generix Collaborative Entreprise

Release note socle technique 8.0

Type diffusion	**Externe**	**Auteur**	***Roa***
Révision *69*		**Date de création**	***22/11/2011***

Ce document résume les nouvelles fonctionnalités disponibles sur le socle technique pour la version commerciale GCE 1.61.

Ce socle se nomme 8.0, dorénavant les versions du socle technique sont découplées de la version de GCE afin d'apporter plus de souplesse quant aux développements de cette couche.

Comme à chaque nouvelle version, des travaux de fond ont été réalisés dans l'optique de limiter encore plus la consommation mémoire, ainsi que d'améliorer les performances brutes de l'application.

Société	:	***GENERIX Group***	Titre	:	*Projet : socle technique 8.0*
Dernière modification	:	*Administrateur*	Sujet	:	*Release note socle technique 8.0*
Temps de modification	:	*11421*	Résumé	:	
Nom du fichier		*Release note socle technique 8.0.doc*			
Liste de diffusion		Interne & externe			

Siège Social et Agence Nord
Immeuble le Verdi
6 rue du Moulin de Lezennes
BP 10215
59654 VILLENEUVE D'ASCQ Cedex
Tél. : +33 (0)3 20 41 48 00
Fax. : +33 (0)3 20 41 48 09
Service Technique :
Tél. : +33 (0)3 20 41 48 07
Fax. : +33 (0)3 20 41 48 08

Agence Région Parisienne

69/71 rue Beaubourg
75003 PARIS

Tél. : +33 (0)1 47 49 36 66
Fax. : +33 (0)1 47 51 22 17

Table des matières

1 Nouvelle version du socle technique

Il convient tout d'abord de lire le release note technique du socle 7.0 livré avec la version commerciale GCE 1.6.0.

Cette nouvelle version du socle technique est la 8.0, elle est **strictement incompatible avec les versions précédentes de GCE**. En effet, un gros travail de fond a été réalisé afin de limiter les dépendances à d'autres librairies (principalement Apache). Il en ressort que le nombre de librairies (jar) dorénavant nécessaire au bon fonctionnement de GCE est très fortement réduit. Un document annexe à celui-ci décrit tout cela.

L'objectif principalement de cette « réduction » a été de rendre l'application la plus proche possible du standard JEE5, basé sur un JDK6 et toutes les possibilités inhérentes à ces 2 versions.

A contrario, de nouvelles librairies ont fait leur apparition pour des besoins strictement « fonctionnel » (ex : code barres).

De nouvelles servlets font leur apparition dans cette version 8.0 du socle technique. Par défaut elles ne sont soumises à aucune restriction d'authentification au niveau du serveur d'application. Le cas échéant, il conviendra de faire le nécessaire sur site au niveau du module Tools permettant de générer les ear (section WAREAR).

2 Génération de code barres

Une nouvelle servlet fait son apparition : ServletBarCode

2.1 Définition

Comme toute nouvelle servlet, elle est doit être déclarée dans web.xml

```
<servlet>
	<display-name>ServletBarCode</display-name>
	<servlet-name>ServletBarCode</servlet-name>
	<servlet-
class>fr.generix.technicalframework.application.servlet.ServletBarCode</servlet-
class>
	<load-on-startup>-1</load-on-startup>
	<security-role-ref>
		<description>Global access role required</description>
		<role-name>sr_gce_overall</role-name>
		<role-link>sr_gce_overall</role-link>
	</security-role-ref>
</servlet>
```

```
<servlet-mapping>
	<servlet-name>ServletBarCode</servlet-name>
	<url-pattern>/ServletBarCode</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>ServletBarCode</servlet-name>
	<url-pattern>/GCE/ServletBarCode</url-pattern>
</servlet-mapping>
```

Il s'agit d'une servlet permettant de générer des codes barres, elle est basée sur le framework ZXing développé par Google.

Un nouveau jar fait donc son apparition dans la distribution : zxing.jar

Cette servlet génère une image de type PNG.

