

***Dossier des normes & procédure de  
validation des développements SAP à  
l'APHP.***

Niveau de confidentialité

(1)

<input type="checkbox"/>	Diffusion libre
<input checked="" type="checkbox"/>	Restreinte : APHP & Collaborateurs externes
<input type="checkbox"/>	Restreinte : APHP
<input type="checkbox"/>	Restreinte : CCDG
<input type="checkbox"/>	Destinataires exclusivement

(1) Mettre une croix dans la ou les cases correspondante(s)

## SUIVI DES MODIFICATIONS

Vers.	Date	Nature de la modification
V0.1	15/03/2015	Création du document (Aimé Badrane).
V0.12	15/03/2015	Ajout Par M. Salhi de Normes abap V4 Avancé
V0.2	07/08/2015	Ajout des chapitres « validations via l'outil Cast »
V0.3	01/09/2015	Intégration des contraintes de livraison selon le marché de l'AT
V0.4	10/09/2015	Relecture croisée avec Ph. Brillant
V1.0	16/09/2015	Emission de la version V1.0
V1.1	Novembre 2015	Réunion d'échanges sur les normes applicables
V1.2	Décembre 2015	Prise en compte des remarques dans la qualimétrie .

Vers.	Date	Élaboration	visa	Vérification	visa	Validation	visa
V1.0	16/09/15	Aimé BADRANE					
V2.0	21/01/16	Aimé BADRANE		Philippe BRILLANT		Jean-Michel GAILLOT	

Diffusion

Version(s)	Date(s)	Noms des destinataires
V1.0	16/09/2015	Personnel du CCDG, ATI , DSI
V1.1	Novembre 2015	Accenture, Sopra, TJC
V1.2	21 Janvier 2016	Domaines CCDG Accenture, Sopra, TJC

## SOMMAIRE

<b>1. PREAMBULE .....</b>	<b>6</b>
<b>1ERE PARTIE :NORMES DE DEVELOPPEMENTS SAP .....</b>	<b>7</b>
1.1 NORMES DE CODIFICATION DES OBJETS SAP .....	7
1.1.1 Généralités .....	7
1.1.2 Programme, Fonction, Include et Report.....	7
1.1.3 Pool de modules et include de pool de modules.....	9
1.1.4 Groupes de fonctions .....	10
1.1.5 Module de fonctions .....	10
1.1.6 Codes Transaction .....	10
1.1.7 Procédures CATT.....	11
1.1.8 Classes de Développement.....	11
1.1.9 Bases de données logiques.....	11
1.1.10 Objets SAP Standards .....	12
1.1.11 Autres objets SAP .....	12
1.1.12 Dictionnaire de données.....	13
1.1.13 Autorisations .....	Erreur ! Signet non défini.
1.1.14 Formulaires.....	Erreur ! Signet non défini.
1.1.15 Correspondances.....	20
1.1.16 Dossiers Batch input .....	20
1.2 NORMES DE CODIFICATION ABAP /4 .....	21
1.2.1 Attributs des programmes .....	21
1.2.2 Codification des noms de variables .....	22
1.2.3 Codification des noms de tables.....	29
1.2.4 Message d'erreur spécifiques .....	30
1.2.5 PF-status .....	31
1.2.6 Fichiers externes .....	32
1.2.7 Variantes .....	32
1.3 NORMES DE PROGRAMMATION DES REPORTS ABAP / 4 .....	33
1.3.1 Structure du programme .....	33
1.3.2 Bloc de documentation principale.....	33
1.3.3 Présentation générale .....	33
1.3.4 Utilisation de include .....	34
1.3.5 Messages .....	34
1.3.6 Reports .....	34
1.3.7 Batch input data.....	34
1.3.8 Accès aux données .....	34
1.3.9 Mise à jour des données.....	35
1.3.10 Utilisation des fichiers externes .....	37
1.3.11 Variantes .....	37
1.3.12 Sous-programmes.....	38
1.3.13 Autorisations .....	38
1.3.14 Debug .....	38
1.3.15 Instructions spécifiques .....	38
1.3.16 Interface utilisateur (GUI) .....	40
1.3.17 Maintenance / correction d'un code existant en production.....	40
1.3.18 Types d'interface .....	43
1.3.19 Données externes.....	43
1.3.20 Tables internes .....	43
1.4 NORMES DE PROGRAMMATION DES MODULES ABAP / 4 .....	45

1.4.1	Structure du programme .....	45
1.4.2	Définitions des écrans .....	45
1.4.3	Interfaces (GUI) .....	45
1.4.4	Fenêtres POP-UP .....	45
1.4.5	OK_CODE (SY-UCOMM) .....	45
1.5	CONTENU DES ABAPS .....	46
1.5.1	Présentation des listes .....	46
1.5.2	Contrôle des autorisations .....	46
1.6	PROFIL DU DEVELOPPEUR .....	47
1.7	TRANSPORT (FONCTIONNEMENT GENERAL) .....	48
1.7.1	OBJET .....	48
1.7.2	DOMAINE D'APPLICATION .....	48
1.7.3	DEROULEMENT D'UN TRANSPORT .....	48
1.8	DEVELOPPEMENT DES MATCHCODES .....	50
1.8.1	Ajout d'un nouvel ID (index) MATCHCODE sur un matchcode standard existant .....	50
1.8.2	Création d'un nouveau MATCHCODE pour développement spécifique .....	50
1.8.3	Création d'un nouveau MATCHCODE pour une transaction standard SAP .....	50
1.9	USER-EXIT .....	51
1.10	CODIFICATION DES JOBS .....	51
<b>2</b>	<b>EME PARTIE :GESTION DES CLES DE DEVELOPPEMENT :</b> .....	<b>51</b>
<b>3</b>	<b>EME PARTIE :GESTION DES LIVRABLES SAP :</b> .....	<b>52</b>
3.1.	DOCUMENTS PRE-REQUIS POUR UN LIVRABLE .....	53
3.2.	LIVRABLES EXIGES .....	53
<b>4</b>	<b>EME PARTIE : VALIDATION DES PROGRAMMES VIA L'OUTIL CAST</b> .....	<b>54</b>
4.1.	CIRCUIT DE VALIDATION : .....	54
4.2.	SUIVI DE LA QUALITE DES SYSTEMES : .....	56
4.3.	VIOLATIONS IMPLIQUANT UN REJET DU LIVRABLE : .....	57
<b>5.</b>	<b>ANNEXES</b> .....	<b>57</b>
5.1.	ANNEXE A : TABLE DES CODES DE ZONES D'APPLICATION .....	58
5.2.	ANNEXE B : PROGRAMME STANDARD .....	59
5.3.	ANNEXE C : CODES SUFFIXES POUR LES NOMS DES ZONES .....	61
5.4.	ANNEXE D : MOT-CLE COURT, MOYEN, LONG .....	62
5.5.	ANNEXE E : EXEMPLE D'ECRAN .....	63
5.6.	ANNEXE F : PACKAGES OU CLASSES DE DEVELOPPEMENT .....	66
5.7.	ANNEXE G : LISTE DES APPLICATIONS R/3 .....	67
5.8.	ANNEXE H : CAST (ISO ISO 9126-3 ) LISTE DES BONNES PRATIQUES EN ABAP) .....	67

## 1. preambule

Ce document s'inscrit dans le cadre du PQP/CCDG. Il détaille le plan d'ensemble des Normes et Standards applicables aux projets suivis au sein du CCDG-APHP et couvrants en particulier :

- les travaux de développements ABAP,
- la mise en œuvre de l'outil retenu pour le contrôle qualité,
- la réalisation des programmes spécifiques,
- la mise à disposition des livrables dans les environnements du paysage système
- la production documentaire (fonctionnelle, technique, mode opératoire, etc ...)

### **Ses objectifs sont :**

- de fixer les normes de programmation et de codification ABAP/4 à utiliser pour les développements SAP/R3 , on y précise également le profil « développeur SAP » (1ere partie du document).
- Dans la 2<sup>ème</sup> partie : est traitée la gestion des clés de développement
- Dans la 3<sup>ème</sup> partie est traitée la gestion des livrables
- Dans la 4<sup>ème</sup> partie du document, seront traitées les règles et processus de validation des développements spécifiques.

### **En résumé dans ce document vous trouverez :**

- les normes de codification des objets SAP,
- les normes de codification ABAP/4,
- les normes de programmation des Reports et des Modules ABAP/4,
- les normes des commentaires de programmation.
- Les règles de validation d'un livrable SAP
- Les règles de rejet d'un livrable
- Le circuit de validation avec l'outil CAST.

La mise à jour de ce document est à la charge de l'équipe de développement du CCDG.

## 1ere Partie :Normes de Développements SAP

### 1.1 Normes de codification des Objets SAP

#### 1.1.1 Généralités

Le but de ce chapitre est de définir les conventions de notation pour les types d'objets SAP suivants

Programmes, Fonctions, Includes et Reports,

- Pool de Modules, Includes de Pool de Modules,
- Groupes de fonctions, Modules,
- Codes Transaction,
- Packages (Classes de développement),
- Bases de données logiques (LDB),
- Objets Standards SAP,
- Éléments du dictionnaire de données.

#### 1.1.2 Programme, Fonction, Include et Report

##### 1.1.2.1 Programme, Fonction, Report

La codification du nom d'un programme "principal" est sur 30 caractères maximum.

##### 1.1.2.1.1 Cas de la copie d'un programme standard

Dans ce cas la règle est de changer uniquement le premier caractère du programme et de le changer par Z.

##### Exemple : (le programme commence par SAP)

la copie du programme SAPMF45A devient ZSAPxZDOMMF45A ou x est le type du programme.

En cas de plusieurs copies, pour des états différents, on incrémentera de 1 les deux dernières positions.

##### Exemple : (le programme ne commence pas par SAP)

la copie du programme RLISTE00 devient ZDOMRLISTE01.

##### 1.1.2.1.2 Cas d'un programme nouveau

Position	Code et signification
1	<b>Y</b> Outils et Utilitaires non destinés à la production (objets locaux = non transportables \$TMP) <b>Z</b> Programmes destinés à la production

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	7/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	------

2 à 4	Code Domaine/Activité = 'DOM' 'DOM' peut prendre les valeurs suivantes selon le rattachement du besoin à un domaine : DOM = { FIN = domaines finance LOG = domaine Logistique HAB = Habilitations BI_ = Domaine Décisionnel DEP = sous-domaine dépense REC = sous-domaine recette SFP = Domaine DSFP (TG/APHP) UTI = Programme utilitaires TRV = Programmes transverse (inter-domaines) LOC = programmes locaux....etc }
5	'Objectif' du programme : <b>R</b> Programmes de reprise de données <b>I</b> Programmes d'Interface <b>n</b> Programmes 'utilisateur' où <i>n</i> est compris entre 0 et 9. <b>B</b> <i>Batch-Input</i> <b>L</b> <i>liste, reporting</i>
6 à 7	<i>xx</i> où <i>xx</i> est un code significatif du domaine métier (sous-ensemble des modules fonctionnels SAP) auquel est destiné le programme ou le code famille société pour tenir compte des spécificités locales.(cf. Annexe G)
8	Zone du code d'application SAP (cf. Annexe A)
9 à 10	<i>Rajout de deux caractères à incrémenter en dernière position</i>

### Exemples :

Z DOM R MM M 01 = Programme de reprise spécifique de l'application. 'MM' destinée aux Achats.

#### 1.1.2.2 Include

La codification du nom d'un programme "**Include**" est sur 30 caractères maximum avec les 8 premiers caractères pouvant suivre la structure définie ci-après.

##### 1.1.2.2.1 Cas de la copie d'un include standard

Dans ce cas la règle est d'ajouter un ZDOM à celui du nom du fichier « include » en tronquant éventuellement le 8ème caractère existant.

##### 1.1.2.2.2 Cas d'un programme nouveau

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	8/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	------



Position	Code et signification
1 à 8	reprend les 8 premiers caractères du (ou des) programme (s) principal (aux) auquel il est lié.
6-30	<b>TOP</b> Déclaration de données de tables, ... <b>Fnn</b> sous - programmes où nn est compris entre 00 et 99.

### Exemples :

PGM 'Principal'	INCLUDES
ZDOMRMMM 01	ZDOMRMMM TOP, ZDOMRMMM F01, ZDOMRMMM F02, ...

### 1.1.3 Pool de modules et include de pool de modules

La codification du nom d'un pool de modules est sur 30 caractères maximum avec les 8 premiers caractères pouvant suivre la structure définie ci-après:

Position	Code et signification
1-3	<b>SAP</b>
4	Type du pool: <b>M</b> Ecran <b>D</b> Dialogue <b>U</b> Update <b>F</b> Sous-programme <b>L</b> Fonction
5	<b>Z</b> Indique un module défini par l'utilisateur
6-8	Code Activité/Domaine = 'DOM'
9-10	N'importe quel nombre de 01 à 99
11-30	Zone du code d'application SAP (cf. Annexe A)

La codification du nom d'un include de pool de modules est basée sur celle du pool c'est-à-dire sur 30 caractères maximum avec les 8 premiers caractères pouvant suivre la structure définie ci-après:

Position	Code et signification
1	Type du pool: <b>M</b> Ecran <b>D</b> Dialogue <b>U</b> Update <b>F</b> Sous-programme <b>L</b> Fonction
2	<b>Z</b> Indique un module défini par l'utilisateur
3-5	Code Activité/Domaine = 'DOM'
6-7	N'importe quel nombre de 01 à 99

8	Zone du code d'application SAP (cf. Annexe A)
9-30	N'importe quels caractères (A-Z, 0-9)

**Exemple :**

Nom du programme	SAPMZDOM01M
Données Include	MZDOM01M001

### 1.1.4 Groupes de fonctions

La codification d'un groupe de fonctions est la suivante (sur 26 positions) :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5	Zone du code d'application SAP (cf. Annexe A)
6-26	23 caractères (A-Z, 0-9) : les plus explicites possible

Un groupe de fonctions doit contenir des modules de fonction et n'est pas globale.

