

# Postgres to Oracle Data Migration

In order to migrate data from Postgres to Oracle db we have followed below mentioned steps:

1. Enable CDC on Postgres to get WAL.
2. Debezium to read the transaction logs from WAL and pushed as kafka message to kafka topics. Debezium will create a separate topic for each table.
3. Kafka JDBC Sink connector to read messages from above kafka topic and pushed to oracle database.
4. Customized SOM sink connector which will transform the data read from kafka queues pushed by debezium to oracle specific datatypes and map the columns correctly.
5. After following above steps the database records will be in sync then we have to correct the part id's per oracle partitions as in postgres the partition is based on hash where as for oracle it should be range partitions.
6. Manually patch odo-adt-lib order object jars as there was mismatch between structure in 10.2 and 10.4 else you will not be able to see service tree and view context store.
7. Update oss\_activity\_instance table spec\_version\_no if there is mismatch in versioning of ref data between postgres and oracle.
8. Need to update oracle SOM table sequences so that it is per new data inserted from postgres.
9. Re-Index Solar

## **Enable CDC on Postgres**

### **--To copy existing k8 file to local**

```
kubectl -n dop cp dop-postgres-0:/var/lib/pgsql/12/data/pg/postgresql.conf ${HOME}/kpg.conf
```

### **--Enable wal logs in conf**

```
wal_level = logical  
max_replication_slots = 10  
max_wal_senders = 10
```



postgres.conf

### **--To copy local to k8 file**

```
kubectl -n dop cp ${HOME}/kpg.conf dop-postgres-0:/var/lib/pgsql/12/data/pg/postgresql.conf
```

### **--To validate changes**












```
cat ./var/lib/pgsql/12/data/pg/postgresql.conf
```

### **--To restart postgres**

```
k -n dop delete pod dop-postgres-0
```

## Setup of Kafka with debezium and confluent connectors



















1. Copy Debezium postgres connector libraries to kafka lib directory

	debezium-api-1.4.1.Final.jar	2/1/2021 7:21 PM	Executable Jar File	20 KB
	debezium-connector-oracle-1.4.1.Final.jar	2/1/2021 6:17 PM	Executable Jar File	194 KB
	debezium-connector-postgres-1.4.1.Final.jar	2/1/2021 6:17 PM	Executable Jar File	311 KB
	debezium-core-1.4.1.Final.jar	2/1/2021 7:21 PM	Executable Jar File	800 KB
	debezium-ddl-parser-1.4.1.Final.jar	2/1/2021 7:21 PM	Executable Jar File	2,661 KB
	debezium-embedded-1.4.1.Final.jar	2/1/2021 7:21 PM	Executable Jar File	46 KB
	debezium-server-core-1.4.1.Final.jar	2/1/2021 7:21 PM	Executable Jar File	18 KB
	failureaccess-1.0.1.jar	2/1/2021 5:32 PM	Executable Jar File	5 KB
	guava-30.0-jre.jar	2/1/2021 5:33 PM	Executable Jar File	2,792 KB
	postgresql-42.2.14.jar	2/1/2021 5:35 PM	Executable Jar File	911 KB
	protobuf-java-3.8.0.jar	2/1/2021 5:35 PM	Executable Jar File	1,597 KB

2. Copy oracle driver i.e ojdbc6-12.1.0.2.0.jar to kafka lib directory.
3. In Kafka config “connect-distributed.properties” we need to provide the plugin path of “confluentinc-kafka-connect-jdbc” libraries which will also contain our **custom som jdbc connector mapper lib**.

### In Windows:

```
# plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors,  
plugin.path=C:\\Gaurav\\Development_Work\\Telstra\\Debezin\\Software\\confluentinc-kafka-connect-jdbc-10.0.1\\lib
```

