

Asynchronous Autolog Change Data Capture Cookbook

January 2007

Asynchronous Autolog CDC Cookbook

Concepts and terminology.....	3
Publishers and subscribers.....	3
CDC implementation methods.....	4
Supplemental logging.....	6
Environment.....	7
Connection settings.....	7
Required initialization parameters.....	8
Autolog online.....	8
Autolog archive.....	11
Set up the database users.....	14
Source Oracle Database 10g Release 2.....	14
Staging Oracle Database 10g Release 2.....	14
Set up and enable CDC.....	15
Create the source tables.....	15
Set up CDC: create the change tables.....	16
Activate CDC.....	20
Create a subscription.....	21
Autolog CDC in action.....	22
Perform DML against the source tables.....	23
Read the captured changes from the change tables.....	23
Extend the subscription window.....	25
Purge the subscription window.....	26
Purge the change tables.....	26
Introduce more DML changes.....	27
Extend the subscription window again.....	28
Final notes.....	28
Conclusion.....	29
Appendix A Cleanup the environment.....	30
Drop the subscription.....	30
Disable CDC.....	30
Drop the CDC database objects.....	31
Drop the source tables.....	33
Drop the users.....	33
Appendix B Known issues and workarounds.....	34
High System Commit Numbers (SCNs).....	34
Queue tables in SYSAUX tablespace.....	34

Asynchronous Autolog CDC Cookbook

Change Data Capture is a generic term that is used to describe technology to capture incremental changes being applied to a data store. With the amount of data in data stores growing and growing Change Data Capture is key enabling functionality for data warehouses, specifically real-time or near real-time data warehouses.

In the context of the Oracle Database, Change Data Capture is database functionality that enables capturing incremental changes against an Oracle Database. Traditionally you would have to modify the source application in order to capture incremental changes. Oracle's Change Data Capture enables incremental change capture without making any changes to a source application.

This cookbook describes how to setup an asynchronous Change Data Capture environment using change data capture on a separate database. Using this mode minimizes the impact on the source database.

CONCEPTS AND TERMINOLOGY

From this point onwards the abbreviation CDC will be used to refer to the Oracle Database Change Data Capture feature.

Publishers and subscribers

CDC uses the concept of publishers and subscribers. A publisher is a database user that publishes captured change data. A subscriber is a database user that consumes the change data through a so-called subscription. For security reasons publisher and subscriber should not be the same database user. One publisher can support multiple subscribers.

CDC makes use of change tables and subscriber views. Changes are written to the change tables to get a scalable infrastructure for using the changes. Subscribers to the changes get their own subscriber views (database views) against the change tables so that they look at a consistent set of data. Subscribers can extend and purge their subscription windows, implicitly changing the data set that is visible through the database views. Data can be purged from the change tables if no subscribers include or have yet to include the data in their views anymore. CDC is controlled using calls to PL/SQL database packages that are part of the Oracle Database.

CDC implementation methods

CDC has different implementation methods:

- Synchronous CDC: with this implementation method you capture changes synchronously on the source database into change tables. This method uses internal database triggers to enable CDC. Capturing the change is part of the original transaction that introduces the change thus impacting the performance of the transaction.
- Asynchronous Autolog CDC: this implementation method requires a staging database separate from the source database. Asynchronous Autolog CDC uses the database's redo transport services to transport redo log information from the source database to the staging database¹. Changes are captured at the staging database. The impact to the source system is minimal, but there is some latency between the original transaction and the change being captured.

Oracle Database supports autolog online CDC (see Figure 1) and autolog archive CDC (see Figure 2).

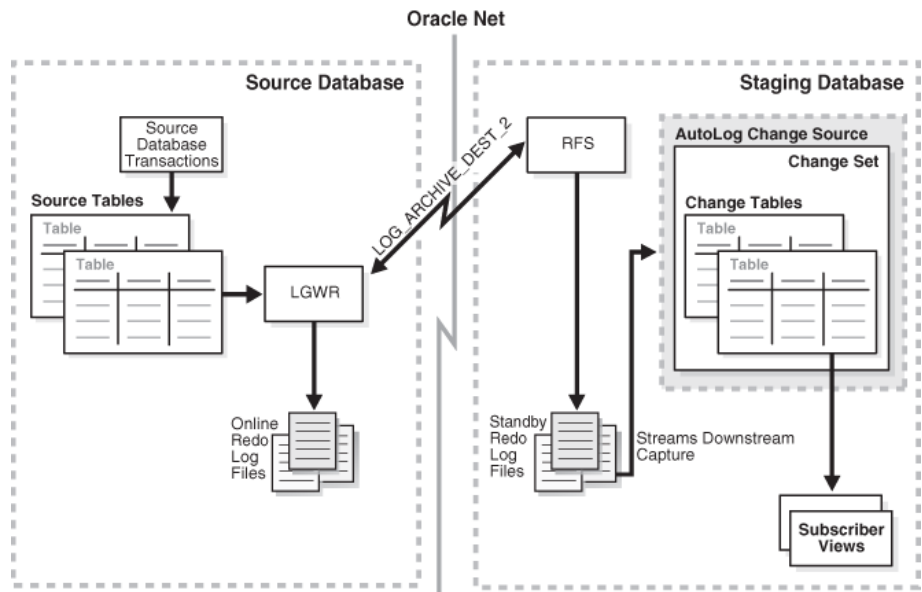


Figure 1: Asynchronous autolog online CDC architecture.

¹ Manually copying the redo log files between the source and the staging database is also supported. You will have to explicitly assign the logs to the staging database in order to do accomplish the change capture. Please refer to the Oracle Streams Concepts and Administration documentation for the details.