### 1.1.5 Module de fonctions

La codification d'un module de fonctions (sous-programme) est la suivante (sur 30 caractères) :

Position	Code et signification
1	<b>Z</b>
2	_ caractère soulignement
3-5	Code Activité/Domaine = 'DOM'
6	Zone du code d'application SAP (cf. Annexe A)
7-30	27 caractères (A-Z, 0-9) : Mots séparés par des caractères de soulignement les plus explicites possible

### 1.1.6 Codes Transaction

La codification d'un code Transaction (TCODE) est la suivante (sur 20 caractères)  
et pourra suivre la codification suivante

Position	Code et signification
1	<b>Y</b> Transactions non destinées à la production (objets locaux) <b>Z</b> Transactions destinées à la production
2-4	Code Activité/Domaine = 'DOM'
5	Zone du code d'application SAP (cf. Annexe A)

6-20	<p>15 caractères possibles permettant d'identifier la transaction. Dans la mesure du possible, essayer de respecter l'ancienne norme SAP.:</p> <p>01 = Création 02 = Modification 03 = Visualisation</p> <p>Remarque : Dans le cas particulier de transactions spécifiques les positions 3 à 20 pourront être codifiées en caractères alphanumériques, exemple : ZART_CREATION.</p>
------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Exemple : ZDOMV01

### 1.1.7 Procédures CATT

CATT est un outil de développement permettant de constituer des jeux de test en automatique.  
Nom de procédure : 30 caractères possibles.

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-10	6 caractères libres (préconisés)

Exemple : ZDOMTEST01

### 1.1.8 Classes de Développement

La codification d'une classe de développement (DEVC stocké dans la table TDEVC) est codifiée sur 20 caractères, et pourra suivre la codification suivante :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	3 caractères significatifs du domaine fonctionnel de destination du programme. (cf. annexe F) pouvant s'étendre sur 18 caractères.

### 1.1.9 Bases de données logiques

Les bases de données logiques SAP standard (LDB) sont nombreuses dans SAP (accès via une commande « get »).

Si une nouvelle base de données logique est nécessaire, la codification peut être la suivante (sur 20 caractères) :

Nom. : ME-NDV.doc	Émetteur : Aimé Badrane	Version : 1.2	Date : 28/04/2025	Page: 11/71
-------------------	-------------------------	---------------	-------------------	-------------

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5- 6	Zone du code d'application SAP (cf. Annexe A)

**Exemple :**

ZFINSD : base de données logique spécifique concernant le module vente

### 1.1.10 Objets SAP Standards

Le but de cette procédure est de s'assurer que les sources fournis par SAP demeurent intacts et d'identifier les programmes modifiés lors des nouvelles versions.

- **Les objets fournis par SAP ne doivent pas être modifiés.**
- Si un objet standard SAP doit être modifié, celui-ci doit être copié et renommé. L'objet copié peut être utilisé dans le programme à la place de l'ancien.
- La règle générale consiste à recopier l'objet SAP en le préfixant par ZDOM.
- Font exception à cette règle :
  - les POOL qui commencent toujours par SAP suivi du type de pool (M, D,U, F, L) ; insérer ZDOM entre le 4<sup>ème</sup> et le 5<sup>ème</sup> caractère du nom standard.
  - les INCLUDE qui commencent toujours par le type de pool (M, D,U, F, L) ; insérer ZDOM entre le 1<sup>er</sup> et le 2<sup>ème</sup> caractère du nom standard.
- Si une modification est demandée dans une transaction standard SAP, il y a lieu d'utiliser le USER-EXIT existant prévu spécifiquement pour cette modification ou de faire une demande de Clé de réparation à SAP via l'équipe de Développement IDS.
- Lors de la copie d'un objet standard SAP en Z, indiquer le code 'demande de développement' sur toutes les lignes ajoutées et / ou modifiées :  
Move kna1-kunnr to t\_kna1-kunnr. 'C019

**Exemple :**

Programme d'édition de sortie de marchandises PPRWARS et formulaires associés PSFC\_GOOD-ISS-SL

On les recopie et on les renomme sous les noms : **ZDOMPPRWARS** et **ZDOMPSFC\_GOOD-ISS-SL**.

### 1.1.11 Autres objets SAP

**-Workflow** (technologie orientée objet):

la création ou la copie d'objets utilisés pour un workflow est codifiés sur 10 caractères, les premiers caractères devant avoir obligatoirement la valeur 'ZDOM'.

La codification d'un workflow spécifique copié ou créé est codifiée sur 12 caractères, les premiers caractères devant avoir obligatoirement la valeur 'ZDOM'.

**-Bapi(s)** (technologie orientée objet):

la création ou la copie d'objets utilisés pour un BAPI est codifiés sur 10 caractères, les premiers caractères devant avoir obligatoirement la valeur 'ZDOM'.

La codification d'une fonction API associée au Bapi (copiée ou créée) est codifiée sur 30 caractères, les premiers caractères devant avoir obligatoirement la valeur 'ZDOM'.

**-Idocs ou EDI**

la création ou la copie d'un IDOC est codifiée sur 26 caractères, les premiers caractères devant avoir obligatoirement la valeur 'ZDOM'.

la création ou la copie d'un segment d'IDOC est codifiée sur 26 caractères, les premier caractères devant avoir obligatoirement la valeur 'ZDOM'.

**-Evènements**

Tout événement déclenchant l'exécution d'un programme sera codifié sur 16 caractères et reprendra la codification de ce même programme en incluant '\_EVT\_' en 5ème position.  
Ex : prog ZDOMCODPRO, événement ZDOM\_EVT\_CODPRO

## 1.1.12 Dictionnaire de données

### 1.1.12.1 Table

Les tables utilisateurs doivent être maintenues uniquement par des programmes interactifs. Il est obligatoire de les définir en tant que table transparente. Des petites tables peuvent être définies en tant que table pool. (table «ATAB»).

#### 1.1.12.1.1 Table maintenue ou gérée en SM30 / SM31

La codification d'une table maintenue par la transaction SM30 ou SM31 est la suivante (sur 10 caractères si possible sinon se limiter à 3 caractères après le 10<sup>ème</sup>).

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	3 caractères (mot clé court, cf. Annexe D)
8	1 caractères (A-Z, 0-9) : code libre....

**Exemple : ZDOMVEN1** : table pour le module vente

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	13/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------



#### 1.1.12.1.2 Table maintenue par programme

La codification d'une table maintenue (gérée) uniquement par programmation **ainsi que les tables de transcodification** peut être la suivante (idem tables SM30 / SM31) :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	3 caractères (mot clé moyen, cf. Annexe D)
8-10	3 caractères (A-Z, 0-9) : code libre

**Exemple** : ZDOMVEN001 : table pour le module vente

#### 1.1.12.1.3 Table de transcodification

La codification d'une table **de transcodification** peut être la suivante (idem tables SM31) :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	'TRC' (mot clé moyen, cf. Annexe D)
8-10	3 caractères (000 à 999)

**Exemple** : ZDOMTRC001 : table pour la 1<sup>ère</sup> table de transcodification.

#### 1.1.12.2 Index de table

L'index de table est un objet permettant d'accéder à une table selon un critère clé particulier.

Position	Code et signification
1	<b>Z</b>
2-3	2 caractères libres

#### 1.1.12.3 Vue

Une vue est une vision des données qui est constituée par une ou plusieurs tables.

La codification d'une vue basée sur une table spécifique maintenue par la transaction SM31 est la suivante (sur 6 caractères) :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	3 caractères (mot clé court, cf. Annexe D)
8-9	1 caractères (A-Z, 0-9) : code libre....
10	<b>V</b> Vue

La codification d'une vue basée sur une table spécifique maintenue par programmation est la suivante :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-7	3 caractères (mot clé moyen, cf. Annexe D)
8-10	3 caractères (A-Z, 0-9) : code libre
11	<b>V</b> Vue

La codification d'une vue basée sur une table standard SAP est la suivante :

Position	
1	<b>Z</b>
2-4	DOM
2-n	Nom de la table SAP
n+1	<b>V</b> Vue

#### 1.1.12.4 Structure (work-area)

Les structures sont identiques aux tables mais aucun enregistrement ne peut y être stocké.

La codification d'une structure pourra être la suivante (sur 30 caractères) :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5	zone du code d'application SAP (cf. ANNEXE A)
6	_ caractère soulignement
7-10	4 caractères (A-Z, 0-9) : code transaction ou classe de développement, .....
11	_ caractère soulignement
12-30	19 caractères (A-Z, 0-9) : les plus explicites possible pour la structure

#### Exemple :

ZDOMM\_MM03\_0001 : structure 0001 pour la transaction MM03 de Material Management

#### 1.1.12.5 Domaine

Utiliser en priorité les domaines standards.

Le domaine définit les caractéristiques d'un champ. Il décrit l'intervalle de valeurs d'une zone.

La codification d'un domaine est la suivante (sur 30 caractères) :

Nom. : ME-NDV.doc	Émetteur : Aimé Badrane	Version : 1.2	Date : 28/04/2025	Page: 16/71
-------------------	-------------------------	---------------	-------------------	-------------



Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-10	6 caractères libres

**Exemple :**

ZDOMADRNB          Adresse de type Number.

### **1.1.12.6 Elément de données**

L'élément de données décrit le rôle d'un domaine dans un contexte particulier (30 caractères MAX).

La codification d'un élément de données peut être faite sur 10 caractères de la manière suivante :

Position	Code et signification
1	<b>Z</b> Imposé par SAP
2-4	Code Activité/Domaine = 'DOM'
5-10	Nom du champ sur 6 caractères

**Exemple :**

ZDOMADRESS          Adresse

### **1.1.12.7 Zone**

Le nom d'une zone (champ de table) doit être parlant et ne doit pas excéder 6 caractères.

Position	Code et signification
1-6	<b>6 caractères laissés libres</b>

**Exemple :**

ADRESS                  Adresse

## **1.1.13 Sécurité Applicative :**

### **1.1.13.1 Objets d'autorisation**

L'implantation et la configuration du système SAP ECC a pour but d'intégrer et d'informatiser au maximum les processus d'affaires de même que les données nécessaires au bon fonctionnement de l'entreprise. En centralisant ainsi les données, elles deviennent accessibles à tout agent possédant un accès au système. Afin de faciliter le développement des autorisations, SAP intègre un système de sécurité. Ce système permet de définir des droits d'accès personnalisés pour chaque utilisateur en lui assignant des autorisations différentes. Ces autorisations donneront aux utilisateurs les accès nécessaires et suffisants afin d'effectuer leurs tâches quotidiennes tout en protégeant les données et les transactions envers une utilisation non souhaitée. Le développement et la mise en place d'un système d'autorisation permettront de protéger l'intégrité et la confidentialité des données. Cependant, le temps et les ressources nécessaires pour la mise en place des autorisations sont systématiquement sous-évalués par les analystes d'affaires causant des retards de livraison du système ou une utilisation risquée sur une période de temps indéterminée.

Alors, étant donné l'importance de sécuriser les applications nous insistons sur les différents concepts que doit maîtriser un développeur afin d'élaborer les autorisations dans SAP. Il sera ainsi plus facile pour le chef de projet métier de prendre de meilleures décisions de gestion s'il comprend les concepts et le fonctionnement de base du système d'autorisation :

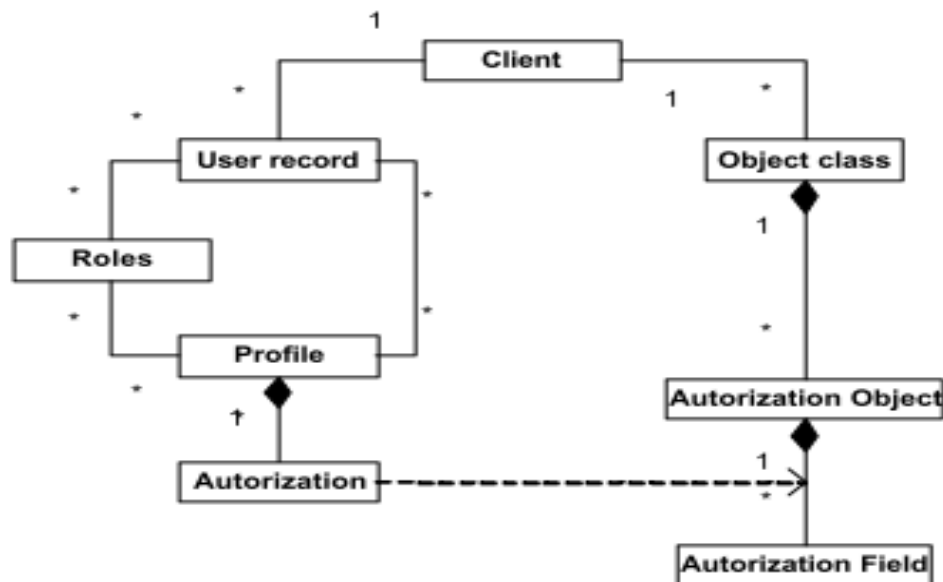
#### ***1.1.13.2 Concept d'autorisation de SAP***

Le système de sécurité de SAP R/3 est basé sur l'attribution aux usagers de groupe d'autorisations permettant d'effectuer différentes transactions dans le système. Une transaction correspond à l'appel de la méthode d'un objet d'affaires du système, et par conséquent l'exécution d'un programme ABAP dont le dessein est de réaliser un certain traitement.

#### **Par exemple :**

la transaction FB50 appelle la méthode *BAPI\_ACC\_GL\_POSTING\_POST* de l'objet *AcctngGLPosting* qui permet de poster une transaction financière dans le grand livre général. Et bien que cette transaction corresponde au programme ABAP *SAPMF05A*, un agent comptable devrait normalement avoir l'autorisation d'exécuter cette transaction dans le cadre de sa fonction.

La figure suivante présente un schéma intégrant les différentes composantes de ce système d'autorisation de SAP.

**Figure 1 : Schéma UML du concept d'autorisation de SAP**

À droite de la figure : on retrouve les éléments «statiques » du système d'autorisation : Object Class, Authorization Class et Authorization Fields. Il s'agit d'une hiérarchie par module des différents objets et champs avec lesquels il est possible de bâtir des autorisations d'accès. Ces dernières ne peuvent donc pas être modifiées et bien qu'il soit possible d'en créer de nouvelles, le système en offre une grande variété permettant de combler la majorité des besoins en sécurité.

À gauche de la figure : on retrouve la partie dynamique: Rôles, profiles et autorisations. Il s'agit des composantes permettant de développer des combinaisons d'autorisations.