	common-utils-6.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	17 KB
	jtds-1.3.1.jar	11/24/2020 5:45 PM	Executable Jar File	311 KB
	mssql-jdbc-8.4.1.jre8.jar	11/24/2020 5:45 PM	Executable Jar File	1,271 KB
	ojdbc6-12.1.0.2.0.jar	1/30/2020 12:09 PM	Executable Jar File	3,606 KB
	ons-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	153 KB
	oracle-jdbc-11.1.0.7.jar	6/5/2020 1:18 AM	Executable Jar File	1,942 KB
	oraclepki-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	304 KB
	orai18n-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	1,625 KB
	osdt_cert-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	206 KB
	osdt_core-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	305 KB
	postgresql-42.2.10.jar	11/24/2020 5:45 PM	Executable Jar File	906 KB
	simplefan-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	32 KB
	slf4j-api-1.7.30.jar	11/24/2020 5:45 PM	Executable Jar File	41 KB
	som-kafka-connect-jdbc-1.0.0.0-SNAPSHOT.jar	2/5/2021 3:55 PM	Executable Jar File	257 KB
	sqlite-jdbc-3.25.2.jar	11/24/2020 5:45 PM	Executable Jar File	6,900 KB
	ucp-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	1,645 KB
	xdb-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	259 KB
	xmlparserv2-19.7.0.0.jar	11/24/2020 5:45 PM	Executable Jar File	1,889 KB

### **Locally Running the servers and connectors:**

1. **Run Zookeeper**  
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
2. **Run Kafka**  
.\bin\windows\kafka-server-start.bat .\config\server.properties
3. **Run kafka connector**  
.\bin\windows\connect-distributed.bat .\config\connect-distributed.properties
4. **To enable debug on kafka connector:-** Edit connect-distributed.bat under bin and add property  
set KAFKA\_DEBUG=Y

Default debug port will be "5005" mentioned in "kafka-run-class.bat"

```
22 SetLocal
23 rem Using pushd popd to set BASE_DIR to the absolute path
24 pushd %~dp0..\..\
25 set BASE_DIR=%CD%
26 set KAFKA_DEBUG=Y
27 popd
28
```

In K8s environment a new pod is created which will automatically run all the above steps so we need to just enable the connector through postman

telstra-kafka-pg-to-oracle-migration-0	2/2	Running	0	7d23h
--	-----	---------	---	-------

### **To check kafka topics**

bin/kafka-topics.sh --list --zookeeper localhost:2181

## **Enable the Debezium connector:**

Once zookeeper, Kafka and confluent connect is up. We need to start the debezium connector which will start reading the data from postgres WAL and push to kafka topics.

**URL:-** localhost:8083/connectors

8083 is connect-distributed.properties default port.

### **Request:**

```
{
  "name": "SOM-debezium-postgre",
  "config": {
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "database.hostname": "10.77.66.66",
    "database.port": "30432",
    "database.user": "postgres",
    "database.password": "postgres",
    "database.dbname" : "paasdb",
    "database.server.name": "SOM",
    "plugin.name": "pgoutput",
    "decimal.handling.mode": "precise",
    "time.precision.mode": "connect",
    "table.exclude.list": "ossdb01db.oss_sync_locks,ossdb01db.oss_sync_buckets,ossdb01db.oss_j
ms_db_sync,ossdb01db.oss_sequence,ossdb01db.act,ossdb01db.oss_repository_request,ossdb01db.act
_name2_spec_mapping,ossdb01db.implementation,ossdb01db.oss_parcel,ossdb01db.oss_parcel_item,os
sdb01db.tlsb2b_correlation,lomsusr.adaptor_message_log,lomsusr.tlsb2b_correlation",
    "schema.include.list": "ossdb01db,lomsusr"
  }
}
```

Provide postgres db details

Schema.include.list : It will contains list of schema from which debezium will read the data.

Table.exclude.list:- It will contain set of tables which will not be read by debezium and will not be migrated.

**Note:** Need to use table.include.list instead of exclude as If someone create temp tables in prod then migration can throw error as it will not find tables in oracle.

