Genesys Administrator Extension Help

Utilizzo della console a riga di comando (CLC, Command Line Console)

4/4/2025

Utilizzo della console a riga di comando (CLC, Command Line Console)

Indice

La console a riga di comando (CLC) consente agli amministratori di utilizzare la riga di comando per eseguire alcune funzioni GAX su definizioni di soluzioni (SPD) e pacchetti di installazione (IP). Ad esempio, la console CLC potrebbe essere utilizzata per distribuire automaticamente SPD su host remoti.

Per utilizzare la console CLC è necessario poter accedere all'interfaccia della riga di comando del sistema. Se non si utilizza il computer host GAX, è necessario che lo strumento CLC (**gaxclc.jar**) sia disponibile sul computer locale.

Per accedere al file della Guida incorporato nello strumento CLC, eseguire uno dei comandi seguenti:

```
java –jar gaxclc.jar help

java –jar gaxclc.jar ?
```

> ### Importante
> Quando si eseguono comandi con CLC, viene generato un file di registro nella stessa posizione in cui viene eseguito lo strumento.

Struttura

CLC supporta comandi che utilizzano la seguente struttura:

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

Nell'esempio precedente:

- `-u:user` è il nome utente per accedere al server di configurazione.
- `-p:password` è la password per accedere al server di configurazione. CLC presume che non vi sia alcuna password se questo flag non specifica un valore.
- `-s` indica a CLC di utilizzare una connessione *https* protetta al server GAX. Se questo flag non è specificato, CLC utilizza *http*.
- `-h:<host>:<port>` specifica l'host e la porta del server GAX. Se questo flag non è specificato, CLC utilizza il valore seguente: `-h:localhost:8080`.
- `<function>` può essere `ip` o `spd`.
- `<operation>` specifica l'operazione da eseguire. I valori validi per questo flag sono specifici alla funzione indicata al punto precedente (`ip` o `spd`).
- `<args>` specifica gli argomenti dell'operazione. I valori validi per questo flag sono specifici ai parametri `<function>` e `<operation>` indicati ai punti precedenti.

Di seguito viene fornito un esempio di un comando CLC:

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

SPD

CLC supporta le operazioni seguenti per SPD:

- `add`
- `query`
- `querybyid`
- `execute`
- `delete`
- `encrypt` (vedere la scheda `execute`)

add

add

Descrizione

Questa operazione consente di aggiungere una SPD al database GAX. Se la SPD esiste già, come determinato dal nome e dalla versione nel file XML SPD, questa operazione sostituisce la SPD esistente.

Se viene completata correttamente, l'operazione restituisce l'ID della SPD aggiunta.

Formato

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

- `"file path"`—Percorso del file XML.

Esempio

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

query

query

Descrizione

Questa operazione esegue le query su tutte le SPD e visualizza una tabella con le informazioni seguenti per ogni SPD:

- Numero ID
- Nome
- Versione
- DBID tenant

Di seguito viene fornito un esempio:

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

Formato

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

Esempio

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

querybyid

querybyid

Descrizione

Questa operazione esegue query su una SPD mediante il relativo ID. Se la SPD non esiste, l'operazione non viene completata.

Se viene completata correttamente, l'operazione visualizza una tabella con le seguenti informazioni sulla SPD:

- ID profilo
- Nome

Ad esempio:

```
1 Install
```

Formato

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

- SPDID—L'ID della SPD sottoposta a query.

Esempio

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

execute

execute

Descrizione

Questa operazione consente di eseguire una SPD.

Formato

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

- SPDID—L'ID della SPD da eseguire.
- profileName|-profileID:profileID|-profileName:profileName—Il profilo SPD da eseguire.

> **Importante**
> Se non è specificato alcun flag, profileName viene considerato come il profilo SPD da eseguire.

- -encrypted—Se specificato, indica se il file di input è crittografato.

[+] Mostra uso

Lo strumento CLC supporta la crittografia per i file di input che includono dati sensibili come password.

Formato:

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

Il file di input crittografato si trova nel percorso specificato in "encrypted output file path". Se il file esiste già in questo percorso, viene sovrascritto.

