version 3.2

Manuel de paramétrage et de développement

Historique des modifications

version	date	modifications
3.2.0	30 mai 2012	Ajout "Propriétés" ("Paramètres des propriétés")
	24 mai 2012	Ajout Recherche globale (§3.3.5) Ajout des nouvelles options HTMLText (§2.1.3.2 et §3.9.12)
	22 mai 2012	Ajout message de log créé à chaque envoi et à chaque erreur (§3.10.2.3.8).
	14 mai 2012	Ajout de la notion d'utilisateur suppléant (§2.2.3.5, §3.13.8)
	9 mai 2012	Ajout des interfaces spécifiques d'administration(§3.9.11)
	3 mai 2012	Modification §3.3.8 Utilisation de méthodes dans l'interface de recherche détaillée
	27 avril 2012	Ajout m0/m3 méthode de transition (§3.10)
	25 avril 2012	Ajout option "esort" pour les énumérés (§2.1.4.2.8) Trie/ordre des choix de l'énuméré (§2.1.4.6.2.3)
	13 avril 2012	Force du mot de passe. §4.5.5
	27 mars 2012	Class Account remplace User §3.13 Login utilisateur devient modifiable §2.2.3.2
	22 mars 2012	balise @templateController §3.9 méthode de menu @apiExpose §2.1.3.12
	21 mars 2012	Ajout des rôles §2.2, §2.3, §3.13
	30 janvier 2012	Dbaccess.php déplacé dans répertoire context §4
	27 janvier 2012	Déplacement des éléments d'exploitation (§4.3.2, §4.7, §4.8) dans le Manuel d'Exploitation
	26 janvier 2012	Ajout §3.1.3.2.4 Modification des éléments d'importations Ajout §2.1.2.4 (erreurs d'importation)
	9 janvier 2012	Ajout téléchargement de fichiers/données binaires (§3.9.4.1)
	16 décembre 2011	Ajout précision option showempty (§2.1.3.2)
	12 décembre 2011	Ajout précision calcul des droits CV (§2.3.4) et cycle (§3.10.1)
	2 décembre 2011	Ajout de la grammaire pour les aides à la saisie, les attributs calculés et les contraintes.
	29 novembre 2011	Ajout procédure création archive d'import sous Windows. Ajout §2.1.6 Limitations sur la mise à jour d'attribut Précision sur importDocument
	28 novembre 2011	Précision sur la construction des liens §2.1.3.6
	14 novembre 2011	Mise à jour usage Action::getArgument (divers chapitres)
3.1.1	21 septembre 2011	Modification §4.3.3 : configuration des log
	15 septembre 2011	Modification §3.11.7 - classe ActionUsage Modification §3.2.2 - classe ApiUsage **en détail sur le tracker**
3.1.0	12 juillet 2011	Modification §3.2 Méthode doc::store() Modification §3.5 Calcul méthodes statiques Modification §3.3 Ajout d'un itérateur de document
	27 juillet 2011	Ajout §3.1 Publication de module Ajout §3.13 Manipulation des utilisateurs Ajout §3.14 Manipulation DbObj et Transaction Ajout §2.4.1.3 Exportation rapport Modification §2.1.3.2 ajout option d'attribut 'creation' pour type docid Modification §2.2.4.1 administration des utilisateurs Modification §2.6 : commande wsh **en détail sur le tracker**

Table des matières

1 Introduction 1
1.1 Présentation 1
1.2 Informations complémentaires 1
1.2.1. Comité de rédaction 1
1.2.2. Anakeen 1
1.2.3. La communauté dynacase 1
2 Paramétrage 2
2.1 Famille 2
2.1.1. Création d'une famille 2
2.1.1.1 Créer une nouvelle famille depuis l'interface Web 2
2.1.1.2 Présentation des champs de base nécessaires à la création d'une famille 2
2.1.1.3 Exemple d'une famille simple 2
2.1.1.4 Fonctionnement de l'héritage entre les famille 3
2.1.1.5 Créer une famille sous OpenOffice.org et l'importer dans dynacase 3
2.1.2. Importation de documents famille 3
2.1.2.1 But de ce chapitre 3
2.1.2.2 Importer des familles en utilisant l'interface Web 3
2.1.2.3 Importer des familles en utilisant la ligne de commandes 3
2.1.2.4 Comprendre les erreurs d'importation 4
2.1.2.5 Supprimer une famille 4
2.1.2.6 Description du fichier d'importation 5
2.1.3. Propriétés 9
2.1.3.1 Paramètres des propriétés 9
2.1.4. Attributs 10
2.1.4.1 Les attributs d'une famille 11
2.1.4.2 Options des attributs 14
2.1.4.3 Modification d'un attribut père 21
2.1.4.4 Types Spéciaux 21
2.1.4.5 Valeurs par défaut 21
2.1.4.6 Les attributs énumérés 22
2.1.4.7 Créer un lien hypertexte 24
2.1.4.8 Création d'une relation entre documents 24
2.1.4.9 Ajouter une aide à la saisie 25
2.1.4.10 Ajouter un attribut calculé 26
2.1.4.11 Contraintes d'attributs 27
2.1.4.12 Créer une entrée de menu 27
2.1.4.13 Créer un extra lien 29
2.1.5. Paramètres 29
2.1.6. Modification d'une famille 29
2.1.6.1 Modifier un attributs père 29
2.1.6.2 Supprimer un attribut 29
2.1.6.3 Supprimer un attribut père 29
2.1.7. Limitations sur la mise à jour de structure 29
2.1.8. Modifier une famille depuis l'interface Web 30
2.1.8.1 Ajouter un attribut 30
2.1.8.2 Paramétrage d'une famille depuis l'interface 30
2.2 Gestion des utilisateurs 30
2.2.1. Présentation 30
2.2.2. Gestion des groupes 31
2.2.2.1 Création de groupes 31
2.2.2.2 Modification d'un groupe 32
2.2.2.3 Consultation du groupe 33
2.2.2.4 Suppression de groupe 33
2.2.3. Gestion des utilisateurs 34
2.2.3.1 Création d'utilisateurs 34
2.2.3.2 Modification d'utilisateurs 34
2.2.3.3 Consultation d'utilisateurs 35
2.2.3.4 Suppression d'utilisateur 35
2.2.3.5 Titulaires et suppléants 35

2.2.4. Administration ... 35
2.2.4.1 Sécurité ... 35
2.2.4.2 Mises à jour ... 37
2.2.4.3 Importation ... 38
2.2.4.4 Catégories ... 42
2.2.4.5 Configuration LDAP ... 43
2.2.5. Connexions anonymes ... 47

2.3 Sécurité ... 47

2.3.1. Les droits applicatifs ... 47
2.3.1.1 FDL - Bibliothèque dynacase ... 47
2.3.1.2 GENERIC - Manipulation d'une famille ... 47
2.3.1.3 ONEFAM - Manipulation groupement de familles ... 48
2.3.1.4 FREEDOM - Doc Admin ... 48
2.3.2. Profils de documents ... 48
2.3.2.1 Définition des droits ... 48
2.3.2.2 Création de profil ... 49
2.3.2.3 Spécification de profils ... 50
2.3.2.4 Affecter manuellement un profil à un document ... 50
2.3.2.5 Profil par défaut des nouveaux documents ... 50
2.3.2.6 Profil par défaut lors de l'enregistrement d'un document dans un dossier ... 50
2.3.2.7 Profil dynamique ... 50
2.3.2.8 Profil de transitions de cycle de vie ... 51
2.3.2.9 Profils d'état de cycle de vie ... 52
2.3.2.10 Points d'emplois des profils de documents ... 53
2.3.2.11 Administration des profils ... 54
2.3.3. Les masques ... 55
2.3.4. Les contrôles de vues ... 56
2.3.4.1 But de ce document ... 56
2.3.4.2 Généralités ... 56
2.3.4.3 Présentation des vues par défaut ... 57
2.3.4.4 Exemple de contrôle de vues ... 58
2.3.4.5 Création de la famille de test ... 58
2.3.4.6 Création des masques ... 58
2.3.4.7 Création du contrôle de vues ... 59
2.3.4.8 Affecter le contrôle de vues à la famille ... 60
2.3.4.9 Test du contrôle de vues manuel ... 61
2.3.4.10 Mise en place de droits sur le contrôle de vues ... 62
2.3.4.11 Test du contrôle de vues automatique ... 62

2.4 Importation et exportation de documents ... 63

2.4.1. Exportation ... 63
2.4.1.1 Exportation CSV ... 63
2.4.1.2 Exportation XML ... 64
2.4.1.3 Exportation Rapport ... 67
2.4.2. Importation ... 68
2.4.2.1 Importation CSV/ODS ... 68
2.4.2.2 Importation XML ... 77
2.4.2.3 Importation en ligne de commande ... 78
2.4.2.4 Création d'archives d'import avec fichiers sous Windows ... 79

2.5 Archives ... 86

2.5.1. Introduction ... 86
2.5.2. Constitution de l'archive ... 86
2.5.3. Archivage des documents ... 88
2.5.4. Destruction des documents archivés ... 89

2.6 Fonctions Wsh ... 91

2.6.1. Comment exécuter une fonction WSH ... 91
2.6.2. listapi ... 91
2.6.3. fdl_adoc ... 91
2.6.4. freedom_clean ... 91
2.6.5. freedom_convert ... 91
2.6.6. importDocuments ... 92
2.6.7. refreshDocuments ... 92

2.6.8. refreshDocuments avec une méthode en argument ... 92
2.6.9. ods2csv ... 93

3 Développement ... 94

3.1 Construire son module ... 94

3.1.1. Introduction ... 94
3.1.1.2 Description de l'espace de développement type ... 95
3.1.1.3 Créer son espace de dévéloppement ... 96
3.1.1.4 Paramétrage et développement Dynacase ... 97
3.1.1.5 Publication ... 97
3.1.1.6 Créer son propre dépôt de paquets ... 99
3.1.2. Construction de modules pour Dynacase ... 99
3.1.3. Fichier ''info.xml'' ... 100
3.1.3.1 Exemple de fichier de info.xml ... 100
3.1.3.2 Description d'un module ... 101
3.1.3.3 Paramètres ... 104
3.1.3.4 Pre/post install/upgrade/etc ... 105
3.1.3.5 Les checks ... 107
3.1.3.6 Les process ... 108
3.1.4. Générer le fichier .webinst ... 111

3.2 Utilisation de l'API ... 113

3.2.1. Présentation de l'utilisation de l'API ... 113
3.2.2. Création d'un script utilisable par wsh ... 113
3.2.3. Manipulation de documents (getParam, new Doc et new_Doc) ... 113
3.2.4. Consultation de documents (getValue, getTValue et getTDoc) ... 115
3.2.4.1 Les propriétés remarquables ... 115
3.2.4.2 Accès aux propriétés (getValue) ... 116
3.2.4.3 Récupérer les différentes valeurs d'un attribut multiple (getTValue) ... 117
3.2.4.4 Récupérer un array complet ou une ligne de l'array (getAValues) ... 118
3.2.4.5 Accès à l'ensemble des attributs (getValues) ... 118
3.2.4.6 Accès aux valeurs des documents (getTdoc) ... 118
3.2.5. Modification de documents (setValue, getValues et deleteValue) ... 119
3.2.6. Création de documents (createDoc) ... 119
3.2.7. Placement de documents dans un dossier (addFile et delFile) ... 121

3.3 Recherche de documents ... 123

3.3.1. Retour de valeurs d'attributs ... 123
3.3.2. Retour d'objets documentaires ... 123
3.3.3. Utilisation des itérateurs ... 124
3.3.3.1 callback sur un itérateur ... 124
3.3.3.2 callback sur un itérateur avec filtrage ... 125
3.3.4. Traitement des erreurs ... 125
3.3.5. Recherche globale ... 126
3.3.5.1 Utilisation de l'option de vérification d'orthographe ... 127
3.3.5.2 Ordonnancement par pertinence ... 127
3.3.5.3 Mise en évidence des mots trouvés ... 127
3.3.6. Itérateur sur une liste d'identifiants ... 128
3.3.7. Recherche dans un dossier et recherche récursive ... 128
3.3.8. Recherche par critère sur les valeurs d'attributs ... 129
3.3.8.1 Recherche sur le titre des documents (title) ... 129
3.3.8.2 Recherche sur n'importe quel attribut (values) ... 129
3.3.8.3 Recherche dans une famille ... 130
3.3.8.4 Recherche avec l'opérateur 'or' ... 130
3.3.8.5 Recherche sur des attributs de type énumérés (getKindDoc) ... 131
3.3.8.6 Recherche dans un array ... 131
3.3.8.7 Recherche avec jointure ... 131
3.3.8.8 Recherche et documents confidentiels ... 132
3.3.8.9 Recherche de familles ... 132
3.3.9. Utilisation de méthodes dans l'interface de recherche détaillée ... 132
3.3.10. Recherche spécialisée ... 134

3.4 Aides à la saisie ... 135

3.4.1. Principe et spécification....135
3.4.2. Syntaxe de la déclaration d'une aide à la saisie....135
3.4.3. Retour unique....136
3.4.4. Retour multiple....137
3.4.5. Cas des attributs relation (docid)....138
3.4.6. Cas des attributs htmltext....139

3.5 Attributs calculés....139

3.5.1. Principe et spécification....139
3.5.2. Syntaxe de la déclaration d'un attribut calculé....139
3.5.3. Calcul simple....139
3.5.4. Utilisation de méthodes statiques....140
3.5.5. Calcul multiple....141
3.5.6. Afficher un message d'avertissement en cas d'erreur (addWarningMsg)....141

3.6 Contraintes....142

3.6.1. Syntaxe de la déclaration d'un attribut calculé....142
3.6.2. La fonction de contrôle....142
3.6.3. Contrainte système....142

3.7 Surcharges des méthodes du document....142

3.7.1. Contrôle à la création....143
3.7.2. Contrôle à la modification....144
3.7.3. Contrôle à la suppression....145
3.7.4. Autres méthodes....145
3.7.5. Ordre d'appel des méthodes....145
3.7.5.1 Consultation de document....145
3.7.5.2 Edition de document....146
3.7.5.3 Création de document....146
3.7.5.4 Sauvegarde de document....146
3.7.5.5 Duplication de document....146
3.7.5.6 Suppression de document....146
3.7.5.7 Import de document....146

3.8 Principales méthodes de la classe Doc....146

3.8.1. lock....146
3.8.2. unlock....147
3.8.3. addRevision....147
3.8.4. attachTimer....147

3.9 Vues et éditions particulières....147

3.9.1. Vues par défaut....147
3.9.2. Syntaxe d'un fichier Layout....149
3.9.2.1 Données atomiques....149
3.9.2.2 Données listes....150
3.9.2.3 Données tableau....151
3.9.2.4 Conditions....152
3.9.2.5 Internationalisation....153
3.9.2.6 Paramètres d'environnement....153
3.9.2.7 Les zones....153
3.9.3. Vue de consultation par défaut....154
3.9.3.1 Consultation avec la méthode par défaut....155
3.9.3.2 Consultation avec méthode spécifique....156
3.9.4. Vues de consultation spécifiques....157
3.9.4.1 Téléchargement de fichiers/données binaires....158
3.9.5. Vues de consultation OpenDocument Text....159
3.9.5.1 Incorporation d'images....161
3.9.5.2 Gestion des attributs de tableaux....161
3.9.5.3 Gestion des conditions....161
3.9.5.4 Attributs multivalués....163
3.9.5.5 Utilisation d'un contrôleur spécifique....163

3.9.5.6 Répétables multi-niveaux....165
3.9.5.7 Précautions d'usage....169
3.9.5.8 Mise à jour des propriétés du document....171
3.9.5.9 Limitations....171
3.9.5.10 Fichiers ODT mixtes....171
3.9.5.11 Vues dynamiques dans les tableaux....173
3.9.5.12 Génération automatique du modèle de document....174
3.9.6. Vues de consultation avec transformation....174
3.9.7. Vues d'édition....175
3.9.8. Interface spécifique de contrôle d'un formulaire document....181
3.9.8.1 Taille de la fenêtre modale....184
3.9.8.2 Manipulation de tableau....184
3.9.8.3 Affectation d'attribut....185
3.9.8.4 Récupération de valeur d'attribut....185
3.9.8.5 Utilisation des attributs de retour définis dans l'aide à la saisie....185
3.9.8.6 Adaptation à l'environnement Ext JS....186
3.9.9. Mise en forme de rangée de tableau....186
3.9.9.1 Exemple de consultation....186
3.9.9.2 Exemple d'édition....188
3.9.9.3 Pour éditer et consulter avec le même template....189
3.9.10. Vues d'attributs....190
3.9.10.1 Vue d'attributs de consultation....190
3.9.10.2 Vue d'attributs d'édition....191
3.9.11. Interfaces spécifiques d'administration....194
3.9.11.1 La zone EDITFAMILYPARAMETER....194
3.9.11.2 La zone EDITAPPLICATIONPARAMETER....195
3.9.11.3 La zone EDITSUBMIT....196
3.9.11.4 Événements liés....198
3.9.12. Interface spécifique d'attribut HTMLText....199
3.10 Cycle de vie....199
3.10.1. Développement....199
3.10.1.1 Définition d'un cycle de vie....199
3.10.1.2 Construction d'un cycle de vie....200
3.10.1.3 Paramètres de transition....207
3.10.1.4 Options de transitions....210
3.10.2. Paramétrage....212
3.10.2.1 Créer un Cycle de vie (Workflow)....212
3.10.2.2 Créer un cycle de vie localisé....222
3.10.2.3 Modèle de courriel pour les cycles....226
3.10.2.4 Ajout de courriel pour les cycles....230
3.10.2.5 Minuteurs....230
3.10.2.6 Les accords....235
3.10.2.7 Cycle de vie de publication de documents....236
3.11 Créer une nouvelle application dynacase....239
3.11.1. But de ce document....239
3.11.2. Arborescence des fichiers d'une application....239
3.11.3. MONAPPLI.app....240
3.11.4. MONAPPLI_init.php....241
3.11.5. Initialisation ou mise à jour de l'application....241
3.11.6. Liens pour avoir d'autres informations....242
3.11.7. Actions particulières....242
3.11.7.1 Principe et spécification des actions particulières....242
3.11.7.2 Introduction....242
3.11.7.3 Fichier 'MONAPP.app'....243
3.11.7.4 Fichier 'MONAPP_init.php'....244
3.11.7.5 Fichier 'monaction1.php....244
3.11.7.6 Fichier monaction1.xml....246
3.11.7.7 Gestion des droits applicatifs....246
3.11.7.8 Initialisation de l'application et des actions....247
3.11.7.9 Action dans les menus contextuels....247
3.11.8. Migration de modules....249
3.11.8.1 Scripts de pré-migration et de post-migration....249
3.11.8.2 Variables d'environnement....250
3.11.8.3 Comparaison de versions....250

3.12 Localisation (Traduction de dynacase dans d'autres langues). .251
3.12.1. But de ce chapitre.... 251
3.12.2. Quel mécanisme utilise dynacase pour la localisation ?....251
3.12.3. Localiser son application.... 251
3.12.3.1 Localisation des familles.... 251
3.12.3.2 Localisation des fichiers .php (Methode, Class, Action, Api,..)....251
3.12.3.3 Localisation des fichiers .xml (Layout et Action)....252
3.12.3.4 Localisation des fichiers .app (Application)....252
3.12.3.5 Localisation des cycles de vie.... 252
3.12.3.6 Localisation des attributs des familles.... 253
3.12.3.7 Recherche des chaines à traduire (Génération fichier .po)....253
3.12.3.8 Comment ça marche.... 253
3.12.4. Traduire un module dynacase dans une autre langue....254
3.12.5. Paramétrage de locale installées.... 256
3.12.6. Outils pour lire les fichiers .po et traduire les chaines....257
3.12.7. Génération des fichiers .mo (binaires).... 257
3.12.8. Initialisation de dynacase et d'Apache pour prendre en compte les localisations. .257
3.12.9. Script pour automatiser les tâches.... 258
3.13 Manipulation des comptes utilisateurs....258
3.13.1. Création d'un utilisateur.... 258
3.13.2. Correspondance entre compte User et document IUSER:....259
3.13.3. Affectation d'utilisateurs dans un groupe.... 259
3.13.4. Suppression d'utilisateur dans un groupe.... 260
3.13.5. Récupération des membres d'un groupe.... 261
3.13.6. Récupération des utilisateurs associés à un rôle....261
3.13.7. Récupération des rôles d'un utilisateur....261
3.13.8. Suppléants et titulaires.... 262
3.14 Manipulation des classes DbObj.... 262
3.14.1. Récupération d'un objet.... 262
3.14.2. Recherche d'objets.... 263
3.14.3. Transactions et savePoint.... 263
3.15 Bases de données PostgreSQL....265
3.15.1. Tables de la base anakeen....265
3.15.1.1 acl....265
3.15.1.2 action....265
3.15.1.3 application....266
3.15.1.4 docfrom.... 266
3.15.1.5 docname....266
3.15.1.6 domain....266
3.15.1.7 groups....266
3.15.1.8 mailaccount.... 266
3.15.1.9 paramdef.... 266
3.15.1.10 paramv.... 267
3.15.1.11 permission.... 267
3.15.1.12 session_conf.... 267
3.15.1.13 sessions.... 267
3.15.1.14 style....267
3.15.1.15 users....267
3.15.2. Tables de la base dynacase.... 268
3.15.2.1 Héritage entre les tables....268
3.15.2.2 doc....268
3.15.2.3 doc<idfam>.... 269
3.15.2.4 docattr.... 269
3.15.2.5 docattrldap.... 269
3.15.2.6 docfam....269
3.15.2.7 docfrom.... 269
3.15.2.8 dochisto.... 269
3.15.2.9 docname....269
3.15.2.10 docperm.... 269
3.15.2.11 docread....269
3.15.2.12 docrel....270

3.15.2.13 docutag....270
3.15.2.14 docvaultindex....270
3.15.2.15 fld....270
3.15.2.16 groups....270
3.15.2.17 pg_ts_cfg, pg_ts_cfgmap, pg_ts_dict et pg_ts_parser....270
3.15.2.18 vgroup....271
3.15.3. Tables du vault (base dynacase)....271
3.15.3.1 vaultdiskdirstorage....271
3.15.3.2 vaultdiskfsstorage....271
3.15.3.3 vaultdiskstorage....271
3.16 Méthodes et fonctions dépréciées....271
3.16.1. Recherches....271
3.16.2. Classe SearchDoc....272
3.16.3. Classe Doc....272
3.16.4. Class OooLayout....272
3.16.5. Class Action....272
3.16.6. Famille IUSER....272
3.16.7. Cycle de vie....272
3.16.8. fonctions Wsh....272
4 Exploitation....274
4.1 dbaccess.php....274
4.1.1. Fichier ''dbaccess.php'' et "local-dbaccess.php"....274
4.1.2. Fichier `dbaccess.php'....274
4.1.3. Fichier `local-dbaccess.php'....274
4.1.3.1 Variables principales....275
4.1.3.2 freedom_authtype....275
4.1.3.3 freedom_authprovider....276
4.1.4. Pré-sélection du mode d'authentification....277
4.1.4.1 Exemple....277
4.2 Développer un provider....278
4.2.1. validateCredential()....278
4.2.2. validateAuthorization()....278
4.2.3. __construct()....279
4.2.4. initializeUser()....279
4.2.5. Exemple....279
4.2.5.1 Fichier `context/local-dbaccess.php'....279
4.2.5.2 Fichier `WHAT/providers/Class.pamProvider.php'....279
4.2.5.3 Fichier `/etc/pam.d/freedom'....281
4.3 Debuggage....281
4.3.1. Affichage des avertissements PHP....281
4.3.2. dynacase en mode debug....281
4.3.3. Tracer le temps d'exécution des actions dans dynacase....282
4.3.4. Firebug....283
4.4 Exécution périodique....283
4.5 Paramétrage de dynacase....284
4.5.1. Les modules de dynacase....284
4.5.2. Accès au paramétrage des modules de dynacase....284
4.5.3. ACCESS - Accessibilités....285
4.5.4. AUTHENT - Authent....285
4.5.4.1 Paramètres pour la fenêtre de connexion....285
4.5.4.2 Paramètre pour la gestion des expirations des comptes internes....285
4.5.4.3 Paramètre pour la force du mot de passe (comptes internes uniquement)....285
4.5.5. APPMNG - Gestion des applications....286
4.5.6. CORE - Noyau....286

4.5.7. FDL - Bibliothèque dynacase ... 287
4.5.8. GENERIC - Manipulation d'une famille ... 287
4.5.9. FREEDOM - Gestion documentaire ... 288

4.6 VAULT - Coffre de stockage des fichiers ... 288

4.6.1. Interface ... 288
4.6.2. Création d'un coffre ... 288
4.6.3. Modification du volume d'un coffre ... 288
4.6.4. Déplacement d'un coffre ... 288
4.6.5. Commande shell dynacase ... 289

4.7 Type MIME ... 289

4.7.1. Fichier d'association local `$CONTEXT_ROOT/admin/mime-user.conf' ... 289
4.7.2. Fichier d'association global `$CONTEXT_ROOT/admin/mime.conf' ... 289

4.8 Export Première Forme Normale ... 290

4.8.1. Principe ... 290
4.8.2. Préambule ... 290
4.8.3. Pré-requis ... 291
4.8.4. Paramètres du script WSH ... 292
4.8.5. Fichier XML ... 292
4.8.6. Fonctionnement ... 293
4.8.6.1 Phases d'exportation ... 293
4.8.6.2 Erreurs détectées ... 293
4.8.6.3 Attributs supportés ... 294
4.8.6.4 Quelques exemples ... 296
4.8.7. Quelques chiffres ... 297

4.9 Aide en ligne ... 298

4.9.1. Principe et fonctionnement général ... 298
4.9.2. Utilisation ... 298
4.9.3. Paramétrage de la famille aide en ligne ... 299
4.9.4. Droits ... 300
4.9.5. Création d'une aide en ligne ... 300
4.9.5.1 Liées à une famille ... 300
4.9.5.2 Autonome ... 300
4.9.6. Édition d'une aide en ligne ... 301
4.9.6.1 Vue d'édition ... 301
4.9.6.2 Traduction du nom de l'aide et de sa description ... 301
4.9.6.3 Ajout d'une rubrique ... 301
4.9.6.4 Organiser les rubriques ... 301
4.9.6.5 Traduction des rubriques ... 302
4.9.6.6 Liens inter-aides ... 302
4.9.7. Consultation d'une aide en ligne ... 302
4.9.7.1 Accès à la consultation via un attribut de famille ... 302
4.9.7.2 Vue de consultation ... 303

1 Introduction

1.1 Présentation

Dynacase-platform permet de construire des applications métiers pour des structures confrontés aux problématiques de gestion de contenu (l'ECM) et du Case Management.

Ce document décrit les mécanismes proposés et leur mise en oeuvre. Il est articulé en deux parties. La première présente le paramétrage qui permet de préciser et modifier les comportements du produit sans avoir recours au scripting ou codage. La seconde apporte des mécanismes plus complets nécessitant eux des bases dans le langage PHP. En préambule à l'utilisation de ce document, nous précisons qu'il est important de maîtriser les concepts dynacase.

1.2 Informations complémentaires

Le wiki accessible à l'adresse http://www.dynacase.org offre l'ensemble de la documentation dynacase-platform et des modules associées.

1.2.1. Comité de rédaction

Ce manuel est le fruit du travail de l'équipe de développement Anakeen amenée par EBR, assisté de MCL et dirigée par YLB.

1.2.2. Anakeen

Anakeen est éditeur du produit dyncase-platform. Société couvrant l'ensemble des prestations habituelles d'un éditeur, Anakeen a décidé depuis sa création de proposer des produits sous licence libre.

1.2.3. La communauté dynacase

Elle existe au travers des outils d'échange accessibles via le site communautaire dynacase. Cette communauté permet à chaque jour de corriger, préciser ce manuel par ses remarques pertinentes.

2 Paramétrage

Cette première partie décrit le paramétrage que l'on peut effectuer sur dynacase sans utilisation de codage PHP.

2.1 Famille

La famille sert à décrire en ensemble de document ayant les mêmes caractéristiques que ce soit en terme de contenu ou en terme de comportement. Les familles peuvent être diverses comme par exemple les compte-rendus, les notes techniques, les factures, les sociétés, les recettes de cuisines.

2.1.1. Création d'une famille

2.1.1.1 Créer une nouvelle famille depuis l'interface Web

Pour créer une nouvelle famille, il faut :

- Aller dans le module “Gestion documentaire”
- Sélectionner le menu “Création / Famille”
- Indiquer de quel famille la nouvelle famille doit hériter
- Indiquer le nom de la famille
- Cliquer sur “Créer une nouvelle famille”
- Faire un clic droit sur la nouvelle famille et sélectionner le menu “Éditer les attributs”

La fenêtre d'édition présentent les attributs de la famille et les attributs hérités qui sont non modifiables.

En bas du tableau, trois nouveaux attributs peuvent aussi être ajoutés. Si vous voulez en ajouter plus de trois il faut le faire trois par trois. Une fois les trois premiers saisis, il faut valider et faire :

- Clic droit sur la famille
- Menu “Éditer les attributs”

2.1.1.2 Présentation des champs de base nécessaires à la création d'une famille

Champ	Description
Identificateur ou ID	Nom interne de l’attribut. Il sert de référence lors des définitions des fonctions et des liens. Cet item n’est plus modifiable une fois crée. Cet identificateur doit être unique dans l’ensemble des familles.
Ordre	Défini l’ordre de présentation des attributs dans le document. Cet ordre n’est pas utile pour les attributs de type *frame.*
Nom	Texte court définissant l’attribut. Ce texte est utilisé pour désigner l’attribut lors de la visualisation ou de l’édition.
Type	Défini le type syntaxique de l’attribut (text, htmltext, frame, date, time,...)
Cadre ou fenêtre	Défini le cadre dans lequel l’attribut doit être présenté. Il faut auparavant avoir défini les cadres.
Résumé ou R?	Si vous voulez que l’attribut soit affiché dans le mode colonne ou dans les aperçus de documents dans l’application GENERIC, il faut cocher la colonne ‘R?’.
Titre ou T?	Le titre d’un document peut être composé par les valeurs d’un ou de plusieurs attributs. Il faut au moins mettre un des attributs en titre en cochant la colonne “T?”.
Obligatoire ou O?	Indique si l’attribut est obligatoire lors de la saisie.
Visibilité ou Vi	Indique dans quel cas l’attribut est affiché.

2.1.1.3 Exemple d'une famille simple

Cet exemple est celui de la famille “de base” :

// famille de base		titre	id	class	name					
BEGIN		0 de base			BASE					
TYPE	C									
//	idattr	idframe	label	T	A	type	ord	vis	need	link
ATTR	FR_BASIC		basique	N	N	frame		0 W		
ATTR	BA_TITLE	FR_BASIC	titre	Y	N	text		1 O	Y	
END										

Commentaires :

- La première ligne permet de définir le cadre (bordure qui entoure les champs)
- La deuxième ligne permet de définir un champ de type “text” qui sera affiché dans la vue résumé et utilisé dans le titre du document. Ce champ est obligatoire

Remarque : Pour avoir d'autres exemples, le plus simple est de regarder le paramétrage des familles disponibles en standard

2.1.1.4 Fonctionnement de l'héritage entre les famille

Les familles sont soumises à une hiérarchie. On peut ainsi définir des familles spécialisées à partir de définition plus générale, on parle alors d’héritage.

Par exemple, Nous avons la famille “société” dans laquelle figure toutes les coordonnées d’une entreprise et on souhaite définir une famille “site” qui permettra de visualiser les différents sites de cette entreprise. Nous allons donc créer la famille “site” à partir de la famille “société”. On dit alors que la famille “site” hérite de la famille “société” et de tous ses attributs. La famille héritée pourra alors être modifiée pour caractériser précisément les sites de l’entreprise.

2.1.1.5 Créer une famille sous OpenOffice.org et l'importer dans dynacase

L'interface Web sert juste à ajuster les attributs d’une famille. Pour accéder à toutes les possibilités d'une famille, il faut la créer dans OpenOffice.org et l'importer dans dynacase. Tout est expliqué dans le chapitre suivant.

2.1.2. Importation de documents famille

2.1.2.1 But de ce chapitre

- Expliquer comment importer des familles dans dynacase en utilisant l'interface Web ou la ligne de commandes.
- Expliquer comment construire un fichier OpenOffice.org ou .csv permettant de définir les attributs d'une famille à importer.

2.1.2.2 Importer des familles en utilisant l'interface Web

Avant d'importer une famille, il est conseillé de suivre le chapitre concernant l'importation de documents en général.

Mais en résumé, voici ce qu'il faut faire :

- Construire un fichier OpenOffice.org ou .cvs en suivant les instructions des chapitres suivants
- Se placer dans le module “Gestion documentaire” de dynacase
- Utiliser le menu “Outils / Importer des documents”
- Cliquer sur le bouton “Parcourir” et sélectionner le document .csv ou OpenOffice.org
- Cliquer sur le bouton “Analyse” pour détecter les anomalies
- Cliquer sur le bouton “Importer les document”

Le bouton "Importer les documents" est grisé (inactif) lors de l'ouverture de la page d'import. Il se dégrise si l'analyse est correcte et si le nombre de document est inférieur au paramètre applicatif " FDL_MAX_FGEXPORTDOC" (par défaut 20 - bibliothèque dynacase). Si le nombre de famille est supérieur seul l'import en tâche de fond sera activable.

2.1.2.3 Importer des familles en utilisant la ligne de commandes

Voici la commande qu'il faut utiliser pour importer un document dans dynacase :

```
$ sudo /var/www/wiff/wiff context Développement exec /bin/bash --login
[sudo] password for eric
www-data@tarfful:~$ ./
./wsh.php --api=importDocuments --file=/chemin/VotreFichier.ods
```

Remarque : Vous pouvez importer directement un fichier OpenOffice.org .ods ou un fichier .csv

Si vous êtes en phase de mise au point vous pouvez utiliser l'option reinitattr pour supprimer les attributs des familles déclarées dans le fichier avant l'importation

```
./wsh.php --api=importDocuments --reinitattr=yes --file=VotreFichier.ods
```

2.1.2.4 Comprendre les erreurs d'importation

La famille est importée si aucune erreur n'est détectée.

Dans le cas où des erreurs sont remontées, elle sont présentées de la façon suivante :

ERROR : {*<CODEERREUR>*} *<message d'erreur>*.

Voici un exemple :

```
www-data@luke:~$ ./wsh.php --api=importDocuments --file=DCPTEST/myFamily.ods

ERROR:{ATTR1260} method (phpfunc) "::badCompute1" is not found for "tst_number1"
attribute
ERROR:{ATTR1261} not enought argument call to use method "::goodCompute2" (need
1 arguments) for "tst_number2" attribute
```

Si vous lancez en ligne de commande les erreurs sont affichées sur la sortie stderr et le retour du wsh est "1" (0 si c'est OK).

Dans cet exemple, deux codes erreur sont remontés ATTR1260 et ATTR1261. Le premier indique que la méthode n'est pas trouvée et la deuxième que la méthode "goodCompute2" nécessite au moins un argument. L'ensemble des définitions des codes erreurs et des explications supplémentaires sont disponibles sur http://api.dynacase.org. Il suffit de taper le code erreur pour avoir le descriptif de l'erreur.

Les codes d'erreurs sont classées par catégories. La catégorie est la partie alphabétique du code. Les catégories de code erreurs sont :

ATTR	Erreurs relatives à la structure de la famille. Déclenchées sur les codes 'ATTR', 'PARAM', 'MODATTR'
WFL	Erreurs relatives aux cycles de vies. Déclenchées lors de l'analyse du fichier "classe" décrivant le cycle
PRFD	Déclenché sur le mot clef 'PROFID' lors de la définition du profil de la famille.
PRFLD	Erreurs relatives aux profils de documents. Déclenché sur le mot clef 'PROFIL'
ACCS	Erreurs relatives aux définitions des accès aux applications. Déclenchées sur le mot clef 'ACCESS'.

Pour chacune des catégories, l'ensemble des codes erreurs sont décrits sur le site web de l'api (cité ci-dessus).

2.1.2.5 Supprimer une famille

Une fois l'importation d'une famille effectuée, il n'est plus possible de supprimer des attributs ni de les renommer. Vous serez donc certainement amené pendant votre phase de mise au point à supprimer des familles.

ATTENTION : La suppression d'une famille entraîne la suppression de tous les documents de cette famille.

Pour supprimer une famille, Il faut utiliser une commande wsh

```
$ sudo /var/www/wiff/wiff context Développement exec /bin/bash --login
[sudo] password for eric
www-data@tarfful:~$ ./wsh.php --api=fdl_deletefamily --famid=ZOO_TEST
Destroying [Test :(ZOO_TEST)]
begin;
delete from doc1182;
drop view family."zoo_test";
delete from docname where name='ZOO_TEST'
delete from docfrom where fromid=1182
drop table doc1182;
delete from docattr where docid=1182;
delete from docfam where id=1182;
commit;
Family ZOO_TEST (id : 1182) is destroyed.
```

:

2.1.2.6 Description du fichier d'importation

Dans les chapitres suivants, vous trouverez la description complète d'un fichier d'importation. Ce fichier peut-être au format OpenOffice.org ou au format .csv. Pour le format cvs, l'encodage du texte peut être en utf-8 ou en iso8858-1 (latin1). Attention, si vous créez un fichier csv sous microsoft windows il est probable que l'encodage soit celui du système à savoir windows-cp1252. Bien que ce format soit globalement compatible avec le format iso8859-1, certains caractères comme l'euro ne seront pas reconnus. L'encodage iso8859-1 est déprécié à l'avantage de l'encodage utf-8 qui est plus universel.

Remarque : Une ligne vide ou toute ligne qui commence par un mot-clef non reconnu sera ignoré.

Lors de l'utilisation de openOffice.org pour éditer les structures de familles, il est conseillé d'inhiber toutes les options pour l'autocorrection et les guillemets typographiques.

2.1.2.6.1 Entête

L'entête défini le titre, le nom et l'héritage de la famille.

//	fromid	titre	id	class	name
BEGIN	SOCIETY	site			SITE

Définition des colonnes :

N° de colonne	Obligatoire	Définition
1	Obligatoire	BEGIN
2 (fromid)	Facultatif	Identifiant logique de la famille père. Mettre vide s'il n'y a pas d'héritage. Cet identifiant peut être numérique mais ce cas d'usage est réservé au document système du noyau.
4 (titre)	Obligatoire	Titre de la famille. Ce titre désigne sera utilisé sur les interfaces pour désigner la famille. Il peut être ajusté avec les catalogues de langues afin d'avoir des libellés suivant la locale de l'utilisateur connecté (voir §localisation).
5 (id)	Réservé	identifiant numérique de la famille. Le laisser à vide pour utiliser un identifiant logique. S'il est valué il faut que cet identifiant ne soit pas déjà pris par un autre document. Les valeurs entre 900 et 999 peuvent être utilisée pour vos besoins spécifiques. L'usage de valeurs numériques fixe est fortement déconseillée.
6 (class)	Facultatif	le nom d'un fichier de définition de classe PHP qui doit être présent sur le serveur (<installdir>/FDL). Ne sert que pour des documents systèmes. Permet un héritage autre que celui prévu par défaut par les classes documentaire.
7 (name)	Obligatoire	nom logique de la famille. Ne doit contenir que des caractères alphanumériques les caractères _ et : sont autorisés. (pas d'espace, ni de ponctuation).

2.1.2.6.2 Caractéristiques générales

Les caractéristiques permettent d'affecter les propriétés du documents. La syntaxe est toujours la même : 1ère colonne : nom de la propriété, 2ème colonne valeur de la propriété.

propriété	description
TYPE	toujours mettre C. Défini la propriété doctype du document. 'C' signifie famille de document.
ICON	nom du fichier image définissant l'icône de la famille. Cette icône doit être une image de taille 48×48 pixels. Elle doit être présente sur le serveur. Cette icône peut être modifié à postériori par l'interface. Si une icône est déjà présente pour cette famille, elle ne sera pas modifiée.
PROFID	identifiant du document profil pour cette famille. Ce document doit être de la famille 'profil de famille'. Si la valeur est vide le profil de famille est enlevé.
CPROFID	identifiant de profil qui sera affecté à tout nouveau document crée par cette famille. Ce document doit être de la famille profil de document. Si la famille hérité de dossier, le document doit être de la famille profil de dossier. Si la famille hérite de recherche, le document doit être de la famille profil de recherche. Si la valeur est vide le profil de document par défaut est enlevé.
DFLDID	identifiant du dossier principal permettant de constituer une arborescence spécifique à la famille . Ce dossier est nécessaire pour manipuler les documents d'une famille depuis l'application "ONEFAM". Il peut être égal au 'auto', ce qui a pour effet de créer un dossier principal automatiquement. Si cette propriété est déjà renseignée, la nouvelle valeur ne sera pas prise en compte.
METHOD	Indique le nom du fichier contenant les méthodes supplémentaires de la famille. Cette propriété peut être utilisé sur plusieurs lignes. Dans ce cas les valeurs de lignes suivant devront être précédée du caractère '+'. Si on précède le valeur par '*', cela indique que ce fichier n'est pas intégré directement dans la famille; mais qu'il est utilisé comme méthodes héritées. Cela implique que les méthodes définies dans ce fichier peuvent être redéfinies dans un autre fichier méthodes. Si la valeur est vide les méthodes associées seront enlevées.

	METHOD;Method.Principale.php METHOD;+Method.Secondaire.php	

	METHOD;Method.Principale.php METHOD;*Method.Secondaire.php Dans ce cas, le fichier Method.Principale.php peut surcharger des méthodes définies dans Method.Secondaire.php	

WID	identifiant du cycle de vie associé à la famille Si la valeur est vide le cycle par défaut est enlevé.
CVID	identifiant du contrôle de vue associé à la famille Si la valeur est vide le contrôle de vue par défaut est enlevé.
SCHAR	caractéristique spéciale pour le révision : 'R' : signifie auto révision à chaque modification 'S' : signifie document non révisable Si la valeur est vide la caractéristique spéciale par défaut est enlevée.
USEFOR	caractère désignant une utilisation spéciale. Seulement pour les documents systèmes : 'W' : pour les cycles de vie 'G' : pour les intercalaires de chemise 'P' : pour les profils Si la valeur est vide l'utilisation spéciale par défaut est enlevée.
TAG	caractère désignant une marque applicative. Cette marque peut être utilisé sur plusieurs lignes ce qui a effet d'ajouter une marque. Les documents crées auront aussi cette marque. Cette clef peut apparaître plusieurs fois, cela ajoute le tag à chaque apparition de cette clef. Une valeur vide ne supprime pas le tag. Les tags applicatifs ne peuvent être supprimés par cette cette directive.
RESET	mettre la valeur "*attributes*" afin d'effacer les attributs avant de réimporter les attributs (équivant à l'option --reinitattr=yes de la commande wsh --api=importDocuments) mais seulement pour la famille concerné. mettre la valeur "*default*" afin d'effacer les valeurs par défaut enregistré. Afin de remettre de nouvelles valeur par défaut. mettre la valeur "*properties*" afin d'effacer les paramètres des propriétés (ex. paramètre "sort" des propriétés).

2.1.2.6.3 Fin de définition

Toutes définitions de familles doit ce terminer par le mot-clef END .

END		

Remarque : Ensuite, après cette ligne, il est possible de définir une autre famille.

2.1.3. Propriétés

2.1.3.1 Paramètres des propriétés

Des paramètres peuvent être positionnés sur les propriétés pour en modifier le comportement.

2.1.3.1.1 Définition des paramètres des propriétés

Les paramètres d'une propriété sont positionnés à l'aide du mot-clef "PROP", suivi du nom de la propriété, puis de la définition du paramètre sous la forme "*parameterName=parameterValue*".

Exemple :

```
BEGIN     BASE0          image                                IMAGE

TYPE      C
SCHAR     R
ICON      photo.gif

PROP      owner          sort=no
PROP      title          sort=asc
PROP      initid         sort=desc

//        idattr         idframe     label          T     A     type     ord     vis    need linkphpf:phpfunc elink     constraint options

...

END       -
```

2.1.3.1.2 Liste des paramètres des propriétés

2.1.3.1.2.1 sort

Le paramètre "*sort*" permet de spécifier si la propriété est disponible dans les recherches et les rapports, et quel est son ordre de trie par défaut.

Syntaxe :

```
sort=<no|asc|desc>
```

"*no*" permet d'inhiber le paramètre "*sort*" défini par défaut pour une propriété.

"*asc*" indique que la propriété est disponible dans les recherches et les rapports, et que le trie ascendant s'applique par défaut.

"*desc*" indique que la propriété est disponible dans les recherches et les rapports, et que le trie descendant s'applique par défaut.

Les propriétés qui ont un paramètre "*sort*" par défaut sont :

Nom de la propriété	Valeur du paramètre "sort"
title	asc
initid	desc
revdate	desc
state	asc

2.1.4. Attributs

exemple famille image :

```
TYPE      C
SCHAR     R
ICON      photo.gif
//        idattr          idframe         label           T   A   type      ord  vis  need linkphpf:phpfunc  elink  constraint  options

ATTR      IMG_FRFILE                      image           N   N   frame        0 W
ATTR      IMG_TITLE       IMG_FRFILE      titre           Y   N   text        10 R
ATTR      IMG_FILE        IMG_FRFILE      image           N   Y   image       20 W   Y                                          iwidth=80
ATTR      IMG_DESCRIPTION IMG_FRFILE      description     N   Y   longtext    30 W
ATTR      IMG_FR_CHAR                     caractéristiques N  N   frame       40 W
ATTR      IMG_CATG        IMG_FR_CHAR     catégorie       N   N   enum        50 W                 1|photo,2|dessin,1.1|paysage,1.2|port
END       -
```

Définition des colonnes :

1. ATTR
2. identifiant de l'attribut
3. identifiant du cadre ou du tableau englobant
4. désignation de l'attribut
5. 'Y' signifie qu'il fait partie du titre, 'N' il ne fait pas partie du titre
6. 'Y' signifie qu'il fait partie du résumé, 'N' il ne fait pas partie du résumé
7. type de l'attribut
8. ordre de l'attribut
9. visibilité de l'attribut
10. 'Y' signifie qu'il est obligatoire, 'N' il ne fait pas obligatoire
11. définition de l'hyperlien
12. nom du fichier php pour l'aide à la saisie
13. nom et attributs de la fonction pour l'aide à la saisie ou nom et attributs de la méthode de calcul (précédé de ::) s'il s'agit d'un attribut calculé (visibilité W)
14. extra lien
15. nom et attribut de la méthode de contrainte

16. options de présentation séparées par des '|'.

2.1.4.1 Les attributs d'une famille

Ce chapitre donne la description des colonnes permettant de créer ou modifier des familles.

Dans les chapitres suivants vous trouverez la description des autres colonnes pour réaliser des aides à la saisie, des menus,..

Rappel : Pour accéder à la modification d'une famille, il faut faire un clic droit sur la famille et sélectionner le menu “Éditer les attributs”

2.1.4.1.1 Colonne "Identificateur" ou "ID"

Nom interne de l'attribut.

Il sert de référence lors des définitions des fonctions et des liens. Cet item n'est plus modifiable une fois crée car il peut être utilisé dans les parties spécifiques de la famille.

Cet identificateur doit être unique dans l'ensemble des familles. Il est conseillé de préfixer chacun des attributs par un trigramme identifiant la famille afin de bien percevoir les attributs par famille.

2.1.4.1.2 Colonne "Ordre"

Défini l'ordre de présentation des attributs dans le document. Cet ordre n'est pas utile pour les attributs de type *frame.*

2.1.4.1.3 Colonne "Nom"

Texte court définissant l'attribut. Ce texte est utilisé pour désigner l'attribut lors de la visualisation ou de l'édition.

2.1.4.1.4 Colonne "Type"

Défini le type syntaxique de l'attribut. Les types de base sont :

Type	Description
text	texte court sur une ligne
longtext	texte long sur plusieurs lignes
htmltext	texte formatable
xml	texte long sur plusieurs lignes en XML
frame	cadre dans lequel les attributs sont présentés
date	défini un attribut date de la forme JJ/MM/AAAA
time	heures et minutes
timestamp	date, heures et minutes
password	texte caché. Il est non visible. Des étoiles remplacent les caractères saisis.
file	fichier à télécharger.
image	image à télécharger. Formats image supportés : GIF, JPEG, PNG.
integer	nombre entier
double	nombre réel
money	nombre réel avec 2 chiffres après la virgule
enum	liste énumérée
docid(”[ID_DE_FAMILLE]”)	Lien vers un document de la famille ID_DE_FAMILLE [1] Attention a bien mettre des d typographique.
docid	identifiant de document[2]
account	identifiant de compte (utilisat
color	code RGB (rouge/vert/bleu) e
ifile	fichier intégré. Utilisable pour directement dans le documer
idoc	document intégré.[3]

1) *si ID_DE_FAMILLE n'existe pas lien vers n'importe quel document*
2) *Déprécié*

menu	lien vers une autre page.

2.1.4.1.4.1 Formater les attributs

Pour les attributs “text” un formatage du type (style langage C) peut être ajouté. Par exemple :

- text(”%s environ”) “pour postfixer `environ` après la valeur,
- text(”<B>%s</B>”) “pour afficher l'attribut en gras.

Ceci est aussi valable pour les type “int” ou “double”

- double(”%.02f”) : pour afficher un nombre avec 2 décimales
- integer(”%d m²”) : pour afficher (100 m²).
- double(”%.02f %%”) : pour afficher le signe pourcentage après le nombre

Pour les attributs “date”,”time” et “timestamp” un formatage (style langage C ”strftime”) peut être ajouté. Par exemple :

- time(”%H:%M:%S”) pour afficher aussi les secondes,
- timestamp(”%A %d %B %Y %X”) donne « samedi 08 janvier 2005 10:13:00 ».

2.1.4.1.5 Colonne "Cadre" ou "fenêtre"

Défini le cadre dans lequel l'attribut doit être présenté. Il faut auparavant avoir défini les cadres.

2.1.4.1.6 Colonne "Résumé" ou "R?"

Si vous voulez que l'attribut soit affiché dans le mode colonne ou dans les aperçus de documents dans l'application GENERIC, il faut cocher la colonne 'R?'.

2.1.4.1.7 Colonne "Titre" ou "T?"

Le titre d'un document peut être composé par les valeurs d'un ou de plusieurs attributs. Il faut au moins mettre un des attributs en titre en cochant la colonne “T?”. Le titre est composé en concaténant les valeurs des attributs titre suivant l'ordre donné.

Si vous modifiez la définition du titre, il est nécessaire d'utiliser la fonction Bash (“refreshDocuments”) pour que les documents déjà créés aient leur titre modifié.

2.1.4.1.8 Colonne "Obligatoire" ou "O?"

Indique si l'attribut est obligatoire lors de la saisie.

2.1.4.1.9 Colonne "Visibilité" ou "Vi"

Cet attribut détermine dans quel cas l'attribut est affiché. La visibilité est :

Code	Description
W	attribut visible en lecture et modifiable en édition.
R	attribut visible en lecture seulement. Généralement ce sont des attributs calculés.
H	attribut caché. Généralement ce sont des attributs servant soit au calcul, soit à la génération des liens. Leur valeur n’est généralement pas exploitable directement.
O	attribut modifiable en édition mais non visible en lecture. Généralement utilisé en corrélation avec les attributs de type menu.
S	attribut visible en lecture et en édition, mais non modifiable en édition
U	uniquement pour le type `array`. Interdit l'ajout et la suppression de rangées dans le tableau (version >= 2.7.4). Correspond à “tableau statique”.
I	invisible : l’attribut n’est pas présent en consultation et en édition dans le document. Un attribut de visibilité I n'est pas modifiable (méthode Doc::setValue inactive). Pour en modifier la valeur, il est nécessaire d'appliquer un masque de saisie modifiant sa visibilité

Notes :

– Pour les attributs de type “menu” qui ont une fonction “phpfunc” déclarée, la visibilité appliquée est la visibilité retournée par cette fonction “phpfunc” et non la propriété “Visibilité”.

3) *plus maintenu*

2.1.4.1.10 Les types d'attributs de structure

Pour positionner les champs dans une famille, il existe 3 types d'attributs particuliers :

- frame : Un attribut de type “frame” permet d'ajouter une bordure et un titre pour regrouper des champs similaires
- array : Un attribut de type “array” permet de créer des tableaux
- tab : Un attribut de type “tab” permet de créer des onglets

2.1.4.1.10.1 Attribut de type "frame" - Regroupement de champs

L'attribut de type “frame” permet d'ajouter une bordure et un titre autour des champs.

Dans cet exemple, les deux champs de type “text” seront entourés par une bordure dont le titre est “Exemple de Frame” :

ID	fenêtre	Nom	T?	R?	type	Ordre	Vi	O?
demo_frame	-	Exemple de Frame			frame	1	W	
demo_text1	demo_frame	Texte 1			text	2	W	
demo_text2	demo_frame	Texte 2			text	3	W	

Pour paramétrer une famille, il faut donc commencer par définir les différentes frames et ensuite il faut placer les champs dans les frames créées en utilisant la colonne “fenêtre”

Remarques :

- Si vous souhaitez placer des frames l'une à coté de l'autre et non pas l'une en dessous de l'autre, il faudra créer une vue particulière comme expliqué dans le chapitre "vues de documents".

2.1.4.1.10.2 Attribut de type "array" - Création de tableaux

L'attribut de type “array” permet de construire un attribut qui contient une liste de tuples composés de un ou plusieurs attributs.

	ID	parent	label	T?	R?	type
ATTR	demo_frame		Cadre exemple			frame
ATTR	demo_fichiers	demo_frame	Fichier(s)			array
ATTR	demo_fichier	demo_fichiers	Fichier	N	N	file
ATTR	demo_comment	demo_fichiers	Commentaire	N	N	text

Dans cette exemple, on a un attribut “demo_fichiers” de type array qui contiendra une liste de couples { fichier, champ commentaire }. On a donc une liste de fichiers avec leur commentaire associé :

Fichier(s)		
	Fichier	**Commentaire**
+/-	fichier1	commentaire 1
+/-	fichier2	commentaire 2
+/-	etc.	

Chaque ligne sera préfixé de boutons pour ajouter/supprimer une ligne.

2.1.4.1.10.3 Attribut de type "tab" - Création d'onglets

Le type "tab" est le type structurant de plus haut niveau. Il ne peut avoir de parent. Il ne peut contenir que des attributs de type "frame".

	ID	parent	label	T?	R?	type
ATTR	demo_fr_ident		Identification	N	N	frame
ATTR	demo_title	demo_fr_ident	Titre	Y	N	text
ATTR	demo_tab_un		Onglet n°1	N	N	tab
ATTR	demo_fr_exun	demo_tab_un	Cadre exemple	N	N	frame
ATTR	demo_fichiers	demo_frame	Références	N	N	array
ATTR	demo_fichier	demo_fichiers	Fichier	N	N	file
ATTR	demo_comment	demo_fr_exun	Commentaire	N	N	text
ATTR	demo_tab_deux		Onglet n°2	N	N	tab
ATTR	demo_t_annexes	demo_tab_deux	Cadre exemple	N	N	frame
ATTR	demo_t_annexes	demo_fr_exdeux	Annexes	N	N	array
ATTR	demo_annexe	demo_t_annexes	Annexe	N	N	file
ATTR	demo_remarques	demo_t_annexes	Remarques	N	N	htmltext

Les attributs qui ne sont inclus dans des onglets sont toujours visibles quelques soit l'onglet. Dans l'exemple l'attribut 'demo_titre' est affiché en permanence.

2.1.4.2 Options des attributs

Différentes options de présentation ou de comportement peuvent être appliquées sur les attributs. Ces options se déclarent dans la colonne "Options". Si plusieurs options sont nécessaires, elles doivent être séparé par le caractère | (pipe).

```
exemple : esize=3|elabel=saisissez votre prénom
```

Option	Description	Restriction
	Options générales	
version	Mettre à 'yes' pour indiquer que la valeur de l'attribut fait partie de la composition de la version	
vlabel	Visualisation du libellé en consultation et en édition. Les valeurs possibles sont :. • "left" par défaut affiche le libellé à gauche de la valeur. • "up" l'affiche dessus en souligné. • "none" ne l'affiche pas. Dans le cas de up et none, la valeur de l'attribut prend la largeur de la fenêtre affichée. Pour les cadres et les onglets, les valeurs ne peuvent être que "up" (par défaut) ou "none" Pour les tableaux si le vlabel n'est pas précisé, il est considéré comme "up" en édition et comme "left" en consultation.	
showempty	En consultation, voir le libellé de l'attribut même si sa valeur est vide. Par exemple 'showempty=RIEN' pour afficher RIEN s'il n'y a pas de valeur. Pour afficher juste le libellé mettre simplement un espace comme valeur 'showempty= '. L'option showempty n'est appliquée que pour les vues HTML (elle n'est donc pas appliquée sur les vues OOo).	
sortable	Prend la valeur yes ou no. Cela vaut yes par défaut. Cette option indique si l'attribut est triable.	Version ≥ 3.0.11
	Options pour les hyperliens	
ititle	texte du tooltip du bouton '...' de l'aide à la saisie. Par défaut : « choisissez une valeur »	
ltitle	Texte affichable en popup sur l'hyperlien lorsque la souris passe dessus	hyperlien valide

ltarget	Nom de la fenêtre destinataire de l'hyperlien. Par défaut _self.	hyperlien valide
lconfirm	indique si on veut un message de confirmation avant l'activation du lien. Mettre lconfirm=yes pour activer la confirmation.	hyperlien valide
tconfirm	texte de la confirmation	si lconfirm=yes
autosuggest	En édition, sur une aide à la saisie, indique que la recherche est lancée à chaque modification du texte saisi. (par défaut yes). Mettre à no pour désactiver l'auto-suggestion	version ≥ 2.9.1
Options pour pour les elink (extra lien)		
eltitle	Texte affichable surgissant sur le bouton généré par l'extra-lien.	extra-lien valide
elsymbol	Caractère symbolique affiché sur le bouton généré par l'extra-lien.	extra-lien valide
eltarget	Nom de la fenêtre destinataire sur le bouton généré par l'extra-lien.	extra-lien valide (version >= 2.9.6)

2.1.4.2.1 Options des onglets

Options type tab		
viewonfly	Indique que les attributs qui sont contenus dans l'onglet ne seront chargés sur une interface web que si l'utilisateur clique sur l'onglet. Cela réduit le temps de réponse pour les documents ayant de très nombreux attributs (plusieurs centaines). Cela n'est applicable que pour la consultation par défaut (pas pour l'édition). Mettre à 'yes' pour activer l'option.	Limité aux type tab
firstopen	Indique que cet onglet doit être sélectionné à l'ouverture du document (consultation et rédaction).	version ≥ 2.11.6 - Limité au type tab

2.1.4.2.2 Options des cadres

Options type frame		
bgcolor	Indique si la couleur de fond d'un cadre. cela peut être une couleur simple (exemple : yellow) ou une couleur exprimée en RGB hexadécimal (exemple #FF335A)	version ≥ 2.11.13 - Limité au type frame

2.1.4.2.3 Options des tableaux

Options type array		
sorttable	Indique que le tableau est triable L'utilisateur peut clique sur les entête de tableau pour trier `sorttable=yes`	version ≥ 2.11.8 - Limité au type array
empty	Indique que le tableau s'il est vide ne doit pas afficher la première rangée. `empty=yes`	version ≥ 2.11.8 - Limité au type array
userowadd	Affiche le bouton pour ajouter une rangée. si userowadd=no le bouton d'ajout ne sera pas affiché. Cela permet de ne pas être en conflit si le tableau doit être rempli par un code spécifique sur l'interface. par défaut userowadd est égale à "yes".	Limité au type array
cellbodystyle	style css appliqué sur le cellule de tableau. cellbodystyle=background-color:red	Limité au type array
className	nom d'une classe css appliquée à la cellule	Limité au type array
twidth	largeur du tableau (par défaut twidth=100%)	Limité au type array
cellheadstyle	style css appliqué sur les cellules de l'entête de tableau	
cellbodystyle	style css appliqué sur les cellules du corps de tableau	
align	alignement horizontal pour les cellules de tableau. Valeur possible `left`, `right`, `center`, `justify`	uniquement pour les attributs insérés dans un tableau.
color	couleur du texte pour les cellules de tableau. Soit #RRGGBB soit nom de couleur (red,yellow,...)	uniquement pour les attributs insérés dans un tableau.

bgcolor	couleur du fond des cellules de tableau. Soit #RRGGBB soit nom de couleur (red,yellow,...)	uniquement pour les attributs insérés dans un tableau.
height	hauteur du corps du tableau. En pixel uniquement (pas de pourcentage). Si le corps du tableau dépasse la hauteur spécifiée, un ascenseur vertical apparaitra :!:Ne fonctionne qu'avec firefox	uniquement pour les attributs de type tableau. (version >= 2.8.1)
cwidth	largeur de la colonne pour des attributs présents dans un tableau peut être exprimer en pixel (100px) ou en poucentage (30%)	pour les attributs de tableaux

2.1.4.2.4 Options des fichiers

Options type file, image		
viewfiletype	Indique que le fichier sera vu directement sur le navigateur. Il peut prendre la valeur 'pdf' ou 'image'. Il ne sera pris en compte que si le fichier a une version 'pdf' générée par le moteur de transformation.	Version ≥ 3.0.18 Limité au type file
pdffile	Utilisé avec l'option viewfiletype. Indique l'attribut qui a le fichier pdf correspondant au fichier original.	Version ≥ 3.0.18 Limité au type file
viewfileheigth	Utilisé avec l'option viewfiletype. Indique la hauteur du rendu affiché sur le navigateur. Cette taille peut être exprimée en pixel ou en %. Si 100% est mis, la hauteur sera lié à la hauteur de la fenêtre du navigateur.	Version ≥ 3.0.18 Limité au type file
template	Prend la valeur static ou dynamic. Indique si l'attribut file est un template et s'il doit être recalculé dynamiquement (dynamic) ou pas (static)	Version ≥ 3.0.7 - Limité aux types file
hideindav	Prend la valeur yes ou no. Cela vaut no par défaut. Le fichier correspondant à cet attribut n'apparaîtra jamais dans le dav.	Version ≥ 3.0.11 Limité aux types file et image
preventfilechange	Pour les attributs fichiers, cela ajoute une contrainte pour que le fichier à remplacer provienne de la dernière version du serveur. Cela ne bloque pas un changement de fichier mais cela averti l'utilisateur dans le cas où le fichier ne correspond pas à cette dernière version Cela implique aussi que lors du téléchargement du fichier un code identifiant la version est ajouté dans le nom du fichier (exemple `todo{i47307-56}.ods` pour le fichier `todo.ods`) Mettre `preventfilechange=yes` pour activer cette option	version ≥ 2.12.9 - Limité aux types file et image
search	Indique si l'attribut est recherchable. Mettre à 'no' (par défaut yes) indique que l'attribut ne fera pas l'objet d'une indexation plein texte.	version ≥ 2.9.3 - Limité au type file
inline	Indique si l'image doit être affichée dans la navigateur (mettre `inline=yes`). Par défaut `inline=no`.	version ≥ 2.9.4 - Limité au type image
rn	Méthode renommage d'un fichier. Le fichier est renommé lors du transfert du fichier sur le serveur. La syntaxe est par exemple `rn=::myNewName()`. myNewName est une méthode de l'objet documentaire qui retourne une chaine de caractères. Le premier argument de la méthode est le nom du fichier téléchargé. Il est recommandé que la méthode fournisse une extension compatible avec le type mime pour l'utilisation ultérieure sur le poste client et les transformations. (pour récupérer l'extension d'un nom de fichier vous pouvez utiliser la fonction getFileExtension de la librairie Lib.FileMime.php)	version ≥ 2.11.1 - Limité aux types file et image
iwidth	largeur des images affichées pour la vue par défaut (défaut 80px). Les valeurs doivent être exprimées en pixel	uniquement pour les attributs de type image.

	uniquement (ex: '100px'). Si la valeur est 'auto', l'image sera affichée à sa taille originale	

2.1.4.2.5 Options des textes

Options type text		
elabel	texte du tootips apparaissant lorsque la souris est sur le zone de saisie	
esize	taille de champs de saisie en caractères	

2.1.4.2.6 Options des textes multilignes

Options type longtext		
editheight	hauteur de la zone d'édition : (400px par exemple) . Les valeurs exprimées en pourcentage ne sont pas possibles.	
elabel	texte du tootips apparaissant lorsque la souris est sur le zone de saisie	

2.1.4.2.7 Options des textes formatés

Options type htmltext		
htmlclean	Nettoie les balises de fonts, de style etc. qui sont généralement issu d'un copier/coller (syntaxe : `htmlclean=yes`)	
editheight	hauteur de la zone d'édition : (400px par exemple) . Les valeurs exprimées en pourcentage ne sont pas possibles.	
toolbar	référence de la barre de menu.	Soit `Simple` (présenté par défaut dans Dynacase), `Basic` (juste gras et souligné), `Default` (toolbar par défaut de ckeditor), full (toolbar contenant toutes les options).
toolbarexpand	indique si la barre de menu doit être repliée ou non. Mettre à 'yes' (par défaut) pour la voir. Mettre à 'no' pour la replier	
doclink	Active l'option doclink du HTMLEditor. Cette option se présente sous la forme d'un nouveau bouton (dans la partie réservée au lien dans les toolbar et en fin de toolbar sur la Basic), en cliquant sur le bouton une interface vous propose de sélectionner un document et ajoute une balise de lien vers ce document. Ce plugin s'active à l'aide d'un objet de paramétrage comme ci-dessous : • doclink={"famId": "DIR"} L'objet est en JSON[4] qui doit contenir au moins la propriété famId qui référence le nom logique d'une famille, et qui peut contenir deux autres propriétés : • docrev : dont les valeurs possibles sont les mêmes que la propriété docrev des docid (latest, fixed, state(keystate)) • filter : un filtre SQL qui sera appliqué à la recherche comme celui-ci dessous : • doclink={"famId": "DIR", "docrev" : "fixed"} **Attention** : Cette option nécessite l'installation du module dynacase-ckeditor-plugins	

4) *https://fr.wikipedia.org/wiki/Json*

jsonconf	Cette option permet de configurer soit même l'éditeur de texte. Elle permet notamment de fixer finement le comportement de l'éditeur et le contenu des toolbar. Les options de configuration sont celle de ckeditor (voir http://docs.cksource.com/ckeditor_api/symbols/CKEDITOR.config.html), et l'objet de configuration doit être présenté en JSON[5] valide. De plus, il existe une option de configuration supplémentaire qui permet d'activer un plugin tiers à l'éditeur : • addPlugin : qui doit contenir un tableau (JSON) de nom logique de plugin jsonconf={"addPlugins": ["docattr"]} Cette option ne fonctionne que si le plugin possède une commande ayant le même logique que le plugin, dans ce cas le plugin est chargé et la commande rajoutée en fin de toolbar. Ci-dessous l'option jsonConf permettant d'activer le plugin docattr et le plugin doclink et avec un menu basique et l'activation du mode resize : • jsonconf={"addPlugins" : ["docattr"], "doclink" : {"famId" : "DIR"}, "toolbar" : "basic", "resize_enabled" : "true"} **Attention** : l'utilisation de ce paramètre plus bas niveau que les précédents désactive les précédents. Il n'est pas possible de cumuler le fonctionnement de toolbar,toolbarexpand, editheight, doclink et de jsonconf. Il est par contre possible de construire un jsonconf qui a le même effet que les options précédentes. **Attention** : les options propres à ckeditor (resize, correction orthographique, etc) ne sont pas maintenues par Anakeen et leur bon fonctionnement n'est pas garanti par Anakeen.	

2.1.4.2.8 Options des énumérés

Options type enum		
eunset	Permet d'indiquer qu'un énuméré sera vide par défaut. Sinon il prendra la première des valeurs de l'énuméré. S'il y a une valeur par défaut l'option eunset est inopérante. Mettre eunset=yes	enum seulement
eformat	Mode d'affichage de l'énuméré 1. list (par défaut) 2. vcheck 3. hcheck 4. auto 5. bool	Voir § 2.1.4.6 page 22
etype	Type de l'énuméré 1. close (par défaut) 2. open 3. free	Voir § 2.1.4.6 page 22
esort	Ordre des entrées de l'énuméré 1. none (par défaut) 2. key 3. label	Voir § 2.1.4.6 page 23
bmenu	mettre à no si l'attribut ne doit pas apparaître dans les menus des application GENERIC.	pour les attributs de type enum
system	mettre à yes si les éléments de l'énuméré doivent être écrasés par la nouvelle définition	pour les attributs de type enum
mselectsize	Indique le nombre d'items présentés pour les attributs énumérés multiples. Par défaut 3	version ≥ 2.11.11 - Limité au type enum multiple sans option

5) *https://fr.wikipedia.org/wiki/Json*

		eformat
boolcolor	En consultation, au lieu d'afficher le libelle, un carré de couleur est affiché. Exemple : `boolcolor=red,green` va afficher un carré rouge si pas coché et un carré vert si coché	limité aux énumérés booléen

2.1.4.2.9 Options des relations

Options type docid		

creation	Indique qu'un document de la famille de la relation pourra être créé depuis le formulaire. Un bouton permet d'afficher un nouveau formulaire pour saisir le nouveau document. A la fin de la sauvegarde celui-ci sera inséré dans le document d'origine. Si la relation est déjà renseignée le formulaire de modification du document lié sera affiché sinon une demande de création sera affichée. Pour les relations multiples (option multiple=yes), le nouveau document sera inséré dans la liste des documents.
	La manière la plus simple de le renseigner et de mettre "creation=yes". Ceci indique simplement la mise à disposition d'une création de document avec les valeurs par défaut.
	
	Cette option peut aussi avoir comme paramétrage un objet écrit de la manière suivante :
	`creation={an_name:CT,an_reference :"une référence", an_target:en_source}`
	Cet exemple indique que lors d'une création, l'attribut an_name du nouveau document aura la valeur saisie dans l'attribut relation, l'attribut an_reference aura la valeur "une référence" et l'attribut an_target aura la valeur de l'attribut en_source du formulaire originel au moment de la demande de création.
	
	De plus les options suivantes sont disponibles :
	autoclose :"yes"
	recallhelper :"yes"
	
	L'option autoclose indique que le formulaire annexe une fois validé sera fermé.
	`creation={autoclose :"yes"}`
	L'option recallhelper indique que l'on réappelera la fonction d'aide à la saisie associée après l'ajout. ceci est peut être utile dans le cas où l'aide à la saisie spécifique remplis d'autres attributs que le titre la relation elle même .
	`creation={recallhelper:"yes"}`
	Si le document lié n'est pas accessible en écriture, il sera alors affiché en lecture. De même, le bouton n'apparaîtra que si l'utilisateur a le droit en création sur la famille liée.
		Version ≥ 3.1.0
		Limité au type docid
		
		Le type doit comporter la référence à une famille.

docrev	Pour les attributs docid, si docrev=latest, cela indique que la relation pointe vers la dernière révision document (ceci est la valeur par défaut). Dans ce cas ce sera l'initid du document pointé qui sera stocké en base Si docrev=fixed, cela sera l'id de la dernière révision au moment de l'appel qui sera affecté. Si docrev=state(keystate) alors le lien portera vers le document à l'état `keystate`. L'aide à la saisie filtrera aussi sur les documents à cet état	version ≥ 3.0 - Limité aux types docid en visibilité W

multiple	Indique si la relation a plusieurs choix possibles afin d'indiquer la possibilité de lier plusieurs documents d'une même famille Mettre `multiple=yes`	
doctitle	Pour les attributs relation, indique si un attribut contenant le titre du document pointé par le lien doit être créé automatiquement. Si doctitle=auto le nom de l'attribut titre sera le nom de l'attribut relation suivi de '_title'. Sinon le nom sera celui précisé dans la valeur de l'option doctitle	
isuser	Sert à indiquer à l'interface d'accessibilité des profils quels sont les identificateurs qui peuvent être pris en compte. Ceci sert à filtrer les liens vers des documents qui ne sont pas des utilisateurs ou des groupes d'utilisateurs Mettre `isuser=yes` pour indiquer l'usage dans les profils dynamiques	version ≥ 2.11.3 - Limité au type docid

2.1.4.2.10 Options des comptes

Options type account		
multiple	Afin d'indiquer la possibilité de lier plusieurs comptes Mettre `multiple=yes`	
role	Restriction sur les comptes. Seuls les utilisateurs ayant ce rôle peuvent être indiqués "role=chimist" indique que seuls les utilisateurs ayant le rôle chimiste sont un choix possible "role=chimist,physician" indique une restriction aux rôles chimiste ou physicien. La référence est celle du l'attribut "role_login" (attribut référence).	voir le mécanisme de délégation.
group	Restriction sur les comptes. Seuls les utilisateurs appartenant à ce groupe peuvent être indiqués "group=desk4" indique que seuls les utilisateurs du groupe "desk4" ou ces descendants sont un choix possible. "group=desk4, desk5" indique que les utilisateurs des groupes "desk4 ou desk5 sont des choix possibles	
match	Type de compte : 1. user (par défaut) : compte utilisateur 2. groupe : groupe d'utilisateur 3. role : rôle 4. all : tout type	

2.1.4.2.11 Options des menus

Options type menu et action		
global	Mettre à yes si le lien n'a pas de relation directe avec un document particulier. Dans ce cas, le menu apparaîtra aussi dans le menu 'outils' des applications issues de GENERIC. Ce menu apparaîtra aussi dans le menu contextuel du document *famille*.	pour les attributs de type menu
onlyglobal	Mettre à yes si option est déjà global et si vous ne voulez pas que le menu n'apparaisse pas dans le menu contextuel du document mais uniquement dans le menu 'outils' des applications issues de GENERIC.	pour les attributs de type menu
lconfirm	Mettre à yes pour afficher un message de confirmation avant l'activation du lien	pour les attributs de type menu
tconfirm	Question fermée apparaissant pour la confirmation. Par défaut : "êtes vous sûr ?"	pour les attributs de type menu
mwidth	Largeur de la fenêtre destinataire de l'hyperlien. Par défaut 400px.	pour les attributs de type menu ou action

		(version >= 2.9.6)
mheight	Hauteur de la fenêtre destinataire de l'hyperlien. Par défaut 300px.	pour les attributs de type menu ou action (version >= 2.9.6)
submenu	nom du sous menu	pour les attributs de type menu et action
barmenu	mettre à yes si l'affichage nécessite la barre de menu du navigateur	pour les attributs de type menu et action
batchfolder	Mettre à yes' si l'action définie doit être appliquée sur tous les éléments du dossier	Pour les attributs de type action. Uniquement dans le cadre de famille hérités de dossier

2.1.4.3 Modification d'un attribut père

Pour modifier la définition d'un attribut d'une famille mère, on utilisera **MODATTR** au lieu de ATTR. Cela indique que toute propriété non valué aura la valeur de l'attribut défini dans la famille mère.

2.1.4.4 Types Spéciaux

2.1.4.4.1.1 IATTR

Copie d'attributs d'une frame d'une autre famille.

famille portail			id	class	name
BEGIN	DIR	dossier événements			SCALENDAR
TYPE	C				
	idattr	idframe	id famille		
IATTR	DCAL_FR_PRESENT		DCALENDAR		
END					

Cette exemple copie les attributs et le frame DCAL_FR_PRESENT de la famille DCALENDAR dans la famille SCALENDAR que l'on défini. Cela est utile dans la cas de ressemblance de famille mais sans héritage souhaité.

2.1.4.5 Valeurs par défaut

Les valeurs par défaut sont déclarées à l'aide du mot-clef DEFAULT dans la première colonne.

1	2	3
DEFAULT	SGATE_ACTION	GATE_WEATHER
DEFAULT	SGATE_IDRED	::userDocId()
DEFAULT	SGATE_RED	::getTitle(SGATE_IDRED)
DEFAULT	SGATE_DATE	::getDate()
DEFAULT	SGATE_ARRAY	ligne1\nligne2\nligne3

Définition des colonnes :

1. DEFAULT
2. identifiant de l'attribut
3. valeur par défaut. Soit du texte statique. Soit l'appel à une méthode connue de la famille. Pour les attributs à l'intérieur d'un tableau (array), il est possible d'initialiser plusieurs lignes du tableau en séparant chaque ligne par un '\n'
4. option pour forcer la mise à jour : "force=yes"

Si des valeurs par défaut sont déjà présentes (non vides), les nouvelles valeurs par défaut ne seront pas prises en compte. Néanmoins il est possible de force la réinitialisation des valeurs par défaut.

Les valeurs par défaut sont modifiables par l'administrateur depuis l'interface web.

Pour forcer une valeur en particulier, mettre dans la colonne n°4 le mot clef : "force=yes". Cela remplacera la valeur

par défaut. Pour réinitialiser, toutes les valeurs par défaut, il faut utiliser le mot-clef RESET avec la valeur "default"

RESET	default

Cela est à placer avant les mots-clefs DEFAULT.

Le mot-clef DEFAULT sert aussi à indiquer la valeur par défaut à appliquer aux paramètres de familles. Il suffit d'indiquer un nom de paramètre au lieu d'un nom d'attribut.

2.1.4.6 Les attributs énumérés

2.1.4.6.1 But

Les attributs énumérés peuvent être utilisés pour définir un ensemble fini de choix de valeurs pour un attribut. On distinguera les attributs énumérés **simples** (type “enum”) pour le choix d'une seule valeur parmi l'ensemble des choix, des attributs énumérés **multiples** (type “enumlist”) pour le choix de plusieurs valeurs possibles.

L'ensemble des valeurs est une suite de couples <clef>|<label> séparés par des virgules. Le label est le texte qui est présenté sur l'interface; la clef la valeur qui est stockée en base de données. On peut ainsi modifier les labels sans pour autant changer la clef. La suppression d'une clef implique que c'est directement la clef qui sera affichée (les valeurs en base de données ne sont pas supprimées).

Par défaut un menu de filtrage sur les valeurs de cet attribut est ajouté dans la barre du haut pour toutes applications dérivées de GENERIC.

2.1.4.6.2 Mise en place

Deux options sont disponibles pour les énumérés bmenu et system. Si bmenu égal no, le menu supplémentaire ne sera pas affiché. Si system égal yes, les valeurs définies seront écrasées par les nouvelles valeurs définies. Par défaut, les valeurs des énumérés une fois initialisées ne sont pas modifiables lors de la redéfinition d'une famille.

Par défaut les attributs énumérés sont présentés à l'édition par une liste déroulante pour les énumérés simples et une liste à sélection multiple pour les énumérés multiples.

