	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		


Suivi des versions-révisions et des validations du document.			
<p>Ce document annule et remplace tout document diffusé de version-révision antérieure.</p> <p>Dès réception de ce document, les destinataires ont pour obligation de détruire les versions-révisions antérieures, toutes les copies, et de les remplacer par cette version.</p> <p>Si les versions-révisions antérieures sont conservées pour mémoire, les destinataires doivent s'assurer qu'elles ne peuvent être confondues avec cette présente version-révision dans leur usage courant.</p>			
Version.	Date.	Auteurs.	Création, modification ou validation.
A	19 nov. 2003	JPD.	Création.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		


1 Tables

1.1 Table des matières

1	Tables	2
1.1	Table des matières	2
1.2	Table des illustrations	4
2	Références	6
2.1	Glossaire	6
2.2	Ressources	6
3	Introduction	7
3.1	Objet du document	7
3.2	Audience	7
3.3	Pré-requis	7
4	Dénominations	8
4.1	Préfixes	8
4.1.1	Préfixe d'un module	8
4.1.2	Paquets de définitions	9
4.2	Paramètres	10
5	Définitions générales	12
5.1	Constantes	12
5.2	Enumérés	12
5.3	Types de données élémentaires	12
6	Appels	13
6.1	Paramètres	13
6.1.1	Prototype	13
6.1.2	Temporaire des paramètres	13
6.1.3	Normalisation des appels	14
6.2	Appels et co-appels	16
6.3	Foncteur	17
6.3.1	Appel	17
6.3.2	Co-appel	20
6.3.3	Dénomination des méthodes particulières	22
6.4	Opérateurs	22
6.5	Déroulement d'un appel	24
7	Types ou interfaces	25
7.1	Propriétés d'un type ou d'une interface	25
7.1.1	Héritage ou implémentation	26
7.1.2	Types ou interfaces polymorphes	26
7.2	Méthodes d'un type ou d'une interface	28
7.2.1	Méthodes génériques du type objet	29
7.2.2	Dictionnaire de données et de traitements	35
7.2.3	Liaison avec Up! Virtual Technical Machine	38
7.2.4	Méthode de lecture de l'objet	39
7.2.5	Méthode d'index	40
7.2.6	Types ou interfaces polymorphes	41
7.2.7	Méthode de cohérence	43
7.3	Déclaration d'un type ou d'une interface public	44
7.3.1	Héritage ou implémentation	44
7.4	Propriétés virtuelles	44
8	Variables	46
8.1	Variables globales	46
8.2	Variables locales	48
8.3	Variables virtuelles	49


	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

9	Exceptions.....	50
10	Files d'attente.....	51
11	Entrepôts	52
12	Modules.....	54
12.1	Propriétés d'un module	54
12.2	Méthodes d'un module	54
12.2.1	Liaison avec Up ! Virtual Technical Machine.....	57
12.2.2	Dictionnaire de données et de traitements	59
12.2.3	Adaptateurs du module	64


	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

1.2 Table des illustrations

Tableau 1 – Préfixes des modules d'Up ! Application System	9
Tableau 2 – Paramètres des dénominations génériques	11
Texte 3 – Exemple de déclaration d'une constante	12
Texte 4 – Exemple de déclaration d'une énuméré	12
Tableau 5 – Types de données élémentaires	12
Texte 6 – Exemple de déclaration du type du temporaire des paramètres d'un appel	14
Tableau 7 – Codification des formes de passage de paramètres	14
Tableau 8 – Codification des formes des prototypes	15
Texte 9 – Exemple d'un co-appel	17
Texte 10 – Exemple d'emploi du foncteur d'un appel	19
Texte 11 – Exemple d'emploi du foncteur d'un co-appel	21
Texte 12 – Dénomination des méthodes particulières	22
Tableau 13 – Codification des opérateurs	24
Texte 14 – Exemple de déclaration des propriétés d'un type ou d'une interface	25
Texte 15 – Exemple de déclaration de la version-révision courante des propriétés d'un type ou d'une interface	26
Texte 16 – Exemple d'héritage d'un type ou d'implémentation d'une interface – Propriétés	26
Texte 17 – Exemple de déclaration de la partie polymorphe d'un type – Propriétés	27
Texte 18 – Exemple de déclaration des méthodes d'un type ou d'une interface	28
Texte 19 – Exemple de déclaration de la version-révision courante des méthodes d'un type ou d'une interface	29
Texte 20 – Méthodes du type Objet	29
Texte 21 – Exemple d'un allocateur explicite	30
Texte 22 – Exemple d'un allocateur explicite de ressource	30
Texte 23 – Exemple d'un constructeur implicite	30
Texte 24 – Exemple d'un destructeur	31
Texte 25 – Exemple d'un libérateur explicite	31
Texte 26 – Exemple d'un libérateur explicite de ressource	31
Texte 27 – Exemple d'un opérateur d'affectation	32
Texte 28 – Exemple d'un opérateur d'égalité	32
Texte 29 – Exemple d'un opérateur de différence	32
Texte 30 – Exemple d'une méthode Cloner	33
Texte 31 – Exemple d'une méthode Contrainte	34
Texte 32 – Exemple d'une méthode Exporter	34
Texte 33 – Exemple d'une méthode Importer	35
Tableau 34 – Dictionnaire de données et de traitements – Types ou interfaces	35
Texte 35 – Exemple d'une méthode EcrirePropriete	36
Texte 36 – Exemple d'une méthode EnumererMethodes	36
Texte 37 – Exemple d'une méthode EnumererProprietes	37
Texte 38 – Exemple d'une méthode IncrementerDecrementerPropriete	37
Texte 39 – Exemple d'une méthode LirePropriete	38
Texte 40 – Exemple d'une méthode SupprimerPropriete	38
Tableau 41 – Liaison avec Up ! Virtual Technical Machine – Types ou interfaces	38
Texte 42 – Exemple d'une méthode NePlusUtiliserObjetsChamps	39
Texte 43 – Exemple d'une méthode UtiliserObjetsChamps	39
Texte 44 – Exemple de déclaration d'une méthode de lecture de l'objet type	40
Texte 45 – Exemple de déclaration de la méthode d'index	41
Texte 46 – Exemple de déclaration d'une méthode de calcul du cas	42
Texte 47 – Exemple de déclaration d'une méthode de mise à jour du cas	43
Texte 48 – Exemple de déclaration d'une méthode de cohérence	43
Texte 49 – Exemple de déclaration d'une méthode de lecture de l'objet	44
Texte 50 – Exemple de déclaration d'une variable globale	48
Texte 51 – Exemple de déclaration d'un type du temporaire des variables locales d'un appel	49
Texte 52 – Exemple de déclaration d'une méthode de lecture de l'objet exception	50
Texte 53 – Exemple de déclaration d'une méthode de lecture de l'objet file d'attente	51
Texte 54 – Exemple de déclaration d'une méthode de lecture de l'objet exception	53
Texte 55 – Exemple de déclaration des propriétés d'un module	54
Texte 56 – Exemple de déclaration des méthodes d'un module	56
Texte 57 – Exemple de déclaration de la version-révision courante des méthodes d'un module	57
Tableau 58 – Liaison avec Up ! Virtual Technical Machine – Modules	57
Texte 59 – Exemple d'une méthode ChargerPersistence	58
Texte 60 – Exemple d'une méthode EnregistrerPersistence	58
Texte 61 – Exemple d'une méthode ExecuterAppelDistant	58
Texte 62 – Exemple d'une méthode LireEnteteDonneesModule	59
Texte 63 – Exemple d'une fonction principale	59
Texte 64 – Exemple d'une méthode RechercherImplementationInterface	59
Tableau 65 – Dictionnaire de données et de traitements – Modules	60

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

Texte 66 – Exemple d'une méthode EcrireVariable	60
Texte 67 – Exemple d'une méthode EnumererAppels	61
Texte 68 – Exemple d'une méthode EnumererEntrepots.....	61
Texte 69 – Exemple d'une méthode EnumererEnumeres.....	61
Texte 70 – Exemple d'une méthode EnumererExceptions	62
Texte 71 – Exemple d'une méthode EnumererTypes	62
Texte 72 – Exemple d'une méthode EnumererValeursEnumeres	62
Texte 73 – Exemple d'une méthode EnumererVariables	63
Texte 74 – Exemple d'une méthode LireVariable	63
Texte 75 – Exemple d'une méthode IncrementerDecrementerVariable.....	63
Tableau 76 – Méthodes de rappel des adaptateurs	64
Texte 77 – Exemple d'une méthode ChercherTypeClientCorba.....	64
Texte 78 – Exemple d'une méthode ChercherTypeClientDCom	65
Texte 79 – Exemple d'une méthode ChercherTypeClientJava	65
Texte 80 – Exemple d'une méthode ChercherTypeServeurCorba	65
Texte 81 – Exemple d'une méthode ChercherTypeServeurDCom	66
Texte 82 – Exemple d'une méthode ChercherTypeServeurJava	66

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		


2 Références

2.1 Glossaire

Liste des définitions des termes employés.	
Ce tableau recense tous les termes, les concepts particuliers ainsi que les abréviations employés dans ce document.	
Terme, concept, abrégé.	Définition du terme, du concept ou de l'abréviation.
Appel	Voir page 16.
Co-appel	Voir page 16.

2.2 Ressources

Liste des documents applicables et en référence.		
Un document est applicable à partir du moment où son contenu est validé et que l'activité ou le projet fait partie de son périmètre d'application. Il est obligatoire d'appliquer son contenu.		
Un document est en référence à partir du moment où son contenu n'est pas validé ou que l'activité ou le projet ne fait partie de son périmètre d'application. Il est recommandé d'appliquer son contenu mais cela n'est pas obligatoire.		
Un document applicable est indiqué par A1, A2, A3 , etc. Un document en référence est indiqué par R1, R2, R3 , etc.		
Index.	Nom du document.	Commentaire.
A1	UpComp-Plan Qualité-000005	Méthode documentaire.
A2	UpComp-UpsVm-000002	Plan de programmation.
A3	UpComp-UpsVm-000003	Méthode de programmation en C-- .
A4	UpComp-UpsVm-000004	Spécification technique du module UpsVm .
A5	UpComp-UpsMod-000003	Spécification technique du module UpsMod .
A6	UpComp-UpsSys-000003	Spécification technique du module UpsSys .
A7	UpComp-UpsMat-000003	Spécification technique du module UpsMat .

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

3 Introduction

3.1 Objet du document

L'objet de se documenter est de présenter la norme d'écriture d'un module de la sorte qu'il soit directement exploitable par les robots suivants :

- **Up ! Compiler.**
Pour fabriquer de nouveaux modules natifs les utilisant.
- **Up ! Virtual Technical Machine.**
Pour exécuter ces modules.
- **Up ! Engine.**
Pour interpréter de nouveaux modules les utilisant en émulant le processeur virtuel **P32**.

Ce plan d'écriture des modules s'applique à tous les modules écrits directement en **C--** d'**Up ! Application System**, en particulier les suivants :

- **Up ! Kernel.**
- **Up ! Mathematical.**
- **Up ! Module.**
- **Up ! Natural Language Support.**
- **Up ! Network.**
- **Up ! Object Management System.**
- **Up ! Object Request Broker.**
- **Up ! Security Management System.**
- **Up ! Starter.**
- **Up ! System.**

Ces normes sont à appliquer sans réserve.


3.2 Audience

Ce document s'adresse à tout ingénieur chargé de l'étude, de la réalisation ou de la maintenance d'un module natif.

3.3 Pré-requis

Le pré-requis est la connaissance de :

- Le **Plan de programmation** [A2].
- La **Méthode de programmation en C--** [A3].
- Les interfaces avec la machine d'**Up ! Virtual Technical Machine** [A4], [A5], [A6] et [A7].

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

4 Dénominations

4.1 Préfixes


4.1.1 Préfixe d'un module

Le préfixe d'un module est un mot-clé identifiant techniquement le module connu sous son nom commercial. Il est composé de la sorte :

- 3 lettres réservées à l'identification de l'éditeur de logiciels.
Par exemple, **Ups** pour **Up ! Software**.
- 3 lettres réservées à l'identification du module chez l'éditeur.
Par exemple, **Krn** pour **Up ! Kernel**.

Voici les préfixes des modules d'**Up ! Application System** en 1.0.0 :

Préfixe	Nom commercial
Ups5gl.	<i>Up ! Fifth Generation Language – Basic.</i>
UpsAna.	<i>Up ! Analyzer.</i>
UpsCom.	<i>Up ! Component Object Module.</i>
UpsCom.	<i>Up ! Java.</i>
UpsCp1.	<i>Up ! Compiler – Level 1.</i>
UpsCrb.	<i>Up ! Corba.</i>
UpsGc1.	<i>Up ! C / C++ Generator – Level 1.</i>
UpsGcb.	Adaptateur serveur Corba pour Up ! C / C++ Generator .
UpsGcm.	Adaptateur serveur DCom pour Up ! C / C++ Generator .
UpsGjv.	Adaptateur serveur Java pour Up ! C / C++ Generator .
UpsGl1.	Adaptateur Up ! 5GL pour Up ! C / C++ Generator .
UpsGr1.	<i>Up ! Grammar – Level 1.</i>
UpsGun.	Adaptateur serveur Up ! Network pour Up ! C / C++ Generator .
Upslcb.	Adaptateur client Idl Corba pour Up ! C / C++ Generator .
Upslcm.	Adaptateur client Idl DCom pour Up ! C / C++ Generator .
Upsljb.	Adaptateur client Java pour Up ! C / C++ Generator .
Upslrc.	Adaptateur client Interface Repository Corba pour Up ! C / C++ Generator .
UpsKrn.	<i>Up ! Kernel.</i>
UpsLg1.	<i>Up ! Fifth Generation Language – Level1.</i>
UpsMat.	<i>Up ! Mathematical.</i>
UpsMod.	<i>Up ! Module.</i>
UpsMsn.	Adaptateur Microsoft Network pour Up ! Network .
UpsNap.	Adaptateur Named Pipes pour Up ! Network .