Esempio:

```
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"—Specifica il file di input che contiene parametri SPD. Se si imposta -encrypted, il file di input viene crittografato.

Il file di input deve essere nel formato JSONObject e includere parametri SPD per un determinato profilo. Il file deve essere codificato in formato UTF-8.

[+] Mostra uso

The input file must be in JSONObject format and include SPD parameters for a specific profile. The file must be encoded in UTF-8 format.

string

The input structure for a *string* type is described below:

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

Example

SPD Profile

```
<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>
```

Input File for Install Profile

```
{
        "Step1" : {
                "NAME_PARAM1" : "Kate"
        },
        "Step2" : {
                "NAME_PARAM2" : "John"
        }
}
```

Boolean

The input structure for a *boolean* type is described below:

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

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

Integer

The input structure for an *integer* type is described below:

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

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

Password

The input structure for a *password* type is described below:

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

> **Importante**
>
> Input files that include sensitive data such as passwords should be encrypted using the SPD encrypt operation.

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

SelectOne

The input structure for a *selectOne* type with an **<objectselect>** tag is described bellow:

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

> **Importante**
>
> CLC intersects (*AND*) filters defined in the SPD file and input file for a *selectOne* input. The filter criteria should be different in an SPD input file and filter names should differ in the same filter definition.

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

SelectMultiple

The input structure for a *selectMultiple* type with **<objectselect>** tag is described below:

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

Filters defined in an SPD input file are joined in union (*OR*) and then intersect (*AND*) with filters defined in an SPD file for a *selectMultiple* input.

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

The operation returns two applications named **SIP_lrm26** and **SIP_lrm27**.

Selection Tag

The input structure for a *selectOne/selectMultiple/boolean* type with **<selection>** tag is described below:

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

CLC selects options defined in the SPD input file. Multiple options can be specified only for the *selectMultiple* input type.

Example

SPD Profile

```
<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>
```

Input File for Install Profile

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

Importante

- If the input file does not specify a value for a SPD parameter, the value defined in the **default** attribute of the input element will be used.

- If an SPD input element has the **required** attribute set to `true`, but there is no corresponding input value that is supplied in either the SPD (as a default) or in the input file, then the SPD execution fails.
- If an SPD input element has the **readonly** attribute value set to `true`, then the value in the **default** attribute value is used for the execution, if defined. If the **readonly** attribute value is set to true, **required** is set to `false`, and the **default** attribute is not defined, then the following logic is used for input value determination:
  1. For the *boolean* input type, the input value is set to `false`.
  2. For the *string* and *password* input types, the input value is set to `""`.
  3. For the *integer* input type, the input is not propagated.
- If a dialog **cond** attribute value evaluates to false, the dialog is skipped by the CLC tool. Example:

```
<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>
```

Esempio

```
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

Descrizione

Questa operazione elimina una SPD. Se la SPD non esiste, l'operazione non viene completata.

Formato

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

- SPDID—L'ID della SPD da eliminare.

Esempio

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

IP

CLC supporta le seguenti operazioni per la funzione `ip`:

- `add`
- `query`
- `querybyid`
- `delete`

add

add

Descrizione

Questa operazione consente di aggiungere un pacchetto di installazione (IP) (sotto forma di file .zip) al database GAX. Se il pacchetto di installazione esiste già, viene sostituito.

Se viene completata correttamente, l'operazione visualizza l'ID del pacchetto di installazione.

> **Importante**
> Il file .zip deve contenere il pacchetto di installazione e la cartella dei modelli per il

pacchetto di installazione.

Formato

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

Esempio

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

query

query

Descrizione

Questa operazione consente di eseguire le query su tutti i pacchetti di installazione e visualizza una tabella con le informazioni seguenti per ogni IP:

- Numero ID
- Nome
- Versione
- Sistema operativo
- Impostazioni internazionali
- Stato

Formato

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

Esempio

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

querybyid

querybyid

Descrizione

Questa operazione consente di eseguire le query su un pacchetto di installazione mediante il relativo ID e visualizza una tabella con le informazioni seguenti:

- Numero ID
- Nome
- Versione
- Sistema operativo
- Impostazioni internazionali
- Stato

Formato

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

- `IPID`—L'ID del pacchetto di installazione da sottoporre a query.

Esempio

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

delete

delete

Descrizione

Questa operazione elimina un IP.

Formato

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

- `IPID`—L'ID del pacchetto di installazione da eliminare.

Esempio

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