Plusieurs formats de sélection sont possibles :

- pour les énumérés simples :
  - vcheck : boutons radio (exclusifs) alignés verticalement
  - hcheck : boutons radio alignés horizontalement
  - auto : liste déroulante filtrante
  - bool : un seul bouton boîte à cocher (checkbox). Utilisable seulement pour **les énumérés à 2 valeurs**. Si la boîte n'est pas cochée, c'est la première valeur qui est affectée à l'attribut, et inversement.
- pour les énumérés multiples :
  - vcheck : boîtes à cocher alignées verticalement
  - hcheck : boîtes à cocher alignées horizontalement
  - auto : liste déroulante filtrante

Le format sera décrit par l'option 'eformat'. Pour indiquer qu'un énuméré est multivalué on indiquera 'multiple=yes' dans les options.

Restriction: les énumérés multiples ne peuvent pas être utilisés dans les tableaux.

Pour les énumérés simples, si la valeur par défaut n'est pas spécifiée, c'est le premier choix qui est présenté. Si on veut donner la possibilité de ne rien choisir, il faut ajouter un élément qui a pour valeur espace (red|rouge, |sans avis). La sélection de 'sans avis' aura pour conséquence la suppression de la valeur de l'attribut.

idattr (ou ID)	label (ou Nom)	type	phpfunc (ou ft)	options
TST_COLOR1	ma couleur préférée	enum	red¦rouge,lightgreen¦vert,darkblue¦bleu foncé,yellow¦jaune	bmenu=no
TST_COLOR2	la couleur du vase	enum	red¦rouge,lightgreen¦vert,darkblue¦bleu foncé,yellow¦jaune	bmenu=no¦eformat=hcheck
TST_COLOR3	la couleur de la table	enum	red¦rouge,lightgreen¦vert,darkblue¦bleu foncé,yellow¦jaune	bmenu=no¦eformat=vcheck
TST_COLOR4	aimez-vous le vert ?	enum	red¦rouge,green¦vert	bmenu=no¦eformat=bool
TST_COLOR5	les couleurs de la	enum	red¦rouge,lightgreen¦vert,darkblue	bmenu=no|m

	fleur		¦bleu foncé,yellow¦jaune	ultiple=yes
TST_COLOR6	les couleurs de l'arbre	enum	red¦rouge,lightgreen¦vert,darkblue ¦bleu foncé,yellow¦jaune	bmenu=no¦eformat =hcheck| multiple=yes
TST_COLOR7	les couleurs du ciel	enum	red¦rouge,lightgreen¦vert,darkblue ¦bleu foncé,yellow¦jaune	bmenu=no| eformat=vcheck| multiple=yes

La liste des choix peut aussi être fournie par une fonction PHP ce qui permet une réutilisabilité et une plus grande dynamique dans les choix proposés. Cette fonctionnalité est décrite dans le manuel de programmation.

2.1.4.6.2.1 Type ouvert

etype=open

Les énumérés de type 'ouvert' permettent aux utilisateurs de rajouter des entrées dans la liste des choix possibles. Si l'utilisateur a le droit d'éditer le document il a le droit de rajouter une entrée dans l'énuméré. Par contre, il ne pourra pas l'enlever directement. Seul l'administrateur pourra supprimer/corriger les entrées de cet énuméré.

2.1.4.6.2.2 Type libre

etype=free

Les énumérés de type 'libre' permettent aux utilisateurs de choisir un autre choix que ceux proposés. Ce choix libre n'est pas ajouté à la liste des entrées.

2.1.4.6.2.3 Trie/ordre des choix de l'énuméré

esort=none

Par défaut, les choix de l'énuméré sont présentés dans l'ordre de leur déclaration dans la fonction PHP.

esort=key

Les choix de l'énuméré sont présentés triés par ordre alphabétique de la clef.

esort=label

Les choix de l'énuméré sont présentés triés par ordre alphabétique du label.

2.1.4.6.3 Créer ou modifier des énumérés avec l'interface Web

Pour créer ou modifier les énumérés avec une une interface Web, il faut :

- Faire un clic droit sur la famille et sélectionner le menu “Éditer les énumérés”
- Cliquer sur le bouton correspondant à l'énuméré à modifier
- Utiliser l'interface, pour ajouter, modifier ou supprimer des énumérés

Il est aussi possible de créer des sous-listes dans l'énumération grâce à l'interface web. Il suffit pour cela:

- De créer un nouvel énuméré
- Le deplacer grâce au flèche en dessous de l'énuméré titre
- Le deplacer vers la droite ou la gauche pour changer son niveau

Remarque : si l'énuméré à l'option "system=yes", l'interface de modification de l'énuméré ne sera pas disponible.

2.1.4.6.4 Localisation

Le texte peut être traduit. S'il contient le caractère ”,” ou ”.”, alors il faut précéder ce caractère de ”\”.

2.1.4.6.5 Restrictions/limites

Les caractères, ou séquences de caractères, suivants ne sont pas compatibles pour la composition d'une clef ou d'un label d'énuméré :

- Les caractères ["] (quote), [’] (apostrophe) et [&] (ampersand) sont à proscrire pour la composition des clefs d'énumérés.
- Les séquences [-dot-] (minus "dot" minus) et [-comma-] (minus "comma" minus) sont à proscrire pour la composition des clefs ou des labels d'énumérés.

2.1.4.7 Créer un lien hypertexte

2.1.4.7.1 But

L'hyperlien sert à afficher une URL dans la fenêtre courante à partir de l'attribut. L'hyperlien peut être une simple URL statique (”http://www.meteo.fr”). L'URL peut être décrite avec des paramètres issus du document.

2.1.4.7.2 Création d'un hyperlien

Les références aux attributs du document sont écrites entre les caractères % en indiquant l'identifiant de l'attribut.

Soit l'attribut US_MAIL définissant le mail d'une personne. Pour déclencher l'édition d'un mail vers une personne il suffit de mettre l'hyperlien suivant :

```
mailto:%US_MAIL%
```

Autre exemple, soit l'attribut SI_TOWN indiquant la ville de la famille société. Pour avoir la météo de la ville il suffit de mettre l'hyperlien suivant :

```
http://www.viamichelin.com/viamichelin/fra/dyn/controller/weatherAmbiguous?
strLocation=%SI_TOWN%&strCountry=EUR
```

Le mot-clef %SI_TOWN% sera remplacé par la valeur de cet attribut.

S'il n'y aucune valeur pour un des attributs de l'URL, l'hyperlien ne sera pas affiché (on ne pourra pas cliquer sur l'attribut).

Les propriétés du document peuvent aussi être utilisées en les entourant par le caractère %.

Les propriétés les plus couramment utilisées dans les liens sont :

- %ID% : identifiant du document
- %INITID% : identifiant initial du document (invariable quelque soit la révision du document)
- %TITLE% : titre du document
- %FROMID% : identifiant de la famille du document

Les noms des attributs et des propriétés entre % peuvent être écrites en majuscules ou en minuscules. Le caractère % est obtenu en doublant celui-ci. Le lien "TEST%%25" sera converti en "TEST%25". Les valeurs contenues dans les attributs ou les propriétés sont encodées (RFC 3986). Par exemple si my_test est égal à "bonjour jane & john" alors la valeur encodée de '%my_test%" sera "bonjour%20jane%20%26%20john".

Les mots-clefs spéciaux suivants peuvent être utilisés pour la composition de l'URL :

- %S% : est remplacé par l'URL relative vers dynacase Il doit être utilisé en début de lien.
- %I% : est remplacé par l'identifiant (équivalent à %ID%)
- %T% : est remplacé par le titre (équivalent à %TITLE%)

Les paramètres applicatifs peuvent aussi être utilisés si on les encadre avec des accolades.

Exemple : **%S%**app=TEST&action=TESTONE&arg=**{CORE_CLIENT}**

Ici {CORE_CLIENT} sera remplacé par la valeur du paramètre CORE_CLIENT. Seuls les paramètres de CORE et les paramètres globaux sont accessibles pour la composition de l'url.

2.1.4.8 Création d'une relation entre documents

2.1.4.8.1 Relation simple

Pour créer une relation entre deux documents nous utiliserons le type 'docid'. Pour créer une relation entre le document et un document de la famille 'X' nous déclarerons l'attribut suivant

id	type	visibility
MA_RELATION	docid(“X”)	W

Cela donnera en édition une aide à la saisie classique vers un document de la famille X. En consultation l'utilisateur verra le titre du document lié avec un hyperlien vers celui-ci. Si la famille n'est pas précisé dans le type l'aide à la saisie ne sera pas proposée.

Par contre, seul la n° du document sera stocké et donc l'utilisateur ne pourra pas rechercher par le titre. Si vous voulez que cette relation soit recherchable par le titre il faut rajouter l'option doctitle.

2.1.4.8.2 Relation détaillé

L'aide à la saisie peut être personnalisée. Les documents ci-dessous donne plus d'explications sur la création de liens entre documents :

Soit US_IDSOC l'attribut identifiant le document société de la famille personne. Pour créer un hyperlien vers le document lié il faut saisir l'URL suivante :

```
%S%app=FDL&action=OPENDOC&id=%US_IDSOC%
```

Cette URL indique que l'application FDL va effectuer l'action OPENDOC (affichage d'un document) dont l'identificateur est celui défini dans l'attribut US_IDSOC.

Il est possible d'indiquer un texte lorsque le curseur est sur le lien en utilisant l'option “ltitle”.

Link	...	Options
%S% app=FDL&action=OPENDOC &id=%US_IDSOC%		ltitle=détail de la société

2.1.4.9 Ajouter une aide à la saisie

2.1.4.9.1 Pour les attributs relations

Il est possible de modifier

id	type	vis	phpfile	phpfunc
MA_RELATION	docid("TST_FAM")	W	fdl.php	lfamily(D,FAM,CT):MA_RELATION,CT

L'attribut relation est composé de l'identificateur et du titre du document. Le titre du document n'est pas forcément un attribut défini comme cela est décrit ici. Pour le référencer dans un aide à la saisie on utilise le mot-clef "CT". Ce mot-clef permet de référencer le titre sur l'interface en édition.

2.1.4.9.1.1 Mise en place du fichier PHP et modification du paramétrage de la famille

Défini le nom du fichier PHP qui contient la fonction décrite ci-après. Ce fichier ("phpfile") se trouve dans le répertoire Families/Externals si vous avez utilisé le template de module.

Ce fichier se trouve sur la machine serveur dans le répertoire /<root_install_directory>/EXTERNALS

Les fonctions de base sont décrites dans le fichier “**fdl.php**”.

2.1.4.9.1.2 Mise en place

Les fonctions d'aide à la saisie doivent être des fonctions PHP décrites dans le fichier défini. Leur spécification est décrite dans le chapitre Dévoppement.

La spécification de l'appel suit la syntaxe suivante :

```
NomFonction(P1,P2,...):R1,R2,R3,...
```

- NomFonction : Nom de la fonction PHP à appeler
- P1,P2,...: Paramètres de la fonction. Pour ces attributs, il est possible de mettre:
  - D : coordonnée de la base de donnée Dynacase
  - I : numéro de référence du document
  - {FAMNAME} : FAMNAME étant le nom interne de la famille. Est remplacé par le numéro de référence de la famille
  - {PARAM} : PARAM étant un identificateur de paramètre. Est remplacé par la valeur du paramètre
  - un chiffre : chiffre mis en argument tel quel
  - chaîne de caractère : mise en argument telle quelle
- R1,R2,R3,...: Identificateur du ou des attributs à modifier

La syntaxe complète est donnée au paragraphe "aide à la saisie" de la section "développement".

2.1.4.9.1.3 Exemple de déclaration d'aide à la saisie

La fonction d'aide à la saisie calcule une liste de choix en fonction des paramètres “P”. Lorsque l'utilisateur fait son

choix les attributs “R” sont modifiés.

Une fonction générale est décrite dans le fichier fdl.php. Elle peut être utilisée pour la recherche et pour lier des documents.

Cette fonction est la suivante

```
lfamily($dbaccess, $famid, $name, $catgid=0)
```

Elle possède quatre paramètres dont un optionnel :

- $dbaccess : coordonnées de la base de données
- $famid : identificateur de la famille recherchée
- $name : restriction de la recherche au document dont le titre comprend $name
- $catgid : identificateur de la catégorie, soit le numéro de référence du dossier dans lequel on cherche le document. Zéro indique que la recherche est faite dans toute la base.

La fonction retourne deux paramètres : identificateur du document et titre du document.

2.1.4.9.1.4 Exemple

soit la famille test dont les attributs sont :

- TST_IDSOC : identificateur de la famille société
- TST_SOC : nom de la société

on met un hyperlien sur TST_SOC pour pointer sur le document société qui est lié à la famille test:

```
%S%app=FDL&action=FDL_CARD&id=%TST_IDSOC%
```

Le fichier de TST_SOC est fdl.php et la fonction d'aide à la saisie est : lfamily(D,SOCIETY,TST_SOC):TST_IDSOC,TST_SOC

ceci permet de choisir parmis les documents de la famille société dont le titre contient ce qui est déjà saisi dans l'attribut TST_SOC. Le choix déclenche la mise à jour de l'identificateur et du titre. Une fois le document validé (ou créé), l'hyperlien sera visible sur l'attribut TST_SOC.

2.1.4.10 Ajouter un attribut calculé

2.1.4.10.1 But

Un attribut calculé[6] est un attribut dont la valeur est défini par une méthode. Cette valeur ne peut pas être modifié directement par l'utilisateur. Par conséquent, ce type d'attribut ne peut pas avoir d'aide à la saisie. Sa visibilité est soit H (caché) soit R (lecture).

2.1.4.10.2 Exemple de fonctions

La définition du calcul se fait dans la même colonne que celle de l'aide à la saisie. Le calcul est effectué en appelant la méthode décrite. La valeur retournée par la méthode est affectée à l'attribut. La définition de l'appel de la méthode commence par '::'. Toutes les méthodes de la classe du document peuvent être appelées (méthodes générales (Classe Doc) et méthodes spécifiques).

Par exemple si on veut qu'un attribut de type 'date' contienne la date du jour on indiquera la méthode suivante.

```
::getDate()
```

L'attribut sera remis à jour à chaque rafraîchissement du document (à chaque visualisation complète).

Des paramètres peuvent être ajoutés si la méthode le requiert. Ces paramètres peuvent être des valeurs d'attributs ou des chaînes de caractères quelconques.

```
::getFutureDate(3)
::getFutureDate(ZOO_DELAY)
::getStrDate(%d/%m/%Y,ZOO_CDATE)
```

6) *Voir chapitre Développement/Attribut calculé*

S'il y a plusieurs paramètres, ils doivent être séparés par des virgules.

Le résultat du calcul peut être appliqué à un autre attribut que celui où on a défini le calcul. Cela est utile dans la cas où on utilise un attribut avec aide à la saisie qui sera recalculé avec les paramètres issus de l'aide à la saisie.

Exemple classique : Mettre un hyperlien calculé sur un document:

```
::getTitle(ZOO_IDDOC1):ZOO_DOC1
```

idattr	label	vis	link	phpfile	phpfunc
CMC_IDPROPO	id propo	H			::getTitle(CMC_IDPROPO):CMC_PROPO
CMC_PROPO	proposition	W	%S% app=FDL&action=OPENDOC&mode=view&id=%CMC_IDPROPO%	Ext.php	lpropo(D,CMC_PROPO):CMC_IDPROPO,CMC_PROPO

Dans cet exemple, l'utilisateur choisi une proposition avec la fonction d'aide à la saisie. Ceci à pour but de récupérer l'identificateur. Ensuite, cet identificateur est utilisé pour remettre à jour le titre du lien en cas de changement du titre du document lié.

2.1.4.11 Contraintes d'attributs

2.1.4.11.1 But

Certains attributs peuvent faire l'objet de contraintes spécifiques. C'est à dire que la valeur de l'attribut peut être soumis à certaines conditions que ce soit de syntaxe ou plus générales. La contrainte fait appel à une méthode de la classe d'objet documentaire.

2.1.4.11.2 Mise en place

idattr	label	vis	constraint
SI_BIRTHDATE	date de naissance	W	::pastDate(SI_BIRTHDATE)

On peut comme pour le champ *"phpfunc"* mettre comme paramètre à la méthode des attributs du document.

Si la contrainte de l'attribut n'est pas respectée, le document ne peut être enregistré. Par contre, si vous êtes connecté en temps que utilisateur 'admin', un bouton 'Sauver*!*' apparaîtra afin d'outrepasser les contraintes. Seul cet utilisateur peut utiliser et voir ce bouton.

2.1.4.12 Créer une entrée de menu

2.1.4.12.1 But

Ces attributs permettent d'ajouter des actions particulières dans le menu contextuel du document. Le lien définit l'url qui sera activée lors de la sélection du menu.

2.1.4.12.2 Mise en place

idattr (ou ID)	label (ou Nom)	type	link (ou Lien)	phpfunc (ou ft)	option
CMC_M_M1	Proposition n°1	menu	%S %app=FDL&action=FDL_CARD&id=%CMC_IDPROPO1%		submenu=proposition
CMC_M_M2	Proposition n°2	menu	::callExposePropo()		submenu=proposition
CMC_M_PROPO	Voir la proposition	menu	%S %app=FDL&action=FDL_CARD&id=%CMC_IDPROPO%	::verifyProposition()	lconfirm=yes

La colonne "phpfunc" permet d'indiquer un pré-condition à l'affichage du menu. Cette méthode de l'objet documentaire retourne l'activité possible du lien (actif, inactif, invisible).

La colonne "link" peut aussi faire référence à une méthode de la famille en utilisant la notation '::' comme indiquée dans l'attribut 'CMC_M_M2'. Cette méthode de la famille doit comporter la balise '**@apiExpose**' dans le commentaire pour pouvoir être appelée depuis ce menu. Les méthodes. Une fois la méthode appelée, le document est réaffiché. Le retour d'une méthode exposée est le message d'erreur (vide si pas d'erreur). Attention, une méthode "exposée" doit réaliser elle-même les contrôles de sécurité qui lui semble nécessaire.

```
/**
 * @apiExpose add one to tst_sample attribute
 */
public function callExposePropo() {
   $err=$this->canEdit(); // some security control
   if (!$err) {
      $this->setValue("tst_sample", $this->getValue("tst_sample",0)+1);
      $err=$this->store();
      addWarningMsg('new value '.$this->getValue("tst_sample"));
   }
   return $err;
}
```

2.1.4.12.3 Modifier la fenêtre de destination

Par défaut, le résultat de l'activation d'un item de menu s'ouvre dans une fenêtre autonome. Chaque item de menu à sa propre fenêtre de résultat. La fenêtre cliente de destination peut être modifiée en spécifiant le nom de la fenêtre dans l'option “ltarget”.

Exemple, pour avoir le résultat dans la fenêtre 'test' :

link (ou Lien)	...	options
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		**ltarget=test**

Pour avoir le résultat dans la fenêtre courante on utilisera le nom '_self'

link (ou Lien)	...	options
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		**ltarget=_self**

Pour avoir le résultat dans une fenêtre différente à chaque activation, on utilisera le nom '_blank'

link (ou Lien)	...	options
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		**ltarget=_blank**

2.1.4.12.4 Demander une confirmation pour une action critique

Si le résultat d'une action menu est critique, une confirmation peut-être demandée avant l'exécution de l'action. Cette confirmation est indiquée en mettant l'option **“lconfirm=yes”** .

Le texte de la confirmation peut être personnalisée avec l'option "**tconfirm**=<mon texte>"

link (ou Lien)	...	options
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		ltarget=test|**lconfirm=yes| tconfirm=Action très risquée\nVous voulez continuer ?**

2.1.4.12.5 Mettre le menu accessible uniquement avec la touche CTRL

Si l'item du menu ne doit être utilisé que dans de rare occasion, il peut être visible que lors de l'appui conjugué avec la touche 'Ctrl'. On indique cette possibilité en mettant l'option **“lcontrol=yes”**. Cela indique aussi que le menu sera disponible dans le menu "autres" de la barre de menu du document.

link (ou Lien)	...	options
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		lcontrol=yes
%S%app=FDL&action=FDL_CARD&id=%US_IDSOC%		ltarget=test|lconfirm=yes| lcontrol=yes

2.1.4.13 Créer un extra lien

L'extra lien permet d'afficher un bouton supplémentaire lors de l'édition d'un document. Ce bouton est spécifié par un lien comme pour les hyperlien défini ci-dessus. Cet extra-lien est utilisé le plus souvent pour que l'utilisateur crée ou modifie un document secondaire lors de l'édition d'un document principal.

Il est possible de préfixer le lien pour changer le caractère sur la bouton (par défaut '+') et le texte surgissant affiché.

id	type	vis	link	...	eLink	...	Options
MA_RELATION	docid(“FACT”)	W			%S% app=GENERIC &action=GENERIC_EDIT &classid=FACT &id= %MA_RELATION%		eltitle=éditer une facture| elsymbol=$

L'exemple ci-dessus affichera le bouton '$' avec le texte surgissant « éditer une facture ». Le lien GENERIC_EDIT est l'action par défaut d'édition de document.

2.1.5. Paramètres

La définition d'un paramètre est identique à celle d'un attribut normal. Il suffit de mettre PARAM dans la première colonne au lien de ATTR.

//	idattr	idframe	label	T	A	type	ord
PARAM	WSGAT_FR_PAR		météo	N	N	frame	10
PARAM	WSGATE_URL	WSGAT_FR_PAR	url site météo	N	N	text	20

2.1.6. Modification d'une famille

2.1.6.1 Modifier un attributs père

Pour modifier un attribut hérité donc normalement non modifiable, il faut utiliser le mot-clet MODATTR en ayant le même identifiant que l'attribut à modifier.

Seul les parties remplies sont modifiés. Les parties non remplies indiquent que l'on conserve la valeur du père. Si on veut effacer la valeur du père il faut mettre le caractère '-' dans la cellule.

Voici un exemple avec une nouvelle famille basée sur la famille “de base”

Clef	ID	Parent	Label	Type
ATTR	TST_FR_TEST		Cadre	frame
MODATTR	BA_TITLE		Nouveau	

Dans ce cas, le libellé du champ “titre” ne sera remplacé par "Nouveau".

Si par contre vous utilisez la clef "ATTR" au lieu de MODATTR alors toutes les propriétés du père sont perdu. Ce correspond à un écrasement de ce dernier. Si les propriétés de l’attribut de l’ancêtre sont modifiés, l’attribut fils modifié n’est pas impacté.

2.1.6.2 Supprimer un attribut

On ne peut pas supprimer les attributs crées. Par contre on peut les rendre invisibles en affectant leur visibilité à H (caché) ou I (invisible). L'option "RESET attributes" permet néanmoins de supprimer l'ensemble des attributs avant de les reconstruire. Il ne faut pas utiliser cette option si votre fichier de déclaration ne comprend pas l'ensemble des attributs (entre BEGIN et END) au risque de perdre les attributs non déclarés dans cette section attributs.

2.1.6.3 Supprimer un attribut père

Si on ne veut pas hériter d’un des attributs des ancêtres, il faut :

- Modifier l'attribut en mettant sa visibilité à H invisible

2.1.7. Limitations sur la mise à jour de structure

Les types des attributs ne peuvent pas être changés après leur premier import. Si vous êtes en phase de

développement, vous pouvez supprimer la famille (wsh fdl_deletefamily) pour relancer l'importation. L'option "RESET attributes", que vous pouvez indiquer dans la déclaration de la famille, supprime les attributs avant de les reconstruire avec les nouvelles directives. Si vous mettez cette option, le changement de type sera effectué sur la classe mais cela peut induire des incohérences sur la base de données. Une erreur d'importation sera retournée si le type en base de données est en incohérence avec la déclaration de la famille.

Le passage d'un attribut à un paramètre ou inversement n'est pas autorisé. Comme pour les changements de type, il est possible de le forcer avec l'option "RESET attributes" lorsqu'on est en phase de développement.

Le changement d'héritage ne peut être fait sur une famille après un premier import. Il est nécessaire dans ce cas du supprimer la famille ainsi que ses documents ou de lancer un script de migration.

2.1.8. Modifier une famille depuis l'interface Web

Pour modifier une famille, il faut :

- Aller dans le module “Gestion documentaire”
- Sélectionner la recherche “les familles”
- Sélectionner la famille à modifier
- Faire un clic droit sur la famille et sélectionner le menu “Éditer les attributs”

2.1.8.1 Ajouter un attribut

L’ajout d’attribut se fait sans condition en utilisant les trois dernières lignes de l’interface “éditer les attributs”.

2.1.8.2 Paramétrage d'une famille depuis l'interface

2.1.8.2.1 Dossier racine d'une famille

Ce dossier sert de dossier racine dans l’application ONEFAM (Gestion par famille). Si une famille a un dossier par défaut, elle peut être gérée par l’application ONEFAM.

2.1.8.2.2 Icône de la famille

Pour changer l'icône d'une famille, il faut :

- Faire un clic droit sur la famille et sélectionner le menu “Changer d'icône”
- Cliquer sur “Parcourir” et sélectionner une image sur le disque dur
- Cliquer sur le bouton “Importer”

2.2 Gestion des utilisateurs

2.2.1. Présentation

Ce manuel s'adresse aux personnes qui ont en charge de gérer les comptes d'accès à l'applicatif **dynacase**.

L'accès aux applications hébergées par **dynacase** est contrôlé. L'utilisateur doit s'authentifier pour accéder à une application (ou à un service fourni par le serveur).

En fonction de son profil, il est autorisé ou non à utiliser ces applications ou certaines de leurs fonctions.

Afin de simplifier la gestion des utilisateurs et en particulier, la gestion de leur droits d'accès aux différentes applications, fonctions et services, l'administrateur défini des groupes. Un groupe rassemble un ensemble d'utilisateurs ayant des droits identiques pour la majorité d'entre eux au moins. La possibilité de modifier les droits d'un utilisateur (suppression ou ajout) par rapport au groupe reste possible.

L'interface principale se compose de deux parties.

Celle de gauche permet de rechercher des utilisateurs ou des groupes. La partie de droite sert à consulter les informations sur les utilisateurs ou des groupes de manière détaillée et sert aux interfaces d'éditions des groupes et des utilisateurs.

2.2.2. Gestion des groupes

2.2.2.1 Création de groupes

À l'initialisation de dynacase, deux groupes ont été crées : le groupe « utilisateurs » et le groupe « administrateurs ». Le premier de ces groupes est le groupe dans lequel sont affectés les nouveaux utilisateurs par défaut. Le deuxième groupe contient les utilisateurs qui ont les permissions d'administration informatique. Ces administrateurs ont le droit (par défaut) de créer de nouveaux groupes et de nouveaux utilisateurs. Ils peuvent aussi modifier n'importe quelles informations sur les utilisateurs (mot de passes, nom, prénom, ...).

Par défaut, pour qu'un utilisateur accède à l'application « gestion des utilisateurs », il doit appartenir au groupe administrateur. À l'initialisation de dynacase, un super administrateur (login: *admin*) a la responsabilité de créer un administrateur. Dans le cas contraire seul ce super administrateur pourra utiliser cette application.

Pour créer un nouveau groupe il faut cliquer sur l'onglet « **Groupes** » puis sur le bouton « **Créer un nouveau groupe** ».

Cela a pour effet d'afficher l'interface d'édition de groupes. Cette interface est composée de deux parties : une partie « **système** » et une partie « **Identification** ». Les informations qui doivent être obligatoirement renseignées sont toujours en gras. Ceci est valable quel que soit l'interface d'édition proposée par dynacase. Pour créer un groupe il faut saisir au moins les deux informations suivantes :

- **Nom** : dénomination du groupe. Souvent c'est sa fonction. Exemple : commerciaux, recherche & développement.
- **Login** : nom système. Il sert à l'identification du groupe. Ce nom ne doit contenir que des caractères minuscules alphabétique ou des chiffres ainsi que les caractères (tiret),_ (blanc souligné), . point. Il ne doit pas contenir d'espaces ni de caractères accentués. Ce nom doit être unique parmi l'ensemble des autres noms de groupes mais aussi des utilisateurs.
- **Sans calcul des membres**: Le tableaux récapitulant l'ensemble des membres ne sera pas produit. Il est intéressant de cocher cet attribut afin d'améliorer les performances lors des ajout d'utilisateur dans les groupes.
- **Sans adresse mail de groupe** : Ne produit pas l'attribut qui concatène l'ensemble des adresses mail. Cela est intéressant à cocher si le groupe contient beaucoup de membre.

Les autres renseignements sont facultatifs.

- **Rôle** : Tableau indiquant les rôles joués par les utilisateurs membres du groupes. Les rôles sont des éléments déterminants pour la mise en place des droits sur dynacase (voir §. Sécurités).
- **Fonction** : par défaut les groupes constitués sont des groupes fonctionnels. Ils servent à rassembler des utilisateurs ayant des fonctions similaires dans l'entreprise. Cet attribut sert à préciser le lien entre les membres. L'option « service » indique que cela est un groupe qui rassemble les membres d'un même service. Dans ce cas le nom du groupe est généralement le nom du service. L'option « bureau » indique que les membres sont localisés dans le même « bureau ». D'autres options peuvent être ajoutées.

Par défaut le nouveau groupe appartient au groupe « utilisateurs ». Si vous voulez le faire appartenir à un autre groupe (ou à aucun groupe), cliquer sur le menu « **Modifier hiérarchie** ». Cela affiche l'arborescence des groupes. Pour déplier entièrement l'arbre des groupes cliquez sur le premier bouton **+**.

Pour faire appartenir votre groupe à un autre groupe cliquez sur le nom du groupe choisi. Celui-ci s'affiche alors en vert. Les groupes affichés en vert indiquent les groupes dont votre groupe fait parti **directement**. Les groupes affichés en bleu indiquent les groupes dont votre groupe fait parti **indirectement,** c'est à dire qu'un des ses groupes parents fait parti des groupes en bleu. Pour déselectionner un groupe il suffit de re-cliquer dessus. La hiérarchie des groupes suit une logique arborescente. Il est par conséquent interdit de faire appartenir le groupe à un de ses fils ou à lui-même. Dans ce cas si vous sélectionnez un des fils ou lui-même, les groupes non autorisés s'afficheront en rouge. Il faut, dans ce cas, recliquer dessus pour enlever l'alerte.

Pour des raisons de maintenances, il est conseillé de ne pas sélectionner un groupe qui est déjà bleu. Cela n'a pour effet que de répéter le fait qu'il appartient à ce groupe. Au niveau des droits d'accès il n'est fait aucune différence entre un groupe indirect (bleu) et un groupe direct (vert).

Pour voir le descriptif détaillé pour un groupe, vous pouvez cliquer sur l'icone affichée à gauche du nom. Cela ouvre un petit cadre avec la description du groupe.

Lorsque les informations sont remplis, il faut valider votre ajout ou modification en appuyant sur le bouton **Créer** ou **Sauver** suivant que vous être en création ou en modification.

Vous obtenez alors le document du nouveau groupe.

2.2.2.2 Modification d'un groupe

Pour modifier un groupe il faut se placer dans l'onglet « **Groupes** » du cadre gauche. Ensuite vous cliquez sur l'icone placée à gauche du nom du groupe. Cela affiche la description complète du groupe.

Ensuite si vous cliquez sur la document descriptif du groupe avec le bouton droit de la souris un menu contextuel s'affiche. De manière générale, si le curseur de la souris devient une croix cela indique que vous avez accès à un menu contextuel. Ceci est valable pour toute consultation de document sous dynacase. Sélectionnez alors **Modifier** pour afficher l'interface d'édition. Vous pouvez aussi double-cliquer sur un onglet pour passer en mode édition.

L'interface de modification est identique à celle de création et fonctionne de la même façon. L'administrateur peut modifier le "login" après création. Dans ce cas ,l'ancien login n'est plus valide.

2.2.2.3 Consultation du groupe

Vous avez trois manières de voir les membres du groupes.

Dans l'onglet Groupes cliquez sur le nom du groupe. Cela fait apparaître l'ensemble des utilisateurs qui sont directement rattachés au groupe

Sur le document l'attribut « Personnes du groupes » indique les utilisateurs qui sont directement dans ce groupes.

Le tableaux « sous-groupes » contient la liste des groupes directement attachés à ce groupe. Le tableau « groupes parents » contient la liste des groupes dans lesquels le groupe fait partie directement. Si ces tableaux sont vides ils n'apparaissent pas dans la description. Afin d'accéder à la consultation de membre vous pouvez cliquer sur le nom afin d'afficher l'information détaillée. Si vous cliquez sur le nom de l'utilisateur avec le bouton droit de la souris une mini vue s'ouvre avec le détail du groupe ou de l'utilisateur correspondant. Ceci vous permet de voir un détail sans quitter la page.

La troisième manière de voir les membres du groupe est d'utiliser le menu « Ouvrir ». Cela ouvre une fenêtre qui contient la liste des groupes et des utilisateurs appartenant directement au groupe.

2.2.2.4 Suppression de groupe

Pour supprimer un groupe, il faut utiliser le menu contextuel du document descriptif du groupe. Dans ce menu contextuel, il faut sélectionner l'item « **supprimer** » afin de supprimer le groupe. Attention, cette opération est irréversible. Une confirmation de suppression vous est demandée afin de valider votre demande.

Cette opération a pour effet de détacher tous les membres du groupe de ce groupe. Cela ne supprime pas les membres du groupes (utilisateurs et sous-groupes).

Une fois l'opération achevée le document descriptif est affiché avec la mention « Ce document a été supprimé ».git

2.2.3. Gestion des utilisateurs

2.2.3.1 Création d'utilisateurs

La création d'utilisateur se fait en cliquant sur le bouton « **utilisateur** » présent dans l'onglet « **utilisateurs** ». Ceci vous ouvre l'interface de création d'utilisateur. Les informations obligatoires pour la création d'utilisateurs sont indiquées en gras et sont toutes sur l'onglet « système ». Le login d'un utilisateur ne doit contenir que des caractères alphabétiques ou numériques. Il ne doit pas contenir ni d'espaces ni de caractères accentués. De plus ce login doit être unique. Si vous saisissez un login incorrect un bouton rouge apparaîtra à côté du champ de saisie pour vous indiquer que le login saisi n'est pas valide. Le login ne peut être saisi qu'à la création. Vous ne pourrez pas modifier ce login après la création.

Le mot de passe doit être saisie deux fois afin d'éviter des erreurs de saisie (vérifiez bien que le clavier n'a pas le CAPS LOCK enclenché).

La tableaux "Rôles" permet d'indiquer quels sont les rôles joué par l'utilisateur. Ces rôles sont utilisés pour la définition des profils de document et pour la définition des accès aux applications. Le bouton "+" situé à droite du champ de saisie permet de créer un nouveau rôle si le rôle n'est pas déjà sélectionné.

2.2.3.2 Modification d'utilisateurs

L'interface de modification est affichée en cliquant sur le menu « Modifier »lorsque vous êtes sur le document descriptif de l'utilisateur. Les champs affichés sont les mêmes que ceux de la création. La seule contrainte est que vous ne pouvez pas modifier le login.

Affecter un utilisateur dans un groupe : Pour affecter un utilisateur dans un groupe, il faut cliquer sur le menu "Modification groupe" afin d'accéder à l'interface de changement du groupe. Il est aussi possible d'ajouter un utilisateur dans un groupe à partir du document "groupe". Dans cas depuis document "groupe", le menu “Gérer les membres” permet de rajouter ou d'enlever des utilisateurs dans les membres de ce groupe.

Permettre à un utilisateur de gérer les autres utilisateurs : le compte admin permet de gérer les utilisateurs. Malgré cela, si on veut donner le droit de gérer les utilisateurs et les groupes à un utilisateur particulier, il faut alors le mettre dans le groupe “Administrateur” puis de lui affecter l'ACL “FUSERS” de l'application “FUSERS”.

2.2.3.3 Consultation d'utilisateurs

La consultation des utilisateurs peut être fait soit par groupes (onglet « **groupes** ») soit de manière générale onglet « **utilisateurs** ». Vous pouvez rechercher un utilisateur soit par son nom et prénom soit par son login en saisissant dans les champs marqués le début ou une partie du critère recherché. Vous envoyez votre demande de recherche en appuyant sur la touche « entrée » du clavier. Ensuite il suffit de cliquer sur la rangée pour afficher le document complet décrivant l'utilisateur.

Dans l'onglet groupe, la hiérarchie des groupes vous est présentée. Si vous cliquez sur le nom d'un groupe vous retrouvez l'interface de consultation des utilisateurs mais limité au groupe. Si vous faites des recherches dans cette interface, elle portera uniquement sur les membres directs du groupe.

2.2.3.4 Suppression d'utilisateur

Pour supprimer un utilisateur, il faut utiliser le menu "**Supprimer**" du document descriptif de l'utilisateur. Attention, **cette opération est irréversible**. Une confirmation de suppression vous est demandée afin de valider votre demande.

Cette opération a pour effet de ne plus autoriser la connexion à cet utilisateur. Tous les documents (contacts, sociétés, etc.) que cet utilisateur a créé sont conservés.

Une fois l'opération achevée le document descriptif est affiché avec la mention « Ce document a été supprimé ».

2.2.3.5 Titulaires et suppléants

Un compte utilisateur peut désigner un suppléant. Le suppléant reçoit tous les privilèges de celui qui l'a désigné. Ce dernier est nommé le titulaire. Un suppléant peut voir, modifier tous les documents que le titulaire peut voir et modifier. Il peut aussi accéder à toutes les applications auxquelles le titulaire a accès. Par contre, le suppléant ne peut pas modifier le document « utilisateur » de ses titulaires. Ainsi le mot de passe du titulaire ne peut pas être changé par son suppléant. Un titulaire ne peut désigner qu'un seul suppléant. Par contre, un suppléant peut avoir plusieurs titulaires.

Le suppléant est ajouté aux destinataires des courriels qui ont été lancés à partir d'un modèle de mail[7] à destination du titulaire. Le texte « suppléant » est ajouté dans le nom des adresses email des suppléants afin de les différencier des titulaires.

Lors d'une création ou d'une modification d'un document par un suppléant ou lors d'un changement d'états, le texte de l'historique indique que cela est fait par un suppléant dans le cas où celui-ci n'a pas directement (c'est à dire sans qu'il soit suppléant) le droit de le faire.

Les droits sur les suppléants ne sont pas propagés. Si l'utilisateur C est suppléant de B et l'utilisateur B suppléant de A, l'utilisateur C n'a pas les privilèges de A et de B mais seulement les privilèges d'origine de B. B a les privilèges d'origine de A.

La modification d'un suppléant par l'interface est soumis au droit « ESUBSTITUTE » du contrôle de vue « confidentiel utilisateur» qui est associé à la famille « utilisateur ». Par défaut aucun utilisateur n'a la droit de désigner un suppléant.

2.2.4. Administration

Les fonctions d'administration concernent la mise à jours des utilisateurs et groupes ainsi que l'importation et la gestion des attributs « catégorie » des utilisateurs et des groupes.

2.2.4.1 Sécurité

Différents mécanismes permettent de sécuriser et contrôler les accès utilisateur :

7) *Généralement les modèles de mails sont utilisées dans les transitions des cycles de vies*

- la programmation d'une date d'expiration du compte
- le contrôle du nombre de connexions infructueuses
- l'activation/la désactivation du compte à la demande
- l'enregistrement des tentatives de connexion réussi ou non

Ces points et leurs mises en œuvre sont détaillés dans les paragraphes suivants.

2.2.4.1.1 Date d'expiration des comptes

Il est possible de programmer une date d'expiration du compte. Lorsque cette date est atteinte, toute connexion avec se compte est refusée. Un message spécifique indique à l'utilisateur que le compte à expiré et qu'il doit contacter le gestionnaire.

La programmation de cette date est faite en modifiant le compte utilisateur :

Dans l'onglet ***Système***, le cadre ***Sécurité*** contient les informations relatives à la sécurité du compte, en particulier la date d'expiration du compte. Vous pouvez préciser une date, modifier ou supprimer la date présente.

Le paramètre applicatif ***Délai par défaut avant expiration du compte (jours)*** de l'application ***AUTHENT*** permet de fixer la durée par défaut de validité d'un compte. Ce délai exprimé en jour, s'il est valorisé, est utilisé lors de la création de tout compte. La date d'expiration est calculée en ajoutant le nombre de jour paramétré à la date du jour de création.

La présence de cette date d'expiration rend le contrôle systématique. Pour inhiber ce contrôle, la date ne doit pas être renseignée.

Le contrôle est réalisé quelque soit le mode d'authentification (Dynacase, AD/LDAP, SSO)

Ce contrôle n'est pas réalisé pour le super administrateur Dynacase (admin).

2.2.4.1.2 Limitation des tentatives de connexion

Ce mécanisme permet d'interdire la connexion à un compte Dynacase après plusieurs tentatives de connexion avec un mot de passe incorrect.

Dynacase incrémente un compteur à chaque connexion en échec.

Si le nombre de connexion en échec est spécifié, Dynacase vérifie lors d'une connexion correctement authentifiée que ce nombre n'est pas atteint. Si c'est le cas, la connexion est refusée. Un message spécifique indique à l'utilisateur que le compte est bloqué après un trop grand nombre de tentative de connexions en échec. Sinon, le compteur de connexion est remis à 0.

Le paramètre applicatif ***Nombre d'échecs d'authentification avant la désactivation du compte*** de l'application ***AUTHENT*** permet de fixer le nombre maximum de connexion dont l'authentification échoue (mot de passe invalide) avant le blocage du compte. Lors de l'installation de l'application cette valeur est nulle (0). La valeur 0 indique que le contrôle n'est pas réalisée.

Le nombre de tentatives de connexion et le déblocage d'un compte sont disponible en consultant la fiche d'un utilisateur. À l'onglet ***Système***, le cadre ***Sécurité*** présente le nombre de connexion de cet utilisateur pour lesquelles l'authentification a échouée. À partir du menu ***Compte***, l'item ***Réinitialiser échecs de connexion*** permet de remettre à 0 ce nombre de connexion et ainsi autoriser à nouveau la connexion d'un utilisateur bloqué.

Le contrôle est réalisé quelque soit le mode d'authentification (Dynacase, AD/LDAP, SSO)

Ce contrôle n'est pas réalisé pour le super administrateur Dynacase (admin).

2.2.4.1.3 Activation / désactivation du compte à la demande

Il est possible d'activer ou de désactiver un compte utilisateur à la demande. La fiche utilisateur propose au menu ***Compte*** un item :

- ***Désactiver le compte*** lorsque le compte est activé (état par défaut lors de la création d'un compte)
- ***Activer le compte*** lorsque le compte est désactivé

L'activation / désactivation de compte n'influe pas sur les droits applicatifs ou documentaires liés aux comptes.

Le contrôle est réalisé quelque soit le mode d'authentification (Dynacase, AD/LDAP, SSO)

Ce contrôle n'est pas réalisé pour le super administrateur Dynacase (admin).

2.2.4.1.4 Journalisation

L'ensemble des tentatives de connexion utilisateur sont journalisées via le système ***syslog***. La ***facility*** par défaut est ***LOG_AUTH***. Habituellement les informations ainsi journalisées sont enregistrées dans le fichier */var/log/auth.log*. Pour plus d'information, reportez-vous à la documentation de syslog de votre serveur.

Le paramètre applicatif ***Session authentication log facility d***e l'application ***AUTHENT*** permet d'indiquer une facility différente.

Les messages relatifs aux connexions sont formatés de la manière suivante :

<partie syslog> **[*statut de la connexion*] [*message complémentaire au statut*] [*backend d'authentification*] [*adresse IP du poste client*] [*login*] [*user-agent*]**

Par exemple :

```
May 19 09:31:38 server [S] Dynacase:Session:Authentication: [192.168.251.105]  :
[success] [welcome] [freedom] [192.168.251.105] [joe] [Mozilla/5.0 (Windows NT
5.1) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.65 Safari/534.24]

May 19 09:34:18 server [S] Dynacase:Session:Authentication: [192.168.251.105]  :
[failure] [invalid credential] [freedom] [192.168.251.105] [] [Mozilla/5.0
(Windows NT 5.1) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.65
Safari/534.24]
```

Le statut de la connexion est égal à :

- **failure** en cas d'erreur de connexion ;
- **success** lors de la réussite de la connexion ;
- **close** lors de la fermeture de la connexion ;

Le message complémentaire en cas d'erreur est :

- **invalid credential** : ceci indique le la validation du mot de passe est refusée par le backend d'authentification ;
- **login have no Dynacase account** : se produit lorsque l'utilisateur est authentifié par un backend d'authentification externe, mais ne possèdent pas de compte Dynacase ;
- **inactive account** : le compte Dynacase est désactivé;
- **account has expired :** le compte Dynacase a expiré ;
- **max connection (#) attempts exceeded** : le nombre maximum de tentative de connexion non authentifiée par le backend est atteint ;
- **username should exists in session** : la session (cookie) est incorrecte et ne contient pas le login du compte connecté;

2.2.4.2 Mises à jour

Le bouton « **Actualiser les utilisateurs** » effectue une mise à jour des documents utilisateurs et groupe depuis les informations systèmes des comptes. Vous pouvez utiliser ce bouton dans le cas où des mises à jours ont eu lieu directement en base de données. Si vous utilisez uniquement l'interface dynacase pour gérer les utilisateurs, il n'est pas nécessaire d'effectuer cette opération.

“Actualiser” les utilisateurs veut dire de mettre à jours les informations systèmes des utilisateurs depuis la base de données `anakeen` vers la base `dynacase`. Ceci, dans un fonctionnement normal, n'est pas nécessaire. Cela est nécessaire quand :

- la base “anakeen” / table user / a été mise à jour directement à la main (par le requeteur postgreSql)
- lorsque l'on a détruit accidentellement des données dans la base “freedom” / table doc128 (utilisateur) et doc127 (groupes) /

La commande, lorsqu'il y a de nombreux utilisateurs est consommatrice en temps. En effet, l'appartenances aux groupes, les adresses email de groupes, les données LDAP sont recalculés et mises à jour.

La commande shell d'actualisation est : (à effectuer en tant que root sur la machine serveur d'applications)

```
[root@www ~]# wsh --api=usercard_iuser
```

Pour actualiser un seul utilisateur :

```
[root@www ~]# wsh --api=usercard_iuser --whatid=10
```

10 est l'identifiant système de l'utilisateur (indiqué sur la fiche dynacase) - c'est l'attribut id de la table user de la base anakeen.

Le bouton « **Actualiser LDAP** » ré-importe tous les utilisateurs dans l'annuaire LDAP. Ceci n'est valable que si le le paramètre « Activer LDAP » est activé³.

En ce qui concerne les groupes, une partie de leur attributs n'est pas actualisée directement a cause du temps de calcul qui peut être important lorsque les membres du groupes ou des groupes parents sont nombreux. Cela concerne les attributs (membres et mail du groupes°. Lorsque vous créez un nouvel utilisateur ou un nouveau groupe, le bouton « **Tous les groupes sont actualisés** » est remplacé par le bouton « **x groupe(s) à actualiser** ». Cela indique que certain groupes n'ont pas leur liste de membre à jours. Cela n'a aucune incidence sur les droits des utilisateurs. Vous pouvez alors déclencher le rafraîchissement du groupes en appuyant sur ce bouton. Ceci n'est pas obligatoire, FREEDOM vérifie s'il y a des groupes à actualiser toutes les heures et les mets à jours si nécessaire.

2.2.4.3 Importation

L'importation de groupes et d'utilisateurs est effectuée à l'aide d'un fichier d'importation CSV (texte séparé par des point-virgules).

Les quatre premières colonnes” sont des colonnes « système ». Elles permettent de savoir quel type d'importation doit être fait.”

2.2.4.3.1 Importation d'utilisateurs

Pour importer des utilisateurs, les paramètres suivants doivent être renseignés.

DOC	IUSER	0	0

N°	Signification	Valeurs
1	Indique l'action à faire. Mettre toujours “DOC”' pour indiquer l'importation.	DOC
2	Identifiant de la famille à importer. Pour des utilisateurs mettre “IUSER”'.	IUSER
3	Identifiant spécifique pour le document descriptif correspondant à l'importation. Laisser à zéro si c'est un nouvel utilisateur. Si le numéro est celui d'un document descriptif déjà existant, celui-ci sera modifié.	
4	Identifiant du groupe d'appartenance de l'utilisateur. 0 indique qu'il appartiendra par défaut au groupe “utilisateurs”'.	

Les colonnes suivantes définissent les caractéristiques de l'utilisateur. Par défaut les colonnes suivantes concernent tous les attributs modifiables. Par contre certaines colonnes servent à stocker des attributs « système », ils ne doivent être remplis qu'en connaissance de cause.

Ci-dessous, voici les attributs par défaut qui peuvent être renseigné. En vert, ce sont les attributs obligatoirement présent. En bleu, ce sont les attributs optionnels et en rouge ce sont des attributs « système » qui ne doivent pas être renseignés (ils sont calculés par le système). Les attributs oranges sont des attributs qui ne doivent être modifiés que par des administrateurs expérimentés dans dynacase. Leurs définitions et leurs impacts ne sont pas décrite dans ce

manuel.

us_civility	**civilité**
us_lname	**nom**
us_fname	**prénom**
us_photo	photographie
us_initials	initiales
us_pphone	téléphone direct
us_intphone	n°poste
us_pfax	fax direct
us_mobile	mobile
us_meid	utilisateur id
us_login	**login**
us_passwd1	**nouveau mot de passe**
us_passwd2	**confirmation mot de passe**
us_idgroup	id groupe
us_groupe	groupe
us_status	état
us_daydelay	délai d'expiration en jours
us_expiresd	date d'expiration
us_expirest	heure d'expiration
us_iddomain	id domain
us_domain	domaine
us_extmail	mail externe
us_idsociety	id société
us_socaddr	même adresse ?
us_society	site société
us_type	fonction
us_idservice	service id
us_service	service
us_job	métier
us_role	rôle
us_workaddr	adresse
us_workpostalcode	code postal
us_worktown	ville
us_workcedex	cedex
us_country	pays
us_workweb	web
us_scatg	catégorie
us_wgcal_gid	Id

Pour choisir les colonnes à importer, vous pouvez utiliser l'interface suivante. La sélection de gauche concerne les attributs que vous voulez renseigner. La partie de droite concerne les attributs que vous ne voulez pas renseigner.

Vous pouvez soit supprimer des colonnes, soit modifier leur ordre. Pour supprimer il faut sélectionner la (les) colonne(s) et les déplacer dans le cadre de droite (bouton →). Pour changer l'ordre des colonnes, il faut utiliser les boutons |^ et ^| . L'ordre et la définition des colonnes utilisées seront celui du cadre de gauche. Lorsque vous modifiez cet ordre, l'écriture des colonnes en bas est aussitôt modifiée.

Vous pouvez utiliser ces deux lignes inscrites dans le cadre pour votre fichier CSV afin d'indiquer l'ordre de vos colonnes. La première ligne 'ORDER' indique au logiciel le nouvel ordre des attributs d'importation établi pour cette famille. S'il n'y a pas de ligne ORDER dans le fichier, l'ordre sera celui défini dans le cadre gauche. Dans le cas contraire ce sera l'ordre défini dans le fichier qui sera pris en compte. Les quatre premières colonnes d'une ligne ORDER sont composées du mot-clef 'ORDER', de l'identificateur de la famille puis de deux colonnes non utilisées (servant seulement pour l'alignement avec les lignes DOC). Les colonnes suivantes contiennent les identifiants des attributs de la famille. La deuxième ligne inscrite dans le cadre du bas ('#DOC') sert juste à aider l'utilisateur sur la signification des colonnes.

ORDER	IUSER	0	0	us_lname	us_fname	us_login	us_passwd1	us_passwd2	us_extmail
#DOC	IUSER	0	0	nom	prénom	login	nouveau mot de passe	confirmation mot de pa	mail externe
DOC	IUSER	0	0	Deschamps	Arthur	arthur.deschamps	mdparthur	mdparthur	arthur.deschamps@zoo.net
DOC	IUSER	0	0	Trompette	Gilberte	gtrompette	mdpgilberte	mdpgilberte	gtrompette@zoo.net

On peut avoir un fichier minimaliste comme celui-ci:

Le fichier CSV produit par le tableur doit être comme ci-dessous. Il ne doit pas avoir de retour à la ligne entre 2 lignes d'importation. Le fichier CSV ne doit pas contenir de délimiteurs de texte et le séparateur de champ doit être le point-virgule.

```
ORDER;IUSER;0;0;us_lname;us_fname;us_login;us_passwd1; us_passwd2; us_extmail
#DOC;IUSER;0;121; nom; prénom; login; nouveau mot de passe; confirmation mot de
passe; mail externe
DOC;IUSER;0;0;Deschamps;Arthur;arthur.deschamps;Anakeen;Anakeen;arthur.deschamps
@zoo.net
DOC;IUSER;0;0;Trompette;Gilberte;gtrompette;Anakeen;Anakeen;gtrompette@zoo.net
```

Le cadre **dossiers** indique tous les groupes d'utilisateurs dans lesquels les utilisateurs peuvent appartenir. Si vous cochez un ou plusieurs groupes les utilisateurs importés seront rattachés aux groupes sélectionnés. Pour des raisons de simplifications, il est conseillé d'importer les utilisateurs par groupes. C'est à dire, qu'il faut produire un fichier CSV par groupe.

Le cadre 'Politique' permet de choisir la politique d'importation lorsqu'un utilisateur à importer possède des similitudes avec un utilisateur existant (par défaut même nom et prénom). La similitude est détectée par défaut si le titre du document décrivant l'utilisateur possède le

même titre qu'un autre document de la même famille.

Dans le cadre d'importation d'utilisateurs, il n'est pas souhaitable de modifier la politique par défaut. Si vous avez des homonymes, vous pouvez utiliser le login comme clef d'import car il est une clef unique et on ne peut avoir deux utilisateurs avec le même login.

La cadre '**valeurs par défaut** permet d'affecter des valeurs à toutes valeurs nulles (vides) définies dans le fichier d'importation. Ces valeurs par défaut sont uniquement utilisées pour des nouveaux documents, pas pour les mise à jours.

Dans le cas d'une modification si une valeur est vide cela signifiera que la précédente valeur sera inchangée. Pour supprimer une valeur il faut mettre le caractère 'espace'. Ceci indique explicitement la suppression de la valeur.

La case **Analyse seulement** est cochée par défaut. Si vous appuyez sur '**Importation CSV**' alors l'analyse du fichier sera effectuée et le résultat sera affiché dans le cadre du bas.

Résultat de l'analyse de l'import

2 documents à prendre en compte.

ligne	titre	dossier	id	famille	action	message	modifications	erreur
1						nouvel ordre des colonnes us_lname - us_fname - us_login - us_passwd1 - us_passwd2 - us_extmail		
3	Deschamps Arthur	/carnet d'adresses			ajouté	Deschamps Arthur et ajout dans dossier carnet d'adresses	• [nom:Deschamps] • [prénom:Arthur] • [login:arthur.deschamps] • [nouveau mot de passe:Anakeen] • [confirmation mot de passe:Anakeen] • [mail externe:arthur.deschamps@zoo.net]	
4	Trompette Gilberte	/carnet d'adresses			ajouté	Trompette Gilberte et ajout dans dossier carnet d'adresses	• [nom:Trompette] • [prénom:Gilberte] • [login:gtrompette] • [nouveau mot de passe:Anakeen] • [confirmation mot de passe:Anakeen] • [mail externe:gtrompette@zoo.net]	

Les ajouts sont notifiés en vert, les modifications en jaune et les document ignorés en rouge.

Le résultat de l'analyse indique le nombre de lignes à prendre en compte (ajouté ou modifié). Il indique pour chaque ligne du fichier, l'interprétation effectuée et l'action qu'il entreprendra. Si l'analyse est conforme aux attendus vous pouvez décocher 'analyse seulement' est lancer réellement l'importation. Le résultat apparaîtra dans le cadre du bas à la place de l'analyse.

2.2.4.3.2 Importation de groupes

L'interface d'importation des groupes est identiques à celle des utilisateurs. Pour indiquer qu'il s'agit d'une importation de groupe il faut indiquer IGROUP dans la deuxième colonne au lieu de IUSER (réservé pour les utilisateurs).

DOC	IGROUP	0	0

La liste des attributs nécessaires pour la création du groupe est la suivante.

- **grp_name** : libellé du groupe
- **us_login** : référence du groupe

L'exemple suivant montre le fichier de création minimaliste pour la création de groupes

ORDER	IGROUP	0	0	grp_name	us_login
#DOC	IGROUP	0	0	nom	login
DOC	IGROUP	0	0	Recherche & Développement	gdevel
DOC	IGROUP	0	0	Communication	gcomm

Le fichier CSV produit par le tableur aura la forme suivante:

```
ORDER;IGROUP;0;0; grp_name; us_login;;;;
#DOC;IGROUP;0;0; nom; login;;;;
DOC;IGROUP;0;0;Recherche &amp; Développement;gdevel;;;;
DOC;IGROUP;0;0;Communication;gcomm;;;;
```

Comme pour les utilisateurs vous pouvez renseigner les valeurs par défaut et les groupes d'appartenances (onglet **dossiers**). Le résultat de l'importation donnera le résultat suivant.

2 documents à prendre en compte.

ligne	titre	dossier	id	famille	action	message	modifications	erreur
1						nouvel ordre des colonnes grp_name - us_login - - - -		
3	Recherche & Développement	/carnet d'adresses			ajouté	Recherche & Développement et ajout dans dossier carnet d'adresses	• [nom:Recherche & Développement] • [login:gdevel]	
4	Communication	/carnet d'adresses			ajouté	Communication et ajout dans dossier carnet d'adresses	• [nom:Communication] • [login:gcomm]	

L'accès d'un groupe d'utilisateurs à une ACL peut être indiqué lors de l'importation des groupes utilisateurs. Pour cela, il faut ajouter une ligne dans le tableur d'import selon :

mot clé	nom_du_groupe	nom de l'appli	nom des acl	...
ACCESS	Group_A	APPLI LAMBDA	ACL1_APPLI_LAMBDA	ACL2_APPLI_LAMBDA
ACCESS	Group_B	APPLI ALPHA	ACL_APPLI_ALPHA	

2.2.4.3.3 Affectation de groupes

L'affectation des utilisateurs (ou des sous-groupes) à des groupes peut aussi être effectuée à l'aide du fichier CVS d'importation. Il faut pour cela nommer les groupes dans le fichier CSV puis ensuite référencer les utilisateurs (ou sous-groupes) au groupe voulus.

Pour affecter un nom à un groupe, il faut renseigner la colonne n°3 avec une clef. Cette clef est une chaîne de caractères sans espace et ne contenant pas de caractère accentué. Cette clef doit être unique pour tous les utilisateurs, groupes et autre documents nécessaires à dynacase. Pour cela il est fortement conseillé de préfixer la clef par “GRP_” pour les groupes et par “USER_” pour les utilisateurs.

Si vous n'avez pas nommé le groupe à la création, vous pouvez le faire a posteriori en reprenant les informations composant le titre du document décrivant le groupe. Dans le cas du groupe, il faut au moins re-renseigner le nom (“grp_name”) pour que dynacase puisse faire l'association avec un groupe existant (ceci est valable pour la politique d'importation par défaut puisque la clef est le titre). Pour affecter un utilisateur (ou un sous-groupe) à un groupe il faut mettre la clef du groupe dans la colonne n°4.

ORDER	IGROUP		0	0	grp_name	us_login
#DOC	IGROUP	id	fldid		nom	prénom
DOC	IGROUP	**GRP_DEVEL**		0	Recherche & Développement	gdevel
DOC	IGROUP	**GRP_COMM**	**GRP_DEVEL**		Communication	gcomm
ORDER	IUSER		0	0	us_lname	us_fname
#DOC	IUSER	id	fldid		nom	prénom
DOC	IUSER		0	**GRP_COMM**	Deschamps	Arthur
DOC	IUSER		0	**GRP_COMM**	Trompette	Gilberte
DOC	IUSER		0	**GRP_DEVEL**	Deschamps	Arthur

Dans l'exemple ci-dessus, on affecte la clef GRP_DEVEL et GRP_COMM aux groupes “gdevel” et “gcomm”. Le groupe “gcomm” devient un sous groupe de “gdevel”. L'utilisateur « arthur deschamps » appartient aux groupes “gcomm” et “gdevel” et l'utilisatrice « gilberte trompette » appartient au groupe “gcomm”.

Il n'est pas possible par l'importation CSV d'enlever des utilisateurs (ou des sous-groupes) à un groupe.

2.2.4.3.4 Importation de rôles

Les rôles sont importables comme tout document.

Les attributs à renseigner sont :

- role_login : référence du rôle
- role_name : libellé du rôle

Exemple de création de rôles suivis d'affectation de ces rôles sur des groupes et des utilisateurs.

//FAM	Role(ROLE)	<specid>	<fldid>	Référence	Libellé
ORDER	ROLE			role_login	role_name
DOC	ROLE	TST_ROLERED		rred	Rouge
DOC	ROLE	TST_ROLEBLUE		rblue	Bleu
DOC	ROLE	TST_ROLEYELLOW		ryellow	Jaune

//FAM	groupe intranet	<specid>	<fldid>	login	
ORDER	IGROUP			us_login	grp_roles
DOC	IGROUP	TST_GRPRED	-	gred	TST_ROLERED
DOC	IGROUP	TST_GRPGREEN		ggreen	TST_ROLEBLUE\nTST_ROLEYELLOW
DOC	IGROUP	TST_GRPBLUE	TST_GRPRED	gblue	TST_ROLEBLUE
DOC	IGROUP	TST_GRPYELLOW	TST_GRPRED	gyellow	TST_ROLEYELLOW
//FAM	utilisateur(IUS	<specid>	<fldid>	login	rôles
ORDER	IUSER			us_login	us_roles
DOC	IUSER	TST_USERJOHN	TST_GRPBLUE	ublue	TST_ROLEBLUE
DOC	IUSER	TST_USERJANE	TST_GRPGREEN	ugreen	TST_ROLEBLUE\nTST_ROLEYELLOW
DOC	IUSER	TST_USERRGREEN		urgreen	TST_ROLEBLUE\nTST_ROLEYELLOW
DOC	IUSER	TST_USERYELLOW		uryellow	TST_ROLEYELLOW
DOC	IUSER	TST_USERGGREEN	TST_GRPGREEN	uggreen	

L'attribut "us_roles" pour les utilisateurs et l'attribut "grp_roles" pour les groupes permettent d'affecter les rôles aux utilisateurs et aux groupes.

2.2.4.4 Catégories

Les éléments des attributs à choix multiples peut être modifiés en cliquant sur le bouton « **Modifier** » de l'attribut choisi. Cela ouvre l'interface de modification des catégories.

Pour ajouter un élément, il faut renseigner deux champs : la référence et la désignation. La référence est la clef qui sera stockée dans la base de données. La désignation est la signification de la référence. La référence ne doit contenir que des caractères alphabétiques ou numérique (pas d'espaces ni de caractères accentués ni de ponctuations). Une fois la référence et la désignation saisies , cliquer sur le bouton pour ajouter le nouveau choix. Pour supprimer un choix, il faut d'abord sélectionner un élément en cliquant sur la référence puis sur le bouton pour supprimer le choix. Vous pouvez modifier une désignation mais vous ne devez pas modifier une clef. En effet, si vous modifiez une désignation cela aura pour incidence de changer le nom du choix dans les différents documents. Le fait de supprimer une clef implique que les documents qui ont cette clef auront un choix qui n'est plus disponible. Si un document à un choix qui n'est plus disponible, ce sera le nom de la clef qui sera affiché. Pour modifier une désignation il faut d'abord sélectionner la référence. Cela affiche la sélection dans les champs du bas. Ensuite modifiez la désignation puis appuyez sur pour valider le changement.

Ces choix peuvent suivre une hiérarchie. Pour changer un choix de niveau, il suffit de le sélectionner et de cliquer sur

les boutons et pour augmenter ou diminuer le niveau.

Une fois vos modifications effectuées, appuyez sur le bouton pour enregistrer les modifications. Tant que vous n'avait pas enregistré les modifications vous pouvez quitter la page sans sauver.

2.2.4.5 Configuration LDAP

Les documents décrivant les personnes et les utilisateurs peuvent être copiés dans un serveur LDAP. Ce serveur LDAP permet notamment aux clients de messagerie d'être reliés au carnet d'adresses des utilisateurs et des personnes.

2.2.4.5.1 Installation et paramétrage LDAP

Le serveur LDAP est accessible en lecture seulement. Seul l'administrateur du LDAP peut écrire dans cette base. Pour ce connecter au serveur LDAP il faut définir les 5 paramètres suivant:

- nom du serveur LDAP : nom ou adresse IP de la machine où le serveur LDAP est installé.
- n° du port : normalement 389 (n° d'écoute pour le serveur)
- racine du serveur : indique l'arborescence où les données seront placées
- administrateur LDAP : login (LDAP DN) de l'administrateur LDAP
- mot de passe LDAP : mot de passe en clair de l'administrateur

La modification de ces paramètres est accessible en cliquant sur votre nom (en haut à droite) et en choisissant l'onglet « paramètres applicatifs ». Ces 5 paramètres doivent se retrouver dans la configuration système du serveur LDAP.

Pour activer la copie avec le serveur LDAP il faut ensuite mettre le paramètre « Activation LDAP » à “yes” dans le paramétrage applicatif du carnet d'adresses. Dans le cas contraire la copie n'est pas effectuée.

On utilisera comme serveur LDAP le logiciel openldap.

Dans le répertoire ”/etc/openldap” se trouve un fichier de configuration fourni par dynacase : ”/etc/openldap/slapd_anakeen.conf”. Ce fichier reprend la paramétrage par défaut que l'on a sur l'interface web de dynacase.

```
#view only its own organizationalUnit
access to dn.regex="^ou=([^,]*),dc=people,(.*)$"
by dn.regex="uid=$1,dc=users,$2" read
by group.expand="cn=$1,dc=people,$2" read
by * none
#view only its private card
access to dn.regex="^uid.*,ou=([^,]*),dc=people,(.*)$"
by dn.regex="uid=$1,dc=users,$2" read
by group.expand="cn=$1,dc=people,$2" read
by * none
access to attr=userPassword
by anonymous auth
by self read
by * none
#others
access to *
by dn="cn=Manager,o=Anakeen,c=fr" write
by anonymous read
by * read
```

```
database bdb

suffix "o=Anakeen,c=fr"

rootdn "cn=Manager,o=Anakeen,c=fr"

rootpw secret

...
```

Le fichier de configuration de openldap est "/etc/openldap/slapd.conf". Ce fichier doit inclure le fichier "slapd_anakeen.conf" soit par copie soit par "include". Le fichier de configuration principal doit contenir au moins les quatre schémas suivants ainsi que le chargement du module bdb (berkeley database).

```
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema
```

Lorsque le fichier de configuration est correct il suffit de démarrer le serveur LDAP:

```
[root@chewbacca ~]# /etc/rc.d/init.d/ldap start
Checking configuration files for slapd: config file testing succeeded **[ OK ]**
Démarrage de slapd :**[ OK ]**
```

Ensuite il faut initialiser la base LDAP. Pour cela on activera se script "usercard_ldapinit" comme le montre l'exemple suivant.

```
[root@chewbacca ~]#
[root@chewbacca ~]# wsh --api=usercard_ldapinit
4)anonymous guest: **updated**
3)Master Default: **updated**
2)Administrateurs: **updated**
1)Utilisateurs: **updated**
```

Cette dernière opération peut être faite aussi via l'interface (cf §4.1).

Ensuite les documents personnes ou utilisateurs seront copiés sur le serveur LDAP lorsqu'ils seront modifiés ou créés par l'interface dynacase.

Note

Dans le cas ou la racine de la base LDAP ne serait pas sous la forme "`o=xxx,c=xxx`", il faudra manuellement créer au préalable la base correspondante en important la définition LDIF de la racine organisation. Exemple pour une racine `dc=anakeen,dc=fr` :

```
# vi organisation-anakeen.ldif
dn: dc=anakeen,dc=fr
objectClass: dcobject
objectClass: organization
o: Anakeen
dc: Anakeen
# ldapadd -x -D 'cn=Manager,dc=anakeen,dc=fr' -W -f ./organisation-anakeen.ldif
```

2.2.4.5.2 Organisation du LDAP

L'annuaire LDAP est composé de deux branches principales : les utilisateurs ("dc=users") et les personnes du carnet

d'adresses (“dc=people”). Les coordonnées des utilisateurs et des personnes à accès publiques sont disponibles sans authentification. Les personnes déclarés avec un accès restreint ne sont accessibles qu'avec une authentification appropriée.

2.2.4.5.3 Configuration des clients LDAP

Ce paragraphe décrit comment paramétrer l'accès au serveur LDAP pour un client de messagerie.

Le client de messagerie choisi pour l'exemple est le logiciel thunderbird, mais le principe est le même pour Microsoft Outlook.

Sur thunderbird, la création d'un accès LDAP se fait via le carnet d'adresses. Une fois le carnet d'adresse ouvert, aller dans le menu Fichier/Nouveau/Annuaire LDAP. Ceci ouvre une fenêtre de dialogue afin de rentrer les coordonnées du serveur LDAP.

Pour ce connecter au serveur il faut au moins renseigner le nom, le nom d'hôte, et le chemin d'accès à la base LDAP.

2.2.4.5.3.1 Paramétrage annuaire public

Ce type de paramétrage ne demande pas d'authentification. Cela permet à toute personne d'accès aux informations des personnes et des utilisateurs.

Il suffit d'indiquer le nom de votre machine qui héberge le serveur LDAP et la racine de l'annuaire : “o=Anakeen,c=fr “est la valeur par défaut.

2.2.4.5.3.2 Paramétrage Annuaire public et privé

Pour accéder en plus à vos contacts privés il est nécessaire d'indiquer un login d'authentification LDAP. Ce login est constitué de la manière suivante :

“uid=<login>,dc=users,o=Anakeen,c=fr.”

où ”<login>” désigne le nom de connexion de l'utilisateur. Ce login LDAP est indiqué dans tous les documents décrivant les utilisateurs.

2.2.4.5.3.3 Paramétrage Annuaire privé seul

Le paramétrage est le même que le précédent au détail près que vous devez restreindre votre recherche à votre branche privée :

“ou=<login>,dc=people,o=Anakeen,c=fr”

1Pour les souris avec un seul bouton, vous pouvez utiliser la combinaison shift-click à la place.

2La touche « menu » de certains claviers fonctionne aussi si le focus est sur la fenêtre. Dans ce cas il faut ensuite avec les touches de tabulation se positionner sur l'item puis appuyer sur « entrée ».

3Le paramétrage LDAP client est expliqué dans le manuel du carnet d'adresses.

2.2.5. Connexions anonymes

dynacase supporte la connexion et la consultation en anonyme, ne nécessitant donc pas de s'authentifier et d'avoir un compte sur le système.

L'accès au mode anonyme se fait en accédant à la page “guest.php” (à la place du “index.php” par défaut) :

```
http://MON_SERVEUR/dynacase/guest.php
```

La connexion à cette page s'effectue donc automatiquement sous le compte “Anonymous Guest”.

La gestion de ce qu'il est possible de faire en mode anonyme, s'effectue de manière classique dans l'interface “Accessibilité” en modifiant les droits de cet utilisateur “Anonymous Guest” sur les différentes applications installées.

2.3 *Sécurité*

2.3.1. Les droits applicatifs

Les droits applicatifs doivent être enregistrés avec l'application ACCESS (Gestion des accessibilités). Ils permettent de définir des droits sur les actions applicatives. Ils ne s'appliquent pas aux documents mais seulement aux actions dans leur ensemble.

2.3.1.1 FDL - Bibliothèque dynacase

Nom	Définition
NORMAL	Autorise l'envoi des documents par mail. Permet l'aide à la saisie. À activer si GENERIC ou dynacase et activé.
EXPORT	Autorise l'exportation de dossiers ou de recherches. Doit être utilisé en corrélation avec FREEDOM_MASTER.
FAMILY	Autorise la modification des paramètres pour les documents famille. Doit être utilisé en correlation avec FREEDOM_MASTER.

2.3.1.2 GENERIC - Manipulation d'une famille

Nom	Définition
GENERIC_READ	Accès aux documents en lecture seule. Aucune fonction de modification n'est autorisée.
GENERIC	Accès à la modifications des documents.
GENERIC_MASTER	Accès à l'ajout de catégories et à l'import de document

2.3.1.3 ONEFAM - Manipulation groupement de familles

Nom	Définition
ONEFAM_READ	Accès à la liste de famille défini par l'administrateur en lecture seule
ONEFAM	Accès à une liste personnelle modifiable par l'utilisateur
GENERIC_MASTER	Accès en écriture à la liste administrative

2.3.1.4 FREEDOM - Doc Admin

Nom	Définition
FREEDOM_READ	Accès aux documents en lecture seule. Aucune fonction de modification n'est autorisée.
FREEDOM	Accès à la modifications des documents et des plans de classements
FREEDOM_MASTER	Accès aux fonctions d'import/export. Accès à la définition des attributs de familles
FREEDOM_GED	Accès à l'interface principale de Gestion documentaire

2.3.2. Profils de documents

Afin de protéger les informations vis-à-vis d'autrui, chaque document peut posséder un profil. Le profil définit des droits sur un document pour chacun des utilisateurs de la base documentaire. L'administrateur a en charge de définir les profils de création associés aux familles.

2.3.2.1 Définition des droits

Quatre familles de profils sont définies :

- profil de famille (droit de créer des documents ou de voir la famille)
- profil de document (droit de voir, modifier ou supprimer des documents)
- profil de dossier (droit de voir le contenu du dossier)
- profil de recherche (droit d'exécuter la recherche)

Les droits définis sont :

Nom interne	Description	Description longue	Pour
view	voir	Voir les caractéristiques du document, du dossier ou de la recherche. Le fait de ne pas voir un dossier n'implique pas de ne pas voir les documents contenus dans le dossier.	DFR
edit	éditer	Modifier les caractéristiques du document, du dossier. Les recherches ne sont pas modifiables.	DF
delete	supprimer	Supprimer le document, le dossier, la recherche; c'est à dire le mettre à la poubelle.	DFR
send	envoyer	Envoyer par email le document.	D
unlock	déverrouiller	déverrouiller le document.	DFR
viewacl	voir les droits	Voir les droits du document.	DFR
modifyacl	modifier les droits	Modifier les droits du document.	DFR
open	ouvrir	Ouvrir le dossier. Permet de voir le contenu du dossier	F
modify	modifier	Modifier le contenu du dossier. Permet d'ajouter ou de supprimer des documents dans le dossier.	F
execute	executer	Permet d'exécuter la recherche.	R
create	créer	Autorise la création de document de cette famille.	C
icreate	créer manuellement	Autorise la création de document de cette famille à partir de l'interface. Si ce droit n'est	C

		pas mis est que *create* est mis, l'utilisateur ne pourra créer le document que de manière indirecte (soit sur une transition, soit sur toutes autres actions particulière mis en place par l'administrateur). Sans ce droit les menus de création de cette famille sont inaccessibles. Si ce droit est mis il faut que le droit 'create' soit aussi mis.	
confidential	voir document confidentiel	Permet d'utiliser normalement un document qui est confidentiel. (Confidentiel est une propriété de document).	DFR
forum	poster sur un forum	Autorise le post sur un forum de document (utilisateur n'ayant pas le droit d'édition).	DF

D pour document, F pour dossier (folder), R pour recherche et C pour Famille (classe)

Une famille de profil définit un ensemble de droits. La plupart des profils de documents appartiennent à la famille profil de document, mais il est possible de définir une famille avec des droits supplémentaires. Au moins deux droits sont communs aux familles de profil : le droit de voir les droits et celui de modifier les droits.

Le profil d'un document est affiché dans les propriétés du document : profil d'accès. Si un document n'a pas de profil associé, tous les droits sont attribués à tout utilisateur.

Pour changer le profil d'un document, il faut sélectionner l'item "changer de profil" dans le menu contextuel disponible lorsque vous affichez un document. Cet item est visible seulement si l'utilisateur a le droit "modifier les droits" sur le document. L'utilisateur peut alors choisir tous les profils compatibles avec le document : profil de dossier pour les dossiers, profil de recherche pour les recherches et profil de document pour la plupart des autres documents.

Le but des profils est d'être utilisé par un ensemble de documents afin que lorsqu'on modifie les droits d'un profil, la modification soit répercutée sur l'ensemble des documents ayant ce profil.

Lors du changement de profil, l'utilisateur peut choisir "contrôle dédié". Ceci implique que le document a un profil personnel qui ne peut pas être réutilisé. Lors de l'activation d'un contrôle dédié, l'utilisateur courant a tous les droits et les autres aucun droit. L'utilisateur peut a posteriori modifier ce profil personnel comme n'importe quel autre profil.

Plutôt que d'utiliser des contrôles dédiés, il est conseillé de créer un profil de document et d'appliquer ce profil à tout document que l'on veut "privatiser". Ceci permet plus facilement de répertorier les documents personnels. L'utilisation abusive des contrôles dédiés aboutira à ne plus savoir qui a le droit de faire quoi sur la base puisque chaque document pourra avoir des droits différents sans cohérence.

2.3.2.2 Création de profil

La création d'un nouveau profil se fait en cliquant sur la barre de menu "Création/Profil" de l'application "Doc admin". Vous pouvez alors choisir parmi les quatre familles de profil présentes puis choisir un titre. Lorsque vous avez appuyé sur "Créer", un nouveau profil est né, mais il est désactivé. Cela veut dire qu'aucun droit n'est défini pour ce profil et que tous les documents qui se lieraient avec ce profil ne sont pas contrôlés .

Le profil est inactif si dans ses propriétés il est marqué "pas de contrôle d'accès". Il est actif s'il est marqué "contrôle dédié". Pour rendre actif un profil, il faut choisir l'item "Activer" dans le menu du document profil. Le menu désactiver permet de rendre le profil inopérant.

Si le profil est activé, un nouvel item "accessibilités..." apparaît dans le menu du document profil. Comme pour le contrôle dédié du document, l'activation du profil donne tous les droits à l'activateur. Si c'est une réactivation, les précédents droits, autres que celui de l'utilisateur sont conservés.

Pour modifier la définition de profil, il faut sélectionner l'item "accessibilités..." dans le menu du document profil. Cet item ne sera visible que s'il y a un profil lié au document et si vous avez le droit "voir les droits" sur le document.

La fenêtre de visualisation des droits présente tous les droits pour les rôles (présenté avec une puce rose) et les groupes d'utilisateur (puce bleue) et les droits pour l'utilisateur courant (puce verte) . Les groupes sont représentés de manière hiérarchique.

Trois sortes de cercles symbolisent les droits. Le cercle vert indique que le droit est accordé. Le cercle gris indique que le droit est accordé car l'un des rôles qui a le droit lui est affecté ou un groupes d'appartenance de la personne a ce droit. Le droit est hérité du ou des groupes pères.