**Use property "snapshot.mode = never" to indicate debezium to not load existing data and read only new transactional data**

## To check status of Debezium connector

**Hit API:** localhost:8083/connectors/SOM-debezium-postgre/status

Use name of the connector in the url to check the status

GET localhost:8083/connectors/SOM-debezium-postgre/status

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE
-----	-------

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "name": "SOM-debezium-postgre",
3   "connector": {
4     "state": "RUNNING",
5     "worker_id": "10.197.16.117:8083"
6   },
7   "tasks": [
8     {
9       "id": 0,
10      "state": "RUNNING",
11      "worker_id": "10.197.16.117:8083"
12    }
13  ]
14 }
```

Once debezium is running it will read the WAL logs and push the messages to kafka topics based on schema.tablename

Clusters

- SOM-Kafka
  - Brokers
  - Topics
    - connect-configs
    - connect-offsets
    - connect-status
    - \_consumer\_offsets
    - SOM.lomsusr.adaptor\_message\_log
    - SOM.lomsusr.device\_item
    - SOM.lomsusr.device\_management\_order
    - SOM.lomsusr.dispatched\_product\_info
    - SOM.lomsusr.dispatch\_item
    - SOM.lomsusr.logistics\_order
    - SOM.lomsusr.logistics\_order\_item
    - SOM.ossdb01db.act\_ge\_property
    - SOM.ossdb01db.act\_hi\_taskinst
    - SOM.ossdb01db.act\_hi\_varinst
    - SOM.ossdb01db.act\_ru\_task
    - SOM.ossdb01db.act\_ru\_variable
    - SOM.ossdb01db.odo\_order\_attributes
    - SOM.ossdb01db.oss\_activity\_instance
    - SOM.ossdb01db.oss\_activity\_schedule
    - SOM.ossdb01db.oss\_attribute\_store
    - SOM.ossdb01db.oss\_baseline
    - SOM.ossdb01db.oss\_batch
    - SOM.ossdb01db.oss\_batch\_item
    - SOM.ossdb01db.oss\_context\_store
    - SOM.ossdb01db.oss\_correlation

Properties Data Partitions Config

Partition	Offset	Key	Message	Timestamp
0	0	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	1	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	2	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	3	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	4	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	5	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	6	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	7	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	8	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	9	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	10	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	11	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	12	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	13	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	14	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	15	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	16	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	17	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	18	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	19	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	20	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	21	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	22	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	23	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	24	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	25	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	26	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	27	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52
0	28	7B2273636865D61223A7B2...	7B2273636865D61223A7B22...	2021-02-09 11:07:52

## **Enable the JDBC Sink Connector**

Once data is pushed to kafka queues by debezium. We need to enable the JDBC Sink connectors which will perform the below operations:

1. Read the data from kafka topics based on topic configuration in connector.
2. Transform the data per oracle db requirement.
3. Push/Merge the data to Oracle tables.

**URL:** localhost:8083/connectors

### **Enable SOM JDBC Sink**

**Request:**

```
{  
  "name": "som-jdbc-sink",  
  "config": {  
    "connector.class": "com.amdocs.oss.som.kafka.connectors.jdbc.JDBCSinkConnector",  
    "tasks.max": "2",  
    "topics.regex": "SOM.ossdb01db.*",  
    "connection.url": "jdbc:oracle:thin:@indlin5445:1521/TBDZ5445",  
    "connection.user": "SOMUSR01DB",  
    "connection.password": "SOMUSR01DB",  
    "transforms": "unwrap",  
    "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",  
    "transforms.unwrap.drop.tombstones": "false",  
    "auto.create": "false",  
    "auto.evolve": "false",  
    "insert.mode": "upsert",  
    "delete.enabled": "false",  
    "pk.mode": "record_key",  
    "quote.sql.identifiers": "never"  
  }  
}
```

### **Properties:**

Topics.regex : "SOM.ossdb01db.\*" indicates that the connector will read only kafka topics which matched this pattern. So it will read data of AFF/Runtime db only.

Tasks.max : Defines the number of task connector will execute in parallel.

Connector.class : Define the custom connector class which will transform the postgres data and insert into oracle

Insert.mode : It will upsert as we want to update the record if it is already existing instead of creating new one. So, this mode will use merge updates instead of insert statement.

### **Enable LOMS JDBC Sink**

