Executing operating system commands from PL/SQL

An Oracle White Paper
July 2008

Executing shell (operating system) commands from PL/SQL

INTRODUCTION

This document provides PL/SQL packages **OS_COMMAND**, **FILE_PKG** and **FILE_TYPE**, which enable an Oracle database developer to interact with the operating system or with ordinary files. Oracle provides out-of-the-box only limited functionality for accessing the file system and no functionality to execute shell commands or operating system processes. This paper intends to provide this.

In general there are three ways to execute shell commands by the Oracle engine.

1. Coding a native shared library, reference it in the Oracle database using CREATE LIBRARY and create a PL/SQL wrapper on the particular functions. This required a C compiler for the platform Oracle is running on.
2. Using the Oracle-supplied package DBMS_SCHEDULER that is able to execute shell commands. But DBMS_SCHEDULER lacks access to STDIN and STDOUT; if this is needed, DBMS_SCHEDULER is not an option.
3. Since Oracle8i there is a Java SE compliant Virtual Machine embedded in the database kernel; as any other JVM it is capable of executing shell commands via the System.exec method. The JDBC API allows passing the contents of STDIN, STDOUT or the OS return values back to the SQL engine. And since Java is platform-independent one install script does the job for all platforms.

This paper focuses the third alternative i.e., Java in the database.

FILE SYSTEM INTERACTION WITH SQL AND PL/SQL

Installation Options

There are two installation options:

1. Create the java classes and packages as SYS user and make it accessible to all
2. Create the java classes and packages in a non-priviledged user schema

In both the cases the invoking user needs proper java privileges in order to use the package

Software requirement

1. Database version 10.1 or higher
2. Java in database installed and enabled

Run the following command as SYS user, to get details on java installed in database:

```
sqlplus> select comp_name, version from dba_registry where comp_name like ‘%JAVA%’;
```

3. Appropriate java_pool_size (min. 50 MB)

SQL scripts

The installer (http://www.oracle.com/technology/tech/java/jsp/files/os_command.zip) contains the following SQL scripts:

1. **os_command_java.sql**: creates Java class for executing OS commands from SQL or PL/SQL, and for accessing the file system from the SQL layer The execution of the OS commands is done via the Java File system interface and the System.exec call. The results are then passed in SQL engine via JDBC classes. So the caller should have proper java privileges. The class for accessing the file system provides methods to do typical operations with files via java.io.File. Those methods are intended to be called from PL/SQL using an object type FILE_TYPE. This also creates the PL/SQL packages: FILE_PKG is a helper package to obtain one or multiple file handles, OS_COMMAND is a package to execute shell commands, and LOB_WRITER_PLSQL is a helper package to write LOBs (CLOB, BLOB) to operating system files with pure PL/SQL and "traditional" directory objects
2. **grant_public.sql**: run this script if you created the packages and java classes as SYS user and desire to create public synonyms and grant execute privilege to other users.
3. **java_grants.sql**: grants proper Java privileges in order to execute OS commands.
4. **sample_script.sql**: demonstrates extracting the contents of a ZIP file into a table
5. **cleanup.sql**: to deinstall the packages and classes

Installation steps

1. unzip the installer to a directory on your system

2. in this paper we will create the java classes and the packages in schema ‘SCOTT’. As SYS user we need to grant java permissions to ‘SCOTT’ to execute the operating system commands. So connect to database as SYS user
3. run the following script and the command as mentioned below:

sqlplus> @java_grants.sql;

sqlplus> grant create public synonym to scott;

4. connect to database as scott.

sqlplus> connect scott/<password>;

5. run the following scripts in the sequence, it will create the necessary classes and packages:

sqlplus> @os_command_java.sql;

sqlplus> @file_type_java.sql;

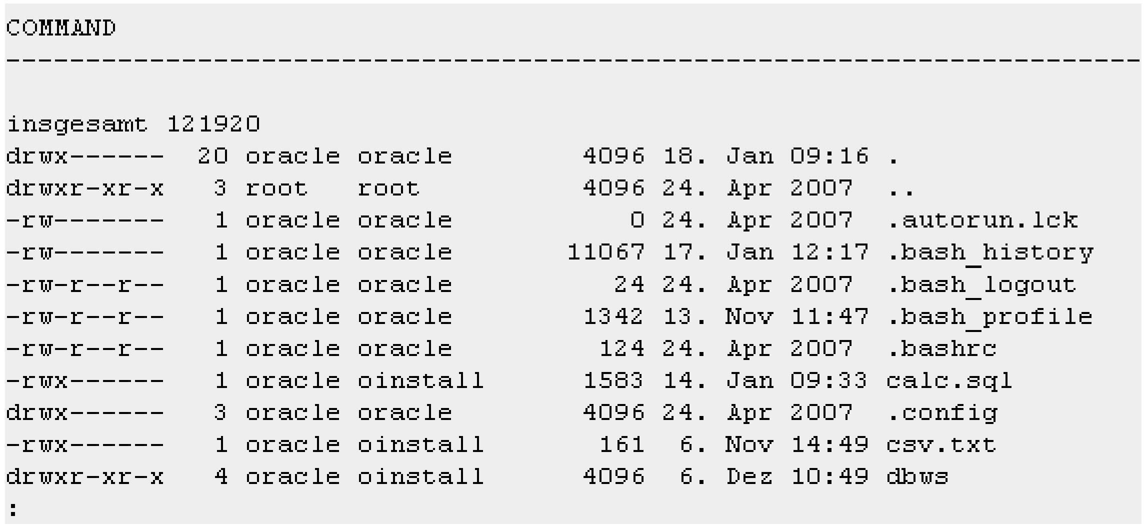
sqlplus> @os_command.pls;

sqlplus> @os_command.plb;

6. to execute a shell command returning text output (ls -la) run the following:

sqlplus> select os_command.exec_clob('/bin/ls -la /home/oracle') COMMAND from dual;

You should get result as shown in screen shot below:

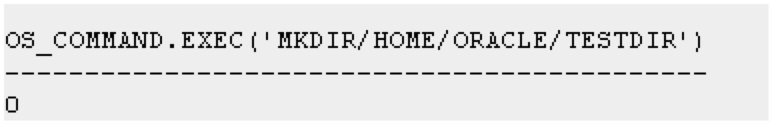


```
COMMAND
----------------------------------------------------------------------

insgesamt 121920
drwx------  20 oracle oracle        4096 18. Jan 09:16 .
drwxr-xr-x   3 root   root          4096 24. Apr 2007  ..
-rw-------   1 oracle oracle           0 24. Apr 2007  .autorun.lck
-rw-------   1 oracle oracle       11067 17. Jan 12:17 .bash_history
-rw-r--r--   1 oracle oracle          24 24. Apr 2007  .bash_logout
-rw-r--r--   1 oracle oracle        1342 13. Nov 11:47 .bash_profile
-rw-r--r--   1 oracle oracle         124 24. Apr 2007  .bashrc
-rwx------   1 oracle oinstall      1583 14. Jan 09:33 calc.sql
drwx------   3 oracle oracle        4096 24. Apr 2007  .config
-rwx------   1 oracle oinstall       161  6. Nov 14:49 csv.txt
drwxr-xr-x   4 oracle oinstall      4096  6. Dez 10:49 dbws
:
```

7. to create a directory run the following command and you will get the result as shown in the screenshot below:

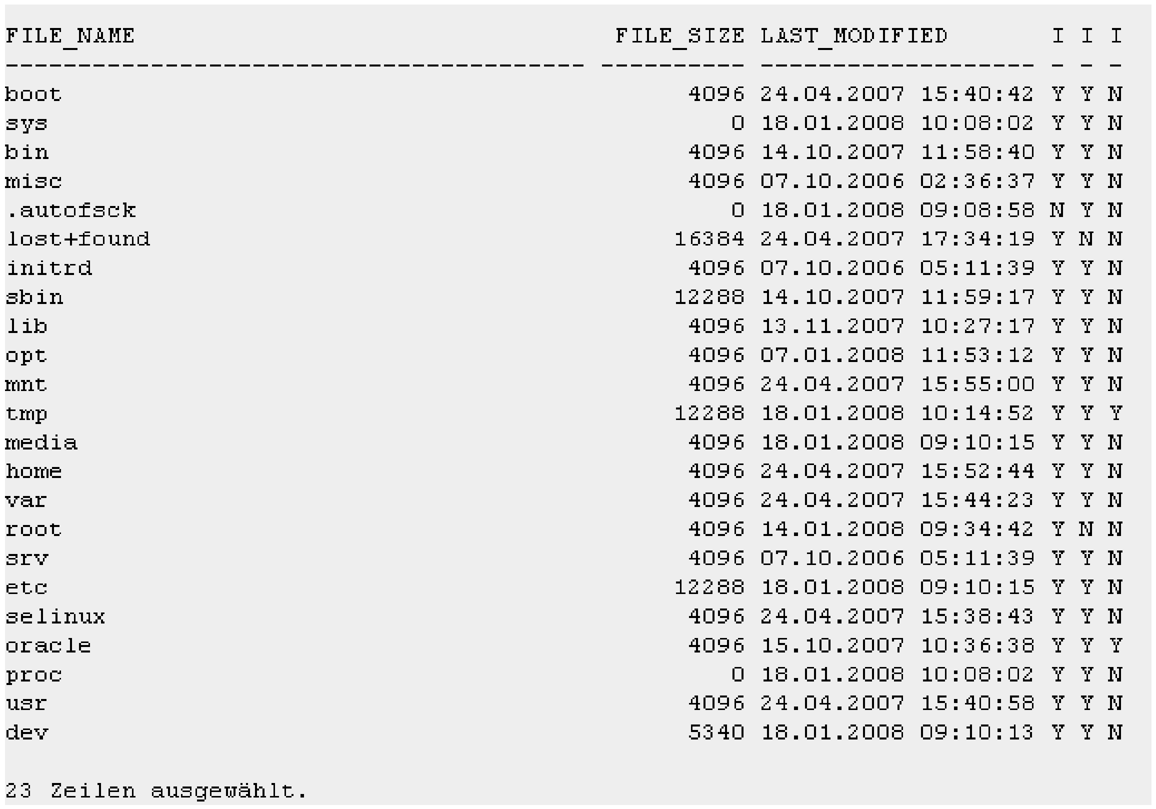
sqlplus> select os_command.exec('mkdir /home/oracle/testdir') from dual;



```
OS_COMMAND.EXEC('MKDIR/HOME/ORACLE/TESTDIR')
--------------------------------------------
0
```

8. to get the files in a directory as a virtual table run the following command and you will get the result as shown in the screenshot below:

```
sqlplus> select * from table(file_pkg.get_file_list(file_pkg.get_file('/')));
```



```
FILE_NAME                                  FILE_SIZE LAST_MODIFIED       I I I
---------------------------------------- ---------- ------------------- - - -
boot                                             4096 24.04.2007 15:40:42 Y Y N
sys                                                 0 18.01.2008 10:08:02 Y Y N
bin                                              4096 14.10.2007 11:58:40 Y Y N
misc                                             4096 07.10.2006 02:36:37 Y Y N
.autofsck                                           0 18.01.2008 09:08:58 N Y N
lost+found                                      16384 24.04.2007 17:34:19 Y N N
initrd                                           4096 07.10.2006 05:11:39 Y Y N
sbin                                            12288 14.10.2007 11:59:17 Y Y N
lib                                              4096 13.11.2007 10:27:17 Y Y N
opt                                              4096 07.01.2008 11:53:12 Y Y N
mnt                                              4096 24.04.2007 15:55:00 Y Y N
tmp                                             12288 18.01.2008 10:14:52 Y Y Y
media                                            4096 18.01.2008 09:10:15 Y Y N
home                                             4096 24.04.2007 15:52:44 Y Y N
var                                              4096 24.04.2007 15:44:23 Y Y N
root                                             4096 14.01.2008 09:34:42 Y N N
srv                                              4096 07.10.2006 05:11:39 Y Y N
etc                                             12288 18.01.2008 09:10:15 Y Y N
selinux                                          4096 24.04.2007 15:38:43 Y Y N
oracle                                           4096 15.10.2007 10:36:38 Y Y Y
proc                                                0 18.01.2008 10:08:02 Y Y N
usr                                              4096 24.04.2007 15:40:58 Y Y N
dev                                              5340 18.01.2008 09:10:13 Y Y N

23 Zeilen ausgewählt.
```

9. to inflate a ZIP file and load the contents into a table run the sample_scripts.sql. It will ask which ZIP file you want to inflate, create a temporary directory where the zip archive is to be extracted, load its contents into the table and finally delete the temporary directory. Before running the script make sure that relevant permissions must be granted from SYS to SCOTT as shown below:

```
sqlplus> connect / as sysdba;
sqlplus> exec DBMS_JAVA.grant_permission('SCOTT', 'java.io.FilePermission', '<<ALL FILES>>', 'read ,write,
execute, delete');
sqlplus> exec Dbms_Java.Grant_Permission('SCOTT', 'SYS:java.lang.RuntimePermission', 'writeFileDescriptor', '');
sqlplus> exec Dbms_Java.Grant_Permission('SCOTT', 'SYS:java.lang.RuntimePermission', 'readFileDescriptor', '');

sqlplus> connect scott/<password>;
sqlplus> @sample_script.sql;
```

10. You can also modify the following code as per your system requirement to inflate a ZIP file and load the contents into a table:

```
declare
  f  file_type;
  fz file_type;

  r  number;
```

```
begin
  -- get a handle for the "tmp" directory
  f:=file_pkg.get_file('/tmp');

  -- create a new temporary directory where the zip archive is being
  -- extracted into ... make the filename unique using TIMESTAMP
  fz := f.create_dir(
    'zipdir_temp_'||user||'_'||to_char(systimestamp, 'YYYYMMDD_HH24MISS.SSSS')
  );

  -- extract the zipfile; the -qq switch is important here otherwise
  -- the OS process will not come back
  r := os_command.exec('unzip -o -qq [/PATH/TO/ZIPFILE] -d '||fz.file_path);

  -- if the result is 0 (=success) load the contents of the temporary directory
  -- (recursively) with ONE (!) SQL INSERT command
  if r = 0 then
    insert into document_table (
      select
        seq_documents.nextval id,
        e.file_path,
        e.file_name,
        file_pkg.get_file(e.file_path).get_content_as_clob('iso-8859-1') content
      from table(file_pkg.get_recursive_file_list(fz)) e
    );
  end if;

  -- finally delete the temporary directory and its contents
  fz := fz.delete_recursive();
end;
/
sho err
```

CONCLUSION

This document explained a PL/SQL method to interact with the operating system files. The Virtual Machine embedded in the database kernel, as any other JVM, is capable of executing shell commands via the System.exec method and we consumed that feature in the present example.

RECOMMENED READINGS

1. Technical Whitepapers on OTN (http://www.oracle.com/technology/tech/java/jsp/index.html)
2. Oracle Database Programming Using Java and Web Services by Kuassi Mensah (http://db360.blogspot.com/2006/08/oracle-database-programming-using-java_01.html)

ORACLE

Calling shell (operating system) commands from PL/SQL
July 2008
Author: Priyanka Sharma
Contributing Authors: Carsten Czarski

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com