Pour modifier un droit pour un groupe ou un utilisateur, il suffit de cliquer sur le bouton carré (à gauche de chaque nom). Des cases à cocher sont alors présentées. Il suffit de cocher pour autoriser et de décocher pour désactiver. Pour prendre en compte les nouveaux droits, il faut alors cliquer sur le bouton "Valider". Si vous avez fait des modifications

pour un rôle ou un groupe les répercussions sur les membres du groupe seront calculées automatiquement.

Pour voir les membres d'un groupe, il faut cliquer sur la flèche située à droite du nom du groupe.

2.3.2.3 Spécification de profils

Le moyen de création et de modification des profils est décrit dans le manuel utilisateur. Nous décrirons dans ce manuel la méthode de création de profil.

Le but du profil est de décrire un ensemble de droits pour les différents rôles joués dans l'entreprise. Les rôles doivent représenter des **fonctions** de l'entreprise(Acheteurs, Commerciaux, Administration, Ressource humaine). Il n'y a pas de notion de hiérarchie dans les rôles. Chaque rôle est indépendant. Un utilisateur ou un groupe peut jouer plusieurs rôles.

Pour que les profils soient utilisées de manière optimum, il est conseillé de ne travailler les profils que sur les rôles et non sur les personnes. Les droits des personnes doivent être déduits de leurs rôles ou de leurs groupes d'appartenances. Si les droits sont bien placés pour chaque rôle, il suffit d'affecter les utilisateur à leur rôles ou de les placer dans les groupes ayant le rôle voulu. Ceci a pour effet de changer leur droits. Il n'est alors plus nécessaire de modifier les profils individuellement.

Pour placer les droits nous vous conseillons de remplir un tableau comme celui-ci pour chaque droit :

Rôles/Familles	Recette	Contrat	Facture
Administrateur		+	+
Commercial		+	
Direction commerciale	+		+
Acheteur			+
Invité			

Les cellules vertes (+) représentent les droits ajoutés,. Ici, on a décidé que la famille *recette* n'avait pas de droit par défaut donc chaque nouveau document de la famille fichier est accessible de tout le monde. Le rôle *acheteur* a le droit de voir les factures.

Une fois les tableaux remplis il suffit de créer un profil par famille avec les droits indiqués dans les cellules vertes. Si des familles ont la même définition de profil, on peut associer le même profil à ces familles. Le but est d'avoir le moins de profil possible pour simplifier les modifications ultérieures.

Pour changer les droits d'une personne, il suffit alors de changer ses rôles ou ses groupes d'appartenances.

Bien sûr, il est toujours possible de modifier un profil pour une personne particulière si cette personne ne correspond à aucun rôle ni aucun groupe.

2.3.2.4 Affecter manuellement un profil à un document

Pour affecter manuellement un profil à un document existant, il faut utiliser le menu :

- Autre / Sécurité / Changer de profil

2.3.2.5 Profil par défaut des nouveaux documents

Pour chaque famille, il est possible d'indiquer le profil à appliquer aux nouveaux documents de cette famille :

- Menu “Sécurité / Changer le profil pour les nouveaux documents

2.3.2.6 Profil par défaut lors de l'enregistrement d'un document dans un dossier

Pour chaque document de type dossier, il est possible d'indiquer le profil à appliquer aux nouveaux documents ajoutés au dossier :

- Menu “Édition restriction”

2.3.2.7 Profil dynamique

Les profils dynamiques permettent de mettre des droits par rapport au contenu du document. Si un attribut est du type docid, et qu'il référence un document de la famille *Utilisateur* ou *Groupe d'utilisateurs* sa valeur peut être utilisée pour définir des droits sur le profil dynamique. Pour indiquer qu'un profil est dynamique, il suffit d'indiquer une famille dans le cadre dynamique lors de l'édition.

PROFIL DE DOCUMENT BROUILLON POUR COMPTES RENDUS profil de document

Basique

Titre : profil de document brouillon pour comptes rendus

Description : Seul le rédacteur peut le voir et le modifier.

Dynamique

Famille : Comptes rendus

Dans l'exemple montré ci-dessus, on crée un profil pour la famille compte-rendu. Dans cette famille plusieurs attributs sont de type "account". Dans l'interface d'accessibilité ces attributs sont présentés en bas avec un bouton jaune pour les attributs qui ne sont pas dans un tableau; en orange pour ceux qui sont multi-valués (présents dans un tableau ou ayants l'option multiple=yes).

Animal	Voir	Modifier	Supprimer	Envoyer	Déverrouiller	Confidentiel	Forum	Voir les réponses	Voir les droits	Modifier les droits
⇒ Contrôleur	●	●	●							
⇒ Inspecteur	●	●								
⇒ Surveillant	●									
⇒ Surveillant chef	●				●					
Consultants	●									
gardien responsable										
Dompteur	●									

9843 documents liés à ce profil

Mise à jour des droits | Annuler | Seulement les cochés | Voir les groupes

Tous les attributs de type "docid" sont présentés si aucune option "isuser" est indiquée et si aucun attribut de type "account" est déclaré. Cependant seuls ceux qui font référence à un utilisateur ou à un groupe d'utilisateurs auront une pertinence dans le calcul des droits.

Les attributs visibles sont de type "account"' ou "docid" (Pensez à utiliser l'option d'attribut `isuser=yes`).

Si un profil n'a pas besoin de dynamique, il ne faut pas renseigner la famille dans le cadre dynamique car cela implique des calculs supplémentaires qui ne sont pas nécessaires.

2.3.2.8 Profil de transitions de cycle de vie

Les famille *cycle de vie* définissent une famille de profil spécifique. Les droits des documents *cycles de vie* permettent de savoir qui a le droit d'effectuer les transitions. Ces droits sont applicables à tous les documents associés à ce document de cycle de vie.

workflow incident	voir	éditer	supprimer	vers enregistré	vers qualifié	vers analysé	vers traité	vers rejeté	vers suspendu	vers clôt	voir les droits	modifier les droits
default group												
Groupe Achats												
Groupe Administrateur	●	●	●	●	●	●	●	●	●	●	●	●
Développement Anakeen												
Groupe Commercial												
Groupe Direction												
Groupe Encadrement												
Groupe Ggestion&financier												
Groupe Incident				●	●	●	●	●	●	●		

Les droits d'un cycle de vie comportent cinq droits classiques (voir, éditer, supprimer, voir les droits et modifier les droits), et N droits portant sur les transitions. Ces droits sur les transitions sont définis lors de la programmation du cycle de vie.

Lorsque l'utilisateur effectue un changement d'état, il ne peut choisir l'état suivant que parmi ceux dont il a le droit. Dans l'exemple ci-dessus, le droit *vers qualifié* doit être activé si l'utilisateur veut changer d'état vers *qualifié*. Si ce droit n'est pas accordé, le changement d'état n'est pas proposé. Pour placer les droits sur une famille cycle de vie, il faut se rapporter à sa fiche de description. Chaque cycle de vie décrit ces droits de transition particuliers en fonction de son cycle de vie.

2.3.2.9 Profils d'état de cycle de vie

Les profils par défaut ne sont pas applicables aux familles *cycle de vie*. Dans les documents *cycle de vie* un profil par état peut être défini. Le profil par défaut peut être considéré comme le profil associé à l'état initial du document. On modifie les profils du document en éditant le document cycle.

À chaque changement d'état, le profil associé est affecté au document. Si un état n'a pas de profil associé, le document conserve son profil courant.

Si le cycle contient des spécificités par rapport à une famille, c'est à dire qu'il contient des actions particulières utilisables seulement par une famille, il est nécessaire de spécifier cette famille. Ainsi le cycle ne pourra être associé qu'avec cette famille ou ses descendants.

2.3.2.10 Points d'emplois des profils de documents

2.3.2.11 Administration des profils

Version 2.14.3 Afin de réaliser la mise en place des droits les plus courants, le document famille dispose d'une interface permettant d'accéder directement à la définition du paramétrage des droits.

2.3.2.11.1 Famille sans cycle de vie

Pour une famille sans cycle de vie par défaut vous avez accès aux droits pour le document famille et pour les documents de la famille.

Si vous n'avez encore effectué aucun profilage, les liens “initialisé les droits” apparaitrons. Si vous cliquez sur le lien concernant la famille cela va créer un profil de famille avec aucun droits positionnés hormis les droits pour celui qui vient de la faire. Le lien est alors remplacé par “modifier les droits”. Si vous cliquez dessus, l'interface de modification des droits apparait et vous pouvez affecter les différents droits aux groupes et utilisateurs.

De même pour les droits sur les documents. L'initialisation va créer un document profil qui sera activé avec aucun droit sauf pour l'utilisateur qui a créé le profil. Vous cliquez ensuite sur “modifier le profil” pour placer les droits des groupes et utilisateurs.

2.3.2.11.2 Famille avec cycle de vie

Si la famille a un cycle de vie par défaut, les différents profils associés aux états apparaitrons. Le profil par défaut des documents n'apparait pas car c'est le rôle du cycle d'imposer sa sécurité. Si vous vous voulez quand même mettre un profil par défaut, le menu “sécurité/changer le profil pour les nouveaux documents” est toujours accessible. Pour chacun des états vous pouvez “initialiser les droits” si aucun profil n'est déjà associé. Si le profil est déjà associé, vous pouvez alors le modifier ou le désactiver. La liste des états associés au profil est indiqué dans la liste (exemple “rédigé, publié”). Si vous voulez lier un profil à plusieurs état, il faudra passer par le cycle de vie pour le faire. Lorsqu'un profil est indiqué en rouge, cela signifie qu'aucun droits n'est positionné (tout le monde peut tout faire).

pas de famille

COMPTE RENDU test

| Créer Compte rendu | Changer d'icone | Éditer les attributs | Éditer les énumérés | Valeurs par défaut | Valeurs des paramètres | Renommer | Sécurité↓ | Changer le dossier racine | Changer la recherche par défaut | Choisir un cycle | Autres↓

Famille de document

Caractéristiques

Forum pour les nouveaux documents :	Interdire les forums
Dossier racine :	racine pour Comptes rendus
Cycle de vie :	Défaut Comptes rendus
Nombre maximum de révisions :	révisions illimités

Sécurité

- Famille
  - Initialiser les droits
- Document
  - États
    - initialisé : Modifier les droits, Désactiver les droits
    - rédigé, publié : Activer les droits
    - refusé : Initialiser les droits
    - Test : Initialiser les droits
  - Transitions
    - Modifier les droits pour les transitions, Désactiver les droits

Cette interface ne donne pas accès à toutes les possibilités de mise en place de sécurités liées à une famille. Son rôle est de permettre à un administrateur d'accéder aux modifications ponctuelles de droits.

2.3.3. Les masques

Le masque permet de masquer ou de démasquer des attributs. Ainsi, l'utilisateur suivant le masque voit et saisie que certaines parties du document.

ENREGISTREMENT INCIDENT masque de saisie

enregistrement pour incident

cadre	Nom	nouvelle visibilité	Nouvelle obligation	visibilité par défaut
	Problème			
	contact technique			
	contact appelant			
	identification			
	Qualification			
	Analyse	caché		
	Traitement	caché		
	hiddens			
identification	titre			**lecture écriture**
identification	idsite			caché
identification	site			lecture écriture
identification	n°contrat			lecture écriture
identification	idcontrat			caché
identification	référence			lecture seule
identification	date de création			lecture seule
identification	utilisé par			lecture seule
Problème	**produits impactés**			lecture écriture
produits impactés	produits			lecture écriture
produits impactés	idproduit(s)			caché

Le masque agit en modifiant la visibilité et le caractère obligatoire de l'attribut. Les masques successifs qui suivent les états permettent de compléter au fur et à mesure le document en indiquant les attributs obligatoires. Ils permettent aussi de voir de nouvelles parties de document suivant les états. L'utilisateur ne voit alors que les parties pertinentes suivant l'état du document.

Pour chaque attribut, la visibilité ou l'obligation peut être modifiée. Pour les attributs *frame* (cadre) ou *array* (tableau) la visibilité affectée peut être propagée aux attributs du cadre ou du tableau concerné. Ces attributs cadre ou tableau

sont indiqués en caractère gras. Les attributs qui sont dans la cadre d'un attribut cadre ou tableau ont leur visibilité modifiée si leur propre visibilité est « supérieure ». Exemple : si le cadre a une visibilité R, les attributs du cadre qui ne sont ni H ni I obtiennent la visibilité R. Le résultat de ces visibilités est affiché lors de la consultation du masque (après validation).

2.3.4. Les contrôles de vues

2.3.4.1 But de ce document

Le but de ce document est d'expliquer :

- Comment fonctionnent et à quoi servent les vues sous dynacase.
- Comment mettre en place un contrôle de vues permettant d'ajouter un menu contextuel sur une famille pour proposer d'autres vues d'affichage ou d'édition.
- Comment modifier la vue d'un document en fonction des droits de l'utilisateur.

2.3.4.2 Généralités

Le mécanisme mis en œuvre par les contrôles de vues permet de proposer et de contrôler l'accès à diverses représentations du document que ce soit en consultation ou en édition.

Les contrôles de vues permettent d'accéder à des attributs qui sont généralement cachés par défaut. La définition d'une vue du document se compose de quatre paramètres :

- L'identificateur : doit être unique dans le document vue et ne doit comporter que des lettres (sans espace).
- La description (label): intitulé de la vue affiché dans le menu contextuel
- Le type : consultation ou édition
- La zone : identifiant de la représentation du document (syntaxe: APP:LAYOUT).
- Le masque : document *masque de saisie* utilisé pour cette vue.
- L'ordre : numéro de préférence de la vue à appliquer dans le cas d'une consultation ou d'une édition par défaut. Si aucun numéro n'est mis, la vue ne sera pas prise en compte par défaut. Si des numéros sont indiqués dynacase choisira le premier accessible par l'utilisateur suivant ses droits dans l'ordre croissant (le premier étant 1).
- Affichable : pour indiquer que la vue est accessible via le menu contextuel

2.3.4.3 Présentation des vues par défaut

Par défaut, dynacase propose 3 vues :

- La vue de consultation (affichage d'un document)
- La vue d'édition (modification d'un document)
- La vue résumée (vue d'affichage simplifiée qui apparaît par exemple dans le module « Une famille »

Une vue permet donc :

- De modifier l'affichage des différents attributs (ou champs) d'une famille
- De masquer certains champs visibles par défaut
- D'afficher des champs invisibles par défaut

- De rendre les champs modifiables ou non

2.3.4.4 Exemple de contrôle de vues

Pour expliquer comment mettre en place un contrôle de vues, nous allons créer une nouvelle famille et plusieurs vues pour arriver à ce résultat : Cadre réservé au demandeur :

- Objet de la demande (en lecture seule par la direction)
- Description de la demande (en lecture seule par la direction)

Cadre réservé à la direction :

- Réponse à la demande (visible en lecture seule par le demandeur)
- Commentaire de la direction (non visible par le demandeur)

2.3.4.5 Création de la famille de test

Pour notre exemple, nous allons créer cette famille de test :

idattr	idframe	label	T	A	type	ord	vis	need
TEST_FRAME_DEM		Cadre réservé au demandeur	N	N	frame	0	W	N
TEST_FRAME_DIR		Cadre réservé à la direction	N	N	frame	1	W	N
TEST_TEXT11	TEST_FRAME_DEM	Objet	Y	N	text	2	R	Y
TEST_TEXT12	TEST_FRAME_DEM	Description	N	N	longtext	3	R	N
TEST_TEXT21	TEST_FRAME_DIR	Réponse	N	N	text	4	R	N
TEST_TEXT22	TEST_FRAME_DIR	Commentaire	N	N	longtext	5	H	N

Pour importer cette famille dans dynacase, vous pouvez télécharger FIXME internalmedia: documentation:mad:famille_test_vue.ods et l'importer avec cette commande :

```
/usr/share/what/wsh.php --api=freedom_import --file=famille_test_vue.ods
```

Dans cette famille, nous pouvons remarquer :

- Que les attributs « Objet », « Description » et « Réponse » sont en lecture seule par défaut
- Que l'attribut « Commentaire » est masqué

2.3.4.6 Création des masques

Un masque permet d'afficher ou masquer certains champs (attributs) ou de modifier leur visibilité. Dans notre cas, nous allons devoir créer 3 masques :

- Un masque d'édition pour que le demandeur puisse modifier les champs « Objet » et « Description »
- Un masque d'édition pour que la direction puisse éditer les champs « Réponse » et « Commentaire »
- Un masque de consultation pour que la direction puisse voir le champ « Commentaire » mais pas le demandeur

Pour créer un masque, il faut :

- Menu « Création / Document système »
- Dans la liste « hérite de », sélectionner « masque de saisie »
- Dans la liste « pour » sélectionner la famille « Test Vue 1 » importée précédemment
- Indiquer un nom à votre masque
- Pour chaque attribut, il est possible d'indiquer sa nouvelle visibilité et sa nouvelle obligation

Le premier masque que l'on nommera « **Masque édition demandeur** » est paramétré comme cela :

masque de saisie

MASQUE ÉDITION DEMANDEUR TEST VUE 1

Famille

familles : Test Vue 1

Masques

cadre	Nom	nouvelle visibilité	visibilité par défaut
Cadre réservé au demandeur	Objet	**lecture écriture**	**lecture seule**
Cadre réservé au demandeur	Description	lecture écriture	lecture seule
Cadre réservé à la direction	Réponse	statique	lecture seule
Cadre réservé à la direction	Commentaire	caché	caché

Le deuxième masque que l'on nommera « **Masque édition direction** » est paramétré comme cela :

masque de saisie

MASQUE ÉDITION DIRECTION TEST VUE 1

Famille

familles : Test Vue 1

Masques

cadre	Nom	nouvelle visibilité	visibilité par défaut
Cadre réservé au demandeur	Objet	**statique**	**lecture seule**
Cadre réservé au demandeur	Description	statique	lecture seule
Cadre réservé à la direction	Réponse	**lecture écriture**	lecture seule
Cadre réservé à la direction	Commentaire	lecture écriture	caché

Le troisième masque que l'on nommera « **Masque consultation direction** » est paramétré comme cela :

masque de saisie

MASQUE CONSULTATION DIRECTION TEST VUE 1

Famille

familles : Test Vue 1

Masques

cadre	Nom	nouvelle visibilité	visibilité par défaut
Cadre réservé au demandeur	Objet	**statique**	**lecture seule**
Cadre réservé au demandeur	Description	statique	lecture seule
Cadre réservé à la direction	Réponse	statique	lecture seule
Cadre réservé à la direction	Commentaire	statique	caché

Remarque : Pour retrouver facilement vos masques, il faut enregistrer une nouvelle recherche sur la famille « masque de saisie »

2.3.4.7 Création du contrôle de vues

Un contrôle de vue permet :

- D'ajouter un menu contextuel sur une famille de documents pour changer de masque manuellement
- De changer de masque automatiquement en fonction des droits de l'utilisateur

Pour créer un contrôle de vues, il faut :

- Menu « Création / Document système »
- Dans la liste « hérite de », sélectionner « contrôle de vues »
- Indiquer un titre au contrôle de vues
- Sélectionner la famille rattachée à ce contrôle de vues. L'attribut *famille* doit être préalablement saisi si on veut utiliser des zones ou des masques propres à cette famille. Si cet attribut est rempli le contrôle de vue ne pourra être appliqué que sur des documents de la famille en question ou pour les familles descendantes.
- Remplir les cases comme indiqué ci-dessous :

id vues	label	type	zone	masque	ordre	affichable
edit_demandeur	Edition Demandeur	Edition	FDL:EDITBODYCARD	Masque édition demandeur Test Vue 1	1	non
edit_direction	Edition Direction	Edition	FDL:EDITBODYCARD	Masque édition direction Test Vue 1	2	non
consult_direction	Consultation Direction	Consultation	FDL:VIEWBODYCARD	Masque consultation direction Test Vue 1	3	non

Remarque sur le champ zone :

Le champ « zone » permet d'utiliser des vues XML personnalisées. Dans ce document, nos utilisons les vues livrées par défaut dans dynacase. Pour apprendre à créer une vue personnalisée, il faut suivre cette documentation .

Remarque, en édition, sur les champs “ordre” et “affichable” :

Si “affichable” est sélectionné, l'utilisateur, qui aura accès au masque dont “affichable” est sélectionné, verra à la fois le menu “Editer” classique et le menu de ce masque. Si “affichable” n'est pas sélectionné, alors “ordre” prend toute sa signification. dynacase vérifiera que l'utilisateur a accès au masque d'édition d'ordre 1. Si l'utilisateur a droit d'accès, alors ce sera ce masque qui sera affiché lorsque l'utilisateur cliquera sur le menu “Editer” classique. Si l'utilisateur n'avait pas accès à ce masque d'ordre 1, alors dynacase testera les accessibilités de cet utilisateur vis à vis du masque d'ordre 2, et ainsi de suite ...

Remarque sur l'ordre des vues et vue par défaut :

Il est recommandé de spécifier les attributs de la famille avec des visibilités restreintes.

Ainsi, si l'utilisateur n'a pas de droits et que le contrôle de vue ne lui trouve aucune vue à appliquer, alors il aura la visibilité restreinte par défaut des attributs de la famille.

Par la suite, si vous voulez, par exemple, que `admin' ait une visibilité complète sur les attributs, il faudra alors créer explicitement une vue d'ordre 1 qui sera accessible à l'utilisateur `admin' et qui lui ouvrira la visilibilité sur tous les attributs avec un masque adéquat.

2.3.4.8 Affecter le contrôle de vues à la famille

Pour affecter le contrôle de vues sur notre famille de test, il faut :

- Afficher le document de votre famille
- Cliquer sur Sécurité↓
- Sélectionner le menu « Changer de profil par défaut pour les nouveaux documents »

- Dans la liste « Contrôle de vues », sélectionner le contrôle de vues créé précédemment.

2.3.4.9 Test du contrôle de vues manuel

Une fois le contrôle de vues affecté à la famille, la création d'un nouveau document doit prendre en compte par défaut la vue « Masque édition demandeur ». Ce qui donne :

Une fois le document validé, un clic droit sur ce dernier doit faire apparaître les menus suivants :

- Éditions spéciales / Édition Demandeur
- Éditions spéciales / Édition Direction
- Vues spéciales / Consultation Direction

La vue « Édition demandeur » correspond à celle par défaut montrée précédemment. La vue « Édition Direction » ressemble à celle-ci :

La vue « Consultation Direction » correspond à celle-ci :

Le nombre de vue géré par le contrôle de vue est limité à 20. Ceci est du aux nombres de droits possibles sur un document qui sont de 32 - 12 prédéfinis.

2.3.4.10 Mise en place de droits sur le contrôle de vues

Pour que le choix de la vue se fasse automatiquement en fonction des droits de la personne connectée, il est possible d'affecter des droits au contrôle de vues. **Attention** : Pour suivre l'exemple ci-dessous, il est nécessaire de créer deux utilisateurs appartenant aux groupes « Demandeur » et Direction » Ensuite, il faut :

- Sélectionner « Autres /Sécurité / Profil dédié » .
- Sélectionner « Autres /Sécurité / Accessibilité »
- En cliquant sur les petits boutons gris à la droite des groupes, il est possible d'afficher des cases à cocher permettant de sélectionner les droits comme indiqué ci-dessous :

Attention : Les droits sur les vues sont calculés par rapport à l'ordre des vues telles quelles ont été définies dans le tableau. La colonne "ordre" n'est pas utilisée pour la définition des droits. Cela veut dire qu'on défini le droit pour la vue de la rangée n°1, n°2 et n°3 (pour l'exemple). Ni le nom de la vue (colonne : id vue) , ni l'ordre (colonne : ordre) n'est pris en compte pour l'association des droits, seul compte l'ordre du tableau. Cela veut dire que si vous supprimez la vue n°2 (edit direction), les droits de l'ancienne vue n°3 (cons direction) devenue n°2 seront ceux de la vue supprimée (edit direction) qui est l'ancienne n°2. Il faut dans ce cas refaire la matrice de droit du contrôle de vue.

2.3.4.11 Test du contrôle de vues automatique

Une fois les droits appliqués, les utilisateurs des groupes « Direction » et « Demandeurs » auront automatiquement accès à leur vue de consultation et d'édition par défaut et ils ne pourront pas choisir une autre vue. Par exemple, le groupe demandeur ne pourra pas voir le champ « Commentaire » car ce dernier n'est accessible qu'à travers la vue « Consultation Direction » :

2.4 Importation et exportation de documents

2.4.1. Exportation

2.4.1.1 Exportation CSV

Le contenu d'un dossier ou le résultat d'une recherche peut être exporté dans un fichier au format CSV avec comme délimiteur le **point-virgule**.

L'export n'est possible que si l'administrateur a le droit EXPORT sur l'application FDL et l'acl FREEDOM_MASTER de FREEDOM.

Pour exporter, il faut cliquer sur le menu *outils/exportation du dossier* depuis la fenêtre d'ouverture de dossier (ou de recherche). L'accès à l'exportation est aussi disponible sur le menu contextuel affichable en cliquant sur les icônes de dossier ou de recherche présentes dans l'arborescence.

L'export produit un fichier CSV (séparé par des point-virgules) ou un fichier archive (Zip) si vous avez sélectionnez avec les fichiers. Explications des options :

option	signification
Encodage	Deux encodages sont proposés. Cela est nécessaire pour les valeurs textuelles qui comportent des accents. Si votre poste client est sur le système Windows et que vous voulez utiliser MicroSoft Excel, il est préférable de choisir ISO 8859-15. Sinon pour openOffice.org il est conseillé d'utiliser UTF-8. Lors d'une importation dynacase détecte automatiquement l'encodage et supporte correctement ces 2 encodages.
Profil	Cela est utile pour sauvegarder les profils associé à un document. Seuls les profils des documents avec profils dédiés (généralement les documents *profils*) sont exportés. Cette option est surtout utile lorsqu'un concepteur de famille veut récupérer son paramétrage profil.
Fichiers	L'option *sans les fichiers* indique que seul le fichier csv contenant les valeurs des attributs sera exporté. Les fichiers liés aux attributs de type *file* ou *image* ne sont pas exportés. L'option *avec les fichiers* indique que l'exportation génèrera un fichier archive de type *Zip*. Cette archive contiendra le fichier `fdl.csv` qui contient les valeurs des attributs et les fichiers attachées aux documents exportés. Cette archive peut être réimporté à l'importation d'archive de dynacase (Voir ci après). Pour des raison de compatibilité avec les différents systèmes d'exploitations les noms des fichiers exportés ne comportent pas d'accents
Identificateur	Permet de sauvegarder les identificateurs numériques des documents. Cela implique que votre fichier ne sera ré-importable que dans la même base. Cela identifie de manière non ambiguë le document dans l'objectif d'une restauration. Dans tous les cas, si les documents comportent des noms logiques (propriété *name*), ces noms sont exportés.

Chaque ligne commençant par DOC présente les attributs visibles et invisibles (type hidden) d'un document. Les documents sont rangés par famille. À chaque changement de famille, une rangée indique les définitions des attributs de la famille (1ère colonne : `/ /FAM`).

L'ordre de présentation des attributs suit l'ordre défini par les attributs de la famille. Les quatre premières colonnes sont communes pour tous les documents.

- 1ère colonne : toujours égale à DOC
- 2ème colonne : identificateur de la famille
- 3ème colonne : identificateur du document
- 4ème colonne : identificateur du dossier (ou de la recherche) où a été extrait le document

Le document est exploitable sur tout tableur tel que Microsoft Excel, OpenOffice.org ou Gnumeric en spécifiant que le caractère séparateur est le **point-virgule**.

	A	B	C	D	E	F	G	H	I	J
1	//FAM	fichiers(FILE)	<specid>	<fldid>	titre	titre	sujet	mots-clés	résumé	autres
2	ORDER	FILE			fi_title	fi_titlew	fi_subject	fi_keyword	fi_description	ft_t_oformat
3	DOC	FILE		-	Divers.tar.gz		71		les collines du Gers	
4	DOC	FILE		-	Collines.zip					
5	DOC	FILE		-	definition_droits_familles.sxc					
6	DOC	FILE		-	archivage.odg					
7	//FAM	fichier(SIMPLEFILE)	<specid>	<fldid>	nom du fichier	titre	sujet	mots-clés	résumé	type court
8	ORDER	SIMPLEFILE			sfi_title	sfi_titlew	sfi_subject	sfi_keyword	sfi_description	sfi_mimetxtshort
9	DOC	SIMPLEFILE		-	C5 (copie).html	C			donc il faut un résumé	HTML document text

2.4.1.2 Exportation XML

Les documents, hormis les familles, peuvent être exportés en XML. Les fichiers XML produits suivent un schéma de famille imposé par dynacase. Chaque famille de document a son propre schéma XML.

L'exportation XML d'un contenu de dossier ou de recherche peut se faire de deux façons : en générant un fichier par document ou avec un seul fichier xml.

Si vous voulez utiliser le fichier XML pour un exportation vers un autre contexte dynacase, il faut indiquer l'option "sans les identificateurs", sinon les documents importés seront considérés comme des mise à jours.

2.4.1.2.1 Un fichier xml par document

Dans ce cas une archive (zip) est constituée avec un fichier par document. Chaque fichier est nommé à l'aide du titre et de l'identifiant du document (ex :alligator {4567].xml). Chaque fichier fait référence à un schéma xml issus de la famille. Ce schéma est aussi présent dans l'archive

Totor Alligator{1148}.xml	1,1 Kio	document XML
Viel Ali Alligator{1289}.xml	1,3 Kio	document XML
zoo_animal.xsd	21,3 Kio	W3C XML schema

2.4.1.2.2 Un fichier xml seul

dans ce cas, la sortie est un seul fichier xml contenant l'ensemble des documents. La balise racine est dans ce cas "documents". La balise documents porte la date de génération et l'auteur.

```
<documents date="2010-06-07T12:46:45"
   author="Default Master"
   name="recherche_animal">
  <zoo_animal .> </zoo_animal>
  <zoo_animal .> </zoo_animal>
</documents>
```

Avec cette option les Xml schema des familles ne sont pas importés.

2.4.1.2.3 Exemple d'un document exporté:

```
<zoo_animal xsi:noNamespaceSchemaLocation="zoo_animal.xsd" id="1147" name=""
title="Alli Alligator" revision="0" modification-date="2010-06-02T13:53:33"
version="" state="">
<an_tab1>
  <an_identification>
    <an_nom>Alli</an_nom>
    <an_descr xsi:nil="true"/>
    <an_tatouage xsi:nil="true"/>
    <an_espece_title>Alligator</an_espece_title>
    <an_espece id="1127">Alligator</an_espece>
    <an_ordre>Crocodiliens</an_ordre>
    <an_classe id="1123" name="Reptilia">Reptilia</an_classe>
    <an_sexe>M</an_sexe>
    <an_photo xsi:nil="true"/>
    <an_photo vid="27" mime="image/png" href="http://localhost:80/dev/?
app=FDL&amp;action=EXPORTFILE&amp;cache=no&amp;inline=no&amp;vid=27&amp;docid=11
46&amp;attrid=an_photo&amp;index=-1" title="alligator.png"/>

    <an_naissance>2010-05-12</an_naissance>
    <an_entree>2010-05-18</an_entree>
    <an_pere id="1289">Viel Ali Alligator</an_pere>
  </an_identification>
</an_tab1>
</zoo_animal>
```

2.4.1.2.4 Exportation xml avec fichier

Si vous cochez l'option avec les fichiers ceux-ci seront importés "inline" dans les fichiers xml. La balise de l'attribut contiendra dans ce cas le fichier encodé en base 64.

```
<an_photo vid="27" mime="image/png" title="alligator.png">
iVBORw0KGgoAAAANSUhEUgAAAQAAAAEACAYAAABccqhmAAAACXBIWXMAAAsTAAALEwEAmpwYAAAABGdB
TUEAALGOfPtRkwAAACBjSFJNAAB6JQAAgIMAAPn/AACA6QAAdTAAAOpgAAA6mAAAF2+SX8VGAAEoA0lE
QVR42mL8//8/wwgE==</an_photo>
```

2.4.1.2.5 Récupération des Xml schema seuls

Pour récupérer un fichier Xml schema de manière autonome, vous devez utiliser l'interface d'importation de document via l'application "gestion documentaire". Sur cette interface un bouton "Xml Schema" permet de récupérer le schéma compatible avec les documents XML exportés.

2.4.1.2.6 Exportation Xml par ligne de commande

Vous pouvez lancer l'export avec wsh. La commande est :

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=<folderid> --eformat=[X|Y]
--wfile=[N|X]
```

export Xml sans fichier

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=Y > t.xml
```

export Xml avec fichier

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=Y --wfile=Y> t.xml
```

export Zip sans fichier

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=X > t.zip
```

export Zip avec fichier

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=X --wfile=Y> t.zip
```

export Xml avec option flat pour une mise à plat des attributs. Ce type d'exportation ne permet pas une réimportation dans dynacase. Les balises de structure de type "tab" et "frame" ne sont pas représentées. Seules la structure concernant les tableaux est maintenu.

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=Y --flat=Y> t.xml
```

export Xml avec log. Cette option permet de connaître le nombre de documents exportées ainsi que la liste de ceux-ci.

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --id=1180 --eformat=Y
--log=/var/tmp/log.txt > t.xml

~$ more /var/tmp/log.txt
EXPORT BEGIN OK : 02/07/2010 15:49:47
EXPORT OPTION FLAT : no
EXPORT OPTION WFILE : yes
EXPORT OPTION CONFIG : yes
EXPORT OPTION ID : ANIMAUX <Les animaux>
EXPORT DOC OK : <a Agouti> [3300]
EXPORT DOC OK : <Albert baleine blanche> [3266]
EXPORT DOC OK : <Alli Baleine à  bosse ou jubarte> [3248]
EXPORT DOC OK : <Zor Antilope> [3317]
EXPORT COUNT OK : 3
EXPORT END OK : 02/07/2010 15:49:47
```

export Xml avec spécification des attributs à exporter. Cela est paramétrable à l'aide d'un fichier de configuration xml : exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <family name="ZOO_ANIMAL">
    <attribute name="AN_NOM"/>
    <attribute name="AN_ESPECE"/>
    <attribute name="AN_ENFANT"/>
  </family>
  <family name="ZOO_ENCLOS">
    <attribute name="EN_NOM"/>
    <attribute name="EN_SURFACE"/>
  </family>
</configuration>
```

Chaque balise "family" défini les attributs qu'il veut voir apparaitre. Seul les attribut "feuille" non structurant sont pris en compte. Les attributs "name" désignent les identifiant de famills et d'attributs. Si on exporte une document d'une famille qui n'est pas dans le fichier de configuration, l'ensemble de ces attributs sera exportés.

```
./wsh.php --app=FDL --action=EXPORTXMLFLD --flat=no --id=ANIMAUX --wfile=yes
--eformat=Y  --config=configexport.xml --log=/var/tmp/log.txt > animaux.xml
```

2.4.1.3 Exportation Rapport

Les documents rapports permettent de construire des recherches sur un ensemble de documents de la même famille et d'en extraire un ensemble de données. Ils possèdent un mode d'export en CSV qui permet d'analyser les résultats du rapport en dehors de Dynacase et notamment d'en faire des analyses statistiques dans excel.

2.4.1.3.1 Exportation Rapport : l'interface WEB

L'interface web fourni un assistant qui permet de spécifier les paramètres d'export :

- le type d'export :
  - simple : l'export sera similaire à la représentation web, c'est à dire que pour chaque attribut sélectionné, il y aura une colonne dans le CSV
  - par pivot : dans ce type, les éléments multiples seront répliqués pour permettre leur analyse via des tableau dynamiques croisés, pour ce faire vous devez sélectionner un pivot
- les options d'exports :

  - le séparateur de cellule : c'est le caractère qui séparera deux cellules dans le CSV
  - le séparateur de texte : c'est le caractère qui délimitera le contenu textuel des cellules
  - le séparateur de décimal : c'est le séparateur utilisé pour les nombreux non entier

- le jeu de caractère : ISO 8859-15 ou UTF8

Il est à noter que les valeurs par défaut sont celles utilisés par excel 2003 pour la lecture des CSV.

2.4.1.3.2 Exportation Rapport : en CLI

Vous pouvez appeler l'action d'export en CLI si vous souhaitez automatiser l'exportation de rapport.

```
./wsh.php --app=FDL --action=REPORT_EXPORT_CSV --id=ID_DU_RAPPORT
```

Les options facultatives sont :

- kind : le type d'export simple ou pivot (simple par défaut),
- pivot : le nom logique de l'attribut pivot (n'est pris en compte que dans un export de type pivot) (id par défaut),
- refresh : true, ou false si true lance la méthode refresh sur les documents avant des les exporter (false par défaut),
- delimiter : le séparateur de cellule (, par défaut)
- enclosure : le séparateur de texte (" par défaut)
- encoding : le jeu de caractère (utf8 par défaut)
- decimalSeparator : le séparateur de décimal (. par défaut)

NB : les réglages par défaut correspondent à un standard US

2.4.2. Importation

2.4.2.1 Importation CSV/ODS

2.4.2.1.1 Constitution du fichier d'importation

Le fichier d'import peut être au format :

- CSV (**séparateur de champ point-virgule** / pas virgule, **séparateur de texte vide**) avec un encodage latin9 ou Unicode UTF-8
- OpenDocument Spreadsheet (tableur OpenOffice.org .ods)

2.4.2.1.1.1 Généralités

Comme pour l'export les quatre premières colonnes définissent les paramètres du document :

- 1ère colonne : toujours DOC
- 2ème colonne : identifiant de la famille
- 3ème colonne : identifiant du document. Utiliser un identifiant existant indique une modification du document existant. Un identifiant vide ou égal à zéro indique une création de document.
- 4ème colonne : identifiant du dossier où insérer le document. Si un identificateur est précisé le document est référencé dans le dossier (s'il n'y est pas déjà). Si l'identificateur n'est pas précisé (vide ou zéro), le document est référencé dans le dossier courant (celui qui est sélectionné dans l'arborescence). Si l'identifiant est '-' alors il ne sera référencé dans aucun dossier.

Les colonnes suivantes décrivent les valeurs des attributs du document.

2.4.2.1.1.2 Ordre des attributs

Fix Me! ORDER

2.4.2.1.1.3 Clés d'imports

Fix Me! KEYS

2.4.2.1.2 Importation générale de documents sans fichier

L'importation de document utilise un fichier de même format que pour l'exportation. **Par contre, certains attributs présents dans le fichiers d'exportation peuvent ne pas être présents dans le fichier d'importation**. Pour réutiliser un fichier issu de l'exportation, il faudra bien s'assurer que les colonnes soient les mêmes (ordre et définition) que celles demandées.

> On ne peux pas importer des attributs qui sont calculés en fonction des données de la base : ils seraient systématiquement écrasé par le nouveau calcul.

Pour importer un fichier décrivant des documents, il faut sélectionner *import de documents* dans la barre de menu Outils. Afin de voir les formats d'importation des familles il suffit de sélectionner la famille souhaitée.

L'import de fichiers associés aux attributs de type *file* et *image* n'est pas possible dans ce cas.

Avant l'import, il est nécessaire d'analyser le fichier d'import pour être sûr de ce qu'on importe. L'analyse indique le titre du document importé si le document est un nouveau ou une mise à jour et dans quel dossier il va être placé.

S'il y a au moins un document, les boutons d'importations sont activés. Si le nombre de documents est supérieur au paramètre FDL_MAX_FGEXPORTDOC, l'importation directe reste désactivée seul l'importation en tâche de fond est possible.

Les nouveaux documents (pas d'identificateur spécifié) qui ont le même titre qu'un document existant de la même famille sont ignorés par défaut. Ils peuvent être ajouté si on change la politique d'import. Dans le cas '*Ajout nouveau document*', un nouveau document avec le même titre qu'un document existant sera créé.

Dans le cas de l'importation en tâche de fond, le rapport d'importation sera envoyé au mail spécifié (par défaut celui de l'utilisateur) une fois l'import effectué.

2.4.2.1.2.1 Importation spécifique de documents sans fichier

Plus d'options d'importation sont disponibles avec ce type d'importation. Par contre la construction du fichier CSV reste la même. Cette importation est disponible via l'application '*Gestion par famille*', dans le menu '*outils*' choisir '*fichier à importer*' ou par le bouton '*import spécifique*' disponible sur l'interface d'importation générale.

Cette importation peut être utilisée pour l'importation d'un ensemble de documents de la même famille. Il faut auparavant sélectionner la famille que l'on veut importer avant de sélectionner '*fichier à importer*'.

Avec ce type d'importation, l'utilisateur peut choisir l'ordre des colonnes composant le CSV en cliquant sur '*ordre des colonnes*'. Les quatre premières colonnes (DOC;<id famille>;<id document>; <id dossier> restent immuables. Par défaut l'ordre des colonnes est celui défini par la famille. Vous pouvez soit supprimer des colonnes, soit modifier leur ordre. Pour supprimer il faut sélectionner la (les) colonne(s) et les déplacer dans le cadre de droite (bouton $-3ÌÍ). Pour changer l'ordre des colonnes, il faut utiliser les boutons ÌÍ et ÌÏ .L'ordre et la définition des colonnes utilisés sera celui du cadre de gauche. Lorsque vous modifiez cet ordre, l'écriture des colonnes en bas est aussitôt modifié. Vous pouvez utiliser ces deux lignes inscrites dans le cadre pour votre fichier CSV afin d'indiquer l'ordre de vos colonnes. La première ligne 'ORDER' indique au logiciel le nouvel ordre des attributs d'importation établi pour cette famille. S'il n'y a pas de ligne ORDER dans le fichier, l'ordre sera celui défini dans le cadre gauche. Dans le cas contraire ce sera l'ordre défini dans le fichier qui sera pris en compte. Les quatre premières colonnes d'une ligne ORDER sont composées du mot-clef 'ORDER', de l'identificateur de la famille puis de deux colonnes non utilisées (servant seulement pour l'alignement avec les lignes DOC). Les colonnes suivantes contiennent les identifiant des attributs de la famille. La deuxième ligne inscrite dans le cadre du bas ('#DOC') sert juste à aider l'utilisateur sur la signification des colonnes.

Le cadre dossier indique tous les dossiers fils et petit-fils du dossier par défaut défini par la famille. Lors d'un importation, vous pouvez choisir l'emplacement des documents importé dans un ou plusieurs des dossiers proposés.

Le cadre 'Politique' permet de choisir la politique d'importation lorsqu'un document à importer possède des similitudes avec un document existant. La similitude est détectée par défaut si un document d'une même famille possède le même titre. L'utilisateur peut modifier les clefs de recherche de similitude en utilisant un ou deux attributs de la famille (exemple : pour un article manufacturé on peut définir que l'on a affaire à un même document si la référence constructeur et le constructeur sont identiques).

La cadre '*valeurs par défaut*' permet d'affecter des valeurs à toutes valeurs nulles (vides) définies dans le fichier d'importation. Ces valeurs par défaut sont uniquement utilisées pour des nouveaux documents, pas pour les mise à jours.

Valeurs Par Défaut Pour L'import

Identification

Référence :

Désignation :

Photo : Browse...

Constructeur :

Unité :

Lien constructeur :

Coûts

Frais généraux % : 0

Marge % : 5

TVA % : 19,6%

La case '*Analyse seulement*' est cochée par défaut. Si vous appuyez sur '*Importation CSV*' alors l'analyse du fichier sera effectuée et le résultat sera affiché dans le cadre du bas.

Résultat de l'analyse de l'import

10 documents à prendre en compte.

ligne	titre	dossier	id	famille	action	message	modifications	erreur
2	CMBPA-10K Powerware	/onduleurs	22113		mis à jour	à mettre à jour CMBPA-10K Powerware [22113]	◆ [référence:CMBPA-10K] ◆ [désignation:coffret métal by-pass automatique 10KVA pour commutation automatique de la charge sur le réseau en cas de panne onduleur et sans coupure] ◆ [id fournisseur:15894] ◆ [id constructeur:19631] ◆ [constructeur:Powerware]	
3	BRDA02 Powerware	/onduleurs			ajouté	BRDA02 Powerware à ajouter	◆ [référence:BRDA02] ◆ [désignation:Boitier report d'alarme onduleur - pose à l'extérieur du local technique pour visualiser les alarmes] ◆ [id fournisseur:15894] ◆ [id constructeur:19631]	

Les ajouts sont notifiés en vert, les modifications en jaune et les document ignorés en rouge.

Le résultat de l'analyse indique le nombre de documents à prendre en compte (ajouté ou modifié). Il indique pour chaque ligne du fichier, l'interprétation effectuée et l'action qu'il entreprendra. Si l'analyse est conforme aux attendus vous pouvez décocher 'analyse seulement' est lancer réellement l'importation. Le résultat apparaîtra dans le cadre du bas à la place de l'analyse.

2.4.2.1.2.2 Importation d'arborescence

L'importation d'une arborescence de fichiers peut être effectuée avec l'importation d'archive. L'archive contient une arborescence de répertoires contenant des fichiers. L'arborescence de répertoires sera rattachée au plan de classement en créant une arborescence de dossiers. Ensuite les fichiers ou les descriptions de documents seront insérés dans ces dossiers.

Cette archive permet :

- de créer un plan de classement
- d'ajouter des documents contenant des fichiers avec des caractéristiques
- d'insérer des fichiers sans caractéristique

Constitution de l'archive

Vous devez créer sur votre machine locale votre arborescence de répertoires. Dans chacun de vos répertoires vous placez les fichiers à importer. Ces fichiers seront par défaut ajoutés dans le dossier correspondant à votre plan de classement.

Dans chaque répertoire, un fichier de description nommé *fdl.csv* doit être créé pour indiquer les caractéristiques des fichiers. Si ce fichier n'est pas présent la conversion par défaut sera effectuée et aucune caractéristique ne sera présente dans les documents contenant les fichiers du répertoire. Ce fichier de description est tel qu'il est décrit dans le paragraphe précédant. Pour les attributs de type *fichiers*, il suffit d'indiquer le chemin relatif vers le fichier.

FAM	image(IMAGE)	<specid>	<fldid>	titre	image	description
ORDER	IMAGE			img_title	img_file	img_description
DOC	IMAGE		-	place principale	photos/Carnaval 005.jpg	vue du balcon\navec les bandas
DOC	IMAGE		-	animations	photos/Carnaval 006.jpg	

Dans l'exemple ci-dessus, le fichier *fdl.csv* indique la création de trois documents en attachant trois fichiers de l'archives. Le répertoire courant contient ici un sous répertoire *photos* contenant les fichiers jpeg.

Pour chaque fichier de description, le mot-clef DFAMID indique la famille de conversion par défaut pour les fichiers non référencés. Ces fichiers sont ceux qui n'ont pas été spécifiés dans un des fichiers de description de l'archive. S'il n'y a pas de redéfinition de DFAMID dans les sous répertoires, la famille par défaut définie par le répertoire père sera réutilisé. Le fichier de description décrit le document du dossier courant. Il ne peut pas décrire des documents d'autres dossiers. Les colonnes *<special id>* et *<special dirid>* doivent être vides dans ce cas pour l'ajout de document.

De même que pour la famille des fichiers par défaut, le mot-clef DFLDID indique la famille du container par défaut pour les répertoires. Par défaut DFLDID est l'identifiant de la famille dossier. Il faut obligatoirement que cet identificateur soit une famille dérivée de dossier.

Une fois l'arborescence, les fichiers et les fichiers de description remplis, il suffit de créer une archive sur votre poste local. Le format de fichier de l'archive peut être un tar compressé (tar.gz, tgz, tar.bz2). La compression du tar peut être effectuée soit par gzip (par défaut), soit par bzip2. Ce format peut aussi être un Zip (répertoire compressé par défaut sous Windows).

Téléchargement de l'archive

Une fois l'archive créé, il faut la transférer sur le serveur documentaire. Si l'archive est assez petite en taille, un transfert classique peut être effectué. La taille maximum d'importation dépend de la configuration de PHP (Upload configuration) au niveau du serveur. Une modification de ce paramètre implique un redémarrage de Apache.

Si l'archive est trop volumineuse, un transfert plus classique (scp, ftp) peut être effectué par l'administrateur. Il faut alors la placer sous le répertoire suivant du serveur : <FREEDOM_UPLOADDIR>/<login>/tars

Où FREEDOM_UPLOADDIR est le paramètre applicatif indiquant le répertoire de stockage (par défaut /tmp/upload), login est le nom de connexion de l'utilisateur. Par exemple pour jean.martin le répertoire de stockage des archives sera /tmp/upload/jean.martin/tars/.

Une fois l'archive téléchargée, il suffit d'aller aux archives pour voir le résultat de l'extraction.

Les fichiers commençant par un point sont ignorés.

Analyse de l'archive

Le résultat de l'extraction est visible dans la première partie de la fenêtre. S'il y a plusieurs archives on peut sélectionner celle que l'on souhaite. Seules les archives téléchargées par l'utilisateur sont présentées ici. Ces archives sont dans un espace temporaire et elles peuvent être supprimées.

La seconde partie montre ce qui sera effectué lors de l'importation dans la base documentaire.

Trois options d'importation sont proposées:

Dans tous les cas les dossiers sont créés conformément à l'arborescence de fichiers.

- Seul fichier CSV : si coché ne traite que des documents qui sont décrits dans les fichiers *fdl.csv*. Sinon traite d'abord les documents issus des fichiers de description puis des fichiers non référencés.
- Famille fichier par défaut : famille utilisée pour les fichiers non référencés dans le cas où DFAMID n'est pas renseigné
- Famille dossier par défaut : famille utilisée pour les répertoires dans le cas où DFLDID n'est pas renseigné
- Dossier racine : ajoute un dossier pour contenir l'ensemble de l'importation de l'archive. Le nom du dossier est celui de l'archive. Il peut être modifié après l'importation.

À chaque modification d'options, il faut relancer l'analyse pour effectuer l'importation.

Importation de l'archive

Une fois que l'analyse est conforme à vos attentes il suffit de lancer l'importation en tâche de fond. Un email vous sera envoyé avec le résultat de l'analyse dès que l'importation sera finie.

Après importation, vous pouvez alors supprimer l'archive afin d'éviter une deuxième importation non souhaitée.

2.4.2.1.2.3 Importation de profils

Création de profil

Pour créer un profil, il faut créer un document profil en utilisant la famille "PDOC".

//FAM	profil de document(PDOC)	<specid>	<fldid>	titre	description
ORDER	PDOC			ba_title	prf_desc
DOC	PDOC	PRF_ARCHIVE	-	Archive	Lecture écriture pour les administrateurs
PROFIL	PRF_ARCHIVE			view=GADMIN	edit=GADMIN

Pour les profils de dossier vous devez utiliser la famille PDIR, pour les recherches la famille PSERACH et pour les familles la famille PFAM.

Affectation de profil

Pour ajouter des droits à un profil de document, vous pouvez utiliser le mot-clef "PROFIL".

Le mot-clef (colonne 3) "ADD" indiquera le comportement par défaut à savoir l'ajout. Le mot-clef "DELETE" indiquera une suppression de droit: on supprimera les droits spécifiés (s'ils existent).

Les droiits sont décrit à partir de la colonne 5. La syntaxe est la suivante :

```
<acl>=<identifiant groupe ou utilisateur>[,<identifiant groupe ou utilisateur>]
```

l'acl est le nom du droit :

- view : voir le document
- edit: : éditer/modifier le document
- delete : supprimer le document
- unlock, déverrouiller un autre verrou que le sien

Consulter le chapitre Sécurité pour voir de manière exhaustive les droits possibles.

Ces droits peuvent être différents suivant les familles. Ainsi pour les profils de dossier, les trouve aussi les droits :

- open : ouvrir le contenu d'un dossier
- modify : modifier le contenu d'un dossier (enlever ou ajouter un document)

PROFIL;MY_PROFIL;;RESET;;view=GDEFAULT;edit=GDEFAULT
PROFIL;MY_PROFIL;;ADD;;view=GDEFAULT;edit=GDEFAULT
PROFIL;MY_PROFIL;;DELETE;;view=GDEFAULT;edit=GDEFAULT

Il est possible d'indiquer plusieurs utilisateur ou groupe

Le signe - indique un droit négatif (boule rouge)

PROFIL;MY_PROFIL;;ADD;;view=GDEFAULT;-edit=GDEFAULT

PROFIL	MYPROF		RESET	view=GDEFAULT,GADMIN	edit=GADMIN
PROFIL	MYOTHERPROF		DELETE	view=GDEFAULT	

Pour les profils dynamiques, il est possible de mettre au lieu de l'utilisateur, l'identifiant de l'attribut:

PROFIL	MY_PROF2			view=us_meid	edit=us_meid

Définitions des accessibilités

Permet d'ajouter des droits sur les applications.

La colonne n°2 indique l'identifiant du groupe. L'identifiant est le nom logique du document ou l'identifiant système du groupe (attribut 'us_whatid').

ACCESS	GRP_DEV	APP_TEST	ACL_USER	ACL_ADMIN	ACL_EXPORT
ACCESS	GRP_COM	APP_TEST	ACL_USER		

Voir aussi chapitre Importation de groupes et gestion des droits sur les actions.

Divers

Affectation d'icone

Permet de changer l'icone d'un document.

DOCICON	MY_DOCU	my_tool.png

La colonne n°2 indique l'iddentifiant d'un document. La colonne n°3 indique un nom de fichier image. Ce fichier doit être présent sur le serveur dans le répertoire 'images'.

2.4.2.1.3 Importation de documents famille

Voir chapitre Famille

2.4.2.1.4 Importation de table d'échanges dynacase -> LDAP

Les familles peuvent être exportés sur un LDAP. Chacunes des valeurs d'attributs peuvent être copiées dans un attribut LDAP.

//	family	ldap attribute	freedom attribute	ldap schema	card index
LDAPMAP	IGROUP	cn	::getLDAPTitle()	inetOrgPerson	1
LDAPMAP	IGROUP	dn	::getLDAPDN(uid,dc=users)	top	1
LDAPMAP	IGROUP	mail	GRP_MAIL	inetOrgPerson	1
LDAPMAP	IGROUP	sn	GRP_NAME	inetOrgPerson	1

Cet extrait de table de correspondance indique que l'attribut grp_mail de la famille "groupe intranet" ira dans l'attribut "mail" du schéma inetOrgPerson. Il est aussi possible de renseigner un attribut LDAP à l'aide d'un méthode. Ainsi l'attribut "cn" aura la valeur retournée par la méthod "::getLDAPTitle()". L'attribut "dn" (distinguish name) indique sur quelle branche la carte LDAP sera attachée.

Il est possible d'enregistrer plusieurs cartes LDAP pour un seul document freedom. Pour différencier les cartes on utilisera un index différent.

//	family	ldap attribute	freedom attribute	ldap schema	card index
LDAPMAP	IGROUP	dn	::getLDAPDN(cn,dc=people)	top	2
LDAPMAP	IGROUP	description	GRP_ROLE	groupOfNames	2
LDAPMAP	IGROUP	cn	US_LOGIN	groupOfNames	2
LDAPMAP	IGROUP	member	::getLDAPMember()	groupOfNames	2
LDAPMAP	IGROUP	ou	US_LOGIN	organizationalUnit	3
LDAPMAP	IGROUP	dn	::getLDAPDN(ou,dc=people)	top	3

2.4.2.1.5 Importation de masse

Pour les importations importantes de document sans fichier, vous pouvez accélérer votre importation en utilisant le script wsh `csv2sql`. Ce script transforme un fichier CSV (avec ';' comme séparateur) (ou ODS) contenant des DOC en commande sql prêtes à l'insertion en base de données.

Ce script permet une importation brute. C'est à dire pas d'appel aux méthodes Doc::PostModify ou Doc::postCreate, pas de vérification d'unicité.

```
#
# wsh --api=csv2sql --file=/var/tmp/test.csv | psql <dynacase database>
```

Il est bien sûr fortement conseillé de bien vérifier la sortie du script avant l'injection.

Dans les fichiers CSV, il faut obligatoirement que les DOC soient précédés d'une ligne ORDER indiquant la correspondance entre les colonnes et les attributs.

2.4.2.2 Importation XML

L'importation de document peut être faite à l'aide de fichier XML tels que ceux produits par l'exportation XML. L'interface graphique pour importer est la même que pour les fichiers CSV et ODS. On y accède depuis la gestion documentaire (menu "outils/importer des documents").

Le type de fichier accepté est soit un fichiers XML avec balise root "<documents>", soit un fichier zip contenant un ensemble de fichier XML importés. Ces fichiers XML doivent être conformes au schéma XML associé à la famille de document.

Si vous importez un XML contenant des attributs relations et que dans votre XML vous n'indiquez pas explicitement un id ou un name alors une recherche par titre sera faite afin de trouver automatiquement le document qui a le même titre en cohérence avec la famille déclarée dans la relation au niveau de la famille. Si aucun id n'est trouvé la valeur de l'attribut sera vide.

```
<an_espece>Alligator</an_espece>
```

S'il y a un attribut 'id' ou 'name' c'est celui ci qui sera utilisé pour effectuer la relation ; dans ce cas le nom (contenu de la balise) est ignoré.

```
<an_espece id="4563">Alligator</an_espece>
<an_classe name="Reptilia">Reptile</an_classe>
```

Les fichiers encodés contenus dans le XML sont insérés dans le documents et remplace le fichier original.

Avec la politique de mise à jour de document (par défaut), les documents sans identifiants recherchent un document similaire avec les clef d'import. La chef d'import par défaut est "title". Si un document et un seul de la même famille avec le même titre a été trouvé alors il sera considéré comme une mise à jour.

La clef d'import peut être changée en utilisant l'attribut key:

```
<zoo_animal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="zoo_animal.xsd"  key="an_tatouage" revision="0"
modification-date="2010-06-16T09:07:37" version="" state="">
  <an_identification>
     <an_nom>Gérard</an_nom>
     <an_espece id="1131">Antilope</an_espece>
     <an_tatouage>32</an_tatouage>
     <an_enfant_t><an_enfant id="1148">Totor Alligator</an_enfant></an_enfant_t>
     <an_enfant_t><an_enfant id="1147">Alli Alligator</an_enfant></an_enfant_t>
  </an_identification>
</zoo_animal>
```

Dans cet exemple, ce sera le tatouage qui sera pris comme référence. Si un animal de même tatouage est trouvé il sera mis à jour sinon il sera ajouté. Il est possible d'avoir une clef secondaire en ajoutant après une virgule un deuxième attribut : key="an_tatouage,an_espece".

2.4.2.2.1 Test de validité des fichiers XML

Les fichiers XML peuvent être validés à l'aide du schéma avant importation.

Cette vérification est à faire manuellement à l'aide du schéma correspondant. Lorsque vous importer une recherche sur une famille spécifique et que vous choisissez l'option "import XML avec Zip" alors cette archive contient le schéma XML. Vous pouvez utiliser le logiciel xmllint pour vérifier vos fichiers XML

```
eric@tarfful:~/Bureau$ ls -l Recherche_totor.zip
-rw-r--r-- 1 dev dev 189675 2010-06-16 13:39 Recherche_totor.zip
dev@tarfful:~/Bureau$ mkdir test
dev@tarfful:~/Bureau$ cd test/
dev@tarfful:~/Bureau/test$ unzip ../Recherche_totor.zip
Archive:  ../Recherche_totor.zip
  inflating: Totor Alligator{1148}.xml
  inflating: Totor junior 1 Alligator{3307}.xml
  inflating: Totor junior 2 Alligator{3308}.xml
  inflating: Totor junior 3 Alligator{3309}.xml
  inflating: Totor junior 4 Alligator{3310}.xml
  inflating: zoo_animal.xsd
dev@tarfful:~/Bureau/test$ xmllint --noout --schema zoo_animal.xsd  *xml
Totor Alligator{1148}.xml validates
Totor junior 1 Alligator{3307}.xml validates
Totor junior 2 Alligator{3308}.xml validates
Totor junior 3 Alligator{3309}.xml validates
Totor junior 4 Alligator{3310}.xml validates
```

2.4.2.3 Importation en ligne de commande

La commande wsh dynacase_import permet d'importer des documents en ligne de commande. Les types CVS, ODS, XML, ZIP, TGZ (pour les archives) sont supportés par le scripts :

```
./wsh.php --api=importDocuments --file=/home/dev/Bureau/recherche_animal.xml
```

Usage :

```
Import documents from description file
Usage :
     --file=<the description file path>
```

```
   Options:
     --userid=<user system id to execute function - default is (admin)>, default
is '1'
     --analyze=<analyze only> [yes|no], default is 'no'
     --archive=<description file is an standard archive (not xml)> [yes|no],
default is 'no'
     --log=<log file output>
     --htmlmode=<analyze report mode in html> [yes|no], default is 'yes'
     --reinitattr=<reset attribute before import family update> [yes|no]
     --to=<email address to send report>
     --strict=<don't import if one error detected> [yes|no], default is 'yes'
```

Si vous voulez un compte rendu et une pré-analyse :

```
 ./wsh.php --api= importDocuments --analyze=yes --htmlmode=yes
--file=/home/eric/Bureau/recherche_animal.xml > report.html
```

Vous pouvez aussi envoyer le rapport par mail

```
 ./wsh.php --api= importDocuments --to=somone@somewhere.org
--file=/home/eric/Bureau/recherche_animal.xml
```

Pour avoir un fichier de log ajouter l'option log

```
./wsh.php --api= importDocuments --file=animaux.xml --htmlmode=yes --analyze=yes
--log=/var/tmp/log.txt
$ cat /var/tmp/log.txt
IMPORT BEGIN OK : 02/07/2010 17:10:29
IMPORT DOC OK : [title:Isabelle] [id:0] [action:added] [changes:{nom:Isabelle}
{espèce:1132}] [message:Isabelle à ajouter]
IMPORT DOC OK : [title:Alli2 Agouti] [id:3296] [action:updated] [changes:]
[message:]
IMPORT DOC KO : [title:] [id:0] [action:ignored] [changes:] [message:]
[error:DOMDocument::load(): Opening and ending tag mismatch: r line 5 and
zoo_animal in /var/tmp/xmlsp
lit4c2e10009374f/3299.xml, line: 10DOMDocument::load(): Premature end of data in
tag zoo_animal line 2 in /var/tmp/xmlsplit4c2e10009374f/3299.xml, line: 11]
IMPORT COUNT OK : 2
IMPORT COUNT KO : 1
IMPORT END OK : 02/07/2010 17:10:31
```

Si une seule erreur est détectée sur le fichier d'importation, aucun document du fichier d'importation ne sera ajouté ou modifié.

2.4.2.4 Création d'archives d'import avec fichiers sous Windows

Ce chapitre décrit comment produire une archive d'import avec fichiers dans un environnement Windows.

2.4.2.4.1 Création du fichier `fdl.csv'

Le fichier `fdl.csv' de description d'import doit être au format CSV, avec jeu de caractères "Unicode (UTF-8)", séparateur de champ ";" (point-virgule), et séparateur de texte vide.

Pour créer un fichier `fdl.csv' compatible Dynacase il faut utiliser OpenOffice et créer un nouveau classeur :

Supprimez les feuilles additionnelles du classeur et ne conserver qu'une feuille.

Enfin, enregistrez le fichier au format "Texte CSV" :

- Menu "Fichier" > "Enregistrer sous..."
- Nommez le fichier "fdl.csv"
- Sélectionnez le type "Texte CSV (.csv)"
- Cochez "[X] Éditer les paramètres du filtre"

Après avoir cliqué sur le bouton "Save", OpenOffice peut demander de confirmer la sauvegarde au format "Texte CSV".

Cliquez alors sur "Conserver le format actuel" pour confirmer la sauvegarde au format "Texte CSV".

OpenOffice va ensuite demander les options pour "Texte CSV".

Les paramètres à utiliser pour un import Dynacase sont alors :

- Jeu de caractères : "Unicode (UTF-8)"
- Séparateur de champ : ";" (le caractère point-virgule)
- Séparateur de texte : <vide> (effacez le contenu et laissez à vide)

2.4.2.4.2 Création de l'archive Zip d'import

Mettre le fichier `fdl.csv' dans le dossier avec les fichiers référencés par le CSV.

Les fichiers doivent être stockés dans l'archive Zip avec un nom en Unicode UTF-8. Pour cela, utilisez l'outils 7-Zip (http://www.7-zip.org/) en version >= 9.20 :

- Sélectionner le répertoire contenant les fichiers
- Clic-droit sur le répertoire > "7-Zip" > "Add to archive..."

Dans la fenêtre de création de l'archive qui s'ouvre, il faut alors sélectionner/entrer les paramètres suivants :

- Archive : le nom de l'archive que vous souhaitez créer
- Format de l'archive : "zip"
- Paramètres : "cu=on"

Validez la création de l'archive en cliquant sur le bouton "[OK]".

2.4.2.4.3 Importer l'archive Zip

Pour importer cette archive, il faut aller sur l'application "Docs admin" de Dynacase, puis dans le menu "Outils" > "Importation Archive" :

La page "Importer une archive" permet alors de sélectionner l'archive à importer (bouton "[Parcourir]") et d'envoyer cette archive au serveur (bouton "[Télécharger l'archive]") :

Après avoir fait "[Télécharger l'archive]", une page de statut va confirmer la réception de l'archive par le serveur :

Note : La page "Résultat de l'importation d'archives" peut indiquer que "L'archive import.zip n'a pue être extraite" si certains fichiers n'ont pas été correctement extraits de l'archive par exemple. Dans ce cas, vous pourrez consulter et vérifier le contenu de l'archive sur la page d'analyse "Voir l'archive -import.zip-".

Cliquez sur "[Voir l'archive - import.zip -]" pour voir l'analyse de l'archive par Dynacase :

Cette analyse permet de voir :

- les fichiers extraits de l'archive dans la section "Choisir l'archive" ;
- les opérations qui seront exécutés par Dynacase lors de l'import effectif de l'archive dans la section "Résultat de l'analyse".

L'import final de l'archive se faisant en tâche de fond, il faut renseigner une adresse mail qui recevra le rapport d'import dans le champ "rapport à (email)", et lancer l'import en cliquant sur le bouton "[Effectuer l'import en tâche de fond de N documents]".

L'archive est alors importée et un rapport est envoyé à la fin de l'opération sur l'adresse mail renseignée précédemment :

Import de l'archive en cours

L'import import.zip est en cours. Lorsque la mise à jour sera finie, un email sera envoyé à <john.doe@example.net> avec le rapport d'importation

2.4.2.4.4 Modification du fichier `fdl.csv'

Pour modifier le fichier `fdl.csv', ouvrir celui-ci avec OpenOffice.

Lors de l'ouverture du fichier, OpenOffice va demander de spécifier les options qui doivent être :

- Jeu de caractères : "Unicode (UTF-8)"
- Séparé par : cochez "[X] Point-virgule", et décochez "[ ] Virgule"
- Séparateur de texte : <vide> (effacez le contenu et laissez à vide)

Pour la sauvegarde, une fois le fichier ouvert, il ne sera alors pas nécessaire de faire "Enregistrer sous..." mais simplement "Enregistrer" ; et les paramètres de jeu de caractères, séparateur de champ et séparateur de texte spécifiés à l'ouverture seront ré-appliqués automatiquement pour la sauvegarde.

2.5 Archives

2.5.1. Introduction

dynacase permet d'archiver des documents afin qu'ils ne soient plus vus comme des documents "vivants".

Un document archivé n'est plus accessible par les recherches classiques.

Un document archivé a un profil donné par l'administrateur de l'archive

Les documents archivés ne sont plus vus dans les recherches par défaut comme pour les documents mis à la poubelle.

2.5.2. Constitution de l'archive

Pour construire une archive, il est nécessaire de créer un document "archive". Le droit de créer des archives et donné par défaut au groupe "administrateurs". Les utilisateurs par défaut ne peuvent ni créer ni voir les documents archives.

Les seules données à remplir pour créer une archive sont de renseigner son nom et sa description.

Une fois créée, l'archive à le statut "Ouvert". Cela signifie que l'administrateur peut ajouter ou enlever des documents dans l'archive. L'archive étant un dossier, tout les moyens d'insertions classiques de document dans un dossiers peuvent être employés pour constituer l'archive. Cependant pour faciliter la constitution de l'archive des menus supplémentaires ont été ajoutés aux documents et aux contenus de collections.

Si vous avez les droits d'insertions (droit "modifier" et "voir") sur les archives ouvertes, alors les boutons "insérer dans l'archive <xxx>" apparaissent sur les documents. Si le document est une collection le menu "tout insérer dans l'archive <xxx>" apparaît en plus. Dans la gestion documentaire, sur les résultats de recherche un menu "tout insérer dans l'archive <xxx>" apparait dans le menu "outils".

Les documents archivés ne peuvent pas être archivés par une autre archive.

Pour voir les documents archivés, il faut aller sur l'archive et cliquer sur le menu "ouvrir". Si le nombre de document archivé est supérieur à 1000 ou si vous recherchez un document précis dans une archive en cours de constitution vous pouvez cliquer sur "voir les documents". Cela permet d'accéder au filtre sur les valeurs des documents.

2.5.3. Archivage des documents

Une fois que l'archive est constituée vous pouvez archiver son contenu. Cela va consister à marquer les documents comme "archivé" et à leur mettre le profil applicable de l'archive. Chaque archive à un profil applicable différent. Vous pouvez à tout moment modifier ce profil : pendant la constitution ou après. Si vous ne voulez pas que les utilisateurs puissent modifier les archives il faut mettre ce profil en lecture seule. Si vous ne voulez pas qu'ils y accèdent vous ne leur donnez pas le droit de voir tout simplement. Le profil applicable par défaut ne donne aucun droit à personne à part au créateur de l'archive. dynacase n'interdit pas que l'on puisse modifier des documents archivés si vous donnez le droit "éditer" sur le profil.

Une fois l'archive close, vous ne pouvez plus ajouter ou enlever des documents de cette archive.

Si vous affichez un document archivé l'emblème indiquera cet état.

Si vous vous êtes précipité lors de l'archivage, il est possible de revenir en arrière en désarchivant l'archive. Dans ce cas l'ensemble des documents archivés par cette archive seront restaurés comme auparavant. Leur profils initiaux seront restaurés. L'archivage comme le désarchivage sont des procédures qui peuvent être longues car elle dépendes du nombre de documents à archiver. Il est donc conseillé de réaliser ces manipulations lorsque peu de personnes sont connectées au risque de pénaliser les performances. Chaque archivage est notifié dans chacun des historiques des documents impactés.

Vous pouvez rechercher dans les documents archivés si vous avez le droit de voir le document archive et si vous avez le droit de voir les documents archivés. Si vous avez ces droits, l'option "Dans l'archive" de la recherche simple vous permettra de trouver des documents archivés. Vous pouvez aussi à partir de menu "voir les documents archivés" de l'archive accéder à la recherche sur son contenu.

2.5.4. Destruction des documents archivés

Une fois l'archive close vous pouvez détruire son contenu. Cette étape permet de supprimer physiquement les documents.

Cette destruction est irréversible.

Si vous activez cette ultime étape, chacun des documents sera supprimé de la base de données. L'archive gardera les titres et références de l'ensemble des documents détruits.

Si vous cliquez sur un lien référençant un document détruit vous pourrez accéder à ses dernières traces conservées dans les log et l'historiques.

Dernières traces du document 1266 : Objet 17/CR17

Derniers logs

Date	Utilisateur	Message	Code
31/05/2010 12:54:51.317445	Default Master		delete
31/05/2010 12:54:36.242389	Default Master	Archiving into Compte rendu de 2009	archive
31/05/2010 10:35:23.532465	Default Master	Unarchiving from Compte rendu de 2009	unarchive
31/05/2010 10:25:54.140537	Default Master	Archiving into Compte rendu de 2009	archive
31/05/2010 10:17:17.661795	Default Master		create

Derniers historiques

Date	Utilisateur	Message
31/05/2010 12:54:51	Default Master	Détruit par la purge de l'archive Compte rendu de 2009
31/05/2010 12:54:36	Default Master	Archiving into Compte rendu de 2009
31/05/2010 10:35:23	Default Master	Unarchiving from Compte rendu de 2009
31/05/2010 10:25:54	Default Master	Archiving into Compte rendu de 2009
31/05/2010 10:17:17	Default Master	Création du document
31/05/2010 10:17:17	Default Master	Document inséré dans le dossier Default Master
31/05/2010 10:17:17	Default Master	mise à jour par import

2.6 Fonctions Wsh

Les fonctions définies dans le répertoire API sont activables via le programme “wsh.php”. Elles doivent être exécutées dans une fenêtre shell sur la machine serveur.

2.6.1. Comment exécuter une fonction WSH

Pour exécuter une fonction WSH, il y a deux méthodes :

1 - Indiquer le chemin complet de la fonction. Exemple :

```
# sudo <control_root>/wiff context "test" exec wsh.php
```

Le <control_root> est le répertoire d'installation de dynacase-control.

2 - Lancer le shell . Exemple :

```
# sudo <control_root>/wiff context "test" exec /bin/bash
$ ./wsh.php
```

2.6.2. listapi

Cette commande, permet de connaître les fonctions de l’API disponible :

```
./wsh.php --listapi
```

Remarque : Le résultat de cette commande, donne en faite la liste des fichiers .php du répertoire ”/usr/share/what/API”

2.6.3. fdl_adoc

Permet de reconstruire les classes PHP conformément à la définition décrite dans la table docattr de la base Postgresql Freedom.

Paramètres :

- docid : référence de la famille à mettre à jour. Si non précisé toutes les familles sont mises à jour.

```
./wsh.php --api=fdl_adoc --docid=100
```

À utiliser si des modifications ont été faites directement dans la base de données.

2.6.4. freedom_clean

Supprime les documents temporaires et nettoie les contenus des dossiers.

Paramètres : aucun.

```
./wsh.php --api=freedom_clean
```

Ce nettoyage est effectué tous les jours de manière automatique.

2.6.5. freedom_convert

Convertit un document dans une autre famille. Le document perd tous les attributs qui n'existent pas dans la nouvelle famille.

Paramètres :

- tofamid : référence de la nouvelle famille
- docid : référence du document

```
./wsh.php --api=freedom_convert --tofamid=124 --docid=4568
```

2.6.6. importDocuments

Permet d'importer des documents au format CSV ou OpenOffice.org dans la base dynacase.

Paramètres :

- file : chemin du fichier
- analyze : (optionnel) si "yes" alors analyse le fichier seulement. Si "no" (défaut), l'importation est effectuée.
- dirid : (optionnel) référence du dossier par défaut pour classer les documents. Le document est classé dans ce dossier si sa référence de dossier (3ème colonne) est vide ou égale à zéro. Si cet attribut vaut zéro et qu'il n'a pas de référence dossier, alors le document n'est pas classé. Si cet attribut n'est pas précisé et qu'il n'a pas de référence dossier, il est classé dans le dossier par défaut de la famille s'il existe sinon il est classé dans le dossier« Import » (référence 10).
- log : (option) chemin du fichier où inscrire le rapport d'importation
- reinitattr : (optionnel) : si yes, dans le cas d'un import de description de famille, alors les précédentes descriptions des attributs (dans la table docattr) sont supprimées. Le fichier de Class (Method.*.php) est re-généré. Dans le cas de l'import des familles de type cycle de vie, il faut “initialiser” tous les documents de ces familles pour reconstituer les attributs. Dans tous les cas, les valeurs en base de données ne sont pas affectées. Remarque : cette option n'a pas à être utilisée autrement que pour importer des familles (par exemple, à ne pas utiliser pour l'import de paramétrages de documents “système” (masque, contrôle de vue, profils, ...))

```
./wsh.php --api=importDocuments --file=/home/eric/Banques.csv --analyze=yes
```

2.6.7. refreshDocuments

Permet de recalculer les valeurs des attributs dynamiques d'un document ainsi que de leur titre. Le rafraîchissement se fait pour tous les documents d'une même famille.

Paramètres :

- famid : référence de la famille des documents à rafraîchir.

```
./wsh.php --api=refreshDocuments --famid=100
```

2.6.8. refreshDocuments avec une méthode en argument

La méthode refreshDocuments permet de passer une méthode en argument. Cet exemple montre comment changer le profil d'un document et de tous les documents d'une famille.

Paramètres :

- famid = id (logique ou nombre) de la famille
- method = méthode de la famille à activer
- docid (OPTIONNEL) = id du doc (si pas précisé = tous les documents)
- arg = chaîne de caractère étant les arguments passé à la méthode

Affecter le profil 1234 au document 8776 de la famille MA_FAMILLE :

```
./wsh.php --api=refreshDocuments --famid=MA_FAMILLE --docid=8776
--method=setProfil --arg=1234
```

Affecter le contrôle de vue 2345 à tous les documents de la famille MA_FAMILLE :

```
./wsh.php --api=refreshDocuments --famid=MA_FAMILLE --method=setcvid --arg=2345
```

Affecter le contrôle de vue 2345 uniquement au document 8776 de la famille MA_FAMILLE :

```
./wsh.php --api=refreshDocuments --famid=MA_FAMILLE --docid=8776
--method=setcvid --arg=2345
```

2.6.9. ods2csv

Cette fonction permet de convertir un fichier OpenOffice.org Calc (.ods) en fichier CSV. Le fichier CSV peut ensuite être utilisé pour réaliser des importations avec la fonction “freedom_import” documentée ci-dessus.

Paramètres :

- odsfile : Permet d'indiquer le nom du fichier .ods à convertir
- csvfile (optionnel) : Permet d'indiquer le nom du fichier .csv de destination. Si cette option n'est pas utilisée, le résultat est donné sur la sortie standard.

```
./wsh.php --api=ods2csv --odsfile=/MonChemin/MonFichier.ods
./wsh.php --api=ods2csv --odsfile=/MonChemin/MonFichier.ods
--csvfile=/MonChemin/MonFichier.csv
```

Remarque : Il n'est pas nécessaire de convertir un fichier OpenOffice.org en fichier CSV avant de l'importer dans dynacase, car dynacase est capable d'importer directement le fichier OpenOffice.org.

3 Développement

3.1 Construire son module

3.1.1. Introduction

définitions, vocabulaire, principes généraux, architecture & composants d'une application

Ce guide a pour objectif de proposer aux développeurs un cadre dans lequel ils peuvent développer et maintenir un module Dynacase.

Ce guide défini les processus et les outils à utiliser pour créer son module. Il ne va pas définir techniquement l'usage des outils à utiliser mais fournira des liens sur les documentations nécessaires à la maîtrise de ces outils.