### 1.1.13.3 Contrôle des autorisations (*Authorization check*)

Lors d'une ouverture de session, les préférences ainsi que les valeurs d'autorisation qui sont assignées à l'utilisateur sont chargées dans la mémoire tampon de la couche application. Le contrôle des autorisations se déroule en deux parties. Un premier contrôle s'effectue chaque fois que l'utilisateur tente d'accéder à une transaction du système. Lorsque l'accès à cette transaction aura été autorisé, un second contrôle sera effectué afin de s'assurer que les actions posées par l'utilisateur (saisies de données, visualisation de données, sélection d'éléments organisationnels,...) sont conformes avec les valeurs des champs d'autorisations qui ont été définis.

Il existe ainsi deux résultats possibles à une action de l'utilisateur :

- 1) La transaction s'exécute au complet et l'utilisateur reçoit une confirmation du système ou
- 2) aucun élément de la transaction ne passe et l'utilisateur reçoit un message d'erreur du système.

### 1.1.14 Formulaires

Si un formulaire (smartform) doit être modifié, copier celui-ci en le préfixant par ZDOM.  
Le programme d'impression est un programme ABAP IV.

### 1.1.15 Correspondances

La codification d'un type de correspondance est la suivante (sur 4 caractères) :

Position	Code et signification
1	<b>Z</b>
2 à 4	xxx où xxx est une chaîne de caractère libre

Exemples :

ZEXP = Correspondance d'extraction Interface des Expéditions.

### 1.1.16 Dossiers Batch input

La codification d'un dossier Batch Input est la suivante (sur **12** caractères) :

Position	Code et Signification
1	<b>Z</b>
2 à 4	Code Activité/Domaine = 'DOM'
5 à 12	Zone libre à définir dans le projet en fonction des niveaux d'autorisations.

#### Remarques :

- dans le cas d'un batch-input multi-transactions, on prendra la famille de transaction principale (FB, VA,...) en y ajoutant le code 00.
- Afin de conserver une trace du dossier batch-input (en maintenance), adopter l'option « KEEP » :  
call function 'BCD\_OPEN\_GROUP'  
exporting  
    client = sy\_mandt  
    group = 'nom\_du\_dossier\_BTCI'  
    user=sy\_uname  
    keep = 'X'.

## 1.2 Normes de codification ABAP /4

Le but de ce chapitre est de définir les attributs et/ou les conventions de notation pour les paramètres et objets associés au langage de programmation ABAP/4 suivants :

- ✓ Attributs des programmes
- ✓ Structures internes
- ✓ Noms de variable
- ✓ Messages d'erreur
- ✓ PF status
- ✓ Fichiers externes
- ✓ Variantes

### 1.2.1 Attributs des programmes

Le but de cette section est de définir les attributs autorisés pour les programmes créés dans l'environnement SAP.

Les attributs possibles lors de la création d'un programme sont :

Type	Programme OnLine, analyse base de données logique sous forme de listes <b>F</b> Groupe Fonction <b>I</b> Module Include <b>M</b> Pool de module <b>S</b> Pool de sous-routines <b>V</b> Report (mise à jour)
Status	<b>K</b> Programmes développés pour la production <b>T</b> Programmes tests (pas en production)
Application	Code d'application SAP (cf. Annexe A)
Classe de développement	Suivant affectation du programme
Groupe d'autorisation	Blanc sauf si autorisation spécifique
Opération en virgule fixe	Cet attribut doit être marqué sauf si on travaille exclusivement avec des nombres entiers.

Le titre du programme doit être fonctionnel, compréhensible par l'utilisateur.

#### Exemple :

Liste des comptes clients créés

## 1.2.2 Codification des noms de variables

Une variable de travail est à usage général. Ce sont tous les types de variables qui ne rentrent pas dans les catégories définies ci-dessus. Ces variables peuvent contenir des valeurs alphanumériques, numériques, packées, binaires, de type date ou heure, hexadécimales.

### 1.2.2.1 Types de variables

TYPE DE VARIABLE	PREFIXE	SIGNIFICATION
Variable "Travail"	L ou W	<u>L</u> ocal ou <u>W</u> orld
Variable "Transfert interne"	P	<u>P</u> aramètre procédure
Variable "Transfert externe"	M	<u>M</u> emory
Variable "Dynpro simple"	U	<u>U</u> ser input
Variable "Dynpro select-option"	S	<u>S</u> elect-option user input
Variable "Pointeur"	PTR	<u>P</u> oin <u>T</u> e <u>R</u>
Constante	K	<u>K</u> onstant

### 1.2.2.2 Codification des variables "travail"

Définition : c'est une variable interne à un programme, non visible par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution du programme. Elle sert à stocker une donnée temporaire qui est soit visible dans tout le module où elles est définie, il s'agit alors d'une variable globale (World), soit visible uniquement dans une procédure ou fonction, il s'agit alors d'une variable locale (Local).

**La première lettre identifie la portée de la variable :**

Identificateur	Signification
L	portée locale ( <u>L</u> ocal) à une procédure ou fonction
W	portée globale ( <u>W</u> orld) à un module

**La seconde lettre identifie le type de variable :**

Identificateur	Signification
T	variable de type <u>T</u> able
S	variable de type <u>S</u> tructure
C	variable de type <u>C</u> aractère
N	variable de type <u>N</u> umérique
I	variable de type <u>I</u> nteger
P	variable de type <u>P</u> aquet
F	variable de type <u>F</u> loating-point
D	variable de type <u>D</u> ate
H	variable de type <u>H</u> eure

B	variable de type <b>B</b> inaire
X	variable de type <b>H</b> exadécimal
Z	variable de type non précédemment défini

**La troisième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**N caractères maximum pour le nom de la variable**

Identificateur	Signification
N caractères	Nom significatif du contenu de la variable

**Suffixe éventuel (underscore + deux caractères)**

Identificateur	Signification
_IX	Variable de travail utilisée comme <b>I</b> nde <b>X</b>
_FG	Variable de travail utilisée comme <b>F</b> la <b>G</b>

**Exemples**

LT_XXXXX	Table locale à une procédure ou fonction
LS_XXXXX	Structure locale à une procédure ou fonction
LC_XXXXX	Caractère ou chaîne de caractères locale à une procédure ou fonction
LC_XXXXX_FG	Caractère local utilisé comme flag (booléen) dans procédure ou fonction
LN_XXXXX	Numérique ou chaîne de numériques locale à une procédure ou fonction
LI_XXXXX	Entier local à une procédure ou fonction
LI_XXXXX_IX	Entier local utilisé comme index dans une procédure ou fonction
WT_XXXXX	Table globale à un module
WS_XXXXX	Structure globale à un module
WC_XXXXX	Caractère ou chaîne de caractères globale à un module
WC_XXXXX_FG	Caractère global utilisé comme flag (booléen) dans un module
WN_XXXXX	Numérique ou chaîne de numériques globale à un module
WI_XXXXX	Entier global à un module
WI_XXXXX_IX	Entier global utilisé comme index dans un module

### 1.2.2.3 Codification des variables "transfert interne"

Définition : c'est une variable interne à une procédure ou une fonction, non visible par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution de la procédure. Elle sert à importer une donnée temporaire dans une procédure ou fonction.

**La première lettre identifie ce type de variable :**

Identificateur	Signification
P	Variable paramètre importée dans la procédure

**La deuxième lettre identifie le type de transfert**

Identificateur	Signification
V	Transfert par <u>V</u> aleur
R	Transfert par <u>R</u> éférence

**La troisième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**La quatrième lettre identifie le type de variable**

(Voir codification des variables "travail")

**La cinquième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**N caractères maximum pour le nom de la variable**

Identificateur	Signification
N caractères	Nom significatif du contenu de la variable

**Suffixe éventuel**

(Voir codification des variables "travail")

**Exemples**

PV_I_XXXXX	Paramètre entier transmis par valeur
PV_I_XXXXX_IX	Paramètre entier transmis par valeur pour utilisation de type index
PV_C_XXXXX	Paramètre caractère ou chaîne de caractère transmis par valeur
PV_C_XXXXX_FG	Paramètre caractère ou chaîne de caractère transmis par valeur comme flag
PV_D_XXXXX	Paramètre date transmis par valeur



PR_I_XXXXX	Paramètre entier transmis par référence
PR_I_XXXXX_IX	Paramètre entier transmis par référence pour utilisation de type index
PR_C_XXXXX	Paramètre caractère ou chaîne de caractère transmis par référence
PR_C_XXXXX_FG	Paramètre caractère ou chaîne de caractère transmis par référence comme flag
PR_D_XXXXX	Paramètre date transmis par référence

#### 1.2.2.4 Codification des variables "transfert externe"

Définition : c'est une variable interne de communication entre process, non visible par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution d'une session. Elle sert à exporter ou importer une donnée temporaire entre différents programmes d'une même session.

**La première lettre identifie ce type de variable :**

Identificateur	Signification
M	Variable de type <u>M</u> emory

**La deuxième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**La troisième lettre identifie le type de variable**

(Voir codification des variables "travail")

**La quatrième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**N caractères maximum pour le nom de la variable**

Identificateur	Signification
N caractères	Nom significatif du contenu de la variable

**Suffixe éventuel**

(Voir codification des variables "travail")

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	25/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

## Exemples

MV_I_XXXXX	Donnée inter-process entière.
MV_I_XXXXX_IX	Donnée inter-process entière pour utilisation de type index
MV_T_XXXXX	Donnée inter-process de type table
MV_D_XXXXX	Donnée inter-process date.

### 1.2.2.5 Codification des variables "dynpro simple".

Définition : c'est une variable interne à un programme, visible et contrôlée par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution du programme. Elle sert d'interface entre l'utilisateur final et le programme. Elle ne permet l'entrée que d'une seule valeur.

**La première lettre identifie ce type de variable :**

Identificateur	Signification
U	Variable de type <u>U</u> ser input

**La deuxième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**Six caractères pour le nom de la variable**

Identificateur	Signification
Six caractères	Nom significatif du contenu de la variable

## Exemples

U_XXXXX	Entrée utilisateur (dynpro)
---------	-----------------------------

### 1.2.2.6 Codification des variables "dynpro select-option".

L'instruction SELECT-OPTIONS <nom de zone> For <zone de table> génère une variable de programme utilisée pour un critère de sélection. Ceci permet à l'utilisateur de saisir un intervalle de valeurs sur un écran de sélection. La variable générée est une table système contenant <variable>-highvalue, <variable>-lowvalue, <variable>-sign et <variable>-option pour la zone de la table sélectionnée. Le nom de ces variables est limité à 8 caractères.

**La première lettre identifie ce type de variable :**

Identificateur	Signification
S	Variable de type <u>S</u> elect-option

**La deuxième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**Six caractères pour le nom de la variable**

Identificateur	Signification
Six caractères	Nom significatif du contenu de la variable

**Exemples**

S_XXXXX	Entrée utilisateur (dynpro)
---------	-----------------------------

#### **1.2.2.7 Codification des variables "pointeur"**

Définition : c'est une variable interne à une procédure, non visible par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution de la procédure. Elle est généralement utilisée pour générer et/ou contrôler des chaînes de caractères. Elle correspond au field-symbol de l'ABAP.

**Les trois premières lettres identifient ce type de variable :**

Identificateur	Signification
PTR	Variable de type <u>P</u> oin <u>T</u> eu <u>R</u>

**La quatrième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**La cinquième lettre identifie la portée du pointeur**

Identificateur	Signification
L	portée locale ( <u>L</u> ocal) à une procédure ou fonction
W	portée globale ( <u>W</u> orld) à un module

**La sixième lettre identifie le type de variable**

Nom. : ME-NDV.doc	Émetteur : Aimé Badrane	Version : 1.2	Date : 28/04/2025	Page: 27/71
-------------------	-------------------------	---------------	-------------------	-------------

(Voir codification des variables "travail")

### La septième lettre est un séparateur

Identificateur	Signification
_	caractère ASCII underscore

### N caractères maximum pour le nom de la variable

Identificateur	Signification
N caractères	Nom significatif du contenu de la variable

### Suffixe éventuel

(Voir codification des variables "travail")

### Exemples

PTR_LT_XXXXX	Pointeur sur Table locale à une procédure ou fonction
PTR_LS_XXXXX	Pointeur sur Structure locale à une procédure ou fonction
PTR_LC_XXXXX	Pointeur sur Caractère ou chaîne de caractères locale à une procédure ou fonction
PTR_LC_XXXXX_F G	Pointeur sur Caractère local utilisé comme flag (booléen) dans procédure ou fonction
PTR_LN_XXXXX	Pointeur sur Numérique ou chaîne de numériques locale à une procédure ou fonction
PTR_LI_XXXXX	Pointeur sur Entier local à une procédure ou fonction
PTR_LI_XXXXX_IX	Pointeur sur Entier local utilisé comme index dans une procédure ou fonction
PTR_WT_XXXXX	Pointeur sur Table globale à un module
PTR_WS_XXXXX	Pointeur sur Structure globale à un module
PTR_WC_XXXXX	Pointeur sur Caractère ou chaîne de caractères globale à un module
PTR_WC_XXXXX_ FG	Pointeur sur Caractère global utilisé comme flag (booléen) dans un module
PTR_WN_XXXXX	Pointeur sur Numérique ou chaîne de numériques globale à un module
PTR_WI_XXXXX	Pointeur sur Entier global à un module
PTR_WI_XXXXX_I X	Pointeur sur Entier global utilisé comme index dans un module

#### 1.2.2.8 Codification des "constantes"

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	28/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

Définition : c'est une "*variable*" interne à un programme, non visible par l'utilisateur final. Sa durée de vie est limitée au temps d'exécution du programme. Elle sert à définir une valeur qui n'est jamais modifiée. Ce type de donnée doit être défini pour être visible dans tout le module, il s'agit explicitement d'une donnée globale (World).

**La première lettre identifie ce type de variable :**

Identificateur	Signification
K	Constante

**La seconde lettre identifie le type de variable**

(Voir codification des variables "travail")

**La troisième lettre est un séparateur**

Identificateur	Signification
_	caractère ASCII underscore

**N caractères maximum pour le nom de la "*variable*"**

Identificateur	Signification
N caractères	Nom significatif du contenu de la variable

**Suffixe éventuel**

(Voir codification des variables "travail")

**Exemples**

KT_XXXXX	Table (données) globale à un module
KS_XXXXX	Structure (données) globale à un module
KC_XXXXX	Caractère ou chaîne de caractères (données) globale à un module
KC_XXXXX_FG	Caractère (donnée) global utilisé comme flag (booléen) dans un module
KN_XXXXX	Numérique ou chaîne de numériques (données) globale à un module
KI_XXXXX	Entier (donnée) global à un module
KI_XXXXX_IX	Entier (donnée) global utilisé comme index dans un module

### 1.2.3 Codification des noms de tables

Le but de cette section est de définir les codifications pour les structures internes de type Table interne et Table système.