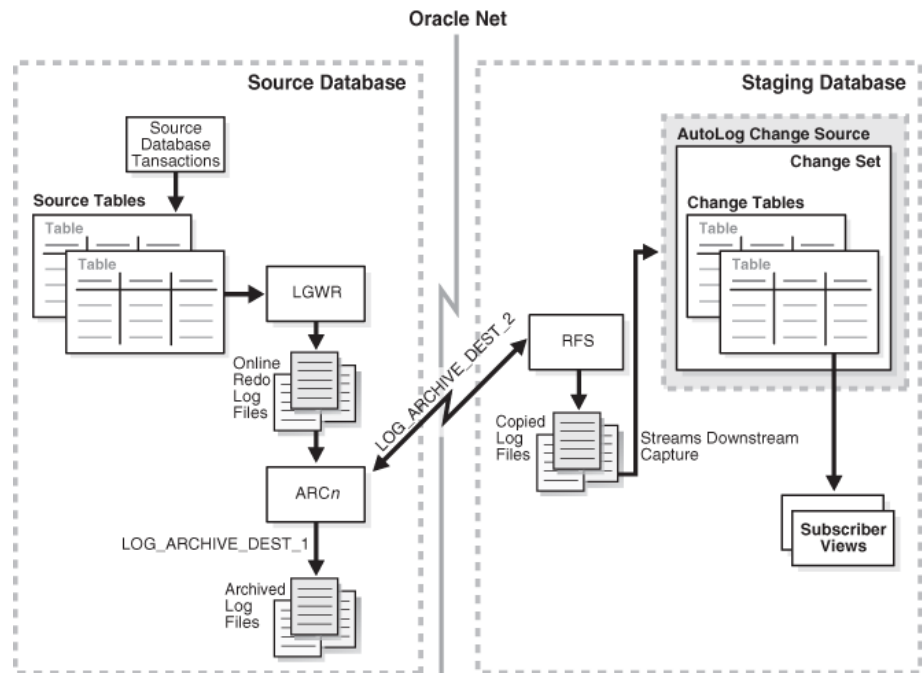


Figure 2: Asynchronous autolog archive CDC architecture.

- **Asynchronous Hotlog CDC:** this implementation method uses the online redo log files to capture changes. The original transaction performs changes to the tables, causing information to be written to the redo log files. A separate process mines the redo log files and captures the changes in the change tables. There is a small latency between the original transaction and the change being captured. For Asynchronous Hotlog CDC the change capture takes place on the same database.
- **Asynchronous Distributed Hotlog CDC:** this implementation method requires a staging database in order to setup the environment. As with the Asynchronous Hotlog CDC method the change capture is performed by a separate process on the source database. The change record is then propagated to the staging database where the change is written to the change tables. Oracle Streams provides the underlying infrastructure for this CDC method. There is some latency between the original transaction and the change being captured.

This cookbook focuses on the asynchronous autolog CDC method, discussing both the online and the archive modes. Asynchronous autolog CDC requires both source and target database to be on the same operating system, and on the same database version. This cookbook assumes Oracle Database 10g Release 2 is used as both the source and the staging database.

Supplemental logging

By default the database only logs sufficient redo information to replay a transaction. Supplemental logging is a database concept to enable logging additional information to the redo log in order to make the logs easier to understand. If you

want to use the change data, then your downstream process becomes much easier knowing the context of the change.

For example, if you change one non-key column in a table with one hundred columns, then Oracle Database will by default only log the changed column. If you want to use this change to update your data warehouse, then it will be difficult to identify the record that was changed. Rather than going through a resource-intensive query to figure out which record has changed, and whether a column was either updated from some value to null, or untouched in the update, you can use supplemental logging in order to log additional column values. Supplemental logging will have some, but typically minimal impact on the performance of update transactions on the source database. Also, depending on the amount of updates relative to other DML, the redo log files will grow somewhat faster.

You can enable supplemental logging either on the database level (e.g. add supplemental logging for all primary keys), or for individual tables. With a supplemental log group you can specify which column values you always want to be logged. Please refer to the Oracle Database documentation for more details.

ENVIRONMENT

The source system for this cookbook is an Oracle Database 10g Release 2, version 10.2.0.3. The source database instance is called ALCDCSRC. This cookbook assumes that you have installed the database software and created a database. Of course you do not have to use the same database name, but then you will have to modify some statements in this cookbook accordingly. Enterprise Edition software is required.

The staging system is a different Oracle Database 10g Release 2 database, version 10.2.0.3, which may run on a different server (note that the operating system for the source database and the target database must be the same). The staging database is called ALCDCSTG. This cookbook assumes that you installed the database software and created a database. Enterprise Edition software is required.

SYS passwords for the source and the staging database must be the same.

Connection settings

In order to use autolog CDC, the source database has to be able to communicate with the target database using TNS connect information.

For the source environment, edit the file <oracle database home>/network/admin/tnsnames.ora and add an entry for the staging database. Alternatively, run the Oracle Net Configuration Assistant (netca) and add a Local Net Service Name configuration.

For example, you may have to add a tnsnames entry for the staging database that looks like this:

```

ALCDCSTG =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
        (HOST = <host name>) (PORT = <listener port staging db>)
      )
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = alcdcstg)
    )
  )

```

Verify the connect information you specified by either using SQL Plus (out of the same Oracle home in which you edited the tnsnames.ora file) using a connection to the remote database as SYSTEM, or use the test button in the Oracle Net Configuration Assistant.

Tip: you may have to verify the value for NAMES.DEFAULT_DOMAIN setting in <oracle database home>/network/admin/sqlnet.ora. You may have to include the value for this parameter in the naming of your tnsnames.ora entry. Refer to the Oracle Database Net Services Reference for details if you cannot get this step to work.

Required initialization parameters

A number of database initialization parameters are mandatory in order to setup asynchronous autolog CDC. The settings below are minimum settings and assume defaults for parameter values not mentioned. Higher parameter values are typically fine. For optimal performance and/or in heavily loaded environments you should increase values appropriately in order to meet your performance needs. Please refer to the Change Data Capture chapter in the Oracle Database Data Warehousing Guide for more details.

The difference in the setup between autolog online CDC and autolog archive CDC is primarily the initialization parameters. This section distinguishes initialization parameters for the autolog online CDC setup versus the autolog archive CDC setup. Use only the relevant subsection that you plan to implement.

Autolog online

Use this subsection only if you plan to setup autolog online CDC.

The easiest way to configure autolog CDC is to first configure the staging database, because the source database references the staging database. Please make sure that the sys passwords on the source and the staging database are identical.

