



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Administrator Extension Help

Utilisation de la console de ligne de commande (CLC)

Utilisation de la console de ligne de commande (CLC)

Sommaire

- **1 Utilisation de la console de ligne de commande (CLC)**
 - 1.1 Structure
 - 1.2 SPD (Définitions de solution)
 - 1.3 Packages d'installation

La console de ligne de commande (CLC) permet aux administrateurs d'utiliser la ligne de commande pour exécuter certaines fonctions GAX sur les **définitions de solution (SPD)** et les **packages d'installation (IP)**. Par exemple, vous pouvez utiliser la CLC pour déployer les SPD silencieusement sur les hôtes distants.

Vous devez pouvoir accéder à l'interface de ligne de commande du système d'exploitation pour utiliser la CLC. Si vous n'utilisez pas l'ordinateur hôte GAX, l'outil CLC (**gaxclc.jar**) doit être disponible sur l'ordinateur local.

Pour accéder au fichier d'aide intégré de la CLC, exécutez l'une des commandes suivantes :

```
java -jar gaxclc.jar help
java -jar gaxclc.jar ?
```

Important

Lorsque vous exécutez les commandes avec la CLC, un fichier journal est généré à l'emplacement d'exécution de l'outil.

Structure

La CLC prend en charge les commandes qui utilisent la structure suivante :

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> <function> <operation> <args>
```

Dans l'exemple ci-dessus :

- -u:user correspond au nom de l'utilisateur utilisé pour se connecter à Configuration Server.
- -p:password correspond au mot de passe utilisé pour se connecter à Configuration Server. La CLC suppose qu'il n'y a pas de mot de passe si aucune valeur n'est précisée pour cet indicateur.
- -s demande à la CLC d'utiliser une connexion *https* sécurisée pour accéder au serveur GAX. Si cet indicateur n'est pas précisé, la CLC utilise *http*.
- -h:<host>:<port> spécifie l'hôte et le port du serveur GAX. Si cet indicateur n'est pas précisé, la CLC utilise la valeur suivante : -h:localhost:8080.
- <function> peut être ip ou spd.
- <operation> spécifie l'action à exécuter. Les valeurs valides de cet indicateur sont spécifiques à la fonction indiquée lors de l'étape précédente (ip ou spd).
- <args> spécifie les arguments d'action. Les valeurs valides de cet indicateur sont spécifiques aux paramètres <function> et <operation> définis lors des étapes précédentes.

Voici un exemple de commande CLC :

```
java -jar gaxclc.jar -u:default -p:password -h:localhost:8080 spd execute 10054 1 "C:/GAX/input.txt"
```

SPD (Définitions de solution)

La CLC prend en charge les actions suivantes concernant les SPD :

- add
- query
- querybyid
- execute
- delete
- encrypt (voir l'onglet execute)

add

add

Présentation

Cette action ajoute une SPD à la base de données GAX. Si la définition de solution existe déjà, et se termine par le nom et la version dans le XML de la SPD, cette action remplace la SPD existante.

En cas de réussite, l'action renvoie l'ID de la SPD ajoutée.

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd add "file path"
```

- "file path"—Chemin d'accès au fichier XML.

Exemple

```
java -jar gaxclc.jar -u:default -p:password spd add "c:\GAX\newSpd.xml"
```

query

query

Présentation

Cette action interroge toutes les SPD et affiche un tableau qui répertorie les détails suivants relatifs à chaque SPD :

- Numéro d'ID
- Nom
- Version
- ID base de données du locataire

Voici un exemple :

```
10054 gvp 8.1.5 1
10060 genesysOne 8.1.5 1
10060 eServices 8.1.5 1
```

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd query
```

Exemple

```
java -jar gaxclc.jar -u:default -p:password -s -h:132.45.43.45:443 spd query
```

querybyid

querybyid

Présentation

Cette action interroge une SPD à l'aide de son ID. Si la SPD n'existe pas, l'action échoue.

En cas de réussite, l'action affiche un tableau qui répertorie les détails suivants sur la SPD :

- ID du profil
- Nom

Par exemple :

```
1 Install
```

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd query SPDID
```

- SPDID—ID de la SPD interrogée.