Toute table interne doit être définie par référence à une unique structure qui lui est propre. Il n'y a de ce fait qu'une unique possibilité pour déclarer une table interne donnée :

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	29/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

```
DATA : BEGIN OF WT_TABLE_TRAV1 OCCURS 0.
      INCLUDE STRUCTURE WS_TABLE_TRAV1.
DATA : END OF WT_TABLE_TRAV1.
```

La structure de référence doit bien évidemment avoir été préalablement définie.

```
DATA : BEGIN OF WS_TABLE_TRAV1.
...
DATA : END OF WS_TABLE_TRAV1.
```

Cette méthode de codage permet, en phase de maintenance corrective, la création aisée de variables de tests pouvant contenir les données de cette table.

```
DATA : LS_ TABLE_TRAV1 LIKE WS_TABLE_TRAV1.
```

Elle permet également un suivi facile de l'évolution des valeurs traitées dans un loop.

```
CLEAR WS_TABLE_ TRAV1.
LOOP AT WT_TABLE_ TRAV1.
  IF NOT WS_TABLE_ TRAV1 IS INITIAL.
    IF WS_TABLE_ TRAV1-XXXX = WT_TABLE_ TRAV1-XXXX.
      .....
    ENDIF.
  ENDIF.
  .....
  MOVE WT_TABLE_ TRAV1 TO WS_TABLE_ TRAV1.
ENDLOOP.
```

### 1.2.4 Message d'erreur spécifiques

Le but de ce paragraphe est de définir la codification des messages d'erreur spécifiques. (Toutefois les messages d'erreur standard SAP devront être utilisés dès que possible pour éviter toute redondance.)

Si un message doit tout de même être ajouté, il faut le définir de la manière décrite ci-dessous en utilisant trois paramètres : identifiant du message, sévérité du message et numéro du message.

Les messages SAP sont dans la table T100 et sont classés par domaine fonctionnel.

#### Identifiant du message :

L'identifiant ou classe de message possède 20 caractères dont la codification pourra être la suivante :

Position	Code et signification
1	<b>Z</b> Imposé par SAP
2 à 4	DOM

5 à 10

Zone du code d'application SAP (cf. Annexe A) 6c

### Exemple :

ZDOMV spécifique vente

### Sévérité :

La sévérité du message définit les actions effectuées après l'exécution du message.

Les différents codes sont les suivants:

A	"Abend"	la transaction s'arrête et l'utilisateur ne peut plus continuer
E	"Error"	la donnée saisie est invalide, l'utilisateur doit la modifier.
W	"Warning"	le programme donne la possibilité à l'utilisateur de passer outre ou de modifier la donnée.
I	"Informational"	affiche une information. L'utilisateur doit presser Enter pour continuer.
S	"Success"	affiche une information sur l'écran suivant.

### Numéro :

Les numéros de message sont les 3 caractères permettant d'identifier de manière unique un message. Le numéro va de 001 à 999.

### 1.2.5 PF-status

PF-status définit les touches de fonctions pouvant être utilisées dans un écran pour effectuer des actions. Le standard SAP doit être utilisé.

Le tableau ci-dessous définit l'usage standard des touches de fonctions. Ces touches de fonction doivent accomplir les actions définies dans ce tableau et ne doivent pas être remplacées.

Touche	OK-CODE	Description
01	HELP	Affiche l'aide pour la zone courante
02	PICK	Sélectionne (idem double-click)
03	BACK	Quitte la transaction courante sans faire de contrôle
04	LIST	Liste des entrées autorisées
08	EXECUTE	Exécution
10	MENU	Barre de menu
11	SAVE	Sauvegarde les entrées
12	RW	Annule la requête courante sans faire de contrôle.
13	PRI	Imprime
14	DLT	Supprime
15	EXIT	Quitte (idem 03)
21	PP--	Première page du document
22	P-	Page précédente

23	P+	Page suivante
24	PP++	Dernière page du document

### 1.2.6 Fichiers externes

Les fichiers externes doivent être définis comme des fichiers logiques (transaction FILE ou transactions SF01, SF02, SF03) à travers le chemin d'accès du menu de paramétrage. Cela permet d'avoir un nom indépendant du nom physique (DOS, UNIX, ..) du fichier.

### 1.2.7 Variantes

Les variantes sont des objets qui définissent les paramètres d'entrées par défaut et extérieurs au programme et nécessaires pour l'exécution du dit programme.

La table TVARV contient toutes les variables avec une valeur déterminée. Elle peut être mise à jour pour y insérer, par exemple, de nouvelles variables.

Cette table sera donc gérée par une transaction ou un batch spécifique.

Le nom possède 14 caractères au maximum dont la codification est la suivante :

les 10 premiers caractères représentent le code programme utilisant la variante

Position	Code et signification
1	Z
2 à 4	DOM
5 à 10	6 caractères
11	Séparateur ' _ '
12	V comme variante
13 à 14	Numéro de variante

#### Exemple :

ZDOMCODPRO\_V01

#### Remarque :

Le libellé de la variante est plus important que la codification et doit permettre à l'utilisateur de se déterminer au sein d'une liste de variantes.



### 1.3 Normes de programmation des ReportS ABAP / 4

Le but de ce chapitre est de définir les règles de base pour le développement des programmes. L'ordre des composants dans un programme doit respecter l'ordre décrit dans le squelette du programme défini en annexe B. Ce chapitre décrit les règles autour de l'utilisation des propriétés et des composants ABAP.

#### 1.3.1 Structure du programme

La structure du programme doit suivre celle proposée en Annexe B.

Les événements qui ne sont pas utilisés dans le programme final doivent être enlevés par souci de clarté. Tous les sous-programmes doivent être placés après l'instruction END-OF-SELECTION.

Les commentaires doivent être utilisés dans les conditions suivantes :

- ✓ Avant chaque section principale
- ✓ Avant chaque petite section de code
- ✓ Pour clarifier une partie complexe

Une section est considérée comme principale si c'est un événement ABAP ou si elle réalise une tâche identifiable. Une section est considérée comme petite si elle comprend au plus 20 lignes de codes et réalise une tâche identifiable.

#### 1.3.2 Bloc de documentation principale

Le bloc de documentation contient une description détaillée du programme, l'historique des modifications et les tables utilisées. Un exemple est proposé dans le programme décrit en annexe B.

Liste des données requises dans le bloc de documentation principale :

- ✓ Nom du programme et titre
- ✓ Date et auteur du programme
- ✓ But du programme (texte libre, le plus explicite possible)
- ✓ Liste des paramètres en entrée/sortie
- ✓ Liste des modules en include et des sous-programmes
- ✓ Liste des tables SAP mises à jour avec la précision entrée/sortie
- ✓ Liste des tables externes et fichiers accédés avec la précision entrée/sortie
- ✓ Liste des codes retour du programme
- ✓ Ligne de révision avec numéro de la requête de développement, la date de la modification, le nom du développeur, la description de la modification.

#### 1.3.3 Présentation générale

Il faut utiliser la commande PRETTY-PRINT (PP) pour indenter automatiquement le programme.

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	33/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

### 1.3.4 Utilisation de include

L'utilisation de module include est recommandé pour accroître les performances et la lisibilité. L'utilisation de modules de fonctions pour les routines communes est préférable à une utilisation excessive de module include.

### 1.3.5 Messages

Tous les messages d'erreur doivent être implantés à travers la procédure MESSAGE ID's. Tous les textes de messages doivent utiliser NUMBERED-TEXT, ceci afin de rendre les messages dépendants de la langue utilisée.

### 1.3.6 Reports

Les lignes blanches doivent être définies en utilisant l'instruction SKIP <n>. Il faut utiliser NUMBERED-TEXT pour tous les textes affichés.

### 1.3.7 Batch input data

Les sessions de batch input data doivent être limitées à environ 250 transactions par BDC groupe. Ce nombre peut varier de manière importante en fonction des données traitées. Les gros programmes batch doivent être évités du fait de la nature mono tâche de l'environnement. Ces programmes doivent être scindés en plusieurs petits programmes.

Le nom d'un dossier batch est constitué par le nom du programme suivi d'un numéro séquentiel.

### 1.3.8 Accès aux données

Il est recommandé d'utiliser au maximum les bases de données logiques afin d'améliorer les performances. Les considérations de sélection et d'efficacité doivent être prises en compte lors de l'écriture des programmes. L'utilisation du DBMS SQL est déconseillée dans tous les objets SAP. Des problèmes peuvent apparaître lors des montées de version (nouveaux champs dans les tables).

**Toutes les tables doivent être définies dans le dictionnaire de données afin de valider les types de structure.**

**Les codes de retour doivent être testés. Il vaudra mieux utiliser l'instruction CHECK ou CASE et non des IF multiples pour tester les valeurs.**

### Performances :

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	34/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

## INSTRUCTION SELECT :

**Il est déconseillé d'imbriquer des ordres SELECT (2 niveaux au maximum).**

L'instruction SELECT SINGLE \* FROM .... WHERE ..... permet de faire un accès direct dans la base de données, elle est donc conseillée.

L'instruction SELECT \* UP TO 1 ROWS FROM..... WHERE... est équivalent au code suivant.

**SELECT \* FROM ..... WHERE .....**

**EXIT.**

**ENDSELECT.**

Ce type d'accès n'est pas recommandé car énormément consommateur (La table est entièrement balayée jusqu'au moment où l'occurrence est trouvée).

Remarque : Dans le cas de l'utilisation de bases de données Logiques, il conviendra d'employer les ordres GET.

### Amélioration des temps d'accès :

Une solution réside dans la création d'un match code. Le match code peut être utilisé dans les zones de sélections avec création d'identificateurs (différentes sélections possibles).

Pour améliorer les accès par GET ou SELECT, il est possible de créer sur les tables des index secondaires (au niveau du DATA DICTIONARY).

### Remarques :

Il est nécessaire de contrôler la création des MATCHCODES et des INDEX secondaires.

A chaque modification de la base de données, les tables SAP seront mises à jour puis ce seront les MATCHCODES et les INDEX. C'est donc énormément consommateur en process de MAJ, c'est pour cette raison qu'il faut contrôler la création de ces objets.

### Tests de performance (Trucs et astuces) :

La transaction SE30 permet de vérifier le niveau de performance des instructions Abap.

Dans cette transaction, la fonction Tips and Tricks offre la possibilité de tester les performances des différentes instructions telles que l'ordre SELECT.

### 1.3.9 Mise à jour des données

**Les bases de données SAP doivent seulement être mises à jour à partir de code fourni par SAP. Ceci peut être fait à travers les transactions existantes SAP ou l'utilisation du Batch-Input, du Direct-Input, ou des BAPIs.**

**Les ordres INSERT, UPDATE, DELETE sont interdits sur les tables de données SAP. Ils ne sont autorisés que sur les tables utilisateurs(spécifiques), tables ATAB ou tables internes. Souvent on leur préfère l'utilisation de BAPI qui remplissent la même fonctionnalité.**

Le dialecte SAP-SQL met à votre disposition les fonctions de mise à jour :

INSERT	(Création d'un enregistrement)
UPDATE	(Mise à jour de l'enregistrement)
DELETE.	(Suppression de l'enregistrement)

Ces trois types d'accès sont supportés par un code retour, ce qui veut dire que la zone SY-SUBRC a la valeur 0 si l'accès a réussi et une valeur autre que 0 dans le cas contraire.

**Ces instructions de mise à jour ne doivent être utilisées que sur les tables spécifiques, la mise à jour des tables SAP est interdite avec ces instructions, dans ce dernier cas c'est la technique du Batch-Input, du Direct-Input ou des BAPIs qui devra être utilisée.**

L'instruction MODIFY est également disponible. Elle a pour effet, soit de modifier un enregistrement existant, soit de l'ajouter s'il n'existait pas encore. **Cette instruction est à proscrire, ses performances sont médiocres.**

Détail de l'instruction INSERT : Création d'un enregistrement dans la base de données.

INSERT : Forme courte de SAP.

L'instruction INSERT <table> vous permet d'insérer un nouvel enregistrement à une table de la base de données. Vous disposerez des données requises à cette fin dans la zone de travail.

**Exemple :**

```
TABLES : ZVENT1.
MOVE SY-UNAME TO ZVENT1-UNAME.
MOVE VBAK-VBELN TO ZVENT1-VBELN.
MOVE ....
INSERT ZVENT1.
INSERT : Forme longue SAP.
```

L'instruction INSERT <table> VALUES <workarea> vous permet d'insérer un nouvel enregistrement à la table de la base de données ; les données sont extraites de la zone de travail <workarea>.

Au niveau performances, ces deux formes sont équivalentes.

Détail de l'instruction UPDATE : Mise à jour d'un enregistrement dans la base de données.

UPDATE : Forme courte de SAP.

L'instruction UPDATE<table> vous permet de modifier un enregistrement de la table de la base de données. Les données doivent être présentes dans la zone de travail <table>.

UPDATE : Forme longue SAP.

Si l'instruction UPDATE<table> SET<zones valeurs> WHERE <zones clés> est utilisée, seules seront modifiées les zones des tables figurant après SET. Ne sont d'ailleurs transférés que les données de mise à jour (performances supérieures à la forme courte SAP).

Pour modifier un enregistrement bien précis, vous ferez suivre WHERE de toutes les zones clés.

Au niveau performance, il est recommandé d'utiliser la forme longue plutôt que la forme courte. En effet la forme courte SAP a pour effet d'écraser toutes les zones de l'enregistrement de la table et de leur substituer les contenus de la zone de travail.

En règle générale, avant toute modification dans la base de données, il est préférable de tester si l'enregistrement est présent afin d'utiliser l'instruction adéquate.

**Exemple :**

```
SELECT SINGLE * FROM ZVENT1....
IF SY-SUBRC = 0.
UPDATE ZVENT1 .... SET ..
ELSE.
INSERT ZVENT1.
ENDIF .
```

Autre méthode de mise à jour de la base : ARRAY OPERATIONS

Il vous est possible de modifier la base de données au niveau d'opérations array. Les avantages de cette méthode en termes de performance vous dicteront de la privilégier autant que faire se peut.