3.1.1.1.1 Vocabulaire :

	Définition
module	Un module Dynacase contient la définition et l'ensemble des éléments nécessaires au fonctionnement de zéro, une ou plusieurs applications. Elle contient la définition et l'ensemble des éléments nécessaires (cycle de vie, profil, ...) à l'utilisation de zéro, une ou plusieurs familles.
application	Une application Dynacase permet de couvrir une fonctionnalité spécifique au système documentaire (exemple: gestion des utilisateurs, administration des contrôle qualité). Une application utilise souvent une ou plusieurs familles de document afin de réaliser leurs fonctions. Une application Dynacase est composée de une à plusieurs actions. Une application possède généralement une interface principale qui permet d'accéder à l'ensemble des actions nécessaires à l'application. Une application est accessible directement par une url directe et est considéré comme autonome vis-à-vis d'autre application.
action	Une action Dynacase réalise une fonction d'une application. Cette action reçoit des paramètres, réalise un traitement et retourne le résultat de ce traitement. L'action Dynacase est composée de deux parties : le traitement et la représentation du résultat du traitement. Chacune de ses deux parties est optionnelle mais il en faut au moins une des deux. Le traitement est réalisé à l'aide d'une fonction PHP qui utilise des fonctions de l'API Dynacase. La représentation est réalisée à l'aide d'un template (appelé aussi layout). Ce template permet de réaliser des sorties HTML avec des parties dynamiques.
interface	L'interface Dynacase est la représentation d'une action.
template, layout	Le template est le modèle de représentation. Ce modèle est généralement du HTML avec des parties dynamiques qui sont construite dans le traitement de l'action. Ce modèle peut aussi être n'importe de n'importe quel type texte tel que csv ou xml. Le seul modèle binaire possible est le format openDocumentText. Ce type de modèle permet d'avoir des représentations utilisables pour des besoins d'impressions. Ces modèles sont utilisés pour les représentations des actions et aussi pour les vues de documents
zone	Une zone Dynacase est une action qui ne peut pas être appelée directement depuis l'interface principale. Une zone peut être considéré comme un fragment d'action. Sa représentation est souvent un fragment HTML (le contenu d'une DIV ou d'une TABLE). La zone est réutilisable dans différentes représentations d'actions. Elle permet de réutiliser des fragments d'interfaces dans les différentes interfaces de haut niveau. Une zone peut elle aussi faire référence à d'autres zones. On peut ainsi assembler des interfaces avec des niveaux différents réutilisable.
document	Le document Dynacase contient un ensemble structuré d'informations. Il peut être associé à un cycle de vie et porte ses propres paramètres de sécurité d'accès. Un document est toujours associé à une famille de document.
attribut	Un attribut de document défini un type syntaxique (texte, nombre, date, ...) ainsi que son emplacement dans la structure d'information du document.
famille	La famille permet de donner une unité à un ensemble de document de même

	type sémantique (compte-rendu, facture, client, recette de cuisine,...). La famille défini la structure de l'information qui sera commune à l'ensemble des documents de cette famille. Cette structure est composée d'attributs du document. Elle défini aussi les règles de sécurités d'accès et l'accès au cycle de vie.
vues	Une vue de document désigne un représentation de celui-ci. Cette vue est généralement associé à un template qui lui confère un aspect particulier. Une permet aussi masquer ou de démasquer un ou plusieurs attributs du documents.
fichier	Le fichier est un type d'attribut particulier. La famille peut déclarer un ensemble d'attribut “fichier” . Le fichier n'est nullement obligatoire dans la définition d'un document. L'accès au fichier ainsi que son contrôle d'accès se fait toujours au travers du document. Les fichiers peuvent se retrouver dans des attribut tel que : “annexes”, “esquisse du modèle”, “photographie”.
cycle de vie	Le cycle de vie définit l'ensemble des activités à réaliser, les états et transitions possibles du documents. Un cycle peut être associé à un document; dans ce cas le document est soumis aux contrôles imposé par le cycle. à chaque changement d'état, l'ancien état du document est conservé afin de tracer et de pouvoir voir les informations des états précédents.
profil	Le profil d'un document décrit une matrice de droits. Les droits les plus usuel sont 'voir','modifier', 'supprimer'. Pour chacun des ces droits, le profil défini quels groupes d'utilisateurs ou quels utilisateurs en particulier ont cette habilitation. Les droits des groupes sont propagés automatiquement sur les membres de groupes et des éventuels sous-groupes. Le même profil peut être partagé par plusieurs documents. Cela permet de modifier plus facilement les accès à un ensembles de document de même niveau de sécurité. Généralement, un même profil est partagé par les documents d'une même famille.

3.1.1.2 Description de l'espace de développement type

Pour développer un module Dynacase, un répertoire contenant les différents fichiers de configuration est nécessaire. Ce répertoire sera nommé “répertoire source”.. Il contient l'ensemble des éléments nécessaire à la publication du module afin de réaliser des tests unitaires et à la création de fichier d'exportation au format “webinst” nécessaire à une installation par dynacase-control.

3.1.1.2.1 hiérarchie des répertoires

Initialisation de la hiérarchie de répertoire pour la création d'un module mono application.

On récupère une archive et on lance le programme d'initialisation (demande de nom du module et de l'application).

Fichier/répertoire	Description
<APP>_app	Fichier de description de l'application ainsi que les définitions des différentes actions possibles[8]
<APP>_init.php.in	Fichier de description des paramètres de l'application.
<APP>_fr.po	Catalogue des traductions pour le français
<APP>_en.po	Catalogue des traductions pour l'anglais
<APP>_post	Fichier de description des post-traitemement à effectuer lors de l'installation ou la mise à jour
info.xml.in	Fichier de description du module
configure.in	Fichier de configuration servant à générer le fichier configure qui servira à générer les fichiers résultats à partir des fichiers .in
Makefile.in	Fichier Makefile décrivant l'ensemble des fichiers à publier et le traitement à réaliser pour la publication

8) *Voir chapitre "Créer une nouvelle application"*

Pubrule	Fichier de règles générales sur lesquels les Makefiles s'appuient.
VERSION	Fichier contenant la version du module sous la forme X.Y.Z. X,Y et Z sont des chiffres. X est le n° de version majeure, Y celui de version intermédiaire et Z celui version mineure.
RELEASE	Fichier contenant le n° de release. contenant un nombre positif ou zéro.
Families	Fichiers ods de définitions des structures Fichiers des méthodes du document. Fichiers des templates de vues de documents Fichiers ods de définitions des profils Fichiers d'aide à la saisie
Families/Views	Fichiers des templates de vues de documents (xml/odt)
Families/Externals	Fichiers d'aide à la saisie Fichier de définition des recherches spécialisées
Actions	Fichiers décrivant les actions. Cela comprend les fichiers PHP pour les traitements, les fichiers templates pour les résultats des traitements
Actions/Zones	Fichiers décrivant les zones. Cela comprend les fichiers PHP pour les traitements, les fichiers templates pour les résultats des traitements
Scripts	Fichiers PHP pouvant être lancé par la commande wsh. Ces fichiers définissent des traitements qui peuvent être fait par ligne de commande. Ils servent généralement à effectuer des opérations d'exploitations comme un recalcul sur en ensemble de documents ou une opération d'importation cyclique par exemple.
Images	Différents fichiers images (PNG/JPEG) utilisé pour les vues, les templates et les icônes de familles.

3.1.1.2.2 Traduction

Les traductions des éléments d'interfaces et des messages se fait à l'aide de l'outil gettext. Cet outil fonctionne à l'aide de catalogue de traduction, un par langue. L'application Dynacase utilise un seul catalogue pour l'ensemble des traduction du noyau et de l'ensemble des modules. Ceci implique que les mots ou les phrases à traduire doivent être unique par module. Pour cela, il est fortement conseillé d'utiliser un préfix unique à ces mots ou phrase par module. Ainsi par exemple le clef du mot “yes” sera “app:yes” et qui sera traduit par “oui” en français et par “yes” en anglais. Les entêtes des fichiers po de votre module doivent indiquer l'utilisation de l'encodage UTF-8. Les programmes d'éditions des .po prennent en compte ces entêtes pour sauvegarder le fichier avec ce bon encodage.

Pour détecter les nouveaux mots à traduire, il faut lancer la commande “make po”. Cela va ajouter les nouveaux mots à traduire et supprimer les éventuels mots qui ne sont plus référencé dans aucun des fichiers. Une fois les catalogues mis à jour vous devez ensuite effectuer les traductions. Ces fichiers doivent être éditer par des logiciels spécifiques afin d'éviter des erreurs de syntaxe. Il existe gtranslator sous gnome et kbabel sous kde.

Les traductions sont remplacés littéralement dans les fichiers produit par le serveur. Il faut être particulièrement vigilant lorsque ceux-ci sont entre des double quotes ou des simple quote. En effet un texte javascript: qui sera traduit "l'automne" pourra donner lieu à une erreur de syntaxe. Il est alors préférable d'utiliser les double quote alert("[TEXT:my:the automn]")qui sont moins utilisés dans des traductions.

3.1.1.3 Créer son espace de dévéloppement

Pour créer un module dynacase, le plus simple est d'utiliser le template disponible sur le site https://github.com/Anakeen/Dynacase-module-template.

```
$ git clone git@github.com:Anakeen/Dynacase-module-template.git template

Cloning into template...
remote: Counting objects: 89, done.
```

```
remote: Compressing objects: 100% (42/42), done.
remote: Total 89 (delta 43), reused 87 (delta 41)
Receiving objects: 100% (89/89), 25.00 KiB, done.
Resolving deltas: 100% (43/43), done.

$ cd template
$  ./create_application.bash -a TEST -m module-test -o ~/Bureau/Test
Application name: TEST
Module name: module-test
Output directory: /home/eric/Bureau/Test
```

Avec cette suite de commande nous avons un environnement de développement constitué avec un module "module-test" et une application TEST.

3.1.1.4 Paramétrage et développement Dynacase

3.1.1.4.1 Les familles

3.1.1.4.1.1 ODS

Les fichiers de définition des structures des familles sont définis dans des fichiers ods (openDocument spreadsheat). Ils sont enregistrés dans le répertoire `Families`. L'ensemble des familles utilisés conjointement dans un module doivent être rassemblé dans un même fichier ods. Chacune des familles occupant un onglet du fichier. Il est possible d'avoir plusieurs fichiers si vous avez des groupes de familles qui sont faiblement liés. Ces fichiers de structures doivent être référencé dans le fichier de post-traitement info.xml.in dans la section update :

```
<post-upgrade>
 <process command="programs/pre_migration MYMODULE" />

 <process command="./wsh.php --api=importDocuments
--file=./MYMODULE/my_family.ods">
 <label lang="en">Importing Testing Families</label>
 </process>
 <process command="programs/post_migration MYMODULE" />
 <process command="programs/update_catalog" />
 </post-upgrade>
```

Les fichiers ods étant publié à la racine de votre module, il faut alors indiquer le chemin d'accès aux fichiers ods du serveur (il n'y a pas de répertoire `Families` comme dans vos sources). Les noms des familles (les noms internes pas les libellés) choisis doivent être préfixé par un trigramme unique lié à votre module. Exemple : 'MYM_ANIMAL' ou 'MYM_CAKE'.

3.1.1.4.1.2 Méthode

Les fichiers “méthodes” permettant de modifier le comportement des documents ou d'ajouter des fonctionnalités sont préfixés par `Method.` suivi par le nom de votre famille. Exemple : `Method.mym_animal.php`. Les fichiers Méthode étant publiés dans un répertoire unique il est primordial d'avoir un nom non utilisé dans d'autres modules. Les fichiers méthodes se trouvent dans le même répertoire que les fichiers ods de structure de famille.

3.1.1.4.1.3 Vues et template

Les vues spécifiques[9] (.html/.xml/.ods) de documents doivent être enregistrées dans le répertoire Families/Views.

3.1.1.5 Publication

3.1.1.5.1 Mise au point

Afin de valider votre module et faire de la mise au point, il est nécessaire d'avoir accès à une installation de Dynacase. Pour la mise au point, vous pouvez partager les fichiers entre votre machine et la machine hébergeant dynacase et déployer directement sur la machine hébergeant dynacase.

3.1.1.5.1.1 Initialisation

à faire la première fois ou lorsque le répertoire de destination change

9) *Voir paragraphe Vue et édition particulière*

```
$ cd /home/eric/Bureau/Test
$ ./deploy-package.sh
/home/eric/Bureau/Test/deploy-package.config does not exists. it will be
created.
please specify wiff directory path [] /var/www/wiff/
[sudo] password for eric:
--- Contextes disponibles ---
     [0] Test 3.0
     [1] Test 3.1
     [2] Tmp
Dans quel contexte souhaitez-vous publier? [] 62
searching if module-test is installed (from module-test-0.1.0-0)
module-test not detected.
      INSTALLATION with /tmp/tmp.mK9y73lDgT/module-test-0.1.0-0.webinst

Processing module 'module-test' (0.1.0-0) for install.
Module 'module-test-0.1.0-0' is already downloaded in '/tmp/module-test-0.1.0-
0.webinst7y2xXh'.
Doing 'pre-install' of module 'module-test'.
Doing 'unpack' of module 'module-test'.
Unpacking module 'module-test'... [OK]
Doing 'post-install' of module 'module-test'.
Running 'Process programs/record_application TEST'... [OK]
Running 'Process programs/update_catalog'... [OK]
Doing 'purge-unreferenced-parameters-value' of module 'module-test'.
Purging unreferenced parameters value...[OK]

Done.

 the script ended with success.
```

Ce script réalise le déploiement de votre module à travers dynacase-control. Il réalise tous les traitements indiqués dans le fichier "info.xml.in", notamment les parties Installation et Upgrade.

3.1.1.5.1.2 Publication

Pour relancer un déploiement en gardant les derniers paramètres enregistré il suffit d'ajouter l'option -y" au programme.

```
$ ./deploy-package.sh -y
```

3.1.1.5.1.3 Mise à jour des catalogues de traduction

Lorsque vous avez mis à jour vos fichiers .po[10] avec la commande "make po". Il suffit de déployer le module. Le déploiement mettra à jour les catalogues.

3.1.1.5.1.4 Mise à jour des familles

Lors que vous effectuez une modification de la structure d'une famille (fichier ods) ou d'un fichier méthode, il est nécessaire de relancer le déploiement.

Si votre fichier de description est décrit dans la partie post-update de info.xml.in alors les modification seront prises en compte lors de déploiement.

Extrait du info.xml.in :

```
  <post-upgrade>
    <process command="programs/record_application @APPNAME@" />
    <process command="programs/update_catalog" />
    <process command="./wsh.php --api=importDocuments --reinitattr=no
--file=./@APPNAME@/test-families.ods">
```

10) *Voir paragraphe "Localisation"*

```
        <label lang="en">Update test families</label>
    </process>
  </post-upgrade>
```

3.1.1.5.2 Empaquetage

Pour installer le module en production ou dans sur un autre serveur, il est nécessaire de produire un fichier de type “webinst” pour que dynacase-control le prennent en compte. Pour fabriquer un fichier webinst, il faut au préalable s'assurer que tous ces fichiers sont enregistrés en configuration. Il est important de modifier la partie “changelog” du fichier info.xml.in pour notifier à l'installateur la nature des modifications apportées.

Ensuite il faut “tagguer” la configuration avec la version. Ensuite vous pouvez effectuer la commande “make webinst” pour générer le fichier webinst. Ce fichier est créé avec le nom du module donné lors de l'initialisation.

Sur votre serveur, à l'aide de dynacase-control, vous pouvez maintenant installer ou mettre à jour votre module en téléchargeant votre fichier webinst..

Si votre serveur dynacase-control est local, vous pouvez utiliser dynacase-control en mode de commande shell pour réaliser votre mise à jour.

```
make webinst
sudo /var/www/dynacase-control/wiff  context Dév module upgrade mymodule-
0.1.0.webinst
```

3.1.1.6 Créer son propre dépôt de paquets

Pour le déploiement de vos paquets sur plusieurs machines, vous pouvez créé votre propre dépôt de paquets, et déclarer celui-ci dans Dynacase-control des machines que vous gérez.

Un dépôt de paquet est donc un répertoire accessible par FTP ou HTTP, qui contient les paquets de vos modules et un fichier d'index `content.xml'. Cet index est généré à l'aide de l'outil mkwebinstrepo, qui va recenser les modules présents dans le répertoire.

3.1.1.6.1 Exemple de création de dépôt

- Créer le répertoire de votre dépôt

```
# mkdir /var/www/my-repo
```

- Copier les paquets de vos modules au format *webinst* dans ce répertoire

```
# cp *.webinst /var/www/my-repo/
```

- Générer l'index `content.xml' à l'aide de l'outil mkwebinstrepo :

```
# mkwebinstrepo /var/www/my-repo
# cat /var/www/my-repo/content.xml
```

3.1.1.6.2 Utilisation du dépôt

Lorsque votre dépôt est créé, vous pouvez déclarer celui-ci sur votre Dynacase-control :

Name	Description	Protocol	Host	Path
my-repo	Mon dépôt	http	www.example.net	/

3.1.2. Construction de modules pour Dynacase

Dans ce paragraphe nous allons détailler les éléments constitutifs d'un module Dynacase et mettre en œuvre ces éléments pour construire un module d'exemple qu'on nommera `dynacase-foo`.

Pour bien suivre cette présentation, il est souhaitable d'avoir bien en tête les notions d'Applications et d'Actions de

Dynacase et du fonctionnement général de ceux-ci.

3.1.3. Fichier ''info.xml''

Le fichier `info.xml` permet de décrire le module Dynacase en fournissant en particulier la version du module, une description, des dépendances avec d'autres modules Dynacase, un ensemble de paramètres, un descriptif des évolutions (changelog), et un ensemble d'action de pre-install, post-install, etc.

3.1.3.1 Exemple de fichier de info.xml

```
<?xml version="1.0"?>
<module name="dynacase-foo" version="1.2.3" release="1" [basecomponent="no"]>

  <description lang="en">dynacase foo</description>
  [<description lang="fr">dynacase toto</description>]

  <requires>
    [<installer version="1.0.0" comp="ge" />]
    <module name="dynacase-bar" />
    <module name="dynacase-baz" version="1.0.0" comp="ge" />
    [...]
  </requires>

  <parameters>
    <param name="foo_dir" label="Directory of FOO" type="text"
      default="/var/foo" needed="yes" />
    <param name="foo_color" label="Color of FOO" type="enum"
      values="red|green|blue" default="green" needed="no" />
  </parameters>

  <changelog>
    <version number="1.1.1" date="2009-01-01">
      <change title='First change' url='http://dev.dynacase.org/issues/111'>
 Comment for first change.
      </change>
      <change title='Second change'>
 Comment for second change.
      </change>
      <change title='Third change'>
      </change>
    </version>
    <version number="1.1.0" date="2008-12-15"/>
  </changelog>

  <pre-install>
    <check type="syscommand" command="zip" />
    <check type="phpfunction" function="pg_connect">
      <help>You might need to install a php-pg package.</help>
    </check>
    <check type="file" file="/var/foo" predicate="is_dir" />
  </pre-install>

  <post-install>
    <process command="programs/app_post FOO I" />
    <process command="programs/record_application FOO" />
    <process command="programs/app_post FOO U" />
    <process command="programs/update_catalog" />
  </post-install>

  <post-upgrade>
    <process command="programs/pre_migration FOO" />
    <process command="programs/app_post FOO U" />
    <process command="programs/record_application FOO" />
    <process command="programs/post_migration FOO" />
```

```
    <process command="programs/update_catalog" />
  </post-upgrade>

  <post-param>
  </post-param>

</module>
```

3.1.3.2 Description d'un module

3.1.3.2.1 Préambule Module

La racine du document `info.xml` est un tag `<module>` avec les attributs suivants :

- **name** : le nom du module
- **version** : la version du module (sous la forme `N.N.N`)
- **release** : le numéro de release de la version
- **basecomponent** : `yes` ou `no`, permet de spécifier si le module est un module de base, obligatoire a toute installation de Dynacase (optionnel, par défaut la valeur est `no`)
- **author** : l'auteur du module (ex. `John Doe john.doe@example.net`)(optionnel)
- **licence** : licence du module (optionnel)
- **vendor** : fournisseur du module (ex. `ACME Corp.`)(optionnel)

```
<?xml version="1.0"?>
<module name="dynacase-foo" version="1.2.3" release="rc1"
  author="John Doe &lt;john.doe@example.net&gt;" licence="GPLV2" vendor="ACME
Corp.">
  [...]
</module>
```

3.1.3.2.2 Description

Le module peut fournir une description textuelle pour expliciter le rôle du module. On pourra fournir des descriptions localisés en utilisant l'attribut `lang`.

Exemple :

```
  <description lang="fr">Ce module permet à Dynacase de se connecter à
FOO</description>
  <description lang="en">This module allows Dynacase to connect to
FOO</description>
```

3.1.3.2.3 Dépendances entre modules

Les dépendances permettent d'exprimer qu'un module requiert d'autres modules Dynacase avec eventuellement une contrainte sur la version de ceux-ci.

Le tag `<requires>` est composés d'éléments `<module>` qui ont les attributs suivants :

- **name** : le nom du module Dynacase requis
- **version** : la version que le module requis doit avoir
- **comp** : `gt` ou `ge`, opérateur de comparaison de version

Le module peut aussi exprimer une contrainte sur la version de l'installeur lui même à l'aide de l'élément `<installer>`. Dans ce cas, les attributs sont :

- **version** : la version de l'installeur que requiert le module
- **comp** : `gt` ou `ge`, opérateur de comparaison de version

Exemple :

```
<requires>
  <installer version="1.0" comp="ge" />
  <module name="dynacase-bar" version="2.0" comp="ge" />
  <module name="dynacase-baz" version="1.9" comp="gt" />
<requires>
```

Dans cet exemple, le module requiert un installeur avec une version >= 1.0, le module dynacase-bar en version >= 2.0 et le module dynacase-baz en version > 1.9.

3.1.3.2.4 Organisation des instructions d'importations des familles/documents

Les contrôles d'importation vérifient la validité des différentes structures et des relations entre les documents.

L'ordre que nous préconisons est le suivant :

Définition	Description de l'opération
Définition de l'application	Instructions de définition de l'application. Lors de l'installation[11] l'instruction se résume à <process command="programs/record_application @APPNAME@"/>, lors de l'upgrade[12] <process command="programs/pre_migration @APPNAME@"/><process command="programs/record_application @APPNAME@"/>
Groupes et utilisateurs	Document qui importe les groupes et les utilisateurs utilisés dans cette application
Droits	Document associant les droits spécifiques (ACL) à l'application aux groupes et aux utilisateurs importés ci-dessus
Famille(s) père(s)[13]	Si la famille en cours possède une famille père qui n'est pas déjà existante, vous devez l'importer avec l'ensemble de ses dépendances
Structure de la famille en cours	Uniquement la structure de la famille en terme d'attributs/paramètres
Famille de workflow	Définition de la famille de workflow associée à la famille en cours.
Paramétrage du workflow	Document de workflow et paramétrage associé à celui-ci (profil dédié, masque, timer,...) (si besoin, ie un workflow est associé à la famille et il n'a pas déjà été importé)
Paramétrage de la famille	Documents associés à la famille (profil de document, profil de famille, contrôle dédié, etc...)
Propriétés de la famille	Ensemble des propriétés de la familles (icône, label, référence du profil de famille, référence du profil de document par défaut, référence du contrôle de vue, référence du cycle de vie associé...)
Documents de la famille en cours	Éventuels documents de la famille en cours d'importation
Famille(s) suivante(s)[14]	Autres familles de l'application, éventuellement héritant de la famille ci-dessus
Fin de la définition de l'application	<process command="programs/post_migration @APPNAME@"/>
Instructions divers[15]	Par défaut : <process command="programs/update_catalog"/> Et toutes les instructions que vous avez pu ajouter lors de votre projet

Cela donne le XML suivant :

11) *balise <post-install>...</post-install>*
12) *balise <post-upgrade>...</post-upgrade>*
13) *La famille père doit être importée comme la famille présentée ici*
14) *Les familles suivantes sont importées de la même manière*
15) *Ces instructions n'ayant pas d'interdépendance avec le reste, elles peuvent être placée n'importe où*

```
<?xml version="1.0"?>

<!-- The @SOMETHING@ variables are completed by the autoconf mechanism -->
<module name="@PACKAGE@" disabled="no" version="@VERSION@" release="@RELEASE@">

   <!-- Label of the application displayed in dynacase control only -->
   <description>My application name</description>

  <!-- Module required with version for the installation/upgrade -->
   <requires>
       <module name="dynacase-platform" version="3.2.0" comp="ge"/>
   </requires>

  <!-- Install instruction -->
   <post-install>
       <process command="programs/record_application @APPNAME@"/>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/GROUPS.ods">
           <label lang="en">Importing GROUPS.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/RIGHT.ods">
           <label lang="en">Importing RIGHT.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/STRUCT_famille_A.ods">
           <label lang="en">Importing STRUCT_famille_A.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/PARAM_famille_A.ods">
           <label lang="en">Importing PARAM_famille_A.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/PROPERTY_famille_A.ods">
           <label lang="en">Importing PROPERTY_famille_A.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/STRUCT_famille_B.ods">
           <label lang="en">Importing STRUCT_famille_B.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/WFL_famille_B.ods">
           <label lang="en">Importing WFL_famille_B.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/PARAM_WFL_famille_B.ods">
           <label lang="en">Importing PARAM_WFL_famille_B.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/PARAM_famille_B.ods">
           <label lang="en">Importing PARAM_famille_B.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/PROPERTY_famille_B.ods">
           <label lang="en">Importing PROPERTY_famille_B.ods</label>
       </process>
       <process command="./wsh.php --api=importDocuments
--file=./@APPNAME@/DOC_famille_B.ods">
           <label lang="en">Importing DOC_famille_B.ods</label>
       </process>
```

```
        <process command="programs/post_migration @APPNAME@"/>
        <process command="programs/update_catalog"/>
    </post-install>

    <post-upgrade>
        <process command="programs/pre_migration @APPNAME@"/>
        <process command="programs/record_application @APPNAME@"/>

        <!-- Same content that the content between record and post migration of
the install part-->

        <process command="programs/post_migration @APPNAME@"/>
        <process command="programs/update_catalog"/>
    </post-upgrade>

</module>
```

3.1.3.2.5 Changelog

Le changelog permet d'indiquer les évolutions produites en rapport avec les versions du module. Ces informations sont contenues dans une balise `<changelog>` contenant des `<version>`.

Les éléments `<version>` ont les attributs suivants :

- **number** : le numéro de version
- **date** : la date de publication de la version

Les éléments `<version>` contiennent des éléments `<change>` qui décrivent chaque changement effectué. Les éléments `<change>` ont les attributs suivants :

- **title** : l'intitulé du changement
- **url** : une url en rapport avec le changement (dans l'interface dynacase-control, les chaînes de forme issues/111/ sont reconnues et donnent comme texte du lien affiché 'issues 111' ; sinon un texte par défaut est utilisé)

La valeur de l'élément `<change>` constitue une description.

Exemple :

```
  <changelog>
    <version number="1.1.1" date="2009-01-01">
      <change title='First change' url='http://dev.dynacase.org/issues/111'>
 Comment for first change.
      </change>
      <change title='Second change'>
 Comment for second change.
      </change>
      <change title='Third change'>
      </change>
    </version>
    <version number="1.1.0" date="2008-12-15"/>
  </changelog>
```

3.1.3.3 Paramètres

Un module peut demander lors de son installation (ou upgrade) l'entrée de certains paramètres.

Les paramètres nécessaires au module sont renseignés avec un élément `<parameters>` contenant des éléments `<param>`.

Les élements `<param>` ont les attributs suivants :

- **name** : le nom du paramètre
- **label** : le label textuel pour présenter le paramètre
- **type** : `text` ou `enum`, permet de spécifier le type de donnée attendu

- **default** : la valeur par défaut présenté à l'utilisateur lors de la saisie des paramètres.
- **needed** : yes ou no, permet de spécifier si la saisie du paramètre est obligatoire ou optionnelle.
- **values** : lorsque type="enum", l'attribut values permet de spécifier une list de choix finis à partir de laquelle l'utilisateur selectionnera une valeur

Exemple :

```
<parameters>
  <param name="foo_dir" label="Directory of FOO" type="text"
    default="/var/foo" needed="yes" />
  <param name="foo_color" label="Color of FOO" type="enum"
    values="red|green|blue" default="green" needed="no" />
</parameters>
```

3.1.3.4 Pre/post install/upgrade/etc.

Lors de l'installation, upgrade, un ensemble d'actions peuvent être effectués avant (pre) et après (post) l'opération suivant l'ordre suivant :

Chaque phase (pre-install, post-install, etc.) spécifie un ensemble de check ou process qui sont exécutés et qui retournent un status d'échec ou de réussite.

Une phase est validé lorsque tous ses sous-éléments check ou process ont retournés un statut de réussite.

Si une phase n'est pas validé, alors l'installation, ou l'upgrade, alors il est présenté à l'utilisateur les messages

d'erreurs rencontrés, et celui-ci peut rejouer la phase après avoir eventuellement corrigé le problème, ou bien il peut choisir d'ignorer les messages d'erreurs et poursuivre l'install/ugprade.

3.1.3.4.1 pre-install

Les éléments de pre-install s'exécutent avant l'installation des fichiers du module sur le système de fichier.

Les actions possible peuvent être des éléments `<check>`.

Chaque élément `<check>` peut fournir un élément `<help>` qui sera présenté à l'utilisateur lorsque l'action échoue.

Exemple :

```
  <pre-install>
    <check type="phpfunction" function="pspell_new">
      <help>Il faut peut-être installer php5-pspell avec apspell et aspell-
fr</help>
    </check>
    <check type="syscommand" command="convert" />
  </pre-install>
```

Les actions de pre-install serviront généralement à vérifier la présence de certains éléments et bloquer l'installation si ces éléments ne sont pas présents/corrects.

3.1.3.4.2 post-install

Les éléments de post-install s'exécutent après l'installation des fichiers du module sur le système de fichier.

Les actions possible peuvent être des éléments `<process>`.

Chaque élément `<process>` peut fournir un élément `<label>` qui sera présenté à l'utilisateur lorsque l'action sera exécuté.

Exemple :

```
  <post-install>
    <process command="programs/record_application FOO">
      <label lang="en">Record FOO application in database</label>
    </process>
    <process command="programs/update_catalog">
      <label lang="en">Generate localization catalog</label>
    </process>
  </post-install>
```

Les actions de post-install serviront généralement à configurer le module qui viens d'être isntallé. Une erreur dans la phase de post-install laissera les fichiers installés en place, mais le paquet sera marqué en erreur de post-install dans l'interface.

3.1.3.4.3 pre-upgrade

Les éléments de pre-upgrade s'exécutent avant l'installation des nouveaux fichiers du module sur le système de fichier.

Les actions possible peuvent être des éléments `<check>`.

Chaque élément `<check>` peut fournir un élément `<help>` qui sera présenté à l'utilisateur lorsque l'action échoue.

Exemple :

```
  <pre-upgrade>
    <check type="phpfunction" function="pspell_new">
      <help>Il faut peut-être installer php5-pspell avec apspell et aspell-
fr</help>
    </check>
    <check type="syscommand" command="convert" />
  </pre-upgrade>
```

Les actions de pre-upgrade serviront généralement à vérifier la présence de certains éléments et bloquer l'upgrade si ces éléments ne sont pas présents/corrects.

3.1.3.4.4 post-upgrade

Les éléments de post-upgrade s'exécutent après l'installation des nouveaux fichiers du module sur le système de fichier.

Les actions possible peuvent être des éléments `<process>`.

Chaque élément `<process>` peut fournir un élément `<label>` qui sera présenté à l'utilisateur lorsque l'action sera exécuté.

Exemple :

```
<post-upgrade>
  <process command="programs/pre_migration FOO">
    <label lang="en">Pre-migration scripts</label>
  </process>
  <process command="programs/record_application FOO">
    <label lang="en">Update application record in database</label>
  </process>
  <process command="programs/post_migration FOO">
    <label lang="en">Post-migration scripts</label>
  </process>
  <process command="programs/update_catalog">
    <label lang="en">Re-generate localization catalog</label>
  </process>
</post-install>
```

Les actions de post-upgrade serviront généralement à configurer le module qui viens d'être isntallé, lancer les scripts de migration, etc. Une erreur dans la phase de post-upgrade laissera les fichiers installés en place, mais le paquet sera marqué en erreur de post-upgrade dans l'interface.

3.1.3.5 Les checks

Les éléments `check` permettent d'executer des actions pour vérifier la présence de certains éléments.

3.1.3.5.1 phpfunction

Le check de type `phpfunction` permet de vérifier la présence d'une fonction PHP.

Le nom de la fonction testé est spécifié avec l'attribut `function`.

Exemple :

```
<check type="phpfunction" function="pg_connect" />
```

3.1.3.5.2 syscommand

Le check de type `syscommand` permet de vérifier la présence d'une commande disponible sur le système.

Le nom de la command testé est spécifié avec l'attribut `command`

Exemple :

```
<check type="syscommand" command="convert" />
```

3.1.3.5.3 phpclass

Le check de type `phpclass` permet de vérifier la présence d'une classe objet PHP.

Le nom de la classe PHP est fournis avec les attributs suivants :

- **include** : le nom du fichier pour inclure la définition de la classe
- **class** : le nom de la classe

Exemple :

```
<check type="phpclass" include="Net/SMTP.php" class="Net_SMTP" />
```

3.1.3.5.4 apachemodule

LE check de type `apachemodule` permet de vérifier qu'un module Apache particulier est activé et chargé par celui-ci.

Le nom du module est spécifié par l'attribut `module`.

Exemple :

```
<check type="apachemodule" module="mod_expires" />
```

3.1.3.6 Les process

Les éléments process servent à exécuter les actions permettant d'effectuer les opérations nécessaires au fonctionnement du module suite à son installation.

3.1.3.6.1 programs/app_post

Prototype :

- `programs/app_post <APPNAME> I|U`

Utilisable dans les phases :

- `post-install`
- `post-upgrade`

Conditions d'utilisation :

- Dans la phase `post-install`, le programme doit être exécuté avec le programme `record_application`
- Dans la phase `post-upgrade`, le programme doit être exécuté après `pre_migration` et avant le programme `record_application`

Le programme `app_post` permet de lancer un script `_post` pour initialiser une application Dynacase.

Les arguments sont :

- Le nom de l'application (le script exécuté sera alors <APPNAME>_post)
- `I` | `U` : Le nom de la phase d'initialisation a exécuter (`I` pour install, `U` pour upgrade)

Exemple :

```
<post-install>
  <process command="programs/post_app WEBDESK I" />
  [...]
</post-install>

<post-upgrade>
  [...]
  <process command="programs/post_app WEBDESK U" />
  [...]
</post-upgrade>
```

3.1.3.6.2 programs/record_application

Prototype :

- `programs/record_application <APPNAME>`

Utilisable dans les phases :

- `post-install`
- `post-upgrade`

Conditions d'utilisation :

- Le programme doit être exécuté après le programme `app_post`

Le programme `record_application` est utilisé pour enregistrer, ou mettre à jour, la définition de l'application en base de données.

La ligne de commande est spécifié par l'attribut `command`.

`record_application` prend en argument le nom de l'application à enregistrer.

Exemple :

```
<process command="programs/record_application FOO" />
```

3.1.3.6.3 programs/update_catalog

Prototype :

- `programs/update_catalog`

Utilisable dans les phases :

- `post-install`
- `post-upgrade`

Conditions d'utilisation :

- Le programme doit être exécuté à la fin de la phase

Le programme `update_catalog` est utilisé pour ré-générer le catalogue des messages de localisation.

Exemple :

```
<process command="programs/update_catalog" />
```

3.1.3.6.4 programs/pre_migration

Prototype :

- `programs/pre_migration`

Utilisable dans les phases :

- `post-upgrade`

Conditions d'utilisation :

- Le programme doit être exécuté au début de la phase, avant le programme `app_post`

Le programme `pre_migration` est utilisé pour exécuter les scripts de pre-migration d'un module lors de sa mise-à-jour.

Exemple :

```
<process command="programs/pre_migration" />
```

3.1.3.6.5 programs/post_migration

Prototype :

- `programs/post_migration`

Utilisable dans les phases :

- `post-upgrade`

Conditions d'utilisation :

- Le programme doit être exécuté après le programme `record_application`, et avant le `update_catalog`

Le programme `post_migration` est utilisé pour exécuter les scripts de post-migration d'un module lors de sa mise-

à-jour.

Exemple :

```
  <process command="programs/post_migration" />
```

3.1.3.6.6 ./wsh.php

Prototype :

- `./wsh.php`

Utilisable dans les phases :

- `post-install`
- `post-upgrade`

Conditions d'utilisation :

- Le programme peut être utilisé à n'importe quel moment en fonction des besoins

Le programme `./wsh.php` est utilisé pour exécuter des méthodes sur des classes documentaires et exécuter des API Dynacase.

Exemple :

```
  <process command="./wsh.php --api=refreshDocuments --method=postModify
--famid=FOO" />
```

3.1.3.6.7 Programmes personnalisés

Vous avez la possibilité d'écrire vos propres programmes de post-install, post-upgrade, etc. afin d'effectuer des opérations spécifiques à votre module.

Ces programmes seront généralement développés soit en shell Bash soit en PHP. Ils seront disponible après la phase de décompression de votre paquet, dans le répertoire que vous aurez spécifié à l'empaquetage.

Le programme est exécuté dans le répertoire racine de l'installeur DYNACASE-CONTROL, et les variables d'environnement suivantes sont accessible depuis le script :

- **WIFF_ROOT** : Le chemin du répertoire ou est installé DYNACASE-CONTROL (c'est donc aussi le répertoire courant (`$CWD`) dans lequel sera exécuté votre programmes)
- **WIFF_CONTEXT_ROOT** : Le chemin du répertoire du contexte sur lequel est effectué l'opération.
- **WIFF_CONTEXT_NAME** : Le nom du contexte sur lequel est effectué l'opération.

3.1.3.6.7.1 Ecrire un programme personnalisé en shell Bash

Exemple :

```
#!/bin/bash

set -e

# -- Récupérer la valeur du paramètre `foo_dir' spécifié par l'utilisateur
PARAM_FOO_DIR=`"$WIFF_ROOT"/wiff --getValue=foo_dir`

# -- Créer le répertoire s'il n'existe pas
if [ ! -d "$PARAM_FOO_DIR" ];
  mkdir "$PARAM_FOO_DIR"
fi

# -- Ajouter le nom de ce répertoire dans le fichier
# -- `foo_dir.list' dans le sous-répertoire de mon module `FOO'
echo "$PARAM_FOO_DIR" >> "$WIFF_CONTEXT_ROOT"/FOO/foo_dir.list
```

3.1.3.6.7.2 Ecrire un programme personnalisé en PHP

Note : Le programme PHP a aussi accès aux variables d'environnement, comme le script Bash, mais le chemin d'include doit être construit en fonction de vos besoins.

Exemple :

```
#!/usr/bin/env php
<?php

$WIFF_ROOT=getenv('WIFF_ROOT');
if( $WIFF_ROOT === false ) {
  print "WIFF_ROOT environment variable is not set!";
  exit( 1 );
}

$WIFF_CONTEXT_ROOT=getenv('WIFF_CONTEXT_ROOT');
if( $WIFF_CONTEXT_ROOT === false ) {
  print "WIFF_CONTEXT_ROOT environment variable is not set!";
  exit( 1 );
}


# -- Si je dois accéder aux fichier d'include de Dynacase
# -- j'ajoute les répertoire d'include de Dynacase
# -- dans mon include_path PHP
set_include_path( get_include_path().PATH_SEPARATOR."$WIFF_ROOT/include".PATH_SE
PARATOR$WIFF_CONTEXT_ROOT );

set_include_path(join(PATH_SEPARATOR, array(
  get_include_path(),
  "$WIFF_ROOT/include",
  "$WIFF_CONTEXT_ROOT"
)));

# -- A présent, je peux inclure les librairies de l'installeur
require("lib/Lib.Cli.php");
# -- ... et les librairies Dynacase
require("WHAT/Lib.Common.php");

$param_foo_dir = wiff_getParamValue('foo_dir');
if( ! is_dir($param_foo_dir) ) {
  $ret = mkdir($param_foo_dir);
  if( $ret === false ) {
    print sprintf("Error: could not create directory '%s'!", $param_foo_dir);
    exit( 1 );
  }
}

?>

[A compléter]
```

3.1.4. Générer le fichier .webinst

Pour fabriquer votre fichier webinst il faut disposer d'un fichier configure.in d'un fichier Makefile.in comme ceux-ci.

Le configure.in

```
#   Autoconf script for libphp
#
#
AC_REVISION($Id: configure.in $)
```

```
dnl
dnl Process this file with autoconf to produce a configure script.
dnl
AC_PREREQ(2.13)
AC_INIT(./Makefile.in)
AC_SUBST(VERSION)
VERSION=`cat VERSION`
AC_SUBST(RELEASE)
RELEASE=`cat RELEASE`
AC_SUBST(PACKAGE)
PACKAGE=dynacase-myapplication
AC_SUBST(APPNAME)
APPNAME=MYAPP

ac_default_prefix=/usr/share/what
AC_SUBST(PUBRULE)
PUBRULE=
AC_ARG_WITH(pubrule, [  --with-pubrule=dir Path to PubRule], PUBRULE=$withval)
if test "x$PUBRULE" != "x"; then
  PUBRULEDIR=$PUBRULE
else
  if test "x$PUBRULEDIR" == "x"; then
     AC_CHECK_FILE($HOME/anakeen/devtools/PubRule,
PUBRULEDIR=$HOME/anakeen/devtools/)
     if test "x$PUBRULEDIR" = "x"; then
          PUBRULEDIR=`pwd`
     fi
  fi
fi
AC_CHECK_FILE($PUBRULEDIR/PubRule, PUBRULE=$PUBRULEDIR)
if test "x$PUBRULE" = "x"; then
  AC_MSG_ERROR([Could not find PubRule])
fi
AC_MSG_NOTICE([PubRule located at $PUBRULE])



AC_OUTPUT(Makefile  info.xml )
```

Le Makefile :

```
# ============================================
# $Id: Makefile.in $
# ============================================
PACKAGE = @PACKAGE@
VERSION = @VERSION@
RELEASE = @RELEASE@
utildir=@PUBRULE@
pubdir = @prefix@
appname = offline
srcdir = @srcdir@
SUBDIR= Action

TOP_MODULES =

TAR = gtar
GZIP_ENV = --best

export targetdir PACKAGE

pages_not_xml = info.xml
```

```
include $(utildir)/PubRule

DISTFILES += $(SUBDIR) \
            RELEASE VERSION
```

le makefile inclut le fichier Pubrule. Il contient un ensemble de règles pour publier le code et fabriquer les paquets.

Vous devez inclure ce fichier dans le répertoire de vos sources. Ensuite vous tapez :

```
autoconf
./configure
make webinst
```

cela va générer le fichier webinst correspondant à votre module.

3.2 Utilisation de l'API

Ce chapitre a pour but de montrer par l'exemple l'utilisation des méthodes et fonctions les plus courantes de l'API dynacase :

3.2.1. Présentation de l'utilisation de l'API

Ce chapitre a pour but de montrer par l'exemple l'utilisation des méthodes et fonctions les plus courantes de l'API dynacase.

3.2.2. Création d'un script utilisable par wsh

Les exemples présentés dans ce chapitre peuvent être exécuté par le programme wsh. Ce programme, exécutable dans un contexte dynacase, permet de réaliser des traitements sur les documents dans un contexte donné.

wsh exécute des programmes placés dans le répertoire "API" de votre serveur.

Un programme wsh à a sa disposition la variable $action permettant d'accéder au contexte d'utilisation. Ce programme peut aussi utiliser la class apiUsage afin de contrôler ces paramètre d'entrée.

Pour tester ces exemples, il faut copier/coller le code PHP dans un fichier (ex: test.php) et l'enregistrer dans le dossier ”<context_root>/API”.

Cette commande donne la liste des fonctions de l'API disponible :

```
./wsh.php --listapi
```

Cette commande permet d'exécuter une fonction de l'API comme votre fichier de test :

```
./wsh.php --api=test
```

Pour plus d'informations sur l'utilisation des fonctions WSH, il faut se reporter au chapitre wsh.

3.2.3. Manipulation de documents (getParam, new Doc et new_Doc)

Chaque document est identifié par un numéro unique : son identificateur. Cet identificateur est différent pour chaque révision d'un même document. Cet identificateur permet d'accéder à l'objet document correspondant. Le développeur n'a pas besoin de connaître la famille de document pour extraire les caractéristiques propres de ce document. La fonction `new_Doc()` retourne l'objet document dans la classe décrite pour la famille du document précisé.

```
include_once("FDL/Class.Doc.php")
$docid = 38922;
$dbaccess = $action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess, $docid);
print_r($doc);
```

Le résultat sera le suivant :

```
_IUSER Object
(
 [dbtable] => doc128
 [dbseq] => seq_doc128
 [fromid] => 128
 [serveur] =>
 [port] =>
 [racine] =>
 [rootdn] =>
 [rootpw] =>
 [dbaccess] => user=anakeen port=5432 dbname=freedom
 [orginit] =>
 [action] =>
 [defaultabstract] => USERCARD:VIEWABSTRACTCARD
 [defaultedit] => USERCARD:EDITUSERCARD
 [defDoctype] => F
 [defClassname] => DocFile
 [fields] => Array
 (
  [0] => id
  [1] => owner
  [2] => title
  [3] => revision
  [4] => initid
  [5] => fromid
  [6] => doctype
  [7] => locked
  [8] => icon
  [9] => lmodify
  [10] => profid
  [11] => usefor
  [12] => revdate
  [13] => comment
  [14] => classname
  [15] => state
  [16] => wid
  [17] => values
  [18] => attrids
  [19] => postitid
  [20] => us_matricule
  [21] => usm_hobby
  [22] => usm_ecm
...
  [52] => us_idgroup
  [53] => us_service
  [54] => us_idservice
  )
 }
```

L'objet retourné est de la classe doc128 (un utilisateur). Il contient tous les attributs et méthodes de cette classe.

Cela aurait donné le même résultat que si on avait écrit le code suivant :

```
include_once("FDLGEN/Class.**Doc128**.php")

$docid = 38922;
$dbaccess = $action->getParam("FREEDOM_DB");
```

```
$doc = new Doc128($dbaccess, $docid);
print_r($doc);
```

L'utilisation directe des classes spécialisée est à proscrire car elle présuppose de l'appartenance à la classe. De plus l'appel direct court-circuite les optimisations d'accès placées dans la fonction `new_Doc()`. Le constructeur `new Doc()` ne doit pas être utilisé pour instancier des objets documentaire car ces objets n'auraient pas les attributs et méthodes spécifiques qui leur sont dus.

Avant de manipuler un document, il faut s'assurer qu'il existe et qu'il vit. Pour cela, on utilise les méthodes `Doc::isAlive()` ou `DbObj::isAffected()`.

```
include_once("FDL/Class.Doc.php")
$usage = new ApiUsage();
$usage->setText("Test documents ");
$docid = $usage->addNeeded("docid", "the document id to test");
$usage->verify();

$dbaccess = $action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess, $docid);
if ($doc->isAlive()) {
  // le document est valide//
 } else if ($doc->isAffected()) {
  // le document existe mais n'est pas vivant => c'est un zombie//
 } else {
  // le document 38922 n'existe pas//
 }
```

La class apiUsage[16] permet de contrôler les arguments d'un programme wsh.

Note sur getParam :

La méthode getParam permet de récupérer la valeur d'un paramètre d'une application. La déclaration de ce paramètre se fait dans le fichier MONAPPLI_init.php.ini. Exemple :

```
$N = getParam("Name_of_the_parameter", "0.5");
```

$N prendra pour valeur :

- celle du paramètre nommé “Name_of_the_parameter”
- ou 0.5 si l'accès à la valeur de ce paramètre échoue.

3.2.4. Consultation de documents (getValue, getTValue et getTDoc)

Une fois assuré de la validité du document, la consultation des données du document peut avoir lieu. Les informations portées par le document appartiennent à 2 catégories :

17. les propriétés : présentent pour tout document [17]
18. les attributs : définis pour une classe de document et hérités par les classes dérivées.

3.2.4.1 Les propriétés remarquables

id	Identifiant unique du document
owner	Identifiant système de l'utilisateur
title	Titre du document
revision	numéro de révision : 0 est la première révision

16) *Se reporter au site http://api.dynacase.org pour plus de précisions*
17) *ce sont les attributs de la classe Doc*

version	libellé de la version : vide par défaut
initid	Identifiant initial = identifiant de la révision 0
fromid	Identifiant de la famille d'appartenance
doctype	Type de document 'F' (Normal), 'D' Dossier, 'S' Recherche, 'P' Profil, 'T' Temporaire, 'Z' Zombie, 'W' Cycle de vie, 'C' Famille (C pour Class)
locked	Chiffre négatif = Identifiant de l'utilisateur ayant verrouillé le document (Le verrou est supprimé automatiquement à la fermeture de Firefox et par un script lancé toute les nuits). Chiffre positif = Document verrouillé manuellement par l'utilisateur. 0 = Pas de verrou. -1 = Document révisé (figé).
allocated	Identifiant de l'utilisateur alloué au document
icon	référence au fichier icon : référence locale ou référence VAULT
lmodify	Y = Le document a été modifié depuis la dernière révision. L = Révision N-1. N = Document révisé mais non modifié
profid	Identifiant du profil de document. Chiffre négatif = Profil non activé. Si identique à l'id du document = contrôle dédié. Vide = Pas de profil. Chiffre positif = profil actif
attrids	Concaténation des noms attributs valués
values	Concaténation des valeurs des attributs valués (différent de vide)
usefor	type d'utilisation (obsolète)
cdate	date de création de la révision. La date de création du document est celle de la révision 0
adate	date de dernier accès
revdate	date de dernière modification ou date de révision dans le cas d'un document révisé
comment	commentaire de révision (obsolète) ⇒ voir Objet DocHisto
classname	Classe du document - Utilisé seulement pour les cycles de vie
state	état du document ou référence à un identifiant de document 'état libre'
wid	identifiant du document cycle de vie
postitid	identifiant du document 'note'
cvid	identifiant du document contrôle de vue
name	identifiant *logique* du document
dprofid	Identifiant utilisé pour les profils dynamiques. Si le document à un profid alors le profid est égale à l'id.
atags	balises applicatives (Tags positionnés sur le document)
prelid	identifiant du document (dossier) de relation primaire (généralement le dossier parent)
confidential	=1 si document confidentiel (restriction supplémentaire)
ldapdn	chemin LDAP dans le cas d'une copie sur un serveur LDAP
fulltext	vectorisation des attributs
svalues	concaténation des valeurs des attributs recherchable (hors image/password/idoc)

3.2.4.2 Accès aux propriétés (getValue)

L'accès aux propriétés se fait par simple accès à l'objet. L'accès aux valeurs des attributs se fait principalement par la méthode `Doc::getValue($attrid, $default="")`. *L'identificateur de l'attribut peut être indiqué en majuscule ou en en minuscule.*

```
include_once("FDL/Class.Doc.php");

$docid = 38922;
$dbaccess = GetParam("FREEDOM_DB");
$doc = new_Doc($dbaccess, $docid);
if ($doc->isAlive()) {
```

```
  // le document est valide//
  $id = $doc->id;                                  // c'est 38922 ($docid)//
  $rev = $doc->getProperty('revision');                            // le numéro
de révision du document//
  $society = $doc->getValue("US_SOCIETY");         // la société du document
personne intranet//
  $usermail = $doc->getValue("US_MAIL","nomail");  // l'email du document
personne intranet//
  if ($usermail == "nomail") {
    // cet utilisateur n'a pas d'adresse email//
  }
 }
```

3.2.4.3 Récupérer les différentes valeurs d'un attribut multiple (getTValue)

La méthode `Doc::getValue()` retourne toujours une chaîne de caractères même pour les attributs multiples (ceux qui sont dans des tableaux) ou ceux qui ont comme type `*list` tels que `integerlist` ou `textlist`. Pour récupérer les différentes valeurs d'un attribut multiple, on utilise la méthode `Doc::getTValue()`.

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  print "[".($doc->getValue("US_IDGROUP")."]\n");
  print_r ($doc->getTValue("US_IDGROUP")); // affichage du tableau
 }
```

Le résultat de ce programme donne :

```
[38981
 38995
 38985]
Array(
      [0] => 38981
      [1] => 38995
      [2] => 38985
      )
```

Le premier affichage (getValue) donne la valeur sous forme de chaîne de caractère. Les différents éléments sont séparés par un retour chariot. Le deuxième affichage montre la décomposition de la valeur multiple en 3 valeurs distinctes. Ceci permet de contrôler chaque valeur plus facilement.

Dans l'exemple ci-dessous, on affiche les noms des groupes d'appartenance de la personne référencée par le numéro 38922.

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");

$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  $tidgroup = $doc->getTValue("US_IDGROUP");
  foreach($tidgroup as $k=>$v) {
    $gdoc = new_Doc($dbaccess, $v);                // constitution du document
groupe de personnes
    if ($gdoc->isAlive()) print $gdoc->getTitle()."\n"; // affichage des titres
des groupes
  }
```

```
 }
```

3.2.4.4 Récupérer un array complet ou une ligne de l'array (getAValues)

getAValues permet de retourner un tableau complet de valeurs ou une rangée si l'argument index est indiqué :

```
$t=$this->getAValues("EN_T_ANIMAUX");
```

3.2.4.5 Accès à l'ensemble des attributs (getValues)

Pour avoir l'ensemble des valeurs des attributs d'un document, on utilise la méthode `Doc::getValues()`. Cette méthode retourne les valeurs (sous forme de chaîne de caractères) de tous les attributs du documents. Les index du tableau retourné sont les identificateurs des attributs.

```
include_once("FDL/Class.Doc.php");

$doc = new_Doc("",38922);
if ($doc->isAlive()) {
  $tvalues = $doc->getValues();
  foreach($tvalues as $k=>$v) {
    if ($v != "") print "$k : $v\n"; // affiche les valeurs non nulles
  }
 }
```

Cet exemple retourne toutes les valeurs non nulles des attributs. Le résultat produit est le suivant :

```
us_lname : martin
us_fname : jean
us_mail : jean.martin@i-cesam.com
us_login : jean.martin
us_passwd : Vto9bHWMZGSPw
us_whatid : 54
us_group : default group
Groupe Incident
Groupe Technique
us_idgroup : 38981
38995
38985
```

3.2.4.6 Accès aux valeurs des documents (getTdoc)

Afin d'optimiser l'accès aux documents, on peut utiliser la fonction `getTdoc()` pour récupérer l'ensemble des valeurs d'un document. Cette fonction ne retourne pas un objet documentaire mais simplement un tableau de valeur des propriétés et attributs.

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$tdoc = getTDoc($dbaccess,38922); // tableau de valeurs//
if ($tdoc) {
  foreach($tdoc as $k=>$v) {
    if ($v != "") print "$k : $v\n"; // affiche les valeurs non nulles
  }
}
```

3.2.5. Modification de documents (setValue, getValues et deleteValue)

La méthode Doc::setValue($attrid, $val) sert à modifier les valeurs d'un document. Cette méthode de mets pas à jour directement en base mais uniquement dans l'objet document. La modification effective en base de données est faite par l'appel à la méthode DbObj::store().

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  $doc->setValue("US_ROLE","responsable développement logiciel");
  $err = $doc->store(); // affectation des valeurs dans la base de données//
  if ($err != "") {
    print $err; // l'utilisateur n'a probablement pas le droit de modifier ce
document//
  } else {
    $tvalues = $doc->getValues();
    foreach($tvalues as $k=>$v) {
      if ($v != "") print "$k : $v\n";
    }
  }
 }
```

Le résultat indique que l'attribut US_ROLE a maintenant une nouvelle valeur :

```
us_lname : martin
us_fname : jean
us_mail : jean.martin@somewhere.com
us_login : jean.martin
us_passwd : Vto9bHWMZGSPw
us_whatid : 54
us_group : default group
Groupe Incident
Groupe Technique
us_idgroup : 38981
38995
38985
us_role : responsable développement logiciel
```

Si la valeur passée en paramètre de setValue est une chaîne vide, la modification est ignorée. Pour supprimer une valeur, il faut utiliser la méthode deleteValue.

```
include_once("FDL/Class.Doc.php");
$dbaccess=$action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  $doc->deleteValue("US_ROLE"); //// suppression du rôle//
  $err = $doc->store(); //// affectation des valeurs dans la base de données//
  if ($err != "") {
    print $err; //// l'utilisateur n'a probablement pas le droit de modifier ce
document//
  }
 }
```

3.2.6. Création de documents (createDoc)

La création d'un document vide (sans valeur) se fait par l'appel à la fonction `createDoc()`. Cette fonction permet d'initialiser un objet document en précisant sa famille. Le document retourné contient les valeurs par défaut de la famille ainsi que le profil d'accès par défaut.

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$doc = createDoc($dbaccess,"USER");
if ($doc === false) {
  // l'utilisateur n'est pas habilité à créer ce type de document//
 } else {
  // l'objet doc120 (USER) est créé//
  $err=$doc->setValue("us_fname","Paul");
  if (! $err) $err = $doc->setValue("us_lname","Dujardin");
  if (! $err) $err = $doc->store();
  if ($err != "") {
    // erreur à la création en base//
    print $err;
  } else {
    // nouveau document en base//
    print "nouvel identifiant :".$doc->id. "\nobjet :".get_class($doc).
      "\ntitre :".$doc->getTitle();
  }
 }
```

Ce programme crée un nouveau document vide de la famille USER (personne). Le résultat de cet exemple est le suivant :

```
nouvel identifiant :39692

objet :_USER

titre :Dujardin 39692
```

La méthode doc::store() vérifie les contraintes[18] d'attributs. Si les contraintes ne sont pas respectées le document n'est pas créé. Toutefois il est possible de ne pas tenir compte des contraintes en utilisant le paramétrage suivant :

```
$du=createDoc("","MYTEST");
  if ($du) {
    $du->setValue("tst_ref","45X67");
    $skipConstraint=false;
    $err=$du->store($info, $skipConstraint);
    print_r($info);
}
```

Ici nous avons posé une contrainte d'unicité sur l'attribut "tst_ref". Supposons qu'un document de même référence existe déjà.

Résultat de $info sans contrainte :

```
stdClass Object
(
    [constraint] =>
    [refresh] => problème de référence
    [postModify] => cette référence existe
    [error] =>
)
```

Dans ce cas le document est bien créé malgré les contraintes

Résultat de $info avec contrainte

18) *Voir chapitre "contraintes"*

```
stdClass Object
(
    [constraint] => Array
        (
            [tst_ref] => Array
                (
                    [id] => tst_ref
                    [label] => référence
                    [pid] => tst_fr_ident
                    [sug] => Array
                        (
                        )
                    [err] => référence déjà utilisé
                )
        )
    [error] => référence déjà utilisé
)
```

Dans ce cas le document n'est pas créé.

La création de document peut aussi ce faire pour duplication d'un autre document. Pour cela on utilise la méthode `Doc::copy()`.

```
include_once("FDL/Class.Doc.php");
$dbaccess=$action->getParam("FREEDOM_DB");

$doc = new_Doc($dbaccess,38922);

if ($doc->isAlive()) {
  $copy = $doc->copy();
  if (! is_object($copy)) {
    // erreur pendant la copie//
    print $copy; //// message d'erreur//
  } else {
    // nouveau document en base//
    print "nouvel identifiant :".$copy->id. "\nobjet :".get_class($copy).
      "\ntitre :".$copy->getTitle();
  }
 }
}
```

Le résultat est que l'on obtient une réplique du document 38922. Il est identique à part son identifiant qui est 39693 dans l'exemple.

```
nouvel identifiant :39693
objet :doc128
titre :martin jean
```

La création d'un document **temporaire** se fait par la fonction `createTmpDoc()`. Cela crée un document temporaire et il n'est pas soumis au contrôle de l'acl `create` au contraire de la fonction `createDoc()`. Le document temporaire s'il est mis en base (appel de `Doc::store()`) sera supprimé la nuit par le robot (cron) de nettoyage.

3.2.7. Placement de documents dans un dossier (addFile et delFile)

Les documents que l'on créé par `createDoc()` ou `Doc::copy()` ne sont rattachés à aucun dossier[19]. Si on veut les référencer[20] dans des dossiers précis il faut le demander. Ces ajouts ou suppressions de référence à des documents dans les dossiers se font à l'aide des méthodes `Dir::AddFile()` et `Dir::DelFile()`.

19) *ce n'est pas génant en soi... dynacase tout est dans tout et réciproquement*
20) *un document peut être référencé plusieurs fois, n'oubliez pas*

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  $fld = new_Doc($dbaccess,38618);
  if ($fld->isAlive()) {
    if ($fld->doctype != "D") {
      // ce n'est pas un dossier//
      print $fld->id." n'est pas un dossier";
    } else {
      $err = $fld->addFile($doc->id);
      if ($err != "") {
 // le document n'a pas pu être inséré//
 print $err;
      } else {
 //// le document est maintenant inséré//
      }
    }
  }
 }
```

Pour l'exemple le document n°38618 est un dossier. L'exécution du programme produit le résultat suivant :

LOG::(I)::Maître Chewie [1] - Ajout martin jean dans dossier What Master

La méthode `Dir::AddFile()` mets un message de log si l'insertion réussi. Si on ré-éxécute le programme nous obtenons :

déjà existant : pas ajouté

Dans le deuxième cas, la méthode Dir::AddFile() retourne un message d'erreur car on ne peut référencer deux fois le même document dans un dossier.

Pour supprimer la référence à un document d'un dossier, il suffit d'appeler la méthode `Dir::DelFile()`.

```
include_once("FDL/Class.Doc.php");

$dbaccess=$action->getParam("FREEDOM_DB");
$doc = new_Doc($dbaccess,38922);
if ($doc->isAlive()) {
  $fld = new_Doc($dbaccess,38618);
  if ($fld->isAlive()) {
    if ($fld->doctype != "D") {
      // ce n'est pas un dossier//
      print $fld->id." n'est pas un dossier";
    } else {
      $err = $fld->delFile($doc->id);
      if ($err != "") {
 // le référence au document n'a pas pu être supprimée//
 print $err;
      } else {
 // la référence au document est maintenant supprimée//
      }
    }
  }
 }
```

L'exécution du programme produit le résultat suivant :

LOG::(I)::Maître Chewie [1] - Suppression n°38922 du dossier What Master

Si on ré-éxécute le programme nous obtenons :

échec de la suppression de la référence (document 38922, dossier 38618)

La méthode `Dir::DelFile()` indique qu'on a essayé de supprimer la référence à un document qui n'est pas dans le dossier.

3.3 Recherche de documents

Utilisation de la classe SearchDoc[21]. Cette classe est a utiliser en remplacement de la fonction getChildDoc qui est plus délicate à appeler à cause du nombre croissant de paramètres.

3.3.1. Retour de valeurs d'attributs

Ce programme permet d'écrire tous les titres des documents visible [22] de la famille personne intranet. Ici, on n'a utilisé que le critère d'appartenance à une famille. Le retour de méthode DocSearch::Search() est un **tableau de valeurs** (tableau de tableaux).

Avec une boucle `foreach`.

```
include_once("FDL/Class.SearchDoc.php");

$db=GetParam("FREEDOM_DB");
$s=new SearchDoc($db,"IUSER");
$t=$s->search();

foreach ($t as $k=>$v) {
  print "$k)".$v["title"]."(".getv($v,"us_mail","nomail").")\n";
 }
```

Avec une boucle while:

```
include_once("FDL/Class.SearchDoc.php");

$db=GetParam("FREEDOM_DB");
$s=new SearchDoc($db,"IUSER");
$s->search();

$k=0;
while ($v=$s->nextDoc()) {
  print "$k)".$v["title"]."(".getv($v,"us_mail","nomail").")\n";
  $k++;
 }
```

3.3.2. Retour d'objets documentaires

Ce programme permet d'écrire tous les titres des documents visible 123 de la famille personne intranet. Ici, on n'a utilisé que le critère d'appartenance à une famille.

L'appel à la méthode `DocSearch::setObjectReturn()` indique que le retour de méthode DocSearch::Search() est un **tableau d'objets documentaires**. Dans ce cas, il est possible d'appliquer les méthodes des objets sur les retours (exemple `Doc::getValue`).

```
include_once("FDL/Class.SearchDoc.php");

$db=getParam("FREEDOM_DB");

$s=new SearchDoc($db,"IUSER");
$s->setObjectReturn(); // retour d'objets documentaires
$s->addFilter('us_extmail is not null'); // retour d'objets documentaires
$s->search(); // déclenchement de la recherche
```

21) *L'accès à l'api complète se fait ici*
22) *La recherche retourne seulement les documents que l'utilisateur à le droit de voir -accès VIEW-*

```
$c=$s->count();
print "count $c\n";
$k=0;
while ($doc=$s->nextDoc()) {
  print "$k)".$doc->getTitle()."(".$doc->getValue("US_MAIL","nomail").")\n";
  $k++;
 }
```

On peut limiter le nombre de résultats avec la méthode `setSlice`.

```
$s->setSlice(10); // limit to first 10 results
```

On peut utiliser la pagination en combinant avec `start` et `slice` : Les résultats de 15 à 25

```
$s->setStart(15); // commence au 15ème
$s->setSlice(10); // limit to first 10 results
```

3.3.3. Utilisation des itérateurs

Pour récupérer la liste des documents, il est aussi possible d'utiliser un itérateur PHP afin de parcourir la liste des documents/

```
$s=new SearchDoc("","IUSER");
$s->setObjectReturn(); // retour d'objets documentaires
$s->addFilter('us_extmail is not null'); // retour d'objets documentaires
$s->search(); // déclenchement de la recherche
$dl=$s->getDocumentList();

foreach ($dl as $docid=>$doc) {
  print "$docid)".$doc->getTitle()."(".$doc->getValue("US_MAIL","nomail").")\n";
 }
```

La méthode ::getDocumentList() retourne un objet DocumentLIst qui est un itérateur. Il est possible avec cet objet de l'utiliser dans une boucle classique "foreach".

Attention : ne pas utiliser directement la méthode getDocumentList dans le foreach car cela retournerait un nouvel itérateur à chaque boucle.

3.3.3.1 callback sur un itérateur

Il est possible d'appliquer une fonction sur chacun des documents d'une liste de documents provenant d'un iterateur.

```
$s=new SearchDoc("","IUSER");
$s->setObjectReturn(); // retour d'objets documentaires
$s->addFilter('us_extmail is not null'); // retour d'objets documentaires
$s->search(); // déclenchement de la recherche

$dl=$s->getDocumentList();
$test=1;
$callback=function (&$doc) use ($test) {
              if ($test) { // test for fun
                $doc->lock();  // here lock document
              }
};
$dl->listMap($callback);
foreach ($dl as $docid=>$doc) {
```

```
  print "$docid)".$doc->getTitle()."(".$doc->getProperty("locked").")\n";
 }
```

Dans cet exemple l'ensemble des documents sera verrouillé.

La fonction de mapping est appelée sur chacun des documents lors de l'itération.. S'il n'y a pas d'itération la fonction de mapping ne sera pas appelée.

3.3.3.2 callback sur un itérateur avec filtrage

La fonction de callback permet aussi d'exclure des documents de l'itérateur. Si la méthode retourne le boolean false alors le document sera exclu de la liste.

Tout autre retour que le boolean "false", retournera le document, même s'il n'y a pas de retour explicite.

```
$s=new SearchDoc("","MY_DOCS");
$s->setObjectReturn(); // retour d'objets documentaires
$s->search(); // déclenchement de la recherche
$dl=$s->getDocumentList();

$callback=function (&$doc)  {
                    if (! $doc->isLocked()) {
                     $doc->lock();
                     return true;
                    }
                  return false;
};
$dl->listMap($callback);
foreach ($dl as $docid=>$doc) {
  print "$docid)".$doc->getTitle()."(".$doc->getProperty("locked").")\n";
 }
```

Cet exemple ne verrouillera et ne retournera que les documents venant d'être verrouillés.

3.3.4. Traitement des erreurs

Si la requête échoue suite à des erreurs de filtres, vous pouvez voir les erreurs

en utilisant les méthodes ::searchError() et ::getError().

```
<?php
include_once("FDL/Class.SearchDoc.php");
$s=new SearchDoc($dbaccess,$famId);
$s->setObjectReturn();
if ($docid > 0) $s->addFilter("id = %s",$docid);
if ($fldid > 0) $s->useCollection($fldid);
if ($allrev) $s->latest=false;
$s->addFilter("pas bon"); // ici une erreur de filtre
$s->search();

if ($s->searchError()) {
     $action->exitError(sprintf("search error : %s",$s->getError()));
}
?>
Erreur : search error : ERREUR:  erreur de syntaxe sur ou près de « bon »
LIGNE 1 : ... and (doctype != 'T') and (locked != -1) and (pas bon) ORDER...
```

La classe SearchDoc permet de récupérer les informations sur la requête afin de débugguer votre recherche. Ces informations sont consultables avec la méthode DocSearch::getSearchInfo() après avoir exécuté la recherche.:

```
include_once("FDL/Class.SearchDoc.php");

$db=getDbAccess();
$s=new SearchDoc($db,"IUSER");
$s->addFilter('us_extmail is not null');


$s->search();

// Affiche les messages de debug avec leur type
foreach($s->getSearchInfo() as $type => $msg) {
  error_log("DEBUG [$type] = $msg");
}
```

3.3.5. Recherche globale

La classe searchDoc peut aussi être utilisée pour rechercher des valeurs sur l'ensemble des attributs des documents. Pour cela il faut utiliser la méthode ::addGeneralFilter(). Cette méthode prends en argument un mot ou plusieurs mots. Ce filtre ne tient pas compte des minuscules ou majuscule. Il n'est pas possible de filtrer sur un mot en tenant compte de sa casse.

```
$s=new SearchDoc('',"ANIMAL");
$s->addGeneralFilter('cheval');
$s->search();
```

L'exemple ci-dessus permet de rechercher dans la famille "Animal" tous les documents dont au moins une valeur contient le mot "cheval". Dans cet exemple cela permets aussi de retrouver les documents contenant leur forme plurielle, dans ce cas : "chevaux". Par contre cette recherche ne retournera pas les documents contenant "chevaleresques". Cette recherche ne tient pas compte des accents.

Pour rechercher un mot exact il faut le mettre entre double quote. Ainsi

```
$s->addGeneralFilter('"cheval"');
```

retournera les documents contenant le mot "cheval" mais pas "chevaux", cette recherche tient aussi compte des accents.

Pour recherche une partie de mot, il faut utiliser le caractère ~ devant la partie à rechercher.

```
$s->addGeneralFilter('"~cheval"');
```

L'exemple ci-dessus, recherchera la documents contenant "cheval" ou "chevaleresque" mais pas "chevaux". Cette recherche tient compte des accents.

Pour rechercher plusieurs mots il suffit de les indiquer en les séparent par un espace ou le mot clef AND en majuscule.

```
$s->addGeneralFilter('cheval noir');
$s->addGeneralFilter('cheval AND noir'); // forme équivalente avec AND
```

Cette exemple recherche tous les documents dont un attribut contient le mot "cheval" et aussi le mot "noir" (sur le même attribut ou un autre).

Attention le filtre suivant

```
$s->addGeneralFilter('"cheval noir"');
```

recherche les documents contenant "cheval" suivit de "noir"

La disjonction est aussi disponible en utilisant le mot clef OR en majuscule entre deux mots

```
$s->addGeneralFilter('cheval OR poulain');
```

Différentes combinaisons peuvent ensuite être utilisées pour réaliser un filtre plus complexe

```
$s->addGeneralFilter('(cheval noir) OR (jument blanche)');
```

Cet exemple recherche les chevaux noirs ou les juments blanches.

3.3.5.1 Utilisation de l'option de vérification d'orthographe

La méthode SearchDoc::addGeneralFilter() dispose d'une option (deuxième paramètre) pour vérifier l'orthographe des mots en français.

```
$s->addGeneralFilter('méson',true);
```

Cet exemple lance une recherche sur les mots "méson" ou "maison". Cette vérification n'est pas effectuée sur les termes exacts (avec double quote) ni sur les expressions (usage du ~).

3.3.5.2 Ordonnancement par pertinence

Par défaut les documents retournés par SearchDoc sont triés par titre. Il est possible de les trier par pertinence.

Le calcul de la pertinence est effectué par PostgreSql. Elle prend en compte l'information lexicale, la proximité et la structure ; en fait, elles considèrent le nombre de fois où les termes de la requête apparaissent dans le document, la proximité des termes de la recherche avec ceux de la requête et l'importance du passage du document où se trouvent les termes du document. Les poids des mots sont en fonction de l'endroit où le terme est trouvé.

- Poids A : titre du document
- Poids B : attributs résumés
- Poids C : autres attributs non résumé
- Poids D : contenu des fichiers attachés (si indexation activé)

Si le terme recherché est de poids A, la pertinence sera plus élevée que s'il est est trouvé avec un poids B.

Si on a utilise une recherche générale sans expression ni mot exact, on peut utiliser la méthode ::setPertinenceOrder() sans argument :

```
$s=new SearchDoc('',"ANIMAL");
$s->addGeneralFilter('cheval');
$s->setPertinenceOrder();
$s->search();
```

Dans ce cas, les clefs d'ordonnancement sont celles du filtre. Le résultat est trié par pertinence sur le mot "cheval" (ou "chevaux").

Si on utilise le mot exact, la pertinence sera équivalente à la recherche par mot. On obtiendra le même ordre avec un calcul sur la forme lemmatisé de "cheval" (incluant "chevaux").

```
$s->addGeneralFilter('"chevaux"');
$s->setPertinenceOrder(); // la pertinence est sur le mot cheval
```

La pertinence sans argument ne peut pas être utilisée avec une recherche d'expression.

Si on veut maîtriser plus précisément la pertinence, il est possible d'utiliser ses propres mots.

```
$s=new SearchDoc('',"ANIMAL");
$s->addFilter("an_description ~ * 'Equus'");
$s->setPertinenceOrder('cheval poulain jument');
$s->search();
```

Dans ce cas, la recherche étant sur des attributs précis, il faut préciser explicitement le critère de pertinence. Dans cet exemple, la pertinence se fera sur les occurrences des mots "cheval", "poulain" ou "jument" trouvés dans les documents retournés qui contiennent 'Equus' dans leur description.

3.3.5.3 Mise en évidence des mots trouvés

La méthode DocSearch::getHighlightText() permet de retourner la partie où le texte recherché a été rencontré. Cette fonctionnalité ne peut être utilisée qu'avec une recherche de mots. Cela ne fonctionne pas avec les expressions ni

avec les mots exacts.

```
$s = new SearchDoc('', 'ANIMAL');
$s->addGeneralFilter("cheval noir");
$s->setObjectReturn();
$s->search();
$dl = $s->getDocumentList();
foreach ($dl as $doc) {
  $ht = $dl->getSearchDocument()->getHighLightText($doc);
  print "HighLight=$ht\n";
}
```

Les mots trouvés sont par défaut entourés des balises '' et ''. Elles peuvent être modifiées avec les arguments optionnels de la méthode.

3.3.6. Itérateur sur une liste d'identifiants

Si vous disposez d'une liste d'identifiant de document, il n'est pas conseillé de réaliser une boucle avec l'appel au constructeur new_doc car cela se révèle lent et consommateur en mémoire.

Il est conseillé dans ce cas d'utiliser l'itérateur de document.

```
$dl=new DocumentList();
$dl->addDocumentIdentificators(array(1112, 1110, 1120, 2034));
foreach ($dl as $fam) {
  print $fam->getTitle()."\n";
}
```

La méthode addDocumentIdentificators permet de renseigner la liste des identifiants de document. Si dans la liste un ou plusieurs identifiant n'existe pas alors l'itérateur ne les retournera pas. Dans ce cas le nombre de document retourné sera inférieur au nombre de d'identifiant donné.

Par défaut seuls les document que l'utilisateur a le droit de voir sont retournés.

Si vous voulez affiner les critères de recherche vous pouvez le faire en utilisant la recherche stockée dans l'iterateur.

```
$dl=new DocumentList();
$dl->addDocumentIdentificators(array(1180, 3852, 3853));
$dl->getSearchDocument()->addFilter("title ~ 'Doe'");
$dl->getSearchDocument()->noViewControl();
```

Cela ne retournera que les documents dont le titre contient Doe parmi les trois documents donnés sans tenir compte des droits de visibilités.

Il est possible aussi d'utiliser la fonction de callback pour l'appliquer à l'ensemble de la liste (voir ::listMap()).

3.3.7. Recherche dans un dossier et recherche récursive

La classe SearchDoc permet de faire des recherches dans un dossier ou une recherche spécifique au moyen de la méthode `DocSearch::useCollection()`. **Cette recherche n'est pas récursive par défaut**, c'est à dire qu'elle ne recherchera que dans le dossier indiqué.

il est possible de faire des recherches récursives à l'aide de la méthode `DocSearch::setRecursiveSearch()`. Le niveau de profondeur de la recherche est ensuite défini au moyen de la propriété `DocSearch::folderRecursiveLevel`, positionné à 2 par défaut[23]

```
<?php
include_once("FDL/Class.SearchDoc.php");

$dirid=2345; // identifiant d'un document recherche
$s=new SearchDoc($dbaccess);
$s->setObjectReturn();
```

23) *On notera que c'est le niveau de **récursivité**. par exemple, folderRecursiveLevel=0 vaut dire que l'on recherche dans le dossier, alors que folderRecursiveLevel=1 indique de rechercher dans le dossier et ses sous-dossiers.*

```
$s->useCollection($dirid);
$s->search();

if ($s->searchError()) {
     $action->exitError(sprintf("search error : %s",$s->getError()));
}
?>
```

3.3.8. Recherche par critère sur les valeurs d'attributs

Ce type de recherche est souvent combiné avec le critère d'appartenance à la famille. S'il n'y a pas de critère d'appartenance à une famille seules les propriétés peuvent être utilisés comme conditions de sélection.