2.2 Utilisation

http://localhost:8888/gcetrunk/gce/ServletBarCode?barcode=1234567890123&barcodeType=EAN_13

http://localhost:8888/gcetrunk/gce/ServletBarCode?barcode=1234567890123&barcodeType=QR_CODE

http://localhost:8888/gcetrunk/gce/ServletBarCode?barcode=1234567890121&barcodeType=EAN_13&barcodeWidth=50&barcodeHeight=10

Liste des codes barres disponibles :

AZTEC, CODABAR, CODE_39, CODE_93, CODE_128, DATA_MATRIX, EAN_8, EAN_13, ITF, MAXICODE, PDF_417, QR_CODE, RSS_14, RSS_EXPANDED, UPC_A, UPC_E, UPC_EAN_EXTENSION

Remarque : par défaut, si le type de code n'est pas reconnu, on génère un QRCode.

En cas d'impossibilité de générer le code barre, l'image suivante est affichée en remplacement :

Les paramètres utilisables par cette servlet sont les suivants :

- barcode : la valeur du code barre à générer
- barcodeType : le type de code barre. Attention cette valeur est sensible à la casse et doit être contenue dans la liste fournie plus haut. Si le type demandé n'existe pas, on génère un QR_CODE par défaut
- barcodeHeight : la hauteur de l'image à générer
- barcodeWidth : la largeur de l'image à générer

La valeur par défaut des largeurs/hauteurs est de 100 pixels. Suivant le type de code barre à générer, les valeurs sont automatiquement modifiées par le framework ZXing (exemple : un qr code a forcément des dimensions carrées, un EAN13 ne peut pas faire moins de 100 pixels de large …)

3 Accès aux fichiers du « file system »

Il s'agit d'une nouvelle servlet qui permet d'accéder de manière restreinte aux ressources du « file system » (que l'on va appeler **FS**). L'ancienne méthode via paramétrage d'un serveur Apache … reste valide, cela vient en plus de l'ancien mode qui peut donc annuler et remplacer une mise en place technique complexe.