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		


Préfixe	Nom commercial
UpsNet.	<i>Up ! Network.</i>
UpsNls.	<i>Up ! Natural Language Support.</i>
UpsOms.	<i>Up ! Object Management System.</i>
UpsOrb.	<i>Up ! Object Request Broker.</i>
UpsSec.	<i>Up ! Security Manager.</i>
UpsStr.	<i>Up ! Starter.</i>
UpsSys.	<i>Up ! System.</i>
UpsTcp.	Adaptateur <i>Transmission Control Protocol</i> pour <i>Up ! Network.</i>
UpsTlb.	Adaptateur client <i>Tlb DCom</i> pour <i>Up ! C / C++ Generator.</i>
UpsVm.	<i>Up ! Virtual Machine.</i>
UpsWin	<i>Up ! Windows.</i>

Tableau 1 – Préfixes des modules d'Up ! Application System

4.1.2 Paquets de définitions

Il existe un paquet par accès à l'information :


- **Le paquet des définitions publiques des composants du module.**
Elles sont déclarées dans l'interface d'extension **upi** du module.
Le préfixe est la chaîne de caractères **vide ""**.
- **Le paquet des définitions protégées d'un composant du module.**
Elles sont déclarées dans l'interface d'extension **upi** du composant et elles ne sont pas déclarées dans l'interface d'extension **upi** du module.
Le préfixe est la chaîne de caractères **"Pro\$NumeroDuComposant\$_"**.
- **Le paquet des définitions privées d'un composant du module.**
Elles ne sont déclarées ni dans l'interface d'extension **upi** du composant et ni dans l'interface d'extension **upi** du module.
Le préfixe est la chaîne de caractères **"Pri\$NumeroDuComposant\$_"**.
- **Le paquet des définitions locales à une procédure, une fonction ou une méthode.**
Elles ne sont déclarées ni dans l'interface d'extension **upi** du composant et ni dans l'interface d'extension **upi** du module.
Le préfixe est la chaîne de caractères **"Loc\$NumeroDuComposant\$_"**.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

4.2 Paramètres


Ces paramètres des dénominations génériques sont employés dans la suite du document :

Symbole	Sémantique
\$NumeroAppel\$	Séquence de déclaration d'un appel. Il y a une séquence pour le module. Elle commence à 1 .
\$NumeroCas\$	Séquence de déclaration d'un cas d'une partie polymorphe d'un type ou d'une interface. Il y a une séquence par partie polymorphe. Elle commence à 1 .
\$NumeroChampCache\$	Séquence de déclaration d'un champ caché par type ou interface. Il y a une séquence par type ou par interface. Elle commence à 1 .
\$NumeroConstante\$	Séquence de déclaration d'une constante. Il y a une séquence par accès – public, privé, protégé. Elles commencent à 1 .
\$NumeroEnumere\$	Séquence de déclaration d'un énuméré. Il y a une séquence par accès – public, privé, protégé. Elles commencent à 1 .
\$NumeroException\$	Séquence de déclaration d'une exception. Il y a une séquence par accès – public, privé, protégé. Elles commencent à 1 .
\$NumeroFileDAttente\$	Séquence de déclaration d'une file d'attente. Il y a une séquence par accès – public, privé, protégé. Elles commencent à 1 .
\$NumeroMethode\$	Séquence de déclaration d'une méthode. Il y a une séquence par type ou interface. Elle commence à 1 .
\$NumeroParametre\$	Séquence de déclaration d'un paramètre. Il y a une séquence par prototype. Elle commence à 1 .
\$NumeroPropriete\$	Séquence de déclaration d'une propriété d'un type ou d'une interface. Il y a une séquence par type ou par interface. Elle commence à 1 . Les déclarations Cas ... Fin Cas sont parcourues en in-fixé.
\$NumeroPrototype\$	Séquence de déclaration d'un prototype. Il y a une séquence par procédure, fonction et méthode. Elles commencent à 1 .
\$NumeroSelon\$	Séquence de déclaration de la partie polymorphe d'un type ou d'une interface. Il y a une séquence par type ou par interface. Elle commence à 1 .
\$NumeroType\$	Séquence de déclaration d'un type ou d'une interface. Il y a une séquence par accès – public, privé, protégé. Elles commencent à 1 .
\$NumeroValeurEnumere\$	Séquence de déclaration d'une valeur d'un énuméré. Il y a une séquence par énuméré. Elle commence à 1 .
\$NumeroVariable\$	Séquence de déclaration d'une variable. Il y a une séquence par accès et une séquence par appel pour les variables locales. Elles commencent à 1 .

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

\$PrefixeModule\$	Préfixe du module. Par exemple UpsKrn pour Up ! Kernel .
\$PrefixePaquet\$	Préfixe du paquet. Par exemple " pour Public .
\$VersionRevision\$	Numéro de version, de révision et de correction de l'application. Par exemple 1_0_0 .

Tableau 2 – Paramètres des dénominations génériques

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

5 Définitions générales

5.1 Constantes

Les constantes sont des définitions dont le nom générique est :

UpsCon\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroConstante\$

Voici un exemple :

```
#define UpsConUpsMat_1 2.718281828459
/* Constante : e */
#define UpsConUpsMat_2 3.14159265359
/* Constante : Pi */
```

Texte 3 – Exemple de déclaration d'une constante

5.2 Enumérés

Les énumérés sont des définitions dont le nom générique est :

UpsEnu\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroEnumere\$_\$NumeroValeurEnumere\$

Voici un exemple :

```
/*-----*/
/* Enumere Booleen */
/*-----*/
#define UpsEnuUpsKrn_1_1 0
/* Valeur : Faux */
#define UpsEnuUpsKrn_1_2 1
/* Valeur : Vrai */
```

Texte 4 – Exemple de déclaration d'une énuméré


5.3 Types de données élémentaires

Voici les types de données élémentaires et notamment les scalaires :

Définition en <i>Up! 5GL</i> .	Equivalence en <i>C--</i> .
Adresse d'un objet.	<i>TypUpsVmAdresse.</i>
Scalaire énuméré court – Au plus 255 valeurs.	<i>TypUpsVmUnsignedChar.</i>
Scalaire énuméré long – Au moins 256 valeurs.	<i>TypUpsVmUnsignedShort.</i>
Scalaire nombre entier.	<i>TypUpsVmLong.</i>
Scalaire nombre réel.	<i>TypUpsVmDouble.</i>

Tableau 5 – Types de données élémentaires

Les types de données sont dénombrés dans l'énuméré *EnuUpsVmTypeDeDonnee*.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

6 Appels

6.1 Paramètres

6.1.1 Prototype

Les paramètres d'un appel sont normalisés de la sorte :

- **Session.**
- **Adresse de l'objet résultat.**
Si l'appel est une fonction ou une méthode retournant un objet. Sinon, il n'existe pas.
- **Adresse de l'objet d'application.**
Si l'appel est une méthode. Sinon, il n'existe pas.
- **Référence du temporaire des paramètres transmis.**
Si l'appel possède des paramètres. Sinon, elle n'existe pas.

6.1.2 Temporaire des paramètres

Les paramètres d'un appel sont regroupés dans un temporaire qui est transmis par référence à celui-ci.


Voici la convention de dénomination de la structure :

- **Pour un appel de procédure ou de fonction.**
TypPrm\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroAppel\$_\$NumeroPrototype\$
- **Pour une méthode.**
Cela varie selon l'accès à la méthode :
 - **Méthode publique.**
TypPrm\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$
 - **Méthode protégée.**
TypPrm\$PrefixeModule\$_\$PrefixePaquet\$Pro_ \$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$
 - **Méthode privée.**
TypPrm\$PrefixeModule\$_\$PrefixePaquet\$Pri_ \$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

Les paramètres sont énumérés par ordre de déclaration dans le prototype, de la gauche vers la droite. Voici la convention de dénomination d'un paramètre :

UpsVar\$NumeroParametre\$

Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****
typedef struct typprmpskrn_2_4_7
/* Objet : Parametre de Caractere.Constructeur(E:Nul Ou Entier, */
/* Filtre:Nul Ou Caractere, PointVirgule:Nul Ou Caractere) */
/*****
{
TypUpsVmAdresse UpsVar1;
/* E : Nul Ou Entier. */
TypUpsVmAdresse UpsVar2;
/* Filtre : Nul Ou Caractere. */
TypUpsVmAdresse UpsVar3;
/* PointVirgule : Nul Ou Caractere. */
} TypPrmUpsKrn_2_4_7;

```

Texte 6 – Exemple de déclaration du type du temporaire des paramètres d'un appel

6.1.3 Normalisation des appels


Compte tenu de la normalisation des types de données, il est possible de normaliser les formes de passage de paramètres d'un appel et de les dénombrer, ce qui est fait par l'énuméré **EnuUpsVmModePassageParametre** :

Enumération.	Déclaration en <i>Up ! 5GL.</i>
PR_EnumereCourtParEntree.	E : EnuAMoinsDe255Valeurs Entree
PR_EnumereLongParEntree.	E : EnuAPlusDe256Valeurs Entree
PR_EntierParEntree.	E : Entier Entree
PR_ReelParEntree.	R : Reel Entree
PR_ObjetParEntree.	O : Objet Entree
PR_EnumereCourtParSortie.	E : EnuAMoinsDe255Valeurs Sortie
PR_EnumereLongParSortie.	E : EnuAPlusDe256Valeurs Sortie
PR_EntierParSortie.	E : Entier Sortie
PR_ReelParSortie.	R : Reel Sortie
PR_ObjetParSortie.	O : Objet Sortie

Tableau 7 – Codification des formes de passage de paramètres

Compte tenu de la normalisation des types de données et du format d'un prototype, il est possible de normaliser les types d'appel et de les dénombrer, ce qui est fait par l'énuméré **EnuUpsVmModeDAppel** :

Enumération.	Déclaration en <i>Up ! 5GL.</i>
TA_ProcedureSansParametre.	Procedure P();
TA_ProcedureAvecParametre.	Procedure P(PP : Objet, ...);
TA_FonctionEnumereCourtSansParametre.	Fonction F() Retourner Enumere;
TA_FonctionEnumereCourtAvecParametre.	Fonction F(PP : Objet, ...) Retourner Enumere;


	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

Enumération.	Déclaration en <i>Up ! 5GL</i> .
TA_FonctionEnumereLongSansParametre.	Fonction F() Retourner Booleen;
TA_FonctionEnumereLongAvecParametre.	Fonction F(PP : Objet, ...) Retourner Enumere;
TA_FonctionEntierSansParametre.	Fonction F() Retourner Entier;
TA_FonctionEntierAvecParametre.	Fonction F(PP : Objet, ...) Retourner Entier;
TA_FonctionReelSansParametre.	Fonction F() Retourner Reel;
TA_FonctionReelAvecParametre.	Fonction F(PP : Objet, ...) Retourner Reel;
TA_FonctionObjetSansParametre.	Fonction F() Retourner Objet;
TA_FonctionObjetAvecParametre.	Fonction F(PP : Objet, ...) Retourner Objet;
TA_MethodeSansParametre.	Procedure T.P();
TA_MethodeAvecParametre.	Procedure T.P(PP : Objet, ...);
TA_MethodeEnumereCourtSansParametre.	Fonction T.F() Retourner Booleen;
TA_MethodeEnumereCourtAvecParametre.	Fonction T.F(PP : Objet, ...) Retourner Enumere;
TA_MethodeEnumereLongSansParametre.	Fonction T.F() Retourner Booleen;
TA_MethodeEnumereLongAvecParametre.	Fonction T.F(PP : Objet, ...) Retourner Enumere;
TA_MethodeEntierSansParametre.	Fonction T.F() Retourner Entier;
TA_MethodeEntierAvecParametre.	Fonction T.F(PP : Objet, ...) Retourner Entier;
TA_MethodeReelSansParametre.	Fonction T.F() Retourner Reel;
TA_MethodeReelAvecParametre.	Fonction T.F(PP : Objet, ...) Retourner Reel;
TA_MethodeObjetSansParametre.	Fonction T.F() Retourner Objet;
TA_MethodeObjetAvecParametre.	Fonction T.F(PP : Objet, ...) Retourner Objet;

Tableau 8 – Codification des formes des prototypes

A chaque type d'appel correspond à type de pointeur de traitements donnant son prototype :

- Pour les procédures et les fonctions.
Il y a :
 - ***TypUpsVmProcedureSansParametre.***
 - ***TypUpsVmProcedureAvecParametre.***
 - ***TypUpsVmFonctionEnumereCourtSansParametre.***
 - ***TypUpsVmFonctionEnumereCourtAvecParametre.***
 - ***TypUpsVmFonctionEnumereLongSansParametre.***
 - ***TypUpsVmFonctionEnumereLongAvecParametre.***
 - ***TypUpsVmFonctionEntierSansParametre.***
 - ***TypUpsVmFonctionEntierAvecParametre.***

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

- *TypUpsVmFonctionReelSansParametre.*
- *TypUpsVmFonctionReelAvecParametre.*
- *TypUpsVmFonctionObjetSansParametre.*
- *TypUpsVmFonctionObjetAvecParametre.*
- Pour les méthodes.
Il y a :
 - *TypUpsVmConstructeurAvecParametre.*
 - *TypUpsVmMethodeSansParametre*
 - *TypUpsVmMethodeAvecParametre.*
 - *TypUpsVmMethodeEnumereCourtSansParametre.*
 - *TypUpsVmMethodeEnumereCourtAvecParametre.*
 - *TypUpsVmMethodeEnumereLongSansParametre.*
 - *TypUpsVmMethodeEnumereLongAvecParametre.*
 - *TypUpsVmMethodeEntierSansParametre.*
 - *TypUpsVmMethodeEntierAvecParametre.*
 - *TypUpsVmMethodeReelSansParametre.*
 - *TypUpsVmMethodeReelAvecParametre.*
 - *TypUpsVmMethodeObjetSansParametre.*
 - *TypUpsVmMethodeObjetAvecParametre.*

6.2 Appels et co-appels

& Un **appel** exécute le code définit par sa sémantique. Son point d'entrée est variable en fonction de son prototype.

& Un **co-appel** décrit le prototype d'un appel. Son point d'entrée est fixe et il correspond au type *TypUpsVmCoAppel*. Voici un exemple de co-appel :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****
TypUpsVmUnsignedChar UpsVmAPI UpsCoTrtUpsKrn_9_1(TypUpsVmSession
 *Session, TypUpsVmShort NumeroParametre, TypUpsVmAdresse
 *AdresseResultat, TypPrmUpsVmEnumererParametres *UpsPrm)
/* Objet : DebuterImportation(Lire:Fonction(TaillePaquet:Entier)*/
/* Retourner Binaire, P:Nul Ou Entrepot=Nul). */
/*****
{
if (AdresseResultat&&UpsPrm)
{
UpsKrnChargerDictionnaire(Session);
return(*UpsKrnIntTrtUpsKrnD->CoTrt_9_1)(Session, NumeroParametre,
AdresseResultat,UpsPrm);
}
switch (NumeroParametre)
{
case 1 :
return(PR_ObjetParEntree);
break;

case 2 :
return(PR_ObjetParEntree);
break;
}
return(0);
}

