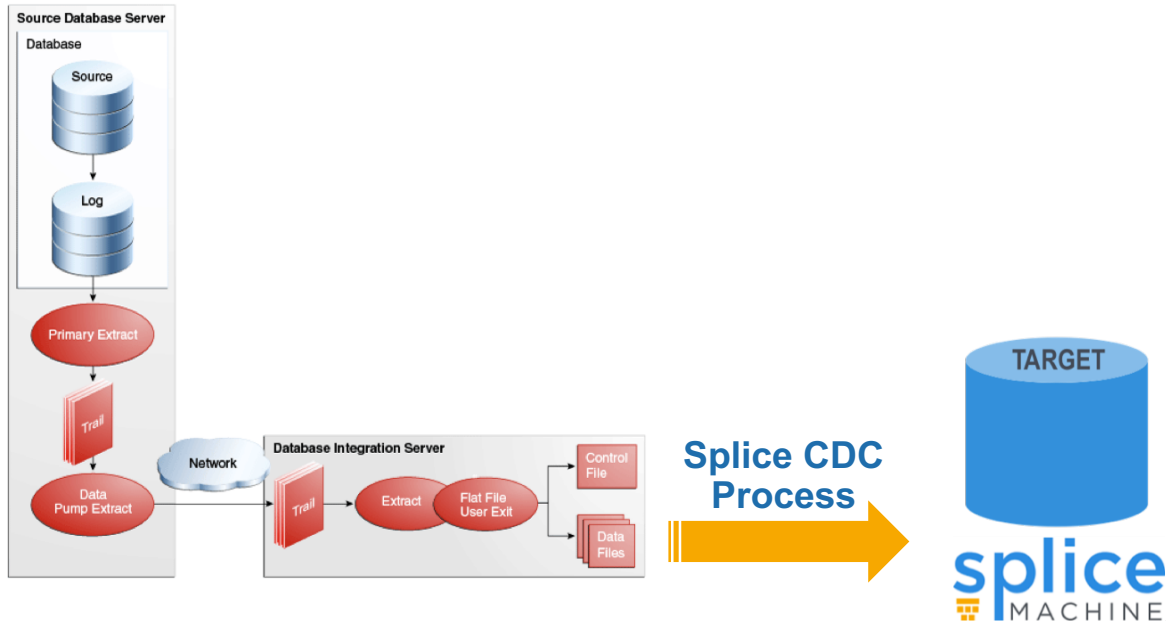
splice MACHINE

Oracle Change Data Capture for Splice Machine

June 2018

Oracle CDC for Splice - Framework

- Splice Machine's CDC solution relies on the "changed" files to be provided by Oracle GoldenGate Adapter for Flat Files.
- The Flat File Adapter outputs transactional data captured by Oracle GoldenGate to rolling flat files that can be used by Splice Machine for processing changed data capture (INSERT, UPDATE and DELETE).
- Oracle CDC for Splice process can be set up as a scheduled job (e.g. cronjob) to periodically check for changed files from Oracle and to process the changes on Splice.



Oracle CDC for Splice - Description

- Configure Oracle Golden Gate Flat File Adapter to include the following METADATA columns in the output files:
  - Position
  - Txind
  - Opcode
  - Timestamp
  - Schema
  - Table

```
dsvwriter.dsv.quotes.datatypes=date
dsvwriter.metacols=position,txind,opcode,timestamp,schema,table
dsvwriter.metacols.txind.fixedlen=1
dsvwriter.metacols.txind.begin.chars=B
dsvwriter.metacols.txind.middle.chars=M
```

Snippet of Flat File Writer properties file

- Both SOURCE Table (in Oracle) and TARGET table (in Splice Machine) must be defined with a PRIMARY KEY
- For each target table in Splice that requires changed data capture from Oracle, we need to create an associated “staging” table. The “staging” table is an intermediary table that stores the raw changed records from Oracle.

