Oracle Replication

Installation Guide and User Manual

for Microsoft Windows, Unix and Linux

Version 2.1

Dbvisit Software Limited

Document version: 2.1.6

www.dbvisit.com

Content

Copyright Notice

This document is the property of Dbvisit Software Limited and Avisit Solutions Limited. Dbvisit Replicate is a product of Dbvisit Software Limited and Avisit Solutions Limited.

http://www.dbvisit.com

Disclaimer, Terms and Conditions

By installing, using and running this software you agree that Dbvisit Software Limited, Avisit Solutions Limited and their associated companies and partners will not be held responsible for anything related to installing or running Dbvisit Replicate.

By installing, using and running this software you agree with the terms and conditions displayed during the installation process of Dbvisit.

For the complete Dbvisit Replicate license agreement (LA) see:

http://www.dbvisit.com/content/pdfs/Dbvisit_Replicate_License_Agreement_v1.06.pdf

Some of the features described in this document only apply to the latest version of Dbvisit Replicate.

Specifications are subject to change without notice.

Audience

This document is intended for a technical audience. Experience with Oracle databases is necessary to install Dbvisit Replicate.

Contact

Dbvisit Support Service desk: http://www.dbvisit.com/support/service_desk. Please include Dbvisit Replicate trace and log files to ensure a fast turnaround on support issues (see special chapter in this document for an automated way to do this).

Please see the Dbvisit website: www.dbvisit.com for other up to date contact information.

Introduction

What is Dbvisit Replicate?

Dbvisit Replicate is a comprehensive software product for enabling real time Oracle replication. Dbvisit Replicate uses efficient redo log-based change data capture (CDC[1]) technology to detect changes to the source database and to replicate and distribute the changed data in real-time across Oracle and non Oracle databases. This is also referred to as redo-log mining. Dbvisit Replicate uses its own internal redo-log mining technology and is not dependent on any third-party software.

CDC technology based on redo log files have distinct advantages that include:

- minimal impact on the source database (even more so as Dbvisit Replicate can use the optional fetcher process for downstream mining on a dedicated separate host).
- no need for programmatic changes to the applications that use the database.
- low latency in acquiring changes in source database.
- transactional integrity: Dbvisit Replicate replays the original transactions in the order they were committed.
- no changes to the database schema required (no triggers, materialised views etc).

Bi-direction (or 2-way) replication is possible allowing master-to-master replication to provide real time information across multiple applications and sites.

One-way replication is possible allowing real time reporting across distributed reader farms reducing the performance impact on the production databases.

Dbvisit Replicate can enable the following:

1. Migration and Upgrades. In these tough economic times, organizations are trying to lower costs in all of their business operations. One key method for cutting IT costs is migration: migration to hardware that is less expensive to purchase and operate; migration to less costly operating systems; and migration away from older versions of Oracle to reduce costs of extended support contracts. Dbvisit Replicate will facilitate the migration and upgrades with almost no outage.
2. One central database with satellite offices. Dbvisit Replicate will replicate only specific data that is applicable to satellite offices.
3. Reporting on MySQL databases. Dbvisit Replicate will replicate the data out of a central Oracle database into an open source MySQL database for reporting purposes, therefore offloading the main central database.
4. Loading of data warehouses. Dbvisit Replicate can detect changes to source data and load these into the staging areas of a data warehouse for business intelligence reporting or ETL processing.
5. Integration with other systems. Dbvisit Replicate can detect changes to source data and push this data into other systems.

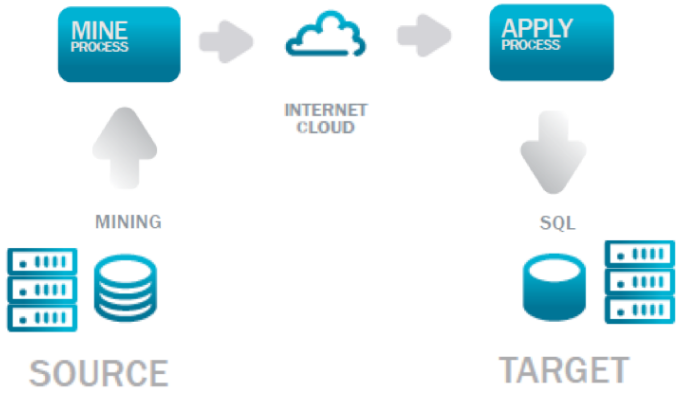
How does Dbvisit Replicate work?

Definition:	Source Database	- Database that contains the data that needs to be replicated.
	Target Database	- Database that receives the replicated data.

Dbvisit Replicate reads (*mines*) Oracle redo logs in real time at the source database, converts them to SQL commands and executes them (*applies*) them at the target database. Dbvisit Replicate is very efficient, has very low system overhead and can handle large transaction volumes without compromising the performance of the source database.

Both databases are opened read-write.

[1]This does not refer to the Oracle product CDC, but rather to the CDC methodology



Only the source database has to be on Oracle; the target database can be Oracle, SQL Server or MySQL. For 2-way replication or master-to-master both both source and target databases have to be Oracle.

What does Dbvisit Replicate replicate?

Dbvisit Replicate can replicate on different levels. It can replicate:

- Whole database (except sys and system objects).
- Whole schemas.
- Individual tables.

Dbvisit Replicate replicates both:

- DML (data changes)
- DDL[2](structure changes)

Dbvisit Replicate notification

Dbvisit Replicate includes full notification features with integrated email and SNMP capability. Thresholds can be set to alert when the replication falls too far behind or when no recoverable errors are detected. Example if notification thresholds are:

- Progress difference in percentage between source and target
- Redo log sequence difference between source and target
- SCN sequence difference between source and target

A daily replicate report is also sent out which lists the replicated objects with their replication progress.

Dbvisit Replicate conflict resolution

Dbvisit Replicate has powerful conflict resolution build in that allows for different resolution depending on the requirements of the business.

When a conflict occurs the whole replication is suspended until the conflict is resolved. This is to ensure transactional integrity.

Example of conflict resolution types are:

- RETRY the change (default)
- PAUSE the whole replication
- ABORT the whole replication
- DISCARD the change

[2] Not all DDL is supported. For example create tablespace is not supported. DDL is only supported for Oracle target database.

- OVERWRITE the change with the new change
- PLSQL – Call a user defined PL/SQL routine that determines how to handle the conflict through programmed business rules
- NEWER change gets precedence
- OLDER change gets precedence
- ERROR, rollback the transaction

Dbvisit Replicate concepts

Mine process

The mine process reads the Oracle online redo logs at the source database in real time. The mine process converts information from the Oracle redo log into a Dbvisit Replicate internal format called a PLOG (parsed log). When the information required is no longer in the redo logs, Dbvisit Replicate will automatically switch to the archive logs and switch back to the redo logs when it has caught up. Redo and archive logs using filesystem and ASM are supported.

There can be more than one Mine process. Each Mine process will have a unique name which is determined by the user. Example: MINE, MINE2, etc.

Apply process

The apply process takes the PLOG (created by the mine process) and converts this information into SQL which can be run against the target database.

There can be more than one Apply process. Each Apply process will have a unique name which is determined by the user. Example: APPLY, APPLY2, etc.

Fetcher process

Fetcher is an optional component. It can be used to offload the mining of the source database to another server. It reads online and archive redo logs at the source database and ships them to the mine process.

There can be more than one Fetcher process. Each Fetcher process will have a unique name which is determined by the user. Example: FETCHER, FETCHER2, etc.

PLOG

Mine reads oracle online redo logs and creates PLOGs (parsed logs). These logs contain parsed information and are filtered to contain information about replicated tables only. PLOGs are binary logs specific to Dbvisit Replicate.

The PLOGs are transferred to the target server where they are used by the apply process.

PLOGs are platform-independent.

Optimistic commit

Dbvisit Replicate uses the same optimistic principle as Oracle applies in transactions. Oracle assumes that most transactions are committed and starts writing the transactions to disk before the commit has been issued.

Dbvisit Replicate applies the same optimistic commit principle during replication which means that Dbvisit Replicate will start mining and applying the changes **before** the actual commit has been issued. For large transaction this can be beneficial as the transactions are not buffered, but are mined and applied as soon as they are written to the Oracle redo logs on the source database. This can reduce the amount of disk and memory required while waiting on commit and it also reduces the gap between mine and apply to a few seconds.

Dbvisit Replicate supports full rollback of the transaction but will require more work to rollback the transaction. This is similar to the work required by Oracle incase of a rollback.

Dbvisit Replicate Architecture

Dbvisit Replicate 2- or 3-tier architecture

The Dbvisit Replicate architecture is a very flexible and powerful architecture and consists of a 2- or 3-tier architecture. The architecture components are as follows:

1. Fetcher process (optional), which runs against the source database and send archive and online redo logs to mine.
2. Mine process, which runs against the source database.
3. Apply process, which runs against the target database.

Dbvisit Replicate 2 tier architecture

The 2-tier architecture is the default architecture and consists of the following 2 processes:

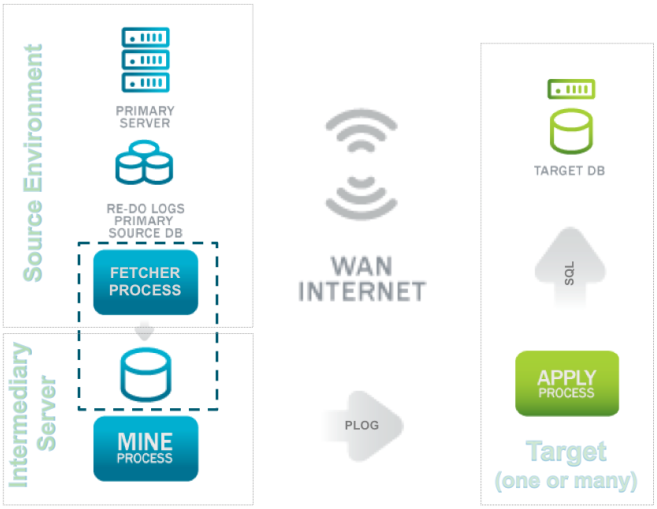
1. Mine process
2. Apply process

Dbvisit Replicate 3 tier architecture

The 3-tier architecture is used to offload the mine process to another server (downstream mine). The processes in the 3-tier architecture are:

1. Fetcher process
2. Mine process
3. Apply process

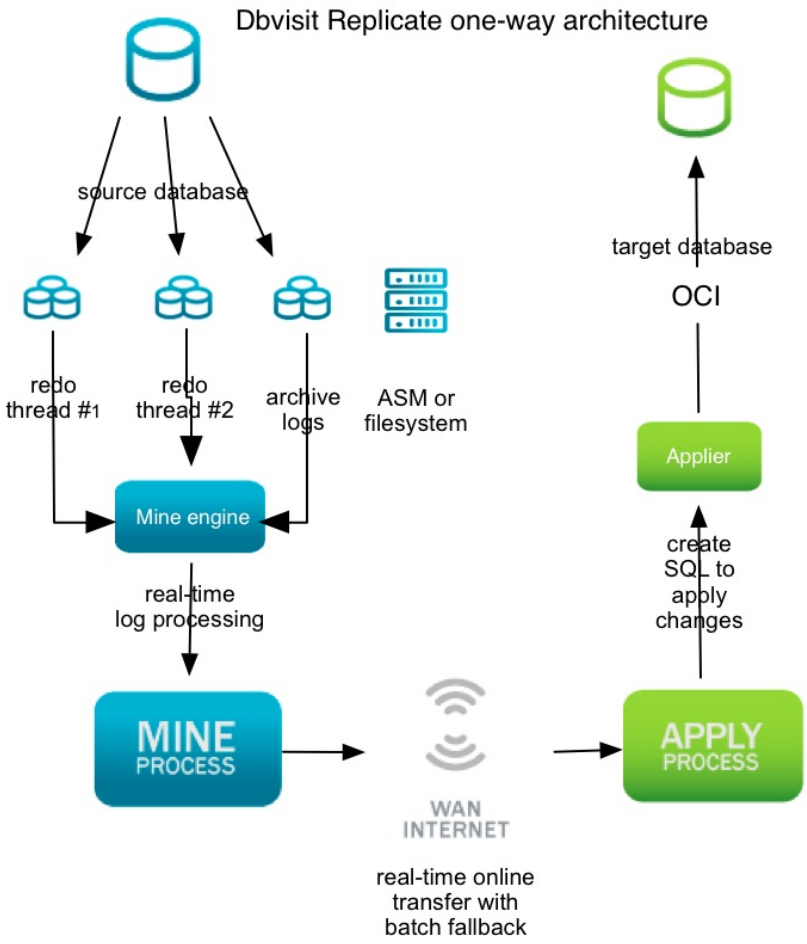
All 3 processes can be run on different operating systems and are platform independent.



The direct impact of the fetcher process on the source database is negligible, as it just stores small amount of state data and simple queries regarding current state of archive and online redo logs are issued against this database.

One-way architecture

Dbvisit Replicate one-way has been designed using a simple architecture to enable speed, performance and efficiency. Dbvisit Replicate does not use intermediate queues or messaging to facilitate the replication.

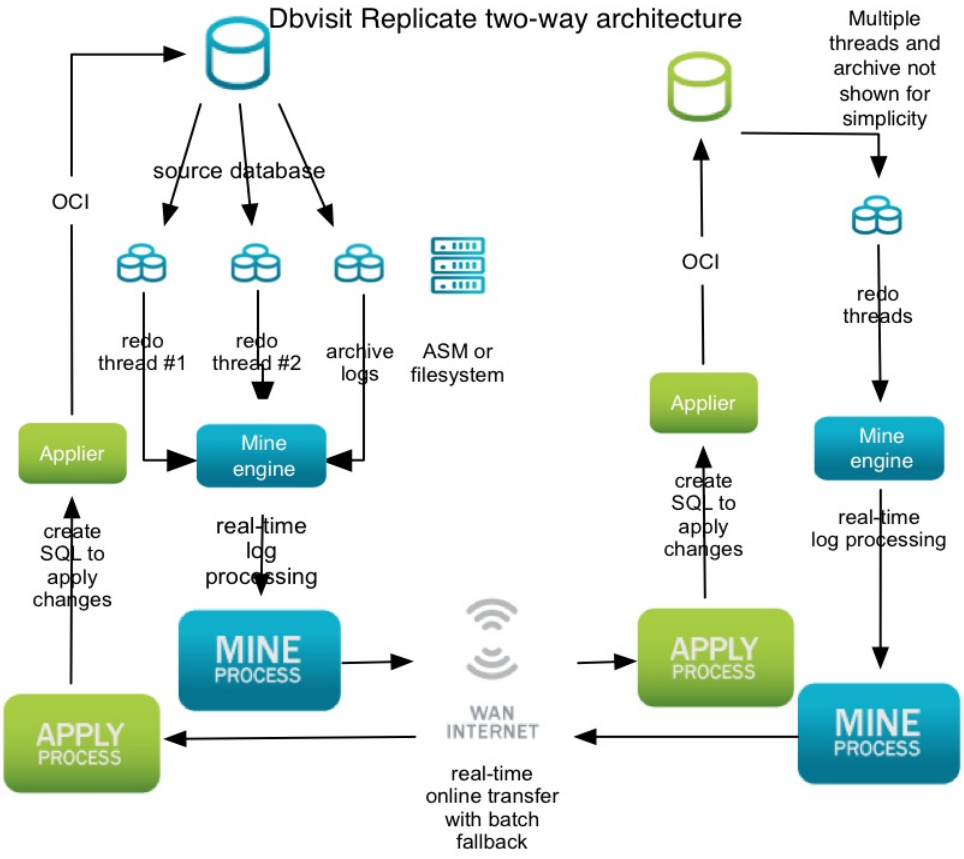


All the above processes can be on the same server, or on different servers. Dbvisit Replicate uses fast Oracle Call Interface (OCI) for direct connections to the database. The Fetcher process is an optional process and is not shown in above diagram.

Two-way architecture

The Dbvisit Replicate two-way architecture is an essence a duplication of the one-way architecture. Each replication from a source to target is a one-way process, but because the replication is going both ways, it is a two x one-way process with the source and target reversed for each process. Dbvisit Replicate ensures that data that is mined on one database and then replicated to the target database, is not mined again on the target database and replicated back as this would cause an infinite feedback loop.