On the 10.2 **staging** database, set the following initialization parameters:

```

compatible='10.2.0'
global_names=TRUE
java_pool_size=50M
log_archive_dest_1='LOCATION=<path to STAGING archive logs>
mandatory reopen=5 valid_for=(online_logfile,primary_role)'
log_archive_dest_2='LOCATION=<path to archive logs that arrive
from SOURCE> mandatory
valid_for=(standby_logfile,primary_role)'
remote_login_passwordfile=shared
streams_pool_size=50M

```

From a pure autolog CDC perspective you do not have to specify an online archive log destination for your staging database. However, as a best practice, and in a production environment, you should do so anyway. Note that either the directory location for the archive log files from the staging database must be different from the directory location for the archive log files that arrive from the source database, or you have to use a different prefix.

You can verify the value for an initialization parameter using the “show parameter” feature in SQL Plus. For example, connect to the database as SYS (as SYSDBA), and type:

```

SQL> show parameter global

```

NAME	TYPE	VALUE
global_context_pool_size	string	
global_names	boolean	TRUE

```

SQL>

```

To change parameter values, you can use the alter system command, for example (connect as SYS, as SYSDBA):

```

alter system set global_names=true scope=spfile ;

```

Due to the scope=spfile portion of the command you have to restart the database to implement the new setting. If you use a pfile, then you can modify the pfile and restart the database pointing to the updated pfile (tip: backup the old pfile).

Your staging database has to run in archive log mode. If you are currently not running in archive log mode, then make sure you set the required archive log initialization parameters as specified before, for example (connect as SYS, as SYSDBA):

```

alter system set log_archive_dest_1 =
'location=/private/oracle/oradata/alcdstg/archive mandatory
reopen=5 valid_for=(online_logfile,primary_role)' scope=spfile
;

```



```
alter system set log_archive_dest_2 =
'location=/private/oracle/oradata/alcdcsstg/alcdcsrc_archive
mandatory valid_for=(standby_logfile,primary_role) '
scope=spfile ;
```

You then have to cleanly shutdown the database in order to enable archivelog mode:

```
shutdown immediate

startup mount

alter database archivelog ;

alter database open ;
```

Verify to make sure you run in archivelog mode:

```
alter system switch logfile ;
```

Check the archive log folder on the file system to make sure a log file was just archived. Do not continue until you successfully run in archivelog mode on the staging database.

You now have to add standby redo log files on the staging database. In order to find out how many and what size standby redo log files you need, you have to logon to the **source** database (SYS as SYSDBA) and run the following query:

```
select group#, bytes/1024/1024 size_mb
from v$log ;
```

If you find n redo log groups of size x on the source database, then you will have to add at least $n+1$ standby redo log groups of size x on the staging database. For example, a default database has three redo log groups of 50MB each. The example below shows the steps for such a default database. Modify the example accordingly if you do not use a default database.

Logon to the **staging** database, using SYS as SYSDBA, and find out what is the first available group number you can use for the standby redo logs:

```
select max(group#)
from v$log ;
```

Start adding standby redo log groups using $\max(\text{group\#}) + 1$ (you cannot leave any gaps in the sequence). The example below assumes that $\max(\text{group\#})$ on the staging database returned 3. Modify the example accordingly if your environment is different. You may also have multiple log files in a logfile group, in which case you have to point to multiple standby redo log files per group. Please refer to the Change Data Capture documentation for more details and an example.

```
alter database add standby logfile group 4
('<path to standby redo log + file name 1>') size 50M
/
```

```

alter database add standby logfile group 5
('<path to standby redo log + file name 2>') size 50M
/

alter database add standby logfile group 6
('<path to standby redo log + file name 3>') size 50M
/

alter database add standby logfile group 7
('<path to standby redo log + file name 4>') size 50M
/

```

On the 10.2 **source** database, you have to set the following initialization parameters:

```

compatible='10.2.0'
log_archive_dest_1='LOCATION=<full path>'
log_archive_dest_2='SERVICE=alcdcstg lgwr async optional
noregister reopen=5 valid_for=(online_logfile,primary_role)'
remote_login_passwordfile=shared

```

Your source database has to run in archive log mode. If you are currently not running in archive log mode, then make sure you set the required archive log initialization parameters as specified and cleanly shutdown the database in order to enable archivelog mode:

```

shutdown immediate

startup mount

alter database archivelog ;

alter database open ;

```

Verify to make sure you run in archivelog mode:

```

alter system switch logfile ;

```

Check the archivelog folder on the file system to make sure a logfile was just archived. An archive log file must also have been propagated to the staging database. Do not continue until you successfully run in archivelog mode on the source database.

Autolog archive

Use this subsection only if you plan to setup autolog archive CDC.

The easiest way to configure autolog CDC is to first configure the staging database, because the source database references the staging database. Please make sure that the sys passwords on the source and the staging database are identical.

On the 10.2 **staging** database, set the following initialization parameters:

```

compatible='10.2.0'
global_names=TRUE
java_pool_size=50M
log_archive_dest_1='LOCATION=<path to STAGING archive logs>
mandatory reopen=5 valid_for=(online_logfile,primary_role)'
log_archive_dest_2='LOCATION=<path to archive logs that arrive
from SOURCE> mandatory
valid_for=(standby_logfile,primary_role)'
remote_login_passwordfile=shared
streams_pool_size=50M

```

From a pure autolog CDC perspective you do not have to specify an online archive log destination for your staging database. However, as a best practice, and in a production environment, you should do so anyway. Note that either the directory location for the archive log files from the staging database must be different from the directory location for the archive log files that arrive from the source database, or you have to use a different prefix.

You can verify the value for an initialization parameter using the “show parameter” feature in SQL Plus. For example, connect to the database as SYS (as SYSDBA), and type:

```

SQL> show parameter global

```

NAME	TYPE	VALUE
global_context_pool_size	string	
global_names	boolean	TRUE

```

SQL>

```

To change parameter values, you can use the alter system command, for example (connect as SYS, as SYSDBA):

```

alter system set global_names=true scope=spfile ;

```

Due to the scope=spfile portion of the command you have to restart the database to implement the new setting. If you use a pfile, then you can modify the pfile and restart the database pointing to the updated pfile (tip: backup the old pfile).