Oracle CDC for Splice - Description

- The “staging” table containing the raw changed records from Oracle, is used for processing the changes to the Splice target table.
- The definition (DDL) of the “staging” table must have the following format (in specified order)
  - Position
  - Txind
  - Opcode
  - Timestamp
  - Schema
  - Table
  - Processed
  - <ALL COLUMNS FROM TARGET TABLE>

These are METADATA columns generated by Oracle Golden Gate Flat File Adapter

flag used internally by Splice’s CDC program to track the processing of raw records

- The “staging” table’s PRIMARY KEY must comprise of the following columns:
  - Position, Txind, Opcode, Timestamp

Oracle CDC for Splice – Example DDLs for TARGET and STAGING

METADATA COLUMNS
- Position
- Txind
- Opcode
- Timestamp
- Schema
- Table

PROCESSED Flag

PRIMARY KEY, consisting:
- Position
- Txind
- Opcode
- Timestamp

Corresponding STAGING Table for TARGET Table

```
create table STAGING.shipment_staging_table (
position char(16) not null,
txindi char(1) not null,
opcode char(1) not null,
transtimestamp timestamp not null,
targetschema varchar(20) not null,
targettable varchar(40) not null,
processed char(1) default 'N',
SHIPMENTID VARCHAR(11) NOT NULL,
ALERT VARCHAR(80),
SHIPMODE VARCHAR(30),
PRODUCT_DESCRIPTION VARCHAR(500),
CONSIGNEE VARCHAR(200),
SHIPPER VARCHAR(100),
ARRIVAL_DATE TIMESTAMP,
GROSS_WEIGHT_LB INTEGER,
GROSS_WEIGHT_KG INTEGER,
FOREIGN_PORT VARCHAR(50),
US_PORT VARCHAR(50),
VESSEL_NAME VARCHAR(40),
COUNTRY_OF_ORIGIN VARCHAR(40),
CONSIGNEE_ADDRESS VARCHAR(150),
SHIPPER_ADDRESS VARCHAR(150),
ZIPCODE VARCHAR(20),
NO_OF_CONTAINERS INTEGER,
CONTAINER_NUMBER VARCHAR(200),
CONTAINER_TYPE VARCHAR(80),
QUANTITY INTEGER,
QUANTITY_UNIT VARCHAR(10),
MEASUREMENT INTEGER,
MEASUREMENT_UNIT VARCHAR(5),
BILL_OF_LADING VARCHAR(20),
HOUSE_VS_MASTER CHAR(1),
DISTRIBUTION_PORT VARCHAR(40),
MASTER_BL VARCHAR(20),
VOYAGE_NUMBER VARCHAR(10),
SEAL VARCHAR(300),
SHIP_REGISTERED_IN VARCHAR(40),
INBOND_ENTRY_TYPE VARCHAR(30),
CARRIER_CODE VARCHAR(10),
CARRIER_NAME VARCHAR(40),
CARRIER_CITY VARCHAR(40),
CARRIER_STATE VARCHAR(10),
CARRIER_ZIP VARCHAR(10),
CARRIER_ADDRESS VARCHAR(200),
NOTIFY_PARTY VARCHAR(50),
NOTIFY_ADDRESS VARCHAR(200),
PLACE_OF_RECEIPT VARCHAR(50),
DATE_OF_RECEIPT TIMESTAMP,
constraint shipment_staging_table_pk primary key (position, txindi, opcode, transtimestamp)
);
```

COLUMNS from TARGET Table

TARGET Table

```
CREATE TABLE SHIPMENT_IN_TRANSIT(
SHIPMENTID VARCHAR(11) NOT NULL PRIMARY KEY,
ALERT VARCHAR(80),
SHIPMODE VARCHAR(30),
PRODUCT_DESCRIPTION VARCHAR(500),
CONSIGNEE VARCHAR(200),
SHIPPER VARCHAR(100),
ARRIVAL_DATE TIMESTAMP,
GROSS_WEIGHT_LB INTEGER,
GROSS_WEIGHT_KG INTEGER,
FOREIGN_PORT VARCHAR(50),
US_PORT VARCHAR(50),
VESSEL_NAME VARCHAR(40),
COUNTRY_OF_ORIGIN VARCHAR(40),
CONSIGNEE_ADDRESS VARCHAR(150),
SHIPPER_ADDRESS VARCHAR(150),
ZIPCODE VARCHAR(20),
NO_OF_CONTAINERS INTEGER,
CONTAINER_NUMBER VARCHAR(200),
CONTAINER_TYPE VARCHAR(80),
QUANTITY INTEGER,
QUANTITY_UNIT VARCHAR(10),
MEASUREMENT INTEGER,
MEASUREMENT_UNIT VARCHAR(5),
BILL_OF_LADING VARCHAR(20),
HOUSE_VS_MASTER CHAR(1),
DISTRIBUTION_PORT VARCHAR(40),
MASTER_BL VARCHAR(20),
VOYAGE_NUMBER VARCHAR(10),
SEAL VARCHAR(300),
SHIP_REGISTERED_IN VARCHAR(40),
INBOND_ENTRY_TYPE VARCHAR(30),
CARRIER_CODE VARCHAR(10),
CARRIER_NAME VARCHAR(40),
CARRIER_CITY VARCHAR(40),
CARRIER_STATE VARCHAR(10),
CARRIER_ZIP VARCHAR(10),
CARRIER_ADDRESS VARCHAR(200),
NOTIFY_PARTY VARCHAR(50),
NOTIFY_ADDRESS VARCHAR(200),
PLACE_OF_RECEIPT VARCHAR(50),
DATE_OF_RECEIPT TIMESTAMP
);
```