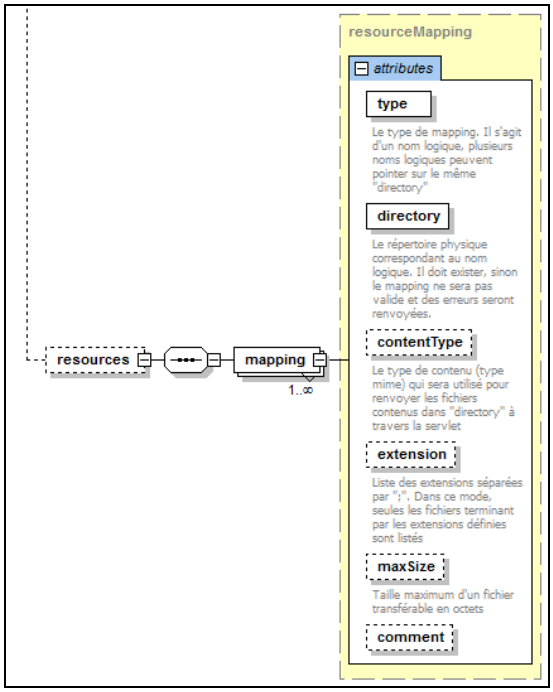
3.1 Définition

Comme toute nouvelle servlet, elle est doit être déclarée dans web.xml

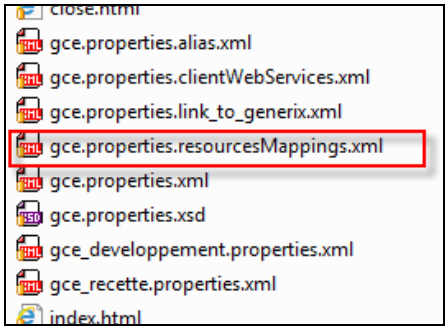
```
<servlet>
	<display-name>ResourceGet</display-name>
	<servlet-name>ResourceGet</servlet-name>
	<servlet-
class>fr.generix.technicalframework.application.servlet.ResourceGet</servlet-class>
	<load-on-startup>-1</load-on-startup>
	<security-role-ref>
		<description>Global access role required</description>
		<role-name>sr_gce_overall</role-name>
		<role-link>sr_gce_overall</role-link>
	</security-role-ref>
</servlet>
```

```
<servlet-mapping>
	<servlet-name>ResourceGet</servlet-name>
	<url-pattern>/ResourceGet</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>ResourceGet</servlet-name>
	<url-pattern>/GCE/ResourceGet</url-pattern>
</servlet-mapping>
```

La restriction aux ressources se fait dans gce.properties via une nouvelle section définie par le schéma XSD :



Exemple : ce fichier est externalisé vis-à-vis de gce.properties via un import XML




```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
	<mapping directory="/e:/generix/log" type="log" contentType="text/plain;
charset=cp1252" extension="log"/>
	<mapping directory="/e:/generix/spl" type="res" extension="spl;pdf"/>
	<mapping directory="/e:/generix/spl" type="excel" extension="xls;xlsx"/>
</resources>
```

On peut définir autant de type que l'on souhaite, l'accès à une ressource donnée se fera selon ce type. Par ailleurs, l'API métier (listerSpool) utilise ce paramétrage pour lister l'ensemble des fichiers liés à une édition, pour que le type soit pris en compte il faut nécessairement que l'attribut extension soit précisé.

On notera qu'il s'agit d'un exemple pour Windows, le fonctionnement est strictement identique sous Linux.
Remarque : le caractère « / » est indispensable sous windows avant la lettre représentant le lecteur.

Important : l'utilisateur qui démarre la JVM doit avoir un accès en lecture aux répertoires et aux fichiers concernés sinon cela ne pourra pas fonctionner correctement. Les répertoires peuvent être un point de montage (linux) sur une ressource partagée ou un lecteur virtuel (windows).

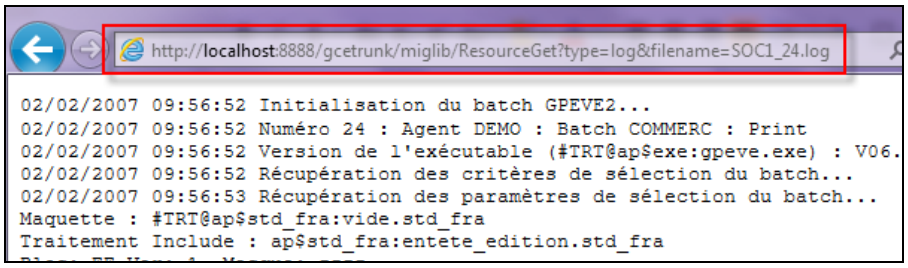
Définition d'un mapping :

- type : un nom logique qui sera utilisé par la servlet pour savoir à quel répertoire il faut accéder
- directory : le chemin du répertoire « physique » ou logique dans le cas d'un point de montage
- contentType : permet de spécifier le type MIME du contenu qui sera renvoyé par la servlet, ainsi on peut préciser un charset (exemple d'utilisation : les logs ne s'affichent pas avec les accents ➔ le charset doit être précisé car le contenu n'est pas interprétable par le navigateur).
- extension : la liste des extensions de recherche pour le répertoire en question. Ils doivent être séparés par des « ; » si on a besoin de définir plusieurs extensions.

