**Cegid XRP Ultimate**

Workflow Information Manager

cegid

Document de release de la version I2.01


COMMENT GARDER L'ISO FONCTIONNALITE
FONCTIONNALITES
- Nouveautés
  - GKTUMELT : Envoi de mail pour un traitement depuis l'interface utilisateur
  - WimServlet GED
  - Appel CURL
  - WimServlet : sécurité SQL
  - WimServlet : sécurité sur actions
- Modifications
  - Boîte de connexion
  - WimServlet
  - WimServlet

Ce document présente les évolutions survenues sur l'application Cegid XRP Ultimate Workflow Information Manager en I2.01.

## Comment garder l'iso fonctionnalité

Aucune modification de paramétrage n'est nécessaire pour que le fonctionnement soit comme avant le passage de la release.

# Fonctionnalités

## Nouveautés

### GKTUMELT : Envoi de mail pour un traitement depuis l'interface utilisateur

Une nouvelle fonctionnalité de l'envoi de mail est accessible depuis n'importe quelle gestion de Cegid : l'envoi de mail depuis un traitement. Comme pour le GKTUMEL classique, il est possible de définir les destinataires (et leur type : principal, copie, copie cachée), le texte qui va être envoyé, les pièces à joindre, le modèle de texte souhaité, etc.

#### *Explications*

Cela se fait au travers de la gestion des messages électroniques pour les traitements (GKTUMELT), accessible uniquement en synchronisation d'une autre gestion.

Contrairement au GKTUMEL classique, le GKTUMELT ne permet pas d'associer de requête WIM, l'envoi des mails se fera de manière générale.

#### *Mise en place*

Dans un premier temps, il est nécessaire de procéder au paramétrage des messages électroniques (GTUPME) et de définir, pour l'application, l'objet et un traitement, les grandes lignes qui vont contrôler le comportement de l'envoi des mails. On peut notamment préciser qui sera l'émetteur du mail, à qui sera adressée la réponse quand le destinataire fera "Répondre", s'il faut mettre des personnes en copie.
A noter : selon l'application d'origine (e-Achats, XRM, etc.), le contenu des menus déroulants est variable.

Puis, définir une synchronisation entre la gestion appelante et la gestion des messages (GKTUMELT). Il faut donc lier les mnémoniques (via GAMN) et donner les paramètres de synchronisation passés en entrée de GKTUMELT.

Exemple : Mise en place depuis l'écran des gestionnaires (GCDA).

- Associer GCDA et GKTUMELT dans GAMN
- Définir une configuration dans GTUPME, pour l'objet "saicda", l'application SAC et un traitement, par exemple "SATECA"
- Définir les paramètres ENTITE (préciser comme constante 'SACDA') et IDENT6 (passer la colonne NUISACDA) en sortie dans GTFPS pour la gestion SAICDA.

Une fois ceci terminé, il sera possible d'accéder à la gestion GKTUMELT.

Le bouton "Générer les messages" permettra la génération de tous les messages électroniques dont le paramétrage existe dans GTUPME. A chaque message créé, tous les destinataires le sont également.
Ces destinataires proviennent de GTUPME ou de GTUPDE, qui permet d'ajouter des destinataires supplémentaires.
Le GTUPDE fonctionne exactement de la même façon que les champs du bloc "Destinataire" du GTUPME, sauf que le nombre de lignes de données n'est pas limité.

### WimServlet GED

GED WimServlet / API Rest

### *Explications*

Une propriété (dans le fichier de config. côté serveur Web) "rest.gedConfigType" permet de préciser quelle configuration sera prise dans GTDTY. Celle de la ligne technologique RIA ou celle de la ligne HTML5. Par défaut, c'est la configuration RIA (afin de rester sur la configuration historique si aucun changement lors de la migration) qui sera utilisée. Sinon, il faut positionner la valeur "HTML5".