Your staging database has to run in archive log mode. If you are currently not running in archive log mode, then make sure you set the required archive log initialization parameters as specified before, for example (connect as SYS, as SYSDBA):

```

alter system set log_archive_dest_1 =
'location=/private/oracle/oradata/alcdstg/archive mandatory
reopen=5 valid_for=(online_logfile,primary_role)' scope=spfile
;

```

```

alter system set
log_archive_dest_2='location=/private/oracle/oradata/alcdcsstg/a
lcdcsrc_archive mandatory
valid_for=(standby_logfile,primary_role)' scope=spfile ;

```

You then have to cleanly shutdown the database in order to enable archivelog mode:

```

shutdown immediate

startup mount

alter database archivelog ;

alter database open ;

```

Verify to make sure you run in archivelog mode:

```

alter system switch logfile ;

```

Check the archivelog folder on the file system to make sure a logfile was just archived. Do not continue until you successfully run in archivelog mode on the staging database.

On the 10.2 **source** database, you have to set the following initialization parameters:

```

compatible='10.2.0'
log_archive_dest_1='LOCATION=<full path>'
log_archive_dest_2='SERVICE=alcdcsstg arch optional noregister
reopen=5 template=<directory path>/%t_%s_%r.dbf'
remote_login_passwordfile=shared

```

Note that the <directory path> value for log_archive_dest_2 has to be identical to the value of <path to archive logs that arrive from SOURCE> you specified for log_archive_dest_2 on the staging database.

Your source database has to run in archive log mode. If you are currently not running in archive log mode, then make sure you set the required archive log initialization parameters as specified before and cleanly shutdown the database in order to enable archivelog mode:

```

shutdown immediate

startup mount

alter database archivelog ;

alter database open ;

```

Verify to make sure you run in archivelog mode:

```

alter system switch logfile ;

```

Check the archivelog folder on the file system to make sure a logfile was just archived. The logfile must also be propagated to the specified folder for the staging database. Do not continue until you successfully run in archivelog mode on the source database.

SET UP THE DATABASE USERS

The following sections are again applicable to both autolog online and autolog archive CDC.

On the source database, you will create one database user: the user who owns the source tables, CDC_SOURCE. On the staging database you will create two database users: (1) the publisher on the staging database, CDC_STG_PUB, and (2) the subscriber on the staging database, CDC_STG_USER.

Source Oracle Database 10g Release 2

Connect to the **source** database using SYS as SYSDBA, and execute the following statements in order to create the users with the right privileges (note that you should choose your own secret password):

```
create user cdc_source
identified by cdc_source
default tablespace users
temporary tablespace temp ;

grant connect, resource, select any table to cdc_source ;
```

The select any table privilege is not required, but will in this case be taken advantage of by using create table as select * from.

Staging Oracle Database 10g Release 2

Connect to the **staging** database using SYS as SYSDBA, and execute the following statements in order to create the users with the necessary privileges (note again that you should use your own secret passwords):

```
create user cdc_stg_pub
identified by cdc_stg_pub
default tablespace users
temporary tablespace temp
quota unlimited on system
quota unlimited on users
quota unlimited on sysaux ;

grant create session, create table, create tablespace, create
sequence, select_catalog_role, execute_catalog_role, dba to
cdc_stg_pub ;

grant unlimited tablespace to cdc_stg_pub ;

grant execute on dbms_cdc_publish to cdc_stg_pub ;
```

```

execute dbms_streams_auth.grant_admin_privilege(grantee =>
'cdc_stg_pub');

create user cdc_stg_user
identified by cdc_stg_user
default tablespace users
temporary tablespace temp ;

grant connect, resource to cdc_stg_user ;

```

Make sure you use a default tablespace for your database users in order to prevent database objects from being created in the SYSAUX tablespace.

SET UP AND ENABLE CDC

First you have to create the source tables and define the supplemental logging settings on the source database. Once that is completed, you setup the entire CDC configuration by using PL/SQL calls on the staging database.

Note that once you started using CDC you should continue to use CDC calls to make modifications to the setup. Directly using calls to the Oracle Streams administration packages to modify the underlying Streams setup is not supported for a CDC setup, and can lead to inconsistencies in the underlying definitions supporting CDC.

The statements below include a lot of data dictionary select statements that you can use to make sure that the underlying setup is correct and help you understand the definitions that get created. Of course running these select statements is not required to successfully setup CDC.

Create the source tables

Connect to user CDC_SOURCE at the **source** database. Issue the following commands:

```

create table emp as select * from scott.emp ;

create table dept as select * from scott.dept ;

alter table emp add supplemental log data (all) columns ;

alter table dept add supplemental log data (all) columns ;

```

Next, connect to SYS as SYSDBA on the source database and issue the following commands:

```

set serveroutput on

variable f_scn number;

```

```

begin
    :f_scn := 0;
    dbms_capture_adm.build(:f_scn);
    dbms_output.put_line('The first_scn value is ' || :f_scn);
end ;
/

```

Record the value for f_scn. You will need that value later.

```

begin
    dbms_capture_adm.prepare_table_instantiation(table_name =>
'cdc_source.emp') ;
    dbms_capture_adm.prepare_table_instantiation(table_name =>
'cdc_source.dept') ;
end ;
/

```

Finally, find out the value for global_name on the source database:

```
select global_name from global_name ;
```

Set up CDC: create the change tables

Connect to the CDC publisher on the **staging** database, CDC_STG_PUB and issue the following command to create the autolog change source on the staging database.

For autolog online CDC, use:

```

begin
    dbms_cdc_publish.create_autolog_change_source(
        change_source_name => 'emp_dept_src',
        description         => 'EMP and DEPT source',
        source_database     => '<global name for alcdcsrc>',
        first_scn           => <first scn>,
        online_log          => 'y') ;
end;
/

```

For autolog archive CDC, use:

```

begin
    dbms_cdc_publish.create_autolog_change_source(
        change_source_name => 'emp_dept_src',
        description         => 'EMP and DEPT source',
        source_database     => '<global name for alcdcsrc>',
        first_scn           => <first scn>) ;
end;
/

```

The `source_database` parameter has to have the `global_name` for the source database. `first_scn` must be the value you captured from the source database.