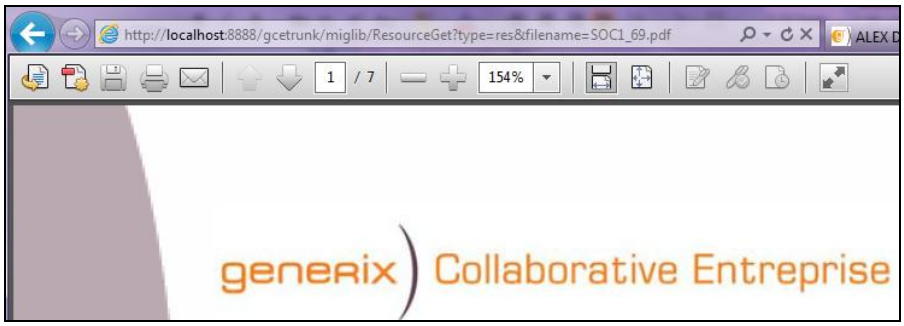
3.2 Utilisation

L'utilisation de la servlet est la suivante (pour la définition donnée dans le chapitre précédent) :

http://localhost:8888/gcetrunk/miglib/ResourceGet?type=log&filename=SOC1_24.log



http://localhost:8888/gcetrunk/miglib/ResourceGet?type=res&filename=SOC1_24.pdf



Les paramètres utilisables par cette servlet sont les suivants :

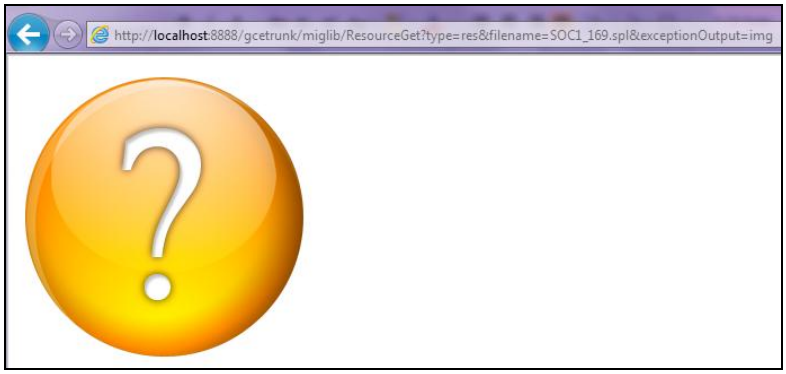
- filename : le nom du fichier (ou son préfixe en mode « list »)
- inline : le mode d'affichage pour le poste utilisateur, par défaut vaut *true*. Si le mode vaut false, il sera proposé à l'utilisateur s'il veut télécharger le fichier ou l'ouvrir directement (en dehors du navigateur donc).

A noter : ceci est possible grâce au paramètre de réponse http *Content-Disposition*

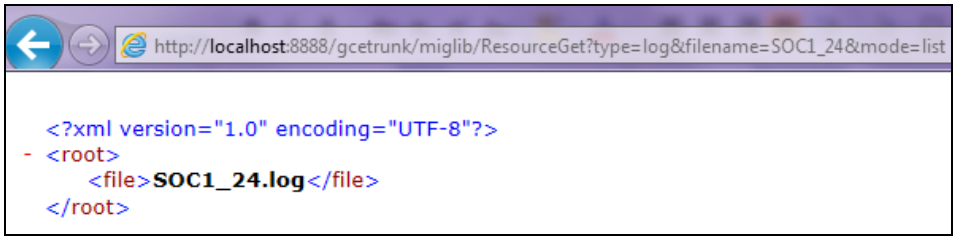
- type : un type défini dans gce.properties
- mode : *list* ou *exist* ou toute valeur (plus simple : pas ce paramètre sur l'URL), cf les exemples plus bas pour ces cas particuliers d'utilisation de cette servlet
- exceptionOutput : *img* ou *xml* (valeur par défaut) pour définir la sortie de la servlet en cas d'erreur

Remarque : en cas d'erreur, si le fichier ne peut pas être récupéré sur le FS (car non existant par exemple), par défaut on renvoie le message suivant :