L'ajout de critère se fait avec la méthode ::addFilter(). Celle ci ajoute une condition supplémentaire sur la requêtes. Si votre requête comporte des parties variable, il faut les déclarer dans les arguments suivant le format de la requête comme pour la fonction sprintf. Cela permet d'éviter une injection sql et garanti que la chaine de caractère sera correctement traduite en sql.

exemple :

```
$s->addFilter("title ~= '%s'",$mytitle);
$s->addFilter("id = %d and cdate > '%s'",$myid, $s->getDate());
```

3.3.8.1 Recherche sur le titre des documents (title)

Cet exemple montre la recherche de document par le titre. Ici, tout type de document est retourné si son titre contient 'jean' en majuscule ou minuscule. Les opérateurs utilisables sont les opérateurs SQL de PostgreSQL [24].

```
<?php
include_once("FDL/Class.Doc.php");
include_once("FDL/Class.SearchDoc.php");

$dbaccess=GetParam("FREEDOM_DB");
$s=new SearchDoc($dbaccess);// famid : toute famille
$nom="jean";
$s->addFilter("title ~* '%s'",$nom); // titre contient jean
$s->setStart(0);
$s->setSlice(10);// les 10 premiers//
$s->setObjectReturn();
$s->search();

while ($doc=$s->nextDoc()) {
  print "$k)".$doc->getTitle()."\n";
 }
?>
```

3.3.8.2 Recherche sur n'importe quel attribut (values)

Dans la cas où on recherche une valeurs dans n'importe quel attribut du document, on utilisera l'attribut **svalues**. Cet attribut contient la concaténation des valeurs des autres attributs.

```
<?php
include_once("FDL/Class.Doc.php");
include_once("FDL/Class.SearchDoc.php");

$dbaccess=GetParam("FREEDOM_DB");
$s=new SearchDoc($dbaccess);// famid : toute famille
$s->addFilter("svalues ~* 'jean'"); // une des valeurs contient jean
$s->setStart(0);
$s->setSlice(10);// les 10 premiers//
```

24) *Se reporter au manuel de référence postgreSQL Chapter 6. Fonctions and Operators*

```
$s->setObjectReturn();
$s->search();

while ($doc=$s->nextDoc()) {
  print "$k)".$doc->getTitle()."\n";
 }
?>
```

3.3.8.3 Recherche dans une famille

```
Dans le cas où une famille d'appartenance est précisée, il est possible
d'utiliser les attributs de cette famille comme critères.

<?php
include_once("FDL/Class.Doc.php");
include_once("FDL/Class.SearchDoc.php");

$dbaccess=GetParam("FREEDOM_DB");
$s=new SearchDoc($dbaccess,"USER");// famid : toute famille
$s->addFilter("us_fname ~* 'jean'"); // prénom contient jean
$s->setStart(0);
$s->setSlice(10);// les 10 premiers//
$s->setObjectReturn();
$s->search();

while ($doc=$s->nextDoc()) {
  print "$k)".$doc->getTitle()."(".$doc>getValue("us_mail","nomail").")\n";
 }
?>
```

L'exemple ci-dessus montre la recherche de toutes les personnes dont l'adresse email est un organisme et dont le prénom contient jean. Le paramètre sqlfilters établit une conjonction[25] de condition.

3.3.8.4 Recherche avec l'opérateur 'or'

Si on veut établir une disjonction[26], il faut l'écrire manuellement en SQL en utilisant l'opérateur `or`.

```
<?php
include_once("FDL/Class.Doc.php");
include_once("FDL/Class.SearchDoc.php");

$dbaccess=GetParam("FREEDOM_DB");
$s=new SearchDoc($dbaccess,"USER");// famid : toute famille
$s->addFilter("us_fname ~* 'jean|patrick'"); // prénom contient jean ou patrick
// mail fini par .org ou cp commence par 31//
$s->addFilter("(us_mail ~ '\.org$') or (us_workpostalcode ~ '^31')");
$s->setStart(0);
$s->setSlice(10);// les 10 premiers//
$s->setObjectReturn();
$s->search();

while ($doc=$s->nextDoc()) {
  print "$k)".$doc->title."(".$doc->getValue("us_mail","nomail").':'.
    $doc->getValue("us_workpostalcode").")\n";
 }
?>
```

25) *ET logique*
26) *OU logique*

3.3.8.5 Recherche sur des attributs de type énumérés (getKindDoc)

Pour rechercher des documents suivant des attributs de type énumérés, une fonction simplifiée de recherche existe. Cette fonction `getKindDoc()` est basée sur `getChildDoc()`, elle a juste pour but de construire la règle de filtrage adéquate pour tenir compte de la hiérarchie dans ce type d'attribut.

```
include_once("FDL/Class.Doc.php");
include_once("FDL/Lib.Dir.php");

$dbaccess=GetParam("FREEDOM_DB");
$tdoc=getKindDoc($dbaccess,
 "USER",          // nom de la famille//
 "us_type",       // attribut énuméré où s'applique le filtrage//
 "chefserv");     // clef à rechercher//

while (list($k, $v) = each($tdoc)) {
  print "$k)".$v["title"]."(".getv($v,"us_mail","nomail").")\n";
 }
```

Cet exemple permet de sélectionner la liste des chefs de service. Le chef de service a pour clef 'chefserv'.

3.3.8.6 Recherche dans un array

Si vous avez une famille avec un array contenant une liste d'id, vous pouvez utiliser ce filtre Postgresql pour retourner tous les documents contenant cet id :

```
$s->addFilter("an_child ~ '\\\\y%d\\\\y'", $initid);
```

3.3.8.7 Recherche avec jointure

Il est possible d'ajouter des critères portant sur une autre table en utilisant un mécanisme de jointure.

Ce mécanisme ne permet pas de récupérer des données provenant de cette autre table mais permet de les utiliser comme critère de recherche.

Exemple : recherche des documents qui ont dans l'historique un ordre de suppression émis par l'utilisateur courant

```
include_once("FDL/Class.SearchDoc.php");
  $dbaccess=getParam("FREEDOM_DB");
  $s=new searchDoc($dbaccess);
  $s->trash='only';
  $s->join("id = dochisto(id)");
  $s->addFilter("dochisto.uid = %d",$this->getSystemUserId());
  $s->addFilter("dochisto.code = 'DELETE'");
  $s->distinct=true;
  $result= $s->search();
```

Il est possible d'utiliser les tables dochisto, docutag ou docrel pour établir un critère de jointure. On ne peut utiliser qu'un seul ordre "join" par requête.

Il est aussi possible de créer un critère sur une famille lié :

```
$s=new SearchDoc($dbaccess,"ZOO_ANIMAL");
// la famille ZOO_ANIMAL est liée à la famille ZOO_ESPECE via l'attribut
AN_ESPECE
$s->join("an_espece::int = zoo_espece(initid)");
$s->addFilter("zoo_espece.es_ordre='Rongeur'");
```

```
$s->setObjectReturn(true);
$s->search();
while ($doc=$s->nextDoc()) {
   print $doc->getTitle()."\n";
}
```

Dans l'exemple décrit on recherche les animaux dont l'ordre déclaré dans l'espèce est 'Rongeur'. Attention à bien lier les deux famille à travers l'identificateur initial (initid) dans le cas des relations créées avec les options par défaut.

3.3.8.8 Recherche et documents confidentiels

Les documents confidentiels sont les documents dont la propriété 'confidential' est égale à 1. L'accès à ces documents doit être contrôlé par l'application qui décide quelles parties elle veut montrer. À la différence du droit voir ('view') des documents, la recherche retourne les documents confidentiels par défaut. Le filtrage est à faire du côté de l'application avec un post-traitement.

Cependant, il est possible de rajouter un filtre permettant de ne pas retourner les documents confidentiels que l'utilisateur n'a pas le droit de voir (droit : 'confidential'). Pour cela, il faut appeler la méthode "excludeConfidential()".

```
$sd=new SearchDoc($dbaccess, $famid);
$sd->excludeConfidential();
$sd->search();
```

3.3.8.9 Recherche de familles

Pour rechercher des familles, il faut utiliser la valeur "-1" lors de la construction de l'objet recherche.

```
$sd=new SearchDoc($dbaccess, -1);
// toutes les familles
$sd->search();
```

3.3.9. Utilisation de méthodes dans l'interface de recherche détaillée

Lors de la création d'une recherche détaillée, il est possible de choisir pour un critère une méthode comme valeur.

Exemple : ma_date > ::getDate(-7)

Cet exemple indique que le critère sera la date courante - 7 jours. La méthode doc::getDate() est une méthode statique de la classe doc.

Les méthodes utilisables dans les recherches détaillées sont les méthodes du document qui ont un commentaire (au format *DocBlock* : http://www.phpdoc.org/docs/latest/for-users/anatomy-of-a-docblock.html) et qui contiennent les tags "***@searchLabel***" et "***@searchType***".

Vous pouvez ainsi déclarer votre propre méthode, utilisable comme critère de recherche, dans le fichier méthode de votre famille et ajouter les commentaires *DocBlock* adéquats.

Cette méthode est généralement statique car elle ne doit pas faire appel à des valeurs de document. Par contre vous pouvez bien entendu utiliser des paramètres de la famille. La valeur de retour de cette méthode sera utilisé comme valeur du critère. Cela ne peut être appliqué que sur des opérateurs nécessitant une seule valeur.

Cette méthode spécifique sera utilisable pour des recherches détaillées portant sur la famille en question.

Exemple de déclaration de méthode :

```
/**
 * Get a random integer between 1 and 10
 *
```

```
 * @searchLabel Random integer between 1 and 10
 * @searchType int
 * @searchType double
 * @searchType money
 */
public function getRandomNumber() {
  return mt_rand(1, 10);
}
```

Le tag "**@searchLabel**" permet de spécifier le libellé qui sera présenté lors de l'affichage de la liste des méthodes compatibles. Ce libellé sera traduit via le mécanisme de localisation de Dynacase (voir chapitre « *3.12 Localisation (Traduction de dynacase dans d'autres langues)* »).

Le tag "**@searchType**" permet de spécifier les types d'attributs sur lesquels cette méthode est utilisable. L'interface de composition des critères de la recherche détaillée ne présentera alors que les méthodes compatibles avec l'attribut du critère.

Pour des besoins plus complexes de sélection des méthodes compatibles, vous pouvez surcharger la méthode "**Doc::getSearchMethods()**" pour enlever ou ajouter des méthodes à la liste générée par défaut. Les méthodes que vous ajoutez devront aussi avoir les tags "@searchLabel" et "@searchType" positionnée.

Exemple de surcharge de "Doc::getSearchMethods" :

```
public function getSearchMethods($attrId, $attrType) {
    $methods = parent::getSearchMethods($attrId, $attrType);
    if ($attrType == 'date' || $attrType == 'timestamp') {
        /*
         * Ajouter avant-hier et après-demain
         * pour les attributs de type 'date' et 'timestamp'
         */
        $methods = array_merge(
            $methods,
            array(
                array(
                    'label' => _("Day before yesterday"),
                    'method' => '::getDate(-2)'
                ),
                array(
                    'label' => _("Day after tomorrow"),
                    'method' => '::getDate(2)'
                )
            )
        );
    }
    return $methods
}
```

3.3.10. Recherche spécialisée

Pour toutes recherches non prévus en standard par l'interface, vous pouvez programmez des recherches spécifiques. Elles pourront ensuite être utilisées comme une recherche “normale” depuis l'interface grâce à la famille “recherche spécialisée”.

Lorsque vous éditer une recherche spécialisée, vous devez renseigner le fichier php où se trouve la fonction de recherche et le nom de cette fonction. Le fichier PHP devra être dans le répertoire EXTERNALS (/usr/share/what) sur votre serveur.

Les arguments de la fonctions sont au minimum de 3 :

- start : start index pour la recherche
- slice : nombre max d'éléments à retourner
- userid : utilisateur courant

Exemple issu du fichier fdlsearches.php (livré en standard).

```
function mytagdoc($start="0", $slice="ALL",$tag,$userid=0) {
  include_once("FDL/Class.SearchDoc.php");
  $dbaccess=getParam("FREEDOM_DB");
  $s=new searchDoc($dbaccess);
  $s->join("id = docutag(id)");
  $s->setSlice($slice);
  $s->setStart($start);
  $s->addFilter("docutag.uid = %d",$userid);
  $s->addFilter("docutag.tag = '%s'",$tag);

  return $s->search();
}

/**
 * function use for specialised search
 * return all document tagged TOVIEWDOC for current user
 *
 * @param int $start start cursor
 * @param int $slice offset ("ALL" means no limit)
 * @param int $userid user system identificator (NOT USE in this function)
 */
function mytoviewdoc($start="0", $slice="ALL",$userid=0) {
  return mytagdoc($start,$slice,"TOVIEW",$userid);
```

```
}
```

Des arguments supplémentaires peuvent être ajoutés dans l'attribut 'Argument PHP'. Il sont ajoutés dans l'appel à partir de la quatrième position. Pour rajouter plusieurs arguments, il faut les séparer par une virgule (exemple : `1234,ceci est un test,dernier argument`).

Dans ces arguments, il est possible de référencer des attributs du document recherche lui même. Il faut alors utiliser la notation suivante %TITLE% pour avoir le titre de la recherche, ou %SE_IDCFLD% pour avoir l'identifiant du dossier dans lequel s'exécute la recherche. N'importe quel attribut ou propriété de la recherche est accessible. Le mot clef %THIS%, permet d'obtenir l'objet recherche dans sa globalité.

La fonction de recherche doit retourner un tableau de document. Ces documents retournés doivent être de type `array` (pas object). Ce type est celui retourné par la classe SearchDoc et les fonctions getChildDoc (mode TABLE), getTdoc ou getDocsFromIds.

3.4 Aides à la saisie

3.4.1. Principe et spécification

Les fonctions d'aide à la saisie permettent de compléter ou de remplir des zones de saisie lors de l'édition des documents dynacase.

Les paramètres d'entrées peuvent être :

- des valeurs déjà saisies lors de l'édition de la fiche
- les coordonnées de la base de données dynacase
- des valeurs statiques

Cette fonction doit toujours retourner une matrice comportant les choix possibles. Cette matrice est un tableau de tableau de chaîne de caractères. La première colonne de chaque rangée contient la chaîne de caractères présentée à l'utilisateur. Les colonnes suivantes contiennent les valeurs qui seront insérées dans les zones de saisies spécifiées à l'appel.

Les fonctions d'aide à la saisie doivent être définies dans des fichiers PHP qui seront placé dans le répertoire `/usr/share/what/EXTERNALS` sur le serveur.

3.4.2. Syntaxe de la déclaration d'une aide à la saisie

```
help ::= <functionName>(<inputs>):<outputs>

functionName ::= <phpName>
inputs ::= nil
inputs ::= <input>[,<input>]*
outputs ::= <output>[,<output>]*
input ::= <attributeName>                  # nom d'un attribut du document
input ::= <parameterFamilyName>            # nom d'un paramètre de famille
input ::= '<unquotText>                    # texte statique
input ::= '<unquotText>'                   # texte statique
input ::= "<undoublequotText>"             # texte statique
input ::= <keyword>                        # mot-clef spécifiques
input ::= <property>                       # propriétés de document
input ::= <parameter>                      # paramètre applicatif global
input ::= <family>                         # identifiant de famille (name)
output ::= <attributeName>                 # nom d'un attribut de famille
output ::= <currentTitle>                  # titre d'une relation

phpName ::= <alphaPlus>[<alphaNumPlus>]*
alphaNumPlus ::= [a-zA-Z0-9_]
alpha ::= [a-zA-Z]
alphaPlus ::= [a-zA-Z_]
attributeName ::= <alpha>[<alphaNumPlus>]{0,63}
unquotText ::= [^',]+
```

```
undoublequotText ::= [^",]+
currentTitle ::= CT                          # current title
currentTitle ::= CT\[<attributeName>\]       # current title of another relation
keyword :: <currentTitle>
keyword ::= D                                # database
keyword ::= I                                # id du document courant
keyword ::= K                                # index de rangée (pour les tableaux)
keyword ::= T                                # object document (this)
keyword ::= A                                # action courante (objet Action)
property ::= ID
property ::= INITID
property ::= TITLE
property ::= FROMID
property ::= VERSION
property ::= REVISION
property ::= STATE
parameter ::= \{<parameterName>\}
parameterName ::= [<alphaNumPlus>]+
family ::= \{<familyName>\}
familyName ::= [<alphaNumPlus>]{1,64}
parameterFamilyName ::= <attributeName>
```

3.4.3. Retour unique

Le premier exemple nous montre comment retourner une liste de choix statique.

```
function getGravite() {
return array(0=> array("mineure","Mi"),
             1=> array("majeure","Ma"),
             2=> array("bloquante","Bl"));
}
```

utilisation possible :

Id	Description	Vis	Phpfile	phpfunc
TE_GRAV	Gravité	W	Test.php	getGravite():TE_GRAV

L'utilisateur, lorsqu'il demandera l'aide à la saisie, verra les trois choix: mineure, majeure et bloquante. S'il choisit *mineure* par exemple, la valeur *Mi* sera insérée dans la zone de saisie spécifiée (TE_GRAV).

Le deuxième exemple montre une utilisation plus complète avec la manipulation de documents. Cela permet de rechercher les personnes qui appartiennent à la société passée en paramètre. Un deuxième filtre sur le titre des personnes permet de restreindre cette recherche. Cette fonction a pour but de rechercher le nom et prénom d'une personne ou de les compléter.

```
function getUserOfSociety($dbaccess,$socid,$name="") {
  include_once("FDL/Class.SearchDoc.php");

  $s=new SearchDoc($dbaccess,"USER");
  $s->addFilter(sprintf("us_idsociety='%d'",$socid));//// filtre sur
l'appartenance à la société//
  if ($name != "")  $s->addFilter(sprintf("title ~*
'%s",pg_escape_string($name)));//// filtre éventuel sur le titre//
  $s->slice=100;
```

```
  $s->setObjectReturn();
  $s->search();
  $tret=array();
  while ($doc=$s->nextDoc()) {
    $tret[]=array($doc->getTitle(), $doc->getTitle());
  }
  return $tret;
}
```

utilisation possible :

Id	Description	Vis	Phpfile	phpfunc
TE_IDSOC	identification société	H		
TE_USER	une personne	W	test.php	getUserOfSociety(D, TE_IDSOC,TE_USER) :TE_USER

Pour retourner une erreur, il suffit de retourner une chaîne de caractère au lieu du tableau. Cela affichera l'erreur au lieu de la liste de choix. Exemple :

```
function searchLDAPinfo($login) {
  $err=searchLDAPFromLogin($login,false,$tinfo);
  if ($err == "") {
    $conf=getLDAPconf(getParam("NU_LDAP_KIND"));
    $tout=array();
    foreach ($tinfo as $k=>$v) {
      $login=$v[$conf["LDAP_USERLOGIN"]];
      $fn=$v["givenName"];
      $ln=$v["sn"];

      $tout[]=array($login,$login,$fn,$ln); // le premier login est la valeur
présentée
    }
  }
  if ($err) return $err;
  return $tout;
}
```

3.4.4. Retour multiple

Les retours multiples permettent de remplir plusieurs zone d'édition avec une seule aide à la saisie. Leur réalisation est identique à celle du retour unique sauf que le nombre de colonne dans la matrice est supérieur à deux.

Nous reprenons l'exemple précédent mais cette fois nous retournons aussi le mail et le téléphone du document personne. Si la personne n'a pas de téléphone, l'aide à la saisie retournera le numéro de téléphone de la société.

Lorsqu'une des valeurs vaut '?' cela signifie que la zone de saisie ne doit pas être modifiée. Par contre si la valeur vaut ' ' (chaîne vide) cela signifie que le zone de saisie sera effacée et par conséquent l'ancienne valeur en base de données sera conservée. Par conséquent, dans l'exemple, si la personne n'a pas de mail et qu'on a déjà saisi une adresse email celle-ci sera conservée.

```
function getUserOfSociety($dbaccess,$socid,$name="") {
  include_once("FDL/Class.Doc.php");
  include_once("FDL/Class.SearchDoc.php");

  $tret=array(); //// matrice à retourner//
  $docsoc= new_Doc($dbaccess, $socid);
  if ($docsoc->isAlive()) {

  $s=new SearchDoc($dbaccess,"USER");
```

```
  $s->setObjectReturn();
  $s->addFilter(sprintf("us_idsociety='%d'",$socid));//// filtre sur
l'appartenance à la société//
  if ($name != "")  $s->addFilter(sprintf("title ~*
'%s'",pg_escape_string($name)));//// filtre éventuel sur le titre//
  $s->slice=100;
  $s->search();

    while ($doc=$s->nextDoc()) {
      $tret[]=array(sprintf("%s (%s)",$doc->getTitle(),$docsoc->getTitle()), //
on présente la personne et la société
                    $doc->getTitle(),
     $doc->getValue("us_mail","?"), //**// ne supprime pas le mail si déjà saisi
et s'il n'existe pas**//
     $doc->getValue("us_phone",$docsoc->getValue("SI_PHONE"))); //**// retourne
le téléphone société si téléphone personnel non trouvé**//
    }
  }
  return $tret;
}
```

utilisation possible :

Id	Description	Vis	Phpfile	phpfunc
TE_IDSOC	identification société	W		
TE_USER	une personne	W	test.php	getUserOfSociety(D,TE_IDSOC,TE_USER):TE_USER,TE_MAIL,TE_PHONE
TE_MAIL	Son mail	W		
TE_PHONE	Son téléphone	W		

Remarque : La fonction « **func_get_args()** » permet de connaitre le nombre d'arguments indiqués dans le fichier OpenOffice.org et donc de retourner les arguments nécessaire dans le code.

3.4.5. Cas des attributs relation (docid)

Pour les attributs relation, une aide à la saisie est automatiquement mise si la décalration fait référence à une famille : `docid("TST_MA_FAMILLE")`.

L'aide à la saisie automatique est la suivante :

Id	Description	Type	Vis	Phpfile	phpfunc
TE_SOC	Société	docid("SOCIETY")	W	fdl.php	lfamily(D,SOCIETY,CT):TE_SOC,CT

Le mot-clef `CT` désigne le titre (Current Title).

Depuis la version 3.0.4 de dynacase, il est aussi possible d'utiliser le mot-clef `CT[TE_DEPT]` pour désigner le titre d'une autre relation.

Id	Description	Type	Vis	Phpfile	phpfunc
TE_SOC	Société	docid("SOCIETY")	W	fdl.php	lfamilyvalue(D,SOCIETY,CT,'title,'id,'soc_depttitle,'soc_dept):TE_SOC,CT,CT[TE_DEPT]
TE_DEPT	Département	docid("DEPT")	W		

3.4.6. Cas des attributs htmltext

Id	Description	Type	Vis	Phpfile	phpfunc
TE_TEXT	Texte	htmltext	W	test.php	addHtml(TE_TEXT):TE_TEXT

```
function addHtml($prefix) {

  $prefix=html_entity_decode($prefix, ENT_QUOTES, 'UTF-8');
  return array(array("Gras", "$prefix<strong>Gras</strong>"),
               array("Italique", "$prefix<em>Italique</em>"));

}
```

Les valeurs reçues par un type htmltext dans les aides à la saisies peuvent être encodées. Pour s'assurer d'avoir des valeurs cohérentes, il est nécessaire de les décoder avec la fonction "html_entity_decode".

3.5 Attributs calculés

3.5.1. Principe et spécification

Les valeurs des attributs calculés sont issues de méthodes de la classe de document. Comme pour les fonctions d'aide à la saisie, les paramètres d'entrées peuvent être

- les coordonnées de la base de données
- des valeurs d'attributs du document
- des valeurs statiques

Ces méthodes retournent un seul résultat sous forme de chaînes de caractère. Le résultat est mis dans l'attribut sur lequel s'applique le calcul (sauf redirection explicite faite lors de la déclaration de l'attribut). Ces fonctions sont appelées lors du rafraîchissement de document `Doc::refresh()`. D'un point de vue utilisateur, ces attributs sont rafraîchis avant chaque consultation de document.

3.5.2. Syntaxe de la déclaration d'un attribut calculé

```
computeMethod ::= <callMethod>
computeMethod ::= <callMethod>:<attributeName>
callMethod ::= <staticClass>::<methodName>(<methodInputs>)
staticClass ::= <phpName>                  # classe contenant la méthode
staticClass ::= nil                        # la classe est celle du document
methodName ::= <phpName>                   # nom de la méthode
methodInputs ::=                           # arguments de la méthode
<spaces><methodInput><spaces>[<spaces>,<spaces><methodInput><spaces>]*

methodInputs ::= nil
methodInput ::= <attributeName>            # nom d'un attribut du document
methodInput ::= '<unquotText>              # texte statique
methodInput ::= '<unquotText>'             # texte statique
methodInput ::= "<unguilText>"             # texte statique
methodInput ::= <methodKeyword>            # mot-clefs
unguilText ::= [^"]*
methodKeyword ::= THIS                     # objet document
methodKeyword ::= K                        # index de rangée pour les tableaux
spaces ::= [ ]*                            # des espaces
```

La déclaration des règles de syntaxe <unquotText> et <phpName> sont définis au paragraphe précédent "aides à la saisie".

3.5.3. Calcul simple

Si vous voulez un attribut qui donne l'heure courante pour la famille TEST, la construction du calcul sera la suivante:

```
// Method.Test.php
public function getTime() {
  return strftime(time());
}
```

Les fonctions de calcul sont des méthodes du document : méthodes standards ou définies dans les fichiers METHOD des familles.

Pour constituer un attribut calculé qui donne l'heure courante, il suffit de rajouter un attribut tel que celui défini ci-dessous.

id	définition	type	vis	phpfunc
TE_HOUR	heure courant	time	R	::getTime()

Les fichiers METHOD doivent contenir l'ensemble des méthodes de calculs nécessaire à la famille.

Exemple :

```
// Method.Test.php

public function getTime() {
  Return strftime(time());
}

public function getMail($userdocid) {
  $udoc = new_Doc($this->dbaccess, $userdocid);
  if ($udoc->isAlive()) return $udoc->getValue("US_MAIL");
  return ""
}

public function getAddress($userdocid) {
  $udoc = new_Doc($this->dbaccess, $userdocid);
  if ($udoc->isAlive()) {
    return sprintf("%s\n%s %s",
    $udoc->getValue("US_WORKADDR"),
    $udoc->getValue("US_WORKPOSTALCODE"),
    $udoc->getValue("US_WORKTOWN"));
  }
  return "";
}
```

La valeur retourné par la méthode est directement inséré dans l'attribut calculé. Si la méthode ne retourne la chaine de caractère vide celle-ci ne sera pas affectée à l'attribut.

Les attributs utilisables peuvent être :

Id	Description	type	vis	phpfunc
TE_IDUSER	Une personne	docid	W	
TE_MAIL	Son mail	text	R	::getMail(TE_IDUSER)
TE_CADDR	Adresse composée	longtext	R	::getAddress(TE_IDUSER)

3.5.4. Utilisation de méthodes statiques

Il est aussi possible d'utiliser des méthodes provenant de classes statiques.

Ceci est pratique si vous voulez partager votre code avec plusieurs familles.

Id	Description	type	phpfunc

TE_NUMBER	Un nombre	int	myNumber::number()
TE_MORE	Nombre suivant	int	myNumber::oneMore(TE_NUMBER)
TE_MSG	Un message	text	myMessage::prefix(THIS)

Voici trois exemples d'appel à des méthodes statiques. Il faut que la classe soit définie sur le serveur. L'autoloader retrouvera la classe automatiquement en fonction du nom de celle-ci.
La syntaxe est la même que précédemment, il faut juste préciser avant les '::' le nom de la classe.
Si vous passez THIS cela donnera le document en paramètre de la méthode.

```
class myNumber {
  static public function number() {
    return rand(0,100);
  }
  static public function oneMore($n) {
    return $n+1;
  }
}
class myMessage {
  static public function prefix(Doc &$doc) {
    return "Test: ".$doc->getTitle();
  }
}
```

3.5.5. Calcul multiple

Si la famille que vous construisez utilise de nombreux attributs calculés pouvant être mis à jour avec le même algorithme, on utilise la méthode `Doc::specRefresh()`.

Cette méthode est une surcharge de la classe père[27]. Elle est appelée par la méthode `Doc::refresh()` comme les autres méthodes de calculs. Au contraire des méthodes de calcul simples, cette méthode doit mettre à jour les valeurs des attributs à l'aide de la méthode `Doc::setValue()`.

```
// Method.Test.php
public function specRefresh() {
  Parent::specRefresh(); //// réutilisation éventuelle de la méthode du père//
  //spécifie les paramètres attribut d'entrée et de sortie
  $this->AddParamRefresh("TE_USERID","TE_MAIL,TE_CADDR");

  $udoc= new_Doc($this->dbaccess, $this->getValue("TE_USERID"));
  if ($udoc->isAlive()) {
    $mail=$udoc->getValue("US_MAIL");
    $caddr=sprintf("%s\n%s %s",
    $udoc->getValue("US_WORKADDR"),
    $udoc->getValue("US_WORKPOSTALCODE"),
    $udoc->getValue("US_WORKTOWN"))
    $this->setValue("TE_MAIL",$mail);
    $this->setValue("TE_CADDR",$caddr);
  }
}
```

La méthode `Doc::specRefresh()` peut retourner un message d'avertissement qui est affiché lors de la consultation du document.

3.5.6. Afficher un message d'avertissement en cas d'erreur (addWarningMsg)

Il est possible d'afficher un popup à l'écran pour avertir l'utilisateur d'un problème quelconque :

27) *initialement la classe Doc*

```
addWarningMsg("Votre message");
```

3.6 Contraintes

La contrainte permet d'interdire la création ou la mise à jour de document.

Elle est appelé à chaque tentative de sauvegarde par l'interface web. Elle est appelée aussi lors de l'appel à la méthode Doc::store() qui permet d'enregistrer les modifications d'un document. La vérification des contraintes peut aussi être appelée explicitement avec la méthode Doc::verifyAllConstraints().

3.6.1. Syntaxe de la déclaration d'un attribut calculé

```
constraintMethod ::= <callMethod>
```

La déclaration de la règle <callMethod> est défini au paragraphe précédent "Attributs calculés". La déclaration doit être écrite dans la colonne "constraint" de votre fichier de spécification de famille.

3.6.2. La fonction de contrôle

La fonction de contrôle déclenchée est une méthode du document. Son prototype standard est le suivant :

```
public function ma_contrainte([P,[P,[...]]) {

   $error_message = "";
   $proposal = array();

   // ... some code ...

  if ($this->getValue("fam_attr")=12 && $P<0) {
     $error_message = "Invalid value given ($P)";
     $proposal[] = "10";
     $proposal[] = "12";
  }

   // ... some code ...

  return array( "err" => $error_message,
                "sug" => $proposal );
}
```

Cette méthode doit retourner un tableau avec deux entrées. Une entrée “err” qui contient le message d'erreur de la contrainte. S'il n'y a pas d'erreur le message d'erreur doit être vide. Cette méthode peut retourner plus simplement une chaîne de caractère. Elle sera alors considérée comme l'erreur à remonter. Si cette chaîne est vide cela sera considéré comme une validation de la contrainte.

3.6.3. Contrainte système

La classe documentaire mets à disposition des contraintes pré-définies:

19. ::parseMail($mail) - permet de teste la valider d'une adresse email

20. ::isFutureDate($date) - vérifie que la date est dans le futur

1. ::isFloat($x,$min,$max) - teste si le nombre est compris entre min et max
2. ::isInteger($x,$min,$max)- teste si le nombre est compris entre min et max
3. ::isString($s,$pattern)- teste si le texte correspond au pattern (pattern au sens perl pattern voir manuel php preg_match).

Exemple mobile ::isString(te_mobile,0[6|7][0-9]{6}) numéro à 8 chiffres commençant par 06 ou 07

3.7 Surcharges des méthodes du document

Les surcharges de méthodes sont faites via les fichiers `METHOD`.

Si vous modifiez les fichiers Method sur le serveur, il faut lancer le shell `fdl_adoc` pour regénérer la classe PHP documentaire correspondante :

```
[root@chewbacca Freedom]# wsh --api=fdl_adoc --docid=MAILBOX
/usr/share/what/FDLGEN/Class.Doc1403.php [boîte aux lettres(MAILBOX)]
```

3.7.1. Contrôle à la création

La création de document est conditionnée par les droits de l'utilisateur courant lorsqu'on utilise la fonction `createDoc()`. La création en base de données est faite par la méthode `Doc:Add()`. Elle est appelée aussi à chaque révision du document[28].
Cette méthode appelle la méthode `Doc::PreCreated()` qui peut être définie par les différentes classes de documents créées. Cette méthode permet de rajouter des contraintes ou des calculs nécessaires avant la création en base. Si cette méthode retourne un message d'erreur l'insertion en base de données sera abandonnée.

```
function preCreated() {

  if ( $this->revision == 0) { //// traitement seulement pour la première
révision//

    //// vérifie si l'état de la commande client est "à facturer" //
    $cmcid=$this->getValue("FACT_IDCMC"); //// recherche commade client
associée//
    if ($cmcid > 0) {

       $cmd= new_Doc($this->dbaccess,$cmcid);
       $cmdid= $cmd->latestId(); //// dernière révision//
       $cmd= new_Doc($this->dbaccess,$cmdid);

       if ($cmd->state != "forbilled")
          return sprintf(_("the state of the commande %s must be forbilled"),
$cmd->title);

    }
  }
  return "";
}
```

L'exemple ci-dessus montre l'utilisation de contraintes supplémentaires : le document *facture* ne sera créé que si la commande client associée est dans l'état « à facturer ».

Des traitements supplémentaires peuvent aussi être effectués après la création du document en base de données. Ces traitements doivent être définis dans la méthode `Doc::postCreated()`. Généralement cette méthode est utilisée pour mettre à jour ou pour créer des documents corollaires.

```
function postCreated() {
  if ( $this->revision == 0) {
    //// mise à jour de numéro de séquence//
    $this->setValue("FACT_NUMBER",$this->getCurSequence());
  }
}
```

Cette exemple montre la mise à jour d'un attribut calculé qui n'est fait qu'une fois à la création. Cette méthode ne retourne rien.

28) *une révision du document est un nouveau document*

3.7.2. Contrôle à la modification

Le contrôle d'accès en modification est faite par la méthode `Doc::preUpdate()`. Cette méthode vérifie que l'utilisateur courant à les droits en modification pour ce document. Ce contrôle peut être inhibé temporairement en utilisant la méthode `Doc::disableEditControl()`. Le contrôle peut être réactivé en utilisant la méthode `Doc::enableEditControl()`.

La méthode `Doc::postModify()` est modifiable par les classes de documents. Cette méthode est appelée lors d'une modification par l'utilisateur (par la fonction modcard()), ou lors d'importation de documents. Elle permet d'effectuer des calculs d'attributs sur le document ou sur les documents corollaires.

Cette méthode est appelée par la méthode doc::store().

Si le calcul d'un attribut est invariable en fonction du contenu du document, il est préférable d'effectuer le calcul lors du `Doc::postModify()` plutôt que lors du `Doc::specRefresh()` : le calcul ne sera pas ré-effectué à chaque consultation.

Si cette méthode retourne un message d'erreur, la modification est quand même effectuée, mais un message d'erreur est affiché lorsque l'utilisateur effectue la modification du document.

3.7.3. Contrôle à la suppression

La méthode `Doc::delete()` contrôle l'accès à la suppression du document (droit delete). Les méthodes `DbObj::preDelete()` et `DbObj::postDelete()` peuvent être surchargées pour effectuer des traitements ou vérifications supplémentaires. Ces deux méthodes sont appelées dans les cas de suppression réelle de la base de données ou de mise en zombie du document. Si la méthode `DbObj::preDelete()` retourne un message d'erreur le suppression n'est pas effectuée.

3.7.4. Autres méthodes

- `Doc:preCopy` et `Doc:postCopy()` : ces méthodes sont exécutées lors de la copie d'un document respectivement avant et après sont stockage dans la base dynacase.
- `Doc:preImport()` et `Doc:postImport()` : ces méthodes sont exécutée lors de l'importation d'un document (avant et après le stockage en base).

Pour plus d'information rendez-vous sur la documentation de l'API dynacase.

3.7.5. Ordre d'appel des méthodes

3.7.5.1 Consultation de document

Lors d'un appel à l'action FDL/FDL_CARD les méthodes “surchargeables” appelées sont :

21. ::specRefresh()
22. ::preConsultation() (version ≥ 2.12.9)

3.7.5.2 Edition de document

Lors d'un appel à l'action GENERIC/GENERIC_EDIT les méthodes “surchargeables” appelées sont :

1. ::specRefresh()
2. ::preEdition() (version > 2.11.11)

3.7.5.3 Création de document

Lors d'un appel à l'action GENERIC/GENERIC_MOD les méthodes “surchargeables” appelées sont :

1. ::preCreated()
2. ::postCreated()
3. ::specRefresh()
4. ::postModify()

Légende : En jaune, les méthodes non surchargeables, en vert les méthodes surchargeables.

3.7.5.4 Sauvegarde de document

Lors d'un appel à l'action GENERIC/GENERIC_MOD les méthodes “surchargeables” appelées sont :

1. ::specRefresh()
2. ::postModify()

3.7.5.5 Duplication de document

Lors d'un appel à l'action FREEDOM/FREEDOM_DUPLICATE les méthodes “surchargeables” appelées sont :

1. ::preCopy()
2. ::postCopy()
3. ::specRefresh()
4. ::postModify()

3.7.5.6 Suppression de document

Lors d'un appel à l'action GENERIC/GENERIC_DEL les méthodes “surchargeables” appelées sont :

1. ::preDelete()
2. ::postDelete()

3.7.5.7 Import de document

Lors d'un appel à l'action FREEDOM/FREEDOM_IMPORT les méthodes “surchargeables” appelées sont :

1. ::preImport()
2. ::preCreated() *(en cas de création seulement)*
3. ::postCreated() *(en cas de création seulement)*
4. ::specRefresh()
5. ::postModify()
6. ::postImport()

3.8 Principales méthodes de la classe Doc

La classe Doc

3.8.1. lock

La méthode lock permet de verrouiller un document avec l'utilisateur couramment connecté.

```
include_once("FDL/Class.Doc.php");

$dbaccess=getParam("FREEDOM_DB");
$doc=new_doc($dbaccess,4567);
if ($doc->isAlive()) {
  $err=$doc->lock();

}
```

La méthode isLocked() permet de savoir s'il y a un verrou posé. La propriété `locked` contient l'identificateur système de l'utilisateur qui a verrouillé.

3.8.2. unlock

3.8.3. addRevision

3.8.4. attachTimer

La méthode attachTimer permet d'attacher un minuteur à un document.

```
include_once("FDL/Class.Doc.php");

$dbaccess=getParam("FREEDOM_DB");
$doc=new_doc($dbaccess,4567);
$timer=new_doc($dbaccess,47232);
if ($doc->isAlive()) {
  $err=$doc->attachTimer($timer);
  if ($err) print "$err";
}
```

Cette méthode possède un deuxième attribut optionnel pour indiquer quel est le document d'origine du minuteur. Cela est utile par exemple lors qu'un document pilote d'autres documents. Le pilote peut ainsi savoir les minuteurs qu'il a posé. Le troisième attribut optionnel est la date d'exécution de la première action. Si elle n'est pas renseignée, la date d'exécution est égale à maintenant + le premier délai. Si elle est renseignée, cela sera la date posée. Si la date est dans un passé récent (voir minuteur dépassé, cela signifiera quelle sera exécutée à la prochaine vérification (toutes les 5 minutes). Autres méthodes relatives :

23. unattachTimer
24. resetDynamicTimers
25. unattachAllTimers
26. getAttachedTimers

3.9 Vues et éditions particulières

Ce chapitre aborde les techniques liées à la modification de la présentation des documents en consultation et édition... :

3.9.1. Vues par défaut

La présentation des valeurs des documents est effectuée à l'aide de vue. La vue permet de définir la représentation graphique du document. Un même document peut avoir plusieurs vues à sa disposition. Ces différentes vues permettent de représenter le document sous différents aspects soit pour mettre le focus sur certaines parties du document ou afin d'être utilisé par différents intervenants. La vue peut être utilisée pour la consultation, pour l'envoi de mail, pour l'impression ou pour l'édition.

Consultation complète	Consultation résumé	Édition

Les vues sont générées à l'aide de la méthode `Doc::viewDoc()`. Cette méthode est utilisée principalement dans les fonctions `editcard()` et `viewcard()`.

Pour construire une vue particulière il est nécessaire de définir un squelette (template) XML. Ce squelette décrit la représentation du document (généralement pour produire du HTML). Ce squelette contient des parties variables qui sont complétées par une méthode spécifique du document (par défaut `Doc::viewdefaultcard()`). La mécanique de génération est effectuée par la classe Layout de WHAT.

La méthode `Doc::viewDoc()` utilise comme paramètre `$layout`. Ce paramètre permet de spécifier la vue à utiliser pour la génération de la représentation. Ce paramètre est une chaîne de caractères composées de trois parties séparées par le caractère ':' (la troisième partie étant facultative)

- la localisation du template [29]
- le nom du template [30]
- un paramètre optionnel -décrit plus tard-

Par défaut pour la vue de consultation, ce paramètre vaut FDL:VIEWBODYCARD. Cela indique que la méthode viewDoc utilisera le fichier squelette <freedom-root>[31]/FDL/viewbodycard.xml et que la méthode appliquée pour compléter le squelette sera ::viewbodycard().

Les trois vues par défaut sont définies par les attributs de la classe Doc. Ces valeurs peuvent être redéfinies par les classes filles . Ces redéfinitions se font dans le fichier Method associé à la famille.

```
//// vue de consultation complète//
var $defaultview = "FDL:VIEWBODYCARD";

//// vue d'édition//
var $defaultedit = "FDL:EDITBODYCARD"

//// vue de consultation résumé//
var $defaultabstract = "FDL:VIEWABSTRACTCARD";
```

On différencie les vues de consultations des vues d'éditions du fait que ces dernières doivent forcément produire du HTML afin de récupérer les informations saisies.

29) *nom de l'application fournissant le template*
30) *celui de la vue en minuscule suffixé de .xml*
31) */usr/share/what*

3.9.2. Syntaxe d'un fichier Layout

Le fichier Layout décrit une présentation graphique. Cette représentation définit le contenu du fichier qui sera transmis au navigateur. Ce fichier contient des parties statiques et des parties dynamiques. Les parties dynamiques sont notées entre crochets. Tout ce qui n'est pas entre crochets est statique et ne sera pas modifié.

La génération du layout consiste à remplir les parties dynamiques. Cette instanciation est faite à l'aide de quatre sources :

- les données propres au squelette : ces données doivent être fournies par une fonction spécifique de remplissage.
- Les données de traduction : ces données sont issues de fichier gettext suivant la langue.
- Les données de paramètres : ces paramètres dépendent de l'application (paramètre applicatifs) et de l'utilisateur (préférences utilisateur).
- Les zones : permettent de faire référence à d'autre squelette. La composition d'un squelette peut alors être faite comme par assemblage de layout.

3.9.2.1 Données atomiques

Les données atomiques sont les données qui ont des valeurs non structurées. Une donnée atomique est référencée dans le layout par la syntaxe **[DATA]** DATA étant le nom de la variable. Leur valeur est donnée explicitement par la fonction d'instanciation du layout. L'association entre le nom de la variable du layout et sa valeur est faite avec la méthode Layout::Set().

```
include_once("Class.Layout.php");

$lay = new Layout(getLayoutFile("TEST","test.xml"),$action);
$x=34;
$y=78;
$lay->set("X",$x);
$lay->set("Y",$y);
$lay->set("XplusY",$x+$y);
$lay->set("XfoisY",$x*$y);
print $lay->gen();
```

test.xml :

```
La somme de [X] + [Y] = [XplusY]
Le produit de [X] + [Y] = [XfoisY]
```

Résultat de la génération :

```
La somme de 34 + 78 = 112
Le produit de 34 + 78 = 2652
```

Comme le montre l'exemple toutes les occurrences d'une même variable sont remplacées dans le fichier. Les noms des variables sont sensibles à la casse de caractères. Ainsi, [X] et [x] identifient deux variables distinctes.

La modification du fichier n'a lieu que lors de la génération (`Layout::gen()`). Les valeurs associées peuvent être modifiées tant que la génération n'a pas eu lieu.

```
include_once("Class.Layout.php");

$lay = new Layout(getLayoutFile("TEST","test.xml"),$action);
$x=34;
$y=78;
$lay->set("X",$x);
$lay->set("Y",$y);

$x1=$lay->get("X"); // récupération de X
$lay->set("X",$x1+1); // mise à jour de X
```

```
$lay->set("XplusY",$x+$y);
$lay->set("XfoisY",$x*$y);

print $lay->gen();
```

test.xml :

```
La somme de [X] + [Y] = [XplusY]
Le produit de [X] + [Y] = [XfoisY]
```

Résultat de la génération :

```
La somme de 35 + 78 = 113
Le produit de 35 + 78 = 2730
```

La méthode `Layout::get()` permet de récupérer la valeur précédemment affectée dans le layout.

3.9.2.2 Données listes

La classe Layout permet d'utiliser les variables de type liste. Pour indiquer une liste dans un layout, on utilise les mot-clef **BLOCK** et **ENDBLOCK**. L'association se fait à l'aide de la méthode `Layout::setBlockData()`.

```
include_once("Class.Layout.php");

$lay = new Layout(getLayoutFile("TEST","test.xml"),$action);


$x=3;
$tmul = array(); // tableau pour le bloc MUL
for ($i=1;$i<11;$i++) {
  $tmul[]=array("Y"=>$i,
 "XfoisY"=>$i*$x);
}

$lay->set("X",$x);
$lay->setBlockData("MUL",$tmul);

print $lay->gen();
```

test.xml

```
Table de multiplication de [X]
[BLOCK MUL]
  - [X] * [Y] = [XfoisY][ENDBLOCK MUL]
```

Résultat :

```
Table de multiplication de 3

  - 3 * 1 = 3
  - 3 * 2 = 6
  - 3 * 3 = 9
  - 3 * 4 = 12
  - 3 * 5 = 15
  - 3 * 6 = 18
```

```
  - 3 * 7 = 21
  - 3 * 8 = 24
  - 3 * 9 = 27
  - 3 * 10 = 30
```

La portée des variables atomiques va aussi dans les blocs. Ainsi [X] est instancié dans chaque occurrence de la liste. Chaque ligne de la liste est un tableau qui contient les différentes valeurs du bloc. Chaque ligne peut alors contenir plusieurs variables.

Au contraire des variables atomiques, un bloc ne peut être utilisé qu'une fois dans le layout avec le même identifiant. Dans l'exemple, on ne peut pas avoir deux blocs nommés MUL.

3.9.2.3 Données tableau

Le principe d'instanciation des tableaux consiste à imbriquer deux niveaux de blocs. Un bloc pour délimiter les rangées du tableau et un bloc pour délimiter les colonnes.

Le nom des sous/blocs est dynamique et doit être unique. Dans l'exemple, on choisit de les préfixer par “rmul”. Leur unicité est donnée par le numéro de rangée $i.

```
include_once("Class.Layout.php");

$lay = new Layout(getLayoutFile("TEST","test.xml"),$action);

$x=3;
$tmul = array(); // tableau pour le bloc MUL
$thmul = array(); // tableau pour le header HMUL
for ($i=1;$i<10;$i++) {
  // construction de la tête de tableau
  $thmul[]=array("X"=>sprintf("%3d",$i));
  // construction du corps du tableau
  $tmul[]=array("RMUL"=>"rmul$i",
 "Y"=>sprintf("%3d",$i));
  $trmul=array(); // nouvelle rangée RMUL
  for ($j=1;$j<10;$j++) { // 9 colonnes
    $trmul[]=array("XfoisY"=>sprintf("%3d",$i*$j));
  }
  // on mémorise la nouvelle rangée
  $lay->setBlockData("rmul$i",$trmul);
}
$lay->setBlockData("MUL",$tmul);
$lay->setBlockData("HMUL",$thmul);

print $lay->gen();
```

test.xml

```
Table de multiplications
   -------------------------------------
  x|[BLOCK HMUL][X]|[ENDBLOCK HMUL]
=========================================[BLOCK MUL]
[Y])[BLOCK [RMUL]][XfoisY]|[ENDBLOCK [RMUL]]
---+---+---+---+---+---+---+---+---+---+---+[ENDBLOCK MUL]
```

Résultat :

```
Table de multiplications
   -------------------------------------
```

```
  x|  1|  2|  3|  4|  5|  6|  7|  8|  9|
=========================================
  1)  1|  2|  3|  4|  5|  6|  7|  8|  9|
---+---+---+---+---+---+---+---+---+---+
  2)  2|  4|  6|  8| 10| 12| 14| 16| 18|
---+---+---+---+---+---+---+---+---+---+
  3)  3|  6|  9| 12| 15| 18| 21| 24| 27|
---+---+---+---+---+---+---+---+---+---+
  4)  4|  8| 12| 16| 20| 24| 28| 32| 36|
---+---+---+---+---+---+---+---+---+---+
  5)  5| 10| 15| 20| 25| 30| 35| 40| 45|
---+---+---+---+---+---+---+---+---+---+
  6)  6| 12| 18| 24| 30| 36| 42| 48| 54|
---+---+---+---+---+---+---+---+---+---+
  7)  7| 14| 21| 28| 35| 42| 49| 56| 63|
---+---+---+---+---+---+---+---+---+---+
  8)  8| 16| 24| 32| 40| 48| 56| 64| 72|
---+---+---+---+---+---+---+---+---+---+
  9)  9| 18| 27| 36| 45| 54| 63| 72| 81|
---+---+---+---+---+---+---+---+---+---+
```

3.9.2.4 Conditions

Certaine partie du layout peuvent être soumise à une condition. La valeur de la condition se fait par la méthode `Layout::set()` comme pour les données atomiques. Par contre, la valeur du paramètre doit être un booléen. Pour indiquer les parties conditionnelles on utilise les mot-clefs **IF** ou **IFNOT** suivis de **ENDIF**.

L'exemple comporte deux valeurs conditionnelles : Interdit et Pair. Le premier est affecté directement à l'aide de la méthode `layout::set()`. Pour valoriser des conditions dans un bloc, il suffit d'affecter la valeur booléenne comme on le ferait pour un mot-clef.

```
include_once("Class.Layout.php");

$lay = new Layout(getLayoutFile("TEST","test.xml"),$action);

$t = array(); // tableau pour le bloc LOOP
for ($i=1;$i<10;$i++) {
  // construction de la liste de 10 nombre
  $t[]=array("X"=>sprintf("%3d",$i),
      "Pair"=>(($i%2)==0));
}
$lay->setBlockData("LOOP",$t);
$lay->set("Interdit",false);

print $lay->gen();
```

test.xml

```
[IF Interdit]Ne pas afficher[ENDIF Interdit]
Détection de la parité
[IFNOT Interdit]Voir absolument[ENDIF Interdit]
[BLOCK LOOP]
[X])[IF Pair][X] est pair[ENDIF Pair][IFNOT Pair][X] est impair[ENDIF Pair]
[ENDBLOCK LOOP]
```

Résultat :

```
Détection de la parité
Voir absolument

  1)  1 est impair
  2)  2 est pair
  3)  3 est impair
  4)  4 est pair
  5)  5 est impair
  6)  6 est pair
  7)  7 est impair
  8)  8 est pair
  9)  9 est impair
```

3.9.2.5 Internationalisation

Si vous voulez des représentations qui dépendent de la langue, il est nécessaire de marquer les textes avec le mot-clef **TEXT**. La langue est stocké dans le paramètre applicatif **CORE_LANG**. Il peut être égal à fr_FR (français) ou en_US (anglais américain). La traduction est faite à l'aide des fichiers au format gettext. Le fichier de traduction utilisé est : <dynacase-root>148/what/locale/[en,fr]/LC_MESSAGES/what.mo. Ce fichier est la concaténation de tous les fichiers .mo disponibles dans ce répertoire.

Si vous voulez rajouter un fichier de traduction (.mo), il faut l'insérer dans le répertoire. Ensuite, il suffit de lancer la commande <freedom-root>148/what/whattext pour refaire la concaténation. Il faut enfin relancer le serveur apache pour prendre en compte les nouvelles traductions.

Si le mot est non traduit ou si la langue est inconnu, le mot original est retourné.

3.9.2.6 Paramètres d'environnement

Tous les paramètres de l'environnement applicatif sont remplacés systématiquement. Ces paramètres peuvent avoir des valeurs différentes suivant l'application WHAT utilisée ou suivant l'utilisateur connecté.

Si le paramètre est inconnu, le texte et les crochets correspondants au paramètre restent affichés.

3.9.2.7 Les zones

Les zones permettent de faire référence à d'autres layout. Cela permet une plus grande réutilisabilité des layout. La référence d'une zone est composée de deux parties séparé par ':'.

Par exemple [ZONE FDL:VIEWSCARD], fait référence au layout “FDL/Layout/viewscard.xml”. Si le fichier “FDL/viewscard.php” existe la fonction `viewscard()` est appelée.

Des paramètres supplémentaires peuvent être indiqués lors de l'appel à la zone. Ces paramètres suivent la même syntaxe que pour les paramètres d'URL. Ils sont transmis à la zone sous formes de variables HTTP et sont accessibles avec la méthode getHttpVars() depuis le code du contrôleur. Exemple

```
[ZONE FDL:VIEWSCARD?id=52965]
```

Vous trouverez des exemples d'utilisation des zones au chapitre suivant :

- Exemples de Layout

3.9.3. Vue de consultation par défaut

La vue de consultation complète est une vue qui est affichée à l'aide de l'action FDL_CARD. Le layout de cette action inclus la zone FDL:VIEWCARD qui elle même inclus le corps du document. C'est dans cette dernière zone que l'on peut personnaliser la représentation de la vue de consultation complète par défaut.

Par défaut, la vue est FDL:VIEWBODYCARD. Elle représente des attributs dans l'ordre donné dans la définition des attributs et dans des cadres arrondis.

Pour modifier la vue par défaut, il est nécessaire d'associer un fichier METHOD à la famille. Dans ce fichier, l'attribut defaultview de la classe Doc doit être redéfini. Ensuite il suffit de créer le layout et la fonction de remplissage. Pour les documents cette fonction sera une méthode de l'objet documentaire. Les méthodes de remplissage utilisées pour les vues doivent avoir trois paramètres:

- $target : nom de la fenêtre graphique qui sera utilisé pour les hyperliens (_self par défaut)
- $ulink : booléen : vrai indique s'il faut générer les hyperliens (vrai par défaut)
- $abstract : booléen : vrai indique qu'il ne faut pas générer les attributs non résumés (faut par défaut).

La déclaration d'une méthode de remplissage doit toujours être :

```
ma_vue($target="_self",$ulink=true,$abstract=false)
```

La partie modifiable pour la consultation par défaut est la partie visible en jaune clair (FDL:VIEWBODYCARD). Le reste est imposé par l'application par soucis d'homogénéité.

Exemple : `fdl_card.xml`.

```
<html>
```

```
<head>
  <title>[TEXT:Properties:] [TITLE]</title>
  <LINK REL="icon" HREF="[CORE_PUBURL]/CORE/Images/logo-1-mini.ico" >
  <LINK REL="SHORTCUT ICON" HREF="[CORE_PUBURL]/[iconsrc]" >
  <LINK REL="stylesheet" type="text/css"
HREF="[CORE_BASEURL]app=CORE&amp;action=CORE_CSS" >
  <LINK REL="stylesheet" type="text/css"
HREF="[CORE_BASEURL]app=FDL&amp;action=FDL_CSS" >
  [CSS:REF]

  <style type="text/css">
        [CSS:CODE]
  </style>
  [JS:REF]
  <script language="JavaScript">
   [JS:CODE]
  </script>
</head>
<body onClick="closeMenu('popupcard');" >
  [ZONE FDL:POPUPCARD]
  <div width="100%" class="tableborder"
       style="cursor:crosshair;"
       onContextMenu="openMenu(event,'popupcard',1);return false"
       onMouseDown="if (shiftKeyPushed(event)) openMenu(event,'popupcard',1)" >
  [ZONE FDL:VIEWCARD]
  </div></body>
</html>
```

3.9.3.1 Consultation avec la méthode par défaut

Pour illustrer les différentes vues, on utilise la famille société. Comme dit précédemment, on attache un fichier METHOD à la cette famille (Method.Society.php).

var $defaultview= “FDL:VIEWSOCIETY”;

Dans ce fichier FDL/Method.Society.php, on indique seulement la référence à la vue de consultation par défaut. La méthode viewsociety() n'existant pas ce sera la méthode Doc::viewDefaultCard() qui sera utilisée pour compléter cette vue. Cette méthode permet d'insérer toutes valeurs d'attributs en indiquant leur identifiant:

- **V_<attr>** : valeur de l'attribut formaté (avec hyperlien si existant et formatage si il est défini)
- **L_<attr>** : description de l'attribut

Cette méthode permet aussi d'afficher la valeur brute des propriétés ou attributs en indiquant leur identification en majuscule.

Afin de simplifier l'affichage, la zone FDL:VIEWFRAME permet d'afficher tous les attributs présents dans un cadre. Cette zone nécessite un paramètre (frameid) pour indiquer l'identificateur du cadre. Cette zone retourne une rangée de tableau HTML (commence par <tr> et fini par </tr>). Cette zone doit donc être incluse dans un tableau (balise père est <table>, <thead>, <tbody> ou <tfoot>).

viewsociety.xml	Représentation
`<H1>TEXTE STATIQUE</H1>`	ZOO NET TEXTE STATIQUE
`<P>le numéro de téléphone de [V_SI_SOCIETY] est <B>[V_SI_PHONE]</P> Autres renseignements : <UL> <UL>[L_SI_FAX] : [V_SI_FAX]</UL> <UL>[L_SI_TOWN] : [V_SI_TOWN]</UL>`	ZOO NET le numéro de téléphone de Zoo Net est **09.89.33.34.90** **Autres renseignements :** **fax : 09.89.33.34.90** **ville : Toulouse** **web : www.zoo.org**

```
  <UL>[L_SI_WEB] : [V_SI_WEB]</UL>
</UL>
```

```
<P>les propriétés sont <UL>
<UL>titre : [TITLE]</UL>
  <UL>date de modification :
[REVDATE]</UL>
  <UL>n° de révision :
[REVISION]</UL>
  <UL>commentaire :
<pre>[COMMENT]</pre></UL> </UL>
```

```
<iframe width="100%" src="http://
[SI_WEB]"></iframe>
  <table width="100%"><tr> <td
style="background-color:moccasin">
    <table width="100%"> [ZONE
FDL:VIEWFRAME?frameid=SI_FR_IDENT]
</table></td>
    <td style="background-
color:burlywood">
       <table width="100%"> [ZONE
FDL:VIEWFRAME?frameid=SI_FR_COORD]
</table></td>
  </tr></table>
```

Le principal inconvénient des vues personnalisées est que la vue est très liées aux attributs de la famille. Ainsi chaque changement d'attribut (ajout/suppression) risque d'impliquer une modification de la représentation. Si la vue indique explicitement des attributs ou des cadres, il faut rajouter ces nouveaux attributs ou cadres à la représentation à chaque changement.

Pour les attributs de type tableaux, ils sont visibles par une zone comme pour les cadres. La syntaxe de la zone est la suivante : [ZONE FDL:VIEWARRAY?arrayid=*SI_T_SITES*] par exemple pour le tableau SI_T_SITES.

3.9.3.2 Consultation avec méthode spécifique

L'utilisation de la méthode spécifique offre plus de possibilités dans la représentation. Cette méthode va permettre de calculer la représentation en fonction du contenu. La méthode de contrôle doit comporter la tag "**@templateController**" dans son commentaire afin d'être autorisé à être utilisé pour la constitution d'une vue de document.

L'objet layout du document est l'attribut Doc::lay de l'objet document. On utilise cet attribut pour insérer les valeurs dans le squelette. Vue circulaire :

Représentation
viewsociety.xml [BLOCK ATTR] <P style="position:absolute;left:[x]px;top:[y]px" title="[frame]/[label]"> [value] </P> [ENDBLOCK ATTR]

```
Method.society.php
/**
 * @templateController Default society view
 */
public function viewsociety($target="_self",$ulink=true,$abstract=false) {
  $rx=300; // rayon X
  $ry=200; // rayon Y
  if ($abstract){
    $listattr = $this->GetAbstractAttributes();
```

```
    } else {
      $listattr = $this->GetNormalAttributes();
    }
    reset($listattr);
    $tattr=array();
    while (list($i,$attr) = each($listattr)) {
      $value = chop($this->GetValue($i));
      if ($value != "") {
        $tattr[]=array("value"=>$this->GetHtmlValue($attr,$value,$target,$ulink),
        "label"=>$this->getLabel($i),
        "frame"=>$this->getLabel($attr->fieldSet->id) );
      }
    }
    reset($tattr);
    $delta = 2*pi()/count($tattr);
    while (list($k,$v) = each($tattr)) { // placement dans le cercle
      $tattr[$k]["y"]=$ry+sin($delta*$k)*$ry+20;
      $tattr[$k]["x"]=$rx+cos($delta*$k)*$rx+20;
    }
    $this->lay->setBlockData("ATTR", $tattr); // enregistrement des valeurs
    $this->lay->set("dy",$ry*2);
  }
```

3.9.4. Vues de consultation spécifiques

Les vues de consultations qui ne sont par défaut, n'ont pas les mêmes contraintes que celle par défaut. Elles ne sont pas obligatoirement un corps HTML et peuvent donc revêtir des formats plus variés comme le RTF, XML ou CSV.

Comme pour les vues par défaut, la création d'une nouvelle vue nécessite un squelette et une méthode de remplissage (optionnelle : par défaut Doc::viewDefaultCard()). L'appel à cette vue peut se faire avec l'action IMPCARD de l'application FDL.

http://zoo.abc.com/what/?sole=Y&app=FDL&action=IMPCARD&**zone=FDL:VIEWBODYCARD**&id=1419

http://zoo.abc.com/what/?sole=Y&app=FDL&action=IMPCARD&**zone=FDL:VIEWSOCIETY**&id=1419

Deux options spéciales peuvent être utilisées lors de l'appel de la zone.

- **S**: signifie que le squelette est autonome, qu'il ne nécessite pas une encapsulation HTML. Cette option est obligatoire pour tous les squelettes non HTML.
- **T** : pour les squelettes HTML, il signifie qu'aucune entête ne sera affichée (pas de titre, ni d'icones). La barre de menu n'est pas affichée non plus
- **U** : idem T mais en plus pas de CSS définie (version > 2.9.2)
- **V** : pour les squelettes HTML, idem T mais la barre de menu est affichée (version > 2.9.2)
- **B** : pour les squelette non HTML binaire (tel que fichier openDocumentText .odt) (version > 2.9.2)

FDL:VIEWSOCIETY:	FDL:VIEWSOCIETY:T	FDL:VIEWSOCIETY:S

ZOO NET Toulouse cedex 8 31176 www.zoo.org 3 rue des girafes Zoo Net 09.89.33.34.90 nous 09.89.33.34.90 standard@zoo.org Vue par défaut avec entête.	Toulouse cedex 8 31176 www.zoo.org 3 rue des girafes Zoo Net 09.89.33.34.90 nous 09.89.33.34.90 standard@zoo.org L'entête n'est pas affichée.	Toulouse cedex 8 31176 www.zoo.org 3 rue des girafes Zoo Net 09.89.33.34.90 nous 09.89.33.34.90 standard@zoo.org Ici le document HTML est incomplet, puisque la balise racine est DIV. Il n'y a pas non plus de css, la représentation du document est brute.

3.9.4.1 Téléchargement de fichiers/données binaires

Dans le cas ou la vue doit transmettre un fichier au client vous pouvez utiliser les fonctions Http_DownloadFile() et Http_Download().

Ces fonctions permettent de générer le contenu de la réponse HTTP à partir du contenu d'un fichier, ou de donnés stocké dans une variable PHP, et de positionner les entêtes HTTP adéquats pour déclencher la demande d'enregistrement dans un fichier par le navigateur du client.

Exemples :

```
/**
 * Télécharger un fichier PDF local
 */
function downloadPdfFile($action) {

    [...]

    /*
     * Envoyer le fichier PDF `/tmp/xxx.pdf' au client
     * avec le nom `mon_fichier.pdf' et le type MIME `application/pdf'
     */
    Http_DownloadFile(
                      '/tmp/xxx.pdf',
                      'mon_fichier.pdf',
                      'application/pdf'
                      );

    exit(0);
}
```

```
/**
 * Télécharger un fichier PDF contenu dans une variable PHP
 */
function downloadPdfData($action) {

    [...]

    /*
     * Envoyer le contenu de $data avec le type MIME `application/pdf'
     * et le nom `mon_fichier.pdf'
     */
    Http_Download($data, 'pdf', 'mon_fichier', true, 'application/pdf');
```

```
    exit(0);
}
```

Important :

Il est important que le code exécuté après l'appel à ces fonctions n'écrivent pas sur la canal de sortie standard au risque de corrompre le contenu de la réponse HTTP générée.

Pour cela, vous pouvez faire un appel à la fonction exit() afin de mettre fin à l'exécution du code PHP après avoir envoyé votre fichier avec Http_DownloadFile() ou Http_Download().

3.9.5. Vues de consultation OpenDocument Text

(version > 2.9.2)

Les vues de ODT (OpenDocument Text), permettent de proposer à l'utilisateur un rendu avec plus de possibilités de composition que le rendu HTML. Cette vue est un fichier au format ODT. Celui-ci peut être produit par le traitement de texte OpenOffice.org Writer.

Ce fichier doit être placé, comme pour les squelettes HTML, dans le répertoire Layout de votre application.

Il faut utiliser la zone : `FDL:personne.odt:B`.

Ne pas oublier le **B** qui indique que le template est un fichier binaire.

La syntaxe pour les éléments de template est le même que pour le template HTML. Par contre les balises BLOCK/ENDBLOCK[32] et ZONE ne sont pas prises en compte.

Pour identifier une partie variable, il faut d'utiliser un « champ d'utilisateur openOffice » . Il faut éviter d'utiliser un texte brut car sinon il y a un risque que l'éditeur openOffice découpe le texte en plusieurs parties si vous retouchez cette partie de texte, le champ ne sera pas reconnu comme une variable. Si vous utilisez un champ utilisateur openOffice vous aurez alors une zone insécable qui sera reconnue plus efficacement par le moteur de template. Pour ajouter un champ utilisateur, aller sur le menu « Insertion/Champs/Autres... » puis aller sur l'onglet « Variables » et choisir « champ d'utilisateur », le format doit être mis à « texte ».

32) *la méthode ::setBlockData traditionnelle est par conséquent inopérante*

L'image ci-dessus montre l'insertion du champ variable « [TODAY] ». Le nom de la variable dans être mis dans le champ utilisateur « valeur » entre crochets. Le nom du champ utilisateur (ici « dateactuelle ») n'est pas utilisée par le moteur de template, il est néanmoins nécessaire à openOffice pour identifier la variable.

Exemple : Soit le document suivant :

En utilisant le template suivant :

On obtient le résultat suivant : `?app=FDL&action=FDL_CARD&zone=FDL:personne.odt:B&id=1342`

Dans le résultat produit, les attributs V_US_IDPRIVGROUP, ne sont pas affichés car leur visibilité est cachée (H - hidden).

3.9.5.1 Incorporation d'images

Pour incorporer des images, il faut insérer une image quelconque dans le fichier (menu Insertion/Image/A partir d'un fichier (sous OOo writer ⚠ pas de lien ni de copier/coller).

Pour référencer l'attribut image, cliquez sur le menu contextuel de l'image et choisissez 'image'; puis renseignez le nom dans l'onglet `Options` avec l'identifiant de l'attribut dynacase entre crochets.

En ce qui concerne la taille de l'image, la largeur sera conservée. La hauteur sera calculée en fonction du format de l'image pour ne pas avoir de déformation d'échelle.

3.9.5.2 Gestion des attributs de tableaux

Pour **les attributs inclus dans des tableaux**, indiquez la référence à la valeur dans la rangée d'un tableau déjà présent. Voir dans l'exemple précédent de l'attribut V_US_PRIVGROUP. Ces attributs peuvent aussi être utilisés dans des listes à puces ou numérotées. S'ils sont utilisés hors de ce contexte, la valeur retournée sera l'ensemble des valeurs séparées par une tabulation.

3.9.5.3 Gestion des conditions

Les conditions IF/ENDIF peuvent être utilisées en respectant la structure du document ODT : il faut que le niveau (structure XML du fichier OpenDocument Text) de la balise IF soit le même que le ENDIF. Par exemple, on ne peut pas mettre un IF au milieu d'un tableau et le ENDIF à l'extérieur du tableau : le IF et le ENDIF doivent être dans la même cellule du tableau. Il est possible de les utiliser dans le même paragraphe ou entre plusieurs paragraphes de même

niveau.

En cas d'erreur de structure, un message d'erreur apparaitra :

Comme pour les variables, il faut renseigner les conditions à l'aide de variables utilisateurs. Il est nécessaire de définir deux variables par condition : la variable de début de condition et la variable de fin de condition.

La condition IFNOT est aussi autorisée pour indiquer la condition inverse.

Ci dessous, l'exemple précédent avec une partie conditionnelle.

3.9.5.4 Attributs multivalués

Les attributs multivalués correspondent aux attributs qui sont dans un tableau ou qui ont l'option "multiple=yes" (cas des types "docid" et "enum").

Les attributs multivalués de premiers niveaux sont ceux qui sont dans un tableaux sans l'option "multiple=yes" ou qui sont hors tableau avec cette même option. Le seul attribut multivalués possible actuellement et le type docid qui peut avoir l'option "multiple=yes" et être dans un tableau. L'énuméré ne peut pas être multiple lorsqu'il est dans un tableau.

Si vous n'utilisez pas de contrôleur spécifique. Les différentes valeurs sont séparées par une virgule si elles ne sont pas déclarées dans une puce, un tableau ou une section. Sinon elles sont traités comme un répétable standard comme un attribut de tableau (voir paragraphe précédent).

Pour l'attribut multivalué de second niveau, il doit être sur une puce qui elle même doit se trouver dans une cellule de tableau si vous voulez voir le second niveau d'imbrication. Dans le cas contraire l'ensemble des valeurs sera affiché à la suite ; elles seront séparées par des virgules.

3.9.5.5 Utilisation d'un contrôleur spécifique

Il est possible d'utiliser une méthode particulière pour instancier le template comme pour les vues HTML. Le nom de la méthode est le nom en minuscule du fichier template sans extension. Exemple pour zone 'FDL:viewpersonne.odt:B', la méthode appelée sera 'viewpersonne'.

Comme pour les templates HTML, il est possible d'appeler la méthode '$this→lay→set()' afin de renseigner des clefs spécifiques. Ces clefs devront être dans des champs utilisateurs avec leur nom entre crochet.

Il est en plus possible d'utiliser la méthode OOoLayout::**setColumn** afin de renseigner une liste de valeurs pour être insérée dans une liste à puces ou dans un tableau. De plus la méthode OOoLayout::**setRepeatable**() peut être utilisée pour renseigner un tableau. (La méthode Layout::setBlockData() peut être utilisée avec un nom de block

vide pour des raisons de compatibilité mais est dépréciée).

Exemple :

```
/**
 * @templateController
 */
public function viewtest($target) {

  $t[]=array("V_X1"=>'A',
             "V_X2"=>'1',
             "V_X3"=>"La");
  $t[]=array("V_X1"=>'B',
             "V_X4"=>'2',
             "V_X3"=>"Si");
  $t[]=array("V_X1"=>'C',
             "V_X2"=>'3',
             "V_X3"=>"Do");

  $x=array('a','b','c','d');

  $this->lay->setColumn("V_XXX",$x);
  $this->lay->setRepeatable($t);
  }
```

Avec le template suivant : `viewtest.odt`

On obtient le résultat : (zone = `FDL:viewtest.odt:B`)

Il est aussi possible d'insérer un répétable à l'intérieur d'une section. Dans ce cas c'est la section englobante qui sera répétée.

Dans un contrôleur spécifique il est possible d'appeler le contrôleur par défaut afin d'accéder à l'ensemble des attribut avec les clefs V_<nom_d'attribut>. Pour cela il suffit d'ajouter l'appel à celui-ci (viewDefaultCard).

```
/**
 * @templateController
 */
public function viewtest($target) {
 // j'appelle le contrôle standard
   $this->viewDefaultCard($target) ;
   // Je fais mon contrôle spécifique
    $this->lay->set("NOW", $this->getDate()) ;
  }
```

En résumé, les répétables peuvent être mis sur :

- des listes à puces
- des listes numérotés
- des cellules de tableaux
- des sections

3.9.5.6 Répétables multi-niveaux

Dans un contrôleur spécifique il est possible d'avoir des imbrications de structures. Il est possible d'imbriquer des tableaux, des sections et des listes à puces ou numérotés.

Il est possible d'avoir des tableaux dans des tableaux, des tableaux dans des sections, des sections dans des tableaux, des listes dans des tableaux et des listes dans des sections. Il n'est pas possible d'avoir des listes et des sous/liste, ni d'avoir des tableaux dans des listes ni des sections dans des listes .

Exemple à deux niveaux d'imbrication :

```
/**
 * @templateController
 */
public function testfruit($target) {
```

```
    $this->lay->set("TODAY",$this->getTimeDate());

    $repeatColor[]=array("COLOR"=>"jaune",
                    "VEGETABLE"=>array("pomme","banane"),
                    "DESCR"=>array("pommier", "bananier"));
    $repeatColor[]=array("COLOR"=>"rouge",
                    "VEGETABLE"=>array("cerise","fraise"),
                    "DESCR"=>array("Cerisier","Fraisier"));
    $repeatColor[]=array("COLOR"=>"vert",
                    "VEGETABLE"=>array("concombre","cornichon","poivron"),
                    "DESCR"=>array("Plante potagère", "Condiment","Capsicum
annuum"));

    $this->lay->setRepeatable($repeatColor);

}
```

Avec le template odt suivant (testfruit.odt):

Nous obtenons :

17/12/2010 18:31
Test fruit

Les couleurs	Les fruits	
Couleur primaire jaune	Niveau 2	Description
	pomme	pommier
	banane	bananier
Couleur primaire rouge	Niveau 2	Description
	cerise	Cerisier
	fraise	Fraisier
Couleur primaire vert	Niveau 2	Description
	concombre	Plante potagère
	cornichon	Condiment
	poivron	Capsicum annuum

Autre exemple à trois niveaux avec des sections.

```
/**
 *@templateController
 */
function testfruit($target) {
    $this->lay->set("TODAY",$this->getTimeDate());
    $repeatColor[]=array("COLOR"=>"jaune",
                    "VEGETABLE"=>array("pomme","banane"),
                    "VARIETY"=>array(array("granny", "fuji","golden"),
                                array("plantain")));
    $repeatColor[]=array("COLOR"=>"rouge",
                    "VEGETABLE"=>array("cerise","fraise"),
                    "VARIETY"=>array(array("bigarreau","griotte"),
                                array("mara des bois","agate","anabelle")));
    $repeatColor[]=array("COLOR"=>"vert",
                    "VEGETABLE"=>array("concombre","cornichon","poivron"),
                    "VARIETY"=>array(array("gynial","Cucumis sativus"),
                                array("vert de paris"),
                                array("doux sonar","jericho")));


    $this->lay->setRepeatable($repeatColor);

}
```

Ici il faut bien imbriquer les sections pour être conforme à la structure du répétable défini dans le code :

- niveau 1 : COLOR
- niveau 2 : VEGETABLE
- niveau 3 : VARIETY

Et voici le résultat obtenu :

Nous avons réussi à avoir des paragraphes de différents niveaux qui sont imbriqués.

Il est aussi possible comme indiqué précédemment d'utiliser les tableaux ou les listes à puces.

3.9.5.7 Précautions d'usage

3.9.5.7.1 Affectation de variables

Lorsque vous utilisez un contrôleur spécifique il faut faire attention aux valeurs que vous affectez dans l'instance. En effet, les documents openDocument sont en XML et suivent une DTD stricte. Si la valeur que vous entrez n'est pas correcte d'un point de vue XML alors le résultat sera incorrect et le fichier openDocument ne pourra être produit.

Par exemple :

```
$a="Bob & Compagnie" ;
$this->lay->set("MY_VAR",$a) ; // erreur XML
```

produira une erreur car cela n'est pas correct en XML. Il faut utiliser

```
$a="Bob &amp ; Compagnie" ;
$this->lay->set("MY_VAR",$a) ;
```

Ceci aussi peut être réalisé par la méthode OooLayout ::eSet() :

```
$a="Bob & Compagnie" ;
$this->lay->eSet("MY_VAR",$a) ; // entité encodée
```

La classe fournie aussi la méthode pour encoder les entités XML :

```
$a=$this->lay->xmlEntities("Bob & Compagnie") ;
$this->lay->set("MY_VAR",$a) ; // correct car encodé auparavant
```

Si vous utilisez la méthode set vous pouvez insérer du XML correct pour ajouter des parties compatibles avec la DTD oasis. Ceci demande des connaissances sur la structure XML d'un fichier openDocument.

```
$a="Bob <text:line-break/> Compagnie" ;
$this->lay->set("MY_VAR",$a) ; // insertion d'un retour à la ligne
```

Il en va de même pour les méthodes setRepeatable ou setColumn il est nécessaire que les valeurs soient correctes d'un point de vue xml.

Si vous utilisez des variables provenant de document, il faut utiliser la méthode doc::getOooValue() qui produit directement le bon format xml en fonction du type d'attribut.

```
$a=$this->getValue("my_attr") ;
$this->lay->set("MY_ATTR",$a) ; // provoquera une erreur si $a est mal formé
```

Par contre cette utilisation fournira une valeur correctement formatée

```
$a=$this->getValue("my_attr") ;
$oooa=$this->getOooValue($this->getAttribute("my_attr"), $a) ;
$this->lay->set("MY_ATTR",$oooa) ; // $oooa est bien formé
```

Les variables déclarées dans le template qui ne sont pas instanciés sont laissées telles que (avec les crochets). De même si vous affectez une variable avec la valeur null cela sera considéré comme non instanciée.

```
$this->lay->set("MY_ATTR",null) ; // [MY_ATTR] non remplacé
$this->lay->set("MY_ATTR','') ; // [MY_ATTR] effacé
```

Les méthodes setColumn, setRepeatable, eSet sont spécifiques à la classe OooLayout et ne sont pas présentes dans la classe Layout utilisée pour les template ascii (html, csv). Suivant le type de fichier du modèle, l'instanciation peut se faire par l'une ou l'autre des classes. Il est préférable de s'assurer qu'on est bien dans un cas de modèle ODT au début du contrôleur.

```
public function my_viewSpecial($target) {
  if ($target!='ooo') {
    // ce n'est pas ce qui a été prévu
   addWarningMsg(sprintf("template %s not an odt",$this->lay->file) :
   return ;
  }
  // do staff..
}
```

3.9.5.7.2 Différence entre setColumn et setRepeatable

La méthode setRepeatable permet d'associer en une seule fois un ensemble de données répétées constituant une matrice (tableaux de tableaux). Cette méthode complète les colonnes de valeurs si certaines colonnes ont plus de valeur que d'autres. Par contre setColumn ne fait pas cette complétion, chaque variable est enregistrée individuellement. L'inconvénient de setRepeatable est qu'il ne sait pas gérer les répétables vides .

```
$this->lay->setRepeatable(array(array("V_N1"=>"A",
                                      "V_N2"=>"B"),
                                array("V_N1"=>"C",
                                      "V_N2"=>"D")));
```

L'exemple ci-dessus génère 2 rangées.

Mais pour générer 0 rangée on se retrouve avec un tableau vide sans référence à V_N1 ou V_N2. Le moteur de template considère alors que V_N1 et V_N2 ne sont pas renseignés et les laisse tels que.

Pour forcer la suppression du répétasble vous pouvez utiliser setColumn

```
$this->lay->setColumn("V_N1", array("A","C")) ; // génère 2 rangés
$this->lay->setColumn("V_N1", array()) ; génère 0 rangée (suppression de la
rangée/section/liste où se trouve V_N1
```

3.9.5.8 Mise à jour des propriétés du document

Si vous mettez des variables [V_XXX] dans les propriétés du document OpenOffice (titre, sujet, commentaires, propriétés personnalisées, etc ...). Ils seront pris en compte et remplacés à la génération.