Verify the change source definition that was created on the staging database:

```
select source_name, source_description, source_type
, source_database
from change_sources
where source_name = 'EMP_DEPT_SRC' ;
```

This query should return one record with the change source you just created. If you use autolog online CDC, then you will see type `AUTOLOG ONLINE`. For autolog archive CDC you will see type `AUTOLOG`.

Next, as `CDC_STG_PUB` user, create a change set on the staging database. This statement will create an associated, still disabled, streams apply process, an apply queue and apply queue table.

```
begin
  dbms_cdc_publish.create_change_set (
    change_set_name      => 'emp_dept_set',
    description          => 'EMP and DEPT change set',
    change_source_name   => 'emp_dept_src',
    stop_on_ddl          => 'y') ;
end ;
/
```

Verify the change set definition that was created on the staging database:

```
select set_name, set_description, change_source_name
, apply_name, queue_name, queue_table_name
from change_sets
where publisher = 'CDC_STG_PUB'
and set_name = 'EMP_DEPT_SET' ;
```

This query should return one record with the change set you just created.

Check the underlying Streams definition that was created as a result of the call to the CDC procedure:

```
select app.apply_name, q.name, app.status, qt.queue_table
from dba_apply app
, dba_queues q
, dba_queue_tables qt
where app.apply_user = 'CDC_STG_PUB'
and q.owner = 'CDC_STG_PUB'
and qt.owner = 'CDC_STG_PUB'
and q.name = app.queue_name
and qt.queue_table = q.queue_table
and app.apply_name like '%EMP_DEPT%' ;
```


The result of this query shows a disabled Streams apply process with its details.

Still as CDC_STG_PUB user, create the change tables to capture the changes from the source tables. This creates the Streams apply rules as well as the Streams capture rules on the staging database.

```
begin
  dbms_cdc_publish.create_change_table(
    owner => 'cdc_stg_pub',
    change_table_name => 'emp_ct',
    change_set_name => 'emp_dept_set',
    source_schema => 'cdc_source',
    source_table => 'emp',
    column_type_list => 'empno number(4), ename varchar2(10),
job varchar2(9), mgr number(4), sal number(7,2), comm
number(7,2), deptno number(2)',
    capture_values => 'both',
    rs_id => 'y',
    row_id => 'n',
    user_id => 'n',
    timestamp => 'y',
    object_id => 'n',
    source_colmap => 'n',
    target_colmap => 'y',
    options_string => null) ;
end ;
/

grant select on emp_ct to cdc_stg_user ;
```

```

begin
    dbms_cdc_publish.create_change_table(
        owner => 'cdc_stg_pub',
        change_table_name => 'dept_ct',
        change_set_name => 'emp_dept_set',
        source_schema => 'cdc_source',
        source_table => 'dept',
        column_type_list => 'deptno number(2), dname varchar2(14),
loc varchar2(13)',
        capture_values => 'both',
        rs_id => 'y',
        row_id => 'n',
        user_id => 'n',
        timestamp => 'y',
        object_id => 'n',
        source_colmap => 'n',
        target_colmap => 'y',
        options_string => null) ;
end ;
/

grant select on dept_ct to cdc_stg_user ;

```

Check the change table definitions on the staging database:

```

select change_table_name, change_set_name
, source_schema_name, source_table_name
from change_tables
where change_table_schema = 'CDC_STG_PUB'
and change_set_name = 'EMP_DEPT_SET'
order by change_table_name ;

```

This query should show you the two change tables you just created and which source tables' changes are going to be captured.

Verify the capture and apply rules definition on the staging database:

```

select streams_name, streams_type, table_owner, table_name
, rule_type, source_database
from dba_streams_table_rules
where rule_owner = 'CDC_STG_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;

```

The result of this query shows eight Streams rules: DML and DDL capture and DML and DDL apply rules for the two source tables.

The setup of the underlying infrastructure is now complete. The capture, propagation and apply processes are still inactive.

Activate CDC

Activate the change set on the staging database by running the following command as CDC_STG_PUB on the staging database:

```
begin
    dbms_cdc_publish.alter_change_set (
        change_set_name => 'emp_dept_set',
        enable_capture   => 'Y') ;
end ;
/
```

Verify that the underlying Streams apply process was enabled:

```
select apply_name, status
from dba_apply
where apply_user = 'CDC_STG_PUB'
and apply_name like '%EMP_DEPT%' ;
```

The value for STATUS must show ENABLED. If it does not, then check the alert.log file and/or any trace files that were generated in the background dump directory for the database.

Explicitly enforce a log switch on the source database. This will send the dictionary to the staging database so that the change data capture can initialize. Logon to the **source** database using SYS as SYSDBA, and issue:

```
alter system switch logfile ;
```

On the staging database you can now follow the progress of the capture:

```
select capture_name, state, total_messages_captured
from v$streams_capture
where capture_name like '%EMP_DEPT%'
/
```

STATE can show several values until it shows CAPTURING CHANGES. Until state is CAPTURING CHANGES you will not see the value for TOTAL_MESSAGES_CAPTURED increase. Other states include:

- WAITING FOR DICTIONARY REDO
- INITIALIZING
- DICTIONARY INITIALIZATION
- MINING (PROCESSED SCN = <some value>)
- LOADING (step <x> of <y>)
- CAPTURING CHANGES

If you use autolog online CDC you can verify whether CDC was correctly activated by running the following query on the staging database:

```
select group#, thread#, sequence#, archived, status from
v$standby_log
/
```

The output should show one active group, for example:

GROUP#	THREAD#	SEQUENCE#	ARC	STATUS
4	1	53	YES	ACTIVE
5	0	0	YES	UNASSIGNED
6	0	0	YES	UNASSIGNED
7	0	0	YES	UNASSIGNED

In a correct autolog online CDC setup one of the groups must be active.

CDC is now active. Changes applied to the source tables using regular DML² will be captured and end up in the change tables at the staging database.