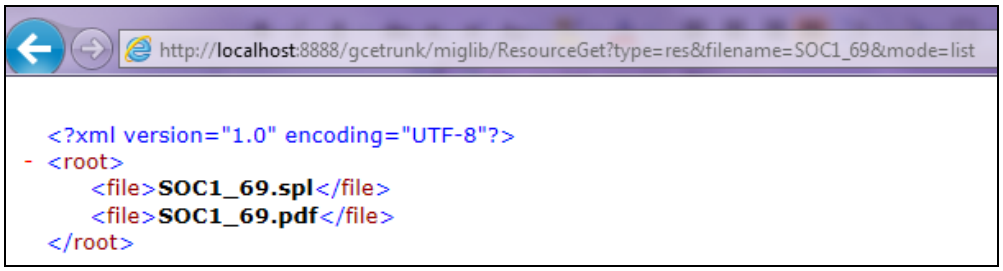
Si le paramètre exceptionOutput est positionné à img, on aura le retour suivant :



Exemple d'une liste fichiers commençant par :
http://localhost:8888/gcetrunk/miglib/ResourceGet?type=log&filename=SOC1_24&mode=list



Ou encore :
http://localhost:8888/gcetrunk/miglib/ResourceGet?type=res&filename=SOC1_69&mode=list



Exemple de test d'existence d'un fichier :
http://localhost:8888/gcetrunk/miglib/ResourceGet?type=res&filename=SOC1_69.spl&mode=exist

Et s'il n'existe pas :

Remarque : les modes list et exist sont prévus pour réaliser des tests de bonne mise en pratique de ce nouveau mécanisme permettant de récupérer des fichiers d'un FS vers le poste client (navigateur), ainsi que potentiellement pour être exploité via l'API XMLHttpRequest disponible en Javascript (communément appelée Ajax).

3.3 Divers

Les retours XML de cette servlet se font systématiquement en UTF-8.

Cette servlet nécessite d'avoir une application connectée, à savoir il faut que le navigateur envoie le cookie UUID à cette servlet pour qu'elle fonctionne correctement.

Le type MIME est déterminé à partir de l'extension du fichier, la définition se fait au niveau de web.xml. Si le type de contenu ne peut pas être déterminé, c'est *application/octet-stream* qui est renvoyé dans la réponse http. Dans certains cas de figure, le navigateur parvient tout de même à afficher correctement le résultat, mais la bonne méthode consiste tout de même à déclarer les bons mappings dans web.xml.

Par défaut, le contenu est renvoyé « gzippé » afin de limiter la consommation de la bande passante réseau. Néanmoins ce mode peut être désactivé via un paramètre -D de démarrage de la JVM, ou encore un paramétrage dans gce.properties (section systemProperties).
Nom du paramètre : *egx.servlet.resourceget.gzip* à passer à false

4 Génération d'URL « tiny »

Il s'agit de générer une URL qui masque la complexité relative des paramètres passés sur une URL via GCE. L'objectif est de stocker dans une nouvelle table ces URL afin de rediriger sur la « vrai » URL.

Ce mécanisme est utilisé sur le WEB, en particulier par des réseaux sociaux type Twitter car cela raccourci considérablement les URL. Dans notre cas, l'objectif est plus de fournir une sortie de raccourci d'URL que l'on envoie par mail.

A ce titre, il existe donc une nouvelle servlet qui permet de générer ces URL, elle permet aussi de rediriger vers l'application GCE dans un mode « forward ».

4.1 Définition

Comme toute nouvelle servlet, elle est doit être déclarée dans web.xml

```
<servlet>
	<display-name>ServletTiny</display-name>
	<servlet-name>ServletTiny</servlet-name>
	<servlet-
class>fr.generix.technicalframework.application.servlet.ServletTiny</servlet-class>
	<load-on-startup>-1</load-on-startup>
	<security-role-ref>
		<description>Global access role required</description>
		<role-name>sr_gce_overall</role-name>
		<role-link>sr_gce_overall</role-link>
	</security-role-ref>
</servlet>
```

```
<servlet-mapping>
	<servlet-name>ServletTiny</servlet-name>
	<url-pattern>/ServletTiny</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>ServletTiny</servlet-name>
	<url-pattern>/GCE/ServletTiny</url-pattern>
</servlet-mapping>
```