Dbvisit Replicate does not use intermediate queues or messaging to facilitate the replication.



All the above processes can be on the same server, or on different servers. The Fetcher process is an optional process and is not shown in above diagram.

Multiple Mine, Apply and Fetcher processes are possible for the same replication, enabling simple to complex replication topologies.

Platform

Dbvisit Replicate runs on the following platforms:

- Microsoft Windows (2003, 2008, Windows 7)
- Linux (x86, AMD64, EMT64, x86_64)
- Solaris (SPARC, x86, AMD64, EMT64, x86_64)
- AIX

Dbvisit Replicate has a 32-bit and 64-bit version and comes prepackaged Oracle Instant Client, thus no separate Oracle client install is necessary.

Oracle versions from Oracle 9.2, 10g and 11g. Dbvisit Replicate works with Standard Edition (SE), Oracle Enterprise Edition (EE), Workgroup Server, Standard Edition One and Oracle XE.

Exact supported version numbers are determined by the Oracle client support; for the 11.1 prepackaged Instant Client, this is described in MOS note 207303.1. In short, it requires 9.2.0.4 or higher. (Solaris version for Intel/AMD uses 11.2 Instant Client, as no 11.1 version was published by Oracle Corporation.)

Dbvisit Replicate works with RAC. ASM, OMF and flash recovery area.

Dbvisit Replicate supports heterogeneous configurations, i.e. the processes can run on different platforms, enabling you e.g. to do a Linux-to-Windows replication, monitor Linux-to-Linux replication using Windows client etc. This is always enabled and does not require any special configuration.

Dbvisit Replicate system requirements

- 100MB of space on the source and target servers for the Dbvisit Replicate software
- 200MB of space for the Dbvisit Replicate database repository in the Oracle database.
- ~1 – 2GB of memory on the source and target server (this can be reduced by setting MEMORY_LIMIT_APPLY_MB and MEMORY_LIMIT_MINE_MB).
- sufficient space for text logs (and traces if enabled).
- About 10GB of space for redo logs, parsed logs and Dbvisit Replicate log and trace files (1GB for test/dev systems)

Dbvisit Replicate functionality

Dbvisit Replicate supports the following functionality.

Functionality included:

- Full 1-way replication (master to slave).
- Full 2-way replication (master to master). Oracle to Oracle only.
- Full 3-tier architecture which includes the offloading of the mine process to another server (using the Fetcher process).
- Replication of whole databases, schemas, tables or columns.
- Multiple Mine, Apply and Fetcher processes in the same replication.
- Full logging and tracing.
- Automatic setup of network transportation parameters depending on WAN or LAN network.
- Network encryption and security.
- Full ASM support.
- bequeath connection to ASM (without listener)
- Oracle RAC support.
- Easy to use Setup wizard.
- MySQL target databases.
- MS SQL Server target databases.
- Windows service. On Windows Dbvisit Replicate runs as a service.
- Managing of the PLOG and log files. Deletes obsolete plogs and log files. (With fetcher, deletes obsolete redo logs delivered to mine as well.)
- Notifications and full alerting when thresholds are exceeded or errors occur.
- SNMP integration to allow for integration with existing monitoring infrastructure.
- Full support provided. Packing of log files to easily send to support services.

Selected functionality items currently NOT supported:

- LOBs (coming end of Dec 2011).
- XMLType.
- Table cluster related DDL.
- IOTs (index-organized tables).
- Switchover/failover/flashback.
- Non-standard redo log block size (including 4k on 11gR2 on Advanced Format disks).
- NCHAR, NVARCHAR, NVARCHAR2 (character data may be corrupted due to incorrect character set handling).
- Object datatypes (includes ADT, VARRAY, nested tables etc.).
- Only English version is available. Different languages will be supported in the future.

Known limitations:

1. With RAC all nodes must be available, and all the threads must be opened by the instances (this will no longer be the case with Dbvisit Replicate 2.2)
2. No support for compression or encrypted data (Transparent Data Encryption TDE).
3. Dbvisit Replicate relies on the target database client settings for handling of NLS issues. For an Oracle target database, set the NLS_LANG environment variable to "AMERICAN_AMERICA.source_db_charset". The AMERICAN_AMERICA part ensures unified number and date formats and specifying the source database charset ensures that the client libraries handle any non-ASCII characters properly.

Glossary of terms

Certain terms are used during the installation and replication process. This section lists the common terms used and their description.

Term	Description
Source database	The database that contains the "source" data. This is the database that Dbvisit Replicate will be replicating FROM. This is also called the "Mine" database. The source database must always be Oracle.
Target database	The database that will be the "target" for the replicating. This is the database that Dbvisit Replicate will be replicating TO. This is also called the "Apply" database.
Mine process	The mine process reads the Oracle redo logs at the source database. The mine process converts information from the Oracle redo log into a Dbvisit Replicate internal format called a PLOG. Example: To start the mine process, the normal Dbvisit Replicate command is: `dbvrep --daemon --ddcfile orcl.ddc start mine` Where orcl.ddc is the DDC file (or configuration file) for a replication called orcl. In more complicated topologies, there can be more than one Mine process. Each Mine process will have a unique name which is determined by the user. Example: MINE, MINE2, etc.
Apply process	The apply process takes the PLOG (created by the mine process) and converts this information into SQL which can be run against the target database. Example: To start the apply process, the normal Dbvisit Replicate command is: `dbvrep --daemon --ddcfile orcl.ddc start apply` Where orcl.ddc is the DDC file (or configuration file) for a replication called orcl. There can be more than one Apply process. Each Apply process will have a unique name which is determined by the user. Example: APPLY, APPLY2, etc
Dbvisit Replicate Command Console	All Dbvisit Replicate commands are given through the Dbvisit Replicate Command Console (DRCC). To start the DRCC type:

	dbvrep To list progress of the replication: dbvrep> list progress To start the Dbvisit Replicate configure wizard: dbvrep> setup wizard To read and set the environment for a specific replication: dbvrep> readddc w112g.ddc (where w112g.ddc is the DDC file name) or: dbvrep --ddcfile orcl.ddc
Fetcher process	Fetcher is an optional component. It can be used to offload the mining of the source database to another server. Example: To start the fetcher process, the normal Dbvisit Replicate command is: dbvrep --daemon --ddcfile orcl.ddc start fetcher Where orcl.ddc is the DDC file (or configuration file) for a replication called orcl. There can be more than one Fetcher process. Each Fetcher process will have a unique name which is determined by the user. Example: FETCHER, FETCHER2, etc.
PLOG	Mine reads oracle redo logs and creates plogs (parsed logs). These logs contain parsed information and are filtered to contain information about replicated tables only. Plogs are binary logs specific to Dbvisit Replicate. The plogs are transferred to the target server where they are used by the apply process.
Source Server	The host or server that runs the source database. Dbvisit Replicate will be installed on this server.
Target Server	The host or server that runs the target database. Dbvisit Replicate will be installed on this server.
Install directory	The directory where Dbvisit Replicate will be installed.
ORACLE_HOME	The directory where the Oracle software or executables are installed.
TNS_ADMIN	The directory where the Oracle network configuration files are located (sqlnet.ora, tnsnames.ora)
Oracle software owner	This is the Windows user or account that owns the Oracle Software. In most cases this is *oracle.*
Dbvisit Database Configuration file (DDC)	A Dbvisit Replicated created text file which contains all the settings for the replication. The DDC file contains all the necessary configuration information for the replication. The file(s) will be in the Dbvisit Replicate install directory and has the format: *replication_name*.ddc Where *replication_name* is the name of the replication (which is in most cases the name of the source database).

	Contents of this file is by default stored in source database (called “DDC DB”) and the DDC file itself contains only credentials to this database. The DDC file must be manually copied to the target server.
Table key	Every row in a replicated table must be uniquely identifiable – in case of duplicates, apply would not be able to pick up the correct row to update/delete. It is desirable that every replicated table has a primary key defined. If there is no primary key, then each row in the table must be uniquely identifiable.
Dbvisit Replicate schema	A dedicated schema in the replicated database (both source and target). Contains various pieces of information required for operation of Dbvisit Replicate. Dbvisit Replicate logs into the database as this schema owner and performs all operations in the source and target database under this credentials. It is strongly recommended to use a dedicated user for this, to make it possible to drop and recreate the whole configuration if needed.
DML	Data Manipulation Language. These are SQL statements that update or insert data into existing tables or objects. Example: INSERT INTO SCOTT.TEST VALUES (1, “TEST”);
DDL	Data Definition Language. These are SQL statements that change the structure of the database. For example to create a new table or create a new user in the database.

Dbvisit Replicate components

The Dbvisit Replicate environment consists of the following components:

1. Dbvisit Replicate software. The Dbvisit Replicate software consists of the following executable:

   `dbvrep` - The main Dbvisit Replicate executable.

2. Dbvisit Replicate Data Dictionary. This is created at time of setup and is used by Dbvisit Replicate to control and keep track of the replication process. By default the schema that owns the Dbvisit Replicate Data Dictionary is dbvrep.

3. Dbvisit Database Configuration (DDC) file. This is similar to the init.ora parameter file of Oracle and contains the Dbvisit Replicate settings for a specific replication. The DDC file is generated by Dbvisit replicate through the setup wizard during setup for each replication. The DDC file can be edited with any text editor or through running dbvisit_setup

   Most of the settings are actually stored in the source database (and called DDC DB), the DDC file contains only credentials to the mine database enabling any process to load the DDC DB settings.

4. Dbvisit Replication configurations. For each replication, the Dbvisit Replicate setup wizard produces the necessary scripts to setup and initiate the replication. These configurations can be used for scripted deployment into production environments.

This environment will be the same on both the source (mine) and target server (apply) that is running Dbvisit Replicate (except for the Dbvisit Replication configurations. This is only available on the source server).

Installation Prerequisites

Before installing Dbvisit Replicate please ensure that the following prerequisites are met:

	Task	Comment	Completed?
1.	Oracle software is installed on source and target server.	The Oracle software version can be different between the source and target servers.	

2.	Oracle ***source*** database is up and running.	We require database to be in ARCHIVELOG mode. To determine if database is in archivelog mode run SQL command:
		SQL> archive log list
		Example output:
		`Database log mode Archive Mode`
		`Automatic archival Enabled`
		`Archive destination /oracle/oraarch/orcl`
		`Oldest online log sequence 3959`
		`..`	

3.	Oracle ***target*** database is up and running.	We recommend database to be in ARCHIVELOG mode. To determine if database is in archivelog mode run SQL command:
		SQL> archive log list
		Example output:
		`Database log mode Archive Mode`
		`Automatic archival Enabled`
		`Archive destination /oracle/oraarch/orcl`
		`Oldest online log sequence 3959`
		`..`	

4.	Ensure Oracle listener is running on ***source*** server		
5.	Ensure Oracle listener is running on ***target*** server		
6.	Ensure Oracle SQL*Net connection to ***target*** database from ***source*** database.	From the source database, a SQL*Net connection is required to the target database.	
7.	Ensure Oracle SQL*Net connection to ***source*** database from ***target*** database.	From the target database, a SQL*Net connection is required to the source database.	
8.	Ensure that the firewall port is opened between the primary and the standby servers for the Dbvisit Replicate ports.	Dbvisit Replicate uses the following default ports for communication between the source and target servers: – 7890 – 7891 For 2-way replication the default ports are: – 7890	

		– 7891 – 7892 – 7893 These ports are configurable and can be changed if required.	
9.	Ensure ***source*** server name can be resolved to correct IP address. (not 127.0.0.1)	On both source and target, run `ping source-server-name`	
10.	Ensure ***target*** server name can be resolved to correct IP address (not 127.0.0.1).	On both source and target, run `ping target-server-name`	

Recommended approach with Dbvisit Replicate

Although Dbvisit Replicate tries to be as simple to use as possible, replication is nevertheless a complicated process. It is thus recommended to allocate enough time and testing resources to familiarize with Dbvisit Replicate before using it for mission critical tasks.

We therefore recommend a simple step-by-step approach to Dbvisit Replicate.

An example step-by-step approach may be:

1. Start small and simple: set up replication of just a handful of tables, Oracle-to-Oracle, on simple test environment, planning no conflicts. Use SCOTT or other familiar schema.
2. Learn how to set up this simple replication using the setup wizard and how to rerun the setup scripts if needed. The setup scripts are created by the setup wizard. Rerunning the scripts will drop and recreate the replication configuration.
3. Learn how to use the command console, monitor replication progress. Monitor log files on disk, check disk space utilization.
4. Learn how to setup automatic monitoring – email notifications, SNMP (if desired), direct SQL queries (if desired). Integrate with your existing monitoring tools. Setup your monitoring tools to check that processes are running or to check errors in log files.
5. Make your setup more complicated, heading towards your desired configuration – add more tables, setup and test conflict handling, change target database (MySQL/MS SQL), test DDL replication, 2-way replication, RAC.
6. Setup your desired configuration on a test environment and test it thoroughly.
7. Setup your desired configuration on your production environment.

Plan enough time to analyze impacts of replication on your application. These impacts apply to any replication technology and are not specific to Dbvisit Replicate;

1. Replication is not disaster recovery technology and cannot replace proper backups.
2. Data is replicated in transaction-consistent manner, but lags behind the source (although this is minimal with Dbvisit Replicate).
3. If the data is changed at target database by other means than replication, conflicts can occur. This is more prominent in case of two-way replication.
4. The data is replicated using standard SQL and thus triggers must be taken into consideration. See special chapters how to set what triggers to fire.

Example environments:

1. Using a single virtual Linux machine as a test system is a good recommended approach. Oracle can be the source database and MySQL can be the target database. Please see step-by-step instructions in section Oracle → MySQL replication setup example on how to configure this environment.

Upgrading Dbvisit Replicate

Upgrading Dbvisit Replicate involves the following steps:

1. Stop the replication:

```
dbvrep> shutdown all
```

2. Installing the Dbvisit Replicate software. This is the same as a new install of the software. There is no need to configure Dbvisit Replicate.
3. Restart the replication.
4. Upgrade the Dbvisit Replicate repository (if necessary). Dbvisit Repository will notify if the repository needs to be upgraded. To upgrade the repository:

```
dbvrep> UPGRADE REPOSITORY
```

New Dbvisit Replicate installation (2 step process) - Windows

The complete installation and configuration should take less than 30 minutes. Microsoft Windows or the database does **not** need to be restarted.

The complete installation and configuration for Windows is a 2 step process. The steps are:

1. Install Dbvisit Replicate - Windows
2. Configure Dbvisit Replicate – Windows/Linux/Unix

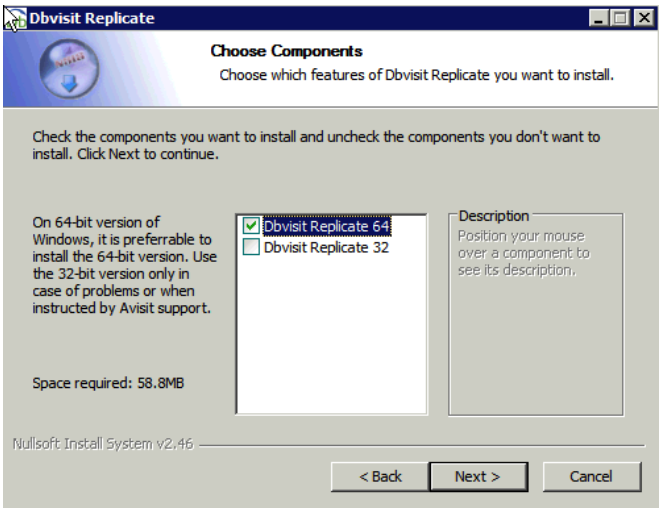
Ensure all Installation Prerequisites are met before continuing with installation.

Dbvisit Replicate software needs to be installed on all servers (source, target and optional fetcher).

Install Dbvisit Replicate - Windows

Dbvisit Replicate can be installed with any Windows account. It does not have to be the same account as the Oracle software.