```

Texte 9 – Exemple d'un co-appel

Par convention :

- Les paramètres sont numérotés à partir de 1.
En suivant l'ordre de déclaration dans le prototype, de la gauche vers la droite.
- Le résultat a pour numéro 0.
Sa forme de paramètre correspond à un paramètre de sortie.

6.3 Foncteur

6.3.1 Appel

Le foncteur d'un appel varie pour une procédure, une fonction ou une méthode :


- Pour un appel de procédure ou de fonction.
 - Le foncteur de l'appel, employé dans le corps du composant dont il fait partie ou dans son interface, porte le nom générique suivant :

UpsTrt\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroAppel\$_\$NumeroPrototype\$

- Le foncteur de l'appel, employé dans l'interface de traitements du module, porte le nom générique suivant :

Trt_ \$NumeroAppel\$_\$NumeroPrototype\$

- Pour une méthode.
Cela varie selon l'accès à la méthode :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

- **Méthode publique.**

- Le foncteur de la méthode publique, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsMth\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

- Le foncteur de la méthode publique, employé dans l'interface de traitements du module, porte le nom générique suivant :

Mth_\$_NumeroAppel\$_\$NumeroPrototype\$

- **Méthode protégée.**

- Le foncteur de la méthode protégée, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsMth\$PrefixeModule\$_\$PrefixePaquet\$Pro_\$_NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

- Le foncteur de la méthode protégée, employé dans l'interface de traitements du module, porte le nom générique suivant :

Mth_Pro_\$_NumeroAppel\$_\$NumeroPrototype\$

- **Méthode privée.**

- Le foncteur de la méthode privée, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsMth\$PrefixeModule\$_\$PrefixePaquet\$Pri_\$_NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

- Le foncteur de la méthode privée, employé dans l'interface de traitements du module, porte le nom générique suivant :

Mth_Pri_\$_NumeroAppel\$_\$NumeroPrototype\$

Voici un exemple :



Plan d'écriture d'un module

Date rédaction :

16 février 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

```

/*****
static TypUpsVmAdresse UpsVmAPI *UpsMthUpsKrn_1_9_1(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse
    *AdresseObjet, TypPrmUpsKrn_1_9_1 *UpsPrm)
/* Fonction Binaire.Gauche(Taille:Nul Ou Entier) Retourner Nul */
/* Ou Binaire; */
/*****
{
...
}


/*****
typedef struct upstrttypupskrn_1_1_0_0
/* Objet : Methodes du type Binaire. */
/*****
{
...
TypUpsVmAdresse UpsVmAPI>(*Mth_9_1)(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsKrn_1_9_1 *UpsPrm);
/* Fonction Binaire. Gauche(Taille:Nul Ou Entier) Retourner */
/* Nul Ou Binaire; */
...
} UpsTrtTypUpsKrn_1_1_0_0;

/*****
TypUpsVmVoid UpsVmAPI UpsTrtUpsKrn_1_1(TypUpsVmSession *Session,
TypPrmUpsKrn_1_1 *UpsPrm)
/* Objet : ActiverMiseAuPoint(B:Booleen). */
/*****
{
...
}

/*****
typedef struct typupskrntraitements_1_0_0
/* Objet: Interface des traitements de Ups Krn. */
/*****
{
...
TypUpsVmVoid UpsVmAPI (*Trt_1_1)(TypUpsVmSession *Session,
    TypPrmUpsKrn_1_1 *UpsPrm);
/* Objet : ActiverMiseAuPoint. */
...
} *TypUpsKrnTraitements_1_0_0;

```

Texte 10 – Exemple d'emploi du foncteur d'un appel

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

6.3.2 Co-appel

Le foncteur d'un co-appel varie pour une procédure, une fonction ou une méthode :

- Pour un co-appel de procédure ou de fonction.

- Le foncteur du co-appel, employé dans le corps du composant dont il fait partie ou dans son interface, porte le nom générique suivant :

`UpsCoTrt$PrefixeModule$_$PrefixePaquet$$NumeroAppel$_$NumeroPrototype$`

- Le foncteur du co-appel, employé dans l'interface de traitements du module, porte le nom générique suivant :

`CoTrt_$_NumeroAppel$_$NumeroPrototype$`

- Pour une co-méthode.

Cela varie selon l'accès à la co-méthode :

- Co-méthode publique.

- Le foncteur de la méthode publique, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

`UpsCoMth$PrefixeModule$_$PrefixePaquet$$NumeroType$_$NumeroAppel$_$NumeroPrototype$`

- Le foncteur de la co-méthode publique, employé dans l'interface de traitements du module, porte le nom générique suivant :

`CoMth_$_NumeroAppel$_$NumeroPrototype$`

- Co-méthode protégée.

- Le foncteur de la co-méthode protégée, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

`UpsCoMth$PrefixeModule$_$PrefixePaquet$Pro_$_NumeroType$_$NumeroAppel$_$NumeroPrototype$`

- Le foncteur de la co-méthode protégée, employé dans l'interface de traitements du module, porte le nom générique suivant :

`CoMth_Pro_$_NumeroAppel$_$NumeroPrototype$`

- Co-méthode privée.

- Le foncteur de la co-méthode privée, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

`UpsCoMth$PrefixeModule$_$PrefixePaquet$Pri_$_NumeroType$_$NumeroAppel$_$NumeroPrototype$`

- Le foncteur de la co-méthode privée, employé dans l'interface de traitements du module, porte le nom générique suivant :

`CoMth_Pri_$_NumeroAppel$_$NumeroPrototype$`

Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
static TypUpsVmAdresse UpsVmAPI *UpsCoMthUpsKrn_1_9_1(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse
    *AdresseObjet, TypPrmUpsKrn_1_9_1 *UpsPrm)
/* Fonction Binaire.Gauche(Taille:Nul Ou Entier) Retourner Nul */
/* Ou Binaire; */
/*****/
{
...
}


/*****/
typedef struct upstrttypupskrn_1_1_0_0
/* Objet : Methodes du type Binaire. */
/*****/
{
...
TypUpsVmAdresse UpsVmAPI>(*CoMth_9_1)(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsKrn_1_9_1 *UpsPrm);
/* Fonction Binaire. Gauche(Taille:Nul Ou Entier) Retourner */
/* Nul Ou Binaire; */
...
} UpsTrtTypUpsKrn_1_1_0_0;

/*****/
TypUpsVmVoid UpsVmAPI UpsCoTrtUpsKrn_1_1(TypUpsVmSession *Session,
TypPrmUpsKrn_1_1 *UpsPrm)
/* Objet : ActiverMiseAuPoint(B:Booleen). */
/*****/
{
...
}

/*****/
typedef struct typupskrntraitements_1_0_0
/* Objet: Interface des traitements de Ups Krn. */
/*****/
{
...
TypUpsVmVoid UpsVmAPI(*CoTrt_1_1)(TypUpsVmSession *Session,
    TypPrmUpsKrn_1_1 *UpsPrm);
/* Objet : ActiverMiseAuPoint. */
...
} *TypUpsKrnTraitements_1_0_0;

```

Texte 11 – Exemple d'emploi du foncteur d'un co-appel

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

6.3.3 Dénomination des méthodes particulières

Les méthodes suivantes possèdent un foncteur imposé parce qu'il correspond à un sélecteur du type *TypUpsVmEnteteMethodes* :

Foncteur.	Sémantique.
Allouer.	<i>Allouer()</i> ;
Constructeur.	<i>Constructeur()</i> ;
Destructeur.	<i>Destructeur()</i> ;
Liberer.	<i>Liberer()</i> ;
MethodeAffecter.	<i>Operateur</i> =(P : <i>Nul Ou</i> Objet);
MethodeAllouerRessource.	<i>AllouerRessource()</i> ;
MethodeCloner.	<i>Cloner</i> (EntrepotCible : <i>Nul Ou</i> Entrepot= <i>Nul</i> , Profondeur : ProfondeurClonage= <i>ClonageObjetSeul</i>) <i>Retourner Nul Ou</i> Objet;
MethodeCoherence.	<i>Contrainte()</i> pour un objet.
MethodeCoherenceEntier.	<i>Contrainte()</i> pour un entier.
MethodeCoherenceReel.	<i>Contrainte()</i> pour un réel.
MethodeDifferent.	<i>Operateur</i> !=(P : <i>Nul Ou</i> Objet) <i>Retourner Nul Ou</i> Booleen;
MethodeEgal.	<i>Operateur</i> ==(P : <i>Nul Ou</i> Objet) <i>Retourner Nul Ou</i> Booleen;
MethodeExporter.	<i>Procedure</i> Exporter();
MethodeImporter.	<i>Fonction</i> Importer(P : <i>Nul Ou</i> Entrepot= <i>Nul</i>) <i>Retourner Nul Ou</i> Objet;
MethodeLibererRessource.	<i>LibererRessource()</i> ;

Texte 12 – Dénomination des méthodes particulières

6.4 Opérateurs

Un opérateur est une méthode sans dénomination particulière. En revanche, les opérateurs sont dénombrés par l'énuméré *EnuUpsVmOperateur* :

Enumération.	Opérateur.
Opérations arithmétiques.	
OP_Additionner.	+
OP_Soustraire.	-
OP_Multiplier.	/
OP_Diviser.	/
OP_DiviserEntier.	\
OP_Moduler.	%
OP_Puissance.	^
OP_EtBit.	&



Plan d'écriture d'un module

Date rédaction :


16 février 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

Enumération.	Opérateur.
OP_OuBit.	
OP_NonBit	~
OP_InferieurInferieur	<<
OP_SuperieurSuperieur	>>
Opérations de comparaison.	
OP_Egal.	=
OP_Different.	!=
OP_Inferieur.	<
OP_Superieur.	>
OP_InferieurOuEgal.	<=
OP_SuperieurOuEgal	>=
OP_Affecter	=
Opérations composées.	
OP_AdditionnerAffecter	+=
OP_SoustraireAffecter	-=
OP_MultiplierAffecter	*=
OP_DiviserAffecter	/=
OP_DiviserEntierAffecter	\=
OP_ModulerAffecter	%=
OP_PuissanceAffecter	^=
OP_EtBitAffecter	&=
OP_OuBitAffecter	=
OP_InferieurInferieurAffecter	<<=
OP_SuperieurSuperieurAffecter	>>=
Opérations booléennes.	
OP_Non	Non
OP_Et	Et
OP_Ou	Ou
OP_OuExclusif	OuExclusif
Opérations ensemblistes.	
OP_Union	Union
OP_Intersection	Intersection
OP_Soustraction	Soustraction
Opérations diverses.	

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		


Enumération.	Opérateur.
OP_EtSi	<code>EtSi</code>
OP_OuSinon	<code>OuSinon</code>
OP_Tableau	<code>[]</code>
OP_Evaluation	<code>()</code>
OP_MiseEnFile	<code> </code>

Tableau 13 – Codification des opérateurs

6.5 Déroulement d'un appel

Le déroulement d'un appel de procédure, de fonction ou de méthode est le suivant :

- L'appelant déclare une variable locale pour le temporaire des paramètres s'ils existent.
- L'appelant déclare une variable locale pour le temporaire du résultat s'il existe.
- S'il y a des paramètres :
 - L'appelant transmet les paramètres d'entrée dans le temporaire des paramètres.
 - L'appelant initialise les paramètres de sortie dans le temporaire des paramètres.
- L'appelant appelle l'appelé.
Les paramètres normalisés sont transmis.
- S'il y a des paramètres, l'appelé vérifie les paramètres obligatoires.
- L'appelé exécute le code demandé.
- L'appelé transmet les paramètres de sortie dans le temporaire des paramètres.
- L'appelé transmet le résultat à l'appelant.
- L'appelant récupère les paramètres de sortie depuis le temporaire des paramètres.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

7 Types ou interfaces

7.1 Propriétés d'un type ou d'une interface

Les propriétés d'un type ou d'une interface sont regroupées dans une structure. Il y a une structure par version-révision de l'application.

Voici la convention de sa dénomination :

UpsTyp\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$_\$VersionRevision\$

Les propriétés sont énumérées par ordre de déclaration dans le type ou l'interface. Voici la convention de dénomination d'une propriété :

UpsChamp\$NumeroPropriete\$

Suite aux propriétés, il peut exister des déclarations propriétaires correspondant par exemple aux ressources.

Voici un exemple :