splice MACHINE

Oracle CDC for Splice – Java Class

- The main Java class behind the Splice's CDC workflow is ApplyCDC
- ApplyCDC class takes the following input parameters:
  - inCDCSchema - Schema where the staging table is hosted
  - inCDCTable - Staging table name
  - inTargetSchema - Schema of Target table
  - inTargetTable - Target table
  - inFromTimeStamp - Process all raw changed records that occurred at time that is >= TimeStamp
  - inJDBCURL - URL for JDBC connection

```
public class ApplyCDC {

    public static void UpdateTargetTable (final String inCDCSchema, final String inCDCTable, final String inTargetSchema,
                                          final String inTargetTable, final String inFromTimeStamp, final String inJDBCURL) {
```

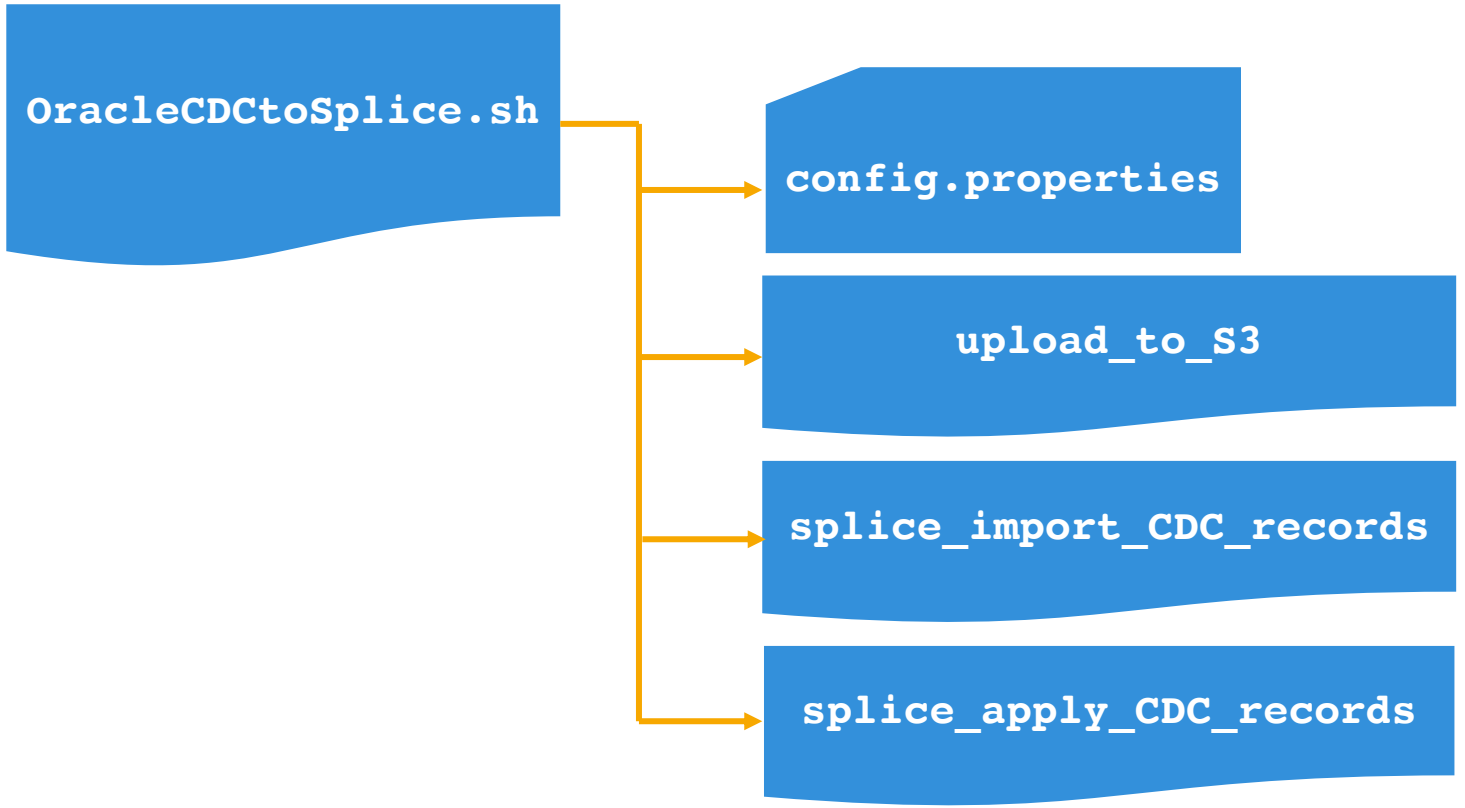
Oracle CDC for Splice – UDF/Stored Procedure Definition