4.2 Table : schéma

C'est table très particulièrement, il est conseillé de la créer en IOT via le script SQL suivant :

```
create table tiny_url (tiny_id varchar2(100), request_url varchar2(4000), uticod
varchar2(8), description varchar2(256), datper varchar2(8),
CONSTRAINT tiny_url_idx1 primary key (tiny_id) enable)
ORGANIZATION INDEX NOCOMPRESS
PCTFREE 10 INITRANS 2 MAXTRANS 255
NOLOGGING
STORAGE
(
       INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0
FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT
       CELL_FLASH_CACHE DEFAULT
)
OVERFLOW;
```

Bien entendu le tablespace de création est à adapter sur site.

4.3 Utilisation

http://localhost:8888/gcetrunk/miglib/ServletTiny?tinyid=NzZiZjgwOTItNjA2My00MjE4LWFhYjQtODk3NmE2MGRmOTRkfjA=

5 Système d'auto-purge

Ce mécanisme est piloté depuis gce.properties, il permet de purger certaines tables utilisées par le socle technique lorsque les enregistrements sont « désuets ».

Les tables concernées sont les suivantes :

- UT_UUID : persistance de l'identification des sessions (par compte utilisateur : UT_LOGIN.USERNAME). Un cookie UUID est créé, sa date de péremption est lié au paramétrage dans le fichier de configuration, par défaut sa valeur est de 10000s à compter de la dernière URL de l'utilisation. Ce paramètre va de paire avec defaultUserRecognize (activé par défaut depuis toujours dans GCE)
- TIMER_REQUEST : les timers d'URL
- TIMER_WS : les timers pour les WS
- UT_DMA : la table de monitoring de l'application GCE
- UT_PRG : la table permettant de recharger « en live » certains paramétrages techniques ou fonctionnels

Des informations plus précises pourront être trouvées sur des documents annexes à celui-ci concernant le monitoring et les nouveaux timers.

5.1 Paramétrage

Il est proposé en standard dans gce.properties :

```
<autoPurge active="true">
       <delay value="14400" comment="purge toutes les 4h"/>
       <day value="2" comment="suppression de tous les éléments datant de plus de 2
jours"/>
</autoPurge>
```

Il est fortement conseillé de ne pas le désactiver ! Suivant les besoins, on pourra vouloir conserver un historique sur les tables décrites plus haut de plus de 2 jours.

5.2 Vision base de données

Il est conseillé de ne pas calculer les statistiques Oracle sur ces tables là car elles doivent normalement être à volumétrie constante, qui plus est leur accès se faut toujours sur la PK (primary key : index *_IDX1) d'un point de vue applicatif. La suppression peut provoquer un « full » sur la table concernée, mais la volumétrie étant faible, il n'est normalement pas nécessaire de créer d'index supplémentaire.

6 Activité JVM

Il s'agit d'une nouvelle servlet : ServletService qui permet de disposer d'un certain nombre d'informations liées à l'infrastructure (JVM / AS). Elle reprend des informations stockées en base sur les tables UT_DMA, UT_UUID et TIMER_REQUEST afin de fournir une vue d'ensemble des connexions et de l'infrastructure démarrée sur une base de données.

Le détail est fourni dans le GMO de Monitoring de la GCE161 (document à part).