```
{
  "name": "loms-jdbc-sink",
  "config": {
    "connector.class": "com.amdocs.oss.som.kafka.connectors.jdbc.JDBCSinkConnector",
    "tasks.max": "2",
    "topics.regex": "SOM.lomsusr.*",
    "connection.url": "jdbc:oracle:thin:@indlin5445:1521/TBDZ5445",
    "connection.user": "lomsusr",
    "connection.password": "lomsusr",
    "transforms": "unwrap",
    "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",
    "transforms.unwrap.drop.tombstones": "false",
    "auto.create": "false",
    "auto.evolve": "false",
    "insert.mode": "upsert",
    "delete.enabled": "false",
    "pk.mode": "record_key",
    "quote.sql.identifiers": "never"
  }
}
```

### **Post Migration Changes:**

After enabling the JDBC sink connectors all the data will be migrated to Oracle Database. Check confluent connector startup logs for any errors.

### **Oracle Table Partitions Updates:**

```
SELECT * FROM all_tab_columns WHERE owner = 'SOMUSR01DB' AND column_name = 'PART_ID'
```

Below tables part\_id column needs to be updated

```
OSS_ACTIVITY_INSTANCE
OSS_EXECUTION_PLAN
OSS_ATTRIBUTE_STORE
OSS_LINKED_PLAN
OSS_REASON
OSS_ACTIVITY_NOTE
OSS_ACTIVITY_SCHEDULE
```

## Update script for part id columns:



Update-Partition.sql

## Select script to fetch incorrect part\_id :



Select-Incorrect-Partitions.sql

## Activity Instance Spec Version Number Updates:

In case there is mismatch between ref database of postgres and oracle i.e activity versioning then service decomposition flow will not load and fail with below error:-

Exception while invoking operation <generateExecutionPlanView>, Error ID: <C282C90D20A2446C9A6219A7E2C3272B>, Failed with message: <EJB Exception: : java.lang.NullPointerException



Activity\_Spec\_version\_update.sql

To fix above error we need to update oss\_activity\_instance table with correct spec version number from ref database of that environment.

To fetch incorrect records having spec version number in oss\_activity\_instance but not in ref db.

Update these spec id with correct ref spec no in oss\_activity\_instance table.

```
SELECT DISTINCT SPEC_VER_ID,SPEC_VER_NUMBER FROM (
select oai.plan_id,oai.id,oai.spec_ver_id,oai.spec_ver_number ,ms.DISPLAY_NAME
,ms.IMPLEMENTATIONTYPE
from oss_activity_instance oai,act_name2_spec_mapping ms
WHERE oai.SPEC_VER_ID = ms.SPEC_ID(+)
AND oai.SPEC_VER_NUMBER = ms.VERSION_NO(+)
)
WHERE DISPLAY_NAME IS null
```

```
SELECT SPEC_ID,DISPLAY_NAME,max(VERSION_NO) FROM act_name2_spec_mapping
GROUP BY SPEC_ID,DISPLAY_NAME
```

Spec Version Id	Spec version No	Activity Name	Ref Max Spec No
37e68427-207f-4fa6-b75b-68faafd648e0	7	Activate MSISDN Mobility	6
8456f638-3acd-4f6e-b48e-7d3c509962b7	11	Reserve SIM Package	10
8a521938-8c07-434b-a1ac-a602043c30f6	7	Complete Mobile Service transition in TRAMAS	6
b64d11c3-edc1-41e1-80da-c9e3afcb1c41	2	Add Device to Consolidated POC Request	1
c8b196dd-2ba6-4222-8256-bda5119e9025	10	Activate SIM Package	9



### **Sync 10.4 and 10.2 context store classes**

Service tree will not be loaded after data migration and you will not be able to query context store. Because the order classes of core have new fields added in 10.4 which needs to be merged in 10.2 as well.

[odo-adt-lib.jar](#) → [com.amdocs.oss.odo.components.ordermanager.internal](#)

**Mail for reference:**



RE\_ Mismatch  
Classes 10\_4 & 10\_2.r

Classes which were patched to 10.2



internal.7z

--Fetch existing jar from the environment

`jar -uf orchcore-be.ear odo-lib/odo-adt-lib.jar`

--Patch attached classes and upload to environment and restart the server.

`jar -xf orchcore-be.ear odo-lib/odo-adt-lib.jar`

## **SOM Oracle Sequences alteration**

After above all the steps we need to alter the sequences to be incremented according to the data migrated from postgres to oracle else the new orders will fail to be created in SOM with error “object already exist”

To fetch postgres current sequences values:-



postgres-fetch-sequences.sql

Alter the oracle sequences based on output from above postgres script

--To check existing oracle sequences and values