1. Unzip the dbvisit_replicate2.1.x_win.zip to a temporary location.
2. Double click the Dbvisit Replicate Windows installer: dbvrep2.1.x-Inst.exe



3. Choose the version to install: on 32-bit Windows, only the 32-bit version can be chosen for installation. On 64-bit, both 32-bit and 64-bit are available and it is recommended to use the 64-bit version.
4. Follow the on-screen instructions and choose the location to install Dbvisit Replicate. **Note**: Dbvisit Replicate will be installed under a subdirectory called “replicate” from the main directory chosen.

5. Start Dbvisit Replicate:
6. Start > Dbvisit Replicate Command Console
7. The steps above need to be done on both the source (mine) database server and the target (apply) database server.

New Replicate installation (2 step process) – Linux and Unix

The complete installation and configuration should take less than 30 minutes. The Linux server or the database does **not** need to be restarted.

The complete installation and configuration for Windows is a 2 step process. The steps are:

1. Install Dbvisit Replicate – Linux and Unix
2. Configure Dbvisit Replicate – Windows/Linux/Unix

Ensure all Installation Prerequisites are met before continuing with installation.

Dbvisit Replicate software needs to be installed on all servers (source, target and optional fetcher).

Install Dbvisit Replicate – Linux and Unix

Dbvisit Replicate can be installed as a TAR file or an RPM. The simplest way is to install the RPM, but this does require root privileges. RPM is only available for Linux.

TAR File

Dbvisit Replicate can be installed with any OS user id. It does not have to be the same user id as the Oracle software. However, if ASM is used on mine, a bequeath connection must be made to the ASM instance and thus the user must be member of the dba or sysasm group.

There is a 32bit version and a 64bit version of Dbvisit Replicate. The install instructions are the same for both versions. Only the 64bit installation instructions are shown here.

1. Unzip the chosen `dbvisit_replicate2.*.zip` file to a directory on your local PC if you have not done so already.

On the source (mine) server:

2. Create the Dbvisit main install directory on your server (if this does not yet exist).
    - We recommend : */usr/local/dbvisit*
    - Make sure this directory has the correct owner and permissions.
    - The tar file will create the *replicate* subdirectory under the main *dbvisit* directory. This is where Dbvisit Replicate will be installed.
3. Copy or ftp the dbvrep2.x.tar file to the Dbvisit directory (ie /usr/local/dbvisit).
4. Untar the dbvrep2.x.tar file with the command:

```
tar xvf dbvrep2.1.tar
```

This will have created the *replicate* subdirectory. This is where Dbvisit Replicate will be installed.

5. cd into the *replicate* directory.

```
cd replicate
```

6. Ensure Dbvisit Replicate is executable with the command:

```
chmod 750 dbvrep
```

7. Make sure that the dbvrep executable is in your PATH (modify your PATH settings or create a symbolic link, e.g. from /usr/local/bin).

On the target (apply) server:

8. Create the Dbvisit main install directory on your server (if this does not yet exist).

- We recommend : */usr/local/dbvisit*
- Make sure this directory has the correct owner and permissions.
- The tar file will create the *replicate* subdirectory under the main *dbvisit* directory. This is where Dbvisit Replicate will be installed.

9. Copy or ftp the dbvrep2.x.tar file to the Dbvisit directory (ie /usr/local/dbvisit).
10. Untar the dbvrep2.x.tar file with command:

```
tar xvf dbvrep2.1.tar
```

11. cd into the *replicate* directory.

```
cd replicate
```

12. Ensure Dbvisit Replicate is executable with command:

```
chmod 750 dbvrep
```

13. Make sure that the dbvrep executable is in your PATH (modify your PATH settings or create a symbolic link, e.g. from /usr/local/bin).

RPM File

For Linux, rpm packages are available, too. Following the standards for rpm, the installation directory is set to */usr/dbvisit* and dbvrep symbolic link is created in */usr/bin*.

Root privilege is needed to install the RPM.

There is a 32bit version and a 64bit version of Dbvisit Replicate available. The install instructions are the same for both versions. Only the 64bit installation instructions are shown here.

On the source (mine) server:

1. Copy the `dbvisit_replicate-2.x.x-1.x86_64.rpm` file to a temp directory on your server. (example: /usr/tmp).
2. Switch to the root user:

```
su -
```

3. Install the RPM package with command:

```
rpm -ivh /usr/tmp/dbvisit_replicate-2.0.02-1.x86_64.rpm
```

This installs Dbvisit Replicate in:

```
/usr/dbvisit/replicate/
```

and creates the following soft link:

```
/usr/bin/dbvrep -> /usr/dbvisit/replicate/dbvrep
```

On the target (apply) server:

4. Copy the `dbvisit_replicate-2.x.x-1.x86_64.rpm` file to a temp directory on your server. (example: /usr/tmp).
5. Switch to the root user:

```
su -
```

6. Install the RPM package with command:

```
rpm -ivh /usr/tmp/dbvisit_replicate-2.0.02-1.x86_64.rpm
```

This installs Dbvisit Replicate in:

```
/usr/dbvisit/replicate/
```

and creates the following soft link:

```
/usr/bin/dbvrep -> /usr/dbvisit/replicate/dbvrep
```

Configure Dbvisit Replicate – Windows/Linux/Unix

Dbvisit Replicate comes with its own command language that is used to configure Dbvisit Replicate, start and show the progression of the replication.

Dbvisit Replicate can be run with any OS user id. However, if ASM is used on mine, a bequeath connection must be made to the ASM instance and the user must be member of the dba or sysasm group.

Setup wizard

A setup wizard is included to generate the required commands to configure the environment and start the replication. The wizard also creates the necessary scripts to create the Dbvisit Replicate schema owner, grant the necessary privileges and create the environment (DDC) file. The setup wizard can be used as a starting point and then the configuration files can be edited to create more complex replications.

The setup wizard is logically split into four parts:

1. Describing the individual databases that are taking part in the replication.
2. Setting up the replication pairs to determine which are the source and target databases.
3. Choosing what to replicate, this can be tables and schemas.
4. Process configuration which includes hostnames, port numbers and notification settings.

The wizard does not make changes to the database, it creates text-based scripts (for SQL*Plus, dbvrep and shell/cmd.exe) that do the actual work. These scripts can edited to tailor it for specific needs before running them.

To recreate or reset the replication environment, the scripts can be rerun at anytime and can be used for scripted deployment into production environments. They will drop and recreate the replication setup.

The scripts created are (default names are listed here, using * for DDC name):

- *.ddc: configuration variables, used when running dbvrep during the setup or anytime in the future. Only basic variables are included here, as most of them are stored in mine database after setup. (See *-onetime.ddc and chapter on DDC files.)
- *-onetime.ddc: configuration variables; these are stored in DDC DB after the scripts are run.
- *-dbsetup*.sql: (re)creates the Dbvisit Replicate schema user and grants appropriate privileges. Because Oracle internal tables and views are involved, sysdba credentials are needed.
- *-grants*.sql: grants the Dbvisit Replicate schema user privileges on the replicated objects.
- *-setup.dbvrep: dbvrep script creating the Dbvisit Replicate schema tables and views, populating them with data and preparing selected schemas and tables for replication.
- Nextsteps.txt: a plaint text file describing the next steps to take to initiate the replication.
- *-wizard*.cfg: saves the configuration of the wizard after each step for the next run of the wizard or in case of a restart. This saves having to type all the information in again.
- *-all.sh / *-all.bat: sample script that invokes the scripts described above and then starts the replication processes.

The scripts contain all changes needed for both source and target database; running the all script once (e.g. on the source server) configures both source and target database.

Starting the wizard

Login as the user that is going to run Dbvisit Replicate[3] and start the Dbvisit Replicate Command Console and start the setup wizard:

On Windows:

```
Start > Dbvisit Replicate Command Console
dbvrep> setup wizard
```

[3]On Windows, the Dbvisit Replicate processes are by default installed as services and run as LOCAL SYSTEM.

On Linux/UNIX (tar file):

```
cd /usr/local/dbvisit/replicate (this may be different)
./dbvrep
dbvrep> setup wizard
```

On Linux (rpm file):

```
cd /home/oracle/dbvrep (this maybe different)
dbvrep
dbvrep> setup wizard
```

The setup wizard is usually run on the source server, although this is not strictly necessary. However, it makes the built-in checking of directories and auto-detection of tnsnames.ora more useful.

The setup scripts are created in user home directory (My Documents) – a subdirectory *DDC_NAME* is created. In this directory, sub-directories mine, mine_stage, apply, log and ddc_backup are created. These are used as defaults for various variable settings in DDC file and can be changed if needed.

Setup wizard example

This example show an Oracle-to-Oracle one-way replication using the following information:

Source database name:	`orcl`
Source server name:	`dbvisit210`
Target database name:	`orcl`
Target server name:	`dbvisit230`
TNS alias for source database:	`orcl_dbvisit210`
TNS alias for target database:	`orcl_dbvisit230`

Throughout the wizard, defaults are shown in [brackets]. Just press enter to accept them. The replication name in this example is “orcl”.

```
dbvrep> setup wizard
This wizard configures Dbvisit Replicate to start a replication process.
The setup wizard creates configuration scripts, which need to be run after the
wizard ends. No changes to the databases are made before that.
The progress is saved every time a list of databases, replications, etc. is shown.
It will be re-read if wizard is restarted and the same DDC name and
script path is selected.
Run the wizard now? [yes]
Accept end-user license agreement? (view/yes/no) [view] yes
Before starting the actual configuration, some basic information is needed. The DDC
name and script path determines where all files created by the wizard go
(and where to reread them if wizard is rerun) and the license key determines which
options are available for this configuration.
(DDC_NAME) - Please enter a name for this replication (suggestion: use the name of
the source database): [] orcl
(LICENSE_KEY) - Please enter your license key (or just enter "(trial)"): [(trial)]
(SETUP_SCRIPT_PATH) - Please enter a directory for location of configuration
scripts on this machine: [/home/oracle/orcl]
```

Network configuration (TNS)

Dbvisit Replicate will detect tnsnames.ora locations on your system. Choose the TNS configuration that is applicable.

```
Network configuration files were detected on this system in these locations:
/u01/app/oracle/product/11.1.0/db_1/network/admin
/u03/app/oracle/product/11.2.0/dbhome_3/network/admin
/u01/app/oracle/product/11.2.0/xe/network/admin
(TNS_ADMIN) - Please enter TNS configuration directory for this machine:
[/u01/app/oracle/product/11.1.0/db_1/network/admin]
```

Describing the databases

During this step information on the databases used in the replication is collected.

The result is a SQL*Plus scripts that recreates the schema user and grants it required privileges.

Please note that being a SQL*Plus script, the resulting file contains plain text passwords.

Describe the first database:

```
Step 1 - Describe databases
=======================================
The first step is to describe databases used in the replication. There are usually
two of them (source and target); however, there can be just one (when source and
target is the same) or more than two (one-to-many or other complex configurations.)
Let's configure the database, describing it's type, connectivity, user names etc.
What type of database is this? (Oracle/MySQL/MSSQL): [Oracle]
Please enter database TNS alias: [] orcl_dbvisit210
Please enter SYSDBA user name: [SYS]
Please enter password for this user: [change_on_install] ******
Please enter user with DBA role on the target database: [SYSTEM]
Please enter password for this user: [manager] ******
Connecting to database orcl_dbvisit210 as SYSTEM to query list of tablespaces and to
detect ASM (by looking whether any redo logs or archived logs are stored in ASM).
Enter the Dbvisit Replicate owner and apply user (this user will be created by this
script): [dbvrep]
Please enter password for this user: [dbvpasswd]
Permanent tablespaces detected on the database: USERS, SYSAUX, SYSTEM.
Please enter default permanent tablespace for this user: [USERS]
Temporary tablespaces detected on the database: TEMP.
Please enter default temporary tablespace for this user: [TEMP]
Does the database use ASM? (yes/no): [YES]

Following databases are now configured:
1: Oracle orcl_dbvisit210, SYS/***, SYSTEM/***, dbvrep/***, USERS/TEMP, dbvrep/,
ASM:YES
Enter number of database to modify it, or "add", or "done": [add]
```

Adding the second database:

```
Let's configure the database, describing it's type, connectivity, user names etc.
What type of database is this? (Oracle/MySQL/MSSQL): [Oracle]
Please enter database TNS alias: [] orcl_dbvisit230
Please enter SYSDBA user name: [SYS]
Please enter password for this user: [change_on_install] ******
Please enter user with DBA role on the target database: [SYSTEM]
Please enter password for this user: [manager] ******
Connecting to database orcl_dbvisit230 as SYSTEM to query list of tablespaces and to
detect ASM (by looking whether any redo logs or archived logs are stored in ASM).
Enter the Dbvisit Replicate owner and apply user (this user will be created by this
script): [dbvrep]
Please enter password for this user: [dbvpasswd]
Permanent tablespaces detected on the database: USERS, SYSAUX, SYSTEM.
Please enter default permanent tablespace for this user: [USERS]
Temporary tablespaces detected on the database: TEMP.
Please enter default temporary tablespace for this user: [TEMP]
Does the database use ASM? (yes/no): [YES]

Following databases are now configured:
1: Oracle orcl_dbvisit210, SYS/***, SYSTEM/***, dbvrep/***, USERS/TEMP, dbvrep/,
ASM:YES
2: Oracle orcl_dbvisit230, SYS/***, SYSTEM/***, dbvrep/***, USERS/TEMP, dbvrep/,
ASM:YES
Enter number of database to modify it or "done": [done]
```

Please note:

1. At this stage specific details about which are the source and target databases are not yet required.

Replication pairs

During this step the source and target databases are set for each replication pair.

```
Step 2 - Replication pairs
==========================================
The second step is to set source and targets for each replication pair. This is
usually just choosing the first database as source and the second one as target,
but many more configurations are possible.
Let's configure the replication pair, selecting source and target.
Following databases are described:
1: orcl_dbvisit210 (Oracle)
2: orcl_dbvisit230 (Oracle)
Select source database: [1]
Select target database: [2]
Will be DDL replication enabled? (If YES, the script will grant more privileges to
the Dbvisit Replicate users and enable database-wide supplemental logging):    [yes]
Use fetcher to offload the mining to a different server? (yes/no) [no]
(NETWORK_QUALITY) - Please specify your network type (LAN or WAN). Autoconfigures
timeouts, use of compression etc. [LAN]
Following replication pairs are now configured:
1: orcl_dbvisit210 (Oracle) ==> orcl_dbvisit230 (Oracle), DDL: yes, fetcher: no,
process suffix: (no suffix), network: LAN
Enter "1" to modify or "done": [done]
```

Process configuration

This steps configures the replication processes for each replication. This includes setting the

- hostnames
- port numbers
- Notification settings and thresholds

Add process information for the MINE process:

```
Step 4 - Process configuration
==========================================
The fourth step is to configure the replication processes for each replication.
Although most options have reasonable defaults, manual input will be required.
Following processes are defined:
1: MINE on orcl_dbvisit210
  Not configured.
2: APPLY on orcl_dbvisit230
  Not configured.
Enter number of process to modify it, or "done": [1]
Fully qualified name of the server for the process (usually co-located with the
database, unless mine is offloaded using fetcher): [] dbvisit210
Enable email notifications about problems (yes/no)? [YES]
Enable SNMP traps/notifications about problems (yes/no)? [NO]
Directory with DDC file and default where to create log files etc. (recommended:
same as global setting, if possible)? [/home/oracle/orcl]
Following settings were pre-filled with defaults or your reloaded settings:
----------------------------------------
[MINE_LISTEN_INTERFACE]: Network listen interface: dbvisit210:7901
[MINE_DATABASE]: Database TNS: orcl_dbvisit210
[TNS_ADMIN]: tnsnames.ora path: /u01/app/oracle/product/11.1.0/db_1/network/admin
[MINE_USER]: Replicate database username: dbvrep
[MINE_PASSWORD]: Replicate database password: *********
[ORACLE_HOME]: ASM ORACLE_HOME: /u01/app/oracle/product/11.1.0/db_1
[MINE_PLOG]: Filemask for generated plogs: /home/oracle/orcl/mine/%S.%E
```

```
[LOG_FILE]: General log file: /home/oracle/orcl/log/dbvrep_%N_%D.%E
[LOG_FILE_TRACE]: Error traces: /home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E
Checking that these settings are valid...
Do you want change any of the settings? [no]
(MAILCFG_SMTP_SERVER) - Specify mail (SMTP) server hostname: [] mail.dbvisit.com
(MAILCFG_USE_SSL) - Use SSL to connect to the mail server (yes/no)? [] no
(MAILCFG_PORT) - Specify port for SMTP server (usually 25; for SSL, usually 465):
[25]
(MAILCFG_AUTH_USER) - If SMTP server requires username and password, specify the
username. Specify OFF if no login is required: [OFF]
(MAILCFG_AUTH_PASSWD) - If SMTP server requires username and password, specify the
password. Specify OFF if no login is required: [OFF]
(MAILCFG_FROM) - Specify the email address to be used as 'From:' address:
[oracle@dbvisit210] oracle@dbvisit210.com
Following processes are defined:
1: MINE on orcl_dbvisit210
  Host: dbvisit210, SMTP: YES, SNMP: NO
2: APPLY on orcl_dbvisit230
  Not configured.
```