Exemple

```
java -jar gaxclc.jar -u:default -p:password -h:132.45.43.45:8080 spd query 4374
```

execute

execute

Présentation

Cette action exécute une SPD.

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd execute SPDID profileName|  
-profileID:profileID|-profileName:profileName -encrypted "input file"
```

- SPDID—ID de la SPD à exécuter.
- profileName|-profileID:profileID|-profileName:profileName—Le profil SPD à exécuter.

Important

Si aucun indicateur n'est spécifié, profileName est considéré comme étant le profil de SPD à exécuter.

- -encrypted—Si spécifié, indique si le fichier d'entrée est chiffré.

[+] Afficher l'utilisation

La CLC prend en charge le chiffrement pour les fichiers d'entrée qui comprennent des données sensibles comme les mots de passe.

Format :

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd encrypt "input file path"  
"encrypted output file path"
```

Le fichier d'entrée chiffré est conservé à l'emplacement spécifié par "encrypted output file path". Si le fichier existe déjà à cet emplacement, il est écrasé.

Exemple :

```
java -jar gaxclc.jar -u:default -p:password spd -encrypted "c:\GAX\input.txt" "c:\GAX\encrypted.txt"
```

```
java -jar gaxclc.jar -u:default -p:password spd -encrypted "input.txt" "encrypted.txt"
```

- "input file"—Spécifie le fichier d'entrée qui contient les paramètres de SPD. Si -encrypted est défini, le fichier d'entrée est chiffré.

Le fichier d'entrée doit être au format JSONObject et comprendre les paramètres de SPD d'un profil spécifique. Le fichier doit être codé au format UTF-8.

[+] Afficher l'utilisation

Type CHAINE

La structure d'entrée du type *chaîne* est décrite ci-après :

```
{
  "Dialog name" : {
    "Input name" : "string"
  }
}
```

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Step1">
    <input name="NAME_PARAM1" title="PERSON NAME" default="birit" type="string"
required="true">
      <description>Please enter the person name</description>
    </input>
  </dialog>
  <dialog step="Step2">
    <input name="NAME_PARAM2" title="PERSON NAME" default="birit" type="string"
required="true">
      <description>Please enter the person name</description>
    </input>
  </dialog>
  <execution>
    <script>
      log('string test' );
    </script>
  </execution>
</profile>
```

Fichier d'entrée du profil d'installation

```
{
  "Step1" : {
```

```
        "NAME_PARAM1" : "Kate"
    },
    "Step2" : {
        "NAME_PARAM2" : "John"
    }
}
```

Type BOOLEEN

La structure d'entrée du type *booléen* est décrite ci-après :

```
{
    "Dialog name" : {
        "Input name" : true/false
    }
}
```

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Step1">
    <input name="STATUS" title="status" type="boolean" required="true">
      <description>status field</description>
    </input>
  </dialog>
  <execution>
    <script>
      log('boolean test');
    </script>
  </execution>
</profile>
```

Fichier d'entrée du profil d'installation

```
{
    "Step1" : {
        "STATUS" : true
    }
}
```

Type ENTIER

La structure d'entrée du type *entier* est décrite ci-après :

```
{
```

```
    "Dialog name" : {
      "Input name" : <integer>
    }
  }
```

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Step1">
    <input name="NUMBER" title="number" type="integer" required="true">
      <description>number field</description>
    </input>
  </dialog>
  <execution>
    <script>
      log('number test');
    </script>
  </execution>
</profile>
```

Fichier d'entrée du profil d'installation

```
{
  "Step1" : {
    "NUMBER" : 132
  }
}
```

Type MOT DE PASSE

La structure d'entrée du type *mot de passe* est décrite ci-après :

```
{
  "Dialog name" : {
    "Input name" : "password"
  }
}
```

Important

Les fichiers d'entrée comprenant des données sensibles comme des mots de passe doivent être chiffrés à l'aide d'une opération de chiffrement SPD.

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Step1">
    <input name="PASSWORD" title="password" type="password" required="true">
      <description>password field</description>
    </input>
  </dialog>
  <execution>
    <script>
      log('password test');
    </script>
  </execution>
</profile>
```

Fichier d'entrée du profil d'installation

```
{
  "Step1" : {
    "PASSWORD" : "xyz9846gdkjg"
  }
}
```

