ORACLE ENTERPRISE MANAGER 12c
EM CLI CRASH COURSE
GÖKHAN ATIL
DBA TEAM LEADER
Gökhan Atil

- DBA Team Leader at bilyoner.com
- Oracle ACE (since 2011)
- Founding Member and Vice President of TROUG (troug.org)
- Co-author of Expert Oracle Enterprise Manager
- Blogger (since 2008) gokhanatil.com
- Active on twitter: @gokhanatil
Agenda

- Introduction to EM CLI
- EM CLI Verbs
- Formatting the Output
- Fetching Information From EM Repository
- Sample Bash Scripts
- EM CLI Interactive Mode and Scripting Mode
- Fundamentals of Python
- Sample Python Scripts
What is EM CLI?
What is EM CLI?

- Agent
- Repository
- OMS
- Web Console
- EM CLI
Why EM CLI?

- Emotional Reasons:
  - We love text-based tools

- Practical Reasons:
  - Sometimes it’s faster (less clicks)
  - Good for scripting (automatisation and bulk operations)

Web interface of Enterprise Manager is so cool!
Jump into the action!

It's already installed and configured on OMS server:

```bash
cd $OMSHOME/bin
./emcli (or emcli.bat on Windows servers)
```

```
oracle@db-cloud /$ cd $OMSHOME/bin
[oracle@db-cloud bin$ .] /emcli
Oracle Enterprise Manager 12c Release 5 EM CLI with Scripting option.  
Copyright (c) 1996, 2015 Oracle Corporation and/or its affiliates. All rights reserved.

Type help() for help and exit() to get out.

emcli>
```
You can install EM CLI to your PC

- Requires **Java** version 1.6.0_43 or greater
- Supported Operating Systems: Windows, Linux, Solaris, HPUX, Tru64, AIX
- Go to “command line interface” page under “setup” page to download the installation JAR file.
- Login is not required, so it’s possible to download the JAR file with “wget”.

To run EM CLI on Java 1.8 or higher, you need apply patch 17555224 to the weblogic server. Use "-no-check-certificate" parameter with wget.
Downloaded and ready to install!

- After you download, set JAVA_HOME and install the JAR file:
  
  java -jar emcliadvancedkit.jar -install_dir=<home>

  java -jar emclikit.jar -install_dir=<home>

- Configure EM CLI:

  emcli help setup

  emcli setup -url="https://host:port/em"
  -username=<emuser> -dir=<instancemdir> -trustall
EM CLI Directory Structure

- EM CLI configuration and log files (.emcli.log) are located in the “instance home” directory (remember the -dir parameter).

- Basic JAR files and bash scripts are stored in “installation directory”.

- After you run “setup”, required JAR files used by verbs will be fetched from OMS are stored under the **bindings** directory.
There are two different installation kits for EM CLI:

- **EM CLI Standard Kit:** This kit supports only the Standard (command line) mode.

- **EM CLI with Scripting mode (option):** This “advanced” kit supports **Interactive and Script** modes in addition to standard mode.
3 modes of EM CLI:

- Standard mode: This mode provides a simple command-line interface to Enterprise Manager, and supports the execution of one verb at a time from the command line.

  `emcli get_targets`

- Interactive mode: This mode enables you to create a single interactive session with OMS. It is a Jython shell.

  `emcli`

- Scripting mode: In Scripting mode, EM CLI runs Jython scripts using EM CLI verbs as Jython functions.

  `emcli @scriptname.py`
In standard mode, EM CLI expect you enter a verb as the first parameter. A verb is a task or action in the form of a user command which exposes Enterprise Manager functionality. Verbs may take zero or more arguments as input.

```
emcli verb 1st_argument
[ -2nd_ argument="value" ]
[ -3rd_ argument="value" ]
[ -4th_ argument ]
...
```
A less known feature

emcli setup -

Syntax Error: Ambiguous argument - matches the beginning of each of these argument names:
-__newline__, -additionalpostparams, -autologin,
-certans, -custom_atrib_file, -dir, -licans,
-localdirans, -noautologin, -nocertvalidate,
-noregister, -novalidate, -offline, -password, -trustall,
-url, -username, -verb_jars_dir