```
SELECT SEQUENCE_NAME,  
       LAST_NUMBER  
FROM all_sequences  
WHERE sequence_owner LIKE '%SOMUSR01DB%'
```

--To alter sequences

```
ALTER SEQUENCE slm_plan_id_seq INCREMENT by 12192;
```

Below attached procedure will update sequences according to their table maximum values.



OracleSequenceProcedure.sql

### **Custom som connector Mapper:**

Below changes are accommodated by extending the kafka jdbc sink connector for postgres to oracle data migration.

1. Converted the BigDecimal datatypes of postgres to return integer/number for Oracle.
2. Changed the data type of kafka payload to “**OPTIONAL\_INT32\_SCHEMA**” for BigDecimal fields.
3. Table OSS\_ACTIVITY\_SCHEDULE table has field “**schedule\_constraint**” in postgres while in oracle the column name was “constraint”. So we write the mapping between these fields.
4. Table OSS\_WBS is having extra field “part\_id” in postgres which was not there in oracle. So we did customization to remove the field while data migration to oracle.
5. OSS\_Activity\_instance part\_id update logic is written but needs to be removed as it is migrated to post db script.
6. Tables having datatypes as clob and blob are divided in to two parts first insert with blank clob/blob values then update the payload.
7. alter table oss\_jms\_db\_sync REPLICA IDENTITY FULL;

### **Re-Indexing in Solar**

<https://indlin5445.corp.amdocs.com:8443/search/#/tasks/dataimport//dataimport>

### **DB Clean Up:**



Truncate\_SOM\_Table  
s.sql



Truncate\_LOMS\_Tab  
es.sql

If primary key is not there for table being migrated then sink will throw null pointer.

```
alter table adaptor_message_log add PRIMARY KEY ("oid",message_payload);
```

```
--In oracle before migration disable the FK in LOMS Schema
```

```
alter table DEVICE_ITEM disable constraint PARCEL_ID_FK;
```

### **Queries to fetch row count to match records post migration:-**



LOMS-select-row-co  
unt.sql



SOM-select-row-cou  
nt.sql

```

SELECT 'SC_PROJECT_ORDER_INSTANCE' tn,count(1) FROM SC_PROJECT_ORDER_INSTANCE spoi
union all
SELECT 'OSS_ACTIVITY_INSTANCE' tn,count(1) FROM OSS_ACTIVITY_INSTANCE oai
union all
select 'oss_execution_plan' tn,count(1) from oss_execution_plan oep
union all
select 'oss_linked_plan' tn,count(1) from oss_linked_plan oep
union all
select 'oss_context_store' tn,count(1) from oss_context_store oep
union all
select 'oss_attribute_store' tn,count(1) from oss_attribute_store oep
union all
select 'oss_error' tn,count(1) from oss_error oep
union all
select 'oss_correlation' tn,count(1) from OSS_CORRELATION oc
union all
select 'ODO_ORDER_ITEM' tn,count(1) from ODO_ORDER_ITEM oi
union all
select 'ODO_ORDER_LINE' tn,count(1) from ODO_ORDER_LINE ol
union all
select 'OSS_EXTERNAL_RELATION_PLAN' tn,count(1) from OSS_EXTERNAL_RELATION_PLAN oe
union all
select 'OSS_HISTORY_EVENT' tn,count(1) from OSS_HISTORY_EVENT oh
union all
select 'oss_wbs' tn,count(1) from oss_wbs oh
union all
select 'oss_dep_entity' tn,count(1) from oss_dep_entity
union all
select 'oss_dep_entity_desc' tn,count(1) from oss_dep_entity_desc
union all
select 'oss_dep_group' tn,count(1) from oss_dep_group
union all
select 'oss_dep_group_desc' tn,count(1) from oss_dep_group_desc
union all
select 'oss_dep_payload' tn,count(1) from oss_dep_payload
union all
select 'oss_dep_plan' tn,count(1) from oss_dep_plan
union all
select 'oss_entity_correlation' tn,count(1) from oss_entity_correlation
union all
select 'oss_dpm_visualization' tn,count(1) from oss_dpm_visualization

```