```


/*****
typedef struct upstypupssys_4_1_0_0
/* Objet : Proprietes du type Tache. */
/*****
{
TypUpsVmAdresse UpsChamp1;
/* NomTache : Caractere; */
TypUpsVmUnsignedChar UpsChamp2;
/* EstTerminee : Booleen. */
TypUpsVmLong UpsChamp3;
/* CodeRetour : Entier; */
TypUpsVmAdresse UpsChamp4;
/* TransactionCourante : Nul Ou Transaction; */
TypUpsVmUnsignedChar UpsChamp5;
/* RelancerAutomatiquement : Booleen; */
TypUpsVmAdresse AdresseObjetTransactionPrincipale;
/* Adresse de l'objet de la transaction principale; */
TypUpsSysPrincipalTache PrincipalTache;
/* Fonction principale de la tache. */
...
} UpsTypUpsSys_4_1_0_0;

```

Texte 14 – Exemple de déclaration des propriétés d'un type ou d'une interface

Quand le type est modifié au cours d'un changement de version-révision, il ne peut qu'être étendu en terme de propriétés, et, auquel, les nouvelles sont ajoutées à la suite.

Un alias indique la version-révision courante des propriétés d'un type ou d'une interface. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*-----*/
/* La version courante est la 1.0.0. */
/*-----*/
typedef UpsTypUpsSys_4_1_0_0 UpsTypUpsSys_4;

```

Texte 15 – Exemple de déclaration de la version-révision courante des propriétés d'un type ou d'une interface

7.1.1 Héritage ou implémentation

Un type qui hérite de types comportent autant d'objets cachés qu'il existe de types hérités. Ceux-ci représentent l'objet du type hérité.

Un type qui implémente des interfaces comporte autant d'objets cachés qu'il existe d'interfaces implémentées. Ceux-ci représentent l'objet de l'interface implémentée.

Une interface qui hérite d'interfaces comporte autant d'objets cachés qu'il existe d'interfaces héritées. Ceux-ci représentent l'objet de l'interface héritée.

Ces objets cachés sont déclarés avant les propriétés. En cas de récurrence, le parcours est in-fixé.

Voici la convention de dénomination d'un objet caché :

UpsObjet\$NumeroChampCache\$

Voici un exemple d'implémentation d'une interface :

```

/*****
typedef struct upstypupssys_8_1_0_0
/* Objet : Proprietes du type Fichier : Flux. */
/*****
{
TypUpsVmAdresse UpsObjet1;
/* Objet pour le flux; */
TypUpsVmFlux Fichier;
/* Descripteur du fichier. */
TypUpsVmLong Position;
/* Position dans le fichier suite a la derniere ecriture ou lecture. */
} UpsTypUpsSys_8_1_0_0;

```

Texte 16 – Exemple d'héritage d'un type ou d'implémentation d'une interface – Propriétés

7.1.2 Types ou interfaces polymorphes

Les déclarations **Selon ... Cas ... Fin Cas** sont déclarées dans une union comportant une structure pour chaque cas et une structure le cas par défaut. Voici les conventions de dénomination :

- Union correspondant à un **Selon ... Fin Selon**.

UpsSelon\$NumeroSelon\$

- Structure correspondant à un **Cas ... Fin Cas**.

UpsCas\$NumeroCas\$

- Structure correspondant à un **Defaut ... Fin Defaut**.

UpsDefaut



Plan d'écriture d'un module

Date rédaction :

16 février 2004.

Diffusion restreinte


Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

Voici un exemple de déclaration d'une partie polymorphe d'un type :

```
/* ***** */
typedef struct upstypupssec_3_1_0_0
/* Objet : Proprietes du type Habilitations. */
/* ***** */
{
TypUpsVmAdresse UpsChamp1;
/* HabilitationsMere : Nul Ou Habilitations */
TypUpsVmAdresse UpsChamp2;
/* RolePere : Nul Ou Role */
TypUpsVmUnsignedChar UpsChamp3;
/* Sorte : SorteHabilitation */
union
{
struct
{
TypUpsVmAdresse UpsChamp4;
/* ModuleHabilite : Nul Ou Module */
TypUpsVmAdresse UpsChamp5;
/* HabilitationCycleDeVie : Nul Ou HabilitationUnitaire */
TypUpsVmAdresse UpsChamp6;
/* ListeDHabilitationsAppels:Nul Ou ListeDe HabilitationUnitaire */
TypUpsVmAdresse UpsChamp7;
/* ListeDHabilitationsTypes:Nul Ou ListeDe Habilitations */
TypUpsVmAdresse UpsChamp8;
/* ListeDHabilitationsVariables:Nul Ou ListeDe HabilitationUnitaire */
} UpsCas1;
struct
{
TypUpsVmAdresse UpsChamp9;
/* TypeHabilite : Nul Ou Type */
TypUpsVmAdresse UpsChamp10;
/* HabilitationCycleDeVie : Nul Ou HabilitationUnitaire */
TypUpsVmAdresse UpsChamp11;
/* ListeDHabilitationsMethodes:Nul Ou ListeDe HabilitationUnitaire */
TypUpsVmAdresse UpsChamp12;
/* ListeDHabilitationsProprietes:Nul Ou ListeDe HabilitationUnitaire */
TypUpsVmAdresse UpsChamp13;
/* ListeDHabilitationsObjets:Nul Ou ListeDe Habilitations */
} UpsCas2;
struct
{
TypUpsVmAdresse UpsChamp14;
/* ObjetHabilite : Nul Ou Objet */
TypUpsVmAdresse UpsChamp15;
/* HabilitationCycleDeVie : Nul Ou HabilitationUnitaire */
TypUpsVmAdresse UpsChamp16;
/* ListeDHabilitationsMethodes:Nul Ou ListeDe HabilitationUnitaire */
TypUpsVmAdresse UpsChamp17;
/* ListeDHabilitationsProprietes:Nul Ou ListeDe HabilitationUnitaire */
} UpsCas3;
} UpsSelon1;
} UpsTypUpsSec_3_1_0_0;
```

Texte 17 – Exemple de déclaration de la partie polymorphe d'un type – Propriétés

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

7.2 Méthodes d'un type ou d'une interface

Les méthodes d'un type ou d'une interface sont regroupées dans une structure. Il y a une structure par version-révision de l'application.

Voici la convention de sa dénomination :

UpsTrtTyp\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$_\$VersionRevision\$

Voici le contenu de cette structure :

- **En-tête des méthodes.**
Elles sont regroupées dans le type *TypUpsVmEnteteMethodes*.
- **Les méthodes stricto sensu.**
Elles sont énumérées par ordre de déclaration dans le type ou l'interface. La convention de dénomination d'une méthode est présentée dans la section 6.3 intitulée *Foncteur* page 17.
- **Les co-méthodes stricto sensu.**
Elles sont énumérées par ordre de déclaration dans le type ou l'interface. La convention de dénomination d'une co-méthode est présentée dans la section 6.3 intitulée *Foncteur* page 17.
- **Des éventuelles méthodes propriétaires.**
Par exemple pour la gestion des ressources.

Voici un exemple :

```

/*****
typedef struct upstrttypupssys_5_1_0_0
/* Objet : Methodes du type Synchronisation. */
/*****
{
TypUpsVmEnteteMethodes EnteteMethodes;
/* Entete des methodes. */

TypUpsVmAdresse UpsVmAPI>(*Mth_4_1)(TypUpsVmSession *Session, TypUpsVmAdresse
*AdresseResultat, TypPrmUpsSys_5_4_1 *UpsPrm);
/* Constructeur(Nom:Caractere, AccesDemande:AccesSynchronisation); */
TypUpsVmVoid UpsVmAPI(*Mth_5_1)(TypUpsVmSession *Session, TypUpsVmAdresse *AdresseObjet,
TypPrmUpsSys_5_5_1 *UpsPrm);
/* Objet : Procedure Prendre(Mode:ModeSynchronisation). */
TypUpsVmUnsignedChar UpsVmAPI(*Mth_6_1)(TypUpsVmSession *Session, TypUpsVmAdresse
*AdresseObjet, TypPrmUpsSys_5_6_1 *UpsPrm);
/* Objet : Fonction TenterDePrendre(Mode:ModeSynchronisation) Retourner Booleen; */
TypUpsVmVoid UpsVmAPI(*Mth_7_1)(TypUpsVmSession *Session, TypUpsVmAdresse *AdresseObjet);
/* Objet : Procedure Lacher(). */

TypUpsVmCoAppel CoMth_4_1;
/* Constructeur(Nom:Caractere, AccesDemande:AccesSynchronisation); */
TypUpsVmCoAppel CoMth_5_1;
/* Objet : Procedure Prendre(Mode:ModeSynchronisation). */
TypUpsVmCoAppel CoMth_6_1;
/* Objet : Fonction TenterDePrendre(Mode:ModeSynchronisation) Retourner Booleen; */
TypUpsVmCoAppel CoMth_7_1;
/* Objet : Procedure Lacher(). */
} UpsTrtTypUpsSys_5_1_0_0;


```

Texte 18 – Exemple de déclaration des méthodes d'un type ou d'une interface

M

Quand le type est modifié au cours d'un changement de version-révision, il ne peut qu'être étendu en terme de méthodes, et, auquel cas, les nouvelles méthodes sont ajoutées à la suite.

Un alias indique la version-révision courante des méthodes d'un type ou d'une interface. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*-----*/
/* La version courante est la 1.0.0. */
/*-----*/
typedef UpsTrtTypUpsSys_5_1_0_0 UpsTrtTypUpsSys_5;

```

Texte 19 – Exemple de déclaration de la version-révision courante des méthodes d'un type ou d'une interface

7.2.1 Méthodes génériques du type objet

La première partie du type *TypUpsVmEnteteMethodes* regroupe les méthodes génériques du type **Objet** :


Sélecteur.	Sémantique.
Constructeur.	Constructeur ();
Destructeur.	Destructeur ();
Liberer.	Liberer ();
MethodeAffecter.	Operateur =(P : Nul Ou Objet);
MethodeAllouerRessource.	AllouerRessource ();
MethodeCloner.	Cloner (EntrepotCible : Nul Ou Entrepot= Nul , Profondeur : ProfondeurClonage= ClonageObjetSeul) Retourner Nul Ou Objet;
MethodeCoherence.	Contrainte () pour un objet.
MethodeCoherenceEntier.	Contrainte () pour un entier.
MethodeCoherenceReel.	Contrainte () pour un réel.
MethodeDifferent.	Operateur !=(P : Nul Ou Objet) Retourner Nul Ou Booleen;
MethodeEgal.	Operateur ==(P : Nul Ou Objet) Retourner Nul Ou Booleen;
MethodeExporter.	Procedure Exporter();
MethodeImporter.	Fonction Importer(P : Nul Ou Entrepot= Nul) Retourner Nul Ou Objet;
MethodeLibererRessource.	LibererRessource ();

Texte 20 – Méthodes du type **Objet**

M La méthode **Allouer** ne se situe pas dans l'en-tête des méthodes mais en tant que propriété cachée du type.

7.2.1.1 Allouer

Le type du prototype de la méthode **Allouer** est *TypUpsVmAllouer*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI Allouer(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *AdresseObjetType)
/* Objet : Allouer(); */
/*****/
{
    ...
}

```

Texte 21 – Exemple d'un allocateur explicite

7.2.1.2 AllouerRessource

Le type du prototype de la méthode **AllouerRessource** est *TypUpsVmAllouerRessource*. Voici un exemple :

```

/*****/
TypUpsVmAdresse UpsVmAPI AllouerRessource(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet)
/* Objet : AllouerRessource(); */
/*****/
{
    ...
}

```

Texte 22 – Exemple d'un allocateur explicite de ressource

7.2.1.3 Constructeur

Le type du prototype de la méthode **Constructeur** est *TypUpsVmConstructeur*. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *Constructeur(TypUpsVmSession *Session,
    TypUpsVmAdresse *UpsObjet)
/* Objet : Constructeur(); */
/*****/
{
    ...
}

```

Texte 23 – Exemple d'un constructeur implicite

7.2.1.4 Destructeur

Le type du prototype de la méthode **Destructeur** est *TypUpsVmDestructeur*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmVoid UpsVmAPI Destructeur(TypUpsVmSession *Session,
    TypUpsVmAdresse *UpsObjet)
/* Objet : Destructeur(); */
/*****/
{
    ...
}