7 Envoi de mail en cas d'erreur

Il est dorénavant possible de paramétrer un compte de messagerie afin d'envoyer un mail dans les cas suivants :

- Démarrage de la JVM
- Arrêt de la JVM
- Erreur grave (toute erreur non fonctionnelle)

7.1 Paramétrage

Comme d'habitude, le paramétrage se fait dans gce.properties :

```
<MailSettings active="true">
	<account admin="GENERIX Group in the sky&lt;tf_request@generixgroup.com&gt;"
user="moncompte@generixgroup.com" port="25" password="xxx"
host="toto.generixgroup.com" auth="true" tls="false"/>
	<startup active="false">
		<mailfrom>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailfrom>
		<mailto>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailto>
		<mailcci>arocher@generixgroup.com</mailcci>
		<filename name="oc4j.log" maxSize="10000000"/>
		<filename name="../../logs/server.log"/>
		<appenderName name="egxAppender"/>
	</startup>
	<shutdown active="false">
		<mailfrom>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailfrom>
		<mailto>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailto>
		<mailcci>arocher@generixgroup.com</mailcci>
	</shutdown>
	<error active="true">
		<mailfrom>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailfrom>
		<mailto>GENERIX Group in the
sky&lt;tf_request@generixgroup.com&gt;</mailto>
		<mailcc>jzemmouri@generixgroup.com</mailcc>
	<mailcci>arocher@generixgroup.com,fcarton@generixgroup.com,gprince@generixgro
up.com</mailcci>
		<filename name="oc4j.log"/>
		<filename name="../../logs/server.log"/>
		<appenderName name="egxAppender" maxSize="10000000"/>
		<appenderName name="bc4jAppender" maxSize="10000000"/>
		<appenderName name="activiteAppender" maxSize="10000000"/>
	</error>
</MailSettings>
```

Les sections sont activables/désactivables via l'attribut *@active*.

Important : il s'agit ici d'un exemple, merci de ne pas me polluer dès que vous avez une erreur ☺

7.2 Possibilités

Le compte de messagerie doit être paramétré de la manière suivante :

- Utilisateur / mot de passe
- Adresse du serveur
- Port SMTP
- Compte authentifié (true/false)
- TLS (true/false)

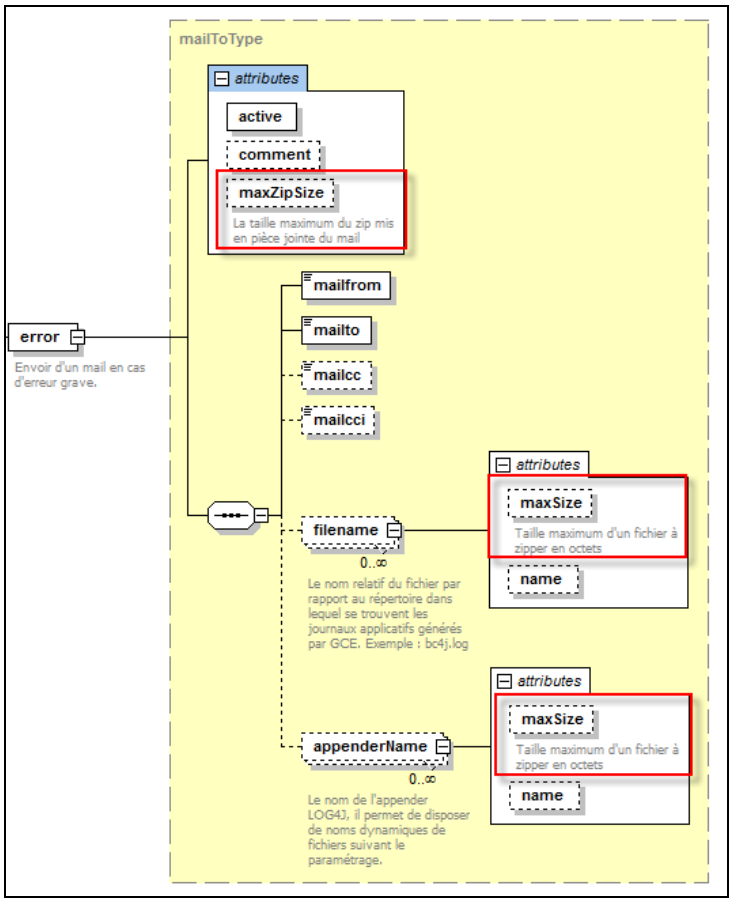
Exemple :

```
      <account admin="GENERIX Group in the sky&lt;tf_request@generixgroup.com&gt;"
user="moncompte@generixgroup.com" port="25" password="xxx"
host="toto.generixgroup.com" auth="true" tls="false"/>
```

