIONNELLES

---

- IOERR

Erreur d'entrée/sortie sur disque

- INVREQ

Longueur des données à écrire nulle ou excédant la taille maximum autorisée

- ITEMERR

Numéro d'ordre spécifié inconnu

- LENGERR

Données lues trop longues

- NOSPACE

Plus de mémoire disponible ou fichier plein

- QIDERR

Nom de TS inconnu en lecture

# **LES DONNÉES TRANSITOIRES (LE T.D.P)**

## OBJECTIFS DU CHAPITRE

---

- Présenter Le TDP
- Intérêt des données transitoires
- Cas d'utilisation des TD
- Les conditions exceptionnelles

# LES DONNÉES TRANSITOIRES

---

- Ce sont des données stockées provisoirement sur des fichiers séquentiels appelés file d'attente (QUEUE TD) ou destination.
- Le TDP (Transient Data Program) permet d'écrire et de lire des données dans une destination prédéfinie.
- Une destination est identifiée par un nom symbolique de 4 caractères, défini dans la table des destinations (DCT).
- On distingue deux types de destination :
  1. Extra partition
  2. Intra partition

## DESTINATION EXTRA PARTITION

---

- Créée par une tâche CICS et traitées hors de CICS ou l'inverse
- Chaque destination extra partition est un fichier séquentiel se trouvant sur un type d'unité séquentiel (disque, bande, imprimante) et est définie dans la DCT comme un fichier d'entrée ou de sortie pour CICS.
- Les données transitoires sont des enregistrements de longueur fixe ou variable, bloqués ou non, dont le format est décrit dans la DCT.



## DESTINATION EXTRA PARTITION

---

- Les destinations extra partition servent, en général, à stocker des informations en vue d'une utilisation ultérieure.
- Exemples :
  - Saisie de données pour un traitement batch
  - Statistiques sur la session CICS
  - Alimentation d'un journal
  - Plusieurs tâches peuvent écrire sur une même file d'attente en sortie.

## DESTINATION INTRA PARTITION

---

- Ces destinations sont des files d'attente provisoirement stockées dans un fichier VSAM ESDS
- Créées par une tâche CICS et traitées par une autre tâche CICS
- Les enregistrements en sont de longueur variable.
- Par défaut, l'espace occupé par la file d'attente sera rendu disponible une fois que tous les enregistrements auront été lus.
- Si l'espace de la file d'attente est déclaré non réutilisable, la file d'attente existera jusqu'à l'exécution d'une commande DELETE explicite.

## DESTINATION INTRA PARTITION

---

- Plusieurs tâches peuvent utiliser la même destination intra-partition. Le TDP traite ces données en employant la technique FIFO.
- Les données ne peuvent être lues qu'une seule fois si la TD est définie réutilisable (option par défaut).
- Les destinations intra partition servent à :
  - Transmettre des messages d'un terminal à un autre
  - Envoyer un message à plusieurs terminaux.
- Les destinations intra-partition permettent de lancer automatiquement une transaction en lui affectant un seuil de déclenchement (Trigger Level).

## DÉCLENCHEMENT AUTOMATIQUE DE TACHE

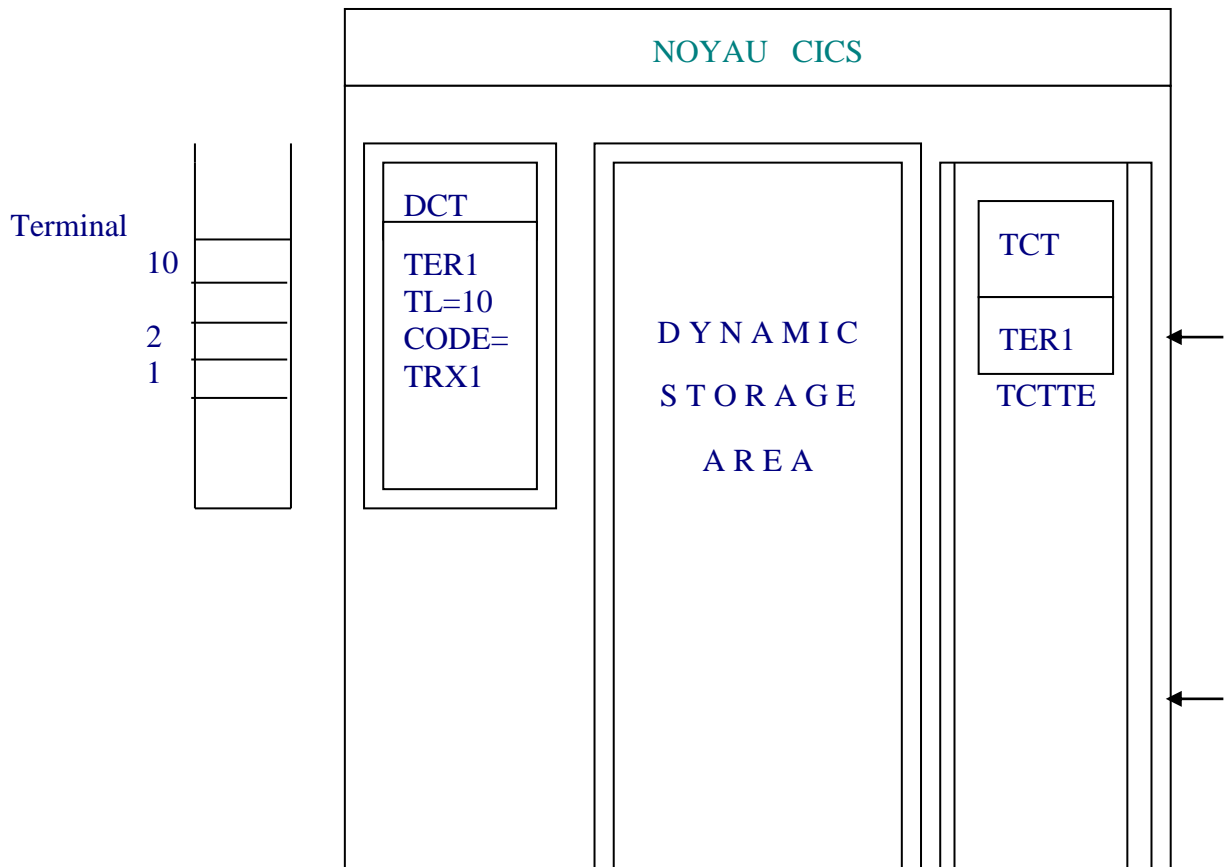
---

- Une transaction qui va lire des données transitoires peut être déclenchée manuellement à partir d'un terminal par CICS lorsque le seuil (trigger level) est atteint
- La tâche créée correspond :
  - A la transaction dont le code est défini dans la DCT, au terminal associé à la destination .
  - Le nom de la destination doit être celui du terminal associé.

## EXEMPLE

Ce schéma représente le lien existant entre les divers éléments de CICS impliqués dans le déclenchement automatique de tâches.

### REGION CICS



TER1 : Nom, dans la TCT, du terminal TER1.  
TER1 : Nom, dans la DCT, d'une destination. Celle ci est matérialisée sur le schéma par un ensemble de 5 enregistrements.  
TL = 10 : Seuil de déclenchement trigger level  
TRX1 : Code transaction.