Instructions :

```
INSERT <table> from <itab>
UPDATE<table> from <itab>
DELETE <table> from <itab>
MODIFY<table> from <itab>
```

**Exemple :**

```
TABLES : ZVENT1.
DATA : BEGIN OF TAB OCCURS 0.
INCLUDE STRUCTURE Z2VENT1.
DATA : END OF TAB.
MOVE ....
APPEND TAB.
INSERT ZVENT1 FROM TAB.
```

**LOGIQUE DE BLOCAGE SAP :**

Pour prévenir les accès parallèles à des objets de données lors de transactions de modifications, un blocage logique et centralisé pourra affecter tous les utilisateurs.

**Il est conseillé d'apposer la mention de blocage (fonction standard ENQUEUE) avant chaque accès en lecture, puis de lever à nouveau le blocage après l'accès de mise à jour(fonction standard DEQUEUE).**

**1.3.10 Utilisation des fichiers externes**

Les fichiers externes doivent être liés à des noms logiques gérés dans SAP.

**1.3.11 Variantes**

Les variantes doivent être définies comme les paramètres par défaut nécessaires à l'exécution du programme. Tous les programmes batch avec des paramètres ont besoin de variantes pour s'exécuter.

### 1.3.12 Sous-programmes

Il faut utiliser des variables locales dès que possibles dans les sous-programmes pour un souci de modularité. Cependant, des sous-programmes fréquemment utilisés peuvent utiliser des variables globales pour éviter à chaque appel le temps de création de ces variables locales.

Il est recommandé d'utiliser l'instruction USING pour passer des données aux sous-programmes ou en récupérer. Ceci ne s'applique pas aux sous-programmes d'initialisation des zones et aux sous-programmes globaux.

L'utilisation de la clause CHANGING est laissée à l'appréciation du développeur, elle est intéressante pour la documentation des sous-programmes.

Les sous-programmes qui sont appelés par plusieurs programmes doivent être créés comme un module de fonction. Les sous-programmes externes doivent aussi être implémentés comme des modules de fonctions.

### 1.3.13 Autorisations

Les objets d'autorisation doivent être contrôlés sur l'écran de sélection ou dans le code de transaction. Tous les objets de contrôle doivent être documentés précisément et donnés à l'administrateur de la sécurité.

### 1.3.14 Debug

Tous les codes de debug doivent être enlevés du programme avant toute migration dans un environnement contrôlé, c'est à dire Intégration ou Production.

Ceci inclut les points d'arrêt et toute partie de codes servant au test.

### 1.3.15 Instructions spécifiques

#### 1.3.15.1 APPEND TAB

Il faut utiliser la clause SORT uniquement s'il y a moins de 100 lignes.

Si une table interne doit être triée et que la condition ci-dessus n'est pas vérifiée, il faut remplir la table dans un ordre quelconque puis la trier selon les zones spécifiées. Pour supprimer les doubles, si nécessaire, il faut mettre les lignes dans une table auxiliaire à partir de la table triée.

#### 1.3.15.2 ASSIGN

L'utilisation de zones symboliques est déconseillée.

Les zones symboliques doivent être documentées dans les commentaires du programme quand elles sont définies et à chaque fois qu'elles sont utilisées.

---

### **1.3.15.3 AT Pfnn**

Il faut utiliser AT USER COMMAND au lieu de AT Pfnn qui améliore la lisibilité.

### **1.3.15.4 CHECK**

Il faut utiliser l'instruction CHECK dès que possible au lieu de IF imbriqués.

### **1.3.15.5 COMPUTE**

L'instruction COMPUTE doit être utilisée.

### **1.3.15.6 DATA**

Il faut définir les variables avec l'instruction DATA en accord avec la codification définie précédemment. Il faut aligner les définitions des variables pour plus de lisibilité.

Il faut utiliser le paramètre LIKE dans l'instruction DATA dès que possible. Ceci permet de garder une cohérence entre les variables SAP et les variables des programmes.

Bien que ABAP/4 initialise toutes les zones à l'exécution du programme, il est toujours bon d'initialiser les variables en utilisant l'instruction CLEAR. Par contre, il faut systématiquement effectuer un CLEAR des structures de données et des variables avant de les affecter, en particulier dans les boucles. Ceci est particulièrement recommandé dans les modules de fonctions puisque de multiples appels y sont faits.

### **1.3.15.7 CHECK, EXIT, EJECT, STOP**

Il faut utiliser ces instructions pour suspendre un traitement et/ou ne pas le faire si cela n'est pas nécessaire ceci dans le but d'améliorer les performances.

### **1.3.15.8 IF/CASE**

Il faut mettre une seule condition par ligne afin d'améliorer la lisibilité.

Il faut utiliser des parenthèses pour grouper les conditions.

Il faut utiliser l'instruction CASE pour éviter les IF's imbriqués sur plus de 3 niveaux.

### **1.3.15.9 MOVE-CORRESPONDING**

Cette instruction doit être utilisée avec précaution puisque seules les zones avec des noms identiques sont affectées. Pour affecter une structure de données avec les données d'une structure équivalente, il faut utiliser l'instruction <LFA1\_DS> = \* <LFA1>.

### 1.3.15.10 READ TABLE

Lors de la lecture d'une table avec l'option WITH KEY, il faut s'assurer que la clé est remplie. Ceci est plus efficace avec une table binaire. Pour construire ou lire de longues tables internes, il faut utiliser BINARY SEARCH.

### 1.3.15.11 SORT

Il faut toujours spécifier la zone de tri dans une instruction SORT afin d'améliorer la lisibilité et la maintenance même si la valeur par défaut est utilisée.

### 1.3.15.12 COLLECT

L'instruction **COLLECT** pour les tables internes peut être très gourmande en CPU dès que la table interne contient plus de 50 lignes. La solution ci-dessous peut être utilisée en remplacement.

```
READ itab WITH KEY new-key BINARY SEARCH.
CASE SY_SUBRC.
    WHEN 0.                new-amt = new-amt + itab-amt.
                        MOVE...
                        MODIFY itab INDEX SY-INDEX.
    WHEN OTHERS.
                        MOVE ...
                        INSERT itab INDEX SY-INDEX.
ENDCASE.
```

## 1.3.16 Interface utilisateur (GUI)

Les status de l'interface utilisateur doivent être utilisés pour les programmes de report en interactif et les programmes on-line.

Il faut utiliser dès que possible une barre de menu pour disposer d'écrans standards dans un pool de modules.

## 1.3.17 Maintenance / correction d'un code existant en production

### 1.3.17.1 Objet

En cours de réalisation, une demande de Maintenance / correction d'un code existant en production peut être demandée. **Ces demandes doivent être cataloguées et suivies de façon rigoureuse.**

Dès qu'un source déjà en production est modifié (écart ou évolution), le commentaire "**Ligne de Révision**" doit être mis à jour dans le bloc de documentation principal du programme (Se référer au paragraphe sur la description du programme standard).

L'ensemble du code modifié doit être entouré par des lignes de commentaire telles que "Début de " et "Fin de " en précisant la nature de la modification du code (ajout, suppression, modification) et le numéro de correction (Référence de la fiche d'écart ou de la fiche de modification).

Aucune ligne du code ne doit être supprimée physiquement : elle doit être mise en commentaire.



Toute modification du code sur une ligne doit passer par la double opération suivante :

- Mise en commentaire de la ligne à modifier,
- Ajout d'une ligne avec le code modifié.

Si un changement majeur de code intervient, il faut insérer une description complète de l'ensemble des modifications et reprendre le programme. Dans ce cas le programme doit être en accord avec les standards de programmation en vigueur au jour de la modification et non avec ceux en vigueur à la date de création du programme. Ceci permettra d'assurer que le programme retravaillé suit bien les nouveaux standards.

### 1.3.17.2 Création ou modification de programme

**En cas de modification, le développeur devra sauvegarder le programme source avant toute intervention.**

**Exemple :**

ZVRVEN01 sauvegardé sous YVRVEN01.

#### 1.3.17.2.1 Attribution du N° de correction

- Un n° de correction SAP est ouvert à chaque intervention sur un programme. Celui-ci sera « released » (fermé) par un numéro de transport au moment de sa mise en système d'intégration. (Remarque : une modification de programme peut être appliquée si aucune correction n'est ouverte (par un autre développeur ou le développeur lui-même) ; c'est à dire jusqu'au transport en intégration.
- remplir la zone 'désignation synthétique' du numéro de correction comme suit :

Objet SAP/ N° demande / Nom du demandeur

Exemple :

ZVRVEN01 /DE2 97 03 01.1 / Dupont

- Renseigner la description dans 'la documentation' par le n° du programme, par exemple : ZVRVEN01

#### 1.3.17.2.2 Tests Unitaires

Les tests unitaires seront faits sur le mandant de développement mais l'utilisation d'un mandant « bac à sable » pourra également être utilisé sur BE2.

#### 1.3.17.2.3 Tests d'intégration

Les tests en machine d'intégration sont faits par le demandeur fonctionnel.

***Si les tests unitaires sont corrects, le développeur ou le demandeur fonctionnel peut alors demander à l'équipe technique le transport en machine d'intégration.***

### **1.3.17.3 Maintenance / réparation d'un programme standard.**

#### **1.3.17.3.1 Définition**

On appelle réparation, toute modification apportée à un objet SAP ( Programme, data élément , domaine etc..), l'affectation d'un numéro de réparation est effectué de la même manière , c'est indépendant de l'objet.

Le code standard ne peut en règle générale pas être modifié. Il existe cependant deux cas bien précis où la modification de standard est autorisée :

#### **1.3.17.3.2 Ajout de « code » client dans les programmes standards.**

▪ Utilisation du 'USER-EXIT' :

Il est possible d'ajouter au sein des programmes SAP des parties de « code client ». De cette manière, de nouvelles fonctionnalités peuvent être introduites dans les programmes SAP (Désactivation de zones par exemple).

Ces USEREXIT ne sont pas modifiés par SAP lors des montées de version.

▪ Utilisation du FIELD-EXIT' :

Il est également possible de rajouter des parties de « code client » à un champ de dynpro (écran d'un programme standard SAP). Cet ajout de codes se fait au travers des transactions CMOD et/ou SMOD

Ces FIELDEXIT ne sont pas modifiés par SAP lors des montées de version.

▪ Utilisation du CUSTOMER-EXIT' :

Il est également possible de rajouter des parties de « code client » à un programme standard en passant par la création d'un projet. Ce projet permet d'activer la fonction CUSTOMMER au travers de laquelle sera ajouté le code client. (Transactions CMOD et SMOD).

Ces CUSTOMEREXIT ne sont pas modifiés par SAP lors des montées de version.

#### **1.3.17.3.3 Modification du code standard à partir de note SAP :**

SAP propose des améliorations des fonctionnalités sur ses programmes standards.

Toutes ses améliorations correspondent à des notes, on les trouve dans le système OSS (On-line Service System). Ce sont en général des lignes de codes qu'il faut introduire ou supprimer dans les programmes standards.

Il est nécessaire de documenter toute modification. En effet il faut, d'une part être capable a tout moment de fournir à SAP une liste des notes OSS qui ont été implémentées et d'autre part, être capable d'enlever le code des notes OSS parce qu'il ne répond pas aux attentes :

L'entête doit comporter:

- Référence de la demande,
- Numéro de la note OSS,
- Nom de la personne qui a effectué la modification,
- Date de la modification.

Au niveau du code, il ne faut jamais supprimer une ligne de code. Il faut la mettre en commentaire.  
A la fin de chaque ligne modifiée ( Ajout ou mise en commentaire, il faut indiqué le numéro de note), par exemple :  
Ajout d'une ligne de code.  
Data : fehlerexit type c.      ‘’ Note 66601

#### 1.3.17.3.4 Attribution d'un numéro de réparation

Un n° de correction SAP est ouvert à chaque intervention sur un programme (table , vue , data élément).  
Celui-ci sera « released » (fermé) par un numéro de transport au moment de sa mise en système d'intégration.

La zone 'désignation synthétique' du numéro de réparation devra être remplie comme suit :

Cas d'un USEREXIT :  
Objet SAP/ N° demande / Nom du demandeur  
Exemple : MV45AFZZ /DE2 99 08 01.1 / Dupont

Cas d'une note OSS :  
Note OSS / N° demande/ Nom du demandeur  
Exemple : OSS 66601 / DE2 99 08 02.1/ Durand

La description dans 'la documentation' devra être renseignée par le n° du programme  
ex : MV45AFZZ

Une fois validé, il faut « releaser » (Fermer) la réparation par un transport de type 'T', l'exporter vers le système cible puis l'importer.

#### 1.3.17.3.5 Centralisation des réparations

Toutes les réparations sont centralisées auprès du Responsable de l'Intégration.

### 1.3.18 Types d'interface

Les types d'interface sont définis par l'utilisateur.

### 1.3.19 Données externes

Les données externes sont définies avec des noms de fichier logique.

### 1.3.20 Tables internes

L'instruction "OCCURS n" devrait se rapprocher le plus possible du nombre de lignes prévues pour la table. Une zone dans le système est réservée en fonction de la taille définie dans l'instruction. Si le nombre de lignes sont ajoutées dans la table interne dépasse le nombre défini dans l'instruction OCCURS alors les enregistrements seront stockés dans une autre zone ce qui entraînera une dégradation des performances pour accéder à la table.

L'instruction "CLEAR <tab>" devrait être utilisée pour initialiser la table.

L'instruction "REFRESH <tab>" devrait être utilisée pour supprimer toutes les entrées de la table et libérer toutes les zones mémoires. Avant d'utiliser une table, il est recommandé d'effectuer un REFRESH puis un CLEAR dans cet ordre. L'instruction REFRESH n'efface pas l'enregistrement header.

L'instruction "FREE <tab>" devrait être utilisée à la fin du programme. L'instruction FREE effectue une libération de toutes les zones mémoire allouées pour la table et supprime toutes les lignes.

Si le nombre d'enregistrements n'est pas connu, mettre « OCCURS » à la valeur 0.

## **1.4 Normes de programmation des Modules ABAP / 4**

Le but de ce chapitre est de définir les règles de base pour le développement des programmes lors de la création ou la maintenance de pool de modules dans l'environnement SAP. Ce chapitre décrit les règles autour de l'utilisation des propriétés et des composants ABAP pour des programmes de type pool de modules.

### **1.4.1 Structure du programme**

Le "Work-Bench" des développeurs ABAP doit être utilisé pour créer et maintenir tous les programmes pool de module. Un exemple est donné dans l'annexe E.