Add process information for the APPLY process:

```
Enter number of process to modify it, or "done": [2]
Fully qualified name of the server for the process (usually co-located with the
database, unless mine is offloaded using fetcher): [] dbvisit230
Enable email notifications about problems (yes/no)? [YES]
Enable SNMP traps/notifications about problems (yes/no)? [NO]
Directory with DDC file and default where to create log files etc. (recommended:
same as global setting, if possible)? [/home/oracle/orcl]
Following settings were pre-filled with defaults or your reloaded settings:
----------------------------------------
[APPLY_LISTEN_INTERFACE]: Network listen interface: dbvisit230:7902
[APPLY_DATABASE]: Database TNS: orcl_dbvisit230
[TNS_ADMIN]: tnsnames.ora path: /u01/app/oracle/product/11.1.0/db_1/network/admin
[APPLY_USER]: Replicate database username: dbvrep
[APPLY_PASSWORD]: Replicate database password: *********
[APPLY_STAGING_DIR]: Directory for received plogs: /home/oracle/orcl/apply
[LOG_FILE]: General log file: /home/oracle/orcl/log/dbvrep_%N_%D.%E
[LOG_FILE_TRACE]: Error traces: /home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E
Checking that these settings are valid...
Do you want change any of the settings? [no]
```

Add notification thresholds settings:

```
(MAILCFG_SMTP_SERVER) - Specify mail (SMTP) server hostname: [mail.dbvisit.com]
(MAILCFG_USE_SSL) - Use SSL to connect to the mail server (yes/no)? [no]
(MAILCFG_PORT) - Specify port for SMTP server (usually 25; for SSL, usually 465):
[25]
(MAILCFG_AUTH_USER) - If SMTP server requires username and password, specify the
username. Specify OFF if no login is required: [OFF]
(MAILCFG_AUTH_PASSWD) - If SMTP server requires username and password, specify the
password. Specify OFF if no login is required: [OFF]
(MAILCFG_FROM) - Specify the email address to be used as 'From:' address:
[oracle@dbvisit210.com]
(NOTIFY_ALL_EMAIL) - Specify 'To:' address(es) to receive all emails (separate
multiple addresses by ',') - OFF for empty list (you can specify success/alert
addressee separately later). [OFF] replicate@mycompany.com
(NOTIFY_SUCCESS_EMAIL) - Specify 'To:' adress(es) for hearbeat/progress emails - OFF
for empty list. [OFF]
(NOTIFY_ALERT_EMAIL) - Specify 'To:' adress(es) for alert (problem-reporting) emails
- OFF for empty list. [OFF]
(NOTIFY_SEQUENCE_DIFFERENCE) - How many plogs (redo plogs) must apply lag behind
mine (or mine behind fetcher) to send notification? [10]
(NOTIFY_SCN_DIFFERENCE) - How many SCNs must apply lag behind mine to send
notification? [1000]
(NOTIFY_PROGRESS_DIFFERENCE_PERC) - How much must apply lag behind mine for any
table to send notification (in %)? [10]
```

```
(NOTIFY_CONFLICT_THRESHOLD) - How many conflicts must apply encounter to send
notification? [100]
(NOTIFY_DAILY_LIST_PROGRESS_TIME24) - When to send periodic progress summary email?
(use 24-h format, separate multiple times by semicolon, e.g.
'0630:1230:1830' or OFF to disable) [0700]
(NOTIFY_SEND_HEARTBEAT_TIME24) - When to send periodic heartbeat email? (use 24-h
format, separate multiple times by semicolon, e.g. '0630:1230:1830' or OFF to
disable) [0800:1300]
Following processes are defined:
1: MINE on orcl_dbvisit210
  Host: dbvisit210, SMTP: YES, SNMP: NO
2: APPLY on orcl_dbvisit230
  Host: dbvisit230, SMTP: YES, SNMP: NO
Enter number of process to modify it, or "done": [done]
```

Final wrap-up

All the information about the replication process has been obtained and now the replication environment can be created.

```
Created file /home/oracle/orcl/orcl-MINE.ddc.
Created file /home/oracle/orcl/orcl-APPLY.ddc.
Created file /home/oracle/orcl/orcl-dbsetup_orcl_dbvisit210.sql.
Created file /home/oracle/orcl/orcl-dbsetup_orcl_dbvisit230.sql.
Created file /home/oracle/orcl/orcl-setup.dbvrep.
Created file /home/oracle/orcl/orcl-grants_orcl_dbvisit210.sql.
Created file /home/oracle/orcl/orcl-grants_orcl_dbvisit230.sql.
Created file /home/oracle/orcl/orcl-onetime.ddc.
Created file /home/oracle/orcl/orcl-run-dbvisit210.sh.
Created file /home/oracle/orcl/orcl-run-dbvisit230.sh.
Created file /home/oracle/orcl/Nextsteps.txt.
Created file /home/oracle/orcl/orcl-all.sh.
================================================================================
Dbvisit Replicate wizard completed
Script /home/oracle/orcl/orcl-all.sh created. This runs all the above created
scripts. Please exit out of dbvrep, review and run script as current user to
setup and start Dbvisit Replicate.
================================================================================
Optionally, the script can be invoked now by this wizard.
Run this script now? (yes/no) [NO]
dbvrep> exit
```

The Dbvisit Replicate wizard has completed and has created the necessary replication environment.

Exit out of dbvrep and run the *-all.sh or *all.bat script created by the wizard

In the above example the script is called orcl-all.sh. This script can be run at any time to reset or recreate the replication environment. The Nextsteps.txt explains the next steps to initiate the replication.

This completes the Dbvisit Replicate setup and configuration.

Before running the script, review the created scripts as indicated by the Nextsteps.txt (shown at the end of all.* script). For example:

- Review the location of dbvrep(.exe) if it differs among the nodes if RAC is used.
- Check TNS_ADMIN and ORACLE_HOME paths and make sure they are correct on all the different servers involved.

After testing, consider adding the starting of dbvrep processes to init scripts on Linux/Unix to automatically start them on server reboot. (This is not needed on Windows, as processes are registered as services by default).

Outcome of running the scripts created by the setup wizard.

Running the *-all.sh script will configure and initiate the replication. It does the following:

- Create the Dbvisit Replicate schemas in the source and target databases (default username dbvrep)
- Grant the necessary privileges to the Dbvisit Replicate schema.
- Create the Dbvisit Replicate repository in the source and target databases.
- Load the Dbvisit Replicate configuration file (DDC) into the Dbvisit Replicate repository. The DDC file is like the init.ora for Oracle and contains the settings for the replication.
- Prepare the schemas or objects for replication.

Example of running the orcl-all.sh

```
Setting up Dbvisit Replicate configuration
Configure database orcl_dbvisit210...
Configure database orcl_dbvisit230...
Object grants for database orcl...
Object grants for database orcl_dbvisit230...
Setting up the configuration
Initializing.....done
WARN-1850: No DDC DB available, dictionary table does not exist.
DDC loaded from database (0 variables).
Dbvisit Replicate
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/orcl/orcl-onetime.ddc loaded.
MINE: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
APPLY: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR EXIT
Variable ON_ERROR set to EXIT for process *.
dbvrep> ENGINE SETUP MINE DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP MINE CREATE DICTIONARY
dbvrep> ENGINE SETUP MINE LOAD DICTIONARY
Supplemental logging on database set.
Loading dictionary table DBRSCOL$
Loading dictionary table DBRSOBJ$
Loading dictionary table DBRSTAB$
Loading dictionary table DBRSUSER$
Loading dictionary table DBRSV_$DATABASE
dbvrep> ENGINE SETUP APPLY DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP APPLY CREATE DICTIONARY
dbvrep> ENGINE SETUP APPLY LOAD DICTIONARY
dbvrep> ENGINE PREPARE_DP SETUP CLEAR
dbvrep> ENGINE SETUP PAIR MINE AND APPLY
ID of mine proces is A8793C78-1EEB-11E1-816F-70EC8BE7EEF4. If not using DDC in
database, set MINE_UNIQUE_ID to this value.
Table dbvrep.DBRSCOL$ instantiated at SCN 20615855
Table dbvrep.DBRSOBJ$ instantiated at SCN 20615855
Table dbvrep.DBRSTAB$ instantiated at SCN 20615855
Table dbvrep.DBRSUSER$ instantiated at SCN 20615855
Table dbvrep.DBRSV_$DATABASE instantiated at SCN 20615855
1 applier SCN set.
dbvrep> PREPARE OFFLINE SCHEMA AVI
dbvrep> PREPARE OFFLINE TABLE SCOTT.AVI_OBJECTS
Table SCOTT.AVI_OBJECTS instantiated at SCN 20626026
dbvrep> ENGINE PREPARE_DP WRITE DP_NETWORKLINK DIRECTORY DATA_PUMP_DIR FILE
/home/oracle/orcl/APPLY.sh DBLINK orcl USERID SYS/oracle@orcl_dbvisit230
```

```
Created Data Pump script /home/oracle/orcl/APPLY.sh, using network import.
dbvrep> create ddcdb from ddcfile
DDC loaded into database (131 variables).
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR SKIP
Variable ON_ERROR set to SKIP for process *.
OK-0: Completed successfully.
```

The replication has been configured and installed. The following steps need to be done to complete and start the replication:

```
1) Create the necessary directory(ies) on the servers:
dbvisit230: /home/oracle/orcl

2) Copy the DDC files to the server(s) where the processes will run:
/home/oracle/orcl/orcl-MINE.ddc
/home/oracle/orcl/orcl-APPLY.ddc

3) Review that path to dbvrep executable is correct in the run scripts:
/home/oracle/orcl/orcl-run-dbvisit230.sh
/home/oracle/orcl/orcl-run-dbvisit210.sh

4) Copy the run script to the server(s) where the processes will run:
/home/oracle/orcl/orcl-run-dbvisit230.sh
/home/oracle/orcl/orcl-run-dbvisit210.sh

5) Ensure firewall is open for listen interfaces dbvisit210:7901, dbvisit230:7902
used by the processes.

6) Start the replication processes on all servers:
/home/oracle/orcl/orcl-run-dbvisit230.sh
/home/oracle/orcl/orcl-run-dbvisit210.sh

7) Start the console to monitor the progress:
/usr/bin/dbvrep --ddcfile /home/oracle/orcl/orcl-MINE.ddc
```

Starting the replication

As indicated by the Nextsteps.txt , the replication can be started by following the steps that are listed above.

Starting the replication does the following:

- Starts the MINE and APPLY process in the background
- Creates the necessary log and plog directories.
- Once the APPLY and MINE process has been started, there is in INITIALIZATION process. This replicates the contents of the Dbvisit Replication repository from the source to the target database. The replication can not be started until this phase has been completed.

Starting the console

The dbvrep console can be started to monitor the real time status of the replication. The command to start the console is also shown in the Nextsteps.txt. In the above example it is:

```
/usr/bin/dbvrep --ddcfile /home/oracle/orcl/orcl-MINE.ddc
```

The console can be started on either the Mine server or the Apply server. It can also be started on an independent server as long as the following holds true:

- The Dbvisit Replicate software has been installed
- It has a copy of the DDC file
- There is a TNS connection to the Mine and Apply databases

Manually Replicating new objects

Once the replication has been configured and is running, new objects can be added to the replication. If a whole schema is replicated, then new tables will automatically be replicated to the target database (The DDL and DML will be replicated).

New tables can be replicated with the command:

```
dbvrep> PREPARE TABLE schema.table_name
```

The table needs to exist on both the source and target before this command can be given. If the table was created recently, make sure that mine has already parsed the logs beyond the time of the table creation, so it learned that the table exists.

Ensure that the appropriate privileges are given to the Dbvisit Replicate owner on both the source and target. The privileges are:

```
grant select, update, insert, delete to dbvrep;
```

Replicating one schema to a different schema in same database

The source and target databases do not need to be separate – in fact, the replication can replicate back to the source database. Provided that the target tables/schemas are different than the source tables/schemas.

To setup such a configuration, specify the same database TNS identifiers during the first step of the setup (case-sensitive).

Additional step

During the setup wizard the following rename question is asked. By answering yes, a rename for the schema can be given.

```
Specify rename name for any of the specified schemas (yes/no): [yes]
Rename name for schema AVI (empty means no rename): []
```

Please note: DDL replication is not supported when replicating from one schema to another schema as the objects have a different name.

Configure different names at source and target

During the setup wizard the following rename question is asked. By answering yes, a rename for the schema can be given.

```
Specify rename name for any of the specified schemas (yes/no): [yes]
Rename name for schema AVI (empty means no rename): []
```

Please note: DDL replication is not supported when replicating from one schema to another schema as the objects have a different name

MySQL replication

In all replications, source must be an Oracle database. Thus, two-way replication with MySQL is not supported.

All target tables must be stored in InnoDB engine, as Dbvisit Replicate depends on correct transaction handling to reliably restart after a shutdown or failure. The internal tables are stored in InnoDB engine as well.

Currently, no DDL replication is supported for MySQL.

Oracle → MySQL replication setup example

The following example shows Oracle to MySQL replication. In this example, Oracle and MySQL are running on a single server.

Environment: Virtualbox using Pre-Built Developer VMs (for Oracle VM VirtualBox)
Template name: Database App Development VM
Oracle Linux 5 (32bit)
Oracle RDBMS 11gR2
MySQL Server 5.0.77

MySQL is not installed on the pre-built VM, but this can be installed with:

```
# yum mysql mysql-server
```

Start MySQL

```
/etc/init.d/mysqld start
```

Install Dbvisit Replicate using the 32bit RPM installation file (as root):

```
# rpm -ivh dbvisit_replicate-2.0.02-1.i386.rpm
Preparing...            ########################################### [100%]
 1:dbvisit_replicate    ########################################### [100%]
```

The following Oracle table is going to be replicated. This is a copy of the sys.dba_objects table but created in the SCOTT schema.

Create this table in Oracle:

```
CREATE TABLE "SCOTT"."AVI_OBJECTS"
 (    "OWNER" VARCHAR2(30),
      "OBJECT_NAME" VARCHAR2(128),
      "SUBOBJECT_NAME" VARCHAR2(30),
      "OBJECT_ID" NUMBER,
      "DATA_OBJECT_ID" NUMBER,
      "OBJECT_TYPE" VARCHAR2(19),
      "CREATED" DATE,
      "LAST_DDL_TIME" DATE,
      "TIMESTAMP" VARCHAR2(19),
      "STATUS" VARCHAR2(7),
      "TEMPORARY" VARCHAR2(1),
      "GENERATED" VARCHAR2(1),
      "SECONDARY" VARCHAR2(1),
      "NAMESPACE" NUMBER,
      "EDITION_NAME" VARCHAR2(30)
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"
```

Create this same table in MySQL. The syntax is slightly different to Oracle.