emcli setup -ur="https://host:port/em" -us=<emuser> -t
Basic Operational Verbs

- help: Get help for emcli verbs (Usage: emcli help [verb_name])
- setup: Setup emcli to work with an EM Management Server
- sync: Synchronize with the EM Management Server (updates)
- login: Login to the EM Management Server (OMS)
- logout: Logout from the EM Management Server
- status: List emcli configuration details
- version: List emcli verb versions or the emcli client version
- argfile: Execute emcli verbs from a file
- invoke_ws: Invoke EM web service.
400+ Verbs in 76 Categories

- Add Host Verbs
- Agent Administration Verbs
- Agent Upgrade Verbs
- BI Publisher Reports Verbs
- Blackout Verbs
- Credential Verbs
- Incident Rules Verbs
- Job Verbs
- Metric Extension Verbs
- Monitoring Templates Verbs
- Oracle Cloud Verbs
- Provisioning Verbs
- Self Update Verbs
- User Administration Verbs
“Grep” is your friend!

Some verbs provide own filtering mechanism, but some of them don’t. Using grep will help you to filter the output of emcli.

emcli help | grep -i blackout

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_blackout_reason</td>
<td>Adds a new blackout reason.</td>
</tr>
<tr>
<td>create_blackout</td>
<td>Create a blackout</td>
</tr>
<tr>
<td>create_rbk</td>
<td>Create a Retro-active blackout.</td>
</tr>
<tr>
<td>delete_blackout</td>
<td>Delete a blackout</td>
</tr>
<tr>
<td>get_blackout_details</td>
<td>Get detailed info for a blackout.</td>
</tr>
<tr>
<td>get_blackout_reasons</td>
<td>List all blackout reasons</td>
</tr>
<tr>
<td>get_blackout_targets</td>
<td>List targets for a blackout</td>
</tr>
<tr>
<td>get_blackouts</td>
<td>List blackouts</td>
</tr>
<tr>
<td>stop_blackout</td>
<td>Stop a blackout</td>
</tr>
<tr>
<td>get_targets</td>
<td>Obtain status and blackout information for targets</td>
</tr>
</tbody>
</table>

emcli get_target_types | grep -i database
Find the exact name of a target

Some verbs require the name and type as input. Target names, target types and even “verbs” are case sensitive, so be sure that you use the exact name of a target and target type:

- `emcli get_targets`
- `emcli get_targets -targets="targetname:targettype"`
- `emcli get_targets -targets="targetname:%"
- `emcli get_targets -targets="targettype"`
Format and redirect the output

```
emcli get_targets -format="name:pretty"
emcli get_targets -format="name:csv"
emcli get_targets -format="name:script"
emcli get_targets -script
emcli get_targets -script > list_of_targets.txt
emcli get_targets -script I parsing_tool
```

Output can be parsed easily by Unix tools
List the targets which has critical alerts

emcli get_targets -alerts

<table>
<thead>
<tr>
<th></th>
<th>Up</th>
<th>host</th>
<th>db02-test.bilyoner.com</th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Up</td>
<td>host</td>
<td>db01-test.bilyoner.com</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Up</td>
<td>host</td>
<td>db03-test.bilyoner.com</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

emcli get_targets -alerts -script -noheader | awk '{ if ($5>0) print $4":"$3"\t"$5 }'

db02-test.bilyoner.com:host  1
db01-test.bilyoner.com:host  1
Gateway to the EM repository

- Although there are more than 70 verbs to help you fetch a specific information from OMS such as list of targets, list of jobs etc, you may still need a more powerful verb which can fetch information from the repository database.