Type SELECTION UNIQUE

La structure d'entrée du type *sélection unique* avec une balise **<objectselect>** est décrite ci-après :

```
{
  "Dialog name" : {
    "Input name" : {
      "objectselect" : {
        "filter" : [{
          "value" : "filter value",
          "name" : "filter name"
        }
      ]
    }
  }
}
```

Important

CLC croise les filtres (ET) définis dans le fichier SPD et le fichier d'entrée de l'entrée

sélection unique. Les critères de filtre doivent être différents dans le fichier d'entrée SPD et les noms des filtres doivent être différents dans la même définition de filtre.

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Step1">
    <input name="APP_OBJ_SELECT_ONE" title="Application Name" hidden="false"
type="selectOne" default="">
      <description>select application</description>
      <objectselect>
        <filter value="CfgApplication" name="type"/>
      </objectselect>
    </input>
  </dialog>
  <execution>
    <script>
      log('test select one' );
    </script>
  </execution>
```

Fichier d'entrée du profil d'installation

```
{
  "Step1" : {
    "APP_OBJ_SELECT_ONE" : {
      "objectselect" : {
        "filter" : [{
          "value" : "SIP_lrm26",
          "name" : "name"
        }
      ]
    }
  }
}
```

Type SELECTION MULTIPLE

La structure d'entrée du type *sélection multiple* avec une balise **<objectselect>** est décrite ci-après :

```
{
  "Dialog name" : {
    "Input name" : {
```

```

        "objectselect" : {
            "filter" : [{
                "value" : "filter value",
                "name" : "filter name"
            }
        ]
    }
}

```

Les filtres définis dans un fichier d'entrée SPD sont associés (OU) puis croisés (ET) avec les filtres définis dans un fichier SPD d'une entrée de type *sélection multiple*.

Exemple

Profil SPD

```

<profile name="Install">
  <dialog step="Step1">
    <input name="APP_OBJ_SELECT_MULTIPLE" title="Application Name" hidden="false"
type="selectMultiple" default="">
      <description>select application</description>
      <objectselect>
        <filter value="CfgApplication" name="type"/>
      </objectselect>
    </input>
  </dialog>
  <execution>
    <script>
      log('test select multiple' );
    </script>
  </execution>

```

Fichier d'entrée du profil d'installation

```

{
  "Step1" : {
    "APP_OBJ_SELECT_MULTIPLE" : {
      "objectselect" : {
        "filter" : [{
            "value" : "SIP_lrm26",
            "name" : "name"
          }, {
            "value" : "SIP_lrm27",
            "name" : "name"
          }
        ]
      }
    }
  }
}

```

L'opération renvoie deux applications nommées **SIP_Irm26** et **SIP_Irm27**.

Type SELECTION UNIQUE

La structure d'entrée du type *sélection unique/sélection multiple/booléen* avec une balise **<selection>** est décrite ci-après :

```
{
  "Dialog name" : {
    "Input name" : {
      "selection" : {
        "option" : [{
parameter",
          "value" : "option value assigned to the input
          "name" : "option name is displayed in UI"
        }
      ]
    }
  }
}
```

CLC sélectionne les options définies dans le fichier d'entrée SPD. Il n'est possible de spécifier plusieurs options que pour le type d'entrée *sélection multiple*.

Exemple

Profil SPD

```
<profile name="Install">
  <dialog step="Application Parameters">
    <input name="DATA_MODEL" title="Binary Version (32-bit or 64-bit)" default="64"
type="selectOne" required="true">
      <description>This parameter defines the 32-bit or the 64-bit version of the
binary to be deployed. </description>
      <selection>
        <option name="32" value="32"/>
        <option name="64" value="64"/>
      </selection>
    </input>
  </dialog>
</execution>
  <script>
    log('test selection support' );
  </script>
</execution>
```

Fichier d'entrée du profil d'installation