Dans l'état du système représenté, si le terminal TER1 est sous tension ET non associé à un tâche, une nouvelle tâche associée

à la transaction TRX1 est déclenchée automatiquement.

## WRITEQ TD

---

Ecriture de données transitoires vers une destination.

```
EXEC CICS WRITEQ TD  
          QUEUE ( name )  
          FROM (data area)  
          [LENGTH (data value)]  
END-EXEC
```

- **QUEUE**

Nom de la destination tel qu'elle est définie dans la DCT

- **FROM**

Zone de travail contenant l'enregistrement à stocker

- **LENGTH**

Demi mot binaire indiquant la longueur de la zone à écrire. Peut être omis lors de l'écriture dans une TD extra partition de longueur fixe.

## READQ TD

---

### Lecture de données transitoires

```
EXEC CICS READQ TD  
          QUEUE ( name )  
          INTO (data area) { SET (pointer ref)  
          [ LENGTH (data value)]  
END-EXEC
```

- **INTO**

Zone de réception de l'enregistrement lu

- **LENGTH**

Demi mot binaire indiquant la longueur maximum de l'enregistrement à lire

- La lecture est **destructrice** lors de la lecture d'une TD intra partition définie réutilisable.



## DELETEQ TD

---

```
EXEC CICS DELETEQ TD QUEUE (Name) END-EXEC
```

- Supprime toutes les données transitoires d'une destination intra-partition.
- Doit être utilisé pour rendre disponible l'espace occupé par une TD intra partition spécifiée non réutilisable.

## LES CONDITIONS EXCEPTIONNELLES

---

- QZERO

La destination précisée dans commande  
READQ TD est vide

- LENGERR

Données lues trop longues

- IOERR

Erreur d'entrée/sortie

- NOSPACE

Plus de place dans la destination

- NOTOPEN

Destination fermée

- QBUSY

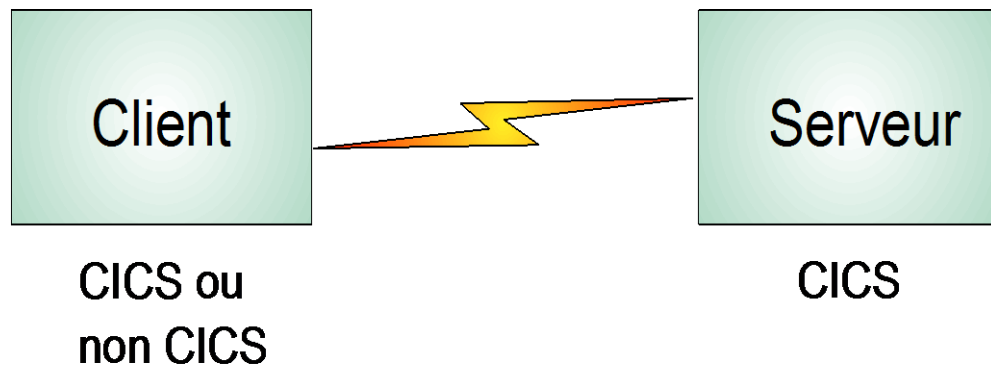
Destination intra partition utilisée par une autre  
tâche au moment de l'exécution d'une  
commande READQ TD

- QIDERR

La Destination n'existe pas

# L'ENVIRONNEMENT CLIENT/SERVEUR

---

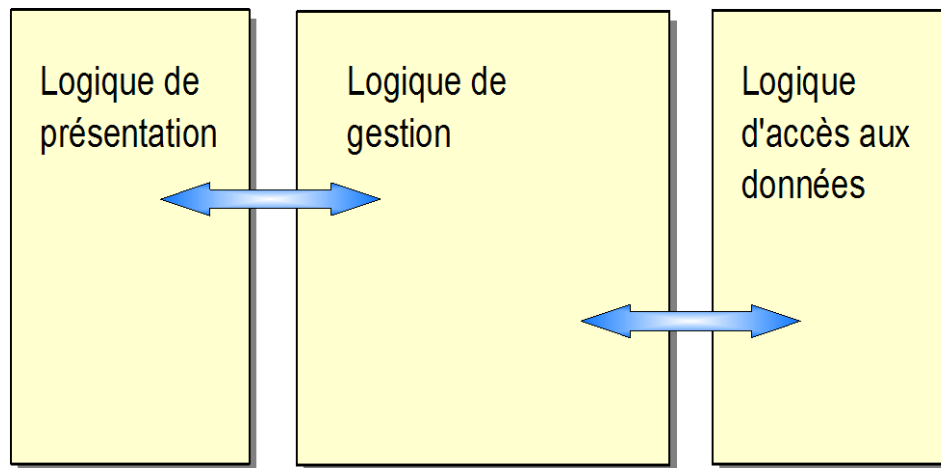


- CICS frontal ou non CICS frontal ?
- Combien de niveaux ?
- Comment concevoir les applications?