- The following command will list roles defined in EM:

  emcli list -resource="Roles"

- This command will show all resources that you can query:

  emcli list -help
Gateway to the EM repository

- This command shows the columns of the specified resource:
  
  emcli list -resource="Administrators" -help

- This command will list user names and user types defined as EM administrator:
  
  emcli list -resource="Administrators"
  -columns="USER_NAME,USER_TYPE"

- This command will show details about SYSMAN administrator:
  
  emcli list -resource="Administrators" -search="USER_NAME = 'SYSMAN'"
Gateway to the EM repository

For now, “List” verb can query 27 resources. If you can not find the data you need among these resources, you can even run SQL on repository database:

```
emcli list -sql="<query>"
```

```
./emcli list -sql="SELECT USER FROM DUAL"
USER
MGMT_VIEW
Rows:1
```
emcli login -username="GOKHAN" -password="s3cr3t"

emcli create_blackout -name="DEVDB Blackout" -add_targets="DEVDB:oracle_database" -reason="Maintenance work" -schedule="duration:200"

...

...

emcli stop_blackout -name="DEVDB Blackout"
emcli delete_blackout -name="DEVDB Blackout"
emcli logout
Sample Script: Clear Stateless Alerts

```
login=$((emcli login -username="GOKHAN" -password="s3cr3t" | grep -c "successful")

if [ $login -eq 1 ]; then
    for DBNAME in $(emcli get_targets -targets="oracle_database" -noheader -script | cut -f4);
    do
        emcli clear_stateless_alerts -older_than=0 -target_type="oracle_database" -target_name="$DBNAME"
    done
    emcli logout
fi
```
Sample Script: Delete All Named Credentials

```
login=$(emcli login -username="GOKHAN" -password="s3cr3t" | grep -c "successful")

if [ $login -eq 1 ]; then
    for CN in $(emcli list_named_credentials -script -noheader | cut -f1);
    do
        emcli delete_named_credential -cred_name="$CN"
    done
emcli logout
fi
```
EM CLI with Scripting Mode

Run "emcli" without any parameters;

```
Oracle Enterprise Manager 12c Release 5 EM CLI with Scripting option.
Copyright (c) 1996, 2015 Oracle Corporation and/or its affiliates. All rights reserved.
Type help() for help and exit() to get out.
[emcli]>login( username="GOKHAN" )
[Enter password : ******
Login successful
[emcli]>get_targets( target="TESTDB:%" )
Status  Status  Target Type  Target Name
ID  Up  oracle_database  TESTDB
```

... and you find yourself in a Jython world!
Someone said Jython?

Jython is an implementation of the Python programming language designed to run on the Java platform.
So what’s Python?

- Widely used general-purpose, high-level programming language
- Supports multiple programming paradigms, including OOP, imperative and functional programming
- Dynamic type system and automatic memory management
- Batteries Included (!)
Verbs are defined as Python functions

Arguments are entered as function parameters

```python
get_targets( target="oracle_database")
```

Results of verbs can be formatted as JSON

Some Python modules are included (math, os … )

Programming elements of Python makes scripting easy!
Fundamentals of Python

Variables (dynamic type, no need for declaration but need initialisation):

- ✓ myvar = "TESTDB"
- ✓ myvar = myvar + ":oracle_database"
- ✓ myvar = 10
- ✓ myvar = myvar + 10
- ✗ myvar = 10 + "10"

Type Error: unsupported operand type(s) for +: 'int' and 'str'
Fundamentals of Python

Conversion between integer and string, conditional jumps and code blocks:

```python
myvar = int("2")
myvar = 2
myvar = "2"

if ( myvar >= 10):
    print "myvar is " + str(myvar)
else:
    print "lower than 10"
```

Code blocks are identified by “indentation”
Dictionary objects, list objects and iterations:

```python
mydict = { 'Name':'Gokhan', 'Age':40 }
mydict['Name'] = "Gokhan"

mylist = ['Joe','William','Jack','Averell']
mylist[0] = "Joe"

for i in mylist:
    print i
```

it will print all elements in "mylist"
EM CLI Response Object

emcli.response represents the response object which returned from invocation of a emcli verb from python script.

```
emcli.response.Response

  - error()  error text
  - exit_code()  error code (0=success)
  - isJson()  is JSON? true/false
  - out()  JSON or text
```
EM CLI Response Object