First user scott (or database) has to be created first in MySQL.

```
mysql> create database scott;
Query OK, 1 row affected (0.00 sec)
mysql> use scott;
Database changed
```

Now the MySQL table can be created:

```
DROP TABLE IF EXISTS `avi_objects`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `avi_objects` (
  `OWNER` varchar(30) default NULL,
  `OBJECT_NAME` varchar(128) default NULL,
  `SUBOBJECT_NAME` varchar(30) default NULL,
  `OBJECT_ID` bigint(20) default NULL,
  `DATA_OBJECT_ID` bigint(20) default NULL,
  `OBJECT_TYPE` varchar(19) default NULL,
  `CREATED` date default NULL,
  `LAST_DDL_TIME` date default NULL,
  `TIMESTAMP` varchar(19) default NULL,
  `STATUS` varchar(7) default NULL,
  `TEMPORARY` varchar(1) default NULL,
  `GENERATED` varchar(1) default NULL,
  `SECONDARY` varchar(1) default NULL,
  `NAMESPACE` bigint(20) default NULL,
  `EDITION_NAME` varchar(30) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Start Dbvisit Replicate:

```
$ dbvrep
Initializing.....done
Dbvisit Replicate
Copyright (C) Dbvisit Software Limited.  All rights reserved.
No DDC file loaded.
Run "setup wizard" to start the configuration wizard or try "help" to see all
commands available.
dbvrep>
```

Start the setup wizard and follow the instructions. Most of the defaults can be chosen. Note that MySQL is entered as the target database:

```
dbvrep> setup wizard
This wizard configures Dbvisit Replicate to start a replication process.

The setup wizard creates configuration scripts, which need to be run after the
wizard ends. No changes to the databases are made before that.

The progress is saved every time a list of databases, replications, etc. is
shown. It will be re-read if wizard is restarted and the same DDC name and
script path is selected.
Run the wizard now? [yes]
Accept end-user license agreement? (view/yes/no) [view] yes

Before starting the actual configuration, some basic information is needed. The
DDC name and script path determines where all files created by the wizard go
(and where to reread them if wizard is rerun) and the license key determines
which options are available for this configuration.
(DDC_NAME) - Please enter a name for this replication (suggestion: use the name of
the source database): [] orcl
(LICENSE_KEY) - Please enter your license key (or just enter "(trial)"):
[(trial)]
```

```
(SETUP_SCRIPT_PATH) - Please enter a directory for location of configuration scripts
on this machine: [/home/oracle/orcl]

Network configuration files were detected on this system in these locations:
/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin
(TNS_ADMIN) - Please enter TNS configuration directory for this machine: [
/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin]

Step 1 - Describe databases
=======================================
The first step is to describe databases used in the replication. There are
usually two of them (source and target); however, there can be just one (when source
and target is the same) or more than two (one-to-many or other complex
configurations.)
Let's configure the database, describing it's type, connectivity, user names
etc.
What type of database is this? (Oracle/MySQL/MSSQL): [Oracle]
Please enter database TNS alias: [] ttorcl
Please enter SYSDBA user name: [SYS]
Please enter password for this user: [change_on_install] ******
Please enter user with DBA role on the target database: [SYSTEM]
Please enter password for this user: [manager] ******

Connecting to database ttorcl as SYSTEM to query list of tablespaces and to
detect ASM (by looking whether any redo logs or archived logs are stored in
ASM).
Enter the Dbvisit Replicate owner and apply user (this user will be created by  this
script): [dbvrep]
Please enter password for this user: [dbvpasswd]

Permanent tablespaces detected on the database: APEX_1295922881855015, EXAMPLE,
FLOW_1170420963682633, FLOW_1194425963955800, FLOW_1218408858999342,
FLOW_1242310449730067, FLOW_1266412439758696, USERS, SYSAUX, SYSTEM.
Please enter default permanent tablespace for this user: [APEX_1295922881855015]
USERS

Temporary tablespaces detected on the database: TEMP.
Please enter default temporary tablespace for this user: [TEMP]
Does the database use ASM? (yes/no): [NO]

Following databases are now configured:
1: Oracle ttorcl, SYS/***, SYSTEM/***, dbvrep/***, USERS/TEMP, dbvrep/, ASM:NO
Enter number of database to modify it, or "add", or "done": [add]
Let's configure the database, describing it's type, connectivity, user names
etc.
What type of database is this? (Oracle/MySQL/MSSQL): [Oracle] MySQL
Please enter database hostname: [localhost]
Please enter user name of an administrator: [root]
Please enter password for this user: [password] ******
Enter the user to log into apply database: [root]
Please enter password for this user: [password]
Enter the database (schema) to use for Dbvisit Replicate internal data (will be
created in the script): [dbvrep]

Following databases are now configured:
1: Oracle ttorcl, SYS/***, SYSTEM/***, dbvrep/***, USERS/TEMP, dbvrep/, ASM:NO
2: MySQL database=dbvrep;host=localhost, root/***, root/***, root/***, /,
dbvrep/, ASM:n/a
Enter number of database to modify it or "done": [done]

Step 2 - Replication pairs
```

```
========================================
The second step is to set source and targets for each replication pair. This is
usually just choosing the first database as source and the second one as
target, but many more configurations are possible.
Let's configure the replication pair, selecting source and target.
Following databases are described:
1: ttorcl (Oracle)
2: database=dbvrep;host=localhost (MySQL) (cannot be source, is not Oracle)
Select source database: [1]
Select target database: [2]
Use fetcher to offload the mining to a different server? (yes/no) [no]
(NETWORK_QUALITY) - Please specify your network type (LAN or WAN). Autoconfigures
timeouts, use of compression etc. [LAN]

Following replication pairs are now configured:
1: ttorcl (Oracle) ==> database=dbvrep;host=localhost (MySQL), DDL: no,
fetcher: no, process suffix: (no suffix), network: LAN
Enter "1" to modify or "done": [done]

Step 3 - Replicated tables
========================================
The third step is to choose the schemas and tables to be replicated. If the
databases are reachable, the tables are checked for existence, datatype
support, etc., schemas are queried for tables. Note that all messages are
merely hints/warnings and may be ignored if issues are rectified before the
scripts are actually executed.

Note the following assumptions are made in this wizard - which can be
modified by editing the resulting script:
1. All replicated tables have a primary key defined.
2. All columns of the tables and all tables of the specified schemas are
replicated, and if DDL support is enabled, whenever a new column/table is
added, this should be replicate as well.
3. If an apply conflict arises, the default option is to try again repeatedly,
until a different option is given or the underlying issue is resolved.
4. If an apply conflict arises, no more data will be replicated until the issue
is resolved or ignored.

Following tables are defined for replication pairs:
1: ttorcl (Oracle) ==> database=dbvrep;host=localhost (MySQL), DDL: no, suffix:
(no suffix)
  No tables defined.
Enter number of replication pair to modify it, or "done": [1]

To replicate an entire schema, please enter the schemas to be replicated. Enter
through a comma-delimited list, or enter one by one. Hit Enter when finished
Enter the list of schemas: []

To replicate individual tables, please enter the table names. Do not enter
tables in quotes (" ").
Enter through a comma-delimited list, or enter one by one. Hit Enter when
finished.
Please use fully qualified names (ie owner.table_name).
Enter the list of tables: [] scott.avi_objects
Enter the list of tables: []
Specify rename name or filter condition for any of the specified tables (yes/no): [
no]

Following tables are defined for replication pairs:
1: ttorcl (Oracle) ==> database=dbvrep;host=localhost (MySQL), DDL: no, suffix:
```

```
(no suffix)
  scott.avi_objects
Enter number of replication pair to modify it, or "done": [done]

Step 4 - Process configuration
======================================
The fourth step is to configure the replication processes for each replication.
Although most options have reasonable defaults, manual input will be required.

Following processes are defined:
1: MINE on ttorcl
  Not configured.
2: APPLY on database=dbvrep;host=localhost
  Not configured.
Enter number of process to modify it, or "done": [1]
Fully qualified name of the server for the process (usually co-located with the
database, unless mine is offloaded using fetcher): [] dhcppc13
Enable email notifications about problems (yes/no)? [YES] NO
Enable SNMP traps/notifications about problems (yes/no)? [NO]
Directory with DDC file and default where to create log files etc. (recommended:
same as global setting, if possible)? [/home/oracle/orcl]

Following settings were pre-filled with defaults or your reloaded settings:
----------------------------------------
[MINE_LISTEN_INTERFACE]: Network listen interface: dhcppc13:7901
[MINE_DATABASE]: Database TNS: ttorcl
[TNS_ADMIN]: tnsnames.ora path:
/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin
[MINE_USER]: Replicate database username: dbvrep
[MINE_PASSWORD]: Replicate database password: *********
[MINE_PLOG]: Filemask for generated plogs: /home/oracle/orcl/mine/%S.%E
[LOG_FILE]: General log file: /home/oracle/orcl/log/dbvrep_%N_%D.%E
[LOG_FILE_TRACE]: Error traces:
/home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E

Checking that these settings are valid...
Do you want change any of the settings? [no]

Following processes are defined:
1: MINE on ttorcl
  Host: localhost, SMTP: NO, SNMP: NO
2: APPLY on database=dbvrep;host=localhost
  Not configured.
Enter number of process to modify it, or "done": [2]
Fully qualified name of the server for the process (usually co-located with the
database, unless mine is offloaded using fetcher): [] dhcppc13
Enable email notifications about problems (yes/no)? [YES] NO
Enable SNMP traps/notifications about problems (yes/no)? [NO]
Directory with DDC file and default where to create log files etc. (recommended:
same as global setting, if possible)? [/home/oracle/orcl]

Following settings were pre-filled with defaults or your reloaded settings:
----------------------------------------
[APPLY_LISTEN_INTERFACE]: Network listen interface: dhcppc13:7902
[APPLY_DATABASE]: Database MySQL connection string:
database=dbvrep;host=localhost
[APPLY_USER]: Replicate database username: root
[APPLY_PASSWORD]: Replicate database password: ********
[APPLY_SCHEMA]: Replicate database (schema): dbvrep
[APPLY_STAGING_DIR]: Directory for received plogs: /home/oracle/orcl/apply
```

```
[LOG_FILE]: General log file: /home/oracle/orcl/log/dbvrep_%N_%D.%E
[LOG_FILE_TRACE]: Error traces:
/home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E

Checking that these settings are valid...
Do you want change any of the settings? [no]

Following processes are defined:
1: MINE on ttorcl
  Host: localhost, SMTP: NO, SNMP: NO
2: APPLY on database=dbvrep;host=localhost
  Host: localhost, SMTP: NO, SNMP: NO
Enter number of process to modify it, or "done": [done]
Created file /home/oracle/orcl/orcl-MINE.ddc.
Created file /home/oracle/orcl/orcl-APPLY.ddc.
Created file /home/oracle/orcl/orcl-dbsetup_ttorcl.sql.
Created file /home/oracle/orcl/orcl-dbsetup_database_dbvrep_host_localhost.sql.
Created file /home/oracle/orcl/orcl-setup.dbvrep.
Created file /home/oracle/orcl/orcl-grants_ttorcl.sql.
Created file /home/oracle/orcl/orcl-grants_database_dbvrep_host_localhost.sql.
Created file /home/oracle/orcl/orcl-onetime.ddc.
Created file /home/oracle/orcl/orcl-run-dhcppc13.sh.
Created file /home/oracle/orcl/Nextsteps.txt.
Created file /home/oracle/orcl/orcl-all.sh.
================================================================================

Dbvisit Replicate wizard completed

Script /home/oracle/orcl/orcl-all.sh created. This runs all the above created
scripts. Please exit out of dbvrep, review and run script as current user to
setup and start Dbvisit Replicate.
================================================================================
Optionally, the script can be invoked now by this wizard.
Run this script now? (yes/no) [NO]
```

Exit out of Dbvisit Replicate and start the script that has been generated by the setup wizard:

```
$ ./orcl-all.sh
Setting up Dbvisit Replicate configuration
Configure database ttorcl...
Configure database database=dbvrep
Object grants for database ttorcl...
Object grants for database database=dbvrep
Setting up the configuration
Initializing.....done
WARN-1850: No DDC DB available, dictionary table does not exist.
DDC loaded from database (0 variables).
OK-9056: Directory /home/oracle/orcl/ddc_backup set by variable DDC_BACKUP_DIR does
not exist on this system.
OK-9056: Directory /home/oracle/orcl/ddc_backup set by variable DDC_BACKUP_DIR does
not exist on this system.
Dbvisit Replicate version 2.1.04.1047
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/orcl/orcl-onetime.ddc loaded.
MINE: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
APPLY: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR EXIT
Variable ON_ERROR set to EXIT for process *.
```

```
dbvrep> ENGINE SETUP MINE DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP MINE CREATE DICTIONARY
dbvrep> ENGINE SETUP MINE LOAD DICTIONARY
Supplemental logging on database set.
Loading dictionary table DBRSCOL$
Loading dictionary table DBRSOBJ$
Loading dictionary table DBRSTAB$
Loading dictionary table DBRSUSER$
Loading dictionary table DBRSV_$DATABASE
dbvrep> ENGINE SETUP APPLY DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP APPLY CREATE DICTIONARY
dbvrep> ENGINE SETUP APPLY LOAD DICTIONARY
dbvrep> ENGINE SETUP PAIR MINE AND APPLY
ID of mine proces is 276F6266-243F-11E1-A450-BD91EEFD88F8. If not using DDC in
database, set MINE_UNIQUE_ID to this value.
Table dbvrep.DBRSCOL$ instantiated at SCN 7436161
Table dbvrep.DBRSOBJ$ instantiated at SCN 7436161
Table dbvrep.DBRSTAB$ instantiated at SCN 7436161
Table dbvrep.DBRSUSER$ instantiated at SCN 7436161
Table dbvrep.DBRSV_$DATABASE instantiated at SCN 7436161
1 applier SCN set.
dbvrep> PREPARE OFFLINE TABLE scott.avi_objects NODDL
Table scott.avi_objects instantiated at SCN 7437937
dbvrep> create ddcdb from ddcfile
DDC loaded into database (131 variables).
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR SKIP
Variable ON_ERROR set to SKIP for process *.
OK-0: Completed successfully.
1) Create the necessary directory(ies) on the servers:

2) Copy the DDC files to the server(s) where the processes will run:
/home/oracle/orcl/orcl-MINE.ddc
/home/oracle/orcl/orcl-APPLY.ddc

3) Review that path to dbvrep executable is correct in the run scripts:
/home/oracle/orcl/orcl-run-dhcppc13.sh

4) Copy the run script to the server(s) where the processes will run:
/home/oracle/orcl/orcl-run-dhcppc13.sh

5) Ensure firewall is open for listen interfaces dhcppc13:7901, dhcppc13:7902 used
by the processes.

6) Start the replication processes on all servers:
/home/oracle/orcl/orcl-run-dhcppc13.sh

7) Start the console to monitor the progress:
/usr/bin/dbvrep --ddcfile /home/oracle/orcl/orcl-MINE.ddc
```

Start the replication process by running the orcl-run-dhcppc13.sh script. This starts both the Mine and Apply process:

```
$ ./orcl-run-dhcppc13.sh
Initializing.....done
OK-9056: Directory /home/oracle/orcl/ddc_backup set by variable DDC_BACKUP_DIR does
not exist on this system.
```

```
OK-9056: Directory /home/oracle/orcl/ddc_backup set by variable DDC_BACKUP_DIR does
not exist on this system.
DDC loaded from database (131 variables).
Dbvisit Replicate version 2.1.04.1047
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/orcl/orcl-APPLY.ddc loaded.
Starting process APPLY...Created directory /home/oracle/orcl/ddc_backup
Created directory /home/oracle/orcl/log/
Created directory /home/oracle/orcl/log/trace/
Created directory /home/oracle/orcl/apply
started
Initializing.....[oracle@dhcppc13 orcl]$ done
DDC loaded from database (131 variables).
Dbvisit Replicate version 2.1.04.1047
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/orcl/orcl-MINE.ddc loaded.
Starting process MINE...Created directory /home/oracle/orcl/mine/
started
```

Dbvisit Replicate has now been configured and started in the background. The Dbvisit Replicate can be started in the foreground so that progress can be monitored:

```
$ /usr/bin/dbvrep --ddcfile /home/oracle/orcl/orcl-MINE.ddc
Initializing.....done
DDC loaded from database (134 variables).
Dbvisit Replicate version 2.1.04.1047
Copyright (C) Dbvisit Software Limited.  All rights reserved.
-MINE IS running, initialization NOT yet complete. Currently at plog 264 and SCN
6355055 (08/12/2011 20:11:24).
APPLY IS running, initialization NOT yet complete. Currently at plog 263 and SCN
6353874 (08/12/2011 20:11:20).


DDC file /home/oracle/orcl/orcl.ddc loaded.
Try "help"
dbvrep>
```

When Dbvisit Replicate is first configured, it needs to replicate its own internal tables and this is known as the initialization process. To monitor the progress of initialization, the `LIST PROGRESS ALL` command can be used.

```
dbvrep> list progress all
Progress of replication: total/this execution
------------------------------------------------------------------------------------
DBVREP.DBRSCOL$/dbvrep.DBRSCOL$: 59%  Mine:109211/109211    Applied:65363/65363
DBVREP.DBRSOBJ$:              ---%  Mine:78355/783          Applied:0/0
DBVREP.DBRSTAB$:              ---%  Mine:3516/3516          Applied:0/0
DBVREP.DBRSUSER$:             ---%  Mine:109/109            Applied:0/0
DBVREP.DBRSV_$DATABASE:       ---%  Mine:1/1                Applied:0/0
------------------------------------------------------------------------------------
5 tables listed.
```

(not all columns are displayed above)

Replication cannot be started until the initialization has been completed. This is completed when the console show MINE and APPLY is running:

```
/MINE IS running. Currently at plog 269 and SCN 6365268 (08/12/2011 21:01:09).
APPLY IS running. Currently at plog 269 and SCN 6365270 (08/12/2011 21:01:10).
```

Insert records into the scott.avi_objects table. A script can be created to do this:

```
sqlplus  scott/tiger@ttorcl<<SQL
insert into scott.avi_objects select * from dba_objects where rownum <= 2000;
commit;
exit
SQL
```

On the Dbvisit Replicate console the replication can be monitored:

```
Progress of replication: total/this execution
------------------------------------------------------------------------------------
SCOTT.AVI_OBJECTS/scott.avi_objects:100%  Mine:2000/2000         Applied:2000/2000
------------------------------------------------------------------------------------
1 tables listed.
 dbvrep>
```

(not all columns are displayed above)

If the replication shows 100%, then it is fully replicated. If the progress is not shown on the command console then `LIST PROGRESS` can be used to display the progress.

The insert script can be repeatedly run to insert more records which are then automatically replicated to the MySQL database.

The Mine and Apply processes are run in the background and can be viewed with `ps`

```
$ ps -ef | grep dbvrep
oracle    4098     1  6 14:27 ?        00:01:17 dbvrep APPLY or -daemon --ddcfile
/home/oracle/orcl/orcl-APPLY.ddc start APPLY
oracle    4107     1  6 14:27 ?        00:01:16 dbvrep MINE orc -daemon --ddcfile
/home/oracle/orcl/orcl-MINE.ddc start MINE
```

MS SQL Server replication

In all replications, the source database must be an Oracle database. Therefore, two-way replication with MS SQL Server is not supported.

ODBC connection is used to connect to MS SQL, thus a DSN (Data Source Name) must be set up by the system administrator first (e.g. by the ODBC Data Source Administrator tool included in MS Windows).

Currently, no DDL replication is supported for MS SQL.

Example DDC file settings

DDC file 1-way replication example:

The following DDC file settings shows an example of a 1-way replication process.

The source server is on Linux server:	rac11202node1
The target is on Linux server:	dbvisit400
The TNS names for the source database is:	orcl
The TNS names for the target database is:	test11202

There is no fetcher process.

DDC DB

The DDC DB is as follows (orcl-onetime.ddc):

```
################################################################################
# This file is used only during the setup, then it's contents is loaded as DDC DB
into mine database
################################################################################
```

```
@/home/oracle/orcl/orcl-MINE.ddc
set DDC_NAME orcl
#please update SETUP_SCRIPT_PATH if you move the setup scripts somewhere else - this
will help SUPPORT command to find them when creating Dbvisit Support packages
SET MINE.DDC_BACKUP_DIR /home/oracle/orcl/ddc_backup
SET MINE.FETCHER_ENABLED NO
SET MINE.LOG_FILE /home/oracle/orcl/log/dbvrep_%N_%D.%E
SET MINE.LOG_FILE_TRACE /home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E
SET MINE.MINE_ASM AUTO
SET MINE.MINE_DATABASE orcl
SET MINE.MINE_LISTEN_INTERFACE rac11202node1:7901
SET MINE.MINE_PASSWORD
53616c7465645f5f092f0f33aeb4be0371675629612d7ce61e4c4b006a1089e9
SET MINE.MINE_PLOG /home/oracle/orcl/mine/%S.%E
SET MINE.MINE_REMOTE_INTERFACE rac11202node1:7901
SET MINE.MINE_USER dbvrep
SET MINE.ORACLE_HOME /u01/app/11.2.0/grid
SET MINE.SETUP_SCRIPT_PATH /home/oracle/orcl
SET APPLY.APPLY_DATABASE test11202
SET APPLY.APPLY_LISTEN_INTERFACE dbvisit400:7902
SET APPLY.APPLY_PASSWORD
53616c7465645f5f3fa67e0ac06f36a653f0905a9e0b2f18c275c4a5e8ace4da
SET APPLY.APPLY_RDBMS Oracle
SET APPLY.APPLY_REMOTE_INTERFACE dbvisit400:7902
SET APPLY.APPLY_STAGING_DIR /home/oracle/orcl/apply
SET APPLY.APPLY_USER dbvrep
SET APPLY.DDC_BACKUP_DIR /home/oracle/orcl/ddc_backup
SET APPLY.LOG_FILE /home/oracle/orcl/log/dbvrep_%N_%D.%E
SET APPLY.LOG_FILE_TRACE /home/oracle/orcl/log/trace/dbvrep_%N_%D_%I_%U.%E
SET APPLY.SETUP_SCRIPT_PATH /home/oracle/orcl
set NETWORK_TRAFFIC_KEY 951f6ca180f83de089e4dd6935d6c58f
set STATUS_BAR STATUS+LIST
```

orcl-MINE.ddc

```
memory_set CHECKVARS ON
memory_set ON_WARNING SKIP
memory_set ON_ERROR EXIT
memory_set DDC_ID 1
memory_set DDC_DATABASE orcl
memory_set DDC_PASSWORD
53616c7465645f5f846088cc178d8bb44f9f0cb27201747399b628865de6c084
memory_set DDC_USER dbvrep
memory_set TNS_ADMIN /u01/app/oracle/product/11.2.0/dbhome_1/network/admin
#load rest of the settings from database
load ddcdb
memory_set ON_WARNING SKIP
memory_set ON_ERROR SKIP
```

orcl-APPLY.ddc

```
The apply DDC file is as follows:
memory_set CHECKVARS ON
memory_set ON_WARNING SKIP
memory_set ON_ERROR EXIT
memory_set DDC_ID 1
memory_set DDC_DATABASE orcl
memory_set DDC_PASSWORD
53616c7465645f5f846088cc178d8bb44f9f0cb27201747399b628865de6c084
memory_set DDC_USER dbvrep
memory_set TNS_ADMIN /u01/app/oracle/product/11.2.0/dbhome_1/network/admin
#load rest of the settings from database
load ddcdb
memory_set ON_WARNING SKIP
```

```
memory_set ON_ERROR SKIP
```

DDC file 2-way replication example:

The following DDC file settings shows an example of a 2-way replication process.

The source/target database is on Linux server: `dbvisit51`
The target/source database is on Windows server: `win2008-01`
The TNS names for the source/target database is: `dbvisit51_w112a`
The TNS names for the target/target database is: `win_w112b`
There is no fetcher process.

```
 set CHECKVARS ON
set ON_WARNING SKIP
set ON_ERROR EXIT
set DDC_ID 1
set DDC_NAME w112a2
set SIMPLE_CONFIG NO
set FETCHER.FETCHER_ENABLED NO
set MINE.FETCHER_ENABLED NO
set MINE.MINE_LISTEN_INTERFACE dbvisit51:7892
set MINE.MINE_REMOTE_INTERFACE dbvisit51:7892
set APPLY.MINE_REMOTE_INTERFACE dbvisit51:7892
set APPLY.APPLY_LISTEN_INTERFACE win2008-01:7893
set MINE.APPLY_REMOTE_INTERFACE win2008-01:7893
set APPLY.APPLY_REMOTE_INTERFACE win2008-01:7893
set MINE.MINE_DATABASE dbvisit51_w112a
set MINE.MINE_USER dbvrep2
set MINE.MINE_PASSWORD 53616c7465645f5f28844935a7ca6429892664945779edf04bccc10797
set MINE.MINE_ASM +ASM
set APPLY.APPLY_DATABASE win_w112b
set APPLY.APPLY_USER dbvrep2
set APPLY.APPLY_PASSWORD 53616c7465645f5fe821fbba607902c8077910024a2b2e22d390386ced
set MINE.MINE_PLOG /home/oracle/dbvrep/mine/%S.%E
set APPLY.APPLY_STAGING_DIR C:\app\oracle\dbvrep
set DEBUG_LEVEL 0
set MINE.LOG_FILE dbvrep_%P_%D.%E
set APPLY.LOG_FILE dbvrep_%P_%D.%E
set APPLY.APPLY_TRACE OFF
set MINE.MINE_TRACE OFF
#2nd direction
set FETCHER2.FETCHER_ENABLED NO
set MINE2.FETCHER_ENABLED NO
set MINE2.MINE_LISTEN_INTERFACE win2008-01:7894
set MINE2.MINE_REMOTE_INTERFACE win2008-01:7894
set APPLY2.MINE_REMOTE_INTERFACE win2008-01:7894
set APPLY2.APPLY_LISTEN_INTERFACE dbvisit51:7895
set MINE2.APPLY_REMOTE_INTERFACE dbvisit51:7895
set APPLY2.APPLY_REMOTE_INTERFACE dbvisit51:7895
set MINE2.MINE_DATABASE win_w112b
set MINE2.MINE_USER dbvrep2
set MINE2.MINE_PASSWORD 53616c7465645f5fe821fbba607902c8077910024a2b2e22d390386ced9
set MINE2.MINE_ASM +ASM
set APPLY2.APPLY_DATABASE dbvisit51_w112a
set APPLY2.APPLY_USER dbvrep2
set APPLY2.APPLY_PASSWORD 53616c7465645f5f28844935a7ca6429892664945779edf04bccc10797
```

```
set MINE2.MINE_PLOG C:\app\oracle\dbvrep\mine\%S.%E
set APPLY2.APPLY_STAGING_DIR /home/oracle/dbvrep/apply
set DEBUG_LEVEL 0
set MINE2.LOG_FILE dbvrep_%P_%D.%E
set APPLY2.LOG_FILE dbvrep_%P_%D.%E
set APPLY2.APPLY_TRACE OFF
set MINE2.MINE_TRACE OFF

set FETCHER.MINE_PEER=MINE
set APPLY.MINE_PEER=MINE
set MINE.FETCHER_PEER=FETCHER
set APPLY.FETCHER_PEER=FETCHER
set MINE.APPLY_PEER=APPLY
set FETCHER.APPLY_PEER=APPLY

set FETCHER2.MINE_PEER=MINE2
set APPLY2.MINE_PEER=MINE2
set MINE2.FETCHER_PEER=FETCHER2
set APPLY2.FETCHER_PEER=FETCHER2
set MINE2.APPLY_PEER=APPLY2
set FETCHER2.APPLY_PEER=APPLY2

set FETCHER.PROCESS_TYPE=FETCHER
set MINE.PROCESS_TYPE=MINE
set APPLY.PROCESS_TYPE=APPLY
set FETCHER2.PROCESS_TYPE=FETCHER
set MINE2.PROCESS_TYPE=MINE
set APPLY2.PROCESS_TYPE=APPLY