### **1.4.2 Définitions des écrans**

Les écrans doivent utiliser les zones du dictionnaire de données pour toutes les variables. Ceci n'inclut pas les multiples sélections comme les radio boutons.

Les écrans doivent être numérotés à partir de 100. Il faut réutiliser les écrans dès que possible.

### **1.4.3 Interfaces (GUI)**

Les interfaces doivent respecter les standards SAP. Les menus utilisateur doivent contenir toutes les actions possibles pour la transaction. Les items qui ne sont pas concernés lors de l'exécution d'un écran doivent être inactifs. Une barre d'icônes doit permettre d'accéder aux actions les plus fréquemment utilisées.

### **1.4.4 Fenêtres POP-UP**

Les fenêtres pop-up sont autorisées.

### **1.4.5 OK\_CODE (SY-UCOMM)**

Une variable OK\_CODE doit être définie dans tous les écrans. Cette variable est définie avec la liste des zones de chaque écran créé (dernière zone dans la liste).

## 1.5 Contenu des ABAPS.

### 1.5.1 Présentation des listes

Le titre d'un programme ABAP spécifique devra faire référence à celui qui sera fourni dans le dossier de spécifications fonctionnelles générales et sera reporté en en-tête du programme tel qu'il est décrit dans l'annexe B.

Dans un programme ABAP spécifique, préciser la commande « REPORT » par l'option :  
line-size nbr\_lin

**ENTETE ( TOP-OF-PAGE).**

La première ligne comprend:

- Nom de programme en haut à gauche (SY-REPID).
- Le titre centré( donné dans les attributs utilisé l'instruction READ TEXTPOOL).
- Le libellé « Page »et le n° de page à droite (SY-PAGNO).

La deuxième ligne comporte:

- La machine (SY-HOST) à gauche.
- La date (SY-DATUM) et heure (SY-UZEIT) d'édition de la liste à droite.
- Nom du dossier batch-input créé (si batch-input) à gauche
- Nombre d'enregistrements traités à gauche
- Code retour de l'ouverture du fichier (si lecture d'un fichier UNIX) à gauche
- Code retour de la fermeture du fichier (si lecture d'un fichier UNIX) à gauche

Les paramètres de sélections et select-options devront être rappelés en première page.

### 1.5.2 Contrôle des autorisations

Afin de vérifier si un utilisateur est autorisé ou non à exécuter cet ABAP ( Exemple : Liste des clients dont le plafond de crédit est dépassé).

Il est nécessaire de connaître le nom des objets qui doivent être contrôlés lors de l ' exécution d'un ABAP.

L'instruction ABAP à utiliser est la suivante :

AUTHORITY-CHECK OBJECT object

ID name1 FIELD f1

ID name2 FIELD f2

Si le code retour est à zéro, l'utilisateur possède les autorisations nécessaires dans sa fiche utilisateur pour exécuter le programme. Par contre si le code retour n'est pas égal a zéro , il convient de bloquer l'exécution par un message de type E (Vous n'êtes pas autorisé a exécuter cet ABAP).

**Les objets d'autorisations devront être fournis dans le dossier fonctionnel et/ou technique..**

## 1.6 Profil du développeur

Un développeur est amené à évoluer dans deux environnements différents (un environnement de développement DE2 ou DB2 et un autre de recette QE2 ou QB2). Voir plus de détails sur le paysage système en annexe I.

Les compétences minimales requises pour un intervenant sur des développements SAP à l'APHP sont :

- Créer , modifier afficher les ABAPS spécifiques
- Deboguer les ABAPS
- Modifier les ABAPS standards (USEREXIT, FIELDEXIT, CUSTOMEREXIT).
- Soumettre on-line et en batch les ABAPS.
- Créer, modifier les jobs.
- Créer , modifier, afficher les formulaires.
- Consulter son spool.
- Soumettre ses dossiers de batch-input
- Créer, modifier, afficher des données applicatives

Les compétences minimales requises pour un intervenant sur un environnement de recette SAP à l'APHP sont :

- Afficher les ABAPS,
- Deboguer les ABAPS,
- Soumettre on-line et en batch les ABAPS,
- Afficher les formulaires,
- Consulter son spool,
- Afficher des données applicatives.

### **Remarques :**

En ce qui concerne les objets SAP, ils dépendent soit du mandant soit de la machine.

(Par exemple, un programme ABAP s'il est modifié dans le mandant 100, sera modifié dans tous les autres mandants. Par contre un formulaire s'il est modifié dans le mandant 100 ne sera pas modifié ailleurs).

Certaines tables, surtout systèmes ne dépendent pas du mandant (inter-mandants)

En outre ,les objets d'autorisations, formulaires, dépendent du mandant.

**Il convient donc de surveiller dans une configuration de plusieurs mandants par machine l'évolution de ces objets.**

## 1.7 Transport (*Fonctionnement général*).

### 1.7.1 OBJET

Cette procédure décrit les règles à appliquer lors d'un transport d'objet SAP R/3 dans les différents systèmes.

### 1.7.2 DOMAINE D'APPLICATION

Cette procédure s'applique à l'ensemble des transports d'objets SAP R/3

### 1.7.3 DEROULEMENT D'UN TRANSPORT

Sous réserve de l'utilisation d'un outil d'exécution des ordres de transport tel que JIRA,  
on appliquera les procédures décrites de la manière suivante :

Tout ordre de transport devra comporter dans son commentaire le préfixe 'ZDOM' suivi du nom de code de l'objet à transporter plus une description minimale.

Un transport s'effectue en deux temps :

- D'abord dans la machine de recette externe (machine d'intégration reflétant le système de production)
- Ensuite dans la machine de production.

Le transport en machine d'intégration sera effectué par l'équipe technique ou le développeur lui-même ,après validation fonctionnelle du responsable de domaines ou validation technique de cette équipe technique. Un formulaire Jira est mis de demande de transport est mis en place et permet de préciser le contenu de l'ordre du transport afin de pouvoir procéder à cette validation.

Seuls les responsables de domaine et/ou de modules ainsi que les administrateurs peuvent créer des ordres de transport ou de modification.

Les consultants fonctionnels ne peuvent créer, ni ordre de transport, ni ordre de correction. Ils peuvent uniquement affecter les changements de customizing aux ordres de modifications déjà créés (via l'outil Spro).

Le transport en machine de production sera effectué par l'équipe technique (packaging par Architecture) après validation du responsable de domaine fonctionnel concerné et validation technique de l'équipe de production sur la machine préproduction (VAT).

#### 1.7.3.1 Transport d'objet SAP R/3 autre que le customizing

##### 1.7.3.1.1 Vérification d'objet

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	48/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------



Vérification, par le responsable des équipes de développement, d'une part du contenu de la correction ou de la réparation, et d'autre part de l'application des standards de développement (transaction SLIN).

#### 1.7.3.1.2 Validation d'un transport

Un numéro de transport est créé pour « releaser= versionner » (fermer) une correction ou une réparation. La désignation synthétique est renseignée par le libellé et le n° de la Jira pour information.

#### Exemple :

transport « Mise à jour des postes techniques GHU »/ NSIGGC-29819

Pour avoir les détails sur les normes de Transport via l'outil Jira, se référer au document sur les normes de validations élaboré par le domaine technique .

Le numéro de correction ou de réparation est ensuite 'releasé' en indiquant le numéro de transport précédemment attribué, le type de transport (K = correction, T : réparation), et le système cible (machine d'intégration ou de production).

#### Remarque :

Il est possible de « releaser» (fermer) directement le numéro de correction par un transport (Bouton : Valider correction).

#### 1.7.3.1.3 Transfert du transport dans les différents environnements

Le transfert s'effectue à travers une session du système d'exploitation soit en machine d'intégration, soit en machine de production.

#### 1.7.3.1.4 Contrôle de l'opération de transport

Un contrôle du code retour sur le transport est effectué dans le système SAP R/3.

Le transport s'est correctement déroulé :

- Informer le fonctionnel soit par messagerie SAP ou par d'autres moyens disponibles.

Le transport a échoué :

- Analyse du rejet, correction et nouvel essai de transport. Impliquer s'il le faut l'équipe qui administre l'outil Jira pour analyser les motifs de rejets.
- Attention : Parfois le rejet a lieu au-delà des environnements de développement. DPP rentre des problèmes de livraison avec des codes retours 4=warning ou 8= erreur qu'il faut analyser par l'équipe projet . Ce genre d'erreur arrive quand le même programme est relivré plusieurs fois de manières partielles.

---

## **1.8 Développement des MATCHCODES**

### **1.8.1 Ajout d'un nouvel ID (index) MATCHCODE sur un matchcode standard existant**

Le développement est accessible par l'IMG (paramétrage)

Il est à la charge de l'équipe chargée du paramétrage (donc n'est pas à la charge de l'équipe de développement)

L'équipe de production reste le support technique de l'équipe de développement pour valider les accès de table.

Cette création de MATCHCODE peut donc faire l'objet d'une demande de validation et / ou de complément d'analyse détaillée non suivie de programmation

### **1.8.2 Création d'un nouveau MATCHCODE pour développement spécifique**

Ce MATCHCODE ne peut être conçu que dans le cadre d'un développement spécifique.

Ce développement suit le processus normal d'une demande de développement spécifique (validation du dossier de spécifications fonctionnelles générales, mise en place du dossier de spécifications fonctionnelles détaillées, programmation, tests, .....).

### **1.8.3 Création d'un nouveau MATCHCODE pour une transaction standard SAP**

C'est une demande de modification du standard SAP.

**Ce type de demande n'est pas autorisé.**

### **1.9 User-exit**

Les demandes de création de user-exit sont faites par l'équipe SAP via des demandes OSS.  
La localisation des user-exit utilisés est fournie par l'équipe SAP.

### **1.10 Codification des jobs**

Elle devra suivre les prérogatives du ou des normes de développement APHP en vigueur.

## **2<sup>ème</sup> Partie :Gestion des clés de développement :**

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	51/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

Nous devons distinguer 2 types de clés :

( un diagramme de flux sera inséré ici ultérieurement)


- Les clés développeurs : Ce type de clé permet à un utilisateur qui a le rôle de développement de pouvoir lire , modifier , et créer un programmes Abap. L'octroi de clés utilisateur est arbitré par la direction du projet sur proposition du responsable des développements.
- Les clés objets de développement :  
Sont octroyés à un développeur pour qu'il puisse modifier un objet standard (table, index, module function, report, ...etc). L'équipe de développement doit analyser ce genre de demande, afin de calculer l'impact sur les autres objets existants sollicités par l'objet à modifier.

Pour ce faire l'équipe de développement seule habilitée à faire ce genre d'opération , doit solliciter ces clés SAP directement à partir du site institutionnel de SAP.

#### SSCR - My Company's Registrations

Developers Objects Download all SSCR keys

This list contains objects registered for those installations for which you have SSCR authorization.

 Use the first line for entering a search/filter term ( \* for fuzzy search).

Objects = 359

  Display

PgmID	Type	Object name	Release	Advance con	Customer	Installation	Installation name	Registration date	Registration key	Registered by
	(all)		(all)			(all)	(all)	(all)		
<input type="checkbox"/> R3TR	TABL	<a href="#">TXW FI HD FR</a>	740		438880	0020277565	Install T ERP	06.07.2015	0309683350251322039	<a href="#">José CAPARRÓS</a>
<input type="checkbox"/> R3TR	TABL	<a href="#">TXW FI HD</a>	740		438880	0020277565	Install T ERP	06.07.2015	05212203731740328489	<a href="#">José CAPARRÓS</a>
<input type="checkbox"/> R3TR	PROG	<a href="#">RM07MLBS</a>	740		438880	0020277565	Install T ERP	03.07.2015	26359796580378601127	<a href="#">Equipe Developpements</a>
<input type="checkbox"/> R3TR	PROG	<a href="#">SEW TRANSP EXT BOSSET ENTRIES</a>	740		438880	0020277565	Install T ERP	14.06.2015	24831480862479363735	<a href="#">Adel BERREGHIS</a>
<input type="checkbox"/> R3TR	FUGR	<a href="#">MGMD</a>	740		438880	0020277565	Install T ERP	11.05.2015	20879447143824569516	<a href="#">Equipe Developpements</a>
<input type="checkbox"/> R3TR	SHLP	<a href="#">DEBI</a>	740		438880	0020277565	Install T ERP	24.04.2015	33865844080505228536	<a href="#">Aimé BADRANE</a>
<input type="checkbox"/> R3TR	DTL	<a href="#">COD RUBRIQ</a>	701		438880	0020277565	Install T ERP	17.04.2015	07167726930571423436	<a href="#">Aimé BADRANE</a>
<input type="checkbox"/> R3TR	PROG	<a href="#">FMKF REPORT01</a>	740		438880	0020277565	Install T ERP	18.03.2015	24289513550830194795	<a href="#">Aimé BADRANE</a>
<input type="checkbox"/> R3TR	DTL	<a href="#">MSAUS</a>	740		438880	0020277565	Install T ERP	05.02.2015	07295764470204514877	<a href="#">Aimé BADRANE</a>
<input type="checkbox"/> R3TR	DTL	<a href="#">EAUSZT</a>	740		438880	0020277565	Install T ERP	05.02.2015	05208278813683708927	<a href="#">Aimé BADRANE</a>

### 3 ème Partie :Gestion des Livrables SAP :

Nom. :	ME-NDV.doc	Émetteur :	Aimé Badrane	Version :	1.2	Date :	28/04/2025	Page:	52/71
--------	------------	------------	--------------	-----------	-----	--------	------------	-------	-------

### 3.1. Documents pré-requis pour un livrable

Quand un développement est traité en interne ou en externe, l'APHP attend de la part du développeur (interne) ou de la part du titulaire les composants suivants:

- La rédaction ou mise à jour des dossiers de spécifications techniques (DST),
- La création ou la modification des paramétrages, programmes spécifiques et rédaction ou mise à jour des documents associés. Ces réalisations, effectuées selon les « règles de l'art » devront être optimisées. L'AP-HP utilise l'outil CAST permettant de s'assurer de la qualité des développements et le mettra en œuvre, afin de s'assurer de la bonne qualité des développements.
- Réalisation des tests unitaires à partir de cas de tests définis par le titulaire (et données de tests associées)
- Réalisation des tests d'intégration à partir de cas de tests définis par le titulaire (et données de tests associées)
- La prise en compte systématique pour toute correction/évolution d'un plan de réversibilité garantissant à l'AP-HP la possibilité de restaurer de façon automatique la situation antérieure à l'installation de la correction / évolution.
- Dans le cas de développements importants, il sera organisé entre le titulaire et l'AP-HP des réunions de suivi particulières visant à valider les choix effectués, détecter au plus tôt les incompréhensions mutuelles, suivre les plannings et les risques projet

NB : Le développeur (interne) ou le titulaire du marché AT, devra s'assurer de livrer un programme optimisé. Les tests d'intégration devront intégrer la réalisation de tests de performance. Dans le cas contraire le développeur (interne) ou le titulaire du marché assumera sa responsabilité si les performances ne sont pas concluants.