```

Texte 24 – Exemple d'un destructeur

7.2.1.5 Liberer

Le type du prototype de la méthode **Liberer** est *TypUpsVmLiberer*. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI Liberer(TypUpsVmSession *Session, TypUpsVmAdresse
    *Objet)
/* Objet : Liberer(); */
/*****/
{
    ...
}

```

Texte 25 – Exemple d'un libérateur explicite

7.2.1.6 LibererRessource

Le type du prototype de la méthode **LibererRessource** est *TypUpsVmLibererRessource*. Voici un exemple :

```


/*****/
TypUpsVmVoid UpsVmAPI LibererRessource(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet)
/* Objet : Liberer(); */
/*****/
{
    ...
}

```

Texte 26 – Exemple d'un libérateur explicite de ressource

7.2.1.7 Opérateur =

Le type du prototype de l'opérateur **=** est *TypUpsVmMethodeAffecter*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeAffecter(TypUpsVmSession *Session,
    TypUpsVmAdresse *UpsObjet, TypUpsVmAdresse *ObjetP)
/* Objet : Operateur =(P:Nul Ou Objet); */
/*****/
{
    ...
}

```

Texte 27 – Exemple d'un opérateur d'affectation

7.2.1.8 Opérateur ==

Le type du prototype de l'opérateur == est *TypUpsVmMethodeEgal*. Voici un exemple :

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEgal(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypUpsVmAdresse *ObjetP)
/* Objet : Operateur =(P:Nul Ou Objet); */
/*****/
{
    ...
}

```

Texte 28 – Exemple d'un opérateur d'égalité

7.2.1.9 Opérateur !=

Le type du prototype de l'opérateur != est *TypUpsVmMethodeDifferent*. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *MethodeDifferent(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypUpsVmAdresse *ObjetP)
/* Objet : Operateur =(P:Nul Ou Objet); */
/*****/
{
    ...
}

```

Texte 29 – Exemple d'un opérateur de différence

7.2.1.10 Méthode Cloner

Le type du prototype de la méthode *Cloner* est *TypUpsVmMethodeCloner*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeCloner(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *AdresseObjet,
    TypPrmUpsVmCloner *UpsPrm)
/* Objet : Cloner(EntrepotCible : Nul Ou Entrepot=Nul, Profondeur :
ProfondeurClonage=ClonageObjetSeul) Retourner Nul Ou Objet; */
/*****/
{
. . .
}

```


Texte 30 – Exemple d'une méthode Cloner

7.2.1.11 Méthode Contrainte

Le type du prototype de la méthode **Contrainte** est :

- **TypUpsVmMethodeCoherenceEntier** pour un nombre entier.
- **TypUpsVmMethodeCoherenceReel** pour un nombre réel.
- **TypUpsVmMethodeCoherence** pour un objet.

Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmChar UpsVmAPI MethodeCoherenceEntier(TypUpsVmSession *Session,
    TypUpsVmLong UpsObjet, TypUpsVmAdresse *AdresseObjet)
/* Objet : Contrainte(). */
/*****/
{
...
}

/*****/
TypUpsVmChar UpsVmAPI MethodeCoherenceReel(TypUpsVmSession *Session,
    TypUpsVmDouble UpsObjet, TypUpsVmAdresse *AdresseObjet)
/* Objet : Contrainte(). */
/*****/
{
...
}

/*****/
TypUpsVmChar UpsVmAPI MethodeCoherence(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet)
/* Objet : Contrainte(). */
/*****/
{
...
}

```

Texte 31 – Exemple d'une méthode Contrainte

7.2.1.12 Méthode Exporter

Le type du prototype de la méthode **Exporter** est *TypUpsVmMethodeExporter*. Voici un exemple :

```


/*****/
TypUpsVmVoid UpsVmAPI MethodeExporter(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet)
/* Objet : Procedure Exporter(); */
/*****/
{
...
}

```

Texte 32 – Exemple d'une méthode Exporter

7.2.1.13 Méthode Importer

Le type du prototype de la méthode **Importer** est *TypUpsVmMethodeImporter*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeImporter(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypPrmUpsVmImporter *UpsPrm)
/* Objet : Fonction Importer(P : Nul Ou Entrepot=Nul) Retourner
Nul Ou Objet; */
/*****/
{
    ...
}

```

Texte 33 – Exemple d'une méthode Importer

7.2.2 Dictionnaire de données et de traitements

La seconde partie du type **TypUpsVmEnteteMethodes** regroupe des méthodes de rappel obligatoires sur le type à destination d'**Up! Kernel** pour constituer le dictionnaire de données et de traitements et agir dessus :


Méthodes de rappel.	Sémantique.
Construction du dictionnaire de données et de traitements.	
MethodeEnumererProprietes.	Enumération des propriétés publiques.
MethodeEnumererMethodes.	Enumération des méthodes publiques.
Actions sur le dictionnaire de données et de traitements.	
MethodeEcrirePropriete.	Ecriture de la valeur d'une propriété publique.
MethodeLirePropriete.	Lecture de la valeur d'une propriété publique.
MethodeIncrementerDecrementerPropriete.	Incrémentation ou décrémentation de la valeur d'une propriété publique.
MethodeSupprimerPropriete.	Suppression de la valeur d'une propriété publique.

Tableau 34 – Dictionnaire de données et de traitements – Types ou interfaces

a Ces méthodes agissent également sur les propriétés dynamiques.

7.2.2.1 Méthode EcrirePropriete

Le type du prototype de la méthode **EcrirePropriete** est **TypUpsVmMethodeEcrirePropriete**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeLirePropriete(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *AdresseObjet,
    TypPrmUpsVmLirePropriete *UpsPrm)
/* Objet : Procedure EcrirePropriete(NomPropriete : Caractere,
ValeurPropriete : Nul Ou Objet); */
/*****/
{
...
}

```

Texte 35 – Exemple d'une méthode EcrirePropriete

7.2.2.2 Méthode EnumererMethodes

Le type du prototype de la méthode **EnumererMethodes** est **TypUpsVmMethodeEnumererAppels**. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererMethodes(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererAppels *UpsPrm)
/* Objet : Fonction EnumererMethodes(NomAppel: Nul Ou
Caractere Sortie, Aide: Nul Ou Caractere Sortie) Retourner Nul Ou
Appel; */
/*****/
{
...
}

```

Texte 36 – Exemple d'une méthode EnumererMethodes

7.2.2.3 Méthode EnumererProprietes

Le type du prototype de la méthode **EnumererProprietes** est **TypUpsVmMethodeEnumererProprietes**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererProprietes(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse
    *AdresseObjet, TypPrmUpsVmEnumererProprietes *UpsPrm)
/* Objet : Fonction EnumererProprietes(NomPropriete : Nul Ou Type Propriete :
Nul Ou Type Sortie, EstNul : Booleen Sortie, Aide:Nul Ou Caractere
Sortie, NomEnumere : Nul Ou Caractere Sortie) Retourner Nul Ou
Caractere;
/*****/
{
...
}

```

Texte 37 – Exemple d'une méthode EnumererProprietes

7.2.2.4 Méthode IncrementerDecrementerPropriete

Le type du prototype de la méthode **IncrementerDecrementerPropriete** est **TypUpsVmMethodeIncrementerDecrementerPropriete**. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *MethodeIncrementerDecrementerPropriete(
    TypUpsVmSession *Session, TypUpsVmAdresse *AdresseResultat,
    TypUpsVmAdresse *AdresseObjet,
    TypPrmUpsVmIncrementerDecrementerPropriete *UpsPrm)
/* Objet : Fonction IncrementerDecrementerPropriete(NomPropriete : Nul
Ou Caractere, Incrementer : Booleen, Prefixe : Booleen) Retourner Nul
Ou Objet;
/*****/
{
...
}

```

Texte 38 – Exemple d'une méthode IncrementerDecrementerPropriete

7.2.2.5 Méthode LirePropriete

Le type du prototype de la méthode **LirePropriete** est **TypUpsVmMethodeLirePropriete**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeEcrirePropriete(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet, TypPrmUpsVmEcrirePropriete *UpsPrm)
/* Objet : Fonction LirePropriete(NomPropriete : Caractere) Retourner
Nul Ou Objet; */
/*****/
{
    ...
}

```

Texte 39 – Exemple d'une méthode LirePropriete

7.2.2.6 Méthode SupprimerPropriete

Le type du prototype de la méthode **SupprimerPropriete** est **TypUpsVmMethodeSupprimerPropriete**. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeSupprimerPropriete(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseObjet, TypPrmUpsVmSupprimerPropriete *UpsPrm)
/* Objet : Fonction LirePropriete(NomPropriete : Caractere) Retourner
Nul Ou Objet; */
/*****/
{
    ...
}

```

Texte 40 – Exemple d'une méthode SupprimerPropriete

7.2.3 Liaison avec Up ! Virtual Technical Machine


La troisième partie du type **TypUpsVmEnteteMethodes** regroupe des méthodes de rappel obligatoires sur le type à destination d'**Up ! Virtual Technical Machine** :

Méthodes de rappel.	Destinataire.	Sémantique.
MethodeUtiliserObjetsChamps.	Up ! Oms.	Utilise les champs d'un objet pour une transaction.
MethodeNePlusUtiliserObjetsChamps	Up ! Oms.	N'utilise plus les champs d'un objet pour une transaction.

Tableau 41 – Liaison avec Up ! Virtual Technical Machine – Types ou interfaces

7.2.3.1 Méthode NePlusUtiliserObjetsChamps

Le type du prototype de la méthode **NePlusUtiliserObjetsChamps** est **TypUpsVmNePlusUtiliserObjetsChamps**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeNePlusUtiliserObjetsChamps(TypUpsVmSession
    *Session, TypUpsVmChar *Adresse)
/* Objet : N'utilise plus les champs d'un objet pour une
transaction. */
/*****/
{
    ...
}

```

Texte 42 – Exemple d'une méthode NePlusUtiliserObjetsChamps

7.2.3.2 Méthode UtiliserObjetsChamps

Le type du prototype de la méthode **UtiliserObjetsChamps** est **TypUpsVmUtiliserObjetsChamps**. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeUtiliserObjetsChamps(TypUpsVmSession
    *Session, TypUpsVmChar *Adresse)
/* Objet : Utilise les champs d'un objet pour une transaction. */
/*****/
{
    ...
}

```

Texte 43 – Exemple d'une méthode UtiliserObjetsChamps

7.2.4 Méthode de lecture de l'objet

Un type ou une interface public comporte une méthode retournant l'objet représentant le concept. Voici les conventions de dénomination de la méthode :


- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsType\$NumeroType\$_get

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

UpsType\$NumeroType\$_get

Voici un exemple de déclaration d'une méthode de lecture de l'objet :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****
TypUpsVmAdresse UpsVmAPI *UpsType1_get(TypUpsVmSession *Session)
/* Type FichierListe. */
/*****
{
...
}

/*****
typedef struct typupssystraitements_1_0_0
/* Objet: Interface des traitements de Ups Sys. */
/*****
{
...
TypUpsVmAdresse UpsVmAPI>(*UpsType1_get)(TypUpsVmSession *Session);
/* Type FichierListe. */
...
} *TypUpsSysTraitements_1_0_0;

```

Texte 44 – Exemple de déclaration d'une méthode de lecture de l'objet type

7.2.5 Méthode d'index

Un type ou une interface public comporte une méthode retournant l'index décompté en octets de chaque propriété non virtuelle dans la structure des propriétés du type.