3.9.5.9 Limitations

- Les répétables ne sont pas gérés dans les entêtes et pied de pages
- Les répétables ne sont pas gérés dans les propriétés du document
- les sections « tpl » ne sont pas gérés dans les entêtes et pied de pages
- Les mots-clefs BLOCK/ENDBLOCK ne sont pas gérés
- Les zones (ZONE) ne sont pas gérés
- Les attributs de type fichier, couleur ne sont pas gérés
- Les liens (hyperliens) vers les relations de document ne sont gérés. Seul le titre du document est affiché.
- Les listes ne peuvent pas avoir plusieurs niveaux d'imbrication.

Pour les attributs de type Htmltext :

- Les couleurs, taille de police, police des attributs htmltext ne sont pas pris en compte
- les images html ne sont pas prises en compte
- Pour la mise en forme, seuls les mises en gras, souligné et en italique sont supportés. Seuls les niveaux de titre de niveau 1 à 4 sont pris en compte.
- Si la clef est dans un paragraphe seul ou une cellule de tableau, sans texte autour, la mise en forme du paragraphe est ignoré mais il sera possible d'avoir un tableau ou des puces HTML.
- Si la clef est contiguë à du texte, le style de paragraphe est conservé mais les tables et les puces HTML seront vus comme du texte.
- Si la clef est mise dans une puce, les tableaux seront affichés comme du texte brut.
- Les clefs htmltext ne peuvent pas être insérées dans les entêtes et pied de pages. Il ne peuvent être utilisés non plus dans les propriétés du document.
- Les imbrications de paragraphes (exemple : Texte Texte) faites avec des balises div, p ou span (ou toutes autres balises de texte) ne sont pas supportées.
- Les styles (css) ne sont pas pris en compte.

Modèle	Fichier généré
Test·Style¶ Le·style·est·ignoré·pour·les·type·htmltext·lorsque·la·clef·est·seule·dans·le·paragraphe¶ [V_HTML_SINGLE]¶ ¶ **Test·Style·inline¶** Le·style·est·gardé·pour·les·type·htmltext·si·la·clef·n'est·pas·seule·dans·son·paragraphe.¶ Par·contre·dans·ce·cas,·les·tableaux·et·les·puces·HTML·ne·seront·affiché·sous·forme·de·texte.¶ Texte·:[V_HTML_SINGLE]¶	**Test·Style¶** Le·style·est·ignoré·pour·les·type·htmltext·lorsque·la·clef·est·seule·dans·le·paragraphe¶ Ligne·n°1¶ **Ligne·n°2·gras¶** *Ligne·n°3·italique¶* ¶ **Test·Style·inline¶** Le·style·est·gardé·pour·les·type·htmltext·si·la·clef·n'est·pas·seule·dans·son·paragraphe.¶ Par·contre·dans·ce·cas,·les·tableaux·et·les·puces·HTML·ne·seront·affiché·sous·forme·de·texte.¶ Texte·:Ligne·n°1 **Ligne·n°2·gras** *Ligne·n°3·italique¶*

- Dans l'exemple ci-dessus le texte "Texte :" qui est écrit dans le même paragraphe que ma clef implique que le mode d'insertion du htmltext passe en mode "inline", le style est conservé mais dans ce cas les éventuels puces ou tableaux seront affichés comme du texte brut.

3.9.5.10 Fichiers ODT mixtes

Les fichiers ODT mixtes sont des fichiers utilisés simultanément comme support rédactionnel pour l'utilisateur et comme vue (modèle ou template) au sens-dynacase-platform.

L'utilisateur modifie le fichier ODT d'un document dynacase (téléchargement et replacement ou via le Webdav).

Ce fichier ODT est contenu dans un attribut de type « file » du document. Ce fichier sera à la fois modifiable et calculé. Certaines parties du fichier seront calculées et d'autres parties seront modifiables par l'utilisateur.

Lors d'une modification du document dynacase, les zones dynamiquement liées à dynacase sont recalculées et le fichier ODT généré. Les modifications faites par l'utilisateur sont conservées. Ce fonctionnement est itératif.

Pour implémenter ce fonctionnement, il faut utiliser des éléments spécifiques d'Open Office pour que les données soient générées correctement.

Dans le cas d'un simple champ, il faut obligatoirement utiliser ce qui s'appelle dans OpenOffice une « liste de saisie » (Insertion / Champs / Autres ; Onglet Fonctions / Liste de saisie) :

Le nom du champ n'est pas modifiable par la suite (en cas d'erreur, il faut supprimer le champ et le recréer). Les champs doivent être nommés obligatoirement **[V_<NOM_DE_MON_CHAMP>]** en majuscule.

Pour produire des listes et des tableaux, les zones les contenant doivent être insérées dans des sections OpenOffice. Le nom de la section est obligatoirement préfixé par « **tpl_** ». Une section ainsi décrite sert de motif et est répétée par dynacase en fonction du nombre d'itération de la liste ou du nombre de ligne du tableau. A chaque génération, ce fonctionnement est reproduit. Cela signifie que les listes ou les tableaux sont entièrement reconstruits par dynacase.

Pour créer une section, utilisez le menu Insertion / Section.

Vous trouverez ci-dessous un exemple issu du zoo mettant en œuvre ces principes (une fiche résumé de l'animal avec ses enfants)

Modèle :

Le tableau est dans une section matérialisée par un fin trait vert. Les éléments en vert sont les champs de liste de saisie.

Le résultat obtenu est le suivant :

Gaspard

Date de naissance : 30/11/1995
Date d'entrée : 30/11/1995
Sexe : Masculin

Liste des enfants :

No	Date naissance	Photo
Leopold Antilope	24/06/2001	
Sarah Antilope	24/08/2001	
Julie Antilope	24/08/2001	

Si on regarde les sections dans le document généré :

On remarque bien la section du tableau qui a été cloné (_tpl_enfants) et l'original masqué (tpl_enfants).

3.9.5.11 Vues dynamiques dans les tableaux

Les fichiers générés peuvent être dans des tableaux. Il est possible d'avoir un contrôleur spécifique de template lorsqu'on déclare une méthode de même nom que l'attribut dans la famille.

Par exemple, on déclare les 3 attributs suivants dans un tableau (array) :

- tst_templated : file
- tst_date : date
- tst_comment : text

Si on veut afficher à la date et au commentaire correspondant au fichier, il faut utiliser l'index disponible depuis la zone utilisée pour le template. La méthode Lay::getZone() vous procure les éléments composant la zone :

Résultat de $this->lay->getZone() : ici appliqué à la première rangée du tableau.

```
Array
(
    [fulllayout] => THIS:tst_templated[0]:B
    [index] => 0
    [app] => THIS
    [layout] => tst_templated
    [modifier] => B
)
```

Dans cette structure, nous retrouvons l'index qui nous servira à récupérer la valeur de la date et du commentaire à l'index courant.

Le template odt contient des mots-clefs [V_DATEINDEX] et [V_TEXTINDEX] dans deux listes de saisies. Pour les instancier nous utiliserons le contrôleur suivant :

```
public function tst_templated()       {
  $index=$this->lay->getZone("index");
  $this->lay->set("V_DATEINDEX",$this->getOooValue(
                                  $this->getAttribute("tst_date"),
                                  $this->getTValue("TST_DATE",'no date',$index)
                                  ));
  $this->lay->set("V_TEXTINDEX",$this->getOooValue(
                                  $this->getAttribute("tst_comment"),
                                  $this->getTValue("tst_comment",'no comment',
$index)
                                  ));
  }
```

Cela va donc afficher la date et le commentaire sur le template dynamique correspondant à son rang dans le tableau.

3.9.5.12 Génération automatique du modèle de document

Si vous souhaitez que le fichier template soit automatiquement généré sur modification d'un des attributs du document, il faut préciser dans l'attribut file correspondant au fichier modèle l'option **template=dynamic**. Cela causera la mise à jour dynamique du template sur n'importe quelle modification du document (même via webdav).

3.9.6. Vues de consultation avec transformation

(version > 2.9.5)

Une transformation peut être appliquée après le composition de cette vue. Cette transformation est effectuée par le module de transformation. Pour appliquer une transformation pdf il suffit de spécifier le moteur à la fin de la zone comme ceci : `FDL:viewperson.odt:B:pdf`.

Si vous voulez faire une transformation pdfa (pdf pour archivage) il suffit de changer le moteur : `FDL:viewperson.odt:B:pdfa`. Vous avez alors le fichier pdf avec les polices.

Ceci indique à dynacase de lancer une demande de transformation (ici moteur pdf) à partir de la vue produite. L'utilisateur doit alors patienter le temps que la conversion soit effectuée.

On peut aussi lancer une vue pdf à partir d'un template HTML : exemple : `FDL:VIEWBODYCARD::pdf`. Dans ce cas le résultat risque de paraître un peu brut par rapport à la vue HTML. En effet le convertisseur PDF utilisé ne comprend pas toutes les possibilités des CSS utilisées. Par contre, pour ce type de rendu à partir de HTML, vous pouvez créer vos propre template en utilisant du HTML basique.

3.9.7. Vues d'édition

Les vues d'éditions sont forcément au format HTML. Les squelettes spécifiques d'édition sont inclus entre des balises FORM pour le formulaire d'envoi. C'est l'action GENERIC_EDIT ou FREEDOM_EDIT qui est utilisé pour afficher les formulaires d'édition.

La vue d'édition modifiable est celle affichée en jaune (par défaut FDL:EDITBODYCARD). Pour modifier la vue d'édition par défaut, il faut associé un fichier METHOD à la famille de document. Dans ce fichier, il suffit de modifier l'attribut $defaultedit pour lui assigner une autre zone.

```
<html>
  <head>
   <title>[TEXT:edition: ] [TITLE]</title>
<LINK REL="stylesheet" type="text/css"
HREF="[CORE_BASEURL]app=CORE&action=CORE_CSS" >
<LINK REL="stylesheet" type="text/css"
HREF="[CORE_BASEURL]app=FDL&action=FDL_CSS" >
[CSS:REF]
 </head>
<body class="freedom"
      onLoad="editOnLoad();autoHresize()"
      onsubmit="selectall()"
      onUnLoad="closechoose();pleaseSave(event) ;autoUnlock('[id]')"
      onResize="resizeInputFields()">

<form  id="fedit"
       class="fborder"
       name="modifydoc"
       onsubmit="document.isSubmitted=true;selectall();"
       method="POST" ENCTYPE="multipart/form-data"

action="[CORE_STANDURL]&app=[APPNAME]&action=GENERIC_MOD&id=[id]&dirid=[dirid]&c
lassid=[classid]" >
<table class="tableborder"  cellspacing="0" width="100%">
<thead>
```

```
<tr class="FREEDOMBack1">
 <td  colspan="2"><IMG   border="0"   alt="icon" align="absbottom" height="25px"
SRC="[iconsrc]">
   <span class="FREEDOMTextBigTitle">[TITLE]</span>
  </td>
 <td  align="right">
   <span class="FREEDOMText">[FTITLE]</span>
  </td>
</tr>

<tr class="FREEDOMBack2"><td  colspan="3"><img src="[IMG:1x1.png]"
border="0"></td></tr>
</thead>
</table>
[ZONE FDL:EDITCARD]

<table class="tableborder"  cellspacing="0" width="100%">
<tfoot>
[ZONE FDL:EDITSTATE]
<tr class="FREEDOMBack2"><td  colspan="3"><img src="[IMG:1x1.png]"
border="0"></td></tr>
<tr class="FREEDOMBack1">
  <td colspan="2" >
<input type="hidden" name="catgid" >
 <input id="iSubmit" type="submit" value="[editaction]"
onmousedown="multiple_for_select();" onclick="if (!canmodify()) return
false;document.isSubmitted=true;enableall();return true">
  </td>
  <td  align="right">
 <input type="button" value="[TEXT:Cancel]"
onclick="document.isCancelled=true;document.location.href='[CORE_STANDURL]&app=G
ENERIC&action=GENERIC_LOGO'">
  </td></tr>
</tfoot>
</table></form></body></html>
```

Comme pour les vues de consultation si la méthode n'existe pas, la méthode de remplissage utilisée sera Doc::viewDefaultCard(). Cette méthode ne construit pas de champ de saisie. Pour utiliser les variables L_<attr> et V_<attr> comme pour la méthode Doc:viewAttr(), on appelera la méthode Doc::editAttr() dans le méthode spécifique d'édition.

Le fichier Method.Society.php sera :

```
var $defaultedit= "FDL:EDITSOCIETY";

/**
 *@templateController
 */
function editsociety() {
  $this->editattr();
}
```

editsociety.xml	Représentation
<H1>TEXTE STATIQUE</H1>	ZOO NET société TEXTE STATIQUE Valider Annuler

```
<H1>TEXTE STATIQUE</H1><P>le numéro de
téléphone de [V_SI_SOCIETY] est
<B>[V_SI_PHONE]</P>
Autres renseignements :
<UL>
<UL>[L_SI_FAX] : [V_SI_FAX]</UL>
<UL>[L_SI_TOWN] : [V_SI_TOWN]</UL>
<UL>[L_SI_WEB] : [V_SI_WEB]</UL>
</UL>
```

```
<table width="100%" cols="2"><tr>
  <td style="background-color:moccasin">
    <table width="100%" >
      [ZONE FDL:EDITFRAME?frameid=SI_FR_IDENT]
    </table></td>
  <td style="background-color:burlywood">
    <table width="100%">
      [ZONE FDL:EDITFRAME?frameid=SI_FR_COORD]
    </table></td>
</tr></table>
```

Les méthodes d'édition particulières n'ont pas d'argument. Elles offrent les mêmes possibilités de représentation que les vues particulières de consultation. Ces méthodes doivent être utilisés dans des cas d'éditions qui ne sont pas prévus par les types d'attributs offerts par dynacase. Elles peuvent être aussi utilisées pour des documents avec beaucoup d'attributs afin de scinder différentes éditions du document.

Pour les attributs de type tableaux, il sont éditable par une zone comme pour les cadres. La syntaxe de la zone est la suivante : **[ZONE FDL:EDITARRAY?arrayid=***SI_T_SITES***]** par exemple pour le tableau SI_T_SITES.

Vue circulaire spécifique

```
viewsociety.xml <div style="height:[dy]px" border=1>
[BLOCK ATTR]
<P style="position:absolute;left:[x]px;top:[y]px" title="[frame]/[label]"> [value] </P>
[ENDBLOCK ATTR]
</div>
```

```
Method.society.php
public $defaultedit= "FDL:EDITSOCIETY";
/**
 * @templateController
 */
public function editsociety() {
  include_once("FDL/editutil.php");
  $rx=300; // rayon X
  $ry=200; // rayon Y

  $listattr = $this->GetInputAttributes();
  reset($listattr);
  $tattr=array();
  while (list($i,$attr) = each($listattr)) {
    $value = chop($this->GetValue($i));

    $tattr[]=array("value"=>getHtmlInput($this, $attr, $value),
    "label"=>$this->getLabel($i),
    "frame"=>$this->getLabel($attr->fieldSet->id) );
  }
  reset($tattr);
  $delta = 2*pi()/count($tattr);
  while (list($k,$v) = each($tattr)) {
    $tattr[$k]["y"]=$ry+sin($delta*($k))*$ry+20;
```

```
    $tattr[$k]["x"]=$rx+cos($delta*$k)*$rx+20;
  }
  $this->lay->setBlockData("ATTR", $tattr);
  $this->lay->set("dy",$ry*2);
  $this->lay->set("ry",$ry/2);
}
```

Comme pour les vues de consultation, les zones définissant les éditions peuvent avoir des options de représentation.

option	Définition	Représentation
	Édition par défaut zone=FDL:EDITBODYCARD	Zoo Net Identification Raison sociale : Zoo Net Description : Parc animalier Coordonnées Mail : Téléphone : 01 02 03 04 01 Fax : 01 02 03 04 00 Adresse : 3 rue des girafes Code postal : Ville : Toulouse Cedex : Web : animaldiversity.ummz.umich.edu Sauver Annuler
S	Les entêtes et les boutons «” sauver “» et «” annuler “» ne sont pas affichés. Par contre le document contient bien le formulaire (balise <form>) pour l'édition. Par contre les boutons pour les changements d'état sont présents lorsqu'il y a un cycle de vie. zone=FDL:EDITBODYCARD:S	Identification Raison sociale : Zoo Net Description : Parc animalier Coordonnées Mail : Téléphone : 01 02 03 04 01 Fax : 01 02 03 04 00 Adresse : 3 rue des girafes Code postal : Ville : Toulouse Cedex : Web : animaldiversity.ummz.umich.edu
V	Idem S sauf que les boutons pour le changement d'état ne sont pas affichés.	
U	Le squelette est affiché de manière brute : pas de formulaire prédéfini. zone=FDL:EDITBODYCARD:U	Identification raison sociale : Zoo Net description : Parc animalier Coordonnées mail : téléphone : 01 02 03 04 01 fax : 01 02 03 04 00 adresse : 3 rue des girafes code postal : ville : Toulouse cedex : web : animaldiversity.ummz.um
T	L'entête n'est pas affiché. Les boutons «” sauver “» et «” annuler “» sont affichés zone=FDL:EDITBODYCARD:T	Identification Raison sociale : Zoo Net Description : Parc animalier Coordonnées Mail : Téléphone : 01 02 03 04 01 Fax : 01 02 03 04 00 Adresse : 3 rue des girafes Code postal : Ville : Toulouse Cedex : Web : animaldiversity.ummz.umich.edu Sauver Annuler

3.9.8. Interface spécifique de contrôle d'un formulaire document

Des aides spécifiques sur un formulaire documentaire peuvent être mises en places. Ces aides permettent de remplir des attributs en utilisant une interface plus spécialisée.

Ces aides sont déclarées dans le fichiers de description des familles comme pour les aides à la saisies classiques. La différence est quelles font références à une interfaces dédiée et à un contrôleur d'interface dédiés.

Exemple :

	idattr	idframe	label	type	vis	phpfile	phpfunc
ATTR	EN_T_ESPECES	EN_IDENTIFICATION	liste espèce	array	W	zoo.php	ZOO:zoo_searchspecies(A,D,ID,en_nom):
ATTR	EN_ESPPROTECTED	EN_T_ESPECES	protégé	image	S		
ATTR	EN_ESPECE	EN_T_ESPECES	espèce	docid(“ZOO_ESPECE”)	W	zoo.php	ZOO:zoo_searchspecies(A,D,ID,en_nom):en_espece,en_comment
ATTR	EN_COMMENT	EN_T_ESPECES	commentaire	text	W		
ATTR	EN_CAPACITE	EN_IDENTIFICATION	capacité	int	W		

La notation 'ZOO:zoo_searchspecies(A,D,ID,en_nom):' indique l'interface est le fichier `zoo_searchspecies.xml` et le contrôleur la fonction `zoo_searchspecies()` du fichier `zoo.php`. Puisqu'il n'y a pas de retour défini (après les :), cela indique que c'est l'interface qui explicitement modifiera les attributs voulus.

Fichier zoo.php :

```
function zoo_searchspecies(&$action,$dbaccess,$id,$nom) {
   // print "DB=$dbaccess, NOM=$nom ID=$id";
    $action->lay->set("enclosname",$nom);
    $doc=new_doc($dbaccess,$id);

    if ($doc->isAlive()) {
        $action->lay->set("CAPACITY",$doc-
>getValue("en_capacite",_("zoo:Capacity not set")));
    } else {
        $action->lay->set("CAPACITY",_("zoo;Capacity not set"));
    }
}
```

Dans la fonction contrôleur on retrouve les paramètres déclarés dans la famille (A ⇒ &$action, D⇒ $dbaccess, ID⇒$id, en_nom⇒$nom). Pour piloter le template, il est nécessaire de déclarer le paramètre action (A). Si le template n'a pas besoin de contrôleur, il est néanmoins nécessaire de déclarer la fonction dans le fichier indiqué dans `phpfile` même si elle ne fait rien. Fichier zoo_searchspecies.xml

```
[ZONE FDL:HTMLHEAD?title=[TEXT:Special Species]]
<script type="text/javascript" src="lib/data/fdl-data.js"></script>
<script>
        var C=null; // the main context

        function searchSpecies() {
                C=new Fdl.Context();
                if (! C.isConnected()) {
                        alert('error connect:'+C.getLastErrorMessage());

                }

                if (C){
```

```
                        var sel = document.getElementById("resultTab");
                        var d = C.getSearchDocument();
                        var f=new Fdl.DocumentFilter({family:'ZOO_ESPECE',
                                                      criteria:[{operator:'~*',
                                                                 left:'title',

right:document.getElementById('specie').value}]});
                var dl=d.search({filter:f});
                        var content=dl.getDocuments();
                        for (var i in content) {
                                var doc=content[i];
                                if (doc && doc.isAlive()) {
                                        var o = new
Option(doc.getTitle(),doc.getProperty('id'));
                                        sel.options[sel.options.length]=o;
                                }
                        }
                }
        }

        function getReference() {
           var ref=Ih.docGetFormValue('en_reference');

           return ref;
        }
        function insertDataFromSelection() {
           var titles=[];
           var icons=[];
           var ids=[];

           var sel=document.getElementById("resultTab");
           for (var i=0;i < sel.options.length;i++) {
                  if (sel.options[i].selected){
                    var d = C.getDocument({id:sel.options[i].value,
useCache:true});
                    console.log(d);
                    Ih.docAddTableRow({en_espece:
{id:sel.options[i].value,title:sel.options[i].text,unique:true},
                                       en_comment:'coucou',
                                       en_espprotected:
(d.getValue('es_protegee')=='1')?{url:'Images/pandared.png',id:'pandared.png'}:
{url:'Images/pandagreen.png', id:'pandagreen.png'},
                                       en_photo:{url:d.getDisplayValue('es_photo',
{url:true}),id:d.getValue('es_photo')}});
                   }
           }
        }

        addEvent(window,"load",autoWresize);
        addEvent(window,"load",searchSpecies);

        function setCapacity(inp,delta) {
           if (isNaN(inp.value)) inp.value=0;
           if (delta) inp.value=parseInt(inp.value)+delta;
           Ih.docSetFormValue({en_capacite:inp.value})
        }
</script>
<h3><b>[TEXT:zoo:Species for] : [enclosname] </b></h3>
<input type="text" id="specie"/>
<input type="button" value="[TEXT:zoo:Search species]"
onclick="searchSpecies()"></input>
<br/>
```

```
<select id="resultTab" multiple="multiple">
</select>
<br/>

<input type="button" value="[TEXT:Insert into list !]"
onclick="insertDataFromSelection()"></input>
<br/>
<input id="capacity" value="[CAPACITY]"/>
<input type="button" value="+"
onclick="setCapacity(document.getElementById('capacity'),+1)"></input>
<input type="button" value="-"
onclick="setCapacity(document.getElementById('capacity'),-1)"></input>
<input type="button" value="[TEXT:zoo:Change capacity]"
onclick="setCapacity(document.getElementById('capacity'),null)"></input>

<input type="button" value="[TEXT:zoo:Retrieve reference]"
onclick="document.getElementById('reference').value=getReference()"></input>
<input type="text" id="reference"/>

[ZONE FDL:HTMLFOOT]
```

Ce template doit comporter les headers HTML standard de dynacase afin d'incorporer automatiquement les scripts nécessaires au contrôle du formulaire. Pour cela il est fortement conseillé d'utiliser la zone `[ZONE FDL:HTMLHEAD]`.

Le résultat par l'image :

Après avoir appuyé sur le bouton “espèces” :

Après avoir appuyé sur le bouton “insérer dans la liste !” :

3.9.8.1 Taille de la fenêtre modale

La fenêtre qui est ouverte lorsqu'on clique sur le bouton espèce peut être précisée au moyen des options *mheight* et *mwidth.*

Cette interface permet d'illustrer 3 aspects des interfaces d'aides à la saisie :

3.9.8.2 Manipulation de tableau

L'interface dispose de la fonction javascript `Ih.docAddTableRow` permettant d'ajouter une rangée à un tableau. Cette fonction a comme argument un objet de configuration avec comme index le nom de l'attribut et comme valeur la valeur de l'attribut. Cette fonction retourne true si l'ajout a été effectuée.

Deux types ont comme valeur un objet : les types docid et les types “images”.

3.9.8.2.1 Type docid

Le docid doit déclarer son identifiant et son titre.

```
Ih.docAddTableRow({en_espece:{id:9,title:'racine',unique:true});
```

De manière optionnelle, il est possible d'ajouter l'argument `unique` afin d'indiquer que cette colonne ne peut avoir 2 rangée avec le même identifiant. L'option `unique` ne peut être utilisé que pour une seule colonne 'docid' du tableau.

3.9.8.2.2 Type image

Uniquement si l'image est déclarée en static (visibilité S).

L'image doit déclarer son url et son identification interne. L'identification interne est une référence au coffre. Cette référence est obtenu par la valeur d'un attribut image déjà existant.

```
Ih.docAddTableRow(en_imgstatic:{url:d.getDisplayValue('es_photo',
{url:true}),id:d.getValue('es_photo')});
```

Il est aussi possible de déclarer une image qui est disponible sur le serveur. Cette image aura pu être déposée par votre module (répertoire images). L'id doit contenir le nom du fichier images (sans chemin) et l'url doit être Le nom précédé du chemin relatif `Images`.

```
Ih.docAddTableRow(en_imgstatic:
{url:'Images/confidential.gif',id:'confidential.gif'});
```

3.9.8.2.3 Type file

Pas implémenté.

3.9.8.2.4 Autre type

On mets directement la valeur :

```
  Ih.docAddTableRow({en_reference:'X345',
                en_comment:'coucou',
                en_enumcolor:'c'});
```

Pour les énumérés il faut mettre la clef.

3.9.8.2.5 Exemple complet

```
Ih.docAddTableRow({en_espece:
{id:sel.options[i].value,title:sel.options[i].text,unique:true},
                en_comment:'coucou',
                en_imgstatic:{url:d.getDisplayValue('es_photo',
{url:true}),id:d.getValue('es_photo')},
                en_enumcolor:'c'});
```

3.9.8.3 Affectation d'attribut

Il est possible de modifier unitairement un attribut mono-valué dans le formulaire. Les attributs multi-valués ne sont pas pris en comptes (attribut dans les tableaux, attribut avec option `multiple=yes`). Cette fonction return true si l'affectation du formulaire a réussi.

```
    Ih.docSetFormValue({en_capacite:'20'});
    Ih.docSetFormValue({en_gardien:{id:6789,title:'Jean Garde'});
```

3.9.8.4 Récupération de valeur d'attribut

Il est aussi possible de récupérer les valeur d'attribut du formulaire. Cette fonction retourne la valeur de l'attribut. Si cette valeur est multi-valués , cela retourne un tableau de valeur.

```
var ref=Ih.docGetFormValue('en_reference');
```

3.9.8.5 Utilisation des attributs de retour définis dans l'aide à la saisie

Il est possible d'utiliser le retour défini dans l'aide à la saisie si on veut faire des interfaces d'aide générique et réutilisable. Pour cela il suffit d'utiliser la fonction Ih.getReturnAttribute(index) qui retourne l'attribut à sa position indiqué (0 est la première position). Retourne null si position incorrecte. Exemple d'utilisation avec l'aide définie sur l'attribut 'en_espece'.

```
function insertDataUsingReturnAttribute() {
 var titles=[];
 var icons=[];
 var ids=[];
 var sel=document.getElementById("resultTab");
 for (var i=0;i < sel.options.length;i++) {
 if (sel.options[i].selected){
 var d = C.getDocument({id:sel.options[i].value, useCache:true});
 var row={};
 // get first result attribute : the attribute 'en_espece'

row[Ih.getReturnAttribute(0)]={id:sel.options[i].value,title:sel.options[i].text
,unique:true};

 // get second result attribute: the attribute 'en_comment'
 row[Ih.getReturnAttribute(1)]='a comment';
 Ih.docAddTableRow(row);
 }
 }
 }
```

3.9.8.6 Adaptation à l'environnement Ext JS

Si on désire que la fenêtre modale prenne le style par défaut de Ext JS lors de son utilisation dans ECM, il faut ajouter dans la vue un conteneur global contenant la classe HTML "document".

Exemple :

```
<div class="document">
... contenu de votre vue
</div>
```

Le style sera automatiquement adapté que l'on soit sous ECM ou en mode standard.

3.9.9. Mise en forme de rangée de tableau

Version > 2.11.2 En test

Lorsque l'on souhaite une mise en forme particulière des lignes d'un tableau (effet de présentation ou nombre de colonne important), il est possible d'utiliser un template XML pour spécifier cette mise en forme.

Pour indiquer qu'un tableau a une mise forme spécifique, mettre l'option `rowviewzone` (pour la vue de consulation) ou `roweditzone` (pour la vue d'édition) dans la définition de l'attribut de type array. Pour utiliser le même template de ligne, il faut déclarer les 2 options avec le même template.

Le template doit fournir un tableau xml <table>....</table> contenant :

- une section **<table-head>...</table-head>** utilisée pour la ligne de titre
- une section **<table-body>...</table-body>** qui est répétée pour chacune des lignes.

Comme un exemple vaut mieux qu'un long discours. Voici un exemple applicable sur la famille “contrôle de vue” CVDOC.

3.9.9.1 Exemple de consultation

id	définition	type	vis	option
CV_T_VIEW	vues	array	W	rowviewzone=FDL:ROWVIEWEDITCV

l'option référence le fichier template se trouvant sur le serveur : /usr/share/what/**FDL**/Layout/**rowvieweditcv**.xml

L'éventuelle méthode rowvieweditcv n'est pas appelée pour renseigner le template comme cela est le cas des templates de document. Les définitions des zones dynamiques sont statiques.

Dans le template XML, on peut utiliser

- les paramètres globaux (exemple [COLOR_B6]),
- les libellés des attributs - commençant par L_ (exemple [L_CV_MSK]),
- les valeurs formatés HTML commençant par 'V_ (exemple [V_CV_KVIEW])
- les valeurs brutes (exemple [CV_KVIEW])
- les traductions commençant par TEXT: (exemple : [TEXT:Mask and zone])

afin d'éviter que les cellules ne contenant pas de texte soient “écrasées”, vous pouvez y mettre un espace insécable. Cependant, dans ce cas, il faut absolument déclarer cette entité dans le fichier xml pour qu'il soit correctement encodé: ajoutez

```
<!DOCTYPE root [
<!ENTITY nbsp " ">
]>
```

juste après l'en-tete xml

Pour éviter des problèmes de présentation sur les docid de famille, il est conseillé d'utiliser

```
style="white-space:nowrap;"
```

```
<?xml version="1.0"?>
<table>
  <table-head>
    <cell style="background-color:[COLOR_B5]">[TEXT:Identification]</cell>
    <cell style="background-color:[COLOR_C3]">[L_CV_KVIEW]</cell>
    <cell style="background-color:[COLOR_C2]">[TEXT:Mask and zone]</cell>
    <cell style="background-color:[COLOR_C3]">[L_CV_ORDER]</cell>
    <cell style="background-color:[COLOR_C2]">[L_CV_DISPLAYED]</cell>
  </table-head>

  <table-body>
    <cell style="background-color:[COLOR_B6]">
      <b>[V_CV_IDVIEW]</b><br/>[V_CV_LVIEW]
    </cell>
    <cell style="background-color:[COLOR_C5];text-align:center">
      [V_CV_KVIEW]
    </cell>
    <cell style="background-color:[COLOR_C4]">
      <table class="transparent" width="100%" cellspacing="0" cellpadding="0">
        <tr>
           <td width="100px">
            [L_CV_MSK] <img resize="1" align="absbottom" width="20px"
src="[IMG:masque.gif]"/></td>
            <td>[V_CV_MSK] [V_CV_MSKID]</td>
         </tr><tr>
            <td>[L_CV_ZVIEW] :</td>
            <td>[V_CV_ZVIEW]</td>
        </tr>
      </table>
```

```
    </cell>
    <cell style="background-color:[COLOR_C5];vertical-align:middle">
      [V_CV_ORDER]
    </cell>
    <cell style="background-color:[COLOR_C4];vertical-align:middle">
      [V_CV_DISPLAYED]
    </cell>
  </table-body >
</table>
```

En ce qui concerne les éléments de style (CSS). Les attributs class et style des balises `cell` sont transmises dans les balises `td` du HTML produit. Le contenu des balises `cell` doit être du **XHTML** (pas de balises non fermés).

Avant : (sans zone spéciale) :

Après :

3.9.9.2 Exemple d'édition

id	définition	type	vis	option
CV_T_VIEW	vues	array	W	roweditzone=FDL:ROWVIEWEDITCV

Dans ce cas on reprend le même template XML.

Avant :

contrôle de vues
DÉTAIL DES RECHERCHES ÉVÈNEMENT RECHERCHE ÉVÉNEMENT
Sauver | Annuler

Basique
Titre : Détail des recherches évènement
Description :
Famille : recherche événement

vues

id vues	label	type	zone	masque	ordre	affichable
detail	Détails	Edition	FRE			☑
res	Par ress	Consultation	FRE			☐
evt	Par évé	Consultation	FRE			☐

Vues par défauts
Création vue :

dynamique
Famille :

Après :

contrôle de vues
DÉTAIL DES RECHERCHES ÉVÈNEMENT RECHERCHE ÉVÉNEMENT
Sauver | Annuler

Basique
Titre : Détail des recherches évènement
Description :
Famille : recherche événement

vues

Identification	Type	Mask and zone	Ordre	Affichable
detail Détails	Edition	Masque : Zone : FREEDOM:EDITDSEARCH		☑
res Par ressources	Consultation	Masque : Zone : FREEEVENT:PLANNER?by		☐
evt Par événement	Consultation	Masque : Zone : FREEEVENT:PLANNER?by		☐

Vues par défauts
Création vue :

dynamique
Famille :

3.9.9.3 Pour éditer et consulter avec le même template

Dans la colonne “option”, mettre :

```
roweditzone=FDL:ROWVIEWEDITCV|rowviewzone=FDL:ROWVIEWEDITCV
```

Note :

Le contenu des cellules est reconstitué à partie de la DOM XML. Les attributs des balises peuvent être transformées par leur forme équivalente.

Le code suivant

```
<span title='test "ok"'>OK</span>
```

sera reconstituée de la manière suivante :

```
<span title="test &quot;ok&quot;">OK</span>
```

3.9.10. Vues d'attributs

Les vues d'attributs permettent de modifier la présentation d'un attribut en consultation ou en édition. La définition des ces vues est basée sur les vues de documents.

3.9.10.1 Vue d'attributs de consultation

Les vues de consultation sont déclarées avec l'option `viewtemplate`. La valeur de cette option est une zone (APPNAME::LAYOUTNAME).

Ces vues ne sont applicables que sur les vues de consultations par défaut (FDL:VIEWBODYCARD): pas les vues résumés, ni les vues spécifiques, ni les zones FDL:VIEWFRAME. Cette option n'est pas applicable aux types 'tab' ni aux types 'array' ni aux attributs contenus dans un tableau. Pour les templates de tableaux se reporter au chapitre "Mise en forme de rangée de tableau".

Dans les templates il est possible d'utiliser les clefs d'accès aux valeurs d'attributs ([V_ATTRNAME]) et aux libellés ([L_ATTRNAME]) comme pour les vues de documents. S'il y a une méthode du nom de la vue celle-ci sera aussi appelée. Vous pouvez alors avoir accès à des clefs personnalisées sur le template. L'accès au pilotage du template se fait comme pour les vues de documents avec l'objet `$this→lay`.

Si la vue comporte l'option 'S' alors le libellé de l'attribut ne sera pas affiché et le template prendra toute la largeur du document, sinon seule la partie affichage de la valeur sera substituée.

Il est bien sûr possible dans un template d'afficher plusieurs attributs. Si c'est le cas il est possible de mettre le template none (viewtemplate=none) pour les attribut déjà affiché pour ne pas afficher deux fois un même attribut. Cela est différent du fait de mettre la visibilité à 'H' car cela empêcherais aussi de la voir dans template spécifique.

Exemple

	idattr	idframe	label	type	option
ATTR	AN_IDENTIFICATION		Identification	frame	
ATTR	AN_NOM	AN_IDENTIFICATION	nom	text	viewtemplate=ZOO:ANIMALNAME
ATTR	AN_TATOUAGE	AN_IDENTIFICATION	tatouage	int	viewtemplate=ZOO:ANIMALTATOO:S

Dans cet exemple les attributs `an_nom` et `an_tatouage` ont des vues spécifiques. La vue `ZOO:ANIMALNAME` référence le fichier `animalname.xml` de l'application ZOO. Fichier `animalname.xml` :

```
<p style="color:green">[TEXT:zoo:It is a beautifull name for this animal]</p>
[V_AN_NOM]
```

Fichier `animaltatoo.xml` :

```
<p style="color:red"><b>[L_AN_TATOUAGE]</b> : [TEXT:zoo:The tatoo is
important] : [V_AN_TATOUAGE]</p>
```

Sans les zones d'attributs:

Avec les zones d'attributs :

Il est aussi possible de définir une zone pour les attributs de type `frame`. La contenu du cadre entier sera alors remplacé par le template. Le cadre reste quant-à lui affiché avec son libellé (mettre en plus l'option vlabel=none pour ne pas voir le libellé).

3.9.10.2 Vue d'attributs d'édition

Les vues de consultation sont déclarées avec l'option `edittemplate`. La valeur de cette option est une zone (APPNAME::LAYOUTNAME).

Ces vues ne sont applicables que sur les vues d'édition par défaut (FDL:EDITBODYCARD) et sur l'interface de saisie des paramètres de transitions.

Dans les templates il est possible d'utiliser les clefs d'accès aux champs du formulaire des attributs ([V_ATTRNAME]) et aux libellés ([L_ATTRNAME]) comme pour les vues d'édition de documents. S'il y a une méthode du nom de la vue celle-ci sera appelée à la place du contrôle par défaut. Si vous souhaitez aussi avoir le contrôle par défaut pour accéder aux clefs d'attribut, il sera nécessaire d'ajouter l'appel à ce dernier :

```
/**
 *@templateController
 */
public function specificControl($target) {
  $this->viewdefaultcard($target); // call to default controller
  $this->lay->set("aspecial", "very special");
}
```

. Vous pouvez alors avoir accès à des clefs personnalisées sur le template. L'accès au pilotage du template se fait comme pour les vues de documents avec l'objet `$this→lay`.

Si la vue comporte l'option 'S' alors le libellé de l'attribut ne sera pas affiché et le template prendra toute la largeur du document, sinon seule la partie affichage de la valeur sera substituée. Si la vue comporte l'option 'U' alors l'input utilisé avec le mot-clef [V_ATTRNAME] prendras la largeur disponible. Cette option est à mettre si le template ne comporte qu'un seul champ input.

Il est bien sûr possible dans un template d'afficher plusieurs attributs. Si c'est le cas il est possible de mettre le template none (viewtemplate=none) pour les attribut déjà affichés pour ne pas afficher deux fois le même champ de formulaire ce qui provoquerai un mauvais enregistrement du document.

Exemple

	idattr	idframe	label	type	ord	vis	option
ATTR	AN_IDENTIFICATION		Identification	frame	0	W	
ATTR	AN_NOM	AN_IDENTIFICATION	nom	text	10	W	edittemplate=ZOO:ANIMALNAME:U
ATTR	AN_TATOUAGE	AN_IDENTIFICATION	tatouage	int	20	W	edittemplate=ZOO:ANIMALTATOO:S

Dans cet exemple les attributs `an_nom` et `an_tatouage` ont des vues spécifiques. La vue `ZOO:ANIMALNAME` référence le fichier `animalname.xml` de l'application ZOO. Fichier `animalname.xml` :

```
<p style="color:green">[TEXT:zoo:It is a beautifull name for this animal]</p>
[V_AN_NOM]
```

Fichier `animaltatoo.xml` :

```
<p style="color:red"><b>[L_AN_TATOUAGE]</b> : [TEXT:zoo:The tatoo is
important] : [V_AN_TATOUAGE]</p>
```

Sans les interfaces d'attributs :

Sauver Annuler

Identification

Nom : Arthur

Tatouage : 23545

Espèce : Antilope

Sexe : Masculin

Photo :

Parcourir...

Date naissance :

Date entree :

liste enfant

enfant

Totor Alligator

Avec les interfaces d'attributs :

Sauver Annuler

Identification

Nom : C'est un joli nom pour cet animal

Arthur

tatouage : La tatouage est important : 23545

Espèce : Antilope

Sexe : Masculin

Photo :

Parcourir...

Date naissance :

Date entree :

liste enfant

enfant

Totor Alligator

Il est aussi possible d'avoir des interfaces d'éditions plus spécifiques. Pour les attributs non multivalués, vous pouvez utiliser la fonction 'setFormValue' pour affecter des valeurs d'attributs dans le formulaire. Il est aussi possible d'utiliser la fonction `addTableRow` pour ajouter une rangée à un type tableau (voir manipulation_de_tableau).

```
[V_AN_TATOUAGE]
<a onclick="setFormValue({an_tatouage:4567})">Set tatoo to 4567</a> ;
```

3.9.11. Interfaces spécifiques d'administration

Ces interfaces sont des zones permettant l'affichage et la modification des paramètres de famille ou d'application.

Pour pouvoir les utiliser, l'intégrateur devra créer une action spécifique intégrant les zones. Cette action devra avoir une acl spécifique afin de n'être accessible que via les administrateurs. Il devra en outre mettre la balise [JS:REF] dans la balise <head> de sa page.

Chaque zone sera représentée par un fragment HTML de type DIV autonome.

3.9.11.1 La zone EDITFAMILYPARAMETER

Cette zone est utilisée pour modifier les paramètres d'une famille.

En l'ajoutant vous pourrez modifier tout paramètre de famille, quel que soit son type.

Elle créera un champ input de forme appropriée (texte, select, array...) en fonction du type de paramètre à changer.

Ce champ aura pour id l'identifiant du paramètre à modifier, et comme name le même identifiant précédé d'un _

Exemple de champ input généré pour un paramètre dont l'id est "ent_prixenfant" :

```
<input type="text" id="ent_prixenfant" value="10" name="_ent_prixenfant"
class="fullresize" onchange="document.isChanged=true">
```

La zone est encapsulée dans un élément HTML de type DIV, portant la classe "editfamilyparameter" et possédant l'attribut "data-parameter" qui a pour valeur l'identifiant du paramètre qui sera affiché/modifié.

Le label lié au formulaire aura pour valeur le label du paramètre décrit par l'attribut "attrid"

3.9.11.1.1 Description des paramètres

```
[ZONE FDL:EDITFAMILYPARAMETER ?famid=<famille name> &attrid=<attr
id>&emptyValue=<valeur si valeur vide> &value=<présentation d'une valeur
différente> &submitOnChange=[yes]|no&localSubmit=yes|[no]&submitLabel=<label du
bouton>]
```

Si "submitOnChange" est à yes, la modification sur le onChange (onBlur suivant input) est prise en compte. Par défaut submitOnChange est à yes .
Si localSubmit est à yes, un bouton submit est affiché. Par défaut localSubmit est à no
Si submitLabel existe, il sera pris en compte comme libellé du bouton(qui n'est affiché que si localSubmit est à yes). Par défaut le label est "Valider".

3.9.11.1.2 Exemple

```
        <div>
            <h1>Paramètrage du tarif d'entrée au zoo pour les enfants</h1>
            [ZONE FDL:EDITFAMILYPARAMETER?
famid=ZOO_ENTREE&attrid=ENT_PRIXENFANT&localSubmit=yes&submitOnChange=no]
        </div>
```

Paramètrage du tarif d'entrée au zoo pour les enfants

Tarifs des entrées pour les enfants (en euros)

10 Valider

Code html généré :

```
<div>
     <h1>Paramètrage du tarif d'entrée au zoo pour les enfants</h1>
      <div data-parameter="ent_prixenfant" class="editfamilyparameter">
           <div data-type="label">
                 <label for="ent_prixenfant">Tarifs des entrées pour les
enfants (en euros)</label>
           </div>
           <div data-type="field">
                 <form data-on-change="" action="" method="POST">
                       <input type="hidden" id="fam_ent_prixenfant"
value="ZOO_ENTREE">
                       <input type="text" id="ent_prixenfant" value="10"
name="_ent_prixenfant" class="fullresize" onchange="document.isChanged=true">
                       <input type="submit" name="ent_prixenfant"
onclick="sendParameterData(this, true)" value="Valider">
                 </form>
           </div>

     </div>
</div>
```

3.9.11.2 La zone EDITAPPLICATIONPARAMETER

Cette zone est utilisée pour modifier les paramètres d'une application.

En l'ajoutant vous pourrez modifier tout paramètre de famille, quel que soit son type.

Elle créera un champ input de type texte ou de type select, suivant le type du paramètre à modifier.

Ce champ aura pour id l'identifiant du paramètre à modifier, et comme name le même identifiant précédé d'un _

Exemple de champ input généré pour un paramètre dont l'id est "ONEFAM_DISPLAYMODE" :

```
<select id="ONEFAM_DISPLAYMODE" name="_ONEFAM_DISPLAYMODE">
     <option value="html">html</option>
     <option selected="selected" value="extjs">extjs</option>
     <option value="ng -beta-">ng -beta-</option>
 </select>
```

La zone est encapsulée dans un élément HTML de type DIV, portant la classe "editapplicationparameter" et possédant l'attribut "data-parameter" qui a pour valeur l'identifiant du paramètre qui sera affiché/modifié.

Le label lié au formulaire aura pour valeur le label du paramètre décrit par l'attribut "parameterid"

3.9.11.2.1 Description des paramètres

```
[ZONE FDL:EDITAPPLICATIONPARAMETER?appId=<nom de l'application>&parameterId=<nom
du paramètre>&submitOnChange=[yes]|no&localSubmit=yes|[no]&submitLabel=<label du
bouton>]
```

Si "submitOnChange" est à yes, la modification sur le onChange (onBlur suivant input) est prise en compte. Par défaut submitOnChange est à yes .
Si localSubmit est à yes, un bouton submit est affiché. Par défaut localSubmit est à no
Si submitLabel existe, il sera pris en compte comme libellé du bouton(qui n'est affiché que si localSubmit est à yes). Par défaut le label est "Valider".

3.9.11.2.2 Exemple

```
        <div>
            <h1>Paramétrage du mode d'affichage du zoo</h1>
            [ZONE FDL:EDITAPPLICATIONPARAMETER?
appId=ZOO&parameterId=ONEFAM_DISPLAYMODE&localSubmit=yes&submitOnChange=no]
        </div>
```

Code html généré :

```
<div>
     <h1>Paramètrage du mode d'affichage du zoo</h1>
     <div data-parameter="ONEFAM_DISPLAYMODE" class="editapplicationparameter">
            <div data-type="label">
            <label for="ONEFAM_DISPLAYMODE">Mode d'affichage du zoo</label>
            </div>
            <div data-type="field">
                   <form data-on-change="" action="" method="POST">
                          <input type="hidden" id="app_ONEFAM_DISPLAYMODE"
value="ZOO">
                          <input type="hidden" id="type_ONEFAM_DISPLAYMODE"
value="A">
                          <select id="ONEFAM_DISPLAYMODE"
name="_ONEFAM_DISPLAYMODE">
                                 <option value="html">html</option>
                                 <option selected="selected"
value="extjs">extjs</option>
                                 <option value="ng -beta-">ng -beta-</option>
                          </select>
                          <input type="submit" name="ONEFAM_DISPLAYMODE"
onclick="sendParameterApplicationData(this, true)" value="Valider">
                   </form>
            </div>
     </div>
</div>
```

3.9.11.3 La zone EDITSUBMIT

Cette zone est utilisée pour faire la sauvegarde de l'ensemble des Zones portant la classe "editfamilyparameter" ou "editapplicationparameter", n'ayant pas de bouton de validation, ni de validation lors de l'activation de l'événement onchange (submitOnChange=no et localSubmit=no)

Elle créera un bouton qui enverra les informations des formulaires au serveur.

Tout formulaire contenu dans une DIV portant la classe "editfamilyparameter" ou "editapplicationparameter", n'ayant pas de système de validation (bouton, input de type submit, envoi lors du onchange...) sera envoyé lors du clique sur ce bouton.

Comme les autres Zones de paramétrage, celle ci sera encapsulée dans un élément HTML de type DIV, portant la classe "editsubmit".

Cette élément ne contiendra qu'un input de type submit.

3.9.11.3.1 Description des paramètres

```
[ZONE FDL:EDITSUBMIT?label=Validation globale]
```

L'attribut label définit le label du bouton (par défaut "Valider").

3.9.11.3.2 Exemple

```
        <div>
            <h1>Réglage de paramètre d'une application et d'une famille.</h1>
            [ZONE FDL:EDITFAMILYPARAMETER?
famid=ZOO_ENTREE&attrid=ENT_PRIXENFANT&localSubmit=no&submitOnChange=no]
            [ZONE FDL:EDITAPPLICATIONPARAMETER?
appId=ZOO&parameterId=ONEFAM_DISPLAYMODE&localSubmit=no&submitOnChange=no]
            [ZONE FDL:EDITSUBMIT?label=Validation globale]
        </div>
```

Code html généré :

```
<div>
     <h1>Réglage de paramètre d'une application et d'une famille.</h1>
     <div data-parameter="ent_prixenfant" class="editfamilyparameter">
            <div data-type="label">
                   <label for="ent_prixenfant">Tarif des entrées au zoo pour les
enfants (en euro)</label>
            </div>
            <div data-type="field">
                   <form data-on-change="" action="" method="POST">
                          <input type="hidden" id="fam_ent_prixenfant"
value="ZOO_ENTREE">
                          <input type="text" id="ent_prixenfant" value="10"
name="_ent_prixenfant" class="fullresize" onchange="document.isChanged=true">
                   </form>
            </div>
     </div>
            <div data-parameter="ONEFAM_DISPLAYMODE"
class="editapplicationparameter">
            <div data-type="label">
            <label for="ONEFAM_DISPLAYMODE">Mode d'affichage du zoo</label>
            </div>
            <div data-type="field">
                   <form data-on-change="" action="" method="POST">
                          <input type="hidden" id="app_ONEFAM_DISPLAYMODE"
value="ZOO">
```

```
                        <input type="hidden" id="type_ONEFAM_DISPLAYMODE"
value="A">
                        <select id="ONEFAM_DISPLAYMODE"
name="_ONEFAM_DISPLAYMODE">
                              <option value="html">html</option>
                              <option selected="selected"
value="extjs">extjs</option>
                              <option value="ng -beta-">ng -beta-</option>
                        </select>
                  </form>
            </div>
     </div>
            <div class="editsubmit">
            <input type="submit" onclick="sendAllParameters()" value="Validation
globale">
     </div>
</div>
```

3.9.11.4 Événements liés

Suite à l'envoi du formulaire, un événement javascript sera émis afin que l'interface principale réagisse à l'application du changement. L'événement s'appelle "MODPARAMETER". Un objet est fourni en second paramètre sous forme clé/valeur, contenant :

- success : true ou false, booléen indiquant si l'appel a réussi ou échoué
- error : chaîne de caractères décrivant le type d'erreur, ou une chaîne vide s'il n'y a pas eu d'erreur
- data : En cas de succès, contient le nom du paramètre qui a été modifié et un booléen indiquant s'il a été modifié (sous forme {"parameterid" : ONEFAM_DISPLAYMODE, "modify" : true}). Lors d'une erreur elle peut contenir en plus une page html à afficher ({"parameterid" : ONEFAM_DISPLAYMODE, "modify" : false, "responseText" : <html>...</html>})

Envoi lors d'une erreur:

```
$("body").trigger("MODPARAMETER", {
                    "success":false,
                    "error":errorMsg,
                    "data":{
                        "parameterid":id,
                        "modify":false,
                        "responseText":data.responseText
                    }
                });
```

Envoi lors d'un succès :

```
 $("body").trigger("MODPARAMETER", {
                    "success":$status.attr("code") ? true : false,
                    "error":$status.attr("warning"),
                    "data":{
                        "parameterid":$data.attr("parameterid"),
                        "modify":$data.attr("modify") ? true : false
                    }
                });
```

Exemple de récupération :

```
$("body").bind("MODPARAMETER", function (event, rsp) {
    if (!rsp.success) {
        alert("An error has occured : "+rsp.error);
        if (rsp.data.responseText) {
            $("body").replaceWith(rsp.data.responseText);
        }
    }
});
```

3.9.12. Interface spécifique d'attribut HTMLText

(version 3.2 et +)

A partir de la version 3.2, il est possible de customiser le fonctionnement des attributs de type HTMLText. Vous pouvez spécifier le fonctionnement de l'éditeur de texte de deux manières :

- en spécifiant à l'aide de l'option jsonconf sont fonctionnement de manière précise (voir option jsonconf)
- et/ou en ajoutant un plugin à l'éditeur. L'ajout d'un plugin passe par la création de celui-ci en suivant la documentation de ckeditor (http://docs.cksource.com/, http://docs.cksource.com/CKEditor_3.x/Tutorials) et en déployant ensuite la structure du plugin dans le répertoire (<contexteDcP>/ckeditor/plugins/). Une fois le plugin créé vous devez l'ajouter au htmltext désiré en utilisant l'option de présentation jsonconf. Celle-ci possède un paramètre de configuration addPlugin qui permet d'ajouter le plugin à la toolbar en cours si celui-ci possède une celle commande et que cette commande a le même nom logique que le plugin, si ce n'est pas le cas vous devez l'ajouter en utilisant les options décrites dans les tutoriels de ckeditor. NB : Des exemples de plugins sont disponibles dans les sources de Dynacase dans le répertoire share : les sous répertoire ckeditor-*.

3.10 Cycle de vie

3.10.1. Développement

3.10.1.1 Définition d'un cycle de vie

Le cycle de vie dans dynacase permet à un document de changer d'état suivant un cycle défini. Pour changer d'état, le document doit franchir une transition. Cette transition s'accompagne d'une pré-action, d'une condition de transition et d'une post-action.

cycle de vie dynacase

Un document peut aller d'un état E1 à un état E2 seulement si une transition a été définie entre E1 et E2. Le passage de l'état E1 à l'état E2 est soumis à la condition C définie pour la transition. Cette condition est découpé en deux sous-conditions. La première est une condition générale sur les droits de passage de la transition. Ce droit est géré par les profils d'accès utilisateur. La deuxième est une condition spécifique à la transition. Cette dernière est généralement liée au contenu du document.

cycle de vie dynacase

Les rôles et activités décrivent qui fait quoi avant d'exécuter la transition. L'activité peut être une décision humaine permettant de choisir quelle transition adopter ou simplement une action humaine à exécuter (ex: appeler le directeur, envoyer par recommandé). La notion d'activité est décrite dans un attribut (activité) du cycle. Un attribut 'activité' est crée par état. La notion de rôle n'est pas intégré dans le processus informatique. Il est implicitement donné par l'utilisateur affecté à l'activité (qui doit modifier le document) et par le profilage du cycle : qui a le droit de modifier le document, qui a le droit de passer telle ou telle transition.

La pré-action A1 est exécutée systématiquement avant les conditions. Cette pré-action est généralement utilisée pour mettre à jour certains attributs utiles pour la transition et pour l'état suivant. La post-action A2 est exécutée si les conditions de transitions sont réunies. Lorsque A2 s'exécute le document est déjà dans l'état E2. Cette action est généralement utilisée pour créer ou mettre à jour des documents connexes au document courant.

Plusieurs transitions peuvent avoir l'état E1 en état courant ou avoir l'état E2 en état suivant. Le même état peut être à la fois courant et suivant. Par contre on ne peut définir qu'une seule transition de E1 à E2.

Les post-action, pré-action et condition spécifiques étant facultatives, le schéma du cycle ne représente que celles qui sont significatives. Les conditions sur les profils étant systématiques ne sont pas représentées. Les états initiaux, entourés de vert, n'ont pas d'état courant. Un seul état initial est possible. Les états finaux, entourés de rouge, n'ont pas d'état suivant. Lorsqu'un document est dans un état final, il ne peut plus changer d'état. Cela ne veut pas dire que le contenu du document n'est plus modifiable mais seulement que son état n'est plus modifiable.

3.10.1.2 Construction d'un cycle de vie

Le cycle de vie dans dynacase est défini dans une classe de document héritant de WDoc (workflow document). Cette classe décrit les états, les transition, les conditions de transition ainsi que les pré et post-action. Les rôles et les activités ne sont pas décrits pour l'instant.

La famille de cycle de vie est construite en créant une famille hérité de cycle de vie (WDOC) et à l'aide d'une classe documentaire héritée de WDoc.

	fromid	**title**	**id**	**classname**	**name**
BEGIN	WDOC	ma famille cycle		WDocTest	MONCYCLE
ICON	cycle.gif				
TYPE	C				
USEFOR	W				

END					

La propriété classname définie le fichier cycle (classe PHP) définissant les états et les transitions. Ici, il identifie le fichier /usr/share/what/FDL/Class.**WDocTest**.php

Tous les fichiers de classes PHP doivent être accessibles sous le répertoire `/usr/share/what/FDL`. Le nom du fichier doit correspondre au no mde la classe PHP.

Pour définir un cycle de vie, seuls les transitions sont obligatoires. Pour définir les transitions la classe WDoc se sert de la notion de *forme de transition*. Les formes de transitions sont nommées et recensées dans l'attribut tableau $transition de la classe. Une forme transition dans la classe WDoc contient la pré-action, la condition spécifique et la post-action. Elle ne contient pas les informations état courant et état initial. Ces informations sont décris dans l'attribut $cycle. Ceci permet de réutiliser une même forme de transition dans plusieurs transitions possibles.

Version > 2.11.12 . Les activités liées à un état sont définis dans le tableau $stateactivity. Il peuvent être défini dans la classe ou dans l'instance de cycle. S'ils sont définis dans la classe du cycle ils ne pourront être modifiés par l'interface.

Le nombre de type de transitions est actuellement limité à **20**. Cette limite est due au profil des transitions codé sur un entier de 32 bits (13 bits étant réservés pour les documents en général)

```
<?php
include_once("FDL/Class.WDoc.php");

Class WDocTest extends WDoc {
  public $attrPrefix="WDT";

  const initialised="initialised"; # N_("initialised")
  const transmited="transmited"; # N_("transmited")
  const accepted="accepted"; # N_("accepted")
  const refused="refused"; # N_("refused")
  const realised="realised"; # N_("realised")

  const Ttransmited="Ttransmited"; # N_("")
```

```
   const Taccepted="Taccepted"; # N_("Taccepted")
   const Trefused="Trefused"; # N_("Trefused")
   const Tretry="Tretry"; # N_("Tretry")
   const Trealised="Trealised"; # N_("Trealised")
   public $firstState=self::initialised;

   public $transitions=array(
       self::Ttransmited =>array(),
       self::Taccepted => array(),
       self::Trefused =>array(),
       self::Trealised=>array(),
       self::Tretry =>array();

   public $cycle=array(
      array("e1"=>self::initialised,
            "e2"=>self::transmited,
            "t"=>self::Ttransmited),
      array("e1"=>self::transmited,
            "e2"=>self::accepted,
            "t"=>self::Taccepted),
      array("e1"=>self::transmited,
            "e2"=>self::refused,
            "t"=>self::Trefused),
      array("e1"=>self::accepted,
            "e2"=>self::realised,
           "t"=>self::Trealised),
      array("e1"=>self::transmited,
            "e2"=>self::initialised,
            "t"=>self::Tretry) );

   public $stateactivity=array(self::initialised=>"adoption writting",
        self::transmited=>"adoption verification"); # _("adoption writting")
_("adoption verification")

}
```

Dans l'exemple ci-dessus, cinq formes de transitions sont utilisées. Elles sont équivalentes en terme de pré, post actions et de conditions spécifiques de transition. Le fait ici d'avoir trois formes de transitions implique que des droits différents de passage peuvent être appliqués.

Test	Voir	éditer	Supprimer	Vers transmis	Vers accepté	Vers refusé	Vers réalisé	Vers retour	Voir les droits	Modifier les droits
Utilisateurs	●			●	●		●	●		
Administrateurs		●				●			●	●

Valider | Réinitialisation | Seulement les cochés

Ainsi si la transition 'transmitted' vers 'initialised' est autorisée, elle le sera aussi pour 'accepted' vers 'initialised'.

Attention : Les droits sur les transitions sont calculés par rapport à l'ordre des transitions telles quelles ont été définies dans la variable "$transitions". Cela veut dire qu'on défini le droit pour le modèle de transition n°1, n°2 jusqu'à n°5. Le nom de la transition n'est pas pris en compte pour les droits, seul compte l'ordre. Cela veut dire que si vous supprimez la transition n°3 (Trefused), les droits de l'ancienne transition n°4 (Trealised) seront ceux de la transition supprimée (Trefused) qui est l'ancienne n°3. Il faut dans ce cas refaire la matrice de droit des transitions pour le cycle. Cela n'impacte pas les profils liés aux états du documents.

Les pré-actions, les conditions spécifiques et les post-actions sont des méthodes de la classe. Les pré-actions et les conditions spécifiques sont déclarées dans des méthodes spécifiques du cycle. Ces méthode doit retourner la chaîne vide en cas de succès. Si le condition spécifique n'est pas remplie, la méthode doit retourner le texte correspondant à l'erreur. Ces méthodes sont identifiées par les index m0 et m1 des formes de transitions. La méthode "m0" est la méthode de pré-condition qui vérifie si la transition peut être appliquée. Le message retourné par cette méthode est affiché dans les menu "changer d'état" disponible depuis les documents. La méthode "m1" est appelée si seulement "m0" est valide. Cette méthode peut réaliser des pré-traitements avant le changement d'état. Comme pour la méthode "m0" si elle retourne une message alors le passage de transition sera abandonné.

Les post-action sont les méthodes identifiées par les index m2 et m3. Si ces méthode retourne un texte, celui-ci sera affiché comme un message d'avertissement. La méthode m2 est appelée juste après le changement d'état. La méthode "m3" est appelée après l'envoie des courriels et l'attachement des minuteurs qui ont été paramétrés.

initialised
Activité pour initialised
Ttransmited
0
CA1
Ttransmited
transmited
Activité pour transmited
Taccepted
Trefused
Tretry
1
accepted
Taccepted
A3
2
refused
Trefused
A2
4
initialised
Activité pour accepted
Trealised
3
C0
Trealised
C1
Trealised
realised
Trealised
A2
Trealised
A3

```
<?php
include_once("FDL/Class.WDoc.php");

Class WDocTest extends WDoc {
  public $attrPrefix="WDT";
 const initialised="initialised"; # N_("initialised")
  const transmited="transmited"; # N_("transmited")
  const accepted="accepted"; # N_("accepted")
  const refused="refused"; # N_("refused")
  const realised="realised"; # N_("realised")

  const Ttransmited="Ttransmited"; # N_("")
  const Taccepted="Taccepted"; # N_("Taccepted")
  const Trefused="Trefused"; # N_("Trefused")
  const Tretry="Tretry"; # N_("Tretry")
  const Trealised="Trealised"; # N_("Trealised")
  public $firstState=self::initialised;

  public $transitions=array( self::Ttransmited =>array("m1"=>"CA1"),
      self::Taccepted => array("m3"=>"A3"),
      self::Trefused =>array("m2"=>"A2"),
      self::Trealised=>array("m0"=>"C0", "m1"=>"C1","m2"=>"A2","m3"=>"A3"),
      self::Tretry =>array();

  public $cycle=array(array("e1"=>self::initialised,
     "e2"=>self::transmited,
     "t"=>self::Ttransmited),
       array("e1"=>self::transmited,
     "e2"=>self::accepted,
     "t"=>self::Taccepted),
       array("e1"=>self::transmited,
     "e2"=>self::refused,
     "t"=>self::Trefused),
       array("e1"=>self::accepted,
     "e2"=>self::realised,
     "t"=>self::Trealised),
       array("e1"=>self::transmited,
     "e2"=>self::initialised,
     "t"=>self::Tretry) );

  public $stateactivity=array(self::initialised=>"adoption writting",
       self::transmited=>"adoption verification"); # _("adoption writting")
_("adoption verification")



  public function A1($newstate,$oldstate="",$comment="") {}
  public function C1($newstate) {}
  public function C2($newstate) {}
  public function CA1($newstate) {}
  public function A2($newstate) {}
  public function A3($newstate) {}
}
?>
```

Chacune des méthodes m1 ou m2 doit posséder un argument. Cet argument est le nom de l'état suivant. Pour les méthodes m1, l'état courant est donné par par l'attribut state du document courant ($this→doc).

```
// comment est le commentaire de transition. Il est vide si "nr=>true" (nr : no
raison) dans la description de la transition
// oldstate : état courant
// newstate : état qui va suivre
```

```
public function A1($newstate,$oldstate,$comment) {
  $cstate = $this->doc->state; // état courant == oldstate
}
```

Lorsque l'utilisateur change l'état du document (actions : changer d'état ou modifier), la nouvelle révision du document s'affiche à la place de l'ancienne. Dans la définition des méthodes de m1 ou m2, il est possible de rediriger l'URL d'affichage. Cette redirection se fait en modifiant les variables HTTP redirect_app et redirect_act.

```
public function A3($newstate) {
  SetHttpVar("redirect_app","FREEDOM"); //// application WHAT//
  SetHttpVar("redirect_act","HISTO&id=".$this->doc->id);//// action et argument
de l'application//
return "";
}
```

Dans cet exemple, lorsque le document passe de l'état E2 à E3 (forme de transition T3), la post-action A3 indique une redirection afin d'afficher l'historique du document à la place de la vue classique.

Le résultat transition peut être déroutée juste avant le passage de la transition. C'est la condition de transition qui indique la redirection. Si la méthode m1 retourne « → » cela indique que la transition ne sera pas effectué et que la redirection spécifiée par les variables HTTP sera appliquée. Pour effectuer cette transition il faudra forcer le passage (paramètre $force=true dans la méthode WDoc::changeState()). Cette redirection est généralement utilisée pour demander une confirmation pour la forme de transition.

Soit l'action confirmt qui demande la confirmation de la transition :

confirmt.php

```
<?php
include_once("FDL/Class.Doc.php");


function confirmt(Action &$action) {
  // Get All Parameters
  // document to change
  $docid = $action->getArgument("id",0);
  // new state
  $nextstate = $action->getArgument("state");

  $action->lay->Set("docid",$docid);
  $action->lay->Set("state",$nextstate);
  $action->lay->Set("tstate",_($nextstate));

}
?>
```

confirmt.xml

```
<form  id="fedit"
       class="fborder"
       name="modifydoc"
       method="POST" ENCTYPE="multipart/form-data"

       action="[CORE_STANDURL]&app=FREEDOM&action=MODSTATE">

<input type="hidden" name="fstate" value="yes">
<input type="hidden" name="id" value="[docid]">
<input type="hidden" name="comment" value="[TEXT:confirmed]">
<input type="hidden" name="newstate" value="[state]">
```

```
 <input type="submit"
        value="[TEXT:Confirm change the document state to] [tstate]" >
 <input type="button" value="[TEXT:cancel]"

onclick="location.href='[CORE_STANDURL]&app=FDL&action=FDL_CARD&id=[docid]'">
</form>
```

Cette action a comme argument l'identificateur de document et l'état suivant du document. Lorsque l'utilisateur appuis sur le bouton de confirmation, une nouvelle demande de changement d'état a lieu mais cette fois ci avec l'attribut force (appel de l'action MODSTATE avec l'attribut fstate=yes).

Nous reprenons, l'exemple précédent en modifiant les méthodes A1 et C1. Ceci permet de déclencher la confirmation sur la forme T1 de transition.

```
public function A1($newstate) {
  SetHttpVar("redirect_app","TEST");
  SetHttpVar("redirect_act","CONFIRMT&state=$newstate&id=".$this->doc->id);
}
public function C1($newstate) {
  return "->";
}
```

Le changement d'état forcé n'ignore pas la condition C1. Cette condition est recalculée et si le retour est « → » le changement est opéré au contraire du mode normal où une redirection est effectuée. Dans le mode forcé, la redirection des méthodes m1 est ignorée par contre, si une redirection est mise en place dans m2, elle sera prise en compte.

3.10.1.3 Paramètres de transition

Lors de la définition des types de transitions il est possible d'appliquer des paramètres. Ces paramètres seront présentés lors d'un changement d'état et seront accessibles depuis les pré et post-actions. Ces paramètres doivent être définis dans la famille du cycle de vie sous forme d'attribut paramètre.

//	fromid	title	special id	classname	name							
BEGIN	WDOC	cycle action de prospection		WProActio	WPROACT							
ICON	cycle.gif											
TYPE	C											
USEFOR	W											
//	idattr	idframe	label	T	A	type	ord	vis	need	link	phpfile	phpfunc
PARAM	WPROA_FR_TRANS		Paramètres de transition	N	N	frame	10	W				
PARAM	WPROA_DATEREPOI	WPROA_FR_TRANS	date de report	N	N	date	20	W	Y			
PARAM	WPROA_IDPERS	WPROA_FR_TRANS	id responsable	N	N	docid	30	H				
PARAM	WPROA_PERS	WPROA_FR_TRANS	responsable	N	N	text	40	W	Y	%S%app=I	prospect.ph	[list]lprospectsociety(1 WPROA_PERS
END												

Pour indiquer quels attributs doivent être demandé pour les transition, il faut ajouter l'élément “ask” dans le tableau des transitions comme le montre l'exemple suivant. Cet élément est composé d'un tableau des identificateurs des paramètres. Soit le fichier **Class.WProAction.php** (défini par colonne *classname*):

Le résultat de ce cycle donne l'affichage suivant lors d'un changement d'état.

```
<?php
include_once("FDL/Class.WDoc.php");

Class WProAction extends WDoc {
  const todo="todo"; # N_("todo")
  const done="done"; # N_("done")
  const redo="redo"; # N_("redo")
  const Tredo="Tredo"; # N_("Tredo")
  const Tdone="Tdone"; # N_("Tdone")
  public $attrPrefix="WPROA"; // prefix attribute

  public $transitions = array(
       self::Tdone =>array("m1"=>"",
```

```
"ask"=>array("WPROA_DATEREPORT",
     "WPROA_IDPERS","WPROA_PERS"),
"m2"=>""),
    self::Tredo =>array("m1"=>"askDate",
"ask"=>array("WPROA_DATEREPORT"),
"m2"=>""));
```

```
  public $cycle = array(
    array("e1"=>self::todo,
          "e2"=>self::done,
          "t"=>self::Tdone),
    array("e1"=>self::todo,
          "e2"=>self::redo,
          "t"=>self::Tredo) ,
    array("e1"=>self::redo,
          "e2"=>self::redo,
          "t"=>self::Tredo)
    );

  public $firstState=self::todo;
}
public function askDate($newstate) {
  $err="";
  $mydate=$this->getValue("WPROA_DATEREPORT");
  // traitement de la date ...
  return "$err";
}
?>
```

Il est possible d'indiquer les paramètres obligatoires si l'information est indispensable. La récupération des valeurs ce fait par la méthode “getValue”()”” du document cycle de vie. Si l'information des paramètres n'est pas mémorisée dans les méthodes de pré ou post-actions, elles seront perdues.

Depuis la version 2.13.12, il est possible de pré-renseigner les valeurs des paramètres de transitions. Si vous renseignez la valeur du paramètre d'un cycle, cette valeur sera automatiquement proposée lors de la demande de changement d'état. La valeur de ce paramètre peut faire référence à une méthode du cycle (pas du document instancié). Cela permet de proposer des valeur différente suivant le contexte d'exécution.

BEGIN	WDOC	Cycle de compte rendu		WMinuteMee	WMINUTEMEETING				
ICON	cycle.gif								
USEFOR	W								
TYPE	C								
//	idattr	idframe	label	T	A	type	ord	vis	needed
PARAM	WMM_FR_PARAM		Paramètre			frame	10	W	
PARAM	WMM_DATE	WMM_FR_PARAM	Une date	N	N	date	20	W	
PARAM	WMM_AUSER	WMM_FR_PARAM	Quelqu'un	N	N	docid("IUSER")	25	W	Y
PARAM	WMM_AVALUE	WMM_FR_PARAM	Le lieu	N	N	text	30	W	
DEFAULT	WMM_AVALUE	::getInstanceValue('SERT_PLACE)							
DEFAULT	WMM_AUSER	::getInstanceValue('SERT_CONTROLLER)							
DEFAULT	WMM_DATE	::getDate(7)							
END									

Dans l'exemple ci-dessus, le paramètre 'WMM_DATE' sera proposée avec la date de la semaine prochaine (`::getDate(7)`), `WMM_AVALUE` aura la valeur du contrôleur du document auquel est attaché le cycle et `WMM_AVALUE` aura la valeur du lieu du document courant.

3.10.1.4 Options de transitions

3.10.1.4.1 Transition principale

Dans la définition du cycle, un chemin principal allant de l'état initial vers l'état final peut se dégager. Lors d'un changement d'état il est possible de proposer un état suivant déterminé pour indiquer le chemin « normal ». Par défaut, l'interface de changement propose « inchangé » (pas de transition). Si un état suivant est indiqué dans la spécification du cycle, l'interface proposera cet état en premier (inchangé sera toujours disponible). Ce chemin principal est indiqué dans l'attribut autonext de la classe cycle de vie comme le montre l'exemple suivant.

```
<?php
include_once("FDL/Class.WDoc.php");
```

```
Class WTest extends WDoc {
  public $attrPrefix="TWF"; // prefix attribute
  const E1="E1";
  const E2="E2";
  const E3="E2";
  const E4="E2";
  const T1="T1";
  const T2="T2";
  const T3="T2";
  const T4="T2";
  public $firstState=self::E1;
  public $transitions = array(self::T1 =>array(),
       self::T2 =>array(),
       self::T3 =>array(),
       self::T4 =>array());

  public $autonext=array(self::E1=>self::E2, // on propose E2 si l'état est E1
 self::E2=>self::E4); // on propose E4 si l'état est E2

  public $cycle = array(
 array("e1"=>self::E1,
       "e2"=>self::E2,
       "t"=>self::T2),
 array("e1"=>self::E1,
       "e2"=>self::E3,
       "t"=>self::T3),
 array("e1"=>self::E2,
       "e2"=>self::E3,
       "t"=>self::T3),
 array("e1"=>self::E2,
       "e2"=>self::E4,
       "t"=>self::T4),
 array("e1"=>self::E3,
       "e2"=>self::E4,
       "t"=>self::T4));
}
?>
```

3.10.1.4.2 Transition forcée

Afin de simplifier et de forcer le passage d'une transition, il est possible d'indiquer que la sauvegarde sans changement d'état n'est pas possible. Cela implique que l'interface de changement d'état n'indique plus « inchangé » pour le nouvel état.

Cette restriction est indiquée dans l'attribut de type tableau **nosave**. Ce tableau contient tous les états à partir desquels il est obligatoire de changer d'état afin de modifier un document.

Reprenant l'exemple précédent, la ligne suivante est rajoutée pour un indiquant que l'état E2 ne peut être sauvegardé

sans changement d'état.

```
public $nosave=array("E2");
```

On remarque dans la capture d'écran que « inchangé » n'est pas proposé. L'utilisateur est alors forcé de choisir entre E3 et E4

3.10.1.4.3 Transition sans confirmation

Un passage de transition demande toujours la raison du changement d'état. Ceci a pour conséquence un affichage d'une fenêtre de dialogue afin de demander cette raison. Si votre transition n'implique pas de création de document et n'est pas critique, vous pouvez dire que la raison n'est pas obligatoire, et donc que la fenêtre de dialogue ne sera plus présentée, sauf en cas de demande de paramètre de transition (ask). Pour activer l'option sans confirmation il faut indiquer que l'index “nr” (*no reason*) est vrai.

```
public $transitions = array(
        self::Tdone =>array("m1"=>"",
        "nr"=>true, // sans confirmation
        "m2"=>""),
        self::Tredo =>array("m1"=>"askDate",
        "ask"=>array("WPROA_DATEREPORT"),
        "m2"=>""));
```

3.10.1.4.4 Interface boutons

Lorsque les possibilités de transitions sont réduites, l'interface de changement d'état peut être changé pour utiliser des boutons au lieu d'une liste déroulante. Dans ce cadre, il est nécessaire de traduire les états (E1, E2, etc.) et aussi les actions liées à ces états (ToE1, ToE2, etc.). Les actions liées aux états sont nommées avec le préfixe « To » suivi du nom de l'état.

La première capture d'écran reprend les conditions du paragraphe §8.4.1 et la seconde représente la même fonctionnalité que celle montrée au paragraphe précédent, c'est à dire le fait de ne pas pouvoir sauver sans changer d'état.

```
// pour l'i18n
# N_("E1") N_("E2") N_("E3") N_("E4")
# N_("ToE1") N_("ToE2") N_("ToE3") N_("ToE4")

public $viewlist="button";
```

Pour indiquer ce mode de présentation, il suffit d'affecter l'attribut $viewlist à « button » dans la classe du cycle. Par défaut, cet attribut vaut « list ». Il est nécessaire d'indiquer qu'il faut aussi traduire les actions (N_(« **To***xxx* »).

3.10.2. Paramétrage

3.10.2.1 Créer un Cycle de vie (Workflow)

3.10.2.1.1 But de ce document

Le but de ce document est de décrire les étapes pour mettre en place un Workflow.

Dans dynacase un Workflow est appelé “Cycle de vie”.

3.10.2.1.2 Principe général

Pour mettre en place un cycle de vie, il faut créer ou modifier plusieurs choses :

- Un fichier PHP qui définit le fonctionnement du cycle de vie (États, transitions,..)
- Une famille de cycle de vie utilisant le fichier PHP précédent
- Un document cycle de vie basé sur la famille précédente. Il est possible de créer plusieurs cycles de vie basés sur la même famille de cycle de vie.
- Une famille de documents sur laquelle le cycle de vie sera appliqué. Il est possible d'affecter un cycle de vie à plusieurs familles de documents.
- Un profil général affecté au cycle de vie ou un profil particulier pour chaque étape du cycle de vie. Cela permet de gérer qui a le droit de faire quoi sur chaque étape du cycle de vie.
- Il est possible également de créer un masque particulier pour chaque étape du cycle de vie pour afficher ou masquer certains champs en fonction de l'état d'avancement du document dans le cycle.
- Les modèles de courriel à envoyer.
- Les déclencheurs pour ajouter des alarmes suivant des critères de temps.

3.10.2.1.3 Exemple de cycle de vie

Dans ce document, nous allons décrire toutes les étapes nécessaires à la mise en place du cycle de vie ci-dessous :

Dans cet exemple, un document peut avoir 4 états :

- Brouillon
- Validé,
- Archivé
- Poubelle

Les transitions pour passer d'un état à un autre sont nommées :

- Validation : Pour passer de l'état « Brouillon » à l'état « Validé »
- Correction : Pour passer de l'état « Validé » à l'état « Brouillon »
- Archivage : Pour passer de l'état « Validé » à l'état « Archivé »
- Destruction : Pour passer de l'état « Brouillon » à « Poubelle » ou de l'état « Validé » à l'état « Poubelle »

Pour décrire ces états et ces transitions, il faut créer un fichier PHP comme indiqué au chapitre suivant.