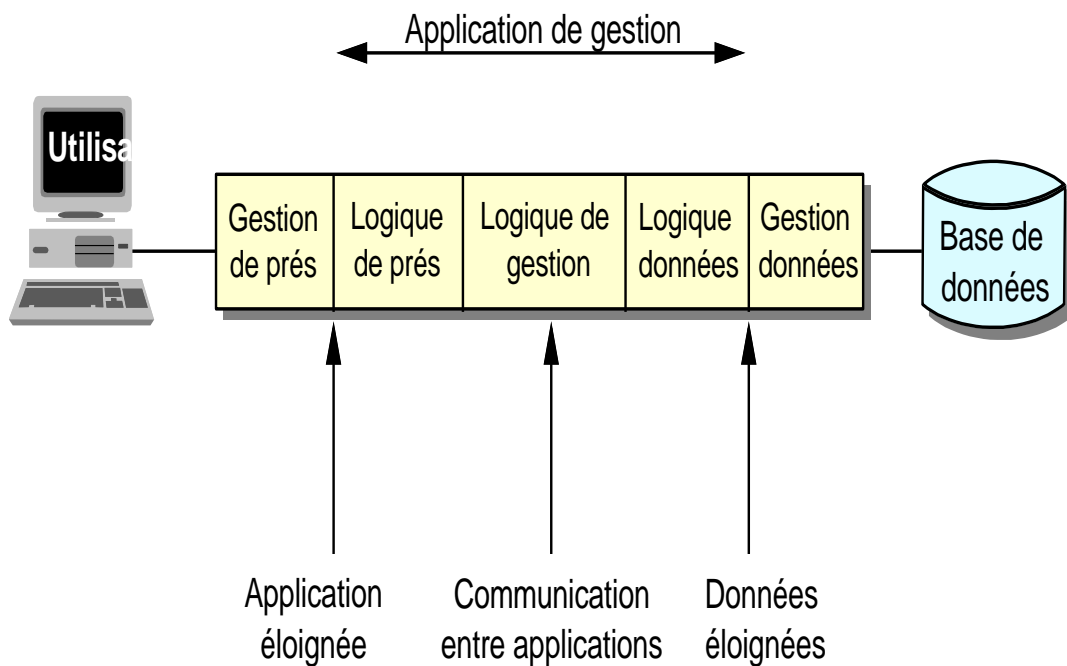
# L'ENVIRONNEMENT CLIENT/SERVEUR

---

## ARCHITECTURE A 3 NIVEAUX



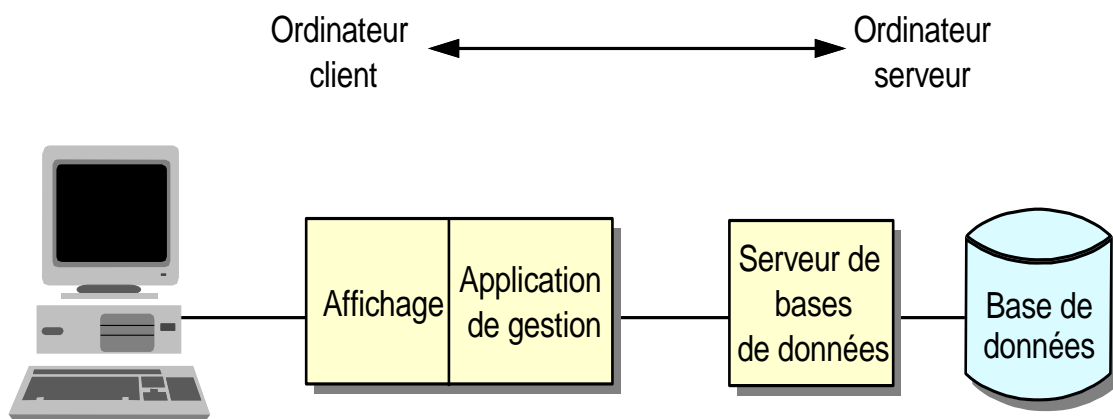
## MODELE D'APPLICATION CLIENT/SERVEUR



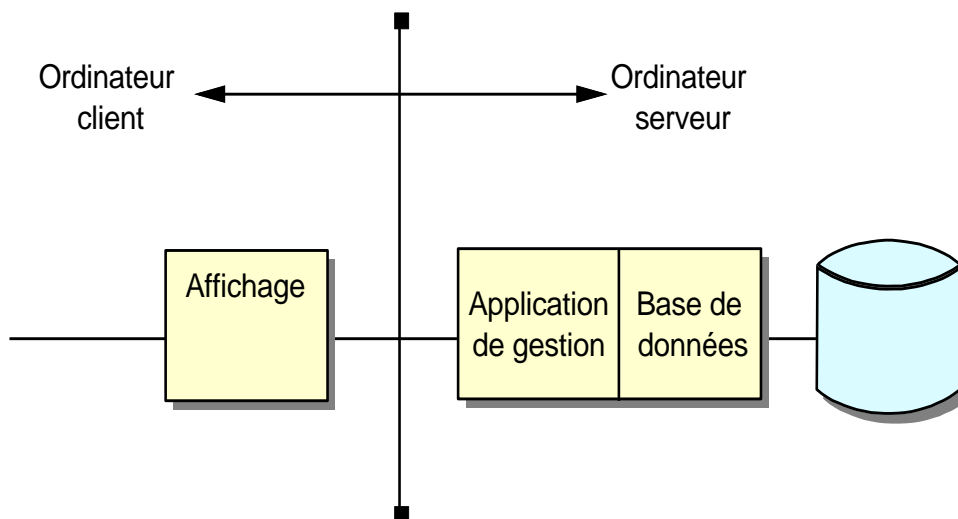
# L'ENVIRONNEMENT CLIENT/SERVEUR

---

## MODELE D'ACCES AUX DONNEES ELOIGNEES



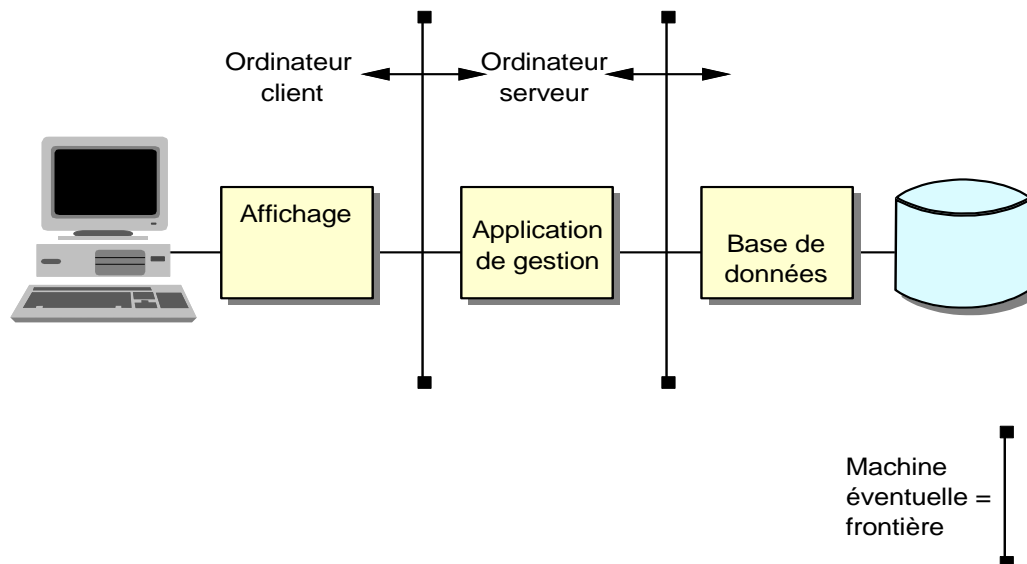
## MODELE DU SERVEUR DE DONNEES



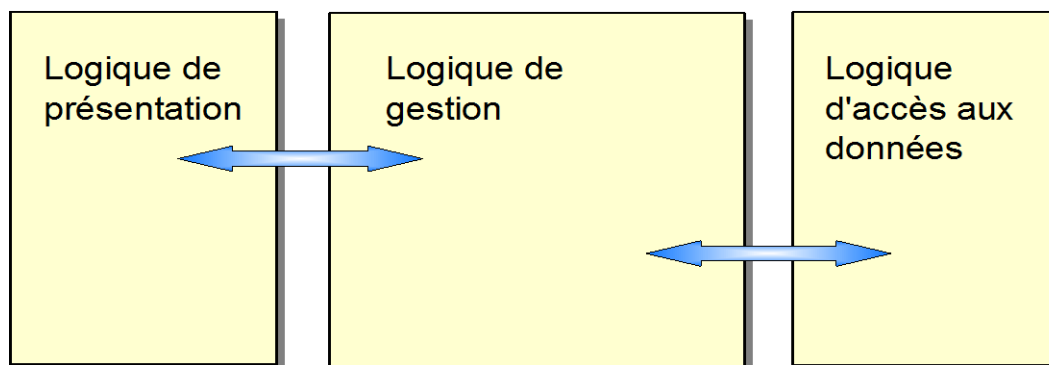
# L'ENVIRONNEMENT CLIENT/SERVEUR

---

## MODELE DU SERVEUR D'APPLICATION



## ARCHITECTURE A TROIS NIVEAUX



- Zone de communication : COMMAREA
- Communication entre programmes
- Programmation pseudo-conversationnelle



## LA COMMUNICATION ENTRE PROGRAMMES

CALL (non CICS, peut être utilisé avec CICS)

LINK (CICS / CICS)