Voici les conventions de dénomination de la méthode d'index :

- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsIndex\$NumeroType\$_\$VersionRevision\$_get

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

UpsIndex\$NumeroType\$_get

Voici un exemple de déclaration d'une méthode d'index :



Plan d'écriture d'un module

Date rédaction :

16 février 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

```
/* **** */
TypUpsVmLong UpsVmAPI UpsIndex12_1_0_0_get(TypUpsVmSession *Session,
TypUpsVmShort Numero)
/* Propriétés de Flux. */
/* **** */
{
switch (Numero)
{
case 1 :
return(0);
break;

case 2 :
return(4);
break;

case 3 :
return(36);
break;
}
(*UpsSysIntTrtUpsKrn->EnvoyerExceptionStandard)(Session,
(*UpsSysIntTrtUpsKrn->UpsException54_get)(Session),
(*UpsSysIntTrtUpsNls->Traduire)(UpsSysNumeroModule, UpsEnuUpsNls_1_4, _T("Flux")),
NULL, NULL, NULL, NULL);
return(0);
}

/* **** */
typedef struct typupssystraitements_1_0_0
/* Objet: Interface des traitements de Ups Sys. */
/* **** */
{
...
TypUpsVmLong UpsVmAPI (*UpsIndex12_get)(TypUpsVmSession *Session,
TypUpsVmShort Numero);
/* Type Flux. */
...
} *TypUpsSysTraitements_1_0_0;
```

Texte 45 – Exemple de déclaration de la méthode d'index

7.2.6 Types ou interfaces polymorphes

Les déclarations **Selon ... Cas ... Fin Cas** comporte deux méthodes spécifiques permettant de gérer les parties polymorphes d'un type ou d'une interface. Les voici :

- **Méthode de calcul du cas.**
L'objet de cette méthode fonctionnelle est de calculer quelle est la partie polymorphe du cas à activer en testant la valeur de l'énuméré.
- Le foncteur de la méthode privée du type, employé dans le corps du composant dont il fait partie, porte le nom générique suivant :

UpsMthCasSelon_ \$PrefixePaquet\$ \$NumeroType\$ _ \$NumeroSelon\$

- Le foncteur de la méthode privée du type, employé dans la table de ses méthodes, porte le nom générique suivant :

CasSelon\$NumeroSelon\$

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

Voici un exemple de déclaration d'une méthode de calcul du cas :

```

/*****/
static TypUpsVmShort UpsVmAPI CasSelon1(TypUpsVmUnsignedShort Valeur)
/* Sorte. */
/*****/
{
...
}


/*****/
typedef struct upstrttypupssec_3_1_0_0
/* Objet : Methodes du type Habilitations. */
/*****/
{
...
TypUpsVmShort UpsVmAPI (*CasSelon1)(TypUpsVmUnsignedShort Valeur);
/* Sorte. */
...
} UpsTrtTypUpsSec_3_1_0_0;

```

Texte 46 – Exemple de déclaration d'une méthode de calcul du cas

- **Méthode de mise à jour du cas.**
L'objet de cette méthode fonctionnelle est de mettre à jour les propriétés des parties polymorphes d'un objet lors du changement de la valeur du sélecteur. Certaines propriétés sont détruites alors que d'autres sont initialisées.
 - Le foncteur de la méthode privée du type, employé dans le corps du composant dont il fait partie, porte le nom générique suivant :
UpsMthMAJSelon_ \$PrefixePaquet\$ \$NumeroType\$ _ \$NumeroSelon\$
 - Le foncteur de la méthode privée du type, employé dans la table de ses méthodes, porte le nom générique suivant :
MAJSelon\$NumeroSelon\$

Voici un exemple de déclaration d'une méthode de mise à jour du cas :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
static TypUpsVmVoid UpsVmAPI MAJSelon1(TypUpsVmSession *Session,
    TypUpsVmAdresse *UpsObjet, TypUpsVmUnsignedShort ValeurSuivante)
/* Sorte. */
/*****/
{
...
}

/*****/
typedef struct upstrttypupssec_3_1_0_0
/* Objet : Methodes du type Habilitations. */
/*****/
{
...
TypUpsVmVoid UpsVmAPI (*MAJSelon1)(TypUpsVmSession *Session,
    TypUpsVmAdresse *UpsObjet, TypUpsVmUnsignedShort ValeurSuivante);
/* Sorte. */
...
} UpsTrtTypUpsSec_3_1_0_0;

```

Texte 47 – Exemple de déclaration d'une méthode de mise à jour du cas

7.2.7 Méthode de cohérence

Si un type est un alias comportant contrainte, il y alors une méthode de cohérence. Voici la convention de dénomination de la méthode de cohérence dans le corps du composant dont il fait partie :

- Pour un alias sur le type **Entier**.
UpsMthCohérenceEntier\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$
- Pour un alias sur le type **Reel**.
UpsMthCohérenceReel\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$
- Pour un alias sur un type objet.
UpsMthCohérence\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$


Voici un exemple de déclaration d'une méthode de lecture de l'objet :

```

/*****/
TypUpsVmChar UpsVmAPI UpsMthCohérenceReelUpsMat_1(TypUpsVmSession *Session,
    TypUpsVmDouble UpsObjet, TypUpsVmAdresse *AdresseObjet)
/* Type Degres. */
/*****/
{
...
}

```

Texte 48 – Exemple de déclaration d'une méthode de cohérence

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

7.3 Déclaration d'un type ou d'une interface public

Un type ou une interface public se déclare dans l'interface de traitements du module de la manière suivante :

- **Table des méthodes.**
Voici la convention de sa dénomination :

UpsTrt\$NumeroType\$

- **Méthode de lecture de l'objet.**
Voir précédemment.
- **Méthode d'index.**
Voir précédemment.

7.3.1 Héritage ou implémentation

Un type qui hérite de types comportent autant de tables de méthodes supplémentaires qu'il existe d'interfaces implémentées. Celles-ci permettent de surclasser l'objet du type hérité.

Un type qui implémente des interfaces comporte autant de tables de méthodes supplémentaires qu'il existe d'interfaces implémentées. Celles-ci permettent de surclasser l'objet de l'interface implémentée.

Une interface qui hérite d'interfaces comporte autant de tables de méthodes supplémentaires qu'il existe d'interfaces héritées. Celles-ci permettent de surclasser l'objet de l'interface héritée.

En cas de récurrence, le parcours est in-fixé.

Voici un exemple de déclaration des tables de méthodes surclassées :

```

/*****
typedef struct typupssystraitements_1_0_0
/* Objet: Interface des traitements de Ups Sys. */
/*****
{
...
UpsTrtTypUpsSys_8 *UpsTrt8;
/* Methodes de Fichier. */
UpsTrtTypUpsSys_12 *UpsTrt8_1;
/* Methodes de Fichier.Flux. */
...
} *TypUpsSysTraitements_1_0_0;


```

Texte 49 – Exemple de déclaration d'une méthode de lecture de l'objet

7.4 Propriétés virtuelles


Une propriété virtuelle est simulée par une méthode possédant deux prototypes :

- **Pour une propriété énumérée.**
Le prototype de lecture correspond au type **TypUpsKrnVirtuelVariableEnumereCourtLecture** ou **TypUpsKrnVirtuelVariableEnumereLongEcriture**.
Le prototype d'écriture correspond au type **TypUpsKrnVirtuelVariableEnumereLongLecture** ou **TypUpsKrnVirtuelVariableEnumereLongEcriture**.
- **Pour une propriété entière.**
Le prototype de lecture correspond au type **TypUpsKrnVirtuelVariableEntierLecture**.
Le prototype d'écriture correspond au type **TypUpsKrnVirtuelVariableEntierEcriture**.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

- **Pour une propriété réelle.**
Le prototype de lecture correspond au type ***TypUpsKrnVirtuelVariableReelLecture.***
Le prototype d'écriture correspond au type ***TypUpsKrnVirtuelVariableReelEcriture.***
- **Pour une propriété objet.**
Le prototype de lecture correspond au type ***TypUpsKrnVirtuelVariableObjetLecture.***
Le prototype d'écriture correspond au type ***TypUpsKrnVirtuelVariableObjetEcriture.***

La méthode est déclarée dans l'ordre d'énumération des propriétés. L'accès à la méthode correspond à l'accès à la propriété – public, protégé ou privé. La dénomination de la méthode suit à dénomination usuelle.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

8 Variables

8.1 Variables globales

Selon leur accès public, protégé ou privé, les variables globales sont déclarées dans un segment de données différent. Pour cela, confère **Spécification technique du module UpsMod** [A5].

Une variable globale est accessible au travers de plusieurs méthodes du module qui la définit :

- **Une méthode de lecture.**

Si son accès en lecture est public.

- Le foncteur de la méthode publique du module, employé dans le corps du composant dont il fait partie, porte le nom générique suivant :

`UpsVar$NumeroVariable$_get`

- Le foncteur de la méthode publique du module, employé dans la table de ses méthodes, porte le nom générique suivant :

`UpsVar$NumeroVariable$_get`

Le prototype est le suivant :

- **TypUpsKrnGlobalVariableEnumereCourtLecture** pour un énuméré court.
 - **TypUpsKrnGlobalVariableEnumereLongLecture** pour un énuméré long.
 - **TypUpsKrnGlobalVariableEntierLecture** pour un nombre entier.
 - **TypUpsKrnGlobalVariableReelLecture** pour un nombre réel.
 - **TypUpsKrnGlobalVariableObjetLecture** pour un objet.
 - **Une méthode d'écriture.**
- Si son accès en écriture est public.
- Le foncteur de la méthode publique du module, employé dans le corps du composant dont il fait partie, porte le nom générique suivant :


`UpsVar$NumeroVariable$_set`

- Le foncteur de la méthode publique du module, employé dans la table de ses méthodes, porte le nom générique suivant :

`UpsVar$NumeroVariable$_set`

Le prototype est le suivant :

- **TypUpsKrnGlobalVariableEnumereCourtEcriture** pour un énuméré court.
 - **TypUpsKrnGlobalVariableEnumereLongEcriture** pour un énuméré long.
 - **TypUpsKrnGlobalVariableEntierEcriture** pour un nombre entier.
 - **TypUpsKrnGlobalVariableReelEcriture** pour un nombre réel.
 - **TypUpsKrnGlobalVariableObjetEcriture** pour un objet.
 - **Une méthode d'incrément et de décrémentation.**
- Si son accès en écriture est public et que la variable est de type entier ou réel.
- Le foncteur de la méthode publique du module, employé dans le corps du composant dont il fait partie, porte le nom générique suivant :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

UpsVar\$NumeroVariable\$_inc

- Le foncteur de la méthode publique du module, employé dans la table de ses méthodes, porte le nom générique suivant :

UpsVar\$NumeroVariable\$_inc

Le prototype est le suivant :

- **TypUpsKrnGlobalVariableEntierIncrementationDecrementation** pour un nombre entier.
- **TypUpsKrnGlobalVariableReelIncrementationDecrementation** pour un nombre réel.

Voici un exemple :



Plan d'écriture d'un module

Date rédaction :

16 février 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

```
/* ***** */
TypUpsVmDouble UpsVmAPI UpsVar33_get(TypUpsVmSession *Session)
/* UnDollar. */
/* ***** */
{
...
}

/* ***** */
TypUpsVmVoid UpsVmAPI UpsVar33_set(TypUpsVmSession *Session,
TypUpsVmDouble UpsPrm, enum EnuUpsVmOperateur Operateur)
/* UnDollar. */
/* ***** */
{
...
}

/* ***** */
TypUpsVmDouble UpsVmAPI UpsVar33_inc(TypUpsVmSession *Session,
TypUpsVmChar Incrementer, TypUpsVmChar Prefixe)
/* UnDollar. */
/* ***** */
{
...
}

/* ***** */
typedef struct typupsnlstraitements_1_0_0
/* Objet: Interface des traitements de Ups Nls. */
/* ***** */
{
...
TypUpsVmDouble UpsVmAPI (*UpsVar33_get)(TypUpsVmSession *Session);
/* UnDollar. */
TypUpsVmVoid UpsVmAPI (*UpsVar33_set)(TypUpsVmSession *Session,
TypUpsVmDouble UpsPrm, enum EnuUpsVmOperateur Operateur);
/* UnDollar. */
TypUpsVmDouble UpsVmAPI (*UpsVar33_inc)(TypUpsVmSession *Session,
TypUpsVmChar Incrementer, TypUpsVmChar Prefixe);
/* UnDollar. */
...
} *TypUpsNlsTraitements_1_0_0;
```

Texte 50 – Exemple de déclaration d'une variable globale


8.2 Variables locales

Les variables locales d'un appel sont regroupées dans un temporaire qui peut être transmis par référence quand l'appel est fractionné en plusieurs procédures ou fonctions.

Voici la convention de dénomination de la structure :

- Pour un appel de procédure ou de fonction.

TypLoc\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroAppel\$_\$NumeroPrototype\$

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

- **Pour une méthode.**
Cela varie selon l'accès à la méthode :
 - **Méthode publique.**

TypLoc\$PrefixeModule\$_\$PrefixePaquet\$\$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

- **Méthode protégée.**

TypLoc\$PrefixeModule\$_\$PrefixePaquet\$Pro_ \$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

- **Méthode privée.**

TypLoc\$PrefixeModule\$_\$PrefixePaquet\$Pri_ \$NumeroType\$_\$NumeroAppel\$_\$NumeroPrototype\$

Les variables locales sont énumérées par ordre de déclaration dans la procédure, la fonction ou la méthode. Voici la convention de dénomination d'un paramètre :

UpsVar\$NumeroVariable\$

Voici un exemple :

```

/*****/
typedef struct typlocupskrn_2_4_7
/* Objet : Variable locale de Caractere.Constructeur(E : Nul Ou */
/* Entier, Filtre : Nul Ou Caractere, PointVirgule : Nul Ou      */
/* Caractere)                                                    */
/*****/
{
TypUpsVmAdresse UpsVar1;
/* Tmp : Nul Ou Caractere. */
} TypLocUpsKrn_2_4_7;


```

Texte 51 – Exemple de déclaration d'un type du temporaire des variables locales d'un appel

8.3 Variables virtuelles

Une variable virtuelle est simulée par une méthode du module possédant deux prototypes pour la lecture et l'écriture. Les types des prototypes sont ceux des prototypes pour la lecture et l'écriture d'une variable globale.

La méthode est déclarée dans l'ordre d'énumération des variables. L'accès à la méthode correspond à l'accès à la propriété – public, protégé ou privé. La dénomination de la méthode suit à dénomination usuelle pour une procédure ou une fonction.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

9 Exceptions

Une exception publique comporte une méthode retournant l'objet représentant le concept. Voici les conventions de dénomination de la méthode :

- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsException\$NumeroException\$_get

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

UpsException\$NumeroException\$_get

Voici un exemple de déclaration d'une méthode de lecture de l'objet :


```

/*****
TypUpsVmAdresse UpsVmAPI *UpsException1_get(TypUpsVmSession *Session)
/* Exception PlusDeMemoire. */
/*****
{
...
}