On a la possibilité de lier des pièces jointes différentes pour chaque section, elles sont de 2 types :

- Soit un fichier
- Soit un appender log4j, ce qui permet d'utiliser des noms de fichiers « dynamiques » paramétrés dans log4j

Le schéma XSD définit des limites de tailles sur les pièces jointes afin de ne pas encombrer la messagerie :



Par défaut, la taille d'un fichier ne doit pas dépasser 10 000 000 octets (avant zip), le zip lui-même ne doit pas dépasser 10 000 000 octets. En fonction des possibilités du serveur de messagerie, il conviendra de paramétrer cela correctement sur site.

7.3 Restrictions

Les tests ont été réalisés avec les types de messageries suivantes :

- Orange
- Gmail
- Exchange 2010+

Il est possible de prendre en compte d'autres serveurs de messagerie, dans ce cas il s'agira d'un développement spécifique lié à facturation.

Il n'est pas prévu de pouvoir utiliser des serveurs de messagerie à protocole sécurisé.

7.4 Divers

Si la JVM ne peut pas démarrer car gce.properties ne peut pas être lu (fichier non conforme au XSD par exemple), aucun mail ne pourra être envoyé.

Les mails sont envoyés dans un Thread créé spécialement à cet effet afin de ne pas polluer le fonctionnement normalement de l'application. En cas d'erreur sur l'envoi d'un mail, on pourra trouver des traces dans egx.log, comme habituellement.

8 Nouvelles servlets

La plupart des nouvelles servlets nécessitent une connexion à la base de données, ces servlets étant indépendantes de ServletControl, la connexion bdd utilisée ne sera pas celle de l'utilisateur.

Pour cette raison, il est fortement conseillé de définir ses connexions via des **DataSources poolées**. Nous avons l'exemple d'un client qui est passé de 1600 connexions à la base en simultané à une valeur plus raisonnable de 400. Cela apporte une consommation réduite de la même sur le serveur de base de données, en particulier au niveau des « process shadow ».

Néanmoins, le socle propose un système de pooling paramétrable suivant quelques critères. Un nouvel acronyme fait son apparition, DCM=Database Connexion Manager. Comme habituellement, tout est défini dans gce.properties :

```
<DCM>
      <connection type="pooled" maxPoolSize="2" idleTime="30"/>
</DCM>
```

Si la section n'est pas définie dans gce.properties, on prend les valeurs par défaut suivantes :

- type=native
- maxPoolSize=1
- idleTime=60

Le type permet de préciser si la DataSource est de type poolée ou non. Dans le cas d'une DS poolée, on désactive le mécanisme socle : pas la peine de pooler 2 fois (et donc les 2 attributs suivants ne sont pas utilisés).

Dans le cas contraire (DataSource dite « native » ou connexion JDBC directe), il faut passer le type à « native ». De cette manière c'est le socle qui poolera les connexions JDBC en fonction des 2 autres paramètres. Sachant qu'il s'agit d'un pool à débordement, la valeur « 2 » est très certainement un bon compromis. L'attribut idleTime définit le temps au-delà duquel la connexion sera relâchée car pas utilisée, ce temps est en secondes.

En résumant, en fonction de la source de données paramétrée au niveau du serveur d'application :

- Si DS native ➔ type DCM=native
- Si Connexion JDBC directe ➔ type DCM=native
- Si DS poolée ➔ type DCM=pooled

On notera que la mise en place de ce mécanisme est liée au fait qu'on ne peut pas connaitre automatiquement (au niveau socle technique) le type de DataSource paramétré dans le serveur d'application.