Create a subscription

You will now create a subscription for the changes as the CDC_STG_USER user. Connect to the **staging** database as CDC_STG_USER, and issue:

```
begin
  dbms_cdc_subscribe.create_subscription(
    change_set_name => 'emp_dept_set',
    description      => 'EMP and DEPT change subscription',
    subscription_name => 'emp_dept_sub1') ;
end ;
/

begin
  dbms_cdc_subscribe.subscribe(
    subscription_name => 'emp_dept_sub1',
    source_schema     => 'cdc_source',
    source_table      => 'emp',
    column_list       => 'empno, ename, job, mgr, sal, comm,
deptno',
    subscriber_view   => 'emp_chg_view') ;
end ;
/
```

2 At this point direct path DML without enforced logging and implicit DML as part of partition maintenance operations are not supported. Note that these are not common for typical OLTP applications.

```

begin
    dbms_cdc_subscribe.subscribe(
        subscription_name => 'emp_dept_sub1',
        source_schema     => 'cdc_source',
        source_table       => 'dept',
        column_list        => 'deptno, dname, loc',
        subscriber_view    => 'dept_chg_view') ;
end ;
/

```

Activate the subscription:

```

begin
    dbms_cdc_subscribe.activate_subscription(
        subscription_name => 'emp_dept_sub1') ;
end ;
/

```

Verify the CDC subscription definition on the staging database (still as CDC_STG_USER):

```

select s.subscription_name, s.set_name, s.description
, st.source_schema_name, st.source_table_name, st.view_name
, sc.column_name
from user_subscriptions s
, user_subscribed_tables st
, user_subscribed_columns sc
where s.subscription_name = 'EMP_DEPT_SUB1'
and st.handle = s.handle
and sc.handle = s.handle
and st.source_schema_name = sc.source_schema_name
and st.source_table_name = sc.source_table_name
order by st.source_schema_name, st.source_table_name
, sc.column_name ;

```

The result of this query shows ten records for the ten columns you subscribed to, across two subscriber views.

AUTOLOG CDC IN ACTION

In order to see CDC in action, you will introduce changes to the source tables, verify the data in the change tables and extend and purge the subscription window.

Every time you extend the subscription window you will see more change data, assuming more changes have arrived in the change tables. The extend window functionality moves the end date/time of your subscription window forward implicitly by using the System Commit Number (SCN). The `PURGE_WINDOW` command moves the start data/time forward (again, through the use of the SCN).

The `EXTEND_WINDOW` and `PURGE_WINDOW` commands are independent, and can be called independently.

At any point in time you can purge the change tables. CDC keeps track of the subscriptions and knows whether or not change data is or will still be visible through a subscriber view. Data for which no subscribers exist anymore is removed from the change tables when you purge the change tables.

Perform DML against the source tables

Connect as `CDC_SOURCE` to the source database and introduce the following changes:

```
-- 10% raise for all salesmen:
update emp
set sal = 1.1 * sal
where job = 'SALESMAN' ;

commit ;

-- Hire John Doe in New York as analyst, reporting to KING
insert into emp values
(8000, 'DOE', 'ANALYST', 7839, trunc(sysdate), 4000, null, 10)
/

-- Close the operations department
delete from dept where dname = 'OPERATIONS' ;

commit ;

-- Give all clerks in Dallas a 5% commission
update emp
set comm = 0.05 * sal
where job = 'CLERK'
and deptno =
(select deptno from dept where loc = 'DALLAS') ;

commit ;
```

If you are using autolog archive CDC, force a log switch on the **source** database. Connect to SYS as SYSDBA and execute:

```
alter system switch logfile ;
```

Read the captured changes from the change tables

Connect to `CDC_STG_PUB` on the staging database to verify the data in the change tables:

```

select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_ct
order by timestamp$ ;

```

The outcome should show something like³ ⁴:

OP	THIS_TIME	EMPNO	ENAME	SAL	COMM
UO	28-dec-2006 14:48:34	7499	ALLEN	1600	300
UN	28-dec-2006 14:48:34	7499	ALLEN	1760	300
UO	28-dec-2006 14:48:34	7521	WARD	1250	500
UN	28-dec-2006 14:48:34	7521	WARD	1375	500
UO	28-dec-2006 14:48:34	7654	MARTIN	1250	1400
UN	28-dec-2006 14:48:34	7654	MARTIN	1375	1400
UO	28-dec-2006 14:48:34	7844	TURNER	1500	0
UN	28-dec-2006 14:48:34	7844	TURNER	1650	0
I	28-dec-2006 14:48:44	8000	DOE	4000	
UO	28-dec-2006 14:48:59	7369	SMITH	800	
UN	28-dec-2006 14:48:59	7369	SMITH	800	40
UO	28-dec-2006 14:48:59	7876	ADAMS	1100	
UN	28-dec-2006 14:48:59	7876	ADAMS	1100	55

Note that:

- Every update shows up twice, once as UO (Update Old) and once as UN (Update New). You can influence this behavior through the `capture_values` parameter when you create the change tables (allowed values OLD, NEW, BOTH).
- The `timestamp$` column reflects the date/time when the DML operation took place on the source, and is hence relative to the source database time.
- Thanks to the supplemental logging on all columns in the EMP table all values are always captured, even if you don't change them.

-
- 3 How quickly you will see the changes appear in the change table depends on the speed of the database servers you use, the network speed and latency, as well as the amount of memory you configured the database to use. For autolog online CDC, once up and running, all changes can be captured and appear in the change tables within seconds after the transaction commits (depending on the size of the transaction; very large transactions modifying thousands or even millions of records would obviously take longer to capture entirely). For autolog archive CDC you have to wait for a log switch after which the staging database will start mining the log. As a result the latency is likely much longer using autolog archive CDC.
- 4 The `timestamp$` column is available because during the creation of the change tables the `timestamp` parameter was set to 'y'. If you don't set this parameter to 'y' the `timestamp$` column will not be available in the change table.

```

select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_ct
order by timestamp$ ;

```

The query result should show:

OP	THIS_TIME	DEPTNO	DNAME	LOC
D	28-dec-2006 14:48:47	40	OPERATIONS	BOSTON

Extend the subscription window

As a subscriber to change data, you will now extend your subscription window in order to see the change data through your subscription views. Connect to the staging database as CDC_STG_USER and execute the following statement:

```

begin
    dbms_cdc_subscribe.extend_window(
        subscription_name => 'emp_dept_sub1' ) ;
end ;
/