In scripting mode, default output format type is JSON. In interactive mode, to change the output format to JSON, you need to run the following command:

```python
set_client_property('EMCLI_OUTPUT_TYPE', 'JSON')
```

It’s better to use JSON because it directly matches with Python “dictionary” object. There’s no need for extra effort to parse the output.
EM CLI Response Object

Let’s get the response from a verb:

```python
mytargets = get_targets( target="oracle_database" )

type(mytargets.out())
<type 'dict'>

mytargets.out()

{'data': [{'Status': 'Up', 'Warning': '6', 'Status ID': '0', 'Target Type': 'oracle_database', 'Critical': '64', 'Target Name': 'TESTDB'}, {...}]
```
EM CLI Response Object

A response always contain a “data” item which is a list object:

```python
type(mytargets.out()['data'])
<type 'list'>
```

```python
mytargets.out()['data']
[
{'Status': 'Up', 'Warning': '6', 'Status ID': '0', 'Target Type': 'oracle_database', 'Critical': '64', 'Target Name': 'TESTDB'}, {...}
]```
EM CLI Response Object

After we reach the list, we can iterate over the items of the list:

```python
for DB in mytargets.out()['data']:
    print DB['Target Name'], DB['Status']
```

```plaintext
[...  print DB['Target Name'], DB['Status']
[...
  TESTDB Up
  TESTERE Up
  NOVA Up
[emcli>
```
Sample Script: Clear Stateless Alerts

Save the following code in a file (clearalerts.py) and run it as "emcli @clearalerts.py":

```python
if login( username="GOKHAN" ).exit_code()==0:
    mytargets=get_targets( targets="oracle_database" )

    for DB in mytargets.out()]['data']:
        clear_stateless_alerts(
            target_name=DB['Target Name'],
            target_type=DB['Target Type'],
            older_than="0"
        )
```
Sample Script: Change DBSNMP Passwords

if len(sys.argv) <> 2:
    print "Usage: emcli @change_dbsnmp_passwords.py oldpwd newpwd"
    exit()
if login( username="GOKHAN" ).exit_code()==0:
    mytargets=get_targets( targets="%_database" )
    for DB in mytargets.out()['data']:
        try:
            update_db_password (target_name=DB['Target Name'],
                                target_type=DB['Target Type'],
                                change_at_target="yes",user_name="dbsnmp",
                                old_password=sys.argv[0],
                                new_password=sys.argv[1],
                                retype_new_password=sys.argv[1])
        except emcli.exception.VerbExecutionError, e:
            print e.error()
Sample Script: Promote Unmanaged Databases

lg = login( username="SYSMAN" )

if lg.exit_code() <> 0:
    print lg.error()

utargets=get_targets( "unmanaged", "properties",
                        targets="oracle_database" )

for target in utargets.out()['data']:
    add_target( name=target['Target Name'], type=target['Target Type'],
                host=target['Host Info'].split(';')[0].split(':')[1],
                credentials="UserName:dbsnmp;password:xyz;Role:Normal",
                properties=target['Properties'] )
Sample Script: Promote Unmanaged Databases

```python
# Sample script to promote unmanaged databases

import emcli

# Login with admin credentials
lg = login(username="SYSMAN")
if lg.exit_code() != 0:
    print(lg.error())

# Get unmanaged targets
utargets = get_targets("unmanaged", "properties")
for target in utargets.out()['data']:
    add_target(name=target['Target Name'],
               type=target['Target Type'],
               host=target['Host Info'].split(':')[0].split(':')[1],
               credentials="UserName:dbsnmp;password:xyz;Role:Normal",
               properties=target['Properties'])
```

Further Reading:

- OEM 12c Command-Line Interface
- My blog: http://www.gokhanatil.com
- Kellyn Pot'Vin-Gorman http://dbakevlar.com
- Ray Smith https://oramanageability.wordpress.com
- Sete Miller http://sethmillner.org
- The Definitive Guide to Jython http://www.jython.org/jythonbook/en/1.0/
THANK YOU FOR ATTENDING!
ANY QUESTIONS?