3.10.2.1.4 Création du fichier PHP de définition du cycle de vie

Le fichier PHP définissant le cycle de vie, sera placé dans le dossier « **/usr/share/what/FDL** » Son nom sera sous la forme : « Class.**WDocTestCycle1**.php » correspond au nom de la classe de la famille de cycle de vie que nous allons créer au chapitre suivant. Voici le contenu de ce fichier qui permet de définir notre cycle de vie :

```
<?php
include_once("FDL/Class.WDoc.php");

Class WDocTest extends WDoc {

var $attrPrefix="WDT";
var $firstState="wdt_Draft";

var $transitions = array("wdt_Approvement"=>array(),
 "wdt_Correction"=>array(),
 "wdt_Chechin"=>array(),
 "wdt_Destroy"=>array());

 var $cycle = array(array("e1"=>"wdt_Draft",
   "e2"=>"wdt_Approved",
   "t"=>"wdt_Approvement"),
     array("e1"=>"wdt_Approved",
   "e2"=>"wdt_Check",
   "t"=>"wdt_chechin"),
     array("e1"=>"wdt_Approved",
   "e2"=>"wdt_Draft",
   "t"=>"wdt_Correction"),
     array("e1"=>"wdt_Draft",
   "e2"=>"wdt_Obsolete",
   "t"=>"wdt_Destroy"),
     array("e1"=>"wdt_Approved",
   "e2"=>"wdt_Obsolete",
   "t"=>"wdt_Destroy"));
}
?>
```

Commentaires sur les variables :

- $firstState : État initial. Si cette variable n'est pas initialisée, le document sera dans un état indéterminé à la création ce qui en général, n'est pas souhaitable.
- $cycle : Cette variable permet de définir le nom des états et des transitions. Elle permet également de définir quelle transition utilisée pour passer d'un état à un autre
- $transition : Cette variable détermine le nom des transitions

Attention :

- Le nom de la « class » doit correspondre au nom du fichier et au nom indiqué dans les attributs de la famille (attention également au respect des minuscules/majuscules).
- Il est possible de modifier ce fichier à tout moment[33]. Les modifications sont immédiatement prises en compte dans tous les documents utilisant cette famille de cycle de vie.
- Il ne faut pas utiliser de caractères spéciaux (accents) et d'espaces dans le nom des transitions et des étapes (Pour avoir des accents dans les documents finaux, il faut passer par gettext pour traduire les différents termes)

3.10.2.1.5 Création de la famille « Cycle de vie » dans OOo

Il n'est pas possible de créer une famille cycle de vie dans dynacase. Il faut donc créer cette famille dans OOo et l'importer dans dynacase. Vous trouverez dans FIXME internalmedia: freedom_2.14:admin:family:cycle_famille_testcycle1.ods la définition de cette famille :

//	héritage	titre	id	class	name
BEGIN	WDOC	Test Cycle		WDocTest	TestCycle
ICON	cycle.gif				
TYPE	C				
USEFOR	W				
END					

Dans ce paramétrage, la seule chose vraiment importante est le nom de la classe et donc du fichier PHP à utiliser. Dans notre cas, c'est « **WDocTest** »

Remarques :

- Il faut obligatoirement créer le fichier PHP de définition du cycle avant d'importer la famille sinon l'importation échouera.
- Une fois cette famille de cycle de vie importée, elle apparaît dans la liste des familles mais pas dans la liste des cycles de vie.

3.10.2.1.6 Création d'un document « Cycle de vie »

Une fois la famille importée, il faut créer un document basé sur cette famille. Ce document apparaîtra dans la liste des cycles de vie disponible. C'est sur ce document qu'il sera possible d'affecter un profil et donc des droits différents en fonction des besoins.

Voici comment créer ce document :

- Sélectionnez la famille de cycle de vie “Test” importée précédemment.
- Faire un clic-droit sur cette famille et sélectionnez “Créer cycle de test”.
- Donnez un titre à votre document cycle de vie.

Une fois ce document créé, il est possible de voir graphiquement le cycle de vie en cliquant sur le 'Voir le graphe' :

33) *à éviter sur un site en production*

Remarque :

- Il est possible de modifier le fichier PHP « Class.WDoc**Test**.php » et de voir instantanément le changement au niveau du graphe dans ce document cycle de vie et dans tous les documents créés à partir de la famille cycle de vie importée précédemment.
- Il est possible de supprimer, modifier et ajouter des états et des transitions

3.10.2.1.7 Affectation d'un cycle de vie à une famille de documents

Le cycle de vie créé doit être associé à une famille de document.

Pour affecter un cycle de vie à une famille, il faut :

27. Sélectionner la famille de documents de votre choix
28. Sélectionner « Choisir un cycle »
29. Sélectionner le cycle de vie créé précédemment

Une fois le cycle choisi, les document de cette famille sont lié à ce nouveau cycle. Si des documents de cette famille étaient déjà liés à un autre cycle, ceux-ci ne sont impactés par ce changement.

Remarque : Il est possible d'utiliser un même cycle de vie pour plusieurs familles de documents.

3.10.2.1.8 Initialisation du document cycle de vie

L'initialisation du cycle de vie permet de réinitialiser les attributs du cycle. Ceci est nécessaire lors de la première édition ou lors du modification du fichier de définition du cycle (classe documentaire). ensuite en éditant ce dernier de sélectionner :

- un profil en fonction de l'état
- une couleur en fonction de l'état
- un masque en fonction de l'état

Pour initialiser un cycle de vie, il faut :

- Sélectionner le document cycle de vie créé précédemment
- Cliquer sur “Initialisation”

Une fois le cycle de vie initialisé, des attributs correspondants aux états sont ajoutés à la famille du cycle de vie.

Pour voir ces attributs ajoutés, il faut :

- Faire un clic droit sur la famille de cycle de vie importée
- Sélectionner “Éditer les attributs”

ATTENTION : Il est conseillé d'initialiser le cycle de vie une fois les états et transitions figés car ensuite, il n'est plus possible de supprimer les attributs ajoutés à la famille du cycle de vie.

3.10.2.1.9 Affectation d'un profil dédié au document cycle de vie

L'affectation d'un profil au document cycle de vie permet d'affecter des droits différents pour chaque étape de ce cycle. Pour cela, il faut :

- Sélectionner le document de cycle de vie dans le dossier de recherche « les cycles »
- Sur le document du cycle de vie, sélectionner « Sécurité/Changer de profil » et sélectionner « Contrôle dédié »
- Sur le document du cycle de vie et sélectionner « Sécurité / Accessibilité »
- Dans le tableau qui apparaît, il faut cliquer sur les petits rectangles gris à droite des groupes pour faire apparaître des cases à cocher

* Il faut sélectionner les cases à cocher en fonction de vos besoins et valider

Pour plus d'information sur la gestion des profils, il faut suivre la documentation correspondante.

Pour plus d'explications sur la modification des droits du profil, il faut suivre la documentation sur les profils

Remarques :

- Par défaut, un simple utilisateur n'a pas accès aux transitions du cycle de vie. Il faut donc obligatoirement modifier le profil pour autoriser les personnes à modifier les étapes du cycle de vie.
- L'administrateur de dynacase a accès à toutes les étapes d'un cycle de vie et à tout moment et quelque soit les transitions autorisées.
- Il n'est pas possible d'affecter un profil à un document cycle de vie. Il faut obligatoirement créer un « Contrôle dédié ». Cependant, il est possible d'affecter un profil différent pour chaque étape du cycle de vie une fois le cycle initialisé.
- Il est possible de créer un cycle de vie générique avec de nombreuses étapes et transitions. A partir de là, il est possible de créer plusieurs documents de cycles de vie avec chacun un profil différent.

3.10.2.1.10 Test du cycle de vie

Avant de tester le cycle de vie, il faut affecter des droits différents à au moins deux utilisateurs au niveau du profil et voir comment évolue le document en fonction des états. En fonction de l'état du document et de l'utilisateur, les boutons permettant de passer d'un état à un autre sont modifiés.

Pour tester le cycle de vie il faut simplement créer un nouveau document basé sur la famille utilisant ce cycle de vie.

Remarque : L'administrateur de dynacase a toujours accès à tous les boutons de changement d'état quelque soit le paramétrage du cycle de vie.

3.10.2.1.11 Affectation d'un profil différent pour chaque étape du cycle de vie

Pour gérer encore plus finement les droits, il est possible d'affecter un profil différent sur chaque étape du cycle de vie.

Pour cela, il faut créer un profil différent pour chaque étape du cycle de vie. Pour créer un profil, il faut suivre cette documentation.

Ensuite, il faut modifier le document cycle de vie et affecter les profils aux différents états

Remarque : Cela est possible seulement si le cycle de vie à été initialisé comme indiqué dans un chapitre précédent.

3.10.2.1.12 Ajouter un repère de couleur à chaque étape du cycle de vie

Pour identifier plus facilement dans quel état est le document, il est possible d'affecter des couleurs différentes pour chaque état du cycle de vie. Pour cela, il faut :

- Modifier le document cycle de vie
- Indiquer pour chaque état du cycle de vie, la couleur souhaitée

Une fois les couleurs définies, voici le résultat obtenu dans les différents écrans :

- Changement d'état d'un document sous forme de liste déroulante

- Liste des documents dans le module “Une Famille”

3.10.2.1.13 Ajouter un masque différent à chaque étape du cycle de vie

En fonction de l'état du document il est possible d'afficher ou masquer certains champs en appliquant un masque différent à chaque étape.

Pour créer un masque, il faut :

- Depuis le module “Gestion documentaire”, sélectionner le menu “Création / Document système”.
- Dans la liste “hérite de”, sélectionner “masque de saisie.
- Dans la liste “pour” sélectionner une famille.
- Indiquer le nom du masque de saisie
- Pour chaque attribut, il est possible de modifier sa visibilité et son obligation

Pour affecter un masque sur une étape du cycle de vie, il faut :

- Créer un masque différent pour chaque étape comme expliqué ci-dessous.
- Modifier le document cycle de vie.
- Indiquer pour chaque état du cycle de vie, le masque souhaité.

Remarque : Pour retrouver facilement la liste des masques de saisie, il est possible de sauvegarder une recherche simple basée sur la famille “masque de saisie”

3.10.2.1.14 Ajouter un contrôle de vue par état

Si le masque ne suffit pas pour renseigner différentes parties du document, il est possible d'associer un contrôle de vue par état. Le contrôle de vue permettra d'avoir des masques différents par états. Le profilage de chaque contrôle de vue permettra d'appliquer différent masques suivant l'utilisateur.

3.10.2.1.15 Définir les activités par états

Version 2.11.12

Les états du cycle définissent l'état du document à un moment donné. Il sont définis par des adjectifs comme 'validé',

'archivé', 'contrôlé'. Pour chacun de ces états, une activité peut être définie afin d'indiquer ce que doit faire celui qui a en charge du document. Cette activité peut être par exemple 'rédaction','vérification','à confirmer'. Les activités peuvent être notées dans les attributs 'activité' du cycle de vie. Ils peuvent ainsi être différent par instance de cycle. Ils peuvent aussi être définis dans la classe du cycle, dans ce cas les attributs seront non modifiable par l'interface. Dans l'exemple ci-dessous, pour l'état 'validé' nous avons défini l'activité 'en cours d'archivage'.

3.10.2.1.16 Affecter un utilisateur

Lorsqu'on a défini une activité, on peut aussi définir qui doit faire l'activité. Bien qu'une activité puisse être réalisée à plusieurs, un seul responsable peut être désigné. Un document ne peut être affecté qu'à un seul utilisateur à la fois.

Paramètres pour l'état Archivé

Profil Archivé :
Masque Archivé :
Couleur Archivé : #8CBFD9
Contrôle de vue Archivé :
Modèle de courriel Archivé :
Minuteur Archivé :
Activité Archivé :

Type d'affectation Archivé	Utilisateur affecté Archivé	auto-verrouillage Archivé	courriel automatique Archivé
Relation document	bt_idchecker (document:	☐	☐

On peut récupérer l'acteur à partir d'attribut relation (type 'docid') référençant un utilisateur. Cet attribut peut être récupéré à partir du document (attribut ou paramètre de famille) ou à partir du cycle (attribut ou paramètre de famille du cycle). Dans l'exemple ci-dessus, le documentaliste sera affecté comme responsable lorsque le document passera dans l'état archivé. Si vous cochez “auto-verrouillage” le document sera verrouillé pour l'acteur. Si vous cochez “courriel automatique”, un courriel d'affectation sera envoyé à l'acteur au moment du changement d'état.

Une fois l'utilisateur affecté, il peut ré-affecter le document s'il a le droit d'éditer le document. Dans ce cas le menu “ré-affecter” apparaîtra sur le document. L'affectation est une des propriétés du document. On retrouvera la personne responsable lorsqu'on consulte les propriétés du document. Pour connaître la liste des documents qui vous sont affectés on peut utiliser le service portail “mes documents affectés” depuis l'application portail. Ce service utilise la recherche 'MY_AFFECT_DOCS' qui peut être utilisés aussi dans la gestion documentaire et comme flux RSS.

recherche détaillée

MES DOCUMENTS AFFECTÉS

| Éditer | Supprimer | Historique | Ajouter un post-it | Ouvrir | Autres

Basique

Utilisable comme flux RSS : oui
Flux RSS système : oui
À utiliser dans les menus : non

Critère

Révision : courante

Conditions

satisfait toutes les conditions

- affecté égal ::getMyAttribute(us_whatid)

3.10.2.1.17 Pour aller plus loin

Pour aller plus loin dans l'utilisation des cycles de vie, il faut suivre le chapitre correspondant dans le manuel de programmation.

3.10.2.2 Créer un cycle de vie localisé

3.10.2.2.1 But de ce document

Dans la définition des étapes et des transitons d'un cycle de vie, nous avons vu qu'il n'était pas possible de mettre des espaces et des caractères accentués dans les noms.

En effet, ces noms sont utilisés comme attributs de la famille cycle de vie. Ils correspondent également aux noms des champs de la table dans la base de données.

Le but de ce document est donc d'expliquer comment utiliser la localisation pour traduire ces attributs en textes plus complet dans la langue de votre choix (en français ou en anglais)

Voici l'exemple de cycle que nous allons réaliser dans ce document :

3.10.2.2.2 Pré-requis

Avoir déjà mis en place un cycle de vie simple comme indiqué dans cette documentation :

- Cycle de vie

3.10.2.2.3 Exemple de définition de cycle de vie utilisé pour ce docuement

Pour ce document, nous allons mettre en place un cycle de vie très simple avec seulement deux étapes et deux transitions.

Attention : Les lignes en commentaire avec les _() et N_() sont importantes car ce sont elles qui nous permettrons de localiser les textes dans les différentes langues (français et anglais)

```
<?php
include_once("FDL/Class.WDoc.php");

define ("i18n","i18n");

Class WDocTest extends WDoc {
  public $attrPrefix="WDT";
  public $firstState="wdt_Draft";

  public $transitions = array("wdt_Approvement"=>array(),# _("wdt_Approvement")
_("wdt_Correction") _("wdt_Chechin")
       "wdt_Correction"=>array(),
       "wdt_Chechin"=>array(),
       "wdt_Destroy"=>array());

  public $cycle = array(array("e1"=>"wdt_Draft", # _("wdt_Destroy")
_("wdt_Draft") _("wdt_Approved") _("wdt_Check") _("wdt_Obsolete")
       "e2"=>"wdt_Approved",
       "t"=>"wdt_Approvement"),
 array("e1"=>"wdt_Approved",
```

```
        "e2"=>"wdt_Check",
        "t"=>"wdt_Chechin"),
 array("e1"=>"wdt_Approved",
        "e2"=>"wdt_Draft",
        "t"=>"wdt_Correction"),
 array("e1"=>"wdt_Draft",
        "e2"=>"wdt_Obsolete",
        "t"=>"wdt_Destroy"),
 array("e1"=>"wdt_Approved",
        "e2"=>"wdt_Obsolete",
        "t"=>"wdt_Destroy"));
  function postConstructor() {
    $this->stateactivity=array("wdt_Draft"=>N_("writting"),
         "wdt_Approved"=>N_("check in"));
  }
}
?>
```

3.10.2.2.4 Traduction des constantes dans différentes langues

La première étape consiste à récupérer le contenu des constantes du fichier .php précédent pour générer un nouveau fichier .po qui nous permettra de renseigner les traductions :

```
# xgettext --keyword='N_' --language=c --keyword='_'  -j  -o test_fr.po
Class.WDocTest.php
```

Ensuite, il faut éditer ce fichier texte « .po », indiquer le bon encodage sur la ligne « Content-Type » et ajouter les traductions. Exemple :

```
msgid ""
msgstr ""
"Project-Id-Version: TEST\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-01-02 23:42+0100\n"
"PO-Revision-Date: 2009-01-02 23:56+0100\n"
"Last-Translator: me\n"
"Language-Team: french <LL2@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=iso-8859-1\n"
"Content-Transfer-Encoding: 8bit\n"

#: Class.WDocTest.php:10
msgid "wdt_Approvement"
msgstr "Approbation"

#: Class.WDocTest.php:10
msgid "wdt_Correction"
msgstr "Correction"

#: Class.WDocTest.php:10
msgid "wdt_Chechin"
msgstr "Archivage"

#: Class.WDocTest.php:15
msgid "wdt_Destroy"
msgstr "Destruction"

#: Class.WDocTest.php:15
msgid "wdt_Draft"
msgstr "Initialisé"
```

```
#: Class.WDocTest.php:15
msgid "wdt_Approved"
msgstr "Validé"

#: Class.WDocTest.php:15
msgid "wdt_Check"
msgstr "Archivé"

#: Class.WDocTest.php:15
msgid "wdt_Obsolete"
msgstr "Obsolète"

#: Class.WDocTest.php:31
msgid "writting"
msgstr "En rédaction"

#: Class.WDocTest.php:32
msgid "check in"
msgstr "En cours d'archivage"
```

Remarques : * Pour mettre en place une traduction en anglais, il faut créer un deuxième fichier .po et indiquer les traductions en anglais.

- Pour éditer ce fichier, il est possible d'utiliser des interfaces graphiques spécialisées comme gtranslator (gnome), kbabel (kde) ou poedit (gtk)

3.10.2.2.5 Mise en place de la traduction

Il faut commencer par convertir le fichier texte « .po » en fichier binaire « .mo » :

```
# msgfmt test_fr.po -o /usr/share/what/locale/fr/LC_MESSAGES/test.mo
```

Remarque : La commande « msgunfmt » permet de faire l'inverse :

```
# msgunfmt fichier.mo -o fichier.po
```

Remarque : Pour mettre en place une traduction en anglais, il faut utiliser le dossier “locale/en/LC_MESSAGES/”

Assembler les différents fichiers « .mo » dans « what.mo » utilisé par dynacase :

```
# /usr/share/what/whattext
```

Recharger Apache :

```
# /etc/init.d/apache2 reload
```

Remarque : Le cycle de vie doit apparaître traduit immédiatement.

Pour avoir le graphe en anglais, on procède de la même façon en copiant le fichier de traduction dans le répertoire “locale/en/LC_MESSAGES/”.

3.10.2.2.6 Paramétrer dynacase pour avoir une interface en anglais

Pour avoir l'interface de dynacase en anglais pour tout les utilisateurs, il faut modifier ce paramètre :

- Administration / Gestion des applications / paramètres applicatif / Noyau / langue

Pour avoir l'interface de dynacase en anglais pour un utilisateur particulier, il faut modifier ce paramètre :

- Administration / Gestion des applications / mes paramètres / Noyau / langue

Remarques :

- Après avoir modifié la langue, il est nécessaire de se déconnecter et de se reconnecter à dynacase.

Version > 2.11.12

3.10.2.3 Modèle de courriel pour les cycles

Il est possible de paramétrer le cycle par l'interface afin d'envoyer des courriels lors d'un changement d'état. Les courriels peuvent être définis pour un état ou pour une transition. Si on associe un ou plusieurs courriels à un état, ils

seront envoyés à chaque fois qu'un document aura ce nouvel état (quel que soit la transition). Les courriels liés aux transitions seront envoyés lors du passage de la transition.

Soit un document basique associé au cycle de test.

Voici le document de test en édition. Il comprend un rédacteur et un valideur.

Le but est d'envoyer un courriel pour avertir le documentaliste d'archiver le document lorsque celui-ci est validé (passage à l'état *validé*).

Pour cela on édite le cycle de test et clique sur le '+' de l'attribut 'Modèle de courriel Validé'. Cela affiche le document suivant :

éditer Demande d'archivage - Mozilla Firefox

http://localhost/freedom/?sole=Y&&app=GENERIC&action=GENERIC_EDIT&rzone=&id=24701

modèle de mail

DEMANDE D'ARCHIVAGE

Sauver | Annuler

Entête

Titre : Demande d'archivage

Famille : Test de base

Famille cycle : cycle de test

Emetteur

type	De
Attribut relation	bt_idredac (rédacteur)

Destinataires

-	type	destinataire
À	Attribut relation	bt_idchecker (documentaliste)
Cc	Attribut relation	bt_idvalid (valideur)
Cc	Paramètre cycle	wdt_mailchief (Courriel du chef)

Sujet : Demande d'archivage [TITLE]

Contenu

Avec lien : ☑

Corps :

Format Normal Size B I U ABC Source

De [V_BT_IDREDAC],

Bonjour veuillez archiver le document suivant [V_TITLE].

[WCOMMENT]

Il a été validé par [V_BT_IDVALID].

Cordialement [V_BT_IDREDAC]

Attachements

Attachement
bt_fileref (Référence)

Terminé

Le modèle de courriel permet de définir l'émetteur, les destinataires et le corps du message.

3.10.2.3.1 Définir l'émetteur

L'émetteur doit faire référence à une adresse email valide. Si l'émetteur reste vide, c'est l'adresse de l'utilisateur qui fait la transition qui sera utilisée.

3.10.2.3.2 Types d'attribut possible pour les destinataires et l'émetteur

Plusieurs possibilités de choix de l'émetteur/des destinataires sont possibles:

- adresse fixe : choisir dans la carnet d'adresse une personne. C'est l'adresse mail de cette personne qui sera utilisée. (attribut US_MAIL).L'adresse indiquée doit uniquement contenir l'adresse “pure”.
- attribut texte : un attribut du document portant le cycle. Cet attribut doit renseigner une adresse email. Il est possible d'utiliser la notation ':' (TST_MYID:THE_MAIL) pour aller chercher des valeurs sur les documents liés.
- attribut relation : un attribut de type 'docid' du document portant le cycle. Cet attribut doit renseigner une personne ou un groupe. Si une méthode ::getMail() est associée à la famille, elle sera utilisée pour renseigner l'émetteur. Sinon, ce sera l'attribut 'US_MAIL'/'GRP_MAIL' de la personne/groupe lié à cet attribut qui sera utilisé. Pour l'émetteur, les adresses de groupes ne peuvent être utilisés, elles ne peuvent être utilisées que pour les destinataires.
- attribut paramètre : un attribut paramètre (PARAM) de la famille du document portant le cycle. Cet attribut doit renseigner une adresse email. Il est possible d'utiliser la notation ':' pour aller chercher des valeurs sur les documents liés.
- attribut cycle : un attribut du document cycle de vie. Cet attribut doit renseigner une adresse email. Il est possible d'utiliser la notation ':' pour aller chercher des valeurs sur les documents liés.
- relation cycle : un attribut de type 'docid' du document cycle de vie. Si une méthode ::getMail() est associée à la famille, elle sera utilisée pour renseigner l'émetteur. Sinon, ce sera l'attribut 'US_MAIL'/'GRP_MAIL' de la personne/groupe lié à cet attribut qui sera utilisé.
- paramètre cycle : un attribut paramètre (PARAM) de la famille du cycle de vie portant le cycle. Cet attribut doit renseigner une adresse email. Il est possible d'utiliser la notation ':' pour aller chercher des valeurs sur les documents liés.

3.10.2.3.3 Définir les destinataires

La définition des destinataires se fait comme pour l'émetteur. Vous pouvez choisir ensuite le mode d'envoi :

- à : destinataire principal
- cc : destinataire en copie
- bcc : destinataire en copie cachée

Si aucun destinataire n'est renseigné, le mail ne sera pas envoyé et un message sera logué permettant de le savoir.

3.10.2.3.4 Définir le sujet

Le sujet est un texte libre. Il peut contenir des parties variables issues du document qui va être envoyé. On notera les attributs (en majuscules) entre crochets. Exemple [BT_APPROVDATE] si le document à un attribut 'BT_APPROVDATE'. La partie entre crochets sera remplacée par la valeur de l'attribut.

3.10.2.3.5 Définir le corps du message

Le corps du messages est un texte HTML. Il peut contenir des parties variables qui sont les attributs du document et les paramètres de transitions du cycle. Ces parties variables peuvent être notées de 3 formes :

1. [MY_ATTR] : cela affichera la valeur *brute* , c'est à dire la valeur inscrite en base de données.
2. [V_MY_ATTR] : cela affichera la valeur *formatée*, c'est à dire telle qu'elle est présentée à l'utilisateur sur l'interface web.
3. [L_MY_ATTR] : cela affichera le label de l'attribut.

Concrètement, par exemple, la valeur brute d'un énuméré sera la clef, la valeur formaté sera la traduction du libellé. Le barre de menu de l'éditeur de texte propose les différentes possibilités pour les attributs formatés du document. Le commentaire de transition peut être récupéré par le mot clef '[WCOMMENT]'. Les propriétés du document peuvent aussi être affichées : [ID] [TITLE] [CDATE]. Par contre, les [V_ID] ne sont pas possibles; ce ne sont pas des attributs. Seul [V_TITLE] est possible pour afficher un lien vers le document.

Si on utilise les valeurs formatées, des hyperliens peuvent apparaitre dans le courriel à destination du serveur d'envoi. Si vous ne souhaitez pas avoir de liens il faut décocher la case 'avec liens'. Ceci est à faire si vous destinez votre courriel à des personnes extérieures à votre système d'information.

3.10.2.3.6 Ajouter des fichiers attachés

Les fichiers attachés font référence à des attributs de type fichier (ou image) du document. Les fichiers seront alors en pièce jointe du courriel. Bien sûr ces attributs peuvent être des listes de fichiers (attribut fichier ou image dans un tableau). Il est possible d'utiliser la notation ':' pour aller chercher des valeurs sur les documents liés.

3.10.2.3.7 Enregistrer le messages

Depuis la version 2.14.2 de dynacase, vous pouvez conserver l'enregistrement du message envoyé. Celui-ci sera

stocké dans la famille “message envoyé”. Son profil sera celui du document servant à l'envoi. Pour stocker le message il faut cliquer sur le champ “Enregistrer une copie”.

3.10.2.3.8 Informations sauvegardées

Pour chaque message envoyé, un message de log sera créé contenant le sujet du message ainsi que ses destinataires.

Si une erreur survient lors de l'envoi du message, elle sera loguée et retournée en pop-up sur l'interface.

3.10.2.4 Ajout de courriel pour les cycles

Une fois le modèle de courriel créé, il faut l'associer au cycle (utiliser les '...' pour sélectionner le modèle. Les envois de courriel sont visibles dans le graphe du cycle avec l'option 'avec les traitements'.

Les modèles de courriels peuvent être associés à un état ou à une transition. Lorsqu'on mets le courriel sur un état il sera envoyé dès que le document changera vers cet état, même si la transition est non définie (cas possible lorsque l'utilisateur 'admin' utilise le cycle).

Version > 2.11.12

3.10.2.5 Minuteurs

Les minuteurs (“timers” in english language) permettent de déclencher des actions sur des documents à des moments définis.

3.10.2.5.1 Définition

Le minuteur dans le cadre du cycle de vie va permettre par exemple de donner une durée limite pour l'activité à réaliser dans un état défini.

Exemple : “vous avez une semaine pour valider ce contrat sinon il sera automatiquement déclaré comme obsolète.”

Nous reprenons ici le même cycle que pour les paragraphes précédents.

Pour mettre en place l'exemple, nous allons créer un minuteur qui fera un changement d'état vers “obsolète” 7 jours après le passage en “initialisé” (ou ici dès que le document est créé). Pour avertir le rédacteur, nous allons l'avertir 3 jours avant en lui précisant qu'il faudrait y penser et ensuite un jour avant, avec en copie le valideur, avant afin qu'il finisse son travail (non non on lui mets pas la pression).

3.10.2.5.2 Création

Pour faire cela nous allons créer un minuteur. Nous éditons le cycle et avec le bouton ”+” de l'attribut “Minuteur Initialisé”.

Nous pouvons indiquer les actions à faire une fois que le minuteur sera attaché au document. Trois types d'actions sont possibles :

1. envoyer un courriel (ou plusieurs) en utilisant les modèles de courriel
2. changer d'état
3. appeler une méthode du document.

Lors d'un changement d'état le minuteur associé à l'état précédent est enlevé. S'il y a un autre minuteur pour l'état suivant il sera activé. Le minuteur peut être mis sur un état ou sur une transition.

Pour associer un minuteur à un état, il faut éditer le cycle de vie et renseigner les attributs minuteurs dans le cadre état ou dans le cadre de la transition voulu. Sur les transitions, les minuteurs “persistants” ne sont pas détachés de manière implicite lors d'un changement d'état. Ils sont détachés lorsqu'ils sont indiqués dans les attributs “Minuteur à détacher”. Ceci permet de lancer des actions qui peuvent durer sur plusieurs transitions.

Lorsqu'un niveau est exécuté (le délai est écoulé), le niveau suivant (rangée suivante du tableau 'configuration') est armé s'il existe. Il est possible de faire des répétitions. Je veux envoyer le même courriel tous les jours pendant 7 jours. Pour faire cela on mettra le nombre d'itérations à 7.

S'il n'y a plus de niveau le minuteur est détaché du document.

Dans notre exemple, nous avons associé le minuteur à l'état “initialise”. Dès que l'on crée un document “test de base”, le minuteur sera activé.

Si l'on regarde l'historique après la création de notre document nous y voyons l'affectation du minuteur.

3.10.2.5.3 Date dynamique

version > 2.13.12

La date de déclenchement de la première action peut être liée à un des attributs de type 'date' ou 'timestamp' du document auquel est attaché un timer. Ainsi, on peut indiquer que le changement d'état vers 'diffusée' sera exécuté à la date marquée dans l'attribut 'sert_testdate'. Si on veut que ce soit 3 jours après cette date, il faudra ajouter au document un attribut 'sert_threedayafter' qui sera calculé en fonction de 'sert_testdate'.

3.10.2.5.4 Administration

L'interface d'administration des minuteurs est disponibles à partir de l'application 'Administration'.

Lorsque l'on clique sur le titre d'un document, cela affiche les prochaines actions qui vont être exécutées.

3.10.2.5.5 Prochains minuteurs

Cela affiche par ordre chronologique les prochaines actions qui vont s'exécuter. dynacase vérifie toutes les 5 minutes les minuteurs qui doivent être exécuté. L'heure d'activation a donc une précision de 5 minutes. Si l'heure d'activation est dépassé de plus de 2 heures (paramètre FDL_TIMERHOURLIMIT) le minuteur sera ignoré.

3.10.2.5.6 Historique des minuteurs

Cela affiche les actions, issus de minuteurs, déjà exécutés. Il sont ordonnés par ordre chronologique d'éxecution.

3.10.2.5.7 Minuteurs dépassés

Liste les minuteurs qui aurait due se déclencher depuis plus de 2 heures (FDL_TIMERHOURLIMIT). Cela peut arriver si le serveur a été arrêté pendant plus de 2 heures. Le paramètre applicatif FDL_TIMERHOURLIMIT est modifiable avec l'application de paramétrage :administration/paramètre de configuration/paramètres applicatifs/Bibliothèque dynacase.

3.10.2.5.8 Nettoyage de l'historique

Permet d'effacer les traces des minuteurs déjà exécutés depuis un certain nombre de jours. Ce nombre de jour est par défaut de 7. Il peut être modifié dans la zone de saisie contigüe. Le nettoyage prend en compte le filtre sur les documents : ne seront nettoyer que ceux dont le titre contient le filtre.

3.10.2.5.9 Désactiver les minuteurs

Cela permet d'annuler des minuteurs actifs. Si le filtre est vide tous les minuteurs seront annulés. Si le filtre n'est pas vide seul les minuteurs attachés aux documents filtrés seront pris en compte.

3.10.2.6 Les accords

Sur certains états d'un document, vous pouvez associer des questions telles que 'Approuvez-vous le documents ?' ou 'Donnez votre appréciations'. La réponse à ces questions est un ensemble de choix fini.

Lors de l'édition d'un cycle vous associez un ou plusieurs “Accord” à un état. L'accord est un document de la famille “accord” qui défini la question et les réponses possibles.

3.10.2.6.1 Configurer un accord

Pour définir qui doit répondre à la question associées on utilise le mécanisme de profil du document accord. Premièrement vous changez le profil en “contrôle dédié” (menu sécurité/changer de profil). Ensuite vous lancer l'interface de configuration du profil (menu sécurité/accessibilités).

Couleur	Voir	éditer	Supprimer	Envoyer	Déverrouiller	Confidentiel	Forum	Voir les réponses	Répondre	Voir les droits	Modifier les droits
Approbateurs	●								●		
Chef de pole/département (CD)	●	●							●		
Chef de section (CS)	●	●							●		
Utilisateurs	●										

Le droit 'Répondre' ('answer') indique quels sont les groupes et les utilisateurs qui doivent répondre à la question. Ceux qui n'ont pas ce droit n'ont pas à répondre au questionnaire. Le profil peut être dynamique si vous renseigné l'attribut “famille” dans la cadre “Profil dynamique”. Dans ce cas les attributs relations de la famille défini apparaîtrons dans le profil. Vous pouvez ainsi indiquer en fonction du document qui doit répondre si vous avez défini dans votre document des attributs relations qui contient les questionnés.

3.10.2.6.2 Fonctionnement des accords

Une fois l'accord créer et configuré, nous pouvons l'associer à un état du cycle. Lorsqu'un document consulte un document par l'interface, il consulte généralement la dernière révision du document, celle qui vie et qui, peut-être, a été modifiée depuis le dernier changement d'état. **La question ne peut être posée que sur un document figé.** On ne peut demander un avis sur un document qui peut être modifié. Imaginons que l'on mette un accord sur l'état “publié”, la question de l'accord sera posé que sur les documents “publiés”. Si le document est dans sa dernière révision, une fenêtre avertira l'utilisateur qu'il doit fournir une réponse sur le document publié (dans sa dernière

version).

Lorsque vous cliquer sur “aller au dernier publié” alors le document qui fait l'objet de la question apparaît ainsi que le questionnaire. Une fois que l'utilisateur à répondu aux questions, il peut revenir à la dernière révision en cliquant sur “Révision courante”. La fenêtre d'avertissement concernant les accords ne s'affiche plus s'il a déjà répondu. Pour changer sa réponse, il faut passer par l'historique afin d'avoir le dernier document publié puis ensuite cliquer sur “Voir mes accords” pour changer son avis.

3.10.2.6.3 Gérer les réponses

Dans les profils de document vous avez le droit 'Voir les réponses' (wask). Si vous disposez de ce droit dans le document où porte l'accord, le menu “Voir les réponses” sera disponible dans le document. Ce menu liste les utilisateurs qui ont répondu (trié par réponse) et ceux qui n'ont pas répondu. Cette liste est issue de la configuration du profil de l'accord : ceux qui ont le droit 'Répondre'.

3.10.2.7 Cycle de vie de publication de documents

3.10.2.7.1 But de ce document

Expliquer comment mettre en place un cycle de vie de publication de documents classique du genre « Bouillon → Validé » mais avec la possibilité de repasser à l'état brouillon tout en laissant la dernière version validée accessible aux utilisateurs.

3.10.2.7.2 Règles à mettre en place

Voici les règles de fonctionnement que devra respecter notre famille de documents :

- Un document à l'état brouillon n'est ni visible ni modifiable par le groupe « Lecteurs »
- Un document à l'état brouillon est modifiable uniquement par le groupe « Rédacteurs »
- Seule le groupe « Rédacteur » peut changer l'état du document pour le passer à l'état « Validé » ou le repasser à l'état « Brouillon »
- Un document à l'état « Validé » est visible du groupe « Rédacteur » et « Lecteur », mais n'est pas modifiable
- Pour modifier un document « Validé », le groupe « Rédacteur » doit le repasser à l'état « Brouillon »

3.10.2.7.3 Cycle de vie et profils

Notre cycle de vie aura donc deux états avec deux profils associés :

- Brouillon : Le profil associé donne accès au document uniquement au groupe « Rédacteurs »
- Validé : Le profil associé donne un accès en lecture seule à tout le monde

Droits sur les changements d'états :

- Les rédacteurs seront les seules à accéder au cycle de vie pour changer le document d'état et seront également les seules à pouvoir modifier le document quand il sera à l'état « Brouillon »

Au final, il faut donc :

- Créer la famille de cycle de vie
- Créer le document cycle de vie
- Associer le cycle de vie la famille
- Créer les deux profils
- Associer les deux profils au cycle de vie

3.10.2.7.4 Rapports permettant d'accéder aux documents

Un rapport pour le groupe « Lecteur » permettra de consulter la liste de tous les documents à l'état « Publié » et d'y accéder. Un autre rapport pour le groupe « Rédacteur » permettra d'accéder aux documents à l'état « Brouillon »

3.10.2.7.5 Liste des profils nécessaires

- Profil pour la famille
- Profil pour le document à l'état « Brouillon »
- Profil pour le document à l'état « Publié »
- Profil du cycle de vie (Contrôle dédié)

Envoi d'un mail Un mail sera envoyé aux personnes du groupe « Lecteurs » lors de la publication du document.

3.10.2.7.6 Menu personnalisés pour la famille

- Un menu dans le document permettra d'accéder au rapport affichant la liste des documents publiés
- Un autre menu accessible uniquement par le groupe « Administrateur » affichera le rapport des documents à l'état Brouillon
- Le menu « Révision courante » sera accessible uniquement au groupe « Rédacteur ».

3.10.2.7.7 Installation des documents de l'exemple

Voici la liste des documents à installer :

-

```
FIXME internalmedia: freedom_2.14:admin:cycle_de_vie:class.wpublication.php.txt
```

contient la Class de notre cycle de vie. Il faut enregistrer le contenu de ce fichier dans “FDL / Class.WPublication.php”

-

```
FIXME internalmedia: freedom_2.14:admin:cycle_de_vie:method.publication.php.txt
```

- contient la Méthode associé à notre famille pour limiter l'accès aux menus. Il faut enregistrer le contenu de ce fichier dans “FDL / Method.Publication.php”

```
FIXME internalmedia: freedom_2.14:admin:cycle_de_vie:mail_publication.xml.txt
```

- contient le contenu du message envoyé aux lecteurs lors du passage à l'état “Publié”. Il faut enregistrer le contenu de ce fichier dans “FDL / Layout / mail_publication.xml”

```
FIXME internalmedia: freedom_2.14:admin:cycle_de_vie:publication_0.1.ods
```

- permet d'importer dans dynacase tous les documents nécessaires

Une fois le fichier OOo importé, vous devriez avoir dans le dossier “Default Master / Publication”, les fichiers suivant :

Remarques :

- Après avoir importé une première fois le document OOo, il faut initialiser le document cycle de vie et importer une deuxième fois le document OOo pour que le cycle de vie soit correctement paramétré.
- Il faut également activer les profils et modifier les accessibilités pour que le système soit opérationnel (ex : Il faut compléter le groupe des rédacteurs pour que le rapport permettant d'accéder aux brouillons soit accessible)

3.10.2.7.8 Résultats

Exemple de document à l'état publié :

Exemple de document à l'état brouillon en édition :

Rapport affichant la liste des documents publiés :

3.11 Créer une nouvelle application dynacase

3.11.1. But de ce document

Montrer comment créer de nouvelles applications dans dynacase. Une application, permet de regrouper des familles ou des actions.

3.11.2. Arborescence des fichiers d'une application

Pour créer une nouvelle application, il faut au minimum deux fichiers :

```
MONAPPLI.app
```

et

```
MONAPPLI_init.php.in
```

Si l'application est visible, il faut ajouter l'icône de l'application (image carrée entre 48 et 68 px, format png si possible) :

```
Images/votre_image.png
```

Si l'application contient des actions, il faut ajouter les fichiers .php et/ou .xml des différentes actions :

```
MONAPPLI/Actions/action1.php
MONAPPLI/Actions/action1.xml
```

3.11.3. MONAPPLI.app

Ce fichier PHP décrit l'application : éléments de présentation `$app_desc = array(…)`, actions de l'application `$action_desc = array(…)` et les acl (droits) `$app_acl = array (…)`. Exemple de contenu minimun :

```
<?php

$app_desc = array (
 "name"          =>"MONAPPLI",
 "short_name"    =>"Mon application",
 "description"   =>"Mon application de test",
 "access_free"   =>"Y",
 "icon"          =>"cycle.gif",
 "displayable"   =>"Y",
 "with_frame"    =>"Y",
 "childof"       =>"ONEFAM"
);

?>
```

Pour un aperçu plus complet de ce fichier, rendez-vous à la page actions particulières.

Variables	Descriptions
name	Nom de l'application tel qu'il apparaîtra dans les menus de configuration de dynacase. Ce nom doit également correspondre exactement au nom du dossier contenant l'application.
short_name	Nom de l'application qui apparaîtra dans le menu général pour les utilisateurs (si l'application est visible).
description	Nom qui apparaît en info-bulle en laissant la souris sur le nom de l'application dans le menu général (si l'application est visible).
access_free	Indique que les droits d'accès à l'application ne sont pas contrôlés
icon	Nom du fichier disponible dans « **/usr/share/what/Images** » ou dans « **/usr/share/what/MONAPPLI/Images** »**Remarque** : Le script « **/usr/share/what/wstart** » créera automatiquement un lien de cette image dans le dossier « **/usr/share/what/Images** »
displayable	Si « Y », l'application apparaîtra dans le menu général pour les utilisateurs sinon, elle sera invisible.**Remarque** : Une application invisible peut être utilisée pour y stocker des actions. Par exemple l'application « GENERIC » est invisible mais dispose de nombreuses actions utilisées par l'application « ONEFAM »
with_frame	n'est plus utilisé mais obligatoire pour le fonctionnement des anciennes applications
childof	Indique si cette application hérite d'une autre. Il est possible de faire dériver une application de n'importe quelle autre application

	existante (ex : ONEFAM, WGCAL,..). Cette technique est principalement utilisée pour créer des applications basées sur ONEFAM pour regrouper les familles par fonctionnalités.

3.11.4. MONAPPLI_init.php

Ce fichier PHP contient les variables déclarées par l'application (globales, applicatives, utilisateurs). Il contient au moins la version de l'application.

Ces variables sont accessibles par l'administrateur, via l'application Administration.

Remarque : Il n'est pas obligatoire d'augmenter le numéro de version pour que la mise à jour soit prise en compte avec la commande ”–method=update”. Par contre la commande “wcheck” vérifie que la version a changé pour effectuer la mise à jour.

Déclaration de paramètres applicatifs : Une ou plusieurs valeurs, de type string, integer, ... peuvent être nécessaires pour le fonctionnement d'une application, tout en restant modifiables par l'administrateur. Pour déclarer ces paramètres, il faut ajouter un tableau dans le code du fichier MONAPPLI_init.php.ini. Exemple :

```
<?php
global $app_const;
$app_const= array(
  "VERSION" =>"@VERSION@-@RELEASE@",
  "Name_of_the_parameter"    =>array("val"=>"1",
                                     "descr"=>N_("Description of the
parameter"),
                                     "global"=>"Y",
                                     "user"=>"N")
                 );
?>
```

Variables	Descriptions
val	indique la valeur par défaut du paramètre. Cette valeur est modifiable par l'administrateur (Menu “Administration”/“Paramètres applicatifs”/ puis choisir l'application).
descr	permet de noter une description, ce texte étant affiché dans le champs du même menu que ci-dessus.
global	indique si le paramètre est global.
user	indique si la valeur du paramètre est personnalisable pour chaque utilisateur ayant accès à l'application : “Y” ou “N”. En cas de “Y”, le choix apparaîtra, pour l'administrateur, dans le menu “Administration”/“Paramètres utilisateurs”/ puis choisir l'utilisateur).
kind	Le type de paramètre. Si pas précisé cela est un type texte. Les valeurs acceptées sont : enum, color, password. static enum : Pour la valeur "enum" il faut préciser les valeurs possible de l'énuméré : exemple : enum(yes|no) color : un color picker sera alors proposé pour choisir la couleur password : La valeur ne sera pas affichée. Elle sera remplacé par '*****' pour montrer qu'il y a un mot de passe static : La valeur est non modifiable par l'interface d'administration

3.11.5. Initialisation ou mise à jour de l'application

Pour initialiser l'application, il faut générer le paquet webinst (make webinst) et l'installer ou le mettre à jour via dynacase-control.

Remarque : Une fois l'initialisation effectuée, il sera possible de faire les mises à jour avec cette même commande ou en passant par l'interface graphique avec ce menu :

- Menu “Administration / Les applications” et cliquer sur l'icône « mise à jour » en forme de flèche qui tourne.

3.11.6. Liens pour avoir d'autres informations

- Le chapitre Actions particulières vous donnera un exemple de création d'une application contenant des actions.
- L'application Web Externe (FREEDOM-URL) utilise ce principe pour intégrer des sites Web dans dynacase.

3.11.7. Actions particulières

les actions particulières des documents font appel à des actions WHAT. Une action est le déroulement d'un code PHP et la production d'une sortie (en général HTML), l'action peut être vue comme le traitement d'une requête particulière :

3.11.7.1 Principe et spécification des actions particulières

3.11.7.2 Introduction

Les actions particulières des documents font appel à des actions du noyau. Ces actions doivent être définies au sein d'une application.

L'application déployée sur le serveur aura la structure suivante:

arborescence application

La description de l'action se fait dans le fichier .app. Ce fichier contient la liste des actions ainsi que la liste des droits d'accès gérés par cette l'application.

Le fichier d'exemple MONAPP.app définit 3 droits d'accès et 4 actions. Une action est définie à l'aide d'un fichier PHP et d'un layout XML. Le fichier PHP ou le layout est optionnel, mais il faut au moins un des deux. A moins que l'application soit déclarée libre d'accès, chaque action doit définir son droit d'accès (ACL). Si l'utilisateur n'a pas le

droit défini, l'exécution de l'action sera interdite.

On définit généralement un fichier PHP par action. Ce fichier contient la fonction action dont le nom est le même que celui du fichier. Les fichiers PHP et XML sont calculés par la fonction d'interprétation du fichier .app. Le nom de l'action ne doit pas dépasser 30 caractères!

Les actions doivent se trouver dans le répertoire Actions de votre module.

action	fichier PHP	fichier XML
MONACTION1	monaction1.php fonction : monaction1(&$action)	monaction1.xml
MONACTION2	monaction2.php fonction : monaction2(&$action)	monaction2.xml
MONACTION3		
MONACTION4	monaction4.php fonction : monaction4(&$action)	**monaction2**.xml

La fonction d'action appelée a un seul paramètre qui est la référence à l'objet action. Cet objet contient la référence au layout (Action::lay).

3.11.7.3 Fichier 'MONAPP.app'

```
<?php
$app_desc = array (
    // nom
    "name" => "MONAPP",
    //  description court
    "short_name" => N_("mon application"),
    // description complète
    "description" => N_("mon application de test"),
    // "N" => L'application n'apparaîtra pas, par défaut, dans le
    // menu des utilisateurs.
    // Il faudra alors spécifier l'accès, au cas par cas, dans
    // le menu "Accessibilités".
    "access_free" => "N",
    // Icone
   "icon" => "monappli.png",
   // Doit être affiché dans le bandeau en haut (Y,N)
   "displayable" => "N",
   // "N" => les balises <html><head> seront rajoutées au
   // layout de l'application
   "with_frame" => "N",
   // héritage d'une application
   "childof" => ""
);

$app_acl = array (
  array(
   "name" =>"NORMAL",
   "description" =>N_("Access to common action"),
   // "Y" => le groupe "Utilisateurs" sera affecté à l'ACL
   "group_default"       =>"Y"),

  array(
   "name" =>"EDIT",
   "description" =>N_("Access to edit action"),
   "group_default"       =>"Y"),
  array(
   "name" =>"EXPORT",
   "description" =>N_("For export functions"),
   "group_default"       =>"N")
);
```

```
$action_desc = array (
  array(
   "name" =>"MONACTION1",
   "short_name" =>N_("action one"),
   "acl" =>"NORMAL" ,
   // action initiale lorsque l'utilisateur appuis sur
   // l'icone de l'application
   "root" =>"Y") ,
  array(
   "name" =>"MONACTION2",
   "short_name" =>N_("action two"),
   "acl" =>"EXPORT" ) ,
  array(
   "name" =>"MONACTION3",
   "short_name" =>N_("action three"),
   "acl" =>"EDIT" ,
   "script" =>"monaction1.php",
   "function" =>"monaction3") ,
  array(
   "name" =>"MONACTION4",
   "short_name" =>N_("action four"),
   "acl" =>"NORMAL" ,
   "layout" =>"monaction2.xml")
);
?>
```

Le fichier <NOMAPP>_init.php est obligatoire. Il contient la liste des paramètres propres à l'applicatif. Parmi ces paramètres, VERSION est obligatoire car il est utilisé pour les mises à jour éventuelles de l'application.

3.11.7.4 Fichier 'MONAPP_init.php'

```
global $app_const;
$app_const= array(
"VERSION" =>"0.0.1"
);
```

3.11.7.5 Fichier 'monaction1.php

Le fichier *monaction1.php* doit être au nom de l'action, de préférence en minuscule. Il contient au moins une fonction du même nom que l'action, avec comme paramètre *&$action* (passage par référence) :

```
function monaction1(Action &$action) {
//...
}
```

Cette fonction peut en appeler d'autres, définies elles aussi dans ce fichier php.

3.11.7.5.1 Récupérer des paramètres

Un paramètre applicatif :

```
$action->getParam("nom_du_parametre");
```

La base de données :

```
$db=getParam("FREEDOM_DB");
```

L'utilisateur courant :

```
$action->user ;
```

Pour avoir accès à l'application (retourne un objet de la classe application) :

```
$app=$action->parent
```

3.11.7.5.2 Récupérer des valeurs passées dans l'URL

L'action peut être lancée par l'activation d'un lien hypertexte. L'URL appelée peut contenir des paramètres, dont les valeurs sont récupérées par l'intermédiaire de la méthode Action::getArgument :

```
$val = $action->getArgument("url_val","val_par_defaut");
```

3.11.7.5.3 Vérifier les arguments et donner l'usage

La classe ActionUsage permet de valider que les arguments sont valides (attribut obligatoire / optionnel).

Si l'option strict est mise à 'true' (valeur par défaut), tout argument non compris dans l'usage provoquera une erreur et l'action sera déroutée dès l'appel à ::verify().

```
function zoo_color(Action &$action)
{
    $red = $quality = '';
    $usage = new ActionUsage($action);
    $red = $usage->addOption("red", "red level", array(), 128);
    $quality = $usage->addOption("quality", "quality", array(), 20);
    $usage->strict(true);
    $usage->verify();

    if (!is_numeric($red)) {
        $usage->exitError('red must be a integer');
    }
    if (!is_numeric($quality)) {
        $usage->exitError('quality must be a integer');
    }
```

La méthode ActionUsage::exitError() exécute un Action::exitError() en ajoutant l'usage en plus. Ces méthodes renseignent le champ "Warning" dans le header http avec la texte indiqué (limité à la première ligne).

De plus, la méthode ActionUsage::exitError() peut-être appelée avec son paramètre $useException à true ActionUsage::exitError(true), elle lancera alors une exception de type ApiUsageException si le verify échoue.

3.11.7.5.4 Afficher/Transmettre des données dans le layout

La valeur *val_to_be_sent* sera affichée à la place du code [xml_data] (du fichier xml) :

```
$action->lay->set("xml_data","val_to_be_sent");
```

Paramètre	Signification
xml_data	Nom du champ destinataire dans le fichier XML
val_to_be_sent	Valeur à afficher/transmettre

3.11.7.5.5 Warning/Astuces

- L'appel à *$this*→ ne fonctionne pas,
- Pour manipuler un objet de classe documentaire, ne pas oublier :

```
include_once("FDL/Class.Doc.php");
```

- Sortir de la méthode :

```
$action->exitError("message");
```

3.11.7.6 Fichier monaction1.xml

Le layout de l'action peut faire référence au layout FDL. Pour cela, le contenu du fichier est inséré entre les 2 balises :

```
[ZONE FDL:HTMLHEAD]
[...]
[ZONE FDL:HTMLFOOT]
```

Dans la balise [ZONE FDL : HTMLHEAD], on peut rajouter

```
&title=mon titre
```

ce qui changera le titre de la page web.

Afficher une zone de texte composée d'un libellé et d'une zone de saisie :

```
  <input type="text" name="texte_du_libelle"
    value="Texte_a_afficher_dans_le_libelle">
```

Afficher un bouton “Valider” :

```
<input type="submit" value="Valider">
```

3.11.7.7 Gestion des droits applicatifs

Certaines actions (par ex : exécuter MONACTION2) peuvent être réservées à un (ou plusieurs) groupe(s) d'utilisateurs. Pour limiter cet accès, vous pouvez l'indiquer par import de tableur dont une ligne respecte la mise en forme suivante : L'accès d'un groupe d'utilisateurs à une ACL peut être indiqué lors de l'importation des groupes utilisateurs. Pour cela, il faut ajouter une ligne dans le tableur d'import selon :

mot clé	identifiant du groupe	identifiant de l'application	nom de l'acl
ACCESS	Group_A	MONAPP	EXPORT

L'identifiant du groupe peut être le nom logique d'un groupe ou son identifiant système (attribut 'us_whatid' des documents "utilisateur" et "groupe intranet").

Le nom de l'acl peut être précédé d'un signe '-' pour spécifier que le droit est enlevé.

Exemple pour enlever le droit 'FREEDOM_GED' sur le groupe "Utilisateurs" :

```
     ACCESS;GDEFAULT;FREEDOM;-FREEDOM_GED
```

3.11.7.7.1 Export/import des droits applicatifs

Les droits applicatifs peuvent être exportés dans un fichier au format CSV, et peuvent être importés à partir de ce même fichier au format CSV.

3.11.7.7.1.1 Export des droits applicatifs

L'export est accessible dans l'application "Accessibilités" > onglet "Import/Export" > "Télécharger la configuration des droits applicatifs".

Un fichier "access.csv" est alors proposé au téléchargement.

Lors de l'export, les groupes et utilisateurs qui n'ont pas de nom logique seront exportés avec leur identifiant système.

3.11.7.7.1.2 Import des droits applicatif

L'import d'un fichier "access.csv" est accessible dans l'application "Accessibilités" > onglet "Import/Export" > "Importer un fichier de configuration des droits applicatis".

Une interface d'import de documents dynacase est alors affichée, et vous pouvez entrer votre fichier CSV dans le champ "Fichier à importer", faire une "Analyse" de celui-ci, et lancer l'import avec "Importer les documents".

3.11.7.8 Initialisation de l'application et des actions

Pour initialiser l'application, on peut utiliser l'utilitaire wcheck. Si les fichiers .app et _init.php sont présents et corrects la nouvelle application doit être listée par cet utilitaire.

L'utilitaire wcheck ne fait les mises à jours d'application que si la version est supérieure à celle déjà enregistrée. Pour forcer une mise à jour, vous devez lancer la commande suivante :

```
# ./wsh.php --api=appadmin --method=update --appname=MONAPP
MONAPP...updateLOG::(I)::Init : MONAPP
LOG::(I)::Acl Modify : NORMAL, Access to common action
LOG::(I)::Acl Modify : EDIT, Access to edit action
LOG::(I)::Acl Modify : EXPORT, For export functions
LOG::(I)::Update Action MONACTION1
LOG::(I)::Update Action MONACTION2
LOG::(I)::Update Action MONACTION3
LOG::(I)::Update Action MONACTION4
```

3.11.7.9 Action dans les menus contextuels

L'appel à une action se fait à l'aide de l'URL

```
%S%app=<appname>&action=<actname>&...<autres paramètres>.
```

Par exemple, on ajoute l'attribut menu suivant dans la famille société :

Id	Description	Vis	lien
SI_MONMENU	Les sites	W	%B %app=MONAPP&action= MONACTION1&docid=%I %

L'action MONACTION1 revoit la liste des sites de la société. Les sites sont présentés avec la vue résumé. L'utilisateur peut cliquer sur le nom du site pour avoir le descriptif complet du site.

```
<?php
include_once("FDL/Class.Doc.php");
function monaction1(Action &$action) {
  // les paramètres HTTP
  $docid = $action->getArgument("docid",0);  // document société
  if ($docid == 0) $action->exitError("identifiant non spécifié");
  $dbaccess = $action->getParam("FREEDOM_DB");
  $doc= new_Doc($dbaccess, $docid);
  $s=new SearchDoc($dbaccess, "SITE");
  $s->addFilter("si_idsoc='%s'", $doc->initid);
  $tdoc=$s->search();
  // remplissage des données pour le bloc
  $tsites=array();
  foreach($tdoc as $k=>$v) {
    $tsites[]=array("site"=>$v["title"],
     "idsite"=>$v["id"]);
  }
  // on renseigne le layout
```

```
  $action->lay->set("societe",$doc->title);
  $action->lay->setBlockData("SITES",$tsites);
}
?>
```

Exemple : liste des sites

```
Liste des sites de [societe]

<UL>
[BLOCK SITES]
<LI><A href="[CORE_STANDURL]app=FDL&action=FDL_CARD&id=[idsite]"
target="s[idsite]">
[site] </A>
<iframe name="s[idsite]" width="100%"
src="[CORE_STANDURL]app=FDL&action=IMPCARD&id=[idsite]&zone=FDL:VIEWABSTRACTCARD
:T"></iframe>
</LI>
[ENDBLOCK SITES]
<UL>
```

Le résultat de cette action sur la société zoo net montre ses deux sites. La deuxième capture d'écran montre la représentation lorsque l'utilisateur a cliqué pour voir les vues complètes.

résultat de cette action sur la société zoo

vue complète

Cette action peut être aussi utilisée dans la famille site :

Id	Description	Vis	lien
SI_MONMENU	Les autres sites	W	%S %app=MONAPP&action=MONACTION1 &docid=%SI_SOCID%

Ce type d'action peut aussi ce faire avec une vue. On préférera créer une action lorsqu'elle celle-ci n'a pas de rapport avec une famille de document précise et lorsqu'elle peut être réutilisée dans d'autre contexte.

La visibilité de l'attribut menu peut être contrôlée par une méthode de l'objet documentaire.

Id	Description	Vis	lien	phpfunc
SI_MONMENU	Les autres sites	W	%S %app=MONAPP&action=MONACTION1&docid=%SI_SOCID%	::controle_site()

Pour cela on indique la méthode utilisée dans la colonne phpfunc. Cette méthode doit retourner la visibilité voulue :

- MENU_ACTIVE : affiche l'action dans le menu
- MENU_INVISIBLE : n'affiche pas l'action dans le menu
- MENU_INACTIVE : affiche l'action mais n'est pas activable dans le menu

Ces trois valeurs sont des énumérés[34]. Il faut donc retourner cette valeur sans guillemets.

```
/**
   * return true if it is the latest revision
   * @return enum
   */
   public function controle_site() {

    if ($this->locked != -1) return MENU_ACTIVE;
    return MENU_INVISIBLE;
  }
}
```

3.11.8. Migration de modules

3.11.8.1 Scripts de pré-migration et de post-migration

Lors de la mise à jour d'un module, il est possible d'exécuter des scripts pour effectuer des opérations de migrations ou des traitements spécifiques.

Deux types de scripts sont possibles :

- scripts de **pré-migration** (premigr) : les scripts de pré-migration (premigr) sont prévus pour être exécutés **avant** que l'application et les familles en base de données soient mis-à-jour.
- scripts de **post-migration** (postmigr) : les scripts de post-migration (postmigr) sont prévus pour être exécutés **après** que l'application et les familles en base de données aient été mis-à-jour.

Les scripts doivent être situés dans le sous-répertoire de l'application et leur nom doit être de la forme :

- scripts de pré-migration (premigr) : "${APPNAME}**_premigr_**${VERSION}"
- scripts de post-migration (postmigr) : "${APPNAME}**_postmigr_**${VERSION}"

NOTE : les scripts de migration sont exécutés sur des changements de VERSION sans tenir compte des RELEASE. On ne peut donc pas exécuter de script de migration lors d'un changement de RELEASE.

Le lancement de ces scripts est commandé dans le fichier `info.xml' du module par l'utilisation d'une directive `**<process command="programs/(pre|post)_migration ${APPNAME}" />**' dans la section `<post-upgrade/>'.

Exemple de déclaration type pour le lancement de scripts de pré-migration et de post-migration :

```
<module name="FOO" version="2.0.0" release="1">
  ...
  <post-upgrade>
    <process command="programs/pre_migration FOO" />
    <process command="programs/record_application FOO" />
    <process command="programs/app_post FOO U" />
    <process command="programs/post_migration FOO" />
```

34) *constante PHP, pas des chaînes de caractères*

```
    <process command="programs/update_catalog" />
  </post-upgrade>
</module>
```

Les programmes `**pre_migration**' et `**post_migration**' exécutent tous les scripts de "premigr" ou "postmigr" dont la version est comprise entre la version du module actuellement installé, et la version du module mise-à-jour.

Les scripts de migration sont généralement écrits en "**bash**".

3.11.8.2 Variables d'environnement

Lorsque les scripts "premigr" et "postmigr" sont exécutés, les variables d'environnement suivantes sont définis :

Variable	Valeur
`wpub`	Le chemin d'accès au contexte sur lequel est effectué la mise-à-jour. Exemple : "/var/www/dynacase"
`httpuser`	L'UID du process httpd/Apache. Exemple : "www-data"
`pgservice_freedom`	Le nom du service PostgreSQL d'accès à la base de données Dynacase. Exemple : "dynacase"
`freedom_context`	La valeur est toujours "`default`".

`MODULE_VERSION_FROM`	La version du module actuellement installée (de la forme "VERSION-RELEASE"). Exemple : "1.2.3-4"
`MODULE_VERSION_TO`	La nouvelle version du module (de la forme "VERSION-RELEASE"). Exemple : "2.3.4-5"

`WIFF_CONTEXT_NAME`	Le nom du contexte sur lequel est effectué la mise-à-jour. Exemple : "dynacase"
`WIFF_CONTEXT_ROOT`	Le chemin d'accès au contexte sur lequel est effectué la mise-à-jour. Exemple : "/var/www/dynacase"
`WIFF_ROOT`	Le chemin d'accès au répertoire dans lequel est installé dynacase-control. Exemple : "/var/www/dynacase-control"

3.11.8.3 Comparaison de versions

Il peut être nécessaire, pour un script de migration, d'effectuer des opérations différentes en fonction de la version de départ du module.

Pour cela, la variable d'environnement `**$MODULE_VERSION_FROM**' peut-être utilisée pour obtenir la version du module actuellement installé.

Ensuite, pour comparer des versions représentés sous forme de chaîne de caractères, il vous faudra utiliser la fonction `**versionCompare**' déclarée dans le fichier bash `**$wpub/libutil.sh**' :

Syntaxe :

```
versionCompare $V1 $V2
```

La fonction prend deux arguments, qui sont les versions à comparer, et affiche sur STDOUT un entier :

- inférieur à 0 si $V1 est inférieur à $V2
- supérieur à 0 si $V1 est supérieur à $V2
- égal à 0 si $V1 est égal à $V2

Il faut ensuite utiliser les opérateurs de comparaison arithmétique ("-lt", "-gt", "-le", "-ge", "-eq", "-ne") de bash :

```
#!/bin/bash

. "$wpub/libutil.sh"

CMP=$(versionCompare "$MODULE_VERSION_FROM" "1.0.0")
if [ $CMP -lt 0 ]; then
  echo "La version de départ est inférieure à 1.0.0"
else
  echo "La version de départ est supérieure ou égale à 1.0.0"
fi
```

3.12 Localisation (Traduction de dynacase dans d'autres langues)

3.12.1. But de ce chapitre

Expliquer comment mettre en place la localisation dans **dynacase**.

Pour développer dans de bonnes conditions il est indispensable d'utiliser les techniques énoncées dans ce document même si **dynacase** ne sera utilisé qu'en Français pour plusieurs raisons :

- Il est déconseillé de mettre des accents dans les fichiers utilisés par **dynacase** (.php, .xml, .app,..)
- Il n'est pas possible de mettre des accents aux noms des étapes des cycles de vie sans passer par ces techniques de localisation

3.12.2. Quel mécanisme utilise dynacase pour la localisation ?

dynacase utilise le système le plus répandu dans le monde des logiciels libres à savoir Gettext

3.12.3. Localiser son application

3.12.3.1 Localisation des familles

Les familles dynacase sont décrites via les fichiers .ods. Les différentes informations issues de ces fichiers et présentées à l'utilisateur (libellé, énumérés, etc...) peuvent être traduites via la localisation. Aucune syntaxe particulière, ni mise en évidence des textes à traduire n'est nécessaire.

3.12.3.2 Localisation des fichiers .php (Methode, Class, Action, Api,..)

A chaque fois qu'une chaine de caractères est susceptible d'être traduite ou contient des accents il est fortement

conseillé d'utiliser la fonction “gettext()” ou sa forme simplifiée “_()”

Exemple avec un extrait du fichier “Method.Report.php” :

```
function _getInternals() {
       return array("title"    => _("doctitle"),
             "revdate"  => _("revdate"),
       "revision" => _("revision"),
       "state"    => _("state"));
  }
```

Autre exemple avec la fonction “sprintf” :

```
$err=sprintf(_("Unable to connect to LDAP server %s"),$ldaphost);
```

En conclusion, il suffit de placer à l'intérieur de la fonction “_()” chaque chaine de caractères à localiser.

3.12.3.3 Localisation des fichiers .xml (Layout et Action)

Comme il n'est pas possible dans un ficheir .xml d'utiliser une focntion PHP, il existe une autre technique. Il faut placer la chaine à traduire entre crochets et la faire précédé du mot clé réservé “TEXT”. Exemple :

```
<h1>[TEXT:Title of document]</h1>
```

Remarque : Vous trouverez de nombreux exemples dans les fichiers .xml fourni par défaut dans dynacase.

3.12.3.4 Localisation des fichiers .app (Application)

Le contenu des fichiers .app est enregistré dans la base de données et ne doit pas être localisé.

Pour ces fichiers, il faut utiliser la fonction spécifique à dynacase “N_()”. Cette fonction retourne la chaine elle même sans la traduire. Cette fonction permet d'identifier les chaines à traduire pour pouvoir utiliser les traductions dans les différentes interfaces d'administration. Exemple :

```
$app_desc = array (
   "name" => "ONEFAM",          //Name
   "short_name" => N_("Onefam"), //Short name
   "description" => N_("One Familly Management"), //long description
```

3.12.3.5 Localisation des cycles de vie

Pour localiser un cycle de vie, c'est un peu particulier car nous ne pouvont pas utiliser la fonction “_()” directement dans les tableaux de définition des étapes et des transitions.

Il faut donc placer cette fonction en commentaire mais en utilisant uniquement le signe ”#” et sans le placer en début de ligne car dans le cas contraire il ne serait pas analysé.

Voici un un exemple pour les transitions :

```
var $transitions = array ( # _("T1") _("T2") _("T3")
   "T1" =>array(),
                         "T2" =>array(),
                         "T3" =>array());
```

Et un autre pour les étapes :

```
var $cycle = array( # _("Etape1") _("Etape2") _("Etape3")
          array("e1" => "Etape1",
        "e2" => "Etape2",
```

```
        "t"  => "T1"),
...
```

3.12.3.6 Localisation des attributs des familles

Pour activer la localisation des libellés d'attribut, il est nécessaire d'indiquer dans le Makefile principal les fichiers ods à scruter. Ceci se fait en ajoutant les chemins des fichiers dans la variable `TRANSODS`.

```
# ============================================
# $Id: Makefile.in,v  $
# ============================================
PACKAGE = @PACKAGE@
VERSION = @VERSION@
...
include $(utildir)/PubRule

TRANSODS += mesfamilles.ods
...
```

Ensuite cela produit des entrées dans les fichiers po comme ci-dessous :

```
#, fuzzy
msgid "ENTITE#title"
msgstr "Entité"

 #, fuzzy
msgid "ENTITE#ae_type"
msgstr "Type"

#, fuzzy
msgid "ENTITE#ae_campagne"
msgstr "Campagne"

#, fuzzy
msgid "ENTITE#ae_type#yes"
msgstr "Type"

#, fuzzy
msgid "ENTITE#ae_type#no"
msgstr "Type"
```

l'identificateur est composé du nom de la famille, suivi de dièse puis de l'identifiant de l'attribut. Le titre de la famille est aussi traduisible avec la clef nom de la famille suivi de `#title`. Les libellés des valeurs d'énuméré sont aussi traduisibles. Elle se retrouve à l'aide du nom de la famille, du nom de l'attribut puis de la clef de l'énumeré.

3.12.3.7 Recherche des chaines à traduire (Génération fichier .po)

Pour rechercher les chaines à traduire dans les différents fichiers, il faut utiliser `make po`. Cela va mettre à jour les fichiers <APP>_fr.po et <APP>en.po.

3.12.3.8 Comment ça marche

Voici le code permettant de récupérer les données de tous nos fichiers et de générer un fichier .pot (Ce code est inclus dans votre fichier PubRule) :

```
xmlfiles=`find . -name "*.xml" -o -name "*.js"`;
grep TEXT: $xmlfiles | sed -e's/\[TEXT:/\n\[TEXT:/g' | sed
```

```
-e's/\(.*\)TEXT:\([^]]*\)\(.*\)/N_("\2")/g' > textxml.php

phpfiles=`find . -name "*.app" -o -name "*.php"`;
xgettext  --no-location   $phpfiles --keyword='_' --keyword='N_'
--keyword='text'
--keyword='Text' --language=c -o $appname.pot
```

Ensuite, il faut à partir de ce fichier .pot générer un fichier .mo pour chaque langue à définir. Voici un extrait du code :

```
for lang in  $LANGS; do
 destpo=$appname"_$lang.po"
 msgmerge -v --force-po  $destpo $appname.pot > $destpo.new;
 mv $destpo.new $destpo;
 msgfmt $destpo -o $wpub/locale/$lang/LC_MESSAGES/$appname.mo
done
```

3.12.4. Traduire un module dynacase dans une autre langue

Pour traduire un module dynacase dans une autre langue, on fera un module qui ne fournira que le fichier de traduction

Exemple pour la traduction de `dynacase-core' en espagnol :

On initialisera un module `dynacase-core-langpack-es'

```
$ mkdir dynacase-core-langpack-es
$ cd dynacase-core-langpack-es

$ vi VERSION
1.0.0

$ vi RELEASE
1

$ vi info.xml.in
<?xml version="1.0"?>
<module name="dynacase-core-langpack-es" version="@VERSION@" release="@RELEASE@"
basecomponent="no">

  <description lang="en">dynacase-core spanish (es_ES) locale</description>

  <

  <post-install>
    <process command="programs/update_catalog"><label lang="en">Generate
traduction catalog</label></process>
  </post-install>

  <post-upgrade>
    <process command="programs/update_catalog"><label lang="en">Generate
traduction catalog</label></process>
  </post-upgrade>

  <changelog>
    <version number="1.0.1-1" date="2010-05-17">
      <change title="Initial release" />
    </version>
  </changelog>

</module>
```

```
$ vi Makefile
PACKAGE = @PACKAGE@
VERSION = @VERSION@
utildir=@PUBRULE@
appname = @APPNAME@
pubdir = @prefix@
srcdir = @srcdir@
rootprefix=$(RPM_BUILD_ROOT)

export pubdir utildir appname pidir

TAR = gtar
GZIP_ENV = --best

export targetdir
export utildir

pages_not_xml = info.xml
pages_not_php = fam2po.php po2js.php lang_es.php

LANGS=es

include $(utildir)/PubRule


$ vi configure.in
AC_PREREQ(2.13)
AC_INIT(./Makefile.in)
AC_SUBST(VERSION)
VERSION=`cat VERSION`
AC_SUBST(RELEASE)
RELEASE=`cat RELEASE`
AC_SUBST(PACKAGE)
PACKAGE=dynacase-core-langpack-es
AC_SUBST(APPNAME)
APPNAME=WHAT

ac_default_prefix=/usr/share/what
AC_SUBST(PUBRULE)
PUBRULE=
AC_ARG_WITH(pubrule, [  --with-pubrule=dir      Path to PubRule],
PUBRULE=$withval)
if test "x$PUBRULE" != "x"; then
  PUBRULEDIR=$PUBRULE
else
  if test "x$PUBRULEDIR" == "x"; then
     AC_CHECK_FILE($HOME/anakeen/devtools/PubRule,
PUBRULEDIR=$HOME/anakeen/devtools/)
     if test "x$PUBRULEDIR" = "x"; then
          PUBRULEDIR=.
     fi
  fi
fi
AC_CHECK_FILE($PUBRULEDIR/PubRule, PUBRULE=$PUBRULEDIR)
if test "x$PUBRULE" = "x"; then
  AC_MSG_ERROR([Could not find PubRule])
fi
AC_MSG_NOTICE([PubRule located at $PUBRULE])

AC_OUTPUT(Makefile info.xml)
```

```
$ vi WHAT_es.po
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR Free Software Foundation, Inc.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Project-Id-Version: TEST locale\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2010-03-12 16:27+0100\n"
"PO-Revision-Date: 2010-03-12 16:55+0100\n"
"Last-Translator: Eric Martin <eric.martin@somewhere.com>\n"
"Language-Team: spanish-anakeen <distrib@anakeen.com>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8-bit\n"
"\n"

msgid "userlogin
msgstr "Inicio"

msgid "username"
msgstr "Nombre"

msgid "permissions"
msgstr "derecho"

msgid "none"
msgstr "Ninguno"

msgid "user"
msgstr "usario"

msgid "object"
msgstr "objeto"
```

[...]

```
$ vi lang_es.php
<?php

$lang["es_ES"] = array(
    "label"  => "Español",
    "flag"   => "",
    "locale" => "es",
    "dateFormat" => "%m/%d/%Y",
    "dateTimeFormat" => "%m/%d/%Y %H:%M",
    "timeFormat" => "%H:%M:%S",
);

?>
```

3.12.5. Paramétrage de locale installées

La définition des locales se fait à l'aide du fichier lang.php qui est situé dans le répertoire de la langue (locale/fr ou locale/en). Vous pouvez modifier les droits en ajoutant un fichier local-lang.php pour modifier la configuration par défaut. Si vous modifié directement le fichier lang.php vous perdrez vos modifications lors d'une prochaine mise à jour.

Fichier locale/en/lang.php :

```
<?php
/**
 * @author Anakeen
 * @license http://www.fsf.org/licensing/licenses/agpl-3.0.html GNU Affero
General Public License
 */
$lang["en_US"] = array(
        "label"  => "English",
        "locale" => "en",
        "dateFormat" => "%m/%d/%Y",
        "dateTimeFormat" => "%m/%d/%Y %H:%M",
        "timeFormat" => "%H:%M:%S",
);

/*
 ** Include local/override config
 ** -----------------------------
 */
$local_lang = dirname(__FILE__).DIRECTORY_SEPARATOR.'local-lang.php';
if( file_exists($local_lang) ) {
  include($local_lang);
}
?>
```

Exemple de fichier locale/en/local-lang.php :

```
<?php
$lang["en_US"]["dateFormat"] = "%d/%m/%Y";
$lang["en_US"]["dateTimeFormat"]="%d/%m/%Y %H:%M";
?>
```

Cette exemple montre comment on modifie le format de représentation des date (format court et format long).

3.12.6. Outils pour lire les fichiers .po et traduire les chaines

Une fois les fichier .po généré, il faut traduire les chaines de caractère dans les différentes langues.

Vous pouvez le faire un avec un simple éditeur de texte, mais il est fortement recommandé d'utiliser un outil spécialisé :

- kbabel (KDE)
- gtranslator (Gnome)
- PoEdit (multiplateforme)
- emacs (geek 😀)

3.12.7. Génération des fichiers .mo (binaires)

La commande 'msgfmt' permet de compiler les fichiers .po en fichier binaire .mo :

```
msgfmt $destpo -o $wpub/locale/$lang/LC_MESSAGES/$appname.mo
```

3.12.8. Initialisation de dynacase et d'Apache pour prendre en compte les localisations

Pour finir, il faut assembler les fichiers .mo des différentes applications et les rendre disponible pour Apache :

```
whattext
etc/init.d/apache2 reload
```

3.12.9. Script pour automatiser les tâches

Si vous avez suivis la formation développeur proposé par Anakeen, vous aurez appris à créer vos propres paquets .rpm ou .deb pour fabriquer vos applications basées sur **dynacase**. Et la partie localisation est incluse dans la génération du paquet.

Le but de ce document n'est pas de vous expliquer comment fabriquer un paquet mais simplement vous permettre de localiser vos applications.

FIXME internalmedia: freedom_2.14:devel:make_app.sh.txt reprend le code évoqué précédemment et y apporte quelques améliorations pour vous permettre de localiser une application.

Pour utiliser ce script, il faut le placer dans ”/usr/share/what” et l'exécuter en donnant comme argument le nom de l'application traiter :

```
/usr/share/what/make_app.sh MONAPPLI
```

Voici les fonctionnalités de ce script :

- Création des liens vers les Method, Class et Images
- Génération des fichiers .mo
- Compilation des .mo en .po
- Intégration des traductions dans Apache avec whattext

3.13 Manipulation des comptes utilisateurs

Ce chapitre a pour but de montrer par l'utilisation des méthodes et fonctions les plus courantes pour manipuler les utilisateurs "classique".

Ce chapitre ne décrit pas la manipulation des utilisateurs réseaux disponibles avec le module "networkuser".

3.13.1. Création d'un utilisateur

La création d'un utilisateur peut être fait à l'aide de la famille "IUSER".

Les comptes utilisateurs sont gérés par la classe Account. La famille "IUSER" permet de faire le lien entre le compte "système" (classe Account) et le document (famille IUSER).

Un utilisateur est identifié par un numéro unique. Ce numéro est renseigné par l'attribut "us_whatid" du document IUSER. Il correspond à l'attribut "id" de la classe Account.