XCTL (CICS / CICS)

ECI CALL (Client CICS / CICS)

EXCI (OS/390 / CICS)

Zone COMMAREA supportée partout

# ANNEXE

# UTILISATION DU CURSLOC

---

## 01 DFHBMSCA.

02 DFHBMEOF                    PIC X   VALUE X'80'. (Effacement de champ)  
02 DFHBMCUR                   PIC X   VALUE X'02'. (Curseur positionné)  
02 DFHBMEC                   PIC X   VALUE X'82'. (Effacement de champ avec curseur positionné)

## 02 DFHBMFLG                    PIC X.

88 DFHERASE        VALUES ARE X'80', X'82'.

**88 DFHCURSR        VALUES ARE X'02' X'82'.**

### Dans Programme

**MOVE CHAMP1F TO DFHBMFLG. (1)**

**(CHAMP1F est le généré de la Map pour CHAMP1 QUI génère :**

- 1/ CHAMP1**L**,CHAMP1**F**, CHAMP1**I**    pour    **INPUT**
- 2/ CHAMP1**L**,CHAMP1**A**, CHAMP1**O**    pour    **OUTPUT**

**IF DFHCURSR THEN ...** (=> le curseur était sur ce champ)  
(Ou + simplement **IF CHAMP1F = X'02' ou DFHBMCUR**)  
...

- (1) Lorsque l'on définit une MAP en mettant dans **DFHMSD** ou **DFHMDI** le paramètre **CURSLOC=YES**  $\implies$  lors d'un **RECEIVE MAP** **BMS** renvoie un Flag à X'02' ou X'82' dans le **champF** ou était positionné le curseur.

### Exemple de définition de MAP

```
MAP4C7  DFHMSD  TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL,EXTATT=YES,           X
          CTRL=(FREEKB,FRSET),STORAGE=AUTO,TIOAPFX=YES,CURSLOC=YES
```

(EXTATT=YES permet de générer des attributs étendu pour gérer la couleur)

# DFHBMSCA

DFHBMSCA : Attributs standards	
CHAMP	Explication
DFHBMPPEM	Printer end-of-message
DFHBMPNL	Printer new-line
DFHBMPFF	Printer form feed
DFHBMPCR	Printer carriage return
DFHBMASK	Autoskip
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected and numeric
DFHBMPRO	Protected
DFHBMBRY	Bright
DFHBMDAR	Dark
DFHBMFSE	MDT set
DFHBMPRF	Protected and MDT set
DFHBMAF	Autoskip and MDT set
DFHBMAF	Autoskip and bright
DFHBMPSO	shift-out value X'0E'.
DFHBMPFI	shift-in value X'0F'.
DFHBMEOF	Field erased
DFHBMCUR	Field containing cursor flagged
DFHBMER	Erased field containing cursor (COBOL only)
DFHBMFLG	Flags (COBOL only)
DFHBMDET	Field detected
DFHSA(1)	Set attribute (SA) order
DFHERROR	Error code
DFHCOLOR(1)	Color
DFHPS(1)	Programmed symbols
DFHHLT(1)	Highlight
DFH3270(1)	Base 3270 field attribute
DFHVAL	Validation
DFHOUTLN	Field outlining attribute code
DFHBKTRN	Background transparency attribute code
DFHALL(1)	Reset all to defaults
DFHDFT	Default
DFHDFCOL(1)	Default color
DFHBLUE	Blue
DFHRED	Red
DFHPINK	Pink
DFHGREEN	Green

DFHTURQ	Turquoise
---------	-----------

## DFHBMSCA (SUITE)

DFHYELLO	Yellow
DFHNEUTR	Neutral
DFHBASE(1)	Base programmed symbols
DFHDFHI(1)	Normal
DFHBLINK	Blink
DFHREVRS	Reverse video
DFHUNDLN	Underscore
DFHMFIL(2)	Mandatory fill
DFHMENT(2)	Mandatory enter
DFHMFEE	Mandatory fill and mandatory enter
DFHMT	Trigger
DFHMFT	Mandatory fill and trigger
DFHMET	Mandatory enter and trigger
DFHMFET	Mandatory fill and mandatory enter and trigger
DFHUNNOD	Unprotected, nondisplay, nonprint, nondetectable, MDT
DFHUNIMD	Unprotected, intensify, light-pen detectable, MDT
DFHUNNUM	Unprotected, numeric, MDT
DFHUNNUB	Unprotected, numeric, intensify, intensify, light-pen detectable
DFHUNINT	Unprotected, numeric, intensify, light-pen detectable, MDT
DFHUNNON	Unprotected, numeric, nondisplay, nonprint, nondetectable, MDT
DFHPROTI	Protected, intensify, light-pen detectable
DFHPROTN	Protected, nondisplay, nonprint, nondetectable
DFHDFFR	Default outline
DFHUNDER	Underline
DFHRIGHT	Right vertical line
DFHOVER	Overline
DFHLEFT	Left vertical line
DFHBOX	Underline and right vertical and overline and left vertical
DFHSOSI	SOSI=yes
DFHTRANS	Background transparency
DFHOPAQ	No background transparency
<p>(1) For text processing only. Use for constructing embedded set attribute orders in user text.</p> <p>(2) Cannot be used in set attribute orders.</p>	

# DFHBMSCA

## 01 DFHBMSCA.

```
02 DFHBMPPEM PICTURE X VALUE IS '□'.
02 DFHBMPNL PICTURE X VALUE IS '...'.
02 DFHBMPFF PICTURE X VALUE IS ' '.
02 DFHBMPCR PICTURE X VALUE IS ' '.
02 DFHBMASK PICTURE X VALUE IS '0'.
02 DFHBMUNP PICTURE X VALUE IS ' '.
02 DFHBMUNN PICTURE X VALUE IS '&'.
02 DFHBMPRO PICTURE X VALUE IS '-'.
02 DFHMBRY PICTURE X VALUE IS 'H'.
02 DFHBMDAR PICTURE X VALUE IS '<'.
02 DFHBMFSE PICTURE X VALUE IS 'A'.
02 DFHBMPRF PICTURE X VALUE IS '/'.
02 DFHBMAF PICTURE X VALUE IS '1'.
02 DFHBMASB PICTURE X VALUE IS '8'.
02 DFHBMEOF PICTURE X VALUE IS 'Ä'.
02 DFHBMCUR PICTURE X VALUE IS '□'.
02 DFHBMEC PICTURE X VALUE IS 'b'.
02 DFHBMFLG PICTURE X.
88 DFHERASE VALUES ARE 'Ä', 'b'.
88 DFHCURSR VALUES ARE '□', 'b'.
02 DFHBMDDET PICTURE X VALUE IS 'ÿ'.
02 DFHBMPISO-BIN PIC 9(4) COMP VALUE 3599.
* ABOVE VALUE 3599 = X'0E0F' ADDED BY PTM 81385 (APAR PN23267)
02 FILLER REDEFINES DFHBMPISO-BIN.
03 DFHBMPISO PICTURE X.
03 DFHBMPISI PICTURE X.
02 DFHSA PICTURE X VALUE IS '^'.
02 DFHCOLOR PICTURE X VALUE IS 'i'.
02 DFHPS PICTURE X VALUE IS 'ç'.
02 DFHHLT PICTURE X VALUE IS ' '.
02 DFH3270 PICTURE X VALUE IS '{'.
02 DFHVAL PICTURE X VALUE IS 'A'.
02 DFHOUTLN PICTURE X VALUE IS 'B'.
02 DFHBKTRN PICTURE X VALUE IS '¥'.
02 DFHALL PICTURE X VALUE IS ' '.
02 DFHERROR PICTURE X VALUE IS '□'.
02 DFHDFT PICTURE X VALUE IS 'ÿ'.
02 DFHDFCOL PICTURE X VALUE IS ' '.
02 DFHBLUE PICTURE X VALUE IS '1'.
02 DFHRED PICTURE X VALUE IS '2'.
02 DFHPINK PICTURE X VALUE IS '3'.
02 DFHGREEN PICTURE X VALUE IS '4'.
02 DFHTURQ PICTURE X VALUE IS '5'.
02 DFHYELLO PICTURE X VALUE IS '6'.
02 DFHNEUTR PICTURE X VALUE IS '7'.
02 DFHBASE PICTURE X VALUE IS ' '.
02 DFHDFHI PICTURE X VALUE IS ' '.
02 DFHBLINK PICTURE X VALUE IS '1'.
02 DFHREVRS PICTURE X VALUE IS '2'.
02 DFHUNDLN PICTURE X VALUE IS '4'.
02 DFHMFIL PICTURE X VALUE IS 'œ'.
02 DFHMENT PICTURE X VALUE IS '□'.
02 DFHMFEE PICTURE X VALUE IS '+'.