### 3.2. Livrables exigés

Le tableau ci-dessous précise la liste des livrables minima qui doivent être inclus dans les jiras de livraison :

Code Livrable	Libellé du Livrable
<b>Lxx1</b>	DSF ou DSF Paramétrages et/ou développements
<b>Lxx2</b>	Dossier de spécifications techniques (DST)
<b>Lxx3</b>	Cas de tests unitaires (et jeux de données associés) : optionnel
<b>Lxx4</b>	Cas de tests d'intégration (et jeux de données associés) : selon DSF
<b>Lxx5</b>	Bilan des tests unitaires : optionnel
<b>Lxx6</b>	Bilan des tests d'intégration : selon DSF
<b>Lxx6</b>	Documentation d'installation et d'exploitation (si nécessaire) + Dossier d'analyse ordonnancement VTOM.

<b>Lxx7</b>	Manuel utilisateur (si nécessaire) exigence fonctionnel.
<b>Lxx8</b>	Compte rendu des ateliers et/ou réunion et supports de présentation
<b>Lxx9</b>	Manuel utilisateur mis à jour (si nécessaire)

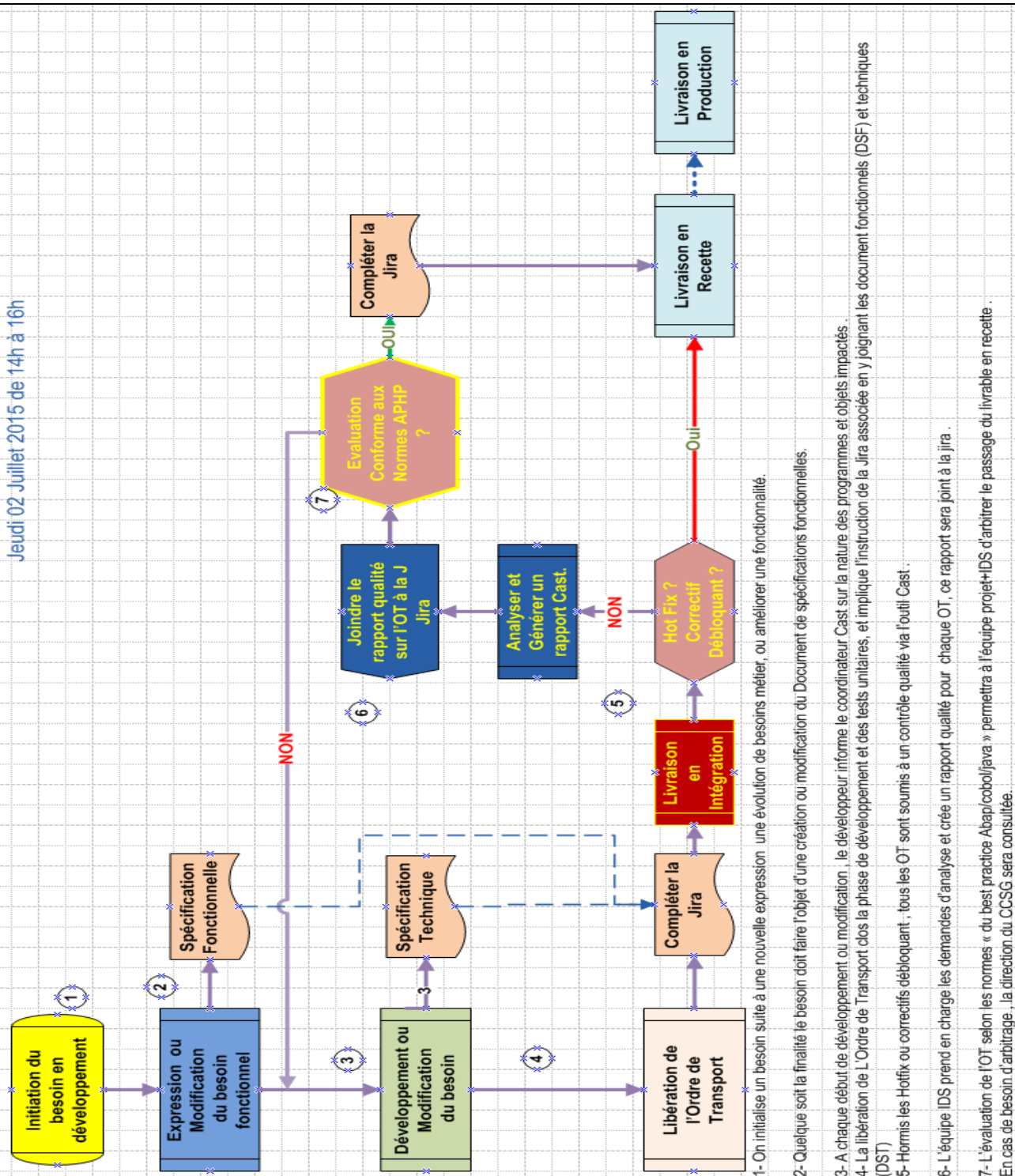
## 4. ème Partie : Validation des programmes via l'outil CAST

Le synoptique suivant a été conçu par l'équipe développement, et présenté aux domaines métiers et validé par la direction du projet. La bonne mise en œuvre de ce processus est assujetti à la validation de chaque ordre de transport par l'équipe de développement via l'outil CAST. Cet outil de qualimétrie , permet de mesurer les indicateurs de bonne santé des livrables .

### 4.1. Circuit de Validation :

# APHP/CCDG/IDS/Développements : Circuit de Validation des Ordres de Transport

Jeudi 02 Juillet 2015 de 14h à 16h



Depuis 2014, l'AP-HP s'est dotée de la solution CAST pour contrôler la qualité des développements (tant internes qu'externalisés) qui doit permettre une meilleure maintenabilité des systèmes SAP. Pour ce faire, elle a arrêté une liste de 150 règles (Cf annexe H) qui doivent permettre de remplir deux objectifs :

#### • **Se prémunir des violations majeures :**

Sur ces 150 règles, un jeu de 15 règles est nommé « violations majeures » dans le modèle APHP. Lors d'une mise à disposition de développements logiciels (livraison par bordereau de livraison) et avant installation en environnement de recette, les ordres de transport disponibles en environnement d'intégration seront vérifiés par l'outil CAST. Le nombre de violation critique sur les OT de la livraison doit être faible. Si ce n'est pas le cas, la qualité de la livraison sera détériorée. La liste des 15 règles concernées est précisée ci-dessous. En fonction des objectifs qualité de l'AP-HP, ce jeu de règle peut être amené à évoluer.

Avoid unchecked return code (SY-SUBRC) after OPEN SQL or READ statement  
Avoid missing WHEN OTHERS in CASE statements  
Avoid using SQL queries inside a loop  
Avoid Open SQL queries in loops  
Avoid READ TABLE without BINARY SEARCH  
Avoid using SELECT ... INTO CORRESPONDING FIELDS OF  
Avoid Open SQL SELECT queries without WHERE condition  
Avoid using dynamic queries  
Avoid accessing SAP standard Tables in modification from custom code  
Avoid using BREAK or BREAK-POINT statement  
Avoid empty catch blocks  
Avoid nested SELECT ... ENDSELECT statements  
Avoid testing specific values for SY-UNAME  
Avoid using Native SQL  
Avoid unchecked return code (SY-SUBRC) after AUTHORITY-CHECK

#### **4.2. Suivi de la qualité des systèmes :**

Les 150 règles, regroupées par thèmes, conformément aux recommandations de la norme ISO 9126-3 donnent lieu à la détermination de 5 indicateurs :

1. Mesure de la performance,
2. Mesure de la robustesse (facilitation de l'exploitation des systèmes),
3. Mesure de la transférabilité (lisibilité et commentaire des programmes)
4. Mesure de la sécurité (données non accessibles),
5. Mesure de l'évolutivité (absence de régression sur changements).

Une mesure de référence de ces indicateurs sera réalisée sur les systèmes SAP lors de la phase d'initialisation du marché. Cette mesure servira de point de comparaison initial pour suivre la qualité des systèmes SAP dans le temps. Mensuellement une mesure de ces indicateurs sera réalisée qui ne pourra être inférieure à celle du mois précédent.

L'équipe de développement définira périodiquement ou ponctuellement selon le type du livrable ; une revue, inspection ou audit du code livré en interne / externe.

Un document sera généré et adressé selon le type d'audit à :

- La direction du projet + CP Processus (document synthèse)
- Responsables Domaine + chef de projet (document détaillé)

Quel que soit la situation des développements, l'équipe de développements publiera de manière bimensuelle un rapport qui sera exposé lors des comités de projet du CCDG.



---

### 4.3. Violations impliquant un rejet du livrable :

Ce paragraphe liste les règles de programmation à respecter au sein de l'AP-HP pour les développements ABAP (SAP)

Ces règles sont à la base du calcul des notes à obtenir pour les facteurs de santé ISO (groupe d'indicateurs n°1 dans le CCTP). Le respect total de ces règles conduit à une notation maximale de 4 sur une échelle de 1 à 4. **L'AP-HP exige que pour tout nouveau développement la notation soit d'au moins 3,2 soit respectée afin de conserver l'intégrité technique de ses nouveaux développements. Ou à minima qu'aucune des 15 règles citées ci-dessus ne fasse partie du compte rendu Excel généré .**

**Pour les anciens programmes qui sont en modification il ne faut pas que de nouvelles violations apparaissent dans le compte rendu généré .**

Le seuil de tolérance est de 0 (zéro). Ces règles ont un impact fort sur l'exploitation ou la performance des Transactions SAP. Aucun nouveau développement contenant une violation critique ne doit être transportée dans les environnements productifs.

Afin de ne pas constater à postériori, lors des contrôles mensuels, l'introduction de mauvaises pratiques dans le code ABAP, l'équipe de développement APHP, a installé sur ses systèmes SAP un automatisme pour contrôler les ordres de transport (transport requests). L'objectif est de corriger au plus tôt les développements avant de basculer les objets ABAP de l'environnement de développement vers l'environnement de recette. Les mauvaises pratiques sont listées et envoyées directement par email au Centre de compétences Domaine Gestion de l'AP-HP. La totalité des règles prises en compte sont détaillées dans **l'annexe K**.

## 5. ANNEXES

### 5.1. Annexe A : Table des codes de zones d'application

La liste proposée non exhaustive ci-dessous est fournie à titre d'exemple, il conviendra de se référer aux codes application définis sous la ou les machines SAP.

Code de zone	Description
<b>B</b>	Business Information Warehouse
<b>L</b>	Gestion emplacements magasin
<b>M</b>	Gestion des articles
<b>S</b>	Noyau système
<b>U</b>	MDE (modèle données entreprise)
<b>V</b>	ADV (Administration des ventes)
<b>Y</b>	Centrale client
<b>I</b>	Inter-application
<b>Z</b>	Client « filiale »

## 5.2. Annexe B : Programme Standard

\*\*\*\*\*

```
* Nom de programme:          xxxxxxxxxxxxxxxxxxxx
*
* Description:               Description           of           program
*
* Date/Author:              Date                 written/Authors      name
*
* Table Updates:            Listing of tables updated
*
*.Liste
```

```
*
*                               Return                      Codes:
*
*                               Special                      Logic:
*
*
*                               Includes:
```

```
*****
*                               MODIFICATION                      LOG
*****
* Date      Programmer      Request#      Description
* -----
* dd/mm/yyyy  xxxxxxxx      nnnnn      New      Program
```

```
*****
REPORTS                      XXXXXXXX.
```

```
*
*                               TABLES
*****
```

```
TABLES:
```

```
*
*                               SELECT                      OPTIONS
*****
```

```
*
*                               PARAMETERS
*****
```

```
PARAMETERS:
```

```
*
*                               DATA
*****
```

```
DATA:
**                               Accumulators          **
**                               Constants              **
**                               Switches               **
```

**	Work	Fields	**
**	Internal	Tables	**
**	Data structures	/ Strings	**

```

*****
*           I N I T I A L I Z A T I O N
*****
INITIALIZATION.
*****
*           A T           S E L E C T I O N           S C R E E N
*****
AT           SELECTION           SCREEN.
*****
*           T O P           O F           P A G E
*****
TOP           OF           PAGE.
*****
*           E N D           O F           P A G E
*****
END           OF           PAGE.
*****
*           S T A R T           O F           S E L E C T I O N
*****
START           OF           SELECTION.
*****
*           E N D           O F           S E L E C T I O N
*****
END           OF           SELECTION.
*****
*           F O R M S
*****

```

\* Input Parameters: (For Includes/Function Modules)

\* Output Parameters: (For Includes/Function Modules)

\*

### 5.3. Annexe C : Codes suffixes pour les noms des zones

Cette liste est fournie à titre d'exemple. Il conviendra si nécessaire de l'adapter aux besoins définis par APHP.

Suffixe	Type
DT	Date
TM	Time
NB	Numeric
PD	Packed decimal
FP	Floating Point
TX	Text
NM	Name
HX	Hexadecimal
BN	Binary
DS	Data structure / String
IX	Index
FG	Flag

#### 5.4. Annexe D : Mot-clé court, moyen, long

Ils pourront si nécessaire être redéfinis pour les besoins du client.

