Il est possible dans EzGED de créer vos propres scripts python pour des besoins spécifiques. Vous pourrez ainsi créer une nouvelle étape de COLD ou un travail cyclique afin de réaliser des opérations non couvertes par les scripts existants.

Templates

Travail cyclique

monscript.py

```
#!/usr/bin/python
# -*- coding: ISO-8859-15 -*-
import traceback
import sys
import _common
import common
import db_common

def execscript(params=None,edb=None):
    """
    La fonction execscript est la fonction principale (comme une
fonction main en C par exemple)
    qui sera appelée par le serveur des travaux afin d'exécuter le
script.

    @param params: la liste des arguments fournis au script.
    @type params: list

    @param edb: Une instance de connexion à la base de données.
    @type edb: instance
    """
    parameters = {}

    if params is None:
        #Dans le cas d'un appel en ligne de commande on récupère les
arguments qui ont été fournies via cette ligne de commande.
        params = sys.argv

    #Ci-dessous on récupère dans un dictionnaire python les arguments
passés au script.
    ret = _common.getscriptparameters(parameters, params,
"instance","jobqueueid","secusrid")

    if ret <> 0 :
        return ret, str(ret)

    gses, vjobdir, dbtype =
libjobdext.get_job_script_session(parameters, exec_session)
```

```
    if edb is None:
        db_common.sql_dbcontext(gses)
        db=gses.db
    else:
        gses.db=edb
        db=edb

    if db == None:
        return -1, gses.lng("Unable to connect to database")

    else:
        #Dans ce bloc on va écrire toute la logique de notre script.
        print "Hello, World!"

    if edb is None:
        #Le script a été appelé sans lui fournir une connexion à la
bdd.
        #Dans ce cas on ferme la connexion que nous avons ouverte au
sein du script.
        db.disconnect()

    return 0,"ok"

if __name__ == "__main__":
    print sys.argv
    try:
        errorcode,errorstr=execscript()
    except:
        print "@error"
        traceback.print_exc(file=sys.stdout)
        sys.exit(-100)

    if errorcode <> 0:
        print "@error",errorstr
    sys.exit(errorcode)
```

Etape COLD

Le canevas pour une étape COLD est sensiblement identique à quelques lignes près

monetapecold.py

```
#!/usr/bin/python
# -*- coding: ISO-8859-15 -*-
import traceback
import sys
import _common
import common
```

```
import db_common
from coldtrt import coldscript

class MONETAPE(coldscript):
    def __init__(self):
        coldscript.__init__(self)
        self.typeIn = ("PDF",) #Liste des flux en entrée
        self.typeOut = ("PDF",)#Liste des flux en sortie


def execscript(params=None,edb=None):
    """
    La fonction execscript est la fonction principale (comme une
fonction main en C par exemple)
    qui sera appelée par le serveur des travaux afin d'exécuter le
script.

    @param params: la liste des arguments fournis au script.
    @type params: list

    @param edb: Une instance de connexion à la base de données.
    @type edb: instance
    """
    parameters = {}

    if params is not None:
        #Dans le cas d'un appel en ligne de commande on récupère les
arguments qui ont été fournies via cette ligne de commande.
        sys.argv = params

    #Ci-dessous on récupère dans un dictionnaire python les arguments
passés au script.
    ret = _common.getscriptparameters(parameters, sys.argv,
"instance","jobqueueid","secusrid")

    if ret <> 0 :
        return ret, str(ret)

    gses, vjobdir, dbtype =
libjobdext.get_job_script_session(parameters, exec_session)
    if edb is None:
        db_common.sql_dbcontext(gses)
        db=gses.db
    else:
        gses.db=edb
        db=edb

    if db == None:
        return -1, gses.lng("Unable to connect to database")

    else:
```

```
        #Dans ce bloc on va écrire toute la logique de notre script.
        print "Quelque chose"

    if edb is None:
        #Le script a été appelé sans lui fournir une connexion à la
bdd.
        #Dans ce cas on ferme la connexion que nous avons ouverte au
sein du script.
        db.disconnect()

    return 0,"ok"

if __name__ == "__main__":
    print sys.argv
    try:
        errorcode,errorstr=execscript()
    except:
        print "@error"
        traceback.print_exc(file=sys.stdout)
        sys.exit(-100)

    if errorcode <> 0:
        print "@error",errorstr
    sys.exit(errorcode)
```

Ajouter une étape de référence

Via interface

Importer le script via l'Administration → Travaux → Travaux de référence → Etapes de référence

Manuellement

Placer votre script dans le dossier nchp > bin > et le déclarer dans les fichiers d'ezGED

Dossier :
C:\nchp\usr\local\nchp\ezged\bin

Dans le fichier db_data.py, la variable jobsteptpl contient tous les travaux enregistrés.

ID
NAME
DESC
SHARED

TYPE
CMD
MAX