```

```

02      DFHUNNOD  PICTURE X    VALUE IS  '(''.
02      DFHUNIMD  PICTURE X    VALUE IS  'I''.
02      DFHUNNUM  PICTURE X    VALUE IS  'J''.
02      DFHUNNUB  PICTURE X    VALUE IS  'Q'.'.
* ABOVE VALUE DFHUNNUB ADDED BY APAR PN67669

```

## DFHBMSCA (SUITE)

---

```

02      DFHUNINT  PICTURE X    VALUE IS  'R'.'.
02      DFHUNNON  PICTURE X    VALUE IS  ') ' '.
02      DFHPROTI  PICTURE X    VALUE IS  'Y'.'.
02      DFHPROTN  PICTURE X    VALUE IS  '% ' '.
02      DFHMT     PICTURE X    VALUE IS  '□'.'.
02      DFHMFT    PICTURE X    VALUE IS  ' ' ' '.
02      DFHMET    PICTURE X    VALUE IS  '□'.'.
02      DFHMFET   PICTURE X    VALUE IS  '□'.'.
02      DFHDFFR   PICTURE X    VALUE IS  ' ' ' '.
02      DFHLEFT   PICTURE X    VALUE IS  '—'.'.
02      DFHOVER   PICTURE X    VALUE IS  'œ'.'.
02      DFHRIGHT  PICTURE X    VALUE IS  '□'.'.
02      DFHUNDER  PICTURE X    VALUE IS  '□'.'.
02      DFHBOX-BIN PIC 9(4) COMP VALUE 15.
* ABOVE VALUE 15 = X'000F' ADDED BY PTM 81385 (APAR PN23267)
02      FILLER REDEFINES DFHBOX-BIN.
    03  FILLER    PICTURE X.
    03  DFHBOX    PICTURE X.
02      DFHSOSI   PICTURE X    VALUE IS  '□'.'.
02      DFHTRANS  PICTURE X    VALUE IS  '0'.'.
02      DFHOPAQ   PICTURE X    VALUE IS  'ÿ'.'.

```



## LE TERMINAL 3270

---

Contenu de l'octet attribut :

Signification des bits

x	x	u/p	A/N	Affich	res	MDT	
0	1	2	3	4	5	6	7

Bits	signification
0	système
1	système, toujours à 1
2	“00”non protégé alphanum.
3	“01”non protégé numérique “10”protégé “11”protégé autoskip
4	“00”affichage normal non détectable
5	“01”affichage normal détect. “10”double brillance “11”non affichable (dark)
6	Toujours à 0
7	Réputé modifié ou non : MDT on si 1 MDT off si 0

# LES ATTRIBUTS

Positionnement des attributs								
prot	a/n	hi	spd	ndp	mdt	ebcd	ascii	char
U						40	20	b (blank)
U					Y	C1	41	A
U			Y			C4	44	D
U			Y		Y	C5	45	E
U		H	Y			C8	48	H
U		H	Y		Y	C9	49	I
U				Y		4C	3C	<
U				Y	Y	4D	28	(
U	N					50	26	
U	N				Y	D1	4A	J
U	N		Y			D4	4D	M
U	N		Y		Y	D5	4E	N
U	N	H	Y			D8	51	Q
U	N	H	Y		Y	D9	52	R
U	N			Y		5C	2A	*
U	N			Y	Y	5D	29	)
P						60	2D	- (hyphen)
P					Y	61	2F	/
P			Y			E4	55	U
P			Y		Y	E5	56	V
P		H	Y			E8	59	Y
P		H	Y		Y	E9	5A	Z
P				Y		6C	25	%
P				Y	Y	6D	5F	<u> (underscore)</u>
P	S					F0	30	0
P	S				Y	F1	31	1
P	S		Y			F4	34	4
P	S		Y		Y	F5	35	5
P	S	H	Y			F8	38	8
P	S	H	Y		Y	F9	39	9
P	S			Y		7C	40	@
P	S			Y	Y	7D	27	'
The attributes in the headings are:								
prot = protected a/n = automatic skip or numeric hi = high intensity					spd = selector pen detectable ndp = nondisplay print mdt = modified data tag			
ASCII : ASCII			EBC : EBCDIC					
H = High P = Protected			N = Numeric U = Unprotected			S = Automatic skip Y = Yes		
GRAMMATION CICS/TS								

# LES ATTRIBUTS

DFHBMSCA : Attributs standards	
CHAMP	Explication
DFHBMPPEM	Printer end-of-message
DFHBMPNL	Printer new-line
DFHBMPFF	Printer form feed
DFHBMPCR	Printer carriage return
DFHBMASK	Autoskip
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected and numeric
DFHBMPRO	Protected
DFHBMCRY	Bright
DFHBMDAR	Dark
DFHBMFSE	MDT set
DFHBMPRF	Protected and MDT set
DFHBMAF	Autoskip and MDT set
DFHBMAF	Autoskip and bright
DFHBMPSO	shift-out value X'0E'.
DFHBMPFI	shift-in value X'0F'.
DFHBMEOF	Field erased
DFHBMCR	Field containing cursor flagged
DFHBMER	Erased field containing cursor (COBOL only)
DFHBMFLG	Flags (COBOL only)
DFHBMDET	Field detected
DFHSA(1)	Set attribute (SA) order
DFHERROR	Error code
DFHCOLOR(1)	Color
DFHPS(1)	Programmed symbols
DFHHLT(1)	Highlight
DFH3270(1)	Base 3270 field attribute
DFHVAL	Validation
DFHOUTLN	Field outlining attribute code
DFHBKTRN	Background transparency attribute code
DFHALL(1)	Reset all to defaults
DFHDFT	Default
DFHDFCOL(1)	Default color
DFHBLUE	Blue
DFHRED	Red
DFHPINK	Pink
DFHGREEN	Green
DFHTURQ	Turquoise
DFHYELLO	Yellow