Mot-clé			Description
Court	Moyen	Long	
MAT	MATRL	MATERIEL	Matériel
TN	TN	TN	Texte Numérique
ADR	ADR	ADRESSE	Adresse
VEN	VEN	VENTE	vente
TRC	TRANS	TRANSCO	

## 5.5. Annexe E : Exemple d'écran

### Flow Logic

```
*-----*
Process                                Before                                Output.
*
module      SET_STATUS.                "Called      by      all      screens
* Enter screen specific modules here i.e. dynamic screen locking, etc .
*
*-----*
Process                                After                                Input.
*
module      PAI_EXIT    At    EXIT-COMMAND.    "Called      by      all      screens
*                               Save            OK_CODE      pressed
module      PAI_INITIALIZE.    "Called      by      all      screens
*
* Enter screen specific modules here
* i.e. Field checks, Loops, Table Controls, etc. .
*
* Process user command (OK_CODE) after all checks completed
module USER_COMMANDS.
```

### Process Before Output modules

```
*-----*
***                                INCLUDE                                MZxxx001
*-----*
*&-----*&
*&  Module                        SET_STATUS                                OUTPUT
*&-----*&
*      text
*&-----*&
MODULE                                SET_STATUS                                OUTPUT.
CASE      SY-DYNNR.                " Interrogate      screen      number
      WHEN '0100'.    "Screen description for screen 0100
      SET      PF-STATUS                                '100MAIN'.
      SET      TITLEBAR                                '100'.
      WHEN '0200'.    "Screen description for screen 0200
      SET      PF-STATUS                                '200MAIN'.
      SET      TITLEBAR                                '200'.
*      ...      Add      any      subsequent      screens
      ENDCASE.
ENDMODULE. "SET_STATUS OUTPUT
```

### Process After Input modules

```

*-----*
***                               INCLUDE                               MZxxx001
*-----*
*&-----*&
*&  Module                        PAI_EXIT                        INPUT
*&-----*&
*      text
*-----*
MODULE                                PAI_EXIT                                INPUT.
CASE      SY-DYNNR.      "      Interrogate      screen      number
      WHEN      '0100'.      "Screen      0100
      SET      SCREEN      0.
      LEAVE      SCREEN.
      WHEN      '0200'.      "Screen      0200
      *      First      process      any      de-queuing      of      entries,
      *      pop-ups      to      confirm      data      loss,      etc.
CASE SY-UCOMM. "Interrogate system field to find out with
fast      exit      pressed
      WHEN      'RW'.
      SET      SCREEN      0.
      WHEN      OTHERS.
      SET      SCREEN      0100.
      ENDCASE.
      LEAVE      SCREEN.
*      ...      Add      any      subsequent      screens
      ENDCASE.
ENDMODULE. "PAI_EXIT INPUT
*&-----*&
*&  Module                        PAI_INIT                        INPUT
*&-----*&
*      text
*-----*
MODULE                                PAI_INIT                                INPUT.
*      set      new      ok_code
SAVE_OK      =      OK_CODE
*      clear      ok_code
CLEAR      OK_CODE
ENDMODULE. "PAI_INIT INPUT
*&-----*&
*&  Module                        USER_COMMANDS                        INPUT
*&-----*&
*      text
*-----*
MODULE                                USER_COMMANDS                                INPUT.
CASE      SY-DYNNR.      "      Interrogate      screen      number
      WHEN      '0100'.      "Screen      0100      (enter      description)
      PERFORM PROCESS_0100_OK_CODES. (Form routines to process
OK      codes)

```



```

      WHEN      '0200'.      "Screen      0200      (enter      description)
      PERFORM  PROCESS_0200_OK_CODES. (Form routines to process
OK                                         codes)
*      ...      Add      any      subsequent      screens
      ENDCASE.
ENDMODULE. "USER_COMMANDS INPUT
  
```

## 5.6. Annexe F : Packages ou Classes de développement

### Norme de codification :

Position	Code et signification
1	<b>Z</b>
2-4	Code Activité/Domaine = 'DOM'
5-20	N'importe quels caractères. Doit être significatif pour la classe

### Liste des classes de développement :

Cette liste est fournie à titre d'exemple mais devra être adaptée aux besoins de APHP.

Code classe	Système d'intégration	Système de consolidation	Libellé
ZDOM001	DE1	QE2	Classe de développement client
ZDOMCOM	DE1	QE2	Développement Commercial
ZDOMDAT	DE1	QE2	Développement Données de Base
ZDOMEXP	DE1	QE2	Programmes et utilitaires d'exploitation
ZDOMLOG	DE1	QE2	Développement Logistique
ZDOMMKG	DE1	QE2	Développement Marketing
ZDOMTES	DE1	QE2	Développement Test, si \$TMP non utilisable

Cette liste étant susceptible d'évoluer, afficher directement la table **TDEV** dans SAP. : exemple

DEVCLASS	INTSYS	CONSYS	CTEXT	KORRFLAG	AS4USER	PDEVCLASS	DEVUNIT
ZFI_IDS				X	ADARNOU	ZDEV	HOME
ZMAR_CMDE_ACHAT				X	ASYLLA	ZDEV	HOME
ZFM_CPT_BDG				X	ASYLLA	ZDEV	HOME
ZFM_COMMON				X	ASYLLA	ZDEV	HOME
ZFM				X	ASYLLA	ZDEV	HOME
ZMAR_FICHE_MARCHE				X	ASYLLA	ZDEV	HOME
ZFI_CASH				X	ASYLLA	ZDEV	HOME
ZRF				X	ASYLLA	ZDEV	HOME
ZRM_GLOBAL				X	ASYLLA	ZDEV	HOME
ZEXT_RF				X	ASYLLA	ZDEV	HOME
ZCO_CPT_ANA				X	ASYLLA	ZDEV	HOME
ZCO				X	ASYLLA	ZDEV	HOME
ZFI_FORMULAIRES				X	CCZUBA	ZDEV	HOME
ZFI_SCHEMA_TVA				X	CCZUBA	ZDEV	HOME
ZFI_TRANCHES_NUMERO				X	CCZUBA	ZDEV	HOME
Z_REC				X	ELAPIERR	ZDEV	HOME
ZUSEUCP_AUTH				X	EPI-USE_LABS	ZCUS	HOME
ZUSE_AUTH				X	EPI-USE_LABS	ZCUS	HOME
ZUSEPDS_AUTH				X	EPI-USE_LABS	ZCUS	HOME
ZUSE				X	EPI-USE_LABS	ZCUS	HOME
ZDPP				X	FHULAUD	ZDEV	HOME
ZBI				X	GBARROIS	ZDEV	HOME
ZDEP_DMP				X	JCAPARROS	ZDEV	HOME
ZZTEMP_NHO				X	JCAPARROS	ZDEV	HOME
ZMM_IDS				X	JNAY	ZDEV	HOME
ZWF				X	MAHAZZAM	ZDEV	HOME
ZTRV_IDS				X	MDJEARAM	ZDEV	HOME
ZMAR_IDS				X	MSALHI	ZDEV	HOME
ZTRV_REPORT				X	MSALHI	ZDEV	HOME
ZFI				X	NHOCQUEL	ZDEV	HOME
ZFI_SUBSTITUTIONS				X	NHOCQUEL	ZDEV	HOME
ZFI_VALIDATIONS				X	NHOCQUEL	ZDEV	HOME

### 5.7. Annexe G : Liste des applications R/3

Cette liste n'est pas exhaustive, il sera nécessaire de la compléter en fonction des besoins APHP.

<b>CA</b>	Cross-Application Functions
<b>LO</b>	Logistics - General
<b>SD</b>	Sales & Distribution
<b>MM</b>	Materials Management
<b>PP</b>	Production Planning
<b>PS</b>	Project System
<b>PD</b>	Personnel Planning and Development
<b>WM</b>	Warehouse Management
<b>BC</b>	Basis Components
<b>IN</b>	International Development

### 5.8. Annexe H : CAST (iso ISO 9126-3 ) Liste des bonnes pratiques en Abap

Quality Rule (règles mesurées mensuellement)
4GL Complexity Distribution
Avoid "SELECT *" or "SELECT SINGLE *" queries
Avoid "SELECT *" queries
Avoid accessing multiple times the same SAP Table or View in an SAP include (DELETE)
Avoid accessing multiple times the same SAP Table or View in an SAP include (INSERT)
Avoid accessing multiple times the same SAP Table or View in an SAP include (UPDATE)
Avoid accessing multiple times the same SAP Table or View in an SAP Program (DELETE)
Avoid accessing multiple times the same SAP Table or View in an SAP Program (INSERT)
Avoid accessing multiple times the same SAP Table or View in an SAP Program (UPDATE)
Avoid accessing SAP standard Tables in modification from custom code
Avoid artifacts having recursive calls
Avoid Artifacts with a Complex SELECT Clause
Avoid Artifacts with Group By
Avoid Artifacts with high Commented-out Code Lines/Code Lines ratio
Avoid Artifacts with High Cyclomatic Complexity
Avoid Artifacts with High Cyclomatic Complexity
Avoid Artifacts with High Depth of Code
Avoid Artifacts with High Depth of Nested Subqueries
Avoid Artifacts with High Essential Complexity
Avoid Artifacts with High Fan-In
Avoid Artifacts with High Fan-Out

Avoid Artifacts with High integration complexity
Avoid Artifacts with High RAW SQL Complexity
Avoid Artifacts With Queries on more than 4 Tables
Avoid Artifacts with SQL statement including subqueries
Avoid Artifacts with Subqueries
Avoid Classes implementing too many Interfaces
Avoid Classes with a High Depth of Inheritance Tree
Avoid Classes with a High Lack of Cohesion
Avoid Classes with a High Lack of Cohesion 2
Avoid Classes with a low comment/code ratio
Avoid cyclic calls between Event and its handled Method
Avoid disabling source code inspection
Avoid empty CATCH blocks
Avoid empty Functions, Forms and Modules
Avoid empty IF-ENDIF blocks
Avoid empty Includes
Avoid empty Programs
Avoid Function pools with more than 20 functions
Avoid Functions with low comment/code ratio
Avoid having multiple artifacts deleting data on the same SQL table
Avoid having multiple Artifacts inserting data on the same SQL Table
Avoid having multiple Artifacts updating data on the same SQL Table
Avoid High Response for Classes
Avoid Include Circular references
Avoid Includes with low comment/code ratio
Avoid Interfaces with a low comment/code ratio
Avoid large Artifacts - too many Lines of Code
Avoid large Classes - too many Data Members
Avoid large Classes - too many Methods
Avoid large Interfaces - too many Methods
Avoid large Methods - too many Lines of Code
Avoid large Programs - too many Lines of Code
Avoid long Table names (SAP SQL)
Avoid long View names (SAP SQL)
Avoid method invocation in a loop termination expression
Avoid Methods with a low comment/code ratio
Avoid missing WHEN OTHERS in CASE statements
Avoid nested loops
Avoid nested SELECT ... ENDSELECT statements
Avoid Open SQL queries in loops
Avoid Open SQL SELECT queries without WHERE condition

Avoid passing parameter by value
Avoid Programs with low comment/code ratio
Avoid Programs with more than 3 levels of inclusion
Avoid Programs with too many includes
Avoid Programs/Includes including large Includes
Avoid Programs/Includes with too many Forms
Avoid raising an exception in a Web Dynpro Supply Function or in a Method called by a Supply Function
Avoid READ TABLE without BINARY SEARCH
Avoid redundant indexes
Avoid SELECT ... BYPASSING BUFFER
Avoid SQL queries not using the first column of a composite index in the WHERE clause
Avoid SQL queries on XXL tables not using the first column of a composite index in the WHERE clause
Avoid SQL queries on XXL tables that no index can support
Avoid SQL queries on XXL Tables using Functions on indexed Columns in the WHERE clause
Avoid SQL queries on XXL Tables with implicit conversions in the WHERE clause
Avoid SQL queries that no index can support
Avoid SQL queries using functions on indexed columns in the WHERE clause
Avoid SQL queries with implicit conversions in the WHERE clause
Avoid Tables having Indexes with a too large Index definition
Avoid testing specific values for SY-UNAME
Avoid Too Many Copy Pasted Artifacts
Avoid too many Indexes on one Table
Avoid unchecked return code (SY-SUBRC) after AUTHORITY-CHECK
Avoid unchecked return code (SY-SUBRC) after OPEN SQL or READ statement
Avoid unchecked return code (SY-SUBRC) after opening and reading dataset
Avoid undocumented Classes
Avoid undocumented Functions
Avoid undocumented Includes
Avoid undocumented Interfaces
Avoid undocumented Methods
Avoid undocumented Programs
Avoid undocumented User-exits
Avoid unreferenced Classes
Avoid unreferenced Functions
Avoid unreferenced Includes
Avoid unreferenced Interfaces
Avoid unreferenced Members
Avoid unreferenced Methods
Avoid User-Exits with low comment/code ratio
Avoid using "ORDER BY" in SELECTS
Avoid using "SELECT DISTINCT", use DELETE-ADJACENT

Avoid using AT Events in combination of LOOP AT .... WHERE constructs

Avoid using BREAK or BREAK-POINT statement

Avoid using dynamic queries

Avoid using EXIT statement in Include

Avoid using FOR ALL ENTRIES IN without emptiness check

Avoid using FOR ALL ENTRIES IN without emptiness check on XXL Tables

Avoid using hardcoded paths

Avoid using IS [NOT] NULL in WHERE condition

Avoid using literals in assignments (hardcoded values)

Avoid using LOOP INTO, use LOOP ASSIGNING instead

Avoid using Native SQL

Avoid using SELECT ... ENDSELECT statement

Avoid using SELECT ... ENDSELECT statement on XXL Tables

Avoid using SELECT ... FOR UPDATE

Avoid using SELECT ... INTO CORRESPONDING FIELDS OF

Avoid using SQL queries inside a loop

Avoid using SYSTEM-CALL

Avoid using the NOT LIKE operator in WHERE clauses

BAPIs must not cause the Program to abort or terminate

Class Complexity Distribution (WMC)

Class Fan-In Distribution

Class Fan-Out Distribution

Class members should be declared as Private

Class naming convention

Commented-out Code Lines/Code Lines ratio (% of LOC)

Complexity Volume (% of LoC)

Copy Pasted Code (% of LOC)

Coupling Distribution

Cyclomatic Complexity Distribution

Declare as Final all classes that will not be sub-classed

Function naming convention

Include naming convention

Interface naming convention

Never use SQL queries with a cartesian product

Never use SQL queries with a cartesian product on XXL Tables

Never use the ON CHANGE OF statement

OO Complexity Distribution

Prefer UNION ALL to UNION

Processing Screen Naming Convention

Program naming convention

Reuse by Call Distribution

SAP database view naming convention

SAP Table naming convention

Size Distribution

SQL Complexity Distribution

Transaction naming convention

Web Dynpro - Avoid changing the program flow

Web Dynpro - Never use direct calls to routine via "me->", use "wd\_This->" instead

Web Dynpro - Never use INCLUDE statement