- A wrapper stored procedure can be defined to utilize the ApplyCDC class

```
CREATE PROCEDURE ORACLE_CDC(inCDCSchema VARCHAR(20), inCDCTable VARCHAR(40),
inTargetSchema VARCHAR(20), inTargetTable VARCHAR(40), inFromTimeStamp
VARCHAR(26), jdbcURL VARCHAR(100))
    LANGUAGE JAVA
    PARAMETER STYLE JAVA
    MODIFIES SQL DATA
    EXTERNAL NAME 'com.splice.custom.trigger.ApplyCDC.UpdateTargetTable';
```

Oracle CDC for Splice – Supporting Scripts

- In addition to the main Java code/class that performs the changed data capture logic, there are other scripts that, together, provide the complete CDC workflow.
  - OracleCDCtoSplice.sh
  - upload_to_S3
  - splice_import_CDC_records
  - splice_apply_CDC_records
  - config.properties



Oracle CDC for Splice – Supporting Scripts

- OracleCDCtoSplice.sh
  - This is the main script which sets the configuration properties provided by `config.properties` and calls `splice_import_CDC_records` followed by `splice_apply_CDC_records`.
- upload_to_S3
  - Uploads raw CSV files generated from Oracle Golden Gate to S3 bucket
- splice_import_CDC_records
  - Imports raw CDC records from CSV files generated by Oracle Golden Gate Flat File Adapter
- splice_apply_CDC_records
  - Execute the changed data capture logic by calling the stored procedure defined base on the ApplyCDC class
- config.properties
  - Contains configuration properties

Oracle CDC for Splice – High-Level of how to set it all up

The following steps assume that there is already an existing/functioning Oracle Golden Gate environment where supplemental database logging is enabled and proper golden gate user and role are defined.

- Install and configure Oracle Golden Gate Flat File Adapter.
- Create corresponding STAGING table for TARGET table in Splice Machine.
- Configure appropriate extract, pump and replication process in Oracle Golden Gate.
- Install ApplyCDC jar in Splice Machine database environment and create the wrapper stored procedure based on the jar.
- Deploy the **`scripts`** directory (in github project) to the server that hosts the raw CDC files generated by Oracle Golden Gate Flat File Adapter.
- Make appropriate changes to **config.properties** file and supporting scripts to suit the target environment.

Oracle CDC for Splice – github project

Contents included in project: https://github.com/bhoang-splicemachine/OracleCDC

- APL_CDC_Demo.jar
  - Jar compiled from java source code. This jar is installed on Splice Machine database to create wrapper stored procedure.
- src/com/splice/custom/trigger/ApplyCDC.java
  - Contains the source code and logic for processing changed records on Splice.
- Scripts
  - Consists of supporting scripts required to set up the automation of changed data capture process.
  - It also contains **splicelibs**, which is a set of Splice Machine jars needed to provide sql interface for executing SQL scripts.
- deploy_jar
  - SQL scripts for installing jar and creating wrapper stored procedure in Splice Machine database.

Oracle CDC for Splice – github project

Contents included in project: https://github.com/bhoang-splicemachine/OracleCDC

- OGG_extracts
  - Has examples of parameter files for extract, pump and flat file adapter
    - dirprm/extship.prm (extract)
    - dirprm/extdpship.prm (pump)
    - dirprm/ffue.prm (flat-file writer)
    - dirprm/ffue.properties (flat-file writer properties)
  - Sample SQL commands for enabling supplemental logging, golden gate replication, and defining extracts in Oracle database instance.
    - how_to_enable_replication_and_extracts_in_Oracle_database.txt