# LES ATTRIBUTS

DFHNEUTR	Neutral
DFHBASE(1)	Base programmed symbols
DFHDFHI(1)	Normal
DFHBLINK	Blink
DFHREVRS	Reverse video
DFHUNDLN	Underscore
DFHMFIL(2)	Mandatory fill
DFHMENT(2)	Mandatory enter
DFHMFEE	Mandatory fill and mandatory enter
DFHMT	Trigger
DFHMFT	Mandatory fill and trigger
DFHMET	Mandatory enter and trigger
DFHMFET	Mandatory fill and mandatory enter and trigger
DFHUNNOD	Unprotected, nondisplay, nonprint, nondetectable, MDT
DFHUNIMD	Unprotected, intensify, light-pen detectable, MDT
DFHUNNUM	Unprotected, numeric, MDT
DFHUNNUB	Unprotected, numeric, intensify, intensify, light-pen detectable
DFHUNINT	Unprotected, numeric, intensify, light-pen detectable, MDT
DFHUNNON	Unprotected, numeric, nondisplay, nonprint, nondetectable, MDT
DFHPROTI	Protected, intensify, light-pen detectable
DFHPROTN	Protected, nondisplay, nonprint, nondetectable
DFHDFFR	Default outline
DFHUNDER	Underline
DFHRIGHT	Right vertical line
DFHOVER	Overline
DFHLEFT	Left vertical line
DFHBOX	Underline and right vertical and overline and left vertical
DFHSOSI	SOSI=yes
DFHTRANS	Background transparency
DFHOPAQ	No background transparency
<p>(1) For text processing only. Use for constructing embedded set attribute orders in user text.</p> <p>(2) Cannot be used in set attribute orders.</p>	

# LES ATTRIBUTS

Positionnement des attributs								
prot	a/n	hi	spd	ndp	mdt	ebcd	ascii	char
U						40	20	b (blank)
U					Y	C1	41	A
U			Y			C4	44	D
U			Y		Y	C5	45	E
U		H	Y			C8	48	H
U		H	Y		Y	C9	49	I
U				Y		4C	3C	<
U				Y	Y	4D	28	(
U	N					50	26	
U	N				Y	D1	4A	J
U	N		Y			D4	4D	M
U	N		Y		Y	D5	4E	N
U	N	H	Y			D8	51	Q
U	N	H	Y		Y	D9	52	R
U	N			Y		5C	2A	*
U	N			Y	Y	5D	29	)
P						60	2D	- (hyphen)
P					Y	61	2F	/
P			Y			E4	55	U
P			Y		Y	E5	56	V
P		H	Y			E8	59	Y
P		H	Y		Y	E9	5A	Z
P				Y		6C	25	%
P				Y	Y	6D	5F	<u>(underscore)</u>
P	S					F0	30	0
P	S				Y	F1	31	1
P	S		Y			F4	34	4
P	S		Y		Y	F5	35	5
P	S	H	Y			F8	38	8
P	S	H	Y		Y	F9	39	9
P	S			Y		7C	40	@
P	S			Y	Y	7D	27	'
prot = protected a/n = automatic skip or numeric hi = high intensity					spd = selector pen detectable ndp = nondisplay print mdt = modified data tag			
ASCII : ASCII			EBC : EBCDIC					
H = High P = Protected			N = Numeric U = Unprotected			S = Automatic skip Y = Yes		

# GLOSSAIRE CICS/TS

## LISTE DES SIGLES

---

ACB	Access Method Control Block
ACF	Advanced Communication Function
ACP	Abnormal Condition Program
AFCB	Authorized Function Control Block (z/OS)
AFCTE	Application File Control Table Entry
AICA	Runaway Transaction Abend Code
AID	Attention Identifier
AID	Automatic Initiate Descriptor
AIX	VSAM Alternate Index
ACP	Abnormal Condition Program
AFCB	Authorized Function Control Block (z/OS)
AFCTE	Application File Control Table Entry
AICA	Runaway Transaction Abend Code
AID	Attention Identifier
AID	Automatic Initiate Descriptor
AIX	VSAM Alternate Index
AKP	Activity Keypoint
ALC	Assembler Language Coding
ALET	Access List Entry Token

## LISTE DES SIGLES

---

ALP	Terminal Allocation Program
AMODE	Addressing Mode Attribute (XA)
AMXT	Active Maximum Tasks
AOR	Application Owning Region (Transaction Routing)
AP	APPLICATION DOMAIN
APAR	Authorized Program Analysis Report
APE	LD Domain Active Program Element
APF	Authorized Program Facility
API	Application Programmer Interface
APPC	Advanced Program To Program Communication
APRM	Application Programmer's Reference Manual
ASCII	American National Standard Code For Information Interchange
ASRA	Program Check, Transaction Abend Code
ASRB	OS Abend, Transaction Abend Code
ATI	Automatic Task Initiation
AWE	Autoinstall Work Element
BB	SNA Begin Bracket



BCCB            Base Cluster Control Block/DSN Control Block

## LISTE DES SIGLES

---

BDAM           Basic Direct Access Method (OS)

BDW           Block Descriptor Word

BLL           Base Locator For Linkage (Cobol)

BMS           Basic Mapping Support Program

BSAM           Basic Sequential Access Method (OS)

BSC           Binary Synchronous Control

BTAM           Basic Telecommunications Access Method

CA            VSAM Control Area

CBIPO          Custom Built Installation Process Offering

CBPDO          Custom Built Product Delivery Offering

CBRC          CICS Database Recovery Control Transaction

CC            LOCAL CATALOG DOMAIN

CCB           Channel Control Block (DOS)

CCB           Connection Control Block (MRO)

CCGD          CC/GC Domains anchor blocks

CCHHR          Cylinder head record (disk address)

CCP           Catalog Control Program

CDBC          CICS database control transaction

CDBI            CICS database control inquiry transaction

## LISTE DES SIGLES

---

CEBR	CICS temporary storage browse transaction
CEBT	CICS master terminal transaction (XRF alternate)
CEC	Central electronic complex
CECI	CICS command-level interpreter transaction
CEDA	CICS resource definition online transaction
CEDF	CICS Command-Level Execution Diagnostic Facility Transaction
CEMT	CICS master terminal transaction
CEOT	CICS terminal status transaction
CESD	Composite external symbol dictionary
CESF	CICS sign-off transaction
CESN	CICS sign-on transaction
CEST	CICS supervisor terminal transaction
CETR	Trace control transaction
CI	VSAM control interval
CICS	Customer Information Control System
CIDF	VSAM control interval definition field
CINIT	SNA control initiate
CLSDST	VTAM close destination macro
CMF	CICS monitoring facility

CMSG CICS message switching transaction

## LISTE DES SIGLES

---

CMXT	Class maximum tasks
CO	Concurrent-mode TCB
COBOL	Common Business Oriented Language
CPE	LD Domain Current Program Element
CPU	Central Processing Unit
CRB	Cross Region Block (MRO)
CRTE	CICS Routing Transaction
CSA	Common Service Area (z/OS)
CSA	Common System Area
CSB	Connection Status Block (MRO)
CSD	CICS System Definition File
CSECT	Assembler Control Section
CSFE	CICS Field Engineering Transaction
CSP	Cross System Product
CSPG	CICS Page Retrieval Transaction
CSV	Journal Buffer Current Sliding Value
CWA	Common Work Area
CWTO	CICS Write To Operator Transaction