```
{
  "Application Parameters" : {
    "DATA_MODEL" : {
      "selection" : {
        "option" : [{
          "value" : "64",
          "name" : "64"
        }]
      }
    }
  }
}
```

Important

- Si le fichier d'entrée ne contient pas de valeur pour un paramètre de SPD, la valeur définie dans l'attribut **par défaut** de l'élément d'entrée est utilisée.
- Si l'attribut **requis** d'un élément d'entrée de la définition de solution est défini sur vrai en l'absence de valeur d'entrée correspondante dans la SPD (valeur par défaut) ou dans le fichier d'entrée, l'exécution de la définition de solution échoue.
- Si la valeur de l'attribut **lecture seule** d'un élément d'entrée de SPD est défini sur vrai, la valeur de l'attribut **par défaut** est utilisée pour l'exécution si elle est définie. Si la valeur de l'attribut **lecture seule** est définie sur vrai, la valeur **requis** est définie sur faux et que l'attribut **par défaut** n'est pas défini, la logique suivante est appliquée pour déterminer la valeur d'entrée :
 1. Avec un type d'entrée *booléen*, la valeur d'entrée est définie sur faux.
 2. Avec les types d'entrée *chaîne* et *mot de passe*, la valeur d'entrée est définie sur "".
 3. Avec le type d'entrée *entier*, l'entrée n'est pas propagée.
- Si la valeur de l'attribut **cond** d'une boîte de dialogue est définie sur faux, cette boîte de dialogue est ignorée par l'outil CLC. Exemple :

```
<dialog step="Role input" cond="false">
  <input name="ROLE" title="Role" hidden="false" type="selectOne"
  required="true">
    <description>Please indicate the role</description>
    <objectselect>
      <filter value="CfgRole" name="type"/>
    </objectselect>
  </input>
</dialog>
```

Exemple

```
java -jar gaxclc.jar -u:default -p:password -s -h:localhost:8080 spd execute 10054  
-profileID:1 "C:/GAX/input.txt"
```

```
java -jar gaxclc.jar -u:default -p:password -h:localhost:8080 spd execute 10054  
-profileName:"Install profile" "C:/GAX/input.txt"
```

```
java -jar gaxclc.jar -u:default -p:password -s -h:localhost:8080 spd execute 10054 1  
-encrypted "C:/GAX/encryptedinput.txt"
```

delete

delete

Présentation

Cette action supprime une SPD. Si la SPD n'existe pas, l'action échoue.

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> spd delete SPDID
```

- SPDID—ID de la SPD à supprimer.

Exemple

```
java -jar gaxclc.jar -u:default -p:password spd delete 5436
```

Packages d'installation

La CLC prend en charge les actions suivantes concernant la fonction ip :

- add
- query
- querybyid
- delete

add

add

Présentation

Cette action ajoute un package d'installation (sous la forme d'un fichier .zip) à la base de données GAX. Si le package d'installation existe déjà, il est remplacé.

En cas de réussite, l'action affiche l'ID du package d'installation.

Important

Le fichier .zip doit contenir le package d'installation et le dossier de modèles du package d'installation.

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> ip add "path to IP zip file"
```

Exemple

```
java -jar gaxclc.jar -u:default -p:password ip add "C:\GAX\TESTS\zippedIpUpload\PRODUCTION\IP_TSrvSIP64_18100079b1_ENU_windows.zip"
```

query

query

Présentation

Cette action interroge tous les packages d'installation et affiche un tableau qui répertorie les détails suivants sur chaque package d'installation :

- Numéro d'ID
- Nom

- Version
- SE
- Paramètres régionaux
- Etat

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> ip query
```

Exemple

```
java -jar gaxclc.jar -u:default -p:password -s -h:132.45.43.45:443 ip query
```

querybyid

querybyid

Présentation

Cette action interroge un package d'installation à l'aide de son ID et affiche un tableau qui répertorie les détails suivants :

- Numéro d'ID
- Nom
- Version
- SE
- Paramètres régionaux
- Etat

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> ip query IPID
```

- IPID—ID du package d'installation à interroger.

Exemple

```
java -jar gaxclc.jar -u:default -p:password -h:132.45.43.45:8080 ip query 543
```

delete

delete

Présentation

Cette action supprime un package d'installation.

Format

```
java -jar gaxclc.jar -u:user -p:password -s -h:<host>:<port> ip delete IPID
```

- IPID—ID du package d'installation à supprimer.

Exemple

```
java -jar gaxclc.jar -u:default -p:password ip delete 547
```