

Setting up Bidirectional Replication

Practice Overview

In this practice you will set up an Oracle GoldenGate Bidirectional Replication between db1 and db2. The practice plan will be as follows:

- Set up the bidirectional replication without CDR
- Test the bidirectional configuration
- Configure a CDR
- Test the configured CDR

With bidirectional configuration, you have both the Extract and Replicat processes configured in each database. Therefore, none of them is considered as source or target. However, one of them should be considered as the “trusted” source. In our example, db1 will be considered as the trusted database.



Setting Up Bidirectional Replication

A. Delete existing configuration

In the following steps you will delete the current GoldenGate unidirectional configuration.

1. In `db1` system, stop the Extract, Data Pump, and the Replicat
2. In `db1` system, remove the existing Extract components
 - a. Delete the Extract and Data Pump processes

```
DBLogin UserIDAlias oggdb1  
Delete Extract esrv1  
Delete Extract psrv1
```

- b. Delete the trail files

```
sh rm ./dirdat/es*
```

3. In `ggsrv2` system, remove the existing Replicat components
 - a. Delete the Replicat process

```
DBLogin UserIDAlias oggdb2  
Delete Replicat rsrv2
```

- b. Delete the trail files

```
sh rm ./dirdat/rt*
```

B. Preparing db1 and db2 databases

All the steps that you did in db1 to prepare it for data capture should now be done in db2. You will also perform some preparation steps for the bidirectional configuration in db1.

4. In db2, drop the HRTRG.EMP_HISTORY table. You created that table in an earlier practice for saving the transactions history and you do not need it for this practice.

```
sqlplus / as sysdba
DROP TABLE HRTRG.EMP_HISTORY;
```

5. In db2 database, verify that all the tables included in your replication have primary key constraints.

```
-- all tables in HRTRG schema
select t.OWNER, t.TABLE_NAME
from DBA_TABLES t
where not exists (select 1
                  from DBA_CONSTRAINTS c
                  where c.OWNER = t.OWNER and c.TABLE_NAME = t.TABLE_NAME
                    and c.CONSTRAINT_TYPE = 'P')
and t.OWNER='HR';
```

6. In db2 system, enable schema-level supplemental logging for HRTRG schema. ALLCOLS is used to log non-key columns. They are needed for conflict resolution that you will configure later.

If you have specific tables to replicate rather than all the tables in a schema, use the Add TranData command to enable the table-level supplemental logging for the tables to replicate.

```
DBLogin UserIDAlias oggdb2
ADD SCHEMATRANDATA hrtrg ALLCOLS
```

7. In db1 system, perform the same command for HR schema.

You executed this command in an earlier practice. You re-execute it here because you want to include the ALLCOLS option.

```
DBLogin UserIDAlias oggdb1
ADD SCHEMATRANDATA hr ALLCOLS
```

8. In db1 database, disable the job that keeps updating the SAMPLE table.

```
conn hr/oracle
exec DBMS_SCHEDULER.DISABLE('UPDATE_SAMPLE_JOB');
```

Note: In general cases, you also need to re-write the triggers in both database so that they do not fire by the operations executed by the Replicat. Fortunately, integrated Replicat automatically handles the triggers. As you are going to configure integrated Replicats in this practice, you do not need to review the triggers.

The same fact applies on the ON DELETE CASCADE constraints. The integrated Replicat automatically handles them. Otherwise, you need to convert them into BEFORE DELETE triggers.

9. Connect to db2 database as sysdba and set the STREAMS_POOL_SIZE parameter.

As you learnt in the “Implementing Integrated Processes” lecture, this memory size is used by the Integrated Extract. The size set here is for education purpose. In production system, you need to monitor its size and make sure it is enough for your work load.

```
alter system set STREAMS_POOL_SIZE=250m scope=spfile;  
alter system set SGA_MAX_SIZE=1536M scope=spfile;
```

10. In db2 database, drop the extra columns that you added to the HRTRG.EMPLOYEES table in a past practice.

If you want to keep those columns, you have to modify the COMPARECOLS setting that you will configure later.

```
ALTER TABLE HRTRG.EMPLOYEES DROP (FULL_NAME, IT_JOB_FLAG, WORKING_DAYS, TITLE);
```

11. Restart the db2 database.



C. Configure the Extract and Replicat in db1

In the following steps, you will create the Extract, Data Pump, and Replicat processes in the `ggsrv1` system, together with all their required components.

Because both databases are Oracle databases of version 12.1, it is recommended to configure the integrated Extract and Replicat processes.

In `ggsrv1` system, perform the following:

12. Edit the Extract parameter file and add the following code to it.

Notice that you use the `EXCLUDETAG` to exclude capturing the transactions made by the Replicat.

```
Extract esrv1
Include /u01/app/oracle/product/ogg/dirprm/header.mac
UserIDAlias oggdb1
ExtTrail ./dirdat/es
LOGALLSUPCOLS
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 250, PARALLELISM 3)
UPDATERECORDFORMAT COMPACT
TRANLOGOPTIONS EXCLUDETAG 0999
Table HR.JOB_HISTORY;
Table HR.EMPLOYEES, TOKENS ( TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),
TK-HOST = @GETENV('GGENVIRONMENT' , 'HOSTNAME'));
Table HR.JOBS;
Table HR.DEPARTMENTS;
Table HR.LOCATIONS;
Table HR.REGIONS;
Table HR.SAMPLE;
Table HR.RTABLE;
```

13. Verify that the Data Pump (`psrv1`) parameter file has the following code in it.

Regarding the `EncryptTrail` parameter, for this practice, it is optional.

```
Extract psrv1
RmtHost ggsrv2, MgrPort 7810
RmtTrail ./dirdat/rt
Passthru
Table HR.*;
```

14. Create a parameter file for the Replicat process (`rsrv1`). Add the following code to it.

Notice that the `SETTAG` option is used as a loop prevention mechanism.

```
edit params rsrv1
```

```
Replicat rsrv1
Include /u01/app/oracle/product/ogg/dirprm/header.mac
DiscardFile ./dirrpt/rsrv1.dsc, Purge
UserIDAlias oggdb1
DBOPTIONS SETTAG 0999
Map HRTRG.*, Target HR.*;
```

15. Add all the processes.

When you configure the integrated Replicat process, it is usually automatically registered by GoldenGate. However, sometimes this auto-registration does not work and that is why I recommend manually registering it as shown in the code below.

```
DBLogin UserIDAlias oggdb1
ADD EXTRACT esrv1 INTEGRATED TRANLOG, BEGIN NOW
ADD EXTTRAIL ./dirdat/es, EXTRACT esrv1
ADD EXTRACT psrv1, EXTTRAILSOURCE ./dirdat/es
ADD RMTTRAIL ./dirdat/rt, EXTRACT psrv1
REGISTER REPLICAT rsrv1 DATABASE
ADD REPLICAT rsrv1 INTEGRATED, EXTTRAIL ./dirdat/rt
```



D. Configure the Extract and Replicat in db2

In the following steps, you will create the Extract, Data Pump, and Replicat processes in the `ggsrv2` system, together with all their required components. CDR will be configured in a later section in this practice.

In `ggsrv1` system, perform the following:

16. Create the Extract (`esrv2`) parameter file and add the following code to it.

```
edit params esrv2
```

```
Extract esrv2
Include /u01/app/oracle/product/ogg/dirprm/header.mac
UserIDAlias oggdb2
ExtTrail ./dirdat/es
LOGALLSUPCOLS
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 250, PARALLELISM 3)
UPDATERECORDFORMAT COMPACT
TRANLOGOPTIONS EXCLUDETAG 0999
Table HRTRG.JOB_HISTORY;
Table HRTRG.EMPLOYEES, TOKENS ( TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),
TK-HOST = @GETENV('GGENVIRONMENT' , 'HOSTNAME'));
Table HRTRG.JOBS;
Table HRTRG.DEPARTMENTS;
Table HRTRG.LOCATIONS;
Table HRTRG.REGIONS;
Table HRTRG.SAMPLE;
Table HRTRG.RTABLE;
```

17. Create the Data Pump (`psrv2`) parameter file and enter the following code into it.

```
edit params psrv2
```

```
Extract psrv2
RmtHost ggsrv1, MgrPort 7809
RmtTrail ./dirdat/rt
Passthru
Table HRTRG.*;
```

18. Edit the parameter file of the Replicat process (`rsrv2`). Add the following code to it.

You might want to take a backup copy of the existing parameter file (`sh mv ./dirprm/rsrv2.prm ./dirprm/rsrv2.bak1`).

```
edit params rsrv2
```

```
Replicat rsrv2
Include /u01/app/oracle/product/ogg/dirprm/header.mac
DiscardFile ./dirrpt/rsrv2.dsc, Purge
UserIDAlias oggdb2
DBOPTIONS SETTAG 0999
Map HR.*, Target HRTRG.*;
```

19. Add all the processes.

```
DBLogin UserIDAlias oggdb2
REGISTER EXTRACT esrv2 DATABASE
ADD EXTRACT esrv2 INTEGRATED TRANLOG, BEGIN NOW
ADD EXTTRAIL ./dirdat/es, EXTRACT esrv2
ADD EXTRACT psrv2, EXTTRAILSOURCE ./dirdat/es
ADD RMTTRAIL ./dirdat/rt, EXTRACT psrv2
ADD REPLICAT rsrv2 INTEGRATED, EXTTRAIL ./dirdat/rt
```

E. Start the processes and verify their status

In the following steps, you will start the Oracle GoldenGate processes and verify their status.

20. In `ggsrv1`, start the processes and verify their status

```
start er *  
info er *
```

21. In `ggsrv2`, start the processes and verify their status



F. Test the bidirectional configuration basic functionality

In the following steps, you will test the bidirectional configuration. You will perform a DML statement in each database and verify that it has been replicated to the other database. Conflicts will not be tested because you did not configure the CDR yet.

22. Verify that the SALARY column is the same for EMPLOYEE ID 120 in both db1 and db2.

```
conn hr/oracle
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=120;

conn hrtrg/oracle
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=120;
```

23. In db1, update one record in a replicated table.

```
UPDATE EMPLOYEES SET SALARY=12000 WHERE EMPLOYEE_ID=120;
COMMIT;
```

24. In db2, verify that the update has been replicated. Wait for a few seconds before you try it.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=120;
```

25. In db2, query the lag in the heartbeat table.

```
conn / as sysdba
col LOCAL_DB format a7
col RMT_DB format a6
col INCOMING_PATH format a30

select REMOTE_DATABASE RMT_DB, LOCAL_DATABASE LOCAL_DB, INCOMING_PATH, INCOMING_LAG
from OGG.GG_LAG;
```

26. In db2, update one record in a replicated table.

```
conn hrtrg/oracle
UPDATE EMPLOYEES SET SALARY=14000 WHERE EMPLOYEE_ID=120;
COMMIT;
```

27. In db1, verify that the update has been replicated. Wait for a few seconds before you try it.

```
conn hr/oracle
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=120;
```

28. In db1, query the lag in the heartbeat table.

```
conn / as sysdba
col LOCAL_DB format a7
col RMT_DB format a6
col INCOMING_PATH format a30

select REMOTE_DATABASE RMT_DB, LOCAL_DATABASE LOCAL_DB, INCOMING_PATH, INCOMING_LAG
from OGG.GG_LAG;
```

29. View statistics on the processes in both systems.

```
-- in ggsrv1
stats esrv1, total
stats rsrv1, total

-- in ggsrv2
stats esrv2, total
stats rsrv2, total
```



G. Prepare the EMPLOYEES table for CDR configuration

In this section of the practice, you will prepare the EMPLOYEES table for CDR configuration.

- 30.** In db1, add a column to the EMPLOYEES table that will save the time of last DML operation executed on a record. Add a trigger that will automatically update the column.

```
conn hr/oracle
ALTER TABLE HR.EMPLOYEES ADD ( DML_TIME DATE DEFAULT SYSDATE );

CREATE OR REPLACE TRIGGER HR.UPDATE_DMLTIME
  BEFORE UPDATE ON HR.EMPLOYEES
  REFERENCING NEW AS NEW
  FOR EACH ROW
  DECLARE
  BEGIN
    :NEW.DML_TIME := SYSDATE ;
  END UPDATE_DMLTIME;
/
```

- 31.** In db2, add the same column and trigger.

```
conn hrtrg/oracle
ALTER TABLE HRTRG.EMPLOYEES ADD ( DML_TIME DATE DEFAULT SYSDATE );

CREATE OR REPLACE TRIGGER HRTRG.UPDATE_DMLTIME
  BEFORE UPDATE ON HRTRG.EMPLOYEES
  REFERENCING NEW AS NEW
  FOR EACH ROW
  DECLARE
  BEGIN
    :NEW.DML_TIME := SYSDATE ;
  END UPDATE_DMLTIME;
/
```

- 32.** In db1, insert a new record in the EMPLOYEES table and verify the value of the new column.

```
INSERT INTO HR.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,
SALARY, JOB_ID) VALUES (223, 'Ahmed', 'Baraka', 'info@ahmedbaraka.com', TO_DATE('01-
01-2000', 'DD-MM-YYYY'), 10000, 'IT_PROG');
COMMIT;
SELECT TO_CHAR(DML_TIME, 'HH24:MI:SS') DML_TIME FROM EMPLOYEES WHERE EMPLOYEE_ID=223;
```

- 33.** In db2, verify the record has been successfully replicated.

```
SELECT LAST_NAME, TO_CHAR(DML_TIME, 'HH24:MI:SS') DML_TIME FROM HRTRG.EMPLOYEES WHERE
EMPLOYEE_ID=223;
```

- 34.** Delete the new record.

```
DELETE EMPLOYEES WHERE EMPLOYEE_ID=223;
COMMIT;
```

Note: as explained in the concepts lecture, when you configure a bidirectional replication in real life scenarios, you should study the mechanism on which the primary key values are generated.



H. Setup a CDR configuration – Case 1

In this section of the practice, you will set up a CDR configuration that contains a group of rules. You will then perform a group of test cases against those rules. The target is to have an understanding on how each CDR rule reacts to the conflicts.

The testing methodology in all test cases will be as follows:

- 1) Stop the Data Pump processes in all the databases to simulate a network lag.
- 2) Execute two conflicting DML operations in db1 and db2.
- 3) Start the Data Pump processes
- 4) Examine the conflict resolution executed by Oracle GoldenGate

35. In `ggsrv1`, edit the Replicat (`rsrv1`) parameter file and add the following CDR rules on the MAP setting of the EMPLOYEES table. Make sure it is put just before the existing MAP statement.

```
Map HRTRG.EMPLOYEES, Target HR.EMPLOYEES,
  CompareCols (ON UPDATE ALL, ON DELETE ALL),
  ResolveConflict (INSERTROWEXISTS, (DEFAULT, USEMAX (DML_TIME))),
  ResolveConflict (UPDATEROWEXISTS, (DEFAULT, USEMAX(DML_TIME))),
  ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
  ResolveConflict (DELETEROWEXISTS, (DEFAULT, OVERWRITE)),
  ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD))
);
```

36. In `ggsrv2`, edit the Replicat (`rsrv2`) parameter file and add the following CDR rules on the MAP setting of the EMPLOYEES table:

```
Map HR.EMPLOYEES, Target HRTRG.EMPLOYEES,
  CompareCols (ON UPDATE ALL, ON DELETE ALL)
  ResolveConflict (INSERTROWEXISTS, (DEFAULT, USEMAX (DML_TIME))),
  ResolveConflict (UPDATEROWEXISTS, (DEFAULT, USEMAX(DML_TIME))),
  ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
  ResolveConflict (DELETEROWEXISTS, (DEFAULT, OVERWRITE)),
  ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD))
);
```

37. Restart the Replicat process in each database.

Test Case 1

In this test case, you will test the resolution of the INSERTROWEXISTS conflict. You will insert a new employee of ID 223 in db1 and then add a new employee of the same ID in db2. According to the CDR rule, the record inserted in db2 should win because it was inserted later.

38. Stop the Data Pump processes in both systems.

39. In db1, insert a new record in the EMPLOYEES table. Make the LAST_NAME value as DB1.

```
INSERT INTO HR.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,
SALARY, JOB_ID) VALUES (223, 'DB1', 'DB1', 'info@ahmedbaraka.com', TO_DATE('01-01-
2000', 'DD-MM-YYYY'), 10000, 'IT_PROG');
COMMIT;
```

40. In db2, insert a new record in the EMPLOYEES table. Make the LAST_NAME value as DB2.

```
INSERT INTO HRTRG.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,
SALARY, JOB_ID) VALUES (223, 'DB2', 'DB2', 'info@ahmedbaraka.com', TO_DATE('01-01-
2000', 'DD-MM-YYYY'), 10000, 'IT_PROG');
COMMIT;
```

41. Start the Data Pump processes.

42. Select the LAST_NAME for the same record in both databases. You should see the record inserted in db2 applied in both databases.

```
-- in db1
SELECT LAST_NAME FROM HR.EMPLOYEES where EMPLOYEE_ID= 223;

-- in db2
SELECT LAST_NAME FROM HRTRG.EMPLOYEES where EMPLOYEE_ID= 223;
```

Test Case 2

The resolution of the conflict UPDATEROWEXISTS is similar to the resolution of the INSERTROWEXISTS conflict. I will leave testing it for you. In this test case, you will test the resolution of the UPDATEROWMISSING conflict.

You will delete the record of employee of ID 223 in db2 and then update the record of the same employee in db1. According to the CDR rule (OVERWRITE), the record should be inserted in db2. It should be deleted from db1.

43. Stop the Data Pump processes in both systems.

44. In db2, delete the record that you inserted in the previous test case.

```
DELETE HRTRG.EMPLOYEES WHERE EMPLOYEE_ID=223;  
COMMIT;
```

45. In db1, make an update on the record of the same employee.

```
UPDATE HR.EMPLOYEES SET FIRST_NAME = 'DB1', LAST_NAME = 'DB1'  
WHERE EMPLOYEE_ID=223;  
COMMIT;
```

46. Start the Data Pump processes.

47. Verify that the record has been inserted in db2, as indicated by the named resolution method.

```
SELECT LAST_NAME FROM HRTRG.EMPLOYEES where EMPLOYEE_ID= 223;
```

48. Verify that the record has been deleted from db1.

```
SELECT COUNT(*) FROM HR.EMPLOYEES where EMPLOYEE_ID= 223;
```

49. To prepare for the new test case, insert the record again in db1.

```
INSERT INTO HR.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,  
SALARY, JOB_ID) VALUES (223, 'DB1', 'DB1', 'info@ahmedbaraka.com', TO_DATE('01-01-  
2000', 'DD-MM-YYYY'), 10000, 'IT_PROG');  
COMMIT;
```

Test Case 3

The resolution method defined for the conflict DELETEROWEXISTS is the same as the resolution method defined for the conflict handled in the previous test case. I will leave testing it for you. In this test case, you will test the resolution of the DELETEROWMISSING conflict. The conflicting record should be saved in the discard file.

You will delete the record of employee of ID 223 in db1 and then delete the record of the same employee in db2. According to the CDR rule (DISCARD), the record should be saved in the discard file in db1 and db2.

50. Stop the Data Pump processes in both systems.

51. In db1, delete the record of the employee ID 223.

```
DELETE HR.EMPLOYEES WHERE EMPLOYEE_ID=223;  
COMMIT;
```

52. In db2, delete the record of the same employee.

```
DELETE HRTRG.EMPLOYEES WHERE EMPLOYEE_ID=223;  
COMMIT;
```

53. Start the Data Pump processes.

54. Checkout the discard file.

```
-- in ggsrv1  
sh cat ./dirrpt/rsrv1.dsc  
  
-- in ggsrv2  
sh cat ./dirrpt/rsrv2.dsc
```

I. Setup a CDR configuration – Case 2

In this section of the practice, you will set up a more complicated CDR configuration. You will then perform test cases on each rule defined in the configuration. You will use the same testing methodology that you used in the previous case.

55. In `ggsrv1`, edit the Replicat (`rsrv1`) parameter file and replace the MAP setting of the EMPLOYEES table with the following one.

```
Map HRTRG.EMPLOYEES, Target HR.EMPLOYEES,
CompareCols(On Update All),
ResolveConflict (UpdateRowExists,
                (delta_method, UseDelta, Cols (SALARY)),
                (max_method, UseMax (DML_TIME), Cols(COMMISSION_PCT, DML_TIME)),
                (Default, Overwrite));
```

56. In `ggsrv2`, edit the Replicat (`rsrv2`) parameter file and replace the MAP setting of the EMPLOYEES table with the following one:

```
Map HR.EMPLOYEES, Target HRTRG.EMPLOYEES,
CompareCols(On Update All),
ResolveConflict (UpdateRowExists,
                (delta_method, UseDelta, Cols (SALARY)),
                (max_method, UseMax (DML_TIME), Cols(COMMISSION_PCT, DML_TIME)),
                (Default, Overwrite));
```

57. Restart the Replicat processes

The CRD configuration that you configured defines three resolution methods to resolve the `UpdateRowExists` conflict. This conflict takes place when the operation is an `UPDATE` statement and the before image (in the source) is different from the current image (in the target).

The CRD works as follows:

- It uses all the column values to determine if the conflict exists
- If the conflict happens because of the change in the `SALARY` value, then the `SALARY` value will be modified. It will take the difference between the new updated value and the before image value and add it to the current value.
- If the conflict happens because of the change in the `COMMISSION_PCT` value, then the record with the higher `DML_TIME` value will overwrite.
- For all other conditions, the record will be overwritten by the incoming updated record.

Test Case 4

In this test case, you will update the salary of the employee of ID 150 in db1 and then update the salary of the same employee in db2. According to the CDR rule, the new salary value in db2 will be the difference between the updated value and the before-update value.

58. Stop the Data Pump processes in both systems.

59. In db1 and db2, retrieve the salary of the employee ID 150, make sure it is the same in both databases.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = 150;
```

60. Write down the Before update value.

Before value: _____

61. In db1, update the salary of the same employee. Add 1000 to it.

```
UPDATE HR.EMPLOYEES SET SALARY=SALARY+1000 WHERE EMPLOYEE_ID = 150;  
COMMIT;
```

62. Write down the After update value.

After value: _____

63. In db2, update the salary of the same employee. Add 400 to it.

This update is executed to make a conflict.

```
UPDATE HRTRG.EMPLOYEES SET SALARY=SALARY+500 WHERE EMPLOYEE_ID = 150;  
COMMIT;
```

64. In db2, retrieve the current value of the salary after the update.

65. Write down the current value.

Current value: _____

66. Start only the Data Pump process psrv1.

67. Retrieve the updated SALARY in db2 for the same employee.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = 150;
```

68. Verify the result matches the following formula.

[After value] - [Before value] + [Current value]

69. Start the Pump process psrv2.

70. Retrieve the updated SALARY in db1 for the same employee.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID = 150;
```

Summary

In this practice you basically performed two major tasks:

- Configure the bidirectional replication processes in two database systems
- Configure CDR rules to govern how the conflicts should be handled.

For any possible logical conflict in a bidirectional configuration, Oracle GoldenGate provides a wide range of resolution methods to handle the conflicts. Some of the methods provide automatic resolution and some of them just report about the caught conflicts. Which one to use depends on the business case.

That is not all about bidirectional configuration. You will learn in the next lecture how to deal with the errors that might be raised.