## LISTE DES SIGLES

---

DAM	Direct Access Method (DOS)
DASD	Direct Access Storage Device
DB2	Data Base Two
DBCS	Double Byte Character Set
DBCTL	IMS/ESA Database Control
DBMS	Database Management System
DBP	Dynamic Backout Program
DCB	Data Control Block (OS)
DCP	Dump Control Program
DCT	Destination control table
DD	Data definition
DECB	Data event control block
DFP	Data facility product
DISOSS	Distributed office support system
DL/I	Data language one
DLBL	Data label (DOS)
DLC	Data link control
DM	DOMAIN MANAGER DOMAIN
DMCB1	DM Domain anchor block

## LISTE DES SIGLES

---

DMS	Development management system
DS	DISPATCHER DOMAIN
DSA	Dynamic storage area
DSA	Dynamic storage area (PL/I)
DSANC	DS Domain anchor block
DSCB	Dataset control block
DSECT	Assembler dummy section
DSN	Data set name
DTA	Dispatcher task area
DTB	Dynamic transaction backout
DTF	Define the file (DOS)
DTP	Distributed transaction processing
DU	DUMP DOMAIN
DWE	Deferred work element
DYP	Dynamic Transaction Routing Program
EB	SNA End Bracket
EBCDIC	Extended Binary-Coded Decimal Interchange Code
ECB	Event Control Block
ECSA	Extended Common Service Area (XA)

## LISTE DES SIGLES

---

EDF	Execution Diagnostic Facility
EDSA	Extended Dynamic Storage Area
EIB	Exec Interface Block (C/L)
EIP	Exec Interface Program (C/L)
EIS	Exec Interface Storage (C/L)
ELPA	Extended Link Pack Area (XA)
ELSQA	Extended Local System Queue Area (XA)
EOJ	End Of Job
EP	Emulation Program
ESA	Enterprise System Architecture
ESCS	Extended Storage Cushion
ESDS	VSAM Entry Sequenced Data Set
ESM	External Security Manager
ESPIE	Extended Specify Program Interrupt Exit (z/OS)
ESQA	Extended System Queue Area (XA)
ESTAE	Extended Specify Task Abnormal Exit (z/OS)
ESWA	Extended Scheduler Work Area (XA)
EXCP	Execute Channel Program

## LISTE DES SIGLES

---

FBO	File Backout Table
FBWA	File Browse Work Area
FCP	File Control Program
FCT	File Control Table
FCTSR	File Control Table Shared Resources Control Block
FFDC	First Failure Data Capture
FIOA	File Input/Output Area
FMH	Function Management Header
FWA	File Work Area
GC	Global Catalog Domain
GCD	Global Catalog
GLUE	Global User Exit
GTF	Generalized Trace Facility
HASP	Houston Automated Spooling Program
HLL	High Level Language

HPO High Performance Option

## LISTE DES SIGLES

---

ICE	Interval Control Element
ICF	Integrated Catalog Facility
ICP	Interval Control Program
ICV	Partition/Region Exit Interval
ICVR	Runaway Task Interval
ICVSWT	Short Wait Interval (DOS)
ICVTSD	Terminal Scan Delay Interval
III	Initialization System Task ID
IMS	Information Management System
IPCS	Interactive Problem Control System
IPL	Initial Program Load
IPO	Installation Product Offering
IRB	Interrupt Request Block
IRC	Inter-Region Communication (MRO)
IRLM	IMS/VS Resource Lock Manager
IRP	Inter-Region Program
ISA	Initial Storage Area (PL/I)
ISAM	Indexed Sequential Access Method
ISC	Inter-System Communication (VTAM)
ISO	International Standards Organization
ISPF	Interactive System Productivity Facility



IWS                      Interactive Work Station

## LISTE DES SIGLES

---

JAP                      Journal Archive Program

JASP                    Journal Archive Submission Program

JCA                      Journal Control Area

JCL                      Job Control Language

JCP                      Journal Control Program

JCT                      Journal Control Table

JES                      Job Entry Subsystem (z/OS)

JJJ                      Journal Control System Task ID

KCB                      KE Domain Anchor Block

KCP                      Task Control Program

KCP                      Task Control System Task ID (Jusqu'en CICS/ MVS)

KCS                      Task Control Static Storage

KE                        Kernel Domain

KSDS                    VSAM Key Sequenced Data Set

KSS                      KE Domain Stack Segment

LACB                    Logon Address Control Block (MRO)

LCB      Logon Control Block (MRO)

## LISTE DES SIGLES

---

LCD	Local Catalog
LD	Loader Domain
LDC	Logical Device Code
LDCBS	LD Domain Anchor Block
LDS	VSAM Linear Dataset
LECB	Logical Event Control Block
LECB	Line Event Control Block
LIFO	Last In-First Out Task/Kernel Level Storage
LIOA	Line Input/Output Area
LLA	Library Lookaside (XA)
LLA	Load List Area
LM	Lock Manager Domain
LMCB1	LM Domain Anchor Block
LMPEO	VTAM Large Message Performance Enhancement Option
LPA	Link Pack Area (z/OS)
LRU	Least Recently Used
LSQA	Local System Queue Area (z/OS)
LSR	VSAM Local Shared Resources
LU	SNA Logical Unit
LUW	Logical Unit Of Work

## LISTE DES SIGLES

---

MBCA	Transient Data Buffer Common Area
MBCB	Transient Data Buffer Control Block
MBO	Message Backout Table
MCS	Multiple Console Support (z/OS)
MCT	Monitoring Control Table
ME	Message Domain
MEPS	ME Domain Anchor Block
MLPA	Modified Link Pack Area (z/OS)
MN	Monitoring Domain
MNA	MN Domain Anchor Block
MQCB	Transient Data Queue Control Block
MRCA	Transient Data String Common Area
MRCB	Transient Data String Control Block
MRO	Multi-Region Operation (IRC)
MRSD	Transient Data Segment Descriptor (Bit Map)
MSR	Magnetic Strip Reader (32xx Devices)
MTO	Master Terminal Operator
MVS	Multiple Virtual Storage
ESA	Enterprise Systems Architecture

## LISTE DES SIGLES

---

MWCB Transient Data Wait Control Block

MXT Maximum Tasks

NACP Node Abnormal Condition Program

NCP Network Control Program

NDT Terminal Control Next Dispatch Time

NEP Node Error Program (VTAM)

NIB Node Initialization Block

NLS National Language Support

NOP Assembler No Operation

NSP VSAM Note String Positioning

NSR VSAM Non-Shared Resource

OCO Object Code Only

OFL Optional Features List

OLTP On-Line Teleprocessing

OPNDST VTAM Open Destination Macro

## LISTE DES SIGLES

---

PA	Parameter Domain
PAA	PA Domain Anchor Block
PCP	Program Control Program
PCS	Program Control Static Storage
PCT	Program Control Table
PCTR	Program Control Table - Remote Entries
PDS	Partitioned Data Set
PEP	Partitioned Emulation Program
PEP	Program Error Program
PFT	Profile Table
PGA	Page Control Area
PGT	Program Global Table (Cobol)
PLH	VSAM Place Holder
PLT	Program List Table
PLTPI	Program List Table, Post Initialization
PLTSD	Program List Table, Shut Down
PLU	SNA Primary Logical Unit
PPA	SM Domain Page Pool Control Area
PPT	Processing Program Table
PPT	Program Properties Table (z/OS)