/*****
typedef struct typupskrntraitements_1_0_0
/* Objet: Interface des traitements de Ups Krn. */
/*****
{
...
TypUpsVmAdresse UpsVmAPI *(*UpsException1_get)(TypUpsVmSession *Session);
/* Exception PlusDeMemoire. */
...
} *TypUpsKrnTraitements_1_0_0;

```

Texte 52 – Exemple de déclaration d'une méthode de lecture de l'objet exception

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

10 Files d'attente

Une file d'attente publique comporte une méthode retournant l'objet représentant le concept. Voici les conventions de dénomination de la méthode :

- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

UpsFileDAttente\$NumeroFileDAttente\$_get

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

UpsFileDAttente\$NumeroFileDAttente\$_get

Voici un exemple de déclaration d'une méthode de lecture de l'objet :


```

/*****
TypUpsVmAdresse UpsVmAPI *UpsFileDAttente1_get(TypUpsVmSession *Session)
/* File d'attente MaFile. */
/*****
{
...
}

/*****
typedef struct typmonmodtraitements_1_0_0
/* Objet: Interface des traitements de Mon Module. */
/*****
{
...
TypUpsVmAdresse UpsVmAPI>(*UpsFileDAttente1_get)(TypUpsVmSession
*Session);
/* File d'attente MaFile. */
...
} *TypMonModTraitements_1_0_0;

```

Texte 53 – Exemple de déclaration d'une méthode de lecture de l'objet file d'attente

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

11 Entrepôts

Un entrepôt public comporte deux méthodes :

- Une méthode retournant l'objet représentant le concept.

Voici les conventions de dénomination de la méthode :

- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

`UpsEntrepot$NumeroEntrepot$_get`

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

`UpsEntrepot$NumeroEntrepot$_get`

- Une méthode retournant le numéro de l'entrepôt selon la convention d'**Up ! Object Management System**.

Voici les conventions de dénomination de la méthode :


- Le foncteur de la méthode publique du module, employé dans le corps du composant dont elle fait partie, porte le nom générique suivant :

`UpsNumeroEntrepot$NumeroEntrepot$_get`

- Le foncteur de la méthode publique du module, employé dans l'interface de traitements du module, porte le nom générique suivant :

`UpsNumeroEntrepot$NumeroEntrepot$_get`

Voici un exemple de déclaration des méthodes d'un entrepôt :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *UpsEntrepot1_get(TypUpsVmSession *Session)
/* Entrepot Systeme. */
/*****/
{
...
}

/*****/
TypUpsVmUnsignedShort UpsVmAPI UpsNumeroEntrepot1_get(TypUpsVmSession
*Session)
/* Numero de l'entrepot Systeme. */
/*****/
{
...
}

/*****/
typedef struct typupskrntraitements_1_0_0
/* Objet: Interface des traitements de Ups Krn. */
/*****/
{
...
TypUpsVmAdresse UpsVmAPI *(*UpsEntrepot1_get)(TypUpsVmSession *Session);
/* Entrepot Systeme. */
TypUpsVmUnsignedShort UpsVmAPI (*UpsNumeroEntrepot1_get)(TypUpsVmSession
*Session);
/* Numero de l'entrepot Systeme. */
...
} *TypUpsKrnTraitements_1_0_0;

```

Texte 54 – Exemple de déclaration d'une méthode de lecture de l'objet exception

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

12 Modules

12.1 Propriétés d'un module

Les propriétés d'un module sont regroupées dans une structure.

Voici la convention de sa dénomination :

Typ\$PrefixeModule\$Donnees

Voici le contenu de cette structure :

- **En-tête des propriétés.**
Il s'agit des propriétés génériques du type **Module**. Elles sont regroupées dans le type **TypUpsVmEnteteDonneesModule**. La convention de dénomination du sélecteur est **EnteteDonneesModule**.
Nous retrouvons notamment l'objet représentant le module.
- **Des éventuelles méthodes propriétaires.**
Par exemple pour la gestion des ressources.

Voici un exemple :

```

/*****
typedef struct typupskrndonnees
/* Objet: Interface des donnees de Ups Krn. */
/*****
{
TypUpsVmEnteteDonneesModule EnteteDonneesModule;
/* Entete generique du module. */

TypUpsVmShort MiseAuPoint;
/* Si Vrai, la mise au point est activee. */
TypUpsVmUnicode FichierMiseAuPoint[CO_TailleNomFichier+1];
/* Fichier de mise au point. */
TypUpsVmUnicode Log[CO_TailleNomFichier+1];
/* Fichier journal. */
TypUpsVmShort CompteurMiseAuPoint;
/* Pour compter l'activation de la trace. */
TypUpsVmShort CompteurAppels;
/* Pour compter les appels. */
} *TypUpsKrnDonnees;

```

Texte 55 – Exemple de déclaration des propriétés d'un module

M

Quand le type est modifié au cours d'un changement de version-révision, il ne peut qu'être étendu en terme de propriétés, et, auquel cas, les nouvelles propriétés sont ajoutées à la suite.


12.2 Méthodes d'un module

Les méthodes d'un module sont regroupées dans une structure. Il y a une structure par version-révision de l'application.

Voici la convention de sa dénomination :

Typ\$PrefixeModule\$Traitements_ \$VersionRevision\$

Voici le contenu de cette structure :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

- **En-tête des méthodes.**
Il s'agit des méthodes génériques du type **Objet**. Elles sont regroupées dans le type **TypUpsVmEnteteMethodesModule**. La convention de dénomination du sélecteur est **EnteteMethodesModule**.
- **Les procédures et les fonctions publiques.**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination d'une procédure ou d'une fonction est présentée dans la section **6.3** intitulée **Foncteur** page 17.
- **Les co-méthodes des procédures et des fonctions publiques.**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination d'une co-méthode est présentée dans la section **6.3** intitulée **Foncteur** page 17.
- **Les tables de méthodes des types publics.**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination d'une table de méthodes est présentée dans la section **7.3** intitulée **Déclaration d'un type ou d'une interface public** page 44.
- **Les méthodes de lecture des objets représentant les types et de lecture de l'index des propriétés.**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination de ces méthodes est présentée dans la section **7.3** intitulée **Déclaration d'un type ou d'une interface public** page 44.
- **Les méthodes de lecture, d'écriture et d'incrémentation des variables publiques.**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination de ces méthodes est présentée dans la section **8** intitulée **Variables** page 46.
- **Les méthodes de lecture des objets représentant les exceptions**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination de ces méthodes est présentée dans le chapitre **9** intitulé **Exceptions** page 50.
- **Les méthodes de lecture des objets représentant les files d'attentes**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination de ces méthodes est présentée dans le chapitre 10 intitulé **Files d'attente** page 51.
- **Les méthodes concernant les entrepôts**
Elles sont énumérées par ordre de déclaration dans l'interface du module. La convention de dénomination de ces méthodes est présentée dans le chapitre **11** intitulé **Entrepôt** page 52.
- **Des éventuelles méthodes propriétaires.**
Par exemple, toutes les **Application Program Interface (API)** d'**Up ! Virtual Technical Machine** en **C--**.

Voici un exemple :



Plan d'écriture d'un module

Date rédaction :

16 février 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

```
/* **** */
typedef struct typupsknrtraitements_1_0_0
/* Objet: Interface des traitements de Ups Krn. */
/* **** */
{
TypUpsVmEnteteMethodesModule EnteteMethodesModule;
/* Methodes generiques au module. */

TypUpsVmVoid UpsVmAPI (*Trt_1_1)(TypUpsVmSession *Session, TypPrmUpsKrn_1_1
*UpsPrm);
/* Objet : ActiverMiseAuPoint. */
TypUpsVmVoid UpsVmAPI (*Trt_2_1)(TypUpsVmSession *Session, TypPrmUpsKrn_2_1
*UpsPrm);
/* Objet : ActiverContrainteNul. */
...

TypUpsVmCoAppel CoTrt_1_1;
/* Objet : ActiverMiseAuPoint. */
TypUpsVmCoAppel CoTrt_2_1;
/* Objet : ActiverContrainteNul. */
...

UpsTrtTypUpsKrn_1 *UpsTrt1;
/* Methodes de Binaire. */
UpsTrtTypUpsKrn_2 *UpsTrt2;
/* Methodes de Caractere. */
...

TypUpsVmAdresse UpsVmAPI>(*UpsType1_get)(TypUpsVmSession *Session);
/* Type Binaire. */
TypUpsVmLong UpsVmAPI(*UpsIndex1_get)(TypUpsVmSession *Session, TypUpsVmShort
Numero);
/* Type Binaire. */
TypUpsVmAdresse UpsVmAPI>(*UpsType2_get)(TypUpsVmSession *Session);
/* Type Caractere. */
TypUpsVmLong UpsVmAPI(*UpsIndex2_get)(TypUpsVmSession *Session, TypUpsVmShort
Numero);
/* Type Caractere. */
...


TypUpsVmAdresse UpsVmAPI(*UpsException1_get)(TypUpsVmSession *Session);
/* Exception PlusDeMemoire. */
TypUpsVmAdresse UpsVmAPI(*UpsException2_get)(TypUpsVmSession *Session);
/* Exception BufferTropGrand. */
...

TypUpsVmAdresse UpsVmAPI(*UpsVar1_get)(TypUpsVmSession *Session);
/* RepertoireTmp. */

TypUpsVmAdresse UpsVmAPI(*UpsEntrepot1_get)(TypUpsVmSession *Session);
/* Entrepot Systeme. */
TypUpsVmUnsignedShort UpsVmAPI(*UpsNumeroEntrepot1_get)(TypUpsVmSession *Session);
/* Numero de l'entrepot Systeme. */

TypUpsVmChar UpsVmAPI(*CalculerBuffer)(TypUpsVmSession *Session, TypUpsVmAdresse
*AdresseBinaireOuCaractere, TypUpsVmUnsignedLong *NumeroVerrouBuffer,
enum EnuUpsOmsVerrou TypeVerrou);
/* Objet : Calcule l'adresse du buffer binaire ou caractere. */
TypUpsVmLong UpsVmAPI(*CalculerTaille)(TypUpsVmSession *Session, TypUpsVmAdresse
*AdresseObjet, TypUpsVmShort CalculerLongueur);
/* Objet : Calcule la taille du buffer binaire ou caractere en octets. */
...
} *TypUpsKrnTraitements_1_0_0;
```

Texte 56 – Exemple de déclaration des méthodes d'un module

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

M Quand le type est modifié au cours d'un changement de version-révision, il ne peut qu'être étendu en terme de méthodes, et, auquel cas, les nouvelles méthodes sont ajoutées à la suite.

Un alias indique la version-révision courante des méthodes d'un type ou d'une interface. Voici un exemple :

```

/*-----*/
/* La version courante est la 1.0.0. */
/*-----*/
typedef TypUpsKrnTraitements_1_0_0 TypUpsKrnTraitements;

```

Texte 57 – Exemple de déclaration de la version-révision courante des méthodes d'un module

12.2.1 Liaison avec Up ! Virtual Technical Machine


La première partie du type *TypUpsVmEnteteMethodesModule* regroupe des informations et des méthodes de rappel obligatoires sur le module à destination d'*Up ! Virtual Technical Machine* :

Informations.	Destinataire.	Sémantique.
Version.	<i>Up ! Module.</i>	Numéro de version de l'interface du module.
Revision.	<i>Up ! Module.</i>	Numéro de révision de l'interface du module.
Correction.	<i>Up ! Module.</i>	Numéro de correction de l'interface du module.
Méthodes de rappel.	Destinataire.	Sémantique.
ChargerPersistence.	<i>Up ! Oms.</i>	Charge les informations persistantes du module.
EnregistrerPersistence.	<i>Up ! Oms.</i>	Enregistre les informations persistantes du module.
ExecuterAppelDistant.	<i>Up ! Network.</i>	Exécute un appel distant selon le message transmis.
LireEnteteDonneesModule.	<i>Up ! Module.</i>	Retourne l'en-tête de données du module.
Principal.	<i>Up ! Module.</i>	Pour le module contenant la fonction principale, exécute celle-ci.
RechercherImplementation Interface.	<i>Up ! Kernel.</i>	Recherche l'objet d'un type implémentant une interface, elle-même identifiée par son objet.