## Appel CURL

Lancer des appels de type CURL depuis une mise en forme

### *Explications*

Possibilité de définir, dans la mise en forme, l'appel à une commande de type CURL pour réaliser des appels de ce type.
Exemple de mise en forme :

```
!$CURL --location
@;@--request@;@POST
@;@'https://xxxx/service/'
@;@--header@;@'Content-Type: application/json'
@;@--data-raw@;@'{
    "appId": "xxxxxxx",
    "appToken": "yyyyyyyyyyyyyyyyyyyyyy",
    "accountName": "demoAccount"
    }'
CURL$!
```

Les paramètres utilisables sont les paramètres classiques de CURL. A noter toutefois le séparateur d'arguments qui est égal à "@;@"

Il est possible de passer un mot de passe dans la commande CURL. Pour cela, il ne faut surtout pas le coder en dur dans la mise en forme. Il faut stocker le mot de passe dans notre gestion sécurisé GTPWD, pour le type CURL_XXX, avec XXX correspondant à la variable GTPWD_XXX définie dans la mise en forme.
Par exemple, nous définissons un mot de passe qui permet d'accéder à une api quelconque. Dans GTPWD, nous créons pour le type CURL_APITST.
Pour l'utiliser dans la mise en forme, il faut saisir GTPWD_APITST

## WimServlet : sécurité SQL

Utilisation possible de la syntaxe SQL de type "bind"

### *Explications*

Pouvoir utiliser (et si besoin imposer) le binding des paramètres dans l'exécution des select.

L'écriture de la clause SQL devra alors se faire en respectant la syntaxe de type "bind", par exemple :

Select numgtets, devgtets from gtets where numgtets = ? and nmegtets = ?

Les paramètres de l'URL d'appel ne portent donc plus de nom, mais leur ordre et leur type ont ici toute leur importance.
Pour une chaîne de caractères, il faudra utiliser BINDEDSX. Pour un entier BINDEDIx et BINDEDF pour un float.
Avec X étant l'ordre de substitution du paramètres.

Nous aurons donc par exemple

http://xxx/com.qualiac.tus.WimServlet?action=select&requete=TESTBIND&$BINDEDS1=CEGID&$BINDEDI2=10.

Pour imposer cette syntaxe sur toutes les actions select, il suffit de positionner la globale WIMS_PRST avec la valeur "TRUE".
Mais attention, dans ce cas, les "anciennes" requêtes WIM qui utilisent des variables de type $ en tant que paramètres ne fonctionneront plus. Il sera nécessaire de les adapter à ce nouveau mode de fonctionnement.

### WimServlet : sécurité sur actions

Pouvoir appliquer la sécurité de l'Application Fondations (TTAU, etc.) sur des actions WimServlet

#### *Explications*

Possibilité d'appliquer la sécurité fondation (TTAU...) sur les API Rest, ainsi que sur les appels WimServlet pour action=select et action=callproc

## Modifications

### Boîte de connexion

Boîte de connexion par défaut

#### *Explications*

Boîte de connexion par défaut légèrement revue pour l'affichage des erreurs.

### WimServlet

Correctifs divers

#### *Explications*

Correctif d'un problème sur mise en forme du retour si plusieurs actions enchaînées

Portage actions multi-lignes

Portage de l'action "reconnect"

### WimServlet

Evolutions diverses

#### *Explications*

La variable globale WIMS_SBTMPW n'est plus indispensable. Si elle n'est pas présente, le répertoire de travail utilisé sera celui de la JVM.

Action getdoc, possibilité de passer le paramètre "typaff". Si présent (quelle que soit sa valeur), ce sera un lien qui sera retourné (et non plus le fichier directement).

Connexion : modification du message en cas d'erreur de login pour rajouter la possibilité d'une expiration du mot de passe.

Ajout action "showparam" pour lister les arguments (et headers) reçus en entrée.

Sur l'authentification, si le paramètre "no_sso" est passé, l'authentification se fera en mode non sso.

Sur l'authentification, si le paramètre no_ldap est passé, l'authentification se fera en mode non ldap.

Nouvelle globale WIMS_AUTHCOD qui permet (certains cas très particuliers le demandent) de personnaliser le code erreur http retourné en cas de problème d'authentification. Par défaut c'est le 401 classique qui est renvoyé.