```

Verify the data visible through the change views:

```

select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_chg_view
order by timestamp$ ;

```

The query should show a result similar to the following:

OP	THIS_TIME	EMPNO	ENAME	SAL	COMM
UO	28-dec-2006 14:48:34	7499	ALLEN	1600	300
UN	28-dec-2006 14:48:34	7499	ALLEN	1760	300
UO	28-dec-2006 14:48:34	7521	WARD	1250	500
UN	28-dec-2006 14:48:34	7521	WARD	1375	500
UO	28-dec-2006 14:48:34	7654	MARTIN	1250	1400
UN	28-dec-2006 14:48:34	7654	MARTIN	1375	1400
UO	28-dec-2006 14:48:34	7844	TURNER	1500	0
UN	28-dec-2006 14:48:34	7844	TURNER	1650	0
I	28-dec-2006 14:48:44	8000	DOE	4000	
UO	28-dec-2006 14:48:59	7369	SMITH	800	
UN	28-dec-2006 14:48:59	7369	SMITH	800	40
UO	28-dec-2006 14:48:59	7876	ADAMS	1100	
UN	28-dec-2006 14:48:59	7876	ADAMS	1100	55

Query the changes on the departments table.

```
select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_chg_view
order by timestamp$ ;
```

This query should show:

OP	THIS_TIME	DEPTNO	DNAME	LOC
D	28-dec-2006 14:48:47	40	OPERATIONS	BOSTON

Purge the subscription window

In a production situation individual subscribers will consume the changes they are interested in. Once done they will purge the subscription window to indicate that the change data is not needed anymore.

For example in a near real-time data warehouse, every 5 minutes you could extend the subscription window, run a SQL statement to load the data from the subscriber view into target data warehouse tables and purge the subscription window.

Logon to the staging database as CDC_STG_USER, and run:

```
begin
    dbms_cdc_subscribe.purge_window(
        subscription_name => 'emp_dept_sub1' ) ;
end ;
/
```

Verify that the change views don't contain any data anymore:

```
select * from emp_chg_view ;

select * from dept_chg_view ;
```

Both queries should return no records.

Purge the change tables

There are 3 different levels of granularity at which you can purge the change tables:

- 1) Per table: use `DBMS_CDC_PUBLISH.PURGE_CHANGE_TABLE`.
- 2) Per change set: use `DBMS_CDC_PUBLISH.PURGE_CHANGE_SET`.
- 3) For the entire staging database: use `DBMS_CDC_PUBLISH.PURGE`⁵.

5 As part of the setup CDC will automatically schedule a call to `DBMS_CDC_PUBLISH.PURGE` and execute it once every 24 hours. This job is visible through the `DBA_JOBS` data dictionary view and can be modified using calls to `DBMS_JOB` or through the Enterprise Manager console.

You will purge the change set. Connect to the **staging** database as CDC_STG_PUB and issue the following statement:

```
begin
  dbms_cdc_publish.purge_change_set(
    change_set_name => 'emp_dept_set') ;
end ;
/
```

Verify that the data previously visible through the change views has disappeared:

```
select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_ct
order by timestamp$ ;
```

Run:

```
select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_ct
order by timestamp$ ;
```

Both queries should not retrieve any rows.

Introduce more DML changes

Connect as CDC_SOURCE on the **source** database and introduce the following changes:

```
-- Open a marketing department in San Francisco
insert into dept
values (50, 'MARKETING', 'SAN FRANCISCO') ;

commit ;

-- Move John Doe under manager Clark
update emp set mgr = 7782 where empno = 8000 ;

commit ;

-- Lay off the clerks in Dallas
delete from emp where job = 'CLERK' and deptno =
(select deptno from dept where loc = 'DALLAS') ;

commit ;
```

If you are using autolog archive CDC, force a log switch on the source database. Connect to SYS as SYSDBA and execute:

```
alter system switch logfile ;
```

Extend the subscription window again

Connect as CDC_STG_USER to the **staging** database and execute the following statement:

```
begin
  dbms_cdc_subscribe.extend_window(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify the data visible through the change views:

```
select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, mgr, comm
from emp_chg_view
order by timestamp$ ;
```

The result should show something similar to⁶:

OP	THIS_TIME	EMPNO	ENAME	MGR	COMM
UN	29-dec-2006 13:45:23	8000	DOE	7782	
UO	29-dec-2006 13:45:23	8000	DOE	7839	
D	29-dec-2006 13:45:32	7369	SMITH	7902	40
D	29-dec-2006 13:45:32	7876	ADAMS	7788	55

```
select operation$ operation
, to_char(timestamp$, 'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_chg_view
order by timestamp$ ;
```

This query should show:

OP	THIS_TIME	DEPTNO	DNAME	LOC
I	29-dec-2006 13:45:16	50	MARKETING	SAN FRANCISCO

From here you would again consume the change data, purge the subscription window, and start all over again.

FINAL NOTES

CDC is built on top of the Oracle Streams infrastructure. In general, restrictions in the Streams infrastructure, such as limited support for certain data types, for

⁶ If you quickly extended the window after you performed the update or switched the logfile then you may only see the first two records for the EMP update, and not the records for the delete (the last transaction that was completed). CDC will only show change data for transactions that are guaranteed to be completed, always providing a transactionally consistent view for all data in a change set.

example binary data types, apply to CDC as well. In order to tune the CDC processes, you tune the Oracle Streams configuration settings. However, note that you should never use Streams call directly to modify a setup that was created using the CDC functionality.

In Oracle Database 10g Release 2 CDC does not support direct path data load operations against the data source without enforced logging nor does it support implicit DML through partition maintenance operations. In a traditional OLTP environment these types of operations are rare.

For more details, please refer to the Oracle Database Data Warehousing Guide. There is a dedicated chapter about Change Data Capture.

CONCLUSION

Oracle's Change Data Capture is extremely powerful functionality that can capture changes against database tables without making any changes to the applications that work with the tables. Autolog CDC absolutely minimizes the performance impact on the source system. CDC is a key enabling feature for near real-time data warehousing, but can be applied to any environment that requires capturing incremental changes out of an Oracle database.

APPENDIX A CLEANUP THE ENVIRONMENT

To cleanup the environment you go through the reverse steps you went through to setup the environment. This appendix shows you how to get back to your original environment. The data dictionary select statements are there to verify whether the cleanup is correct, and to provide a better understanding of the underlying technology. These queries are not required to be run if you just want to remove the CDC setup.

Drop the subscription

Drop the subscription by connecting to CDC_STG_USER on the staging database and issuing the following statement:

```
begin
    dbms_cdc_subscribe.drop_subscription(
        subscription_name => 'emp_dept_sub1' ) ;
end ;
/
```

Verify that the subscription was dropped:

```
select s.subscription_name, s.set_name, s.description
, st.source_schema_name, st.source_table_name, st.view_name
, sc.column_name
from user_subscriptions s
, user_subscribed_tables st
, user_subscribed_columns sc
where s.subscription_name = 'EMP_DEPT_SUB1'
and st.handle = s.handle
and sc.handle = s.handle
and st.source_schema_name = sc.source_schema_name
and st.source_table_name = sc.source_table_name
order by st.source_schema_name, st.source_table_name
, sc.column_name ;
```

This query should not return any rows. The change views have also been dropped.

Disable CDC

Switch of CDC by disabling the change set. Connect to the staging database as CDC_STG_PUB and run:

```
begin
    dbms_cdc_publish.alter_change_set(
        change_set_name => 'emp_dept_set',
        enable_capture => 'N') ;
end ;
/
```

Verify the Streams apply process definition to make sure it was switched off as a result of running the command:

```
select apply_name, status
from dba_apply
where apply_user = 'CDC_STG_PUB'
and apply_name like '%EMP_DEPT%' ;
```

This query should return status DISABLED for the apply process.

Drop the CDC database objects

First you will drop the change tables on the staging database. Connect to CDC_STG_PUB on the staging database and issue the following commands:

```
begin
    dbms_cdc_publish.drop_change_table(
        owner => 'cdc_stg_pub',
        change_table_name => 'dept_ct',
        force_flag => 'y') ;
end ;
/

begin
    dbms_cdc_publish.drop_change_table(
        owner => 'cdc_stg_pub',
        change_table_name => 'emp_ct',
        force_flag => 'y') ;
end ;
/
```

Verify that both change tables have been dropped correctly:

```
select change_table_name, change_set_name, source_schema_name
, source_table_name
from change_tables
where change_table_schema = 'CDC_STG_PUB'
and change_set_name = 'EMP_DEPT_SET'
order by change_table_name ;
```

This query should not return any rows.

Verify that the capture and apply rules on the staging database have been dropped:

```
select streams_name, streams_type, table_owner, table_name
, rule_type, source_database
from dba_streams_table_rules
where rule_owner = 'CDC_STG_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;
```

This query should not return any rows.

Drop the CDC change set on the staging database, connected as CDC_STG_PUB:

```
begin
    dbms_cdc_publish.drop_change_set(
        change_set_name => 'emp_dept_set') ;
end ;
/
```

Verify that the change set was successfully removed:

```
select set_name, set_description, change_source_name
, apply_name, queue_name, queue_table_name
from change_sets
where publisher = 'CDC_STG_PUB'
and set_name = 'EMP_DEPT_SET' ;
```

This query should not return any rows.

Verify that all Streams apply definitions on the staging database were removed as a result of dropping the change set:

```
select app.apply_name, q.name, app.status, qt.queue_table
from dba_apply app, dba_queues q, dba_queue_tables qt
where app.apply_user = 'CDC_STG_PUB'
and q.owner = 'CDC_STG_PUB'
and qt.owner = 'CDC_STG_PUB'
and q.name = app.queue_name
and qt.queue_table = q.queue_table
and app.apply_name like '%EMP_DEPT%' ;
```

This query should not return any rows.

Finally, drop the autolog change source on the source database, by executing the following statement on the staging database, connected as CDC_STG_PUB:

```
begin
    dbms_cdc_publish.drop_change_source(
        change_source_name => 'emp_dept_src') ;
end ;
/
```

Verify that the change source definition on the staging database has been removed:

```
select source_name, source_description, source_type
, source_database, capture_name, capture_queue_name
, capture_queue_table_name
from change_sources
where publisher = 'CDC_STG_PUB'
and source_name = 'EMP_DEPT_SRC' ;
```

The query should not return any rows.

Drop the source tables

Connect to CDC_SOURCE on the source database, and drop the source tables:

```
drop table dept ;
```

```
drop table emp ;
```

Drop the users

Connect to SYSTEM or SYS on the source database and drop the CDC_SOURCE and CDC_SOURCE_PUB users:

```
drop user cdc_source ;
```

Connect to SYSTEM or SYS on the staging database and drop the CDC_STG_USER and CDC_STG_PUB users:

```
drop user cdc_stg_user ;
```

```
drop user cdc_stg_pub cascade ;7
```

⁷ The cleanup process may leave some Streams rules around, that are dropped as well if you use the drop user cascade call. Alternatively you could use the Streams APIs to drop the rules followed by a drop user without the cascade option.

APPENDIX B KNOWN ISSUES AND WORKAROUNDS

This appendix lists known issues that you may run into, depending on your environment and setup.

High System Commit Numbers (SCNs)

In an existing environment with many transactions on the source system you may run into bug 4649767 because of large system commit numbers. A fix for this problem is available in Oracle Database 10.2.0.2 and higher patch sets. You can request a backport for the problem on 10.2.0.1 through Oracle Support at <http://metalink.oracle.com>.

Queue tables in SYSAUX tablespace

Queue tables are created as a memory overflow for the Oracle Streams processes. The queue tables always end up in the SYSAUX tablespace on an Oracle Database 10g. Bug 5024710 describes this problem and is fixed in Oracle Database 10.2.0.3. You can request a backport for the problem on 10.2.0.1 or 10.2.0.2 through Oracle Support at <http://metalink.oracle.com>.



Asynchronous Change Data Capture Cookbook

January 2007

Author: Mark Van de Wiel

Contributing Authors:

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

oracle.com

Copyright © 2007, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.