PSA                      Prefixed Storage Area

## LISTE DES SIGLES

---

PSW                    Program Status Word

PTF                    Program Temporary Fix

QCA                    Queue Control Area (OS)

QEA                    Queue Element Area

QR                     Quasi-Reentrant TCB

QSAM                  Queued Sequential Access Method

RA                     VTAM Receive Any

RACE                  Receive Any Control Element

RACF                  Resource Access Control Facility

RAIA                  Receive Any Input Area

RBA                    Relative Byte Address

RCP                    Recovery Control Program

RCS                    Recovery Control Static Storage

RDF                    VSAM Record Definition Field

RDO                    Resource Definition On-Line (CEDA)

RDW                    Record Descriptor Word

RLD Relocatable Dictionary

## LISTE DES SIGLES

---

RMODE Residency Mode Attribute (XA)

RO Resource-Owning TCB

RPL Relocatable Program Library

RPL Request Parameter List

RRDS VSAM Relative Record Data Set

RSA Register Storage Area

RSD Restart Data Set

RST DBCTL Recoverable Service Table

RTM Recovery/Termination Manager (z/OS)

RU SNA Request/Response Unit

SAA Storage Accounting Area

SAA Systems Application Architecture

SAB Subsystem Anchor Block

SAM Sequential Access Method (DOS)

SCA SM Domain Subpool Control Area

SCACB	Subsystem Connection Address Control Block (MRO)
-------	--

## LISTE DES SIGLES

---

SCCB	Subsystem Connection Control Block (MRO)
SCE	SM Domain Element Descriptor
SCF	SM Domain Free Storage Descriptor
SCP	Storage Control Program
SCQ	SM Domain Quickcell Element
SCTE	Subsystem Control Table Extension (MRO)
SDF	Screen Definition Facility
SDLC	Synchronous Data Link Control
SIP	System Initialization Program
SIT	System Initialization Table
SKP	Subtask Manager Program
SKW	Subtask Work Queue Element
SLCB	Subsystem Logon Control Block (MRO)
SLU	SNA Secondary Logical Unit
SM	Storage Manager Domain
SMA	SM Domain Anchor Block
SMF	System Management Facility
SMP	System Modification Program
SNA	Systems Network Architecture



SNT	Sign-On Table
SOS	Short On Storage

## LISTE DES SIGLES

---

SPI	Systems Programmer Interface
SPIE	Specify Program Interrupt Exit (OS)
SQA	System Queue Area (z/OS)
SQE	SM Domain Suspend Queue Element
SQL	Structured Query Language
SRB	Service Request Block (z/OS)
SRP	System Recovery Program
SRT	System Recovery Table
SSA	Static Storage Address List
ST	Statistics Domain
STAE	Specify Task Abnormal Exit (OS)
STCA	System Task Control Area
STCB	ST Domain Anchor Block
STIMER	Set Timer (OS)
STIMERM	Set Timer Multiple (XA)
STP	System Termination Program
STXIT	Set Exit (DOS)
SUDB	Subsystem User Definition Block (MRO)
SUV	Journal Buffer Shift-Up Value
SVA	Shared Virtual Area (DOS)

---

SVC	Supervisor Call
SWA	Scheduler Work Area (z/OS)

## LISTE DES SIGLES

---

TACB	Transaction Abend Control Block
TACLE	Terminal Abnormal Condition Line Entry
TBO	Transaction Backout Table
TBP	Transaction Backout Program
TBS	Table Builder Services Program (Autoinstall)
TCA	Task Control Area Or User Portion Of TCA
TCAM	Telecommunications Access Method
TCB	Task Control Block (z/OS)
TCP	Terminal Control Program (Non-VTAM Support)
TCP	Terminal Control System Task ID
TCT	Terminal Control Table
TCTLE	Terminal Control Table Line Entry
TCTME	Terminal Control Table Mode Entry
TCTSE	Terminal Control Table System Entry
TCTSK	Terminal Control Table - Skeleton Entries
TCTTE	Terminal Control Table Terminal Entry
TCTWE	TCT Work Element
TSVCA	Temporary storage VSWA control area
TDIA	Transient Data Input Area

TDOA            Transient Data Output Area

TDP            Transient Data Program

## LISTE DES SIGLES

---

TDST           Transient Data Static Storage

TEP            Terminal Error Program (BTAM)

TGT            Task Global Table (Cobol)

TI              Timer Domain

TIA            TI Domain Anchor Block

TIE            Transaction Interface Element

TIOA           Terminal Input/Output Area

TLA            Three Letter Acronym

TMP            Table Manager Program

TOR            Terminal Owning Region (Transaction Routing)

TQE            Transaction Queue Element

TR             Trace Domain

TRA            TR Domain Anchor Block

TRP            Trace Control Program

TRT            Trace Table

TSACA          Temporary Storage Auxiliary Control Area

TSBCA          Temporary Storage Buffer Control Area

TSBM           Temporary Byte Map

TSGID          Temporary Storage Group Identifier

TSIOA Temporary Storage Input/Output Area

TSO Time Sharing Option

## LISTE DES SIGLES

---

TSP Temporary Storage Program

TSQE Temporary Storage Queue Element

TSRE Temporary Storage Request Element

TSREB Temporary Storage Request Element Block

TST Temporary Storage Table

TSUT Temporary Storage Unit Table

TTC DFHTMP Table Type Code

TTR Track And Record (Disk Address)

TWA Transaction Work Area

UEPAR User Exit Parameter List (GLUE & TRUE)

UOW Unit Of Work

URD Unit Of Recovery Descriptor

VB Variable Blocked Record Format

VBS      Variable Blocked Spanned Record Format

VLF      Virtual Lookaside Facility (TS)

## LISTE DES SIGLES

---

VM      Virtual Machine

VSAM    Virtual Storage Access Method

VSCA    VSAM Subtask Control Area

VSWA    VSAM Work Area

VTAM    Virtual Telecommunications Access Method

VTOC    Volume Table Of Contents

WTO      Write To Operator

WTOR    Write To Operator With Reply

XA      Extended Architecture

XECB    Cross Partition Event Control Block (VSE)

XLT      Transaction List Table

XM      Transaction Manager

XPCC    Cross Partition Communication Control (VSE/SP)

XPI      Exit Programmer Interface

XRF      Extended Recovery Facility

## LISTE DES SIGLES

---

ZACT	Terminal Control Activate Queue Chain Subprogram
ZARQ	Terminal Control Application Request Logic (ZCP)
ZATD	Terminal Control Autoinstall Program
ZCP	Terminal Control Program (VTAM)
ZCQ	Terminal Control Bind Analysis Program
ZDSP	Terminal Control Dispatcher Logic (ZCP)
ZLGX	Terminal Control Logon Exit
ZRAC	Terminal Control Race Scan Subprogram