Code minimaliste pour créer un utilisateur via les documents.

```
  include_once("FDL/Class.Doc.php");
  $du=createDoc("","IUSER");
  if ($du) {
    $du->setValue("us_login","jean.martin");
    $du->setValue("us_lname","martin");
    $du->setValue("us_fname","jean");
    $du->setValue("us_passwd1","secret");
    $du->setValue("us_passwd2","secret"); // nécessaire de doubler à cause des
contraintes
    $err=$du->store();
    if (!$err) {
      print "nouvel utilisateur n°".$du->getValue("us_whatid"); // affichage de
l'identifiant système
    } else {
      print "\nerreur:$err";
    }
  }
```

3.13.2. Correspondance entre compte User et document IUSER:

Pour récupérer le compte système à partir du document IUSER :

```
$user=$userDoc->getAccount();
if ($user->id != $doc->getValue("us_whatid") ) {
    // ce n'est pas normal
}
```

Si vous ne disposez que de l'identifiant documentaire vous pouvez aussi utiliser la méthode setFid de la classe Account:

```
$u=new Account();
$u->setFid($docUserId);
if ($u->isAffected()) {
   print $u->login;
}
```

A l'inverse pour récupérer l'objet documentaire à partir du compte système, vous pouvez utiliser l'attribut "fid".

```
$u=new Account();
$u->setLoginName("john.doe");
if ($u->isAffected()) {
  $doc=new_doc("", $u->fid);
}
```

Correspondance Compte User <=> Document IUSER

Compte User	Document IUSER
id	us_whatid
login	us_login
mail	us_mail
firstname	us_fname
lastname	us_lname
fid	id

Exemple de création d'utilisateur via la classe Account

```
$u=new Account();
$u->login='jean.dupond';
$u->firstname='Jean';
$u->lastname='Dupond';
$u->password_new='secret';
$err=$u->add();
if ($err) {
  error_log($err);
} else {
  print "Nouvel utilisateur n°".$u->id;
}
```

3.13.3. Affectation d'utilisateurs dans un groupe

La famille IGROUP permet de rajouter des utilisateurs dans des groupes.

La famille IGROUP comme la famille IUSER utilise la classe User pour identifier les groupes "système".

Comme la famille IUSER, la famille IGROUP dispose de la méthode getAccount pour récupérer l'objet Account

correspondant

Correspondance Compte User <=> Document IGROUP

Compte Group	Document IGROUP
id	us_whatid
login	us_login
lastname	grp_name
fid	id

Le compte "Account" d'un groupe a toujours son attribut "accounttype" à "G".

Utilisation de _IGROUP::AddFile()

Ajout de l'utilisateur n°1009 dans le groupe GDEFAULT :

```
include_once("FDL/Class.Doc.php");

$dbaccess=getDbAccess();

$g=new_Doc($dbaccess,"GDEFAULT");
$u=new_Doc($dbaccess,1009); // 1009 est la référence documentaire de
l'utilisateur

printf("ajout de l'utilisateur %s [%d] au groupe %s [%d]\n",
       $u->getTitle(),$u->id,$g->getTitle(),$g->id);

printf("liste des groupes avant\n");
print_r($u->getTValue("us_idgroup"));

$err=$g->addFile($u->initid);
print "Error:$err\n";

printf("liste des groupes apres\n");
print_r($u->getTValue("us_idgroup"));
```

3.13.4. Suppression d'utilisateur dans un groupe

Utilisation de _IGROUP::DelFile()

```
include_once("FDL/Class.Doc.php");

$dbaccess=getDbAccess();

$g=new_Doc($dbaccess,"GDEFAULT");
$u=new_Doc($dbaccess,1009);

printf("suppression de l'utilisateur %s [%d] du groupe %s [%d]\n",
       $u->getTitle(),$u->id,$g->getTitle(),$g->id);

printf("liste des groupes avant\n");
print_r($u->getTValue("us_idgroup"));

$err=$g->delFile($u->initid);
print "Error:$err\n";
```

```
printf("liste des groupes apres\n");
print_r($u->getTValue("us_idgroup"));
```

3.13.5. Récupération des membres d'un groupe

À partir de l'objet "Account" d'un groupe :Account::getAllMembers()

Exemple d'utilisation :

```
  $docGroup=new_Doc("", "GDEFAULT");
  $group=$docGroup->getAccount();
  $members=$group->getAllMembers();
  $userDocIdList=array();
  foreach ($members as $user) {
    printf("%s) %s\n", $user["id"],$user["login"]);
    $userDocIdList[]=$user["fid"];
  }
  print "---\n";
  // recherche des documents IUSER correspondants
  $dl=new DocumentList();
  $dl->addDocumentIdentificators($userDocIdList);
  foreach ($dl as $docid=>$docIUser) {
    printf("%s)%s\n", $docIUser->id,  $docIUser->getTitle());
  }
```

3.13.6. Récupération des utilisateurs associés à un rôle

Comme pour les groupes, à partir de l'objet "Account" d'un rôle : Account::getAllMember()

Exemple d'utilisation :

```
$docRole=new_Doc("", "TST_ROLE");
$members=$docRole->getAccount()->getAllMembers();
$userDocIdList=array();
foreach ($members as $user) {
  printf("%s) %s\n", $user["id"],$user["login"]);
  $userDocIdList[]=$user["fid"];
}
print "---\n";
// recherche des documents IUSER correspondants
$dl=new DocumentList();
$dl->addDocumentIdentificators($userDocIdList);
foreach ($dl as $docid=>$docIUser) {
  printf("%s)%s\n", $docIUser->id,  $docIUser->getTitle());
}
```

3.13.7. Récupération des rôles d'un utilisateur

À partir de l'objet "Account" , méthode getAllRoles

Exemple d'utilisation :

```
$u=new Account('');
$u->setLoginName('john.doe');
if ($u->isAffected()) {
  $roles=$u->getAllRoles();
  foreach ($roles as $aRole) {
```

```
    printf("%d)%s\n", $aRole["id"],$aRole["login"]);
  }
}
```

3.13.8. Suppléants et titulaires

Affectation d'un suppléant à un utilisateur et récupérations des titulaires.

```
$u=new Account();
$u->setLoginName("j1");
if ($u->isAffected()) {
  $err=$u->setSubstitute("j2"); // J1 est le titulaire de J2
                                //(J2 est suppléent de J1)
}
$u->setLoginName("j3");
if ($u->isAffected()) {
  $err=$u->setSubstitute("j2");// J3 est le titulaire de J2
}
$u->setLoginName("j2");
$incumbents=$u->getIncumbents(); // J2 a comme titulaire J1 et J3
foreach ($incumbents as $k=> $aIncumbent) {
  printf("%d)%s\n", $k,Account::getDisplayName($aIncumbent));
}

www-data@luke:~$ ./wsh.php --api=testSubstitute
0)j1 Doe
1)j3 Doe
```

3.14 Manipulation des classes DbObj

Ce chapitre a pour but de montrer par l'utilisation des méthodes et fonctions les plus courantes pour manipuler les objets provenant des classes DbObj (Database Object) telles que DocHisto, DocLog, User.

3.14.1. Récupération d'un objet

Les objets sont identifiés par les champs référencés par l'attribut "id_fields". Le constructeur attend des valeurs référencé par cet attribut.

Si cet attribut est un tableau de plusieurs champs alors il faut indiquer l'ensemble des champs référencés. Les références ne sont pas toujours des clefs uniques pour les objets de la classe.

```
// premier histo du document n°1004
  $histo=new DocHisto("",1004);
// premier log du document n°1004
  $log=new DocLog("",10);
// permissions du document 1004 pour l'utilisateur 14 (identifiant système)
  $perm=new DocPerm("",array(1004,14));
// caractéristiques de l'utilisateur n°1 (admin).
  $user=new Account("",1);
```

Pour savoir si l'objet a été récupéré depuis la base de données (si la référence existe), il faut utiliser la méthode ::isAffected().

```
 $user=new Account("",1);
 if ($user->isAffected()) {
    print_r($user->getValues()); // on affiche toutes les valeurs de l'objet
    print $user->login; // on affiche un des attributs de l'objet
 }
```

L'accès à un des attributs de l'objet se fait directement en utilisant le nom de l'attribut.

3.14.2. Recherche d'objets

La recherche d'objets est réalisée par la classe QueryDb. Elle permet d'effectuer des requêtes sur un type d'objet.

Le constructeur attend le nom de la classe en deuxième paramètre du constructeur. Le premier paramètre optionnel doit contenir les coordonnées de la base de données.

```
$query=new QueryDb("", "DocHisto");
$query->addQuery("id=1004");
$result=$q->query(0,0,"ITER");

foreach ($result as $histo) {
  print_r($histo->getValues());
}
```

L'exemple ci-dessus donne tous les objets DocHisto du document 1004. Ceci est l'équivalent de ce qui est fait avec la méthode doc::getHisto().

La méthode "QueryDb::addQuery($filter)" permet d'ajouter un filtre sql à la requête.

La méthode "QueryDb:query($start, $slice, $mode)" lance la requête et retourne les résultats dans la fenêtre précisé par $start et $slice. $start est l'offset de départ et $slice le nombre maximum de résultat à retourner.

$mode[35] peut avoir les valeurs "TABLE" ou "ITER". "ITER" retourne un objet itérateur qui permet d'avoir accès aux objets. "TABLE" retourne un tableau de valeurs indexées par leur nom.

```
$result=$query->query(0,0,"TABLE");
foreach ($result as $histo) {
     print_r($histo);
}
```

Résultat :

```
Array
(
    [id] => 1004
    [initid] => 1004
    [uid] => 1
    [uname] => Default Master
    [date] => 14/12/2010 14:49:47
    [level] => 2
    [code] =>
    [comment] => Mise à jour par importation
)
Array
(
    [id] => 1004
    [initid] => 1004
    [uid] => 1
    [uname] => Default Master
    [date] => 14/12/2010 14:49:46
    [level] => 2
    [code] => CREATE
    [comment] => Création du document
)
```

3.14.3. Transactions et savePoint

Des transactions et des points de sauvegarde de base de donnés peuvent être utilisés afin de confirmer ou de revenir à un point de sauvegarde. Ces transactions n'ont aucun effet sur l'objet mémoire manipulé.

35) *les valeurs LIST, LISTC et ITEM sont dépréciés*

```
include_once("FDL/Class.Doc.php");
$d=new_doc("",8160);
$d->savePoint("One");
$d->setValue("us_lname","Test one");
$d->store();  // modification en base doc n°8160 à comme nom "Test One"

$d->savePoint("Two");
$d->setValue("us_lname","Test two");
$d->store();// modification en base doc n°8160 à comme nom "Test Two"
$d->rollbackPoint("Two"); // annulation de toutes les requêtes depuis le point
"Two"

$d->commitPoint("One"); // acquittement des requêtes depuis le point "One"
// le document en BD a maintenant comme nom "Test One"
// ATTENTION : le document conserve en mémoire sa dernière valeur "Test two".
```

Les points de sauvegarde ne sont pas liés à un objet mais impactent toutes les requêtes liées à la base.

Si vous avez déclaré un savePoint il faut **obligatoirement** avoir un commitPoint ou un rollbackPoint de ce point. Dans le cas contraire toutes les modifications sur la base de données, depuis le premier point de sauvegarde, seront abandonnées. Les méthodes de point de sauvegarde sont accessibles depuis tout objet dbObj, pas seulement Doc.

Néanmoins il n'est pas obligatoire d'appliquer un commit ou un rollback sur tous les points, mais il est obligatoire de le faire au moins sur le premier.

```
$d->savePoint("One");
$d->setValue("us_lname","Test one");
$d->store();

$d->savePoint("Two");
$d->setValue("us_lname","Test two");
$d->store();

$d->savePoint("Three");
$d->setValue("us_lname","Test three");
$d->store();
$d->rollbackPoint("Two"); // on retourne au point Two, le point Three est effacé

$d->commitPoint("One");
le document a maintenant comme nom "Test One"
```

3.15 Bases de données PostgreSQL

3.15.1. Tables de la base anakeen

Les tables de la base 'anakeen' sont utilisées par le coeur de dynacase et en particulier tout ce qui concerne l'authentification des utilisateurs.

3.15.1.1 acl

Contenu des acl des fichiers .app :

- id : id de l'acl
- id_application : id de l'application (lien avec la table 'application')
- name : nom de l'acl tel qu'il a été indiqué dans le fichier .app
- grant_level :
- description : Description de l'application indiqué dans le fichier .app
- group_default : Si=Y, le groupe utilisateur sera affecté à l'ACL.

3.15.1.2 action

Liste des actions (Contenu des actions des fichiers .app)

- id : id de l'action
- id_application : id de l'application (Lien avec la table 'application')
- name : Nom de l'action indiqué dans le fichier .app
- short_name : Nom court de l'action indiqué dans le fichier .app
- long_name : Nom long
- script : Fichier .php contenant l'action
- function :
- layout : Fichier .xml (layout) de l'action
- available : Y = l'action est disponible
- acl : acl permettant d'utiliser l'action
- grant_level :
- root :

- icon :
- toc :
- father :
- toc_order :

3.15.1.3 application

Liste des applications (Contenu des fichiers .app)

- id : id de l'application
- name : nom
- short_name : nom court
- description : description
- access_free : Accès libre (pas d'acl)
- available : Y = Application disponible
- icon : Icône de l'application
- displayable : Y = Application visible dans le menu général par les utilisateurs. Certaines applications sont invisibles et servent seulement à fournir des actions à d'autres applications (ex : GENERIC)
- with_frame : N'est plus utilisé
- childof : Indique de quel parent hérite l'application. Il est par exemple courant de créer une application basée sur ONEFAM.
- objectclass :
- ssl :
- machine : Peut-être utilisé pour répartir des applications sur plusieurs serveurs
- iorder :

3.15.1.4 docfrom

3.15.1.5 docname

3.15.1.6 domain

3.15.1.7 groups

Liens entre les id des utilisateurs et les id des groupes. Cette table est répliqué dans la base dynacase à chaque modification. Si la table n'est pas synchronisé cela peu poser des problèmes.

- iduser : id de l'utilisateur (lien avec un utilisateur de la table 'users')
- idgroup : id du groupe (lien avec un groupe de la table 'users')

3.15.1.8 mailaccount

3.15.1.9 paramdef

Table de définition des paramètres

- name : Nom du paramètre
- isuser : Y = Paramètre utilisateur
- isstyle : Y = Paramètre utilisé dans les styles
- isglob : Y = Paramètre global
- appid :
- descr : Description du paramètre
- kind : Format du paramètre

3.15.1.10 paramv

Table contenant les valeurs des paramètres.

- name :
- type : A=Applicatif, G=Global, U1 = Paramètre de l'utilisateur N°1
- appid :
- val : Valeur du paramètre

Remarque : Tous ces paramètres sont accessibles dans les Layout.

3.15.1.11 permission

Permissions sur les actions

- id_user : id de l'utilisateur
- id_application : id de l'application
- id_acl : Valeur de de l'acl sous forme binaire

3.15.1.12 session_conf

3.15.1.13 sessions

Sessions des personnes connecté dans la journée.

- id : Cookie enregistré
- userid : id de l'utilisateur

3.15.1.14 style

Liste des thèmes installés.

- name : Nom du thème
- Description du thème

3.15.1.15 users

Utilisateurs et groupes. Table servant à l'authentification.

- id : id de l'utilisateur ou du groupe
- iddomain :
- lastname : Nom de l'utilisateur ou du groupe
- firstname : prénom
- login : login
- password : mot de passe
- isgroup : Y = C'est un groupe
- expires :
- passdelay :
- status :
- mail : mail de l'utilisateur
- ntpasswordhash :
- lmpasswordhash :
- fid : lien vers l'id du document dynacase

3.15.2. Tables de la base dynacase

3.15.2.1 Héritage entre les tables

dynacase utilise les possibilités d'héritage entre les tables de Postgresql. Exemples :

- La table « doc », contient (virtuellement) tout les documents créés dans dynacase. Cette table contient toutes les propriétés par défaut d'un document (Profil, Cycle,...)
- La table « doc1 » contient tous les documents de la famille « de base ». Elle hérite de la table « doc ».
- La table « doc2 » contient tous les dossiers. Elle hérite de la famille « de base », donc de la table « doc1 » comme la plupart des familles standards de dynacase

Si une famille A hérite d'une famille B, chaque enregistrement sera disponible dans 3 tables :

- doc,
- docA
- docB

La suppression ou la modification d'un enregistrement dans l'une des tables le supprimera dans les 3.

Remarque : pgAdmin permet de consulter ce mécanisme de dépendance entre les tables.

3.15.2.2 doc

Cette table ne contient physiquement aucune donnée mais contient virtuellement grâce au mécanisme d'héritage des tables de Postgresql toutes les propriétés de tous les documents dynacase.

- id, owner, title, revision, initid, fromid, doctype, locked, allocated, icon, lmodify, profid, usefor, revdate, version, cdate, adate, comment, classname, state, wid, values, attrids, postitid, cvid, name, dprofid, prelid, atags, confidential, ldapdn, svalues, fulltext, forumid

3.15.2.3 doc<idfam>

Chaque famille de dynacase est stockée dans une table séparée 'doc<idfam>' ou '<idfam>' correspond à l'id de la famille.

De plus, si une famille hérite d'une autre famille, le mécanisme d'héritage des tables de Postgresql sera utilisé.

3.15.2.4 docattr

Cette table contient la définition des attributs de chaque famille. Cette table est lue uniquement par par fdl_adoc.

3.15.2.5 docattrldap

Cette table permet de faire le lien entre les attributs d'une famille dynacase et les attributs d'un annuaire ldap. Cette table est utilisée en particulier pour le fonctionnement de la famille 'Utilisateur réseau'

3.15.2.6 docfam

Cette table contient la liste des familles.

3.15.2.7 docfrom

Cette table permet de retrouver à quelle famille (fromid) appartient un document (id). Cette table est construite construite automatiquement avec des triggers pour améliorer les performances des requêtes.

- id
- fromid

3.15.2.8 dochisto

Cette table contient l'historique des modifications de tous les documents. Cette table est utilisée en particulier pour afficher le contenu du lien 'historique' d'un document dynacase.

3.15.2.9 docname

Cette table permet de faire le lien entre le nom logique d'un document et son id.

- name : Nom logique du document
- id : id du document
- fromid :

En cas de problème avec cette table, dynacase ne fonctionnera plus correctement mais il est possible de la reconstruire complètement en cas de besoins.

3.15.2.10 docperm

Table des permissions.

Remarques :

- Les points verts sont visibles dans la table.
- Les points gris sont calculés.
- Pour chaque point jaune un group virtuel est créé dans la table vgroup.
- Le userid 100000 est un utilisateur virtuel utilisé pour calculer les droits sur les profils dynamique.
- Le droit est calculé par la fonction postgresql « getuperm » qui indique si l'utilisateur à le droit de voir le document. Si le droit n'est pas encore calculé, il va le faire et remplir les boules grises. Les boules grises sont calculés par « getuperm » au moment de l'accès au document et ajouté dans la table docperm. Une fois la boule grise ajouté, elle n'est plus recalculée. La réinitilaisation est faite également à chaque changement de groupe.

3.15.2.11 docread

Cette table gérée automatiquement par les trigers de Postgresql contient le contenu complet de tous les documents de dynacase (svalue) y compris le contenu des pièces jointes si le moteur de transformation est utilisé (fulltext).

- fulltext : Recherche avec * dans ONEFAM
- svalues : Recherche avec ~ dans ONEFAM

Son but est de pouvoir effectuer rapidement des recherches plein texte sur le contenu de tous les documents. Elle est

utilisée dans les recherches ou aucune famille n'est précisée. Si une famille est précisée, la recherche sera effectuée dans la table de la famille.

3.15.2.12 docrel

Table alimentée par des triggers utilisée pour connaître les relations entre les documents. Cette table est utilisée en particulier par le menu 'Autre / Relations du document' des documents de dynacase.

- sinitid : Source
- cinitid : Cible
- type : folder = Relation de type folder

3.15.2.13 docutag

Table contenant les tags des documents par utilisateur.

Sur chaque document, il est possible d'appliquer un ou plusieurs tags.

Exemples :

- VIEWED : tag ajouté automatiquement sur tous les documents. Il permet de connaitre la date, l'heure et la dernière personne ayant consulté le document.
- TOVIEW : Tag indiquant que la personne doit voir le document. Utilisé par Workspace (Mes documents à voir)

SetUTag : Méthode Permettant d'ajouter un TAG.

Cela peut-être utilisé pour mémoriser des variables sur le document utilisateur.

3.15.2.14 docvaultindex

Table de correspondance entre les id des documents et les numéros des fichiers. Cette table permet uniquement de rechercher les orphelins.

- docid
- vaultid

3.15.2.15 fld

Table permettant de retrouver les documents par dossier (Relations de type folder). Table faisant un peu redondance avec la table docrel mais uniquement pour les dossiers. Cette table contient également les recherches stockés

3.15.2.16 groups

Lien entre les id des utilisateurs et les id des groupes

Cette table est normalement la réplique exacte de la table 'groups' de la base 'anakeen'

Elle existe pour des raisons historique

3.15.2.17 pg_ts_cfg, pg_ts_cfgmap, pg_ts_dict et pg_ts_parser

Tables internes à tsearch de Postgresql utilisées pour la recherche plein texte

3.15.2.18 vgroup

3.15.3. Tables du vault (base dynacase)

3.15.3.1 vaultdiskdirstorage

Sous-dossier de chaque Vault

3.15.3.2 vaultdiskfsstorage

Liste des coffres

3.15.3.3 vaultdiskstorage

Table permettant de retrouver un fichier dans le Vault à partir de son numéro.

- id_file : id du fichier dans le Vault
- id_fs :
- id_dir :
- public_access :
- size : Taille du fichier
- name : Nom réel du fichier
- mime_t : Type mime (ex : Microsoft Office Document)
- mime_s : Type mime (ex : application/vnd.ms-excel)
- cdate :
- mdate :
- adate :
- teng_state : Utilisé par le moteur de transformation
- teng_lname : Utilisé par le moteur de transformation
- teng_id_file : Utilisé par le moteur de transformation
- teng_comment : Utilisé par le moteur de transformation

3.16 Méthodes et fonctions dépréciées

3.16.1. Recherches

La fonction getChildDoc est dépréciée (depuis la version 2.14) au vu des trop nombreux paramètres et doit être remplacée par l'usage de la classe SearchDoc.

3.16.2. Classe SearchDoc

Méthode dépréciée	Alternative	Depuis version
SearchDoc ::setDebugMode()	Plus nécessaire. Les informations de débuggage sont toujours disponibles sur appel de la méthode DocSearch ::getSearchInfo()	3.0.16
SearchDoc ::getDebugInfo()	Remplacé par DocSearch ::getSearchInfo()	3.0.16
SearchDoc ::setValueReturn()	Remplacé par DocSearch ::setObjectReturn(false)	3.0.16
SearchDoc ::dirid	Remplacé par DocSearch ::useCollection($dirid)	3.0.18
SearchDoc ::slice	Remplacé par DocSearch ::setSlice($slice)	3.0.18
SearchDoc ::start	Remplacé par DocSearch ::setStart($$start)	3.0.18
SearchDoc ::orderby	Remplacé par DocSearch ::setOrder($$order)	3.0.18

3.16.3. Classe Doc

Méthode dépréciée	Alternative	Depuis version
Doc ::getWhatUserId()	Remplacé par Doc::getSystemUserId()	3.0.0
Doc::getTitle() pour les templates HTML	Utilisation de Doc::getHTMLTitle(). Cas des titres avec des < > &.	3.0.16
Doc::canUpdateDoc	Doc::canEdit()	3.1.0

3.16.4. Class OooLayout

Méthode dépréciée	Alternative	Depuis version
OooLayout ::setBlockData ()	Utilisation de la méthode OooLayout::setRepeatable() et OooLayout::setColumn()	3.0.17

3.16.5. Class Action

Fonction dépréciée	Alternative	Depuis version
getHttpVars()	Remplacé par Action::getArgument() dans le cas des actions lors de récupération des valeurs des arguments.	3.0.0

3.16.6. Famille IUSER

Méthode dépréciée	Alternative	Depuis version
_IUSER::getOtherGroup()	Remplacé par _IUSER ::getUserGroups()	3.0.17
IUSER::getValue("us_idgroup")	Remplacé par _IUSER ::getUserGroups()	3.0.17

3.16.7. Cycle de vie

Usage déprécié	Alternative	Depuis version
Utilisation des define pour définir des noms d'états ou de transitions	Utilisation des constantes de classe	3.0.0
WDOC::getAction()	Remplacé par WDOC::getActivity()	3.0.0

3.16.8. fonctions Wsh

Usage déprécié	Alternative	Depuis version
freedom_import	Remplacé par importDocument	3.2.0
freedom_refresh	Remplacé par refreshDocuments	3.2.0

4 Exploitation

4.1 dbaccess.php

4.1.1. Fichier ''dbaccess.php'' et "local-dbaccess.php"

Le fichier `dbaccess.php' permet de paramétrer les bases de données utilisées et le mécanisme d'authentification utilisé.

Par défaut, ce fichier `dbaccess.php' est régénéré lors d'une mise à jour de dynacase-platform. Par conséquent, pour déclarer un paramétrage spécifique pour votre installation (utilisation d'un mécanisme d'authentification comme dynacase-networkuser par exemple) il faudra que vous créiez un fichier `local-dbaccess.php' dans lequel vous pourrez ajouter, ou écraser, les paramètres nécessaires.

4.1.2. Fichier `dbaccess.php'

```
$pgservice_core="ged";
$pgservice_freedom="ged";;
$freedom_context="default";
$dbpsql=$pgservice_core;

$freedom_authtype = 'html';

$freedom_authtypeparams = array(
    'html' => array (
        'cookie' => 'freedom_auth',
        'authurl' => 'guest.php?sole=A&app=AUTHENT&action=LOGINFORM',
        'username' => 'auth_user',
        'password' => 'auth_pass',
        ),
    'open' => array(),
    'basic' => array(
        'realm' => 'freedom',
        ),
    );

$freedom_authprovider = 'freedom';

$freedom_providers = array(
    'freedom' => array(
        'connection' => 'service='.$pgservice_core,
        ),
    'file' => array(
        'authfile' => '/var/www/ged/.freedompwd',
        ),
    );

@include_once('local-dbaccess.php');
```

4.1.3. Fichier `local-dbaccess.php'

Exemple de fichier `local-dbaccess.php' pour surcharger le paramétrage de `dbaccess.php' :

```
<?php
```

```
$freedom_authprovider = 'freedom,freedomNu';

$freedom_providers['freedomNu'] = array(
    'allowAutoFreedomUserCreation' => 'no',
    'fix_euro' => 'no',
    'convert_to_utf8' => 'no',
    'options' => array(
        LDAP_OPT_REFERRALS => 0
    )
);

?>
```

4.1.3.1 Variables principales

`$freedom_authtype`	Le mode de l'interface utilisateur d'authentification. Le paramètre peut prendre les valaurs : apache, `basic`, `html` ou open.
`$freedom_authprovider`	Ce paramètre spécifie le, ou les, backend d'authentification à utiliser pour valider les mot de passes. Par défaut dynacase est livré avec les backend suivants : `freedom`, `ldap` et `file`.

4.1.3.2 freedom_authtype

Ce paramètre permet de spécifier quelle interface d'authentification est utilisé pour demander les informations de connexion à l'utilisateur (login/password la plupart du temps) et lui transmettre le résultat de l'authentification.

4.1.3.2.1 apache

Ce mode spécifie que toute la mécanique d'authentification est délégué à Apache en mode HTTP Basic. dynacase ne s'occupe pas d'authentifier les utilisateurs, et Apache lui fournit les utilisateurs qui viennent de se connecter via la variable PHP `$_SERVER['PHP_AUTH_USER']`.

Dans ce mode, le paramètre `$freedom_authprovider` n'est pas utilisé puisque c'est Apache qui gère toute l'authentification.

4.1.3.2.2 basic

Le mode `basic` permet à dynacase de fournir un mécanisme d'authentification HTTP Basic.

Dans ce mode, dynacase gère l'authentification au format HTTP Basic et l'utilisateur rentre son login et son mot de passe dans la boite de dialogue affiché par le navigateur.

Paramètres :

`realm`	Le realm de l'authentification HTTP Basic.

4.1.3.2.3 html

Le mode `html` présente un formulaire HTML pour demander le login et le mot de passe de l'utilisateur. Une fois le login et le mot de passe validé auprès du `freedom_authprovider`, une session par cookie est ouverte afin de valider les accès suivants.

Paramètres :

`cookie`	Le nom du cookie contenant la clef associé à la session de l'utilisateur.
`authurl`	L'URI d'accès au formulaire de login.
`username`	L'id du champ input HTML contenant le nom d'utilisateur.

`password`	L'id du champ input HTML contenant le mot de passe de l'utilisateur.

4.1.3.2.4 open

Le mode open ne présente aucune interface pour la saisie du login et du mot de passe, mais se base sur un jeton présent dans l'URI.

Dans ce mode, le paramètre `$freedom_authprovider` n'est pas utilisé.

Le jeton est contenu dans la variable `privateid'.

Exemple d'URI avec authentification `open' et jeton :

http://<server>/freedom/index.php?
authtype=open&privateid=e2bb65612c70e7ac78d5ccbfe12aa234&app=FOO&action=BAR

L'action `BAR' de l'application `FOO' doit être déclarée `openaccess=Y' pour pouvoir être accessible avec cette authentification par jetons.

Le jeton doit êtres présent dans la table `usertoken' qui stocke les jetons, avec l'utilisateur associé au jeton et une date d'expiration du jeton.

Voir Class.UserToken.php.

4.1.3.3 freedom_authprovider

Le paramètre `$freedom_authprovider` permet de spécifier le sous-mécanisme utilisé pour la validation du login et du mot de passe.

Ces providers prennent en entrée un login et un mot de passe, et retourne s'ils sont valide ou non.

Chaque provider contient un ensemble de paramètres de configuration qui lui sont propre.

4.1.3.3.1 freedom

Le provider `freedom` est le mécanisme historique de dynacase qui utilise la base Postgres et la table `user` pour la validation des login et mots de passe.

On parlera dans ce cas d'authentification « interne », puisqu'elle ne se base sur aucun élément externe à dynacase.

Paramètres :

`connection`	Contient la chaîne de connexion à la base de donnée Postgresql contenant la table `user`. Exemple : `service=freedom`

4.1.3.3.2 ldap

Le provider `ldap` permet de valider le login et le mot de passe auprès d'un serveur LDAP, ou d'un serveur Active Directory, en effectuant un « bind LDAP » auprès de celui-ci.

Ce provider supporte une création basique d'utilisateurs « à la volé ». Dans ce cas, si l'authentification du est validé auprès du LDAP mais qu'il n'y a pas de compte utilisateur dynacase correspondant a ce login, alors le provider peut créer un compte utilisateur dynacase (IUSER) avec ce login. Cela permet de pouvoir « peupler » dynamiquement la base des comptes utilisateurs dynacase au fur et à mesure que les personnes se connectent. Le paramètre `dGroup' permet de spécifier le nom logique, ou l'id, du groupe dynacase (IGROUP) dans lequel seront créé ces nouveaux utilisateurs dynacase.

Paramètres :

`host`	Le nom, ou l'adresse IP, du serveur LDAP.
`port`	Le port du serveur LDAP.
`ssl`	y pour utiliser une connexion SSL au serveur LDAP, n pour ne pas utiliser de connexion SSL.
`options`	Options de la librairie OpenLDAP pour la connexion au serveur LDAP.

`dn`	Le DN de connexion utilisé pour valider le mot de passe de l'utilisateur. La chaine `%s` est remplacé par le login, et le mot de passe fournit par l'utilisateur est utilisé pour effectué un bind avec le DN ainsi construit.
`dGroup`	Le nom logique du groupe dynacase sous lequel seront créés les utilisateurs. Voir
`allowAutoFreedomUserCreation`	yes pour activer la création des comptes utilisateurs par le provider, no dans le cas contraire (par défaut no).

4.1.3.3.3 file

Le provider `file` permet de valider un couple de login/mot de passe auprès d'un fichier à plat au format htpasswd Basic d'Apache.

Ce provider ne supporte pas la création d'utilisateurs « à la volé ».

Paramètres :

`authfile`	Le chemin d'accès au fichier à plat de mot de passe au format htpasswd.

4.1.3.3.4 freedomNu

Le provider `freedomNu` est fourni par l'application `freedom-networkuser', et permet de valider le login et le mot de passe auprès d'un serveur LDAP ou d'un serveur Active Directory.

Ce provider supporte la création d'utilisateurs « à la volé ». Dans ce cas, si l'authentification est validé auprès du LDAP mais qu'il n'y a pas de compte utilisateur dynacase correspondant a ce login, alors le provider peut créer un compte utilisateur dynacase (LDAPUSER) avec ce login. Cela permet de pouvoir « peupler » dynamiquement la base des comptes utilisateurs dynacase au fur et à mesure que les personnes se connectent.

Paramètres : Voir documentation paramètres de networkuser

Les paramètres de connexion au LDAP/AD sont pris dans les paramètres applicatifs de l'application dynacase `NU' (voir documentation configuration Freedom de networkuser).

4.1.4. Pré-sélection du mode d'authentification

Dans certains cas il peut-être utile de pouvoir se connecter sur un même contexte dynacase via deux mécanisme d'authentification différents (freedom_authtype).

Par exemple dans le cas d'un contexte dynacase authentifié par CAS, il peut-être utile de pouvoir se connecter sur ce système sous le compte `admin' local (Master Default) sans devoir avoir un compte `admin' existant sur CAS.

Pour cela, on peut implémenter un « aiguillage » dans le fichier `local-dbaccess.php' afin de sélectionner le mode d'authentification en fonctions d'éléments de la requête (comme le champ `Host:' des requêtes HTTP).

4.1.4.1 Exemple

Fichier `local-dbaccess.php' :

```
<?php

if( $_SERVER['HTTP_HOST'] == 'admin-ged.example.net' ) {
  return;
}

$freedom_authtype = 'cas';
```

```
$freedom_authtypeparams['cas'] = array(
        'cookie' => 'freedom_auth',
        'cas_version' => 'CAS_VERSION_2_0',
        'cas_server' => 'localhost',
        'cas_port' => 8443,
        'cas_uri' => '/cas-server-webapp-3.4'
        );

$freedom_authprovider = 'freedomNu';

$freedom_providers['freedomNu'] = array(
        'allowAutoFreedomUserCreation' => 'yes',
        'fix_euro' => 'no',
        'convert_to_utf8' => 'no'
        );

>?
```

Dans cet exemple, si on se connecte sur l'URL `http://admin-ged.example.net', alors c'est le mécanisme d'authentification définit par défaut dans le `dbaccess.php' qui sera appliqué, et dans le cas contraire, on appliquera l'authentification CAS.

De cette manière, les utilisateurs se connectent par CAS sur leur URL habituelle (`ged.example.net' par ex.), et l'administrateur local peut se connecter sur l'hôte `admin-ged.example.net'.

4.2 Développer un provider

Un provider doit implémenter les méthodes définies dans la classe `WHAT/Class.Provider.php`

Les méthodes à implémenter obligatoirement sont :

- `abstract function validateCredential($username, $password)`
- `abstract function validateAuthorization($opt)`

Les méthodes optionnelles sont :

- `public function __construct($authprovider, $parms)`
- `public function initializeUser($username)`

4.2.1. validateCredential()

Cette méthode prend en entrée deux arguments qui sont :

- `$username` : le login entré par l'utilisateur
- `$password` : le mot de passe entré par l'utilisateur.

La méthode doit retourner :

- `true` si le couple login/mot de passe est correct
- `false` si le couple login/mot de passe est incorrect.

4.2.2. validateAuthorization()

Une fois le couple login/password est validé, cette méthode permet de contrôler si l'utilisateur est autorisé à se connecter.

Cette méthode prend entrée un argument :

- `$opt` : une structure contenant le nom de l'utilisateur.

```
$opt = array(
    'username' => $username
```

```
);
```

La méthode retourne :

- `true` si l'utilisateur est autorisé à se connecter
- `false` dans le cas contraire.

4.2.3. __construct()

C'est le constructeur du Provider que l'on peut étendre si celui-ci nécessite une initialisation particulière.

Cette méthode prend en entrée deux arguments qui sont :

- `$authprovider` :
- `$parms` :

4.2.4. initializeUser()

Si le compte de l'utilisateur n'existe pas dans dynacase, cette méthode est utilisé pour créer le compte de l'utilisateur dans dynacase.

Cette méthode prend en entrée le login de l'utilisateur :

- `$username` : le login de l'utilisateur.

La méthode retourne :

- ”” : une chaîne vide s'il n'y a pas eu d'erreur à la création du compte.
- “`Error message …`” : une chaîne non vide contenant le message d'erreur rencontré.

Cette méthode spécifique implémente la recherche des informations de l'utilisateur à partir du login sur le système d'authentification utilisé, et la création du compte utilisateur dynacase avec les informations obtenus.

4.2.5. Exemple

Dans l'exemple ci-dessous, nous allons écrire un provider pour valider les mots de passe des utilisateurs auprès d'un service PAM à l'aide de la commande `checkpassword-pam`. Ce module ne supportera pas la création d'utilisateurs à la volée.

4.2.5.1 Fichier `context/local-dbaccess.php'

On va déclarer dans le `local-dbaccess.php' que l'on utilise en premier notre provider `pam', et ensuite le provider `freedom' si l'utilisateur n'est pas reconnu par `pam'.

Notre provider `pam' aura un paramètre nommé ”`service`” qui contiendra le nom du service PAM auprès duquel seront validé les login et mot de passe (c'est le nom du fichier situé dans `/etc/pam.d').

```
<?php

$freedom_authprovider = 'pam,freedom';

$freedom_providers['pam'] = array(
    'service' => 'freedom'
    );

?>
```

4.2.5.2 Fichier `WHAT/providers/Class.pamProvider.php'

Notre provider est donc décrit dans le fichier `Class.pamProvider.php' situé dans le sous-répertoire `WHAT/providers' de l'installation dynacase.

Celui-ci fournit une classe `pamProvider' qui étend la classe `Provider', et nous allons décrire nos méthodes spécifiques `validateCredential()' et `validateAuthorization()'.

Dans `validateCredential() nous allons récupérer le paramètre `service' de notre provider, et utiliser la commande `checkpassword-pam' pour valider le username et le password reçu.

Pour simplifier l'exemple, la méthode `validateAuthorization()' retournera `true' systématiquement (on supposera que l'autorisation du compte est établi dans la phase de validation du mot de passe).

```
<?php

include_once('WHAT/Class.Provider.php');

Class pamProvider extends Provider {

  public function validateCredential($username, $password) {
    $service = 'freedom';
    if( array_key_exists('service', $this->parms) ) {
      $service = $this->parms{'service'};
    }

    return $this->checkpassword_pam($username, $password, $service);
  }

  public function validateAuthorization($opt) {
    $username = $opt{'username'};

    return true;
  }

  public function checkpassword_pam($username, $password, $service) {
    $cmd = sprintf("checkpassword-pam --service %s --no-chdir-home --noenv",
    escapeshellarg($service)
    );

    $proc = proc_open($cmd, array(3=>array('pipe', 'r')), $pipes);
    if( ! is_resource($proc) ) {
      error_log(__CLASS__."::".__FUNCTION__." ".
 sprintf("Error running checkpassword-pam")
 );
      return false;
    }

    fwrite($pipes[3], sprintf("%s\0%s\0timestamp\0",
       $username,
       $password
       ));
    fclose($pipes[3]);

    $ret = proc_close($proc);
    if( $ret === 0 ) {
      return true;
    }

    error_log(__CLASS__."::".__FUNCTION__." ".
      sprintf("Authentication failed for user '%s' and service '%s'.",
        $username,
        $service
      )
    );
    return false;
  }
```

```
}

?>
```

4.2.5.3 Fichier `/etc/pam.d/freedom'

Pour finir, le fichier associé au paramètre "`service`" de notre provider. Celui-ci contiendra les règles que l'on souhaite voir appliqué pour la validation des comptes. Dans cet exemple, on utilisera une authentification des comptes sur la base locale des utilisateurs Unix du serveur.

```
# auth       sufficient     pam_securityserver.so # uncomment for OS X
auth       sufficient     pam_unix.so
auth       required       pam_deny.so
account    required       pam_permit.so
session    required       pam_permit.so
```

4.3 Debuggage

4.3.1. Affichage des avertissements PHP

Fichier php.ini (souvent sous /etc/php.ini)

```
;   - Show all errors, except for notices and coding standards warnings
;
error_reporting = E_ALL & ~E_NOTICE
;
; Print out errors (as a part of the output).  For production web sites,
; you're strongly encouraged to turn this feature off, and use error logging
; instead (see below).  Keeping display_errors enabled on a production web site
; may reveal security information to end users, such as file paths on your Web
; server, your database schema or other information.
display_errors = On
```

Ensuite il faut redémarrer apache.

Ceci permet de voir les erreurs directement sur le navigateur. Sinon les log sous écrits dans fichier de log apache (souvent sous /var/log/httpd/error.log) .

4.3.2. dynacase en mode debug

Il est aussi possible d'activer la trace de nombreuses vérification en positionnant le paramètre CORE_LOGLEVEL du noyau freedom. Pour cela, il faut ce connecter `admin`, aller dans l'application `administration` puis dans `gestion des applications/paramètres applicatif/noyau/CORE_LOGLEVEL`. Il faut mettre la valeur `IWEFD` pour tout tracer. Le 'D' indique le mode debug. Ceci entraîne des performances moindre. Ne pas oublier d'enlever ce mode lorque le problème est résolu.

L'affichage de ces traces se trouve dans le même fichier de log que celui d'apache.

Les niveaux de trace :

I	Information	trace les accès aux actions
W	Avertissement	trace les avertissements - problèmes mineurs
E	Erreur	trace les erreurs - problèmes majeur pouvant dégrader le système
F	Fatal	trace les erreurs bloquantes provoquant l'arrêt du système

O	Fonctions dépréciées	trace les appels aux fonctions qui seront amenées à disparaître dans les prochaines versions
D	Debug	trace les messages pour le mode debugage (ne pas utiliser de manière continue car cela impacte les performances).

4.3.3. Tracer le temps d'exécution des actions dans dynacase

En pré-requis, il faut installer le paquet 'php5-xdebug'

Ensuite, il faut se connecter à l'adresse suivante : → http://localhost/dynacase/indexq.php

Cela permet d'afficher en haut à gauche de chaque fenêtre dynacase le temps d'exécution des actions :

Et en cliquant sur l'un des chiffres, cela affiche le détail des temps :

Détail des temps indiqués :

- init : tps inclusion des fichiers : PHP : include + authentification
- app : code PHP hors init : ce que fait l'action
- memory : mémoire utilisée
- server all : somme des temps

- navigator : tps d'affichage par le navigateur

Remarque : Un clic droit sur le chiffre permet de le faire disparaitre.

4.3.4. Firebug

Le plugin 'Firebug' disponible pour Firefox permet :

- de débuguer les applications javascript
- d'analyser ses pages Web
- de connaitre précisément le temps de chargements de chaque élément d'une page Web

Firebug est téléchargeable ici : https://addons.mozilla.org/fr/firefox/addon/1843

Voici un exemple de résultat :

4.4 Exécution périodique

La famille `Processus` (EXEC) permet de définir des tâche/traitements à exécuter de manière périodique ou récurrente.

Le traitement peut-être :

- le lancement d'une action d'une application donnée ;
- ou, le lancement d'un script d'API.

Il est possible ensuite de spécifier une répétition pour l'exécution périodique du processus. Dans ce cas, la date/horaire de la prochaine exécution sera planifiée à la date de fin d'exécution du processus précédent + la période d'exécution spécifié.

Le déclenchement manuel du processus, par l'utilisation du “Exécuter maintenant”, désactivera l'exécution périodique du processus. Pour ré-activer l'exécution périodique, il faudra alors éditer et sauver le processus.

Exemple de déclaration d'un processus pour l'import périodique d'un fichier CSV toutes les 2 heures :

processus
IMPORT NOUVEAUX PRODUITS
FREEDOM_IMPORT
| Sauver | Annuler

Identification
Exécutant : Master Default
Titre : Import nouveaux produits

Traitement
Application :
Action :
Api : freedom_import

paramètres

variable	valeur
file	/tmp/nouveaux_produits.csv

Dates
À exécuter le : 03/11/2009 16:34
Période en jours :
Période en heures : 2
Période en minutes :

La vérification pour les déclenchements des processus périodiques est faite toutes les 5 minutes par un processus CRON. La période de lancement doit être d'au moins de 5 minutes.

4.5 Paramétrage de dynacase

4.5.1. Les modules de dynacase

FREEDOM comprend plusieurs modules applicatifs de base :

- CORE - Noyau
- FDL - dynacase Library
- FREEDOM - Gestion documentaire
- GENERIC - Manipulation d'une famille
- ONEFAM - Interface d'utilisation de GENERIC
- VAULT - Coffre de stockage des fichiers

Chacune des applications comprennent des paramètres.

4.5.2. Accès au paramétrage des modules de dynacase

Le paramétrage est accessible via les menus :

- Administration/Gestion des applications

Les paramètres sont de plusieurs types :

- Type A : Paramètre local à l'application
- Type G : Paramètre global pouvant être utilisés par d'autres applications
- Type U : Paramètre pouvant être modifié par chaque utilisateur via le menu “ Mon compte”

Remarques :

- Les paramètres en gras indiques qu'ils sont globaux(Type = G)
- Pour connaître le nom interne du paramètre, il faut placer la souris au dessus de sa description

4.5.3. ACCESS - Accessibilités

Pas de paramétrage

4.5.4. AUTHENT - Authent

4.5.4.1 Paramètres pour la fenêtre de connexion

Nom du paramètre	Description	Type	Valeur par défaut
AUTHENT_SHOW_REQPASSWD	Voir le bouton "mot de passe oublié" permettant de demander un nouveau mot de passe sur l'interface de connexion.	A	yes
AUTHENT_SHOW_LANG_SELECTION	Voir le sélecteur de langue sur l'interface de connexion.	A	yes
AUTHENT_MAILASKPWD	Référence du modèle de mail utilisé pour l'envoi d'une demande de changement de mot de passe.	A	AUTH_TPLMAILASKPWD

Le modèle de mail fourni par défaut (référence AUTH_TPLMAILASKPWD) sera écrasé à chaque mise à jour de dyncase-platform.. Si vous voulez personnaliser votre courriel, vous devez le dupliquer mettre une référence et l'indiquer dans le paramètre AUTHENT_MAILASKPWD.

Le modèle de mail est fait pour la famille "utilisateur". Il doit indiquer l'attribut 'us_meid' dans les correspondants (c'est celui qui fait la demande). Le mot-clef [LINK_CHANGE_PASSWORD]. indique le lien sur lequel l'utilisateur devra cliquer pour accéder à l'interface de changement de mot de passe.

4.5.4.2 Paramètre pour la gestion des expirations des comptes internes

Nom du paramètre	Description	Type	Valeur par défaut
AUTHENT_FAILURECOUNT	Nombre de tentatives consécutives pour se connecter avec un faux mot de passe avant le blocage du compte.	G	

4.5.4.3 Paramètre pour la force du mot de passe (comptes internes uniquement)

Nom du paramètre	Description	Type	Valeur par défaut

AUTHENT_PWDMINLENGTH	Nombre minimum de caractères d'un mot de passe	G	0
AUTHENT_PWDMINDIGITLENGTH	Nombre minimum de chiffre d'un mot de passe	G	0
AUTHENT_PWDMINUPPERALPHALENGTH	Nombre minimum de lettres majuscule d'un mot de passe	G	0
AUTHENT_PWDMINLOWERALPHALENGTH	Nombre minimum de lettres minuscules d'un mot de passe	G	0
AUTHENT_PWDMINSYMBOLLENGTH	Nombre minimum de caractères qui ne sont ni des lettres ni des chiffres	G	0

Le mot de passe utilisé pour les comptes internes peut comporter aussi les lettres avec accents. On considère un mot de passe fort s'il y a au moins 16 caractères avec au moins un ou deux caractères dans chaque catégorie (source ANSSI).

4.5.5. APPMNG - Gestion des applications

Pas de paramétrage

4.5.6. CORE - Noyau

Nom	Définition	Type	Défaut
	Système		
CORE_URLINDEX	url d'accès à l'index de dynacase Exemple : http://192.168.34.12/dynacase/ Cette URL est utilisé notamment pour les liens dans les mails	G	Note Version > 2.7.0
MEMORY_LIMIT	limite mémoire par processus	G	32M

CORE_DBCONNECT	mode de connexion aux bases données.
	Mettre en persistent peut permettre une accélération des requêtes. Dans ce cas il faut penser à modifier les 2 paramètres PHP (fichier php.ini) : De plus il est nécessaire de bien paramétrer le fichier postgresql.conf
	`# - Connection Settings -`
	`max_connections = 100`
	Le nombre de connection doit être approximativement 3 fois le nombre max de requêtes en parallèles .
	La prise en compte de ce paramètre n'est effectif qu'après avoir lancé le script `/usr/share/what/wstart`.
	La modification en persistent ne doit être fait que si nécessaire après des tests de perfomances. Plus de détails
		G	unpersistent
			Version > 2.7.1

USE_FREEDOM_USER	utilise dynacase comme interface de modification des utilisateurs	G	yes
	Personnalisation client		
STYLE	Style par défaut	GU	default
FONTSIZE	Taille par défaut	GU	normal
CORE_CLIENT	nom du client	G	
CORE_LOGOCLIENT	logo du client	G	

CORE_LANG	langue par défaut	GU	fr_FR
Installation			
CORE_DB	accès base de donnée principale	G	dbname=anakeen user=anakeen
CORE_PUBDIR	répertoire d'installation	G	/usr/share/what
CORE_REALM	nom du royaume de connection (voir conf HTTP)	G	dynacase Connection

4.5.7. FDL - Bibliothèque dynacase

L'application FDL définit un ensemble de fonctions pour la manipulation des documents. Ces paramètres sont tous globaux et sont utilisés par les autres applications : FREEDOM, GENERIC et ONEFAM.

Nom	Définition	Type	Défaut
FREEDOM_DB	base de donnée dynacase paramètres d'accès à la base de données postgresql pour la base documentaire.	G	défini à l'installation
ENUM_TITLE_SIZE	taille maximum de chaines de caractère pour les choix.	GU	40
FDL_BCC	copie envoi de mail. Mettre 'yes' si vous voulez recevoir une copie des mails que vous envoyez via le serveur. Dans ce cas, vous êtes mis en destinataire caché (Blind Carbon Copy).	GU	No
FDL_MAX_FGEXPORTDOC	Nombre maximum de document pouvant être importé directement, c'est à dire pas en tâche de fond. Cela dépend de la puissance du serveur, un timeout de 30s est déclenché si le serveur n'a pas pu traiter tous les documents. Seuls les documents traités dans les 30s seront importés.	G	20
SMTP_HOST	nom ou adresse ip du serveur SMTP	G	localhost
SMTP_PORT	numéro de port du serveur SMTP	G	25
SMTP_LOGIN	en cas de connexion avec authentification. Identifiant de connexion	G	
SMTP_PASSWORD	mot de passe pour connexion authentifiée	G	
SMTP_FROM	adresse email de retour en cas d'envoi de mail par utilisateur sans mail	G	
TE_HOST	nom du serveur de transformation	G	localhost
TE_PORT	numéro port de la connection vers le serveur de transformation	G	10000

dynacase n'a pas par défaut un serveur SMTP pour envoyer les mail. Il est nécessaire de configurer l'accès à un serveur SMTP pour bénéficier des envois de documents par email.

4.5.8. GENERIC - Manipulation d'une famille

L'application GENERIC n'est pas utilisée directement. Ses paramètres sont transmis aux applications héritant de GENERIC. Il n'est pas nécessaire de modifier ces paramètres.

Nom	Définition	Type	Défaut
DEFAULT_FAMILY	famille de document par défaut	A	
DEFAULT_FLD	n° dossier par défaut	A	
CARD_SLICE_LIST	nombre de cartes par	AU	8

	page		

4.5.9. FREEDOM - Gestion documentaire

Nom	Définition	Type	Défaut
ROOT_FLD	Identificateur du dossier racine Ceci permet de définir le dossier racine du plan de classement commun	A	9

4.6 VAULT - Coffre de stockage des fichiers

Le VAULT (Coffre) est un module qui gère le stockage et l'accès aux fichiers attachés aux documents.

La localisation du coffre et sa capacité est modifiable par l'interface web via l'application Coffre (accessible si connecté en tant qu'administrateur).

4.6.1. Interface

Utilisation du vault :

- *Utilisé* volume utilisé par les fichiers liés aux documents dynacase
- *Poubelle* volume utilisé par les fichiers liés aux documents dynacase supprimés
- *Orphelins* sont ceux qui ne sont rattachés à aucun document dynacase. Il ne sont plus accessibles via l'interface classique. Ces orphelins sont créés lors des documents ont été supprimés de la base de données lors du vidage de le poubelle ou par une opération manuelle sur la base de données. Ils peuvent à tout moment être supprimés par l'administrateur pour récupérer de la place.
- *Libre* : espace disponible

4.6.2. Création d'un coffre

Le bouton **Nouveau coffre** situé au dessus de la liste des coffres existants, permet la création d'un coffre.

Les informations à fournir sont :

- le chemin complet de la racine du coffre. Ce dossier doit exister , être accessible en lecture et écriture par le user sous lequel tournent les processus apache et être vide.
- la taille allouée.

4.6.3. Modification du volume d'un coffre

Pour modifier sa capacité il suffit de cliquer sur 'Augmenter la taille du coffre', puis de saisir la nouvelle taille. dynacase n'a pas en charge de vérifier la capacité réelle de stockage. L'administrateur doit au préalable vérifier que cette capacité est réellement disponible. La capacité renseignée est une contrainte logique. Dès que la somme des tailles des fichiers stockées est dépassée, dynacase bloque l'ajout de nouveaux fichiers.

4.6.4. Déplacement d'un coffre

Pour modifier la localisation, il faut procéder à deux opérations. L'administrateur doit déplacer physiquement le répertoire (généralement utilisation de la commande mv unix). Attention, il faut que le répertoire, et l'ensemble de ses fichiers et sous-répertoires) soient accessibles en lecture et écriture par le compte utilisateur unix qui a lancé le serveur web (généralement http ou apache). Ensuite, il doit indiquer via l'interface web le nouveau répertoire.

4.6.5. Commande shell dynacase

- `./wsh.php –api=VaultIndexInit' : reconstruit la table d'index entre le(s) coffre(s) et la base de données dynacase. Utile en cas de modification des bases de données directement. Cette table d'index est utilisée lors de la suppression des orphelins.
- `./wsh –api=VaultExamine –cmd=clean-unref' : Supprime les orphelins

* `./wsh –api=vault_init –size=500000000 –path=/var/freedom/chemin' : Crée un nouveau coffre de 500Mo dans le répertoire /var/freedom/chemin.

A compter de la version 3.1.0 du VAULT, il est possible de fournir un **nom logique** au nouveau coffre. La commande wsh est la la suivante : `./wsh.php –api=vault_init **-name=nom_logique_du_coffre** –size=500000000 –path=/var/freedom/chemin'

4.7 Type MIME

Pour la détection du type MIME d'un fichier, dynacase utilise les mécanismes suivant :

- Association type MIME par l'extension du fichier
  - En utilisant un fichier d'association local s'il est présent ($CONTEXT_ROOT/admin/mime-user.conf)
  - En utilisant un fichier d'association global ($CONTEXT_ROOT/admin/mime.conf)
- Utilisation de la commande système `file'

L'ordre d'application des règles d'association est le suivant :

1) Si le fichier `mime-user.conf' existe, alors appliquer les règles de détection de `mime-user.conf'
2) Si le type n'a pas été trouvé, alors appliquer les règles de détection de `mime.conf'
3) Si le type n'a pas été trouvé, alors appliquer l'utilisation de la commande système `file'

4.7.1. Fichier d'association local `$CONTEXT_ROOT/admin/mime-user.conf'

Le fichier `$CONTEXT_ROOT/admin/mime-user.conf' n'est pas fournit par dynacase, et permet donc à l'administrateur local du système de le créer et d'y insérer ses propres règles d'association.

Ce fichier est donc optionnel, et ne sera pas écrasé par les mises-à-jour de dynacase.

Vous pouvez créer ce fichier à partir du modèle fournit dans `$CONTEXT_ROOT/admin/mime-user.conf.sample' :

```
$ cp admin/mime-user.conf.sample admin/mime-user.conf
```

Le format XML du fichier est le suivant :

```
<?xml version="1.0"?>
<mimes>
  <mime ext="foo" sys="application/foo" text="Foo files" />
  <mime ext="bar" sys="application/bar" text="Bar files" />
</mimes>
```

Les règles d'association sont décrites par des éléments <mime/> avec les attributs suivants :

- 'ext' : l'extension du fichier sans le point
- 'sys' : le type MIME de ce type de fichier
- 'text' : la description textuelle de ce type de fichier

4.7.2. Fichier d'association global `$CONTEXT_ROOT/admin/mime.conf'

Le fichier `$CONTEXT_ROOT/admin/mime.conf' est fournit par dynacase.

Ce fichier ne doit pas être modifié car il sera remplacé lors des mises-à-jour de dynacase. Pour déclarer vos propres

règles d'association, utiliser le fichier `mime-user.conf' décrit ci-dessus.

Le format XML du fichier est identique à celui du fichier `mime-user.conf' décrit ci-dessus.

4.8 Export Première Forme Normale

4.8.1. Principe

dynacase fournit un script WSH permettant d'exporter des familles de dynacase dans un modèle dit de "Première Forme Normale".

Cet export contient les données exportées dans un schéma de base relationnelle "standard" soit au format SQL soit dans une base Postgresql, soit les deux.

4.8.2. Préambule

Pour certains besoins d'analyse, il est utile de pouvoir consulter les données dynacase via des outils de reporting (BO par exemple).

Ces outils ne savent exploiter les données que si elles respectent un modèle de "Première Forme Normale".

Les données dynacase sont stockées dans un modèle documentaire. Elles doivent donc être exportées dans le modèle de "Première Forme Normale" pour pouvoir être exploitées correctement.

Ce document spécifie le mécanisme d'export et le format d'export de la base dynacase exportée.

Exemple simple sur la famille Animal :

Attribut	Type
AN_NOM	text
AN_TATOUAGE	int
AN_ESPECE	docid("ZOO_ESPECE")
AN_ORDRE	text
AN_CLASSE	docid("ZOO_CLASSE")
AN_SEXE	enum
AN_PHOTO	image
AN_NAISSANCE	date
AN_ENTREE	date
AN_ENFANT	docid("ZOO_ANIMAL")
AN_PERE	docid("ZOO_ANIMAL")
AN_MERE	docid("ZOO_ANIMAL")

Voilà le schéma une fois exporté :

Les docid qui sont des liens vers d'autres familles ont engendré la création de plusieurs tables avec des contraintes de relation entre elles.

4.8.3. Pré-requis

L'export 1NF a besoin pour travailler d'une base postgresql temporaire de travail.

Cette table doit exister avec son pgservice associé pour pouvoir exécuter l'export.

Ci-dessous la procédure à suivre pour créer le pgservice si besoin est :

```
$ sudo vi /etc/postgresql-common/pg_service.conf
```

Ajouter le service suivant (personnalisez vos dbname, password et nom de service si besoin)

```
[tmp_1nf]
host=localhost
port=5432
user=freedomowner
password=password
dbname=tmp_1nf
```

Connectez vous à postgres :

```
$ sudo -u postgres psql
```

Créez la base :

```
=# CREATE DATABASE tmp_1nf OWNER freedomowner;
```

Redémarrez Postgresql :

```
$ sudo /etc/init.d/postgresql-8.4 restart
```

L'exécution du WSH doit se faire dans le contexte du wiff :

```
$ sudo /var/www/wiff/wiff context "MON_CONTEXTE" exec /bin/bash --login
```

Enfin, le lancement du script :

```
$ ./wsh.php --api=fdl_export1nf
```

L'aide suivante apparaît :

```
Usage:
  /var/www/freedom1/API/fdl_export1nf.php
      --config=<config.xml>
      --outputsql=<file_name> | --outputpgservice=<pgservice>]
      [--tmppgservice=<tmp_pgservice_name>] (default tmp_1nf)
      [--tmpschemaname=<tmp_schemaname>] (default 1nf)
      [--tmpemptydb=<yes|no>] (default yes)
      [--sqllog=<file>] (default none)
```

4.8.4. Paramètres du script WSH

Paramètre	Optionnel	Défaut	Description
config	N		Fichier xml de config des familles à exporter
outputsql	Y/N		Fichier sql "standard" résultat d'exportation
outputpgservice	Y/N		Pgservice postgresql résultat d'exportation NB: la base associée doit être vide avant l'exportation !
tmppgservice	Y	tmp_1nf	Pgservice temporaire dans lequel est effectué le travail d'exportation
tmpschemaname	Y	tmp_1nf	Nom de schéma postgresql de la base temporaire dans lequel les données seront exportées.
tmpemptydb	Y	yes	Indique si le pgservice temporaire doit être vidé et ré-importé avant de travailler (dump/restore de dynacase dans la base temporaire). Le fait de mettre à **no** ce paramètre permet de faire plusieurs exports successifs différents avec seul le premier qui effectue le dump/restore de dynacase
sqllog	Y		Fichier sql temporaire d'exportation qui contient les requêtes sql pour Postgresql (sert dans le cas ou vous utilisez outputpgservice). La différence avec outputsql réside dans le fait que ce fichier contient des requêtes spécifiques Postgresql pour améliorer les performances d'exportation.

4.8.5. Fichier XML

Petit exemple sur le Zoo :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
```

```
    <column attribute="an_tatouage" />
    <column attribute="an_espece" />
    <column attribute="an_ordre" />
    <column attribute="an_classe" />
    <column attribute="an_classe:cl_nom" />
    <column attribute="an_sexe" />
    <column attribute="an_photo" />
    <column attribute="an_naissance" />
    <column attribute="an_entree" />
    <column attribute="an_enfant" />
    <column attribute="an_pere" />
    <column attribute="an_mere" />
  </table>
  <table family="ZOO_ESPECE" >
    <column property="revdate" />
    <column property="initid" />
    <column attribute="es_identification" />
    <column attribute="es_caracteristique" />
  </table>
</database>
```

La structure est simple : une "database" qui contient des "table" qui contiennent des "column". Rien de bien nouveau.

La seule subtilité réside dans le fait qu'une colonne est soit un "attribute", soit une "property" au sens dynacase des documents.

Une astuce pouvant vous faciliter l'écriture du fichier xml est d'écrire (comme ci-dessus pour ZOO_ESPECE) comme attribut, un onglet (tab), cadre (frame) ou tableau (array) : tous les attributs les constituant seront automatiquement exportés.

Une fonctionnalité agréable est fournie si vous utilisez la syntaxe docid:attribute dans un nom d'attribut. Il s'agit d'une aide pour rajouter une colonne à la famille référencée par le docid. Cela reviendrait à rajouter une table famille dans le fichier xml avec cet attribut (cf. exemple ci-dessus avec an_classe:cl_nom).

4.8.6. Fonctionnement

4.8.6.1 Phases d'exportation

1. Analyse et chargement du fichier XML de config

2. Tests de connexion aux pgservices de dynacase et de la base temporaire

3. Dump de dynacase et restauration dans la base temporaire. On peut décider d'éviter cette phase en utilisant le paramètre "tmpemptydb" à "no".

4. Chargement de la config : opérations de chargement des familles, analyse des attributs etc ...

5. Création du schéma temporaire (il est supprimé s'il existe déjà)

6. Création des tables

7. Remplissage des tables

8. Application des contraintes SQL de relations entre tables

9. Chargement du fichier de log postgresql dans le pgservice destination si celui-ci est indiqué (option "outputpgservice")

4.8.6.2 Erreurs détectées

Les erreurs listées ici sont celles qui sont explicites. D'autres erreurs pourraient survenir qui ne sont pas listées, dans ce cas, cela risque plus probablement d'être des erreurs système (disque full, problème de base/pgservice, ...). Basez-vous sur leur message pour le dépannage.

Phase 1 :

- pas de noeud xml "database"

- plusieurs noeuds xml "database"
- pas d'attribut "family" sur un noeud xml "table" ou attribut vide
- pas d'attribut "attribute" ni "property" sur un noeud xml "column" ou attributs vides
- aucun noeud xml "table" dans le fichier

Phase 2 :

- échec de connexion aux pgservices

Phase 3 :

- impossible de créer le fichier de dump
- impossible de trouver la commande pg_dump
- impossible de trouver la commande psql

Phase 4 :

- impossible de trouver la famille précisée dans le fichier xml
- famille non valide ou pas vivante
- propriété de colonne incorrecte ou inconnue
- impossible de trouver l'attribut du fichier xml dans la famille
- syntaxe spéciale docid:attribute ne référence pas un docid
- aucune colonne détectée dans un attribut array
- type d'attribut incohérent (cette erreur n'est jamais sensée arriver)

Phase 5 :

- pas d'erreur particulière sauf si erreur SQL

Phase 6 :

- pas d'erreur particulière sauf si erreur SQL

Phase 7 :

- table référencée introuvable (pas sensé arriver)
- pas d'erreur particulière sauf si erreur SQL

Phase 8 :

- pas d'erreur particulière sauf si erreur SQL

Phase 9 :

- impossible de trouver la commande psql

4.8.6.3 Attributs supportés

Tous les attributs exportés sont par défaut au format "text", "int" ou "double" selon leur origine.

Quelques attributs spéciaux se voient être transformés lors de l'exportation :

enum	Une table liée contenant les énumérés est automatiquement créée, la liaison est faite par la clé de l'énuméré.
enum *avec multiple=yes*	Une table liée contenant les énumérés est automatiquement créée, la liaison est faite par une table intermédiaire permettant la liaison multiple.
docid("FAMILLE")	Une table FAMILLE est ajoutée et une liaison est automatiquement effectuée.
docid("FAMILLE") *avec multiple=yes*	Une table FAMILLE est ajoutée et une table intermédiaire aussi permettant de gérer la liaison multiple.

docid	On recherche la famille liée (cela nécessite l'existence d'un autre attribut ayant phpfunc = lfamily(D,FAMILLE...):ATTRIBUT_DOCID S'il n'y a pas de famille détectable selon ce principe, l'export échouera avec un message clair. *schéma: cf. docid("FAMILLE")*
docid *avec multiple=yes*	Même principe que docid et docid("FAMILLE") *schéma: cf. docid("FAMILLE") avec multiple=yes*

Cas des attributs placés dans un **tableau (array)** :

attribut "classique"	Une table liée pour le tableau est créée et qui contient les lignes de celui-ci dont une colonne contenant l'attribut. array
enum	Une table liée contenant les énumérés est automatiquement créée, la liaison est faite par la clé de l'énuméré sur la table du tableau. array
enum *avec multiple=yes*	Non géré par dynacase pour l'instant donc par l'export aussi.
docid("FAMILLE")	Une table FAMILLE est ajoutée et une liaison est automatiquement effectuée via la table du tableau. array
docid("FAMILLE") *avec multiple=yes*	Une table FAMILLE est ajoutée et une table intermédiaire aussi permettant de gérer la liaison multiple via la table du tableau. array
docid	On recherche la famille liée (cela nécessite l'existence d'un autre attribut ayant phpfunc = lfamily(D,FAMILLE...):ATTRIBUT_DOCID S'il n'y a pas de famille détectable selon ce principe, l'export échouera avec un message clair. *schéma: cf. docid("FAMILLE")*
docid *avec multiple=yes*	Même principe que docid et docid("FAMILLE") *schéma: cf. docid("FAMILLE") avec multiple=yes*

Remarque sur le type **thesaurus** : ce type est traité comme un docid("THCONCEPT"). Qu'on soit avec multiple=yes ou pas etc ...

Remarque générale concernant les **docid** : dans la version actuelle, l'exportation ne fait les liaisons de docid que sur les document de dernière version qui sont valides (pas supprimés). Les autres liaisons ne sont pas exportées. La raison est simple : les contraintes de relations entre tables dans le modèle de Première Forme Normale impose des contraintes de relations valides. C'est à dire que le document lié doit exister. Sans cela, les contraintes SQL échoueraient et l'import aussi.

Pour chaque famille référencée, l'id et le title des documents sont exportés.

4.8.6.4 Quelques exemples

Exemple simple mono-famille :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
  </table>
</database>
```

Exemple plus étoffé avec enum et docid :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
    <column attribute="an_tatouage" />
    <column attribute="an_espece" />
    <column attribute="an_classe" />
    <column attribute="an_sexe" />
  </table>
</database>
```

Exemple précédent avec l'utilisation de la syntaxe particulière sur les docid :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
    <column attribute="an_tatouage" />
    <column attribute="an_espece" />
    <column attribute="an_classe" />
    <column
attribute="an_classe:cl_nom" />
    <column attribute="an_sexe" />
  </table>
</database>
```

C'est équivalent à l'exemple suivant :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
    <column attribute="an_tatouage" />
    <column attribute="an_espece" />
    <column attribute="an_classe" />
    <column attribute="an_sexe" />
  </table>
  <table family="ZOO_CLASSE" >
    <column attribute="cl_nom" />
  </table>
</database>
```

Enfin un exemple un peu plus complexe avec un tableau et des propriétés :

```
<?xml version="1.0"?>
<database>
  <table family="ZOO_ANIMAL" >
    <column attribute="an_nom" />
    <column attribute="an_tatouage" />
    <column attribute="an_espece" />
    <column attribute="an_ordre" />
    <column attribute="an_classe" />
    <column attribute="an_sexe" />
    <column attribute="an_photo" />
    <column
attribute="an_naissance" />
    <column attribute="an_entree" />
    <column attribute="an_enfant" />
    <column attribute="an_pere" />
    <column attribute="an_mere" />
  </table>
  <table family="ZOO_ESPECE" >
    <column property="revdate" />
    <column property="initid" />
    <column
attribute="es_identification" />
    <column
attribute="es_caracteristique" />
  </table>
</database>
```

4.8.7. Quelques chiffres

Un test d'exportation a été fait sur une base déjà conséquente pour tester l'export en grandeur nature.

L'export a été fait sur une machine de puissance moyenne (laptop).

Phase	Volume	Durée
Dump dynacase	1.7 Go	4 min
Load du dump	1.7 Go	7 min
Export 1NF	8 familles 69 tables générées 383 champs générés 312 500 enregistrements toutes tables confondues	9 min
Load dans le pgservice destination	95 Mo	2 min
	Total	22 min

Découpage de la phase d'Export 1NF de manière grossière :

- le processeur a tourné en moyenne à 60% du CPU
- 2 minutes pour postgresql
- 1 minute pour l'écriture des logs
- 6 minutes pour le reste (principalement PHP)

Schéma relationnel final :

4.9 Aide en ligne

4.9.1. Principe et fonctionnement général

dynacase propose des mécanismes d'aide en ligne. Ils permettent, depuis les interfaces standards des documents, de proposer à l'utilisateur un accès direct à une documentation d'aide.

Les documents d'aide en ligne sont stockés dans dynacase et par conséquent sont modifiables directement depuis les interfaces WEB dynacase par les personnes habilitées.

Les documents d'aides peuvent être traduits en plusieurs langues. L'interface de consultation propose par défaut les traductions pour la langue d'utilisation de dynacase (si elles existent) et donne la possibilité à l'utilisateur d'accéder aux autres traductions disponibles.

Un document d'aide est composée de rubriques, chacune des rubriques est traduisible individuellement.

4.9.2. Utilisation

Un document d'aide en ligne peut être lié à une famille. Dans ce cas il est accessible via le lien d'aide sur l'attribut dans l'édition d'un document de cette famille.

Soit il s'agit d'un document d'aide non lié à une famille et il peut être accédé en consultation comme tout document dynacase.

Les documents d'aide en ligne ont des vues de consultation et d'édition spécifiques afin de faciliter la lecture et la saisie.

Il est possible de mettre des liens vers des documents d'aide en ligne, qu'ils soient liés ou non à une famille, via une syntaxe [ADOC aide] ou [ADOC aide#rubrique]

Consultation

Edition

4.9.3. Paramétrage de la famille aide en ligne

Le paramétrage de la famille est indispensable pour configurer les langues dans lesquelles l'aide sera disponible.

Dans la Gestion Documentaire, vous allez sur le document famille de "Aide en ligne" et vous faites "Valeur des paramètres"

Vous pouvez paramétrer les langues disponibles pour l'Aide en ligne :

Les codes à utiliser sont les codes ISO standards pour les locales.

Les plus courants sont en_GB (Royaume-Uni), en_US (Etats-Unis), fr_FR (France), it_IT (Italie), es_ES (Espagne), ...

Plus d'info sur la construction des codes à cette adresse https://wiki.mozilla.org/L10n:Locale_Codes

4.9.4. Droits

La famille aide (HELPDOC) est une famille système.

Comme tout document dynacase, l'accès aux aides (création, modification et consultation) est soumis aux droits dynacase.

Le droit création de document sur la famille permet de contrôler la création d'aides en ligne.

Seul les utilisateurs ayant le droit de modification sur les documents peuvent saisir les traductions.

Pour que les documents soient consultables par les utilisateurs dynacase, il faut penser à ajouter le droit "Voir" pour le groupe "Utilisateurs" dans "Modifier les droits des documents" du document famille :

Aide en ligne	Voir	Edit	Delete	Envoyer	Déverrouiller	Confidentiel	Forum	Voir les réponses	Voir les droits	Modifier les droits
Administrateur des espaces de travail										
Espace de travail										
lecteur de Premier espace										
rédacteur de Premier espace										
Utilisateurs	●									
Administrateurs	○	●	●						●	●

Valider Réinitialisation Seulement les cochés

4.9.5. Création d'une aide en ligne

4.9.5.1 Liées à une famille

Une aide est constituée d'une nom, d'une description et d'une série de rubriques. Chaque rubrique peut être liée à un attribut spécifique de la famille à laquelle est liée l'aide. Lors de la création d'une rubrique, la "clé" de la rubrique vous est demandée : soit vous mettez le nom de l'attribut soit vous saisissez autre chose.

Une aide à la saisie pour le nom des attributs de la famille sera proposée à partir de dynacase Toolbox 3.0.11

Il est possible de créer une aide liée à une famille en cliquant sur "Créer une aide" depuis le document de la famille concernée.

Si l'aide existe déjà, le menu propose simplement de modifier l'aide associée.

Remarque : une fois l'aide associée à cette famille, il est impossible de rompre ou de changer ce lien aide / famille.

4.9.5.2 Autonome

Une aide est un document au sens dynacase, donc il est possible de créer une aide comme un document standard, à ceci près que l'aide ne sera liée à aucune famille et ne pourra être liée à aucune famille par la suite étant donné que ce lien aide / famille est quelque chose d'immuable et non modifiable.

4.9.6. Édition d'une aide en ligne

4.9.6.1 Vue d'édition

On remarquera dans l'édition deux zones distinctes matérialisées par les cadres : "Aide en ligne" qui propose le libellé et la description de l'aide (localisé) et "Rubriques" qui liste les rubriques de l'aide.

4.9.6.2 Traduction du nom de l'aide et de sa description

Pour modifier "Aide en ligne", il suffit de cliquer sur "Modifier les traductions" ce qui ouvrira une fenêtre supplémentaire ou vous pourrez traduire le nom et la description de l'aide. En cliquant sur les drapeaux, vous faites apparaître les traductions pour la langue sélectionnée. Vous pouvez modifier les traductions ou les supprimer an utilisant le menu "Effacer cette traduction" Le menu Appliquer vous permet de prendre en comptes les traductions réalisées. La fenêtre de traduction est fermée.

4.9.6.3 Ajout d'une rubrique

L'ajout d'une rubrique est possible en utilisant le menu "Ajouter une rubrique". La rubrique est ajoutée (sans traduction) en début de la liste des rubriques existantes.

4.9.6.4 Organiser les rubriques

Il est possible d'organiser les rubriques dans l'ordre souhaité via les liens Monter ↑ Descendre ↓

4.9.6.5 Traduction des rubriques

Pour ajouter/éditer/supprimer des traductions d'une rubrique, vous devez cliquer sur "Modifier les traductions" ce qui vous ouvre une fenêtre supplémentaire dans laquelle vous pouvez faire toutes ces actions :

La clé de la rubrique est affichée en guise d'information, elle n'est pas modifiable.

Vous pouvez changer le libellé et la description de la rubrique langue par langue en sélectionnant la langue voulue en haut à droite de la fenêtre.

Une traduction vide (ie libellé vide et description vide) signifie qu'il n'y pas de traduction dans cette langue.

Si aucune des langues n'est renseignée, la rubrique est tout simplement supprimée car l'existence de la rubrique présuppose l'existence d'au moins une traduction.

4.9.6.6 Liens inter-aides

Au sein des aides, il est possible de mettre des liens internes entre aides.

Pour cela, il suffit simplement d'utiliser la syntaxe dynacase [ADOC docid_aide#cle_rubrique] ou bien [ADOC nom_logique_aide#cle_rubrique]

Sachant que dans le cas d'aide liée à une famille, "cle_rubrique" sera probablement un nom d'attribut.

Vous pouvez omettre "#cle_rubrique" si vous souhaitez faire un lien vers l'aide en général et pas forcément vers une rubrique de celle-ci.

4.9.7. Consultation d'une aide en ligne

4.9.7.1 Accès à la consultation via un attribut de famille

La fenêtre d'édition d'un document d'une famille liée à une aide met en évidence les attributs qui ont une aide via des liens. Si vous cliquez sur ce lien, une fenêtre est ouverte et présente l'aide de la famille en se positionnant sur la rubrique correspondant à cet attribut.

4.9.7.2 Vue de consultation

On distingue plusieurs zones :

1. Libellé de l'aide

2. Liste des rubriques de l'aide, si vous cliquez sur une rubrique, la zone à droite bouge de manière à vous placer sur cette rubrique

3. Liste des aides disponibles, vous pouvez facilement changer d'aide en cliquant dans cette liste

4. Libellé d'une rubrique (localisé selon le changement de langue en 7)

5. Description de la rubrique (localisé selon le changement de langue en 7)

6. Change toutes les langues des rubriques dans la langue indiquée en cliquant sur un drapeau

7. Change la langue de la rubrique voulue via le drapeau adéquat (un drapeau semi transparent indique qu'il n'y a pas de traduction dans cette langue)

fin du document

Ce document est publié sous licence CC.

http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

Vous êtes libres :

- de reproduire, distribuer et communiquer cette création au public
- de modifier cette création

Selon les conditions suivantes :

- **Paternité** — Vous devez citer le nom de l'auteur original de la manière indiquée par l'auteur de l'œuvre ou le titulaire des droits qui vous confère cette autorisation (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'œuvre).
- **Pas d'Utilisation Commerciale** — Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.
- **Partage des Conditions Initiales à l'Identique** — Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.