choose process fetcher
choose process mine
choose process apply
set STATUS_BAR ON
set ON_WARNING SKIP
set ON_ERROR SKIP
```

Note the use of the following processes to enable 2-way replication:

- MINE – default mine process on the first server.
- MINE2 – Second mine process to mine on the second server.
- APPLY – default apply process to apply on the second server.
- APPLY2 – Second apply process to apply on the first server.

Dbvisit Replicate Fetcher process

The mine processes imposes a certain load on the source database when parsing the redo logs (CPU and memory). This maybe around 5% and may not be desirable in certain environments.

This is when the Fetcher process is very useful. It runs on the source database, reading the redo logs and sending them to mine process, which is running on a different machine.

The only operations the fetcher process does are:

- queries the source database for the location of the redo logs.
- repeatedly queries the database to find out when the current log is switched.
- in case of ASM files, connects to ASM instance and reads the logs using internal PL/SQL API.
- in case of filesystem files, reads the files from disk using standard system I/O.
- connects to mine over the network and transfers the redo log files to the mine process.
- listens for user commands.

All of the resource-consuming work of parsing the redo logs, writing the plogs and sending them to apply is done by the mine process.

The fetcher process is optional and is not needed when the mine runs on the primary server. In that case, the database querying and file reading is done directly by mine.

Remote apply replication

The apply process does not need to run directly on the server running the target database, as it uses network connection[4] to that database.

However, it is advised to run it on the target server, as the network communication overhead decreases the performance of the apply.

Still, it can be useful:

- if the target server has no capacity left to accommodate the apply process.
- if the target server is not running on a supported platform.

There is no special configuration to enable this – just use the apply-running-server when configuring listen/remote interfaces in the wizard, and set up database connectivity there so apply can reach the database.

Triggers

Triggers on target tables of any replication deserve special consideration, and Dbvisit Replicate is no exception. The concern is whether triggers defined on the target tables should fire when data changes are applied to the table. This depends on the logic in the trigger and thus has to be decided by the user.

In general, triggers carried over from source database are not intended to fire, but triggers created on apply should fire. But other criteria can be considered as well, e.g. whether other tables referenced in the trigger are replicated or not.

Dbvisit Replicate support

For Oracle 10.2.0.5 and 11.2.0.2 and later:

- Dbvisit Replicate honors setting of dbms_ddl.set_trigger_firing_property[5] – use this procedure for each trigger to set whether it should fire when data change is done by replication.

On every Oracle target, you can check boolean variable DBRSAPPLY_PKG.is_dbreplicate_session to determine whether it's an apply session or not. Using this variable, the trigger can check whether to actually do something or not. This will require code changes in the triggers to enable this.

On every MySQL target, you can check variable @is_dbreplicate_session (valued 0/1) to find out whether it's an apply session or not.

On every MS SQL target, you can check context_info to find out whether it's an apply session (set to 1) or not.

Variable `APPLY_SET_TRIGGER_FIRE_ONCE` can be used to disable or further control this functionality.

Oracle Client and configuration files

For ASM, Dbvisit Replicate uses bequeath connection and thus has to use the ASM home for as the client, setting ORACLE_HOME accordingly (setup wizards thus asks for this ORACLE_HOME when ASM is in use).

For all other Oracle connections, Dbvisit Replicate uses SQL*Net connections. As it comes with it's own Oracle client, it has to be told where to look for sqlnet.ora/tnsnames.ora files. Setup wizard thus asks for TNS_ADMIN.

[4]SQL*Net for Oracle, client-server for MySQL, ODBC for MS MSQL.

[5]According to Oracle documentation, the default is TRUE, i.e. fire only once.

In order of precedence:

- The DDC setting TNS_ADMIN can be used to select the correct configuration files.
- TNS_ADMIN can be also specified in the environment.
- The DDC setting ORACLE_HOME also selects the correct configuration files; note that this is set if ASM is in use.
- ORACLE_HOME can be also specified in the environment.
- Current directory is searched last.

Note that these are standard Oracle precedence rules, with the only addition that DDC file can override variable specified in the environment.

Oracle home detection

Setup wizard tries to detect existing Oracle homes to ease the installation. This is not foolproof, though, and has some limits:

- On 64-bit Windows, it searches only installation of same 32-/64-bit binaries as the Dbvisit Replicate executable is, because ORACLE_HOME has to be set to use same binaries type. However, you can set TNS_ADMIN to point to wrong binaries type, as this is used for text-file network configuration only.
- On Linux and Solaris, oratab is parsed to detect Oracle homes. This usually means it won't find client installations etc., where no database is configured.
- TNS_ADMIN detection works the same, it just checks detected ORACLE_HOMEs and checks for existence of sqlnet.ora/tnsnames.ora in network/admin subdirectory or directly in ORACLE_HOME.
- As with all checks in setup wizard, this detection works on local machine only. Check that the settings are valid for other nodes as well.

Conflicts

The replication is a row-based one; that is, the changes are mined row-by-row on the mine. Thus while any SQL issued on the source database can change arbitrary number of rows, Dbvisit does not care about the actual SQL issued, only about the changes made to individual rows.

The major consequence is that the SQL issued at the apply database is not the same issued against mine – instead, each SQL updates/deletes/inserts exactly one row, applying just one change. Thus if the SQL actually affects zero or more than one rows, data divergence occurred.

Another type of conflict is any error reported by Oracle – this can range from usual primary key or foreign key violation (another type of data divergence) to purely technical reasons (cannot extend datafile).

The last type of conflict is lock timeout – if the apply waits for a row lock more than WATCHDOG_TIMEOUT seconds, a conflict is also reported.

Note that when applying a row-change, previous values of the row at apply are checked (by means of adding them to the where clause). This effectively detects if the data in that row are inconsistent between mine and apply, and this is reported as a conflict, “0 rows updated/deleted.” This can happen for update or delete only (there is no previous row to check for insert) and due to its special role, it can be handled differently.

Configuring conflict handling

The response to a conflict can be configured before the conflict occurs. The configuration can be specified for each replicated table and for each operation type separately, and for the special case (“data” divergence) mentioned above.

The options are as follows:

Operations	“Data” handler	“Error” handler
Update	`discard|retry|overwrite|pause|abort|newer|older|sql|`	`discard|retry|overwrite|pause|abort|plsql|error`

	plsql|error	
Delete	discard|retry|overwrite| pause|abort|newer|older|sql| plsql|error	discard|retry|overwrite| pause|abort|plsql|error
Insert	n/a	discard|retry|overwrite| pause|abort|plsql|error
Transaction	n/a	discard|retry|overwrite| pause|abort|plsql|error

The transaction handler is used for DDL, commits etc.

The “Data” handler is used only once for a conflict – if using this handler leads to a conflict again, “Error” handler is used for the next attempt. For the next change SQL, the Data conflict will be used again.

Additionally, logging may be specified:

- log: the conflict is logged, and a error plog is created with the offending statement
- nolog: the conflict is not logged
- fast_nolog: the conflict is not logged and it may be omitted from conflict counters altogether
- log_transaction: the conflict is logged, and a error plog is created with the offending transaction
- default: same as log

See the command chapter or use HELP SET_CONFLICT_HANDLERS for the exact syntax.

Use SHOW_CONFLICT_HANDLERS to query current settings of handlers.

Available handlers

- `DISCARD`: ignore the offending SQL and continue with the next one
- `OVERWRITE`: do not check old values, try again with just primary key in the where clause (thus this will fail if there is no row at all on apply with the PK value)
- `NEWER,OLDER`: look into target table (by primary key) and get values of specified columns (usually dates or number sequence). If the source row is newer/older, the operation becomes `OVERWRITE`, otherwise `DISCARD`.
- `PLSQL`: call user-specied PL/SQL function. The function must have prototype:

  f(apply_old_data *table*%rowtype, has_found_apply_row boolean, primary_key_data *table*%rowtype, new_data *table*%rowtype) return number;

  The return values are:

  - 0: discard
  - 1: overwrite
  - 2: retry
  - 3: the PL/SQL function resolved the conflict by itself and made all necessary changes.

  The function must not issue a commit, as the transaction may be yet rolled back, if a rollback happened on source.
- `RETRY`: wait a few seconds (set by variable RETRY_TIME) and try again
- `PAUSE`: wait for manual user resolution
- `ABORT`: kill apply process
- `ERROR`: rollback the transaction, continue applying other transactions

Handling current conflict

If the apply was paused due to a conflict, or it is retrying in a loop the same SQL again and again, you can instruct it about what to do next using the command:

RESOLVE CONFLICT id AS resolution

The conflict id is the number shown in the status bar, the resolution can be:

- `IGNORE`: skip the transaction
- `RETRY`: try again
- `ABORT`: abort the apply process
- `RESTART`: rollback and restart the transaction
- `ROLLBACK`: rollbacks the transaction

Handling errors on source database, partially executed statements

The mine (and then apply as well) process follow the changes as they are written to redo logs by Oracle. This also defines it's (sometime peculiar) behavior in case of errors, statement rollbacks, statement restarts etc. In general:

- Oracle handles these cases in transaction-consistent way, and so do mine and apply. Thus after a commit (and thus to any other session as well), the data is in sync with source.
- However, Oracle writes to redo in an optimistic way – and thus in case of an partial execution followed a statement or partial rollback of a statement, the data is actually written to the redo logs and thus replicated as well.
- This means, in consistence with usual behavior on source, triggers can execute on data that are later rolled back. See for example discussion on one way how can this happen at *write "consistency"* at http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:11504247549852
- One interesting case of this behavior is (primary, unique) key checking – Oracle first writes the row data, then goes on to update the index, failing on key violation, then rolls back the row change. The consequence is that apply also updates the row and if the key is present on apply database as well, it will also fail, causing a conflict.

Notifications and remote management

To facilitate easier management, three features are implemented:

- email notifications: the processes send both alerts and periodic summaries
- SNMP traps: the processes send SNMP messages to SNMP management software (HP OpenView, Nagios, or any other SNMP-compatible software package). The alerts are the same as email notification alerts.
- SNMP subagent: the processes can act as an AgentX subagent to an already installed SNMP agent supporting the AgentX protocol (for example, the open source net-snmp agent) and thus provide current information to the SNMP management software, showing both current status and receiving stop/pause/resume requests (not available on Windows).

The MIB file for the SNMP management software describing structure of the SNMP information provided is included in the Dbvisit Replicate distribution.

General SMTP configuration

The MAILCFG_* variables configure usual SMTP protocol settings: SMTP server hostname, port, use of SSL, username/password if required by the SMTP server, and the From: address.

General mail configuration

Email recipients are configured by:

- `NOTIFY_SUCCESS_EMAIL` – get heartbeat / progress emails

- NOTIFY_ALERT_EMAIL – get all other emails
- NOTIFY_ALL_EMAIL – get all emails

Specify OFF to disable a particular group.

General SNMP trap configuration

Only two variables are required: SNMP management software server name/IP (optionally specifying non-default port) and the community string (“password”).

General SNMP subagent configuration

Only one variable is used: SNMP_INDEX. The data about the processes are presented as tables (one for mine processes, one for fetchers, one for appliers), with one row for each process running on the machine. The row numbers in the tables must be static and set by the user using the SNMP_INDEX variable. (Were the numbers dynamic, the management software would mix up progress graphs/history among multiple processes.) For the customary one-replication configuration, the recommended setting is SNMP_INDEX=1.

A value of 0 disables this feature.

SNMP Avisit-MIB DEFINITIONS

Please see the supplied Dbvisit-MIB-SNMP.txt file for the MIB definitions for SNMP definitions.

Notification configuration

The “all-ok” emails (heartbeat and list progress) are configured by specifying list of times in 24-hour format, when the emails should be sent, e.g.:

```
NOTIFY_DAILY_LIST_PROGRESS_TIME24 = 0700
NOTIFY_SEND_HEARTBEAT_TIME24 = 0800:1300
```

The checking for alerts occur every interval specified by NOTIFY_INTERVAL_BETWEEN_CHECK (e.g. 5m, or 1h25m30s)[6]. If an error condition persists for NOTIFY_EXCEEDED_CYCLE_NUM checks in a row, the email/SNMP trap is sent. Set this to more than 1 to “smooth out” short-time bursts, temporary network issues etc.

The alerts check:

- plog sequence difference (lag) between apply and mine
- redolog sequence difference between mine and fetcher
- processes unreachable by their peers (this alert is not fired if all processes are down, as there is then no process to run the check)
- SCN difference between apply and mine
- number of conflicts at apply (for each table)
- % difference of applied changes (for each table). The very percentages from LIST PROGRESS are checked.

[6]This is also the interval when the process checks whether an “all-ok” email should be sent. Thus those might lag a bit behind the specified times.

Starting Dbvisit Replication

Once Dbvisit Replicate has been configured, the replication can be started.

Note: This section is for informational purposes only as the Dbvisit Replicate setup wizard creates the necessary scripts for starting the replication.

Start the Dbvisit Replicate Command Console with the correct DDC file:

On Windows:

```
Start > Dbvisit Replicate Command Console
dbvrep> readddc w112a.ddc
```

OR Start a Windows command prompt and type:

```
dbvrep.exe --ddcfile w112a.ddc
```

On Linux/Unix:

```
dbvrep --ddcfile w112a.ddc
```

(Where w112a.ddc is an example name of a DDC file)

Once the Dbvisit Replicate Command Console has been started the mine, fetcher (optional) and apply processes can be started.

Starting the mine process

On Windows:

On Windows, the Dbvisit Replicate can be started as a local service.

```
dbvrep> start_service mine
```

This starts the Windows service. If the service is not yet created, the service can be created with:

```
dbvrep> create service mine
```

On Linux/Unix:

```
dbvrep --ddcfile w112a.ddc --daemon start mine
```

The –daemon option makes dbvrep disconnect from the console and run in the background. No further commands can be entered. This option is not available on Windows, as similar functionality will be provided by support of Windows services.

To see the progress of the replication, start a new Dbvisit Replicate Command Console and load the same DDC file.

Starting the apply process

On Windows:

On Windows, the Dbvisit Replicate can be started as a local service.

```
dbvrep> start_service  apply
```

This starts the Windows service. If the service is not yet created, the service can be created with:

```
dbvrep> create service apply
```

On Linux/Unix:

```
dbvrep --ddcfile w112a.ddc --daemon start apply
```

The –daemon option makes dbvrep disconnect from the console and run in the background. No further commands can be entered.

To see the progress of the replication, start a new Dbvisit Replicate Command Console and load the same DDC file. See Viewing the status of the replication for more information.

Stopping the replication process

To stop or shutdown the replication process:

```
dbvrep> shutdown all
```

This will stop all the replication processes including mine, apply and fetcher (if started).

Viewing the status of the replication

Once the replication process has started the Dbvisit Replicate Command Console can be used to monitor the replication and to find the current status.

Start the Dbvisit Replication Command console with the correct DDC file:

```
dbvrep --ddcfile w112a.ddc
```

The Dbvisit Replication Command console shows the running processes:

```
-MINE IS running. Currently at redo log sequence 229 and SCN 4794116.
APPLY IS running. Currently at redo log sequence 229 and SCN 4794115 (17/02/2011
14:34:56).
Dbvisit Replicate
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file w112a.ddc loaded.


Try "help"
dbvrep>
```

(Where w112a.ddc is an example name of a DDC file)

The top part of the Dbvisit Replicate Command Console displays the current status of all the replication processes in real time and displays the current SCN (Oracle system change number) of both the source and target databases.

Showing the replication progress

The replication progress can be shown in the Dbvisit Replicated Command Console:

```
dbvrep>list progress
Progress of replication: total/this execution
------------------------------------------------------------------------------------
SCOTT.AVI_OBJECTS: 100% Mine:20/20 Unrecov:0/0 Applied:20/20 Conflicts:0/0
------------------------------------------------------------------------------------
1 tables listed.
```

The list progress can also be shown in real time as part of the status bar by setting in the DDC file (this is on by default):

```
set STATUS_BAR status+list
```

The Dbvisit Replicate console has to be restarted for this setting to take affect.

The list progress tables displays the following information for each object that is replicated:

- Replication in %: 100% means that all data has been replicated.
- Mine: Counts the number of data rows that have been mined in total and for this execution.
- Unrecov: Counts the number of changes (blocks or block ranges) that cannot be replicated due to nologging (also known as unrecoverable) operations in total and for this execution.
- Applied: Counts the number of data rows that have been applied in total and for this execution.
- Conflict: Counts the number of data rows that are in conflict in total and for this execution.
- Last: The last time that replication of data occurred and status (OK or short error description, e.g. ORA error)

Creating and setting the starting point with Dbvisit Replicate

Before data replication can start, the target database must have a copy of the database objects that are going to be replicated. It is also highly desirable that the data in the objects to be replicated are also completely in sync. If the data is not in sync before replication starts, there is a strong possibility that data conflicts can occur. For instance if a record is updated or deleted on the source database that does not exist on the target database.

It is important that the replication starts at the same point as when the source and target data are in sync so that no changes are lost when replication starts. This is done through capturing the Oracle SCN at the point the table is “prepared” in Dbvisit Replicate.

Dbvisit Replicate can assist with creating and setting the starting points to ensure no changes are lost when the replication starts, even when it is not possible for an outage on the source database. Dbvisit Replicate does this by using Oracle Data Pump.

Only when the target database is Oracle will Dbvisit Replicate assist in creating the starting point. If the target database is not Oracle, then this will have to be done manually.

Setting the starting point

When Dbvisit Replicate prepares a table for replication it captures the SCN of the Oracle database to record the starting point. This is done through the `prepare` statement. The prepare statement does not start the replication, but initiates the following:

1. Marks the table on Mine to be mined.
2. Sets the conflict handlers.
3. Records the SCN (or sets the starting point - in some terminology this is called instantiation). When the actual replication begins (through the `start mine` command), it will start mining (or replicating) from this SCN for the table.

Creating the starting point

Dbvisit Replicate assists in creating the starting point by creating Oracle Data Pump scripts which can be run manually to export the data out of the source database and into the target database. The Oracle Data Pump scripts are created when Dbvisit Replicate runs the following dbvrep command:

```
ENGINE PREPARE_DP WRITE DP_NETWORKLINK DIRECTORY DATA_PUMP_DIR FILE...
```

This command is part of the *.dbvrep script that is created by the setup wizard and is run as part of the *-all.sh or *-all.bat script to configure the replication.

The dbvrep `ENGINE PREPARE_DP` command creates the Oracle Data Pump script using the following information:

1. The SCN starting point for each table to be replicated. The SCN is used to set the flashback_scn in Data Pump so that the data will be exported consistent with the starting point of the replication.
2. The SQL*Net TNS connections to connect to the source and target database using network import. The `network_link` has to be created manually in the target database. This can be done with command:

   ```
   CREATE public DATABASE LINK link_name CONNECT TO system IDENTIFIED BY password USING
   connect_string
   ```
3. The system username and password to connect to the source and target database.
4. Oracle Directory on the target database called DATA_PUMP_DIR. This directory is created by install or upgrade of Oracle.

No outage of the source database is required because the SCN is used to extract the data out of the source data. This ensures that the data will be consistent at that particular point in time. Dbvisit Replicate will start replicating at the same point in time and therefore consistency is maintained between the source and target database.

Data Pump Example:

The Oracle Data Pump script (APPLY.sh) generated by Dbvisit Replicate for table SCOTT.DEPT is as follows:

```
impdp SYS/xxxxx@d112f_dbvisit230 table_exists_action=TRUNCATE
network_link=d112f_dbvisit210 directory=DATA_PUMP_DIR flashback_scn=15249735
tables=SCOTT.DEPT
```

In this example the

- Source TNS name is: `d112f_dbvisit210`
- Target TNS name is: `d112f_dbvisit230`
- The flashback_scn is 15249735. This is the starting point for replication and all data before this point will be exported from the source database and into the target database.
- The network_link is: `d112f_dbvisit210`. This has to be created manually in the target database with the following command:

```
CREATE public DATABASE LINK d112f_dbvisit210 CONNECT TO system IDENTIFIED BY xxx
USING ' d112f_dbvisit210'
```

Complete starting point creation and replication start

The following example shows the complete starting point creation and replication start when there are no target objects in the target database.

The approach is:

1. Complete the setup wizard and select the tables or schemas to be replicated. In this example a table (scott.avi_objects) and a schema (avi) will be replicated. Both the table and the schema do not exist on the target database.
2. Run the *-all.sh script. This script will create the replication environment and also create the Oracle Data Pump script for the objects to be replicated.
3. Start the Mine process to start the replication on the source database.
4. Start the Oracle Data Pump script to create the objects in the target database and load them with initial data.
5. Once the Oracle Data Pump load has been completed, the Apply process can be started on the target database.

Notes

1. The order of the step 3 (starting Mine) or 4 (Data Pump) above is not important. Step 4 can be done before step 3. It is however important that step 5 (starting Apply) be done at the completion of step 4.
2. By default Dbvisit Replicate expects the replicated objects to exists on the target database when the `prepare` command is run. This may result in the following error:

   ```
   dbvrep> PREPARE OFFLINE SCHEMA AVI
   ERR-9083: Table AVI.AVI_TEST2 not found at apply.
   FATAL-9043: Error detected and ON_ERROR set to EXIT.
   Error encountered, not starting Dbvisit Replicate.
   ```

   In this case edit the *.dbvrep script and set:

   ```
   set ON_ERROR ERROR
   ```

   to

   ```
   set ON_ERROR SKIP
   ```

   After setting the ON_ERROR to skip, run the *-all script again.
3. The object grants on the target database may also error during Dbvisit Replicate setup:

   ```
   grant select, update, insert, delete on SCOTT.AVI_OBJECTS to dbvrep
                                                          *
   ERROR at line 1:
   ORA-00942: table or view does not exist
   ```

This can be ignored as the Oracle Data Pump will create the necessary object grants as long as the same dbvrep user is used in both source and target databases.

Example

1. Start the setup wizard and setup Dbvisit Replicate. In this example the replication is called d112f, the source and target databases are both called d112f, the source server is dbvisit210 and the target server is dbvisit230.
2. Run the d112f-all.sh script created by the setup wizard on the source server. Note the error and warning messages because the objects do not appear in the target database.

```
$ ./d112f-all.sh
Setting up Dbvisit Replicate configuration
Configure database d112f_dbvisit210...
Configure database d112f_dbvisit230...
Object grants for database d112f_dbvisit210...
Object grants for database d112f_dbvisit230...
grant select, update, insert, delete on SCOTT.AVI_OBJECTS to dbvrep
                                                       *
ERROR at line 1:
ORA-00942: table or view does not exist
Setting up the configuration
Initializing.....done
WARN-1850: No DDC DB available, dictionary table does not exist.
DDC loaded from database (0 variables).
Dbvisit Replicate version 2.2.05.1073
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/d112f/d112f-onetime.ddc loaded.
MINE: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
APPLY: Cannot determine Dbvisit Replicate dictionary version. (no dictionary exists)
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR EXIT
Variable ON_ERROR set to EXIT for process *.
dbvrep> ENGINE SETUP MINE DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP MINE CREATE DICTIONARY
dbvrep> ENGINE SETUP MINE LOAD DICTIONARY
Supplemental logging on database set.
Loading dictionary table DBRSCOL$
Loading dictionary table DBRSOBJ$
Loading dictionary table DBRSTAB$
Loading dictionary table DBRSUSER$
Loading dictionary table DBRSV_$DATABASE
dbvrep> ENGINE SETUP APPLY DROP DICTIONARY
0 dictionary objects dropped.
dbvrep> ENGINE SETUP APPLY CREATE DICTIONARY
dbvrep> ENGINE SETUP APPLY LOAD DICTIONARY
dbvrep> ENGINE PREPARE_DP SETUP CLEAR
dbvrep> ENGINE SETUP PAIR MINE AND APPLY
ID of mine proces is 98796642-23B7-11E1-BC5C-1BD78DA50EF3. If not using DDC in
database, set MINE_UNIQUE_ID to this value.
1 applier SCN set.
dbvrep> PREPARE OFFLINE SCHEMA AVI
WARN-9083: Table AVI.AVI_TEST2 not found at apply.
WARN-1645: No columns found for AVI.AVI_TEST2, check privileges for Dbvisit
Replicate dictionary user at apply or create the
target table if it does not exist - before you start the apply.
WARN-9246: Column AVI.AVI_TEST2.DATESTAMP is not in table AVI.AVI_TEST2 on apply.
WARN-9246: Column AVI.AVI_TEST2.ID is not in table AVI.AVI_TEST2 on apply.
WARN-9246: Column AVI.AVI_TEST2.TEST2 is not in table AVI.AVI_TEST2 on apply.
```

```
Table AVI.AVI_TEST2 instantiated at SCN 15652921
dbvrep> PREPARE OFFLINE TABLE SCOTT.AVI_OBJECTS
WARN-9083: Table SCOTT.AVI_OBJECTS not found at apply.
WARN-1645: No columns found for SCOTT.AVI_OBJECTS, check privileges for Dbvisit
Replicate dictionary user at apply or create
the target table if it does not exist - before you start the apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.CREATED is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.DATA_OBJECT_ID is not in table SCOTT.AVI_OBJECTS
on apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.EDITION_NAME is not in table SCOTT.AVI_OBJECTS
on apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.GENERATED is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.LAST_DDL_TIME is not in table SCOTT.AVI_OBJECTS
on apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.NAMESPACE is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.OBJECT_ID is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.OBJECT_NAME is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.OBJECT_TYPE is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.OWNER is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.SECONDARY is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.STATUS is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.SUBOBJECT_NAME is not in table SCOTT.AVI_OBJECTS
on apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.TEMPORARY is not in table SCOTT.AVI_OBJECTS on
apply.
WARN-9246: Column SCOTT.AVI_OBJECTS.TIMESTAMP is not in table SCOTT.AVI_OBJECTS on
apply.
Table SCOTT.AVI_OBJECTS instantiated at SCN 15652952dbvrep> ENGINE PREPARE_DP WRITE
DP_NETWORKLINK DIRECTORY DATA_PUMP_DIR FILE /home/oracle/d112f/APPLY.sh DBLINK
d112f_dbvisit210
USERID SYSTEM/oracle@d112f_dbvisit230
Created Data Pump script /home/oracle/d112f/APPLY.sh, using network import.
dbvrep> create ddcdb from ddcfile
DDC loaded into database (156 variables).
dbvrep> set ON_WARNING SKIP
Variable ON_WARNING set to SKIP for process *.
dbvrep> set ON_ERROR SKIP
Variable ON_ERROR set to SKIP for process *.
OK-0: Completed successfully.
1) Create the necessary directory(ies) on the servers:
dbvisit210: /home/oracle/d112f
dbvisit230: /home/oracle/d112f

2) Copy the DDC files to the server(s) where the processes will run:
/home/oracle/d112f/d112f-APPLY.ddc
/home/oracle/d112f/d112f-MINE.ddc

3) Review that path to dbvrep executable is correct in the run scripts:
/home/oracle/d112f/d112f-run-dbvisit210.sh
/home/oracle/d112f/d112f-run-dbvisit230.sh

4) Copy the run script to the server(s) where the processes will run:
/home/oracle/d112f/d112f-run-dbvisit210.sh
/home/oracle/d112f/d112f-run-dbvisit230.sh
```

```
5) Ensure firewall is open for listen interfaces dbvisit230:7902, dbvisit210:7901
used by the processes.

6) Start the replication processes on all servers:
/home/oracle/d112f/d112f-run-dbvisit210.sh
/home/oracle/d112f/d112f-run-dbvisit230.sh

7) Start the console to monitor the progress:
/usr/bin/dbvrep --ddcfile /home/oracle/d112f/d112f-MINE.ddc
```

3. Start the Mine process by running the d112f-run-dbvisit210.sh on the source server.

```
$ ./d112f-run-dbvisit210.sh
Initializing.....done
DDC loaded from database (159 variables).
Dbvisit Replicate version 2.2.05.1073
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/d112f/d112f-MINE.ddc loaded.
Starting process MINE...started
```

4. Start the Oracle Data Pump script (ensure it has execution privileges).

```
$ chmod 755 APPLY.sh
$ ./APPLY.sh
Import: Release 11.2.0.1.0 - Production on Sun Dec 11 18:30:07 2011
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Release 11.2.0.1.0 - 64bit Production
Starting "SYSTEM"."SYS_IMPORT_TABLE_01":  SYSTEM/********@d112f_dbvisit230
table_exists_action=TRUNCATE network_link=d112f_dbvisit210 directory=DATA_PUMP_DIR
flashback_scn=15652921 tables=AVI.AVI_TEST2
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
. . imported "AVI"."AVI_TEST2"                                5 rows
Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_TABLE_01" successfully completed at 18:26:06
Import: Release 11.2.0.1.0 - Production on Sun Dec 11 18:30:17 2011
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Release 11.2.0.1.0 - 64bit Production
Starting "SYSTEM"."SYS_IMPORT_TABLE_01":  SYSTEM/********@d112f_dbvisit230
table_exists_action=TRUNCATE network_link=d112f_dbvisit210 directory=DATA_PUMP_DIR
flashback_scn=15652952 tables=SCOTT.AVI_OBJECTS
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 3 MB
Processing object type TABLE_EXPORT/TABLE/TABLE
. . imported "SCOTT"."AVI_OBJECTS"                          20000 rows
Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_TABLE_01" successfully completed at 18:26:13
```

The target objects have now been created and the initial data has been loaded. The data is in sync for the Apply process to start applying the replicated data.

5. Start the Apply process on the target server by running the d112f-run-dbvisit230.sh script.

```
$ d112f-run-dbvisit230.sh
Initializing.....done
DDC loaded from database (159 variables).
Dbvisit Replicate version 2.2.05.1073
```

```
Copyright (C) Dbvisit Software Limited.  All rights reserved.
DDC file /home/oracle/d112f/d112f-APPLY.ddc loaded.
Starting process APPLY...started
```

The replication on both servers has now been started and the data will be kept in sync.

Data divergence

Currently Dbvisit Replicate does not offer a tool to monitor or detect data divergence or to bring the data back in sync should data divergence occur.

However Dbvisit Replicate offers very robust and complete notification. This can be configured according to specific thresholds to ensure the replication does not fall behind too far where the data divergence becomes unmanageable.

In future Dbvisit Replicate may include a tool to detect and manage data divergence.

House keeping

Dbvisit Replicate offers automatic house keeping on the following files:

- Dbvisit Replicate plog files. These will get removed on source and target systems once they are no longer needed.
- Dbvisit Replicate log files. These are set managed to keep one backup file and a size limit of 100MB. This is configurable through the parameters:
  - LOG_FILE_SIZE
  - LOG_FILE_COUNT
  - LOG_FILE_DATE_ROTATE

Manual house keeping is required on the following files:

- Dbvisit Replicate trace files
- Dbvisit Replicate Cache files (see below)

Dbvisit Replicate Cache files

The Dbvisit Replicate executable (dbvrep) is a self contained executable and contains all the necessary required libraries for it to function. These libraries are extracted out of the executable and into the default temporary directory of the system at startup of dbvrep. Dbvisit Replicate will reuse these cache files the next time that it is started. This maybe noticeable in the startup time of dbvrep. If the cache already exists, the startup time is faster.

It maybe possible that there are several cache directories. This can happen if a new version of Dbvisit Replicate is installed. The old cache directories can be removed, however when dbvrep is running, it is difficult to determine which is the current cache and which is the old cache. If the cache needs to be removed, then is preferable to shutdown dbvrep and then remove all cache files and then to restart dbvrep.

The cache files are in the OS default temp directory under a directory called `par-username`.

Dbvisit Replicate Command Reference

Any of the commands described in the reference can be given directly on the command line or fed using standard input or script file:

- Interactively, at the prompt
- On the command line (dbvrep command)
- In file read by @/read command (dbvrep @file or @file at prompt)
- In file fed to standard inout (cat file | dbvrep)

Command-line only options

In addition, command line accepts these options:

- `--ddcfile` – read DDC file
- `--V` – see VERSION command
- `--pause` – will pause before exiting the script (useful to prevent window from closing)
- `--daemon` – become a background process
- `--no-checkdb` – do not connect to mine and apply databases on start and skip checks of repository version etc.
- `--netkey` – network authorization password (alternative way to set NETWORK_TRAFFIC_KEY variable, see chapter on network encryption)
- `--no-ddcdb` – disable the LOAD DDCDB command

Command-line reference

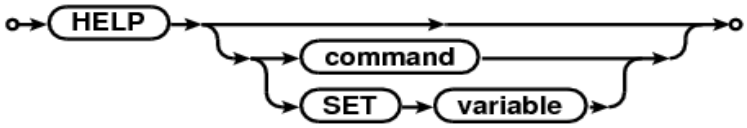
All keywords are case-insensitive.

If a command works directly with mine/apply/fetcher (not by process name), you have to choose your process by “choose” command. (This is not needed in simple configurations with only one mine/apply as this happens automatically).

The syntax diagrams in this document use a variation of Backus-Nauer Form (BNF), a convention familiar to any reader of Oracle documentation and many other documents. Emphasis and symbols have the following meaning in this version of BNF syntax.

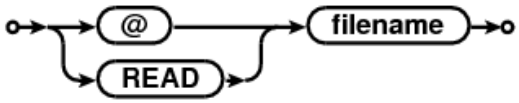
- Keywords are shown in `UPPERCASE`.
- Placeholders for which you must substitute an actual value are shown in lowercase. These can include clauses and other expressions.
- Vertical (|) bars separate multiple choices. They indicate "or".
- Square brackets ([]) are not typed. They indicate that the enclosed syntax is optional.
- Curly braces ({}) usually are not typed. They indicate that you must specify one of the enclosed choices. (The choices are separated by vertical bars.)
- Loops or repetitions are indicated by a second, bracketed appearance of the term, set of terms, or expression, followed by ellipsis points. The brackets indicate that the repetition is optional (all repetitions are optional). The ellipsis points indicate that multiple repetitions are allowed. The bracketed appearance of the term begins with a comma if the repetitions are comma delimited.
- All other punctuation (quotation marks, commas, semicolons, and so on) must be typed as shown.

HELP



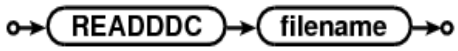
Shows general help, help for specific command or help for a variable.

READ



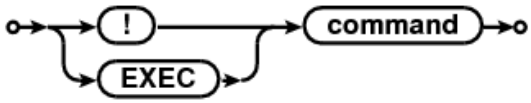
Reads the designated file like it would be entered at command prompt. As with SQL*Plus, both “@file” and “@ file” work.

READDDC



Read the specified DDC file. Same as command line option “--ddcfile file”, which is preferred, as READDDC is from interactive prompt sets some variables too late to be effective (e.g. STATUSBAR).

EXEC

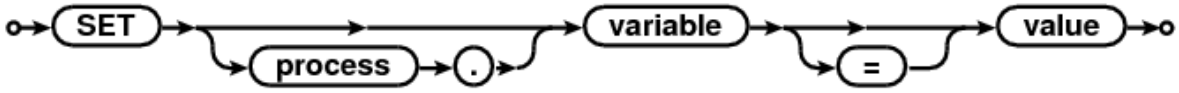


Execute command in a new shell.

HEALTHCHECK

Connects to mine and apply (and fetcher if configured). Checks that they see each other.

SET

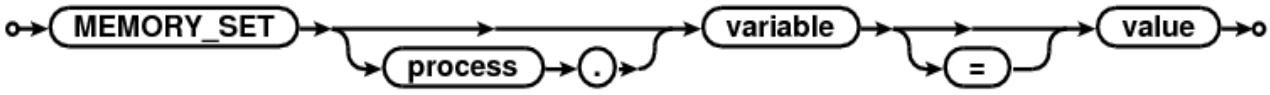


Sets configuration variable in memory and DDC DB.

RESET

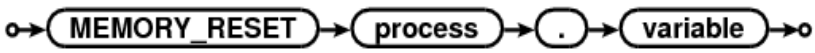
Unsets variable for a specified process, falling back to general non-process-specific setting.