Tableau 58 – Liaison avec Up ! Virtual Technical Machine – Modules

12.2.1.1 Méthode ChargerPersistence

Le type du prototype de la méthode *ChargerPersistence* est *TypUpsVmChargerPersistence*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/***** /
TypUpsVmShort UpsVmAPI MethodeChargerPersistence(TypUpsVmVoid)
/* Objet : Charge la persistence du module. */
/***** /
{
...
}

```

Texte 59 – Exemple d'une méthode ChargerPersistence

12.2.1.2 Méthode EnregistrerPersistence

Le type du prototype de la méthode **EnregistrerPersistence** est **TypUpsVmEnregistrerPersistence**. Voici un exemple :

```

/***** /
TypUpsVmVoid UpsVmAPI MethodeEnregistrerPersistence(TypUpsVmVoid)
/* Objet : Enregistre la persistence du module. */
/***** /
{
...
}

```

Texte 60 – Exemple d'une méthode EnregistrerPersistence

12.2.1.3 Méthode ExecuterAppelDistant

Le type du prototype de la méthode **ExecuterAppelDistant** est **TypUpsVmExecuterAppelDistant**. Voici un exemple :

```


/***** /
TypUpsVmVoid UpsVmAPI MethodeExecuterAppelDistant(TypUpsVmSession
*Session, TypUpsNetProtocoleUpsNet *ProtocoleUpsNet,
TypUpsVmPointeurDonnees Parametres)
/* Objet : Execute un appel distant. */
/***** /
{
...
}

```

Texte 61 – Exemple d'une méthode ExecuterAppelDistant

12.2.1.4 Méthode LireEnteteDonneesModule

Le type du prototype de la méthode **LireEnteteDonneesModule** est **TypUpsVmLireEnteteDonneesModule**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmPointeurDonnees UpsVmAPI MethodeLireEnteteDonneesModule(
    TypUpsVmVoid)
/* Objet : Retourne l'en-tete des donnees du module. */
/*****/
{
    ...
}

```

Texte 62 – Exemple d'une méthode LireEnteteDonneesModule

12.2.1.5 Fonction Principal

Le type du prototype de la fonction **Principal** est **TypUpsVmPrincipal**. Voici un exemple :

```

/*****/
TypUpsVmShort UpsVmAPI Principal(TypUpsVmVoid)
/* Objet : Fonction principale a executer. */
/*****/
{
    ...
}

```

Texte 63 – Exemple d'une fonction principale

12.2.1.6 Méthode RechercherImplementationInterface

Le type du prototype de la méthode **RechercherImplementationInterface** est **TypUpsVmRechercherImplementationInterface**. Voici un exemple :

```

/*****/
TypUpsVmAdresse UpsVmAPI *RechercherImplementationInterface(
    TypUpsVmSession *Session, TypUpsVmAdresse *AdresseResultat,
    TypUpsVmAdresse *Interface)
/* Objet : Retourne l'en-tete des donnees du module. */
/*****/
{
    ...
}


```

Texte 64 – Exemple d'une méthode RechercherImplementationInterface

12.2.2 Dictionnaire de données et de traitements

La seconde partie du type **TypUpsVmEnteteMethodesModule** regroupe des méthodes de rappel obligatoires sur le module à destination d'**Up! Kernel** pour constituer le dictionnaire de données et de traitements et agir dessus :

Méthodes de rappel.	Sémantique.
Construction du dictionnaire de données et de traitements.	
MethodeEnumererEnumeres.	Enumération des énumérés public.
MethodeEnumererValeursEnumeres.	Enumération des valeurs des énumérés publics.

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :

Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc

MethodeEnumererEntrepots.	Enumération des entrepôts publics.
MethodeEnumererTypes.	Enumération des types publics.
MethodeEnumererVariables.	Enumération des variables publiques.
MethodeEnumererExceptions.	Enumération des exceptions publiques.
MethodeEnumererAppels.	Enumération des procédures et des fonctions publiques.
Actions sur le dictionnaire de données et de traitements.	
MethodeEcrireVariable.	Ecriture de la valeur d'une variable globale publique.
MethodeLireVariable.	Lecture de la valeur d'une variable globale publique.
MethodeIncrementerDecrementerVariable.	Incrémentation ou décrémentation de la valeur d'une variable globale publique.

Tableau 65 – Dictionnaire de données et de traitements – Modules

Ces actions sur le dictionnaire de données et de traitements sont complétées par les méthodes dédiées du type **Objet**, **Type** et **Appel**.

12.2.2.1 Méthode EcrireVariable

Le type du prototype de la méthode **EcrireVariable** est **TypUpsVmMethodeEcrireVariable**. Voici un exemple :

```


/*****
TypUpsVmVoid UpsVmAPI MethodeEcrireVariable(TypUpsVmSession *Session,
      TypUpsVmAdresse *UpsObjet, TypPrmUpsVmEcrireVariable *UpsPrm)
/* Objet : Procedure EcrireVariable(NomVariable : Nul Ou Caractere,
ValeurVariable : Nul Ou Objet);
*****/
{
...
}

```

Texte 66 – Exemple d'une méthode EcrireVariable

12.2.2.2 Méthode EnumererAppels

Le type du prototype de la méthode **EnumererAppels** est **TypUpsVmMethodeEnumererAppels**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererAppels(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererAppels *UpsPrm)
/* Objet : Fonction EnumererAppels(Numero : Entier, NomAppel : Nul Ou
Caractere Sortie, Aide : Nul Ou Caractere Sortie) Retourner Nul Ou
Appel; */
/*****/
{
    ...
}

```

Texte 67 – Exemple d'une méthode EnumererAppels

12.2.2.3 Méthode EnumererEntrepots

Le type du prototype de la méthode **EnumererEntrepots** est **TypUpsVmMethodeEnumererEntrepots**. Voici un exemple :

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererEntrepots(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererEntrepots *UpsPrm)
/* Objet : Fonction EnumererEntrepots(Numero : Entier, Aide : Nul Ou
Caractere Sortie) Retourner Nul Ou Entrepot; */
/*****/
{
    ...
}

```

Texte 68 – Exemple d'une méthode EnumererEntrepots

12.2.2.4 Méthode EnumererEnumeres

Le type du prototype de la méthode **EnumererEnumeres** est **TypUpsVmMethodeEnumererEnumeres**. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererEnumeres(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererEnumeres *UpsPrm)
/* Objet : Fonction EnumererEnumeres(Numero : Entier, Aide:Nul Ou
Caractere Sortie) Retourner Nul Ou Caractere; */
/*****/
{
    ...
}

```

Texte 69 – Exemple d'une méthode EnumererEnumeres

12.2.2.5 Méthode EnumererExceptions

Le type du prototype de la méthode **EnumererExceptions** est **TypUpsVmMethodeEnumererExceptions**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererExceptions(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererExceptions *UpsPrm)
/* Objet : Fonction EnumererExceptions(Numero : Entier, Aide : Nul Ou
Caractere Sortie) Retourner Nul Ou Exception; */
/*****/
{
    ...
}

```

Texte 70 – Exemple d'une méthode EnumererExceptions

12.2.2.6 Méthode EnumererTypes

Le type du prototype de la méthode **EnumererTypes** est **TypUpsVmMethodeEnumererTypes**. Voici un exemple :

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererTypes(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererTypes *UpsPrm)
/* Objet : Fonction EnumererTypes(Numero : Entier, Aide : Nul Ou
Caractere Sortie) Retourner Nul Ou Type; */
/*****/
{
    ...
}

```

Texte 71 – Exemple d'une méthode EnumererTypes

12.2.2.7 Méthode EnumererValeursEnumeres

Le type du prototype de la méthode **EnumererValeursEnumeres** est **TypUpsVmMethodeEnumererValeursEnumeres**. Voici un exemple :

```


/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererValeursEnumeres(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererValeursEnumeres *UpsPrm)
/* Objet : Fonction EnumererValeursEnumeres(NomEnumere : Caractere,
Numero : Entier, Aide : Nul Ou Caractere Sortie) Retourner Nul Ou
Caractere; */
/*****/
{
    ...
}

```

Texte 72 – Exemple d'une méthode EnumererValeursEnumeres

12.2.2.8 Méthode EnumererVariables

Le type du prototype de la méthode **EnumererVariables** est **TypUpsVmMethodeEnumererVariables**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmAdresse UpsVmAPI *MethodeEnumererVariables(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmEnumererVariables *UpsPrm)
/* Objet : Fonction EnumererVariables(NomVariable : Nul Ou Caractere :
Nul Ou Type Sortie, EstNul : Booleen Sortie, Aide:Nul Ou Caractere
Sortie, NomEnumere : Nul Ou Caractere Sortie) Retourner Nul Ou
Caractere;
/*****/
{
...
}

```

Texte 73 – Exemple d'une méthode EnumererVariables

12.2.2.9 Méthode LireVariable

Le type du prototype de la méthode **LireVariable** est **TypUpsVmMethodeLireVariable**. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeLireVariable(TypUpsVmSession *Session,
    TypUpsVmAdresse *AdresseResultat, TypUpsVmAdresse *UpsObjet,
    TypPrmUpsVmLireVariable *UpsPrm)
/* Objet : Fonction LireVariable(NomVariable : Nul Ou Caractere)
Retourner Nul Ou Objet;
/*****/
{
...
}

```

Texte 74 – Exemple d'une méthode LireVariable

12.2.2.10 Méthode IncrementerDecrementerVariable


Le type du prototype de la méthode **IncrementerDecrementerVariable** est **TypUpsVmMethodeIncrementerDecrementerVariable**. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeIncrementerDecrementerVariable(
    TypUpsVmSession *Session, TypUpsVmAdresse *AdresseResultat,
    TypUpsVmAdresse *UpsObjet, TypPrmUpsVmIncrementerDecrementerVariable
    *UpsPrm)
/* Objet : Fonction IncrementerDecrementerVariable(NomVariable : Nul Ou
Caractere, Incrementer : Booleen, Prefixe : Booleen) Retourner Nul Ou
Objet;
/*****/
{
...
}

```

Texte 75 – Exemple d'une méthode IncrementerDecrementerVariable

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

12.2.3 Adaptateurs du module

La dernière partie du type *TypUpsVmEnteteMethodesModule* regroupe des méthodes de rappel obligatoires pour les bibliothèques adaptateurs du module concernées à destination d'*Up! Virtual Technical Machine* :

Méthodes de rappel.	Destinataire.	Sémantique.
MethodeChercherTypeClientDCom.	<i>Up! DCom.</i>	Recherche le type <i>Up!</i> encapsulant un type ou une interface <i>DCom</i> à partir de son <i>Riid</i> .
MethodeChercherTypeServeurDCom.	<i>Up! DCom.</i>	Recherche le type ou l'interface <i>DCom</i> encapsulant le type <i>Up!</i> à partir de son objet.
MethodeChercherTypeClientCorba.	<i>Up! Corba.</i>	Recherche le type <i>Up!</i> encapsulant un type ou une interface <i>Corba</i> à partir de son nom.
MethodeChercherTypeServeurCorba.	<i>Up! Corba.</i>	Recherche le type ou l'interface <i>Corba</i> encapsulant le type <i>Up!</i> à partir de son objet.
MethodeChercherTypeClientJava.	<i>Up! Java.</i>	Recherche le type <i>Up!</i> encapsulant une classe ou une interface <i>Java</i> à partir de sa classe.
MethodeChercherTypeServeurCorba.	<i>Up! Java.</i>	Recherche la classe ou l'interface <i>Java</i> encapsulant le type <i>Up!</i> à partir de son objet.

Tableau 76 – Méthodes de rappel des adaptateurs

12.2.3.1 Méthode ChercherTypeClientJava

Le type du prototype de la méthode *ChercherTypeClientJava* est *TypUpsVmMethodeChercherTypeClientJava*. Voici un exemple :

```


/*****/
TypUpsVmVoid UpsVmAPI MethodeChercherTypeClientCorba(TypUpsVmSession
*Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmUnicode *NomType)
/* Objet : Methode de recherche d'un type Corba. */
/*****/
{
...
}

```

Texte 77 – Exemple d'une méthode ChercherTypeClientCorba

12.2.3.2 Méthode ChercherTypeClientDCom

Le type du prototype de la méthode *ChercherTypeClientDCom* est *TypUpsVmMethodeChercherTypeClientDCom*. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/***** /
TypUpsVmVoid UpsVmAPI MethodeChercherTypeClientDCom(TypUpsVmSession
 *Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmPointeurDonnees
 riid)
/* Objet : Methode de recherche d'un type DCom. */
/***** /
{
...
}

```

Texte 78 – Exemple d'une méthode ChercherTypeClientDCom

12.2.3.3 Méthode ChercherTypeClientJava

Le type du prototype de la méthode **ChercherTypeClientJava** est **TypUpsVmMethodeChercherTypeClientJava**. Voici un exemple :

```

/***** /
TypUpsVmVoid UpsVmAPI MethodeChercherTypeClientJava(TypUpsVmSession
 *Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmPointeurDonnees
 Classe)
/* Objet : Methode de recherche d'un type Java. */
/***** /
{
...
}

```

Texte 79 – Exemple d'une méthode ChercherTypeClientJava

12.2.3.4 Méthode ChercherTypeServeurCorba

Le type du prototype de la méthode **ChercherTypeServeurCorba** est **TypUpsVmMethodeChercherTypeServeurCorba**. Voici un exemple :

```


/***** /
TypUpsVmVoid UpsVmAPI MethodeChercherTypeServeurCorba(TypUpsVmSession
 *Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmPointeurDonnees
 *ObjetType)
/* Objet : Methode de recherche d'un type Corba. */
/***** /
{
...
}

```

Texte 80 – Exemple d'une méthode ChercherTypeServeurCorba

12.2.3.5 Méthode ChercherTypeServeurDCom

Le type du prototype de la méthode **ChercherTypeServeurDCom** est **TypUpsVmMethodeChercherTypeServeurDCom**. Voici un exemple :

	Plan d'écriture d'un module	Date rédaction : 16 février 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsKrn-000003-A Plan d'écriture d'un module.doc		

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeChercherTypeServeurDCom(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmPointeurDonnees
    *ObjetType, TypUpsVmPointeurDonnees *riid)
/* Objet : Methode de recherche d'un type DCom. */
/*****/
{
    ...
}

```

Texte 81 – Exemple d'une méthode ChercherTypeServeurDCom

12.2.3.6 Méthode ChercherTypeServeurJava

Le type du prototype de la méthode **ChercherTypeServeurJava** est **TypUpsVmMethodeChercherTypeServeurJava**. Voici un exemple :

```

/*****/
TypUpsVmVoid UpsVmAPI MethodeChercherTypeServeurJava(TypUpsVmSession
    *Session, TypUpsVmAdresse *AdresseObjetType, TypUpsVmPointeurDonnees
    *ObjetType)
/* Objet : Methode de recherche d'un type Java. */
/*****/
{
    ...
}

```

Texte 82 – Exemple d'une méthode ChercherTypeServeurJava

Fin de document