Cloning and Content Movement Administrator's Guide

Enterprise Content Management SDK

Version 1.0

Primary Author: Frank Closheim
Contributing Author: Dave Long

Table of contents

1 Preface

Enterprise Content Management SDK (ECMSDK) is a robust and flexible runtime and development platform for building demanding content management applications. Prior to version 1.0, ECMSDK was known as Oracle Content Management SDK (Oracle CM SDK). The features and capabilities of ECMSDK are designed to help developers bring enterprise-scalable content management-based applications to market faster and better than with any other platform. ECMSDK provides a set of Java APIs that developers can use to customize or extend the product's core functionality in numerous ways to build comprehensive content-centric database solutions.

This document contains necessary information for cloning an existing ECMSDK instance, creating an identical second version.

Note: This document is accurate at the time of publication. inxire will update the documentation periodically after the software release. You can access the latest information and additions to this document on the ECMSDK website at:

http://www.ecmsdk.com

1.1 Audience

This ECMSDK Cloning and Content Movement Administrator's Guide is intended for administrators who need to create an identical clone of an existing ECMSDK instance. One possible purpose could be to clone a production environment to create a new test or development environment containing production data.

1.2 Related Documents

For more information, see these additional resources:

Enterprise Content Management SDK

1. ECMSDK Release Notes
2. ECMSDK Installation Guide

Oracle Database

3. Oracle Database SQL Language Reference
4. Oracle Database Utilities
5. Oracle Text Reference

2 Introduction to ECMSDK Cloning

The cloning instructions described in this document refer to the complete duplication of an ECMSDK source environment which was initially installed using the standard installation procedures. Cloning creates a matching destination ECMSDK environment with its own unique set of access and integration points.

2.1 System Requirements

The following requirements apply for the instructions described in this document:

1. **Software version.** The ECMSDK Software version need to be identical on the source and destination environment.
2. **Installation directories**. The directory structure that was used for ECMSDK installation on the source environment need to be identical with the directory structure on the destination environment.
3. **Disk space**. Both systems need to have sufficient disk space to hold the export and import data. The destination environment need to have additional space for the data coming after the initial import.
4. **LDAP integration.** If an integration with MS Active Directory (AD) or Oracle Internet Directory (OID) has been used on the source environment, the same AD or OID need to be used for the destination environment.
5. **Ports.** The clone environment will use the same ports as the source; therefore all ports used in the source environment must also be available on the destination.

2.2 Required Skills

Cloning an ECMSDK instance requires some combination of the following skills:

1. **Configuring and administering the underlying Operating System**. You must have a basic understanding of the underlying Operating System to be able to create directories, move files and folders around and set the right access permissions. Remote access over a network is as necessary as access via different protocols.
2. **Configuring and administering the Oracle database.** You must be able to install and manage the Oracle database, the repository for the file system. Only basic database administration skills are required, however.
3. **Querying database contents through SQL**. To be able to execute several cloning scripts as well as to query the contents of the ECMSDK schema, you need SQL skills.
4. **Installing, configuring, and administering ECMSDK**. You need to be able to install and manage ECMSDK components. Wherever you install ECMSDK —on the same machine as the database or on a middle-tier machine—is where you will also be hosting many of your application components. Intermediate ECMSDK administration skills are required, particularly when troubleshooting issues during the cloning process.

2.3 What Does Cloning Include?

The destination environment clone will be an exact replica of the source, including:

- Complete topology
- Transactional and setup data
- Identity Management components and data
- Directory structures will be identical between source and destination
- The ECMSDK release version will be identical between source and destination

Since the internal host names and environment specific parameters will be identical between source and destination after the initial cloning procedure, there is a post-clone phase where these environment specific parameters will be re-wired to match your destination environment.

2.4 Terminology

Common terminology used in the cloning includes:

1. **Source Environment**. An existing ECMSDK environment that is already installed against which a copy or clone is to be made.
2. **Target Environment.** The new environment that will be created as a result of the clone activity. Also called the destination environment.
3. **Content Movement.** Refers to the task of refreshing all ECMSDK transactional data in a destination environment by copying all the transactional data from a source environment.

2.5 When Is Cloning Useful?

Cloning enables the copying of an existing ECMSDK environment to a new set of servers and has many applications:

- Standing up a new environment that is a copy of the existing one for testing or development purposes.
- Refreshing non-production systems with production data and production configurations/ binaries.
- Using the gold-image source files as "templates" for creating multiple new environments.

Note: Standard database duplication is another kind of content movement, which is a subsection of cloning (duplicating the database structure and contents), but does not include the rest of the ECMSDK instance.

3 Data Export from the Source Environment

This chapter describes the procedure of exporting the entire ECMSDK data from the source environment.

The instructions in this chapter are specific to a Linux environment; use the appropriate commands for your operating environment.

3.1 Preliminary Work

The following preliminary work has to be done.

- Identifying the expected export size
- Identifying the tablespace sizes for the import
- Stopping all ECSMDK processes

3.1.1 Identifying the Expected Export Size

Connect to the ECMSDK repository database on your source environment with user `SYS` and execute the following SQL statement:

Note: This SQL statement assumes that the names of the tablespaces used in your source environment for ECMSDK are all prefixed with `IFS`. If this is not the case, you need to change the statement accordingly to match your ECMSDK tablespace names.

```
SQL> select SUM( sum( ROUND(files.BYTES/(1024*1024))) -
round(total_bytes/(1024*1024*1024))) "Consumed (MB)" from
dba_free_space_coalesced free, dba_data_files files where
free.tablespace_name = files.tablespace_name and
files.tablespace_name like 'IFS%' group by
files.tablespace_name, free.total_bytes;

Consumed in MB
--------------
32585

SQL> exit
```

The computed value (32585 in this example) specifies the maximum expected value in MB for the export. This value will be referred to hereafter as `<EXPORTSIZE>`.

Choose a destination directory for the export data on your source environment. This directory need to have minimum the free capacity of the computed `<EXPORTSIZE>`. This directory will be referred to hereafter as `<EXPORTDIR>`.

3.1.2 Identifying the Tablespace Sizes for the Import

Connect to the ECMSDK repository database on your source environment with user `SYS` and execute the following SQL statement:

Note: This SQL statement assumes that the names of the tablespaces used in your source environment for ECMSDK are all prefixed with `IFS`. If this is not the case, you need to change the statement accordingly to match your ECMSDK tablespace names.

```
SQL> select files.tablespace_name, sum(
ROUND(files.BYTES/(1024*1024)))-
round(free.total_bytes/(1024*1024)) "Consumed (MB)" from
dba_free_space_coalesced free, dba_data_files files where
free.tablespace_name = files.tablespace_name and
files.tablespace_name like 'IFS%' group by
files.tablespace_name, free.total_bytes;

TABLESPACE_NAME      Consumed (MB)
-------------------- -------------
IFS_MAIN                       143
IFS_CTX                         83
IFS_LOB                        612

SQL> exit
```

3.1.3 Stopping ECMSDK processes

You need to stop all running ECMSDK processes on the source environment. This includes all running ECMSDK Nodes, managed or unmanaged, as well as the Application Server processes that are running any application accessing the ECMSDK repository.

A restart of these processes during the export phase can result in corrupted export data.

Note: Refer to the *ECMSDK Installation Guide* and your specific middleware documentation for detailed start/stop instructions.

The only component that should run on the source environment during the export phase is the ECMSDK database instance.

3.2 Export Procedure

This chapter describes the actual procedure of exporting the transactional data into the filesystem. The tool that is used for this export is called Oracle Data Pump Export utility.

3.2.1 What is Data Pump?

Data Pump Export (invoked with the `expdp` command) is a new utility as of Oracle Database 10g. Although its functionality and its parameters are similar to those of the original Export utility (`exp`), they are completely separate utilities and their files are not compatible.

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called a dump file set. The dump file set can be imported only by the Data Pump Import utility. The dump file set can be imported on the same system or it can be moved to another system and loaded there.

The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

See Also: See *Oracle Database Utilities* documentation for general information about Oracle Data Pump Export and Import.

3.2.2 Creating an Export Directory Object

Because the dump files are written by the server, rather than by the client, the data base administrator (DBA) must create directory objects. To do so, connect to the ECMSDK repository database on your source environment with user SYS and execute the following SQL statement:

```
SQL> CREATE DIRECTORY ecmsdkDump01 AS '<EXPORTDIR>';

SQL> GRANT READ, WRITE ON DIRECTORY ecmsdkDump01 TO system;
```

3.2.3 Starting the Export

Login as the operating system user that owns the database installation on your source environment, typically `oracle`.

```
bash> cd <EXPORTDIR>
bash> export NLS_LANG=AMERICAN_AMERICA.UTF8

bash> expdp directory=ecmsdkDump01 parallel=2
dumpfile=ecmsdk.dmp schemas= ECMSDK,ECMSDK\$CM,ECMSDK\$WF
```

A file named `ecmsdk.dmp` will be created in `<EXPORTDIR>`.

3.2.4 Starting ECMSDK processes

You can start all ECMSDK processes again on the source environment after the export has been finished.

Note: Refer to the *ECMSDK Installation Guide* and your specific middleware documentation for detailed start/stop instructions.

4 Moving Data from the Source to Destination

This chapter guides you through the process of moving the export data as well as external content object if applicable from your source to the destination environment.

4.1 Moving the Export Dump File

Once you have finished the export procedure described in the previous chapter 3, you need to choose a destination directory for the export data on your destination environment. This directory need to have minimum the free capacity of the computed `<EXPORTSIZE>` from chapter 3.1.1. This directory will be referred to hereafter as `<IMPORTDIR>`.

Now, all files from your `<EXPORTDIR>` of you source environment need to be copied into your `<IMPORTDIR>` on your destination environment.

4.2 Copying External LOBs (BFILEs)

If you are making use of BFILEs in your ECMSDK source environment, these external LOBs need to be copied to the destination environment manually, as they are not part of the Dump file created in chapter 3.

To verify if BFILEs are used, you can issue the following SQL command against your source database as the ECMSDK schema user, typically `ECMSDK`. This will show the list of directory objects that have been created.

```
SQL> select * from all_directories where directory_name
like 'ECMSDK%';

OWNER  DIRECTORY_NAME   DIRECTORY_PATH
------ ---------------- --------------------------------
SYS    ECMSDK_BFD_4907  /oracle/app/db/ifsbfiles/ecmsdk
```

Now verify, if there are any files or directories in any of the directory paths listed by the above statement. If so, you need to manually **copy** the entire directory structure to the same directory path on the destination environment.
If there are no files in any of the directory paths listed by the above statement, then BFILEs are currently not used in your source environment; therefore no additional action need to be taken.

5 Data Import into the Destination Environment

This chapter describes the procedure of importing the ECMSDK data from source into destination.

The instructions in this chapter are specific to a Linux environment; use the appropriate commands for your operating environment.

5.1 Preliminary Work

The following preliminary work has to be done.

- Install and configure the destination environment
- Stopping all ECMSDK processes

5.1.1 Install and configure the destination environment

The following steps have to be executed before starting the actual import procedure on the destination environment:

1. **Install ECMSDK.** Install an identical version of ECMSDK on your destination environment, including all required components, such as Oracle Database, ECMSDK and Middleware Software, ensuring you have the exact same version for each of the components.

   Note: Refer to the *ECMSDK Installation Guide* for detailed installation instructions.

2. **Turn off Archive Log.** Due to performance reasons, it is recommended to turn the destination database into `NOARCHIVELOG` during the import phase. However, if doing so, a complete backup after the entire cloning process is mandatory.

   To turn archive log of, issue the following SQL commands:

```
bash> sqlplus "/as sysdba"

SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount exclusive;
SQL> alter database noarchivelog;
SQL> alter database open;
SQL> select name, log_mode from v$database;

NAME      LOG_MODE
--------- ------------
ECM       NOARCHIVELOG

SQL> exit
```

3. **Extend Tablespace.** The tablespaces on the destination environment have to be sized sufficiently for a smooth import. If necessary, the tablespaces need to be extended.

   To determine the current size of the tablespaces, connect to the ECMSDK repository database on your source environment with user `SYS` and execute the following SQL statement:

   Note: This SQL statement assumes that the names of the tablespaces used in your destination environment for ECMSDK are all prefixed with `IFS`. If this is not the case, you need to change the statement accordingly to match your ECMSDK tablespace names.

```
SQL> select files.tablespace_name, sum(
ROUND(files.BYTES/(1024*1024)))-
round(free.total_bytes/(1024*1024)) "Consumed (MB)"
from dba_free_space_coalesced free, dba_data_files
files where free.tablespace_name =
files.tablespace_name and files.tablespace_name like
'IFS%' group by files.tablespace_name,
free.total_bytes;

TABLESPACE_NAME      Consumed (MB)
-------------------- -------------
IFS_MAIN                        22
IFS_CTX                         25
IFS_LOB                        412

SQL> exit
```

   If there is a value smaller than 1 in any of the returning rows, that means that this tablespace uses less than 1 MB of space.

   Compare the above computed values with your computed values from chapter 3.1.2. If the values in the source environment are larger than in the destination environment, you need to extend the tablespace accordingly.

   Example:

```
bash> sqlplus "/as sysdba"

SQL> ALTER DATABASE DATAFILE '<db file
path>/IFS_MAIN.dbf' RESIZE <required size in MB>M;

SQL> exit
```

5.1.2 Stopping all ECSMDK processes

You need to stop all running ECMSDK processes on the destination environment. This includes all running ECMSDK Nodes, managed or unmanaged, as well as the Application Server processes that are running any application accessing the ECMSDK repository.

A restart of these processes during the import phase can result in corrupted import data.

Note: Refer to the *ECMSDK Installation Guide* and your specific middleware documentation for detailed start/stop instructions.

The only component that should run on the destination environment during the import phase is the ECMSDK database instance.

5.1.3 ECMSDK Schema Cleanup

It is required that the destination environment has been stopped following instructions in chapter 5.1.2. The database user ECMSDK should not be connected with the database anymore. This can be verified by the following SQL commands:

```
bash> sqlplus "/as sysdba"

SQL> select USERNAME, SCHEMANAME, OSUSER, SID, PROGRAM from
v$session vs where vs.USERNAME like 'ECMSDK%';

SQL> exit
```

This SQL statement should not return a single row. If so, please make sure that all sessions with the users returned by this statement are terminated before you proceed.

We are now cleaning up any old data that might exist in the destination environment by dropping the ECMSDK schema users.

Note: You can find the SQL scripts used in the following commands in the Appendix

```
bash> sqlplus "/as sysdba"

SQL> spool dropUsers.log
SQL> @dropUsers.sql
SQL> spool off
SQL> exit
```

Verify the generated logfile `dropUsers.log` for any errors. Make sure you resolve all error before you proceed.

With the following commands, a clean ECMSDK schema user will be re-created. The new schema password needs to be passed to the script executed in the instructions below.

```
bash> sqlplus "/as sysdba"

SQL> spool recreateUsers.log
SQL> @recreateUser.sql <Schemapasswort ECMSDK>
SQL> spool off
SQL> exit
```

Verify the generated logfile `recreateUsers.log` for any errors. Make sure you resolve all errors before you proceed.

5.2 *Import Procedure*

This chapter describes the actual procedure of importing the transactional ECMSDK data into the destination database. The tool that is used for this import is called Oracle Data Pump Import utility.

5.2.1 Creating an Import Directory Object

Because the dump files are read by the server, rather than by the client, the data base administrator (DBA) must create directory objects. To do so, connect to the ECMSDK repository database on your destination environment with user SYS and execute the following SQL statement:

```
SQL> CREATE DIRECTORY ecmsdkDump01 AS '<IMPORTDIR>';

SQL> GRANT READ, WRITE ON DIRECTORY ecmsdkDump01 TO system;
```

5.2.2 Starting the Import

Login as the operating system user that owns the database installation on your destination environment, typically `oracle`.

```
bash> cd <IMPORTDIR>
bash> export NLS_LANG=AMERICAN_AMERICA.UTF8

bash> impdb system/<systempwd> directory=ecmsdkDump01
dumpfile=ecmsdk.dmp schemas= ECMSDK,ECMSDK\$CM,ECMSDK\$WF
```

You have successfully imported the date from the source environment into your destination.

6 Post-Clone Procedure

This chapter lists some mandatory post cloning tasks that need to be executed to make use of the full ECMSDK functionality after an import.

The instructions in this chapter are specific to a Linux environment; use the appropriate commands for your operating environment.

6.1 Update Environment Specific Domain Properties

Some data in the ECMSDK repository is environment specific. Therefore we need to update those data values to match our new environment.

6.1.1 Update HostName

```
SQL> update ODM_PROPERTY set
STRINGVALUE='<new_host>.<domain>' where
NAME='IFS.NODE.HostName' and
STRINGVALUE=’<old_host>.<domain>';
```

Example:

```
bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> select STRINGVALUE from ODM_PROPERTY where
NAME='IFS.NODE.HostName';

SQL> update ODM_PROPERTY set STRINGVALUE='test.inxire.com'
where NAME='IFS.NODE.HostName' and
STRINGVALUE='prod.inxire.com';

SQL> select STRINGVALUE from ODM_PROPERTY where
NAME='IFS.NODE.HostName';

SQL> commit;
```

6.1.2 Update DomainName

```
SQL> update ODM_PROPERTY set
STRINGVALUE='<new_ecmsdk_domain>' where
NAME='IFS.NODE.DomainName' and
STRINGVALUE=’<old_ecmsdk_domain>';
```

Example:

```
bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> select STRINGVALUE from ODM_PROPERTY where
NAME='IFS.NODE.DomainName';

SQL> update ODM_PROPERTY set
STRINGVALUE='ifs://test.inxire.com:1521:newdb.inxire.com:EC
```

```
MSDK' where NAME='IFS.NODE.DomainName' and
STRINGVALUE='prod.inxire.com';

SQL> select STRINGVALUE from ODM_PROPERTY where
NAME='IFS.NODE.DomainName';

SQL> commit;
```

6.2 Cleanup Event and Session Data

To clean up all old session and event data from the source environment, execute the following SQL commands:

```
bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> truncate table ODMZ_SESSION;
SQL> truncate table ODMZ_EVENT;
SQL> commit;
```

6.3 Re-Populate Oracle Text Index

It is highly recommended to re-cerate the Oracle Text index after the import was done. This ensures that you have the most up-to-date full text index available for your ECMSDK application.

6.3.1 Drop Oracle Text Index

To do so, we first drop the existing index:

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus system/<password>@<db_sid>;

SQL> @cleanupcontextpreferences.sql ECMSDK

bash> sqlplus ECMSDK/<password>@<db_sid;

SQL> @DropContextPreferences.sql ECMSDK

SQL> quit
```

Note: The script `DropContextPreferences.sql` can log some error if the database objects that need to dropped don’t exist. These errors can be ignored.

6.3.2 Create Context Procedure

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus ECMSDK/<password>@<db_sid>;

SQL> @CreateContextFunnelProcedure.sql ECMSDK
SQL> quit
```

6.3.3 Grant Text Permissions to ECMSDK

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus system/<password>@<db_sid>;

SQL> @GrantContextToIFS.sql ECMSDK

SQL> quit
```

6.3.4 Create Context Preferences

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus ECMSDK/<password>@<db_sid>;

SQL> @CreateContextPreferences.sql IFS_CTX IFS_CTX IFS_CTX
ECMSDK

bash> sqlplus ECMSDK/<password>@<db_sid>;

SQL> exec ctx_ddl.create_preference('IFS_DEF_FILTER',
'INSO_FILTER');

SQL> exec ctx_ddl.set_attribute('IFS_DEF_FILTER',
'TIMEOUT', '120');

SQL> commit;

SQL> quit
```

6.3.5 Create Oracle Text Index

These commands will re-create the Oracle Text Index with the `NOPOPULATE` option.

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus ECMSDK/<password>@<db_sid>;

SQL> update odmz_context_router set contentindex = null;

SQL> @CreateContextIndexNoPopulate.sql

bash> sqlplus ECMSDK/<password>@<db_sid>;

SQL> exec
ctx_ddl.create_preference('IFS_VIEW_AS_HTML_FILTER',
'INSO_FILTER');
```

```
SQL> exec ctx_ddl.set_attribute('IFS_VIEW_AS_HTML_FILTER',
'TIMEOUT', '120');

SQL> exec ctx_ddl.create_policy(policy_name =>
'IfsFilterPolicy', filter => 'IFS_VIEW_AS_HTML_FILTER',
section_group => 'IFS_SECTION_GROUP', lexer =>
'IFS_DEFAULT_LEXER', stoplist => 'CTXSYS.DEFAULT_STOPLIST',
wordlist => 'IFS_DEF_WORDLIST');

SQL> commit;

SQL> quit;
```

6.4 Populate Oracle Text Index

After the Oracle Text Index has been created a script will be called to populate the index in the background. This process can take a long time, depending on the amount of indexable documents that you have in your ECMSDK schema.

However, the destination environment can be used immediately, although the full text search might be limited in its functionality until that process is completely done.

The following SQL commands starts this indexing process:

```
bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> update odmz_context_router set contentindex = 'IFS';
SQL> commit;
SQL> quit
```

Create a script named `oracle-text.sql` under `$ECMSDK_HOME/install/sql`. This script should have the following content:

```
exec ctx_output.start_log('ifsidx.log');
exec ctx_ddl.sync_index('ifs_text');
exec ctx_output.end_log;
```

The script should then be executed as follows:

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus ECMSDK/<password>@<db_sid> @oracle-text.sql
>oracle-text.log >>oracle-text.log -nohup &
```

The logfile under `$ORACLE_DB_HOME/ctx/log/ifsidx.log` will have some information about the data already been processed by the indexing procedure while this task is still running.

6.4.1 Synchronize Oracle Text Index

To keep the index up-to-date in real time while documents are coming into the system, the following commands can be created as database jobs:

- `ctx_ddl.sync_index('IFS_TEXT');`
- `ctx_ddl.optimize_index(idx_name=>'IFS_TEXT', optlevel=>'FULL', maxtime=>60 );`

To create those database background jobs that keep your Oracle Text Index in sync with the ECMSDK content, execute the following SQL scripts:

```
bash> cd $ECMSDK_HOME/install/sql

bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> @ClearContextJobs.sql

bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> @SetupContextJobs.sql
```

6.5 Generating New Database Statistics

New database statistics should be generated before using the imported data.

```
bash> cd $ECMSDK_HOME/install/sql
bash> sqlplus ECMSDK/<password>@<db_sid>

SQL> @analyze.sql ECMSDK
```

6.6 Re-activate Archive Log

Due to performance reasons, you might have disabled Archive Log in the destination database before the import. If so, you should now enable it back again.

```
bash> sqlplus "/as sysdba"

SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount exclusive;
SQL> alter database archivelog;
SQL> alter database open;
SQL> select name, log_mode from v$database;

NAME      LOG_MODE
--------- ------------
ECM       ARCHIVELOG

SQL> exit
```

6.7 Starting ECMSDK Processes

You can start all ECMSDK processes again on the source environment after the export has been finished.

Note: Refer to the *ECMSDK Installation Guide* and your specific middleware documentation for detailed start/stop instructions.

7 Cloning Cleanup

This chapter describes some optional cloning cleanup tasks that can be done after the destination environment is back up and running.

The instructions in this chapter are specific to a Linux environment; use the appropriate commands for your operating environment.

7.1 Remove Export and Import Directory Object

On the database of the source environment:

```
bash> sqlplus "/as sysdba"

SQL> DROP DIRECTORY ecmsdkDump01;
```

On the database of the destination environment:

```
bash> sqlplus "/as sysdba"

SQL> DROP DIRECTORY ecmsdkDump01;
```

7.2 Delete Export Data

You can now delete the following files from your sourec and destination environment:

- `ecmsdk.dmp`
- `export.log`
- `import.log`

7.3 Delete Oracle Text Log

```
bash> cd $ECMSDK_HOME/install/sql

bash> rm oracle-text.sql
bash> rm oracle-text.log
```

8 Need More Help?

For more help troubleshooting ECMSDK cloning, go to http://www.ecmsdk.com

If you have a support contract with inxire, feel free to log a Support Ticket. The steps to download any available ECMSDK patches from ecmsdk.com are:

1. Go to http://www.ecmsdk.com.
2. Click Downloads.
3. View the latest Release Notes and go through the information before downloading the specific ECMSDK release.

If you do not find a solution for your problem, contact us:

inxire GmbH
Hanauer Landstr. 293 B

60314 Frankfurt, Germany

Telefon: +49 69 870000-10
Telefax: +49 69 870000-11

E-Mail: info@ecmsdk.com
Internet: http://www.ecmsdk.com

A Appendix

This appendix contains the source of all scripts used in this document. Feel free to adjust the SQL commands to your personal needs.

```
-- dropUser.sql

drop user ECMSDK$WF cascade;

drop user ECMSDK$CM cascade;

drop user ECMSDK cascade;


-- recreateUsers.sql

CREATE USER ECMSDK IDENTIFIED BY &1 DEFAULT TABLESPACE
IFS_MAIN TEMPORARY TABLESPACE TEMP;

CREATE USER ECMSDK$CM IDENTIFIED BY &1$CM DEFAULT
TABLESPACE IFS_MAIN TEMPORARY TABLESPACE TEMP;

CREATE USER ECMSDK$WF IDENTIFIED BY &1$WF DEFAULT
TABLESPACE IFS_MAIN TEMPORARY TABLESPACE TEMP;

grant connect, resource to ECMSDK, ECMSDK$CM, ECMSDK$WF;

grant execute on ctx_ddl to ECMSDK;

grant execute on ctx_output to ECMSDK;

grant execute on dbms_aqadm to ECMSDK;


grant execute on ctxsys.ctx_adm to ECMSDK;

grant execute on ctxsys.ctx_query to ECMSDK;

grant execute on ctxsys.ctx_thes to ECMSDK;

grant execute on SYS.DBMS_OBFUSCATION_TOOLKIT to ECMSDK;


GRANT CREATE ANY DIRECTORY to ECMSDK;

GRANT DROP ANY DIRECTORY to ECMSDK;

GRANT QUERY REWRITE to ECMSDK;

GRANT UNLIMITED TABLESPACE to ECMSDK;


GRANT AQ_ADMINISTRATOR_ROLE TO ECMSDK;

GRANT CTXAPP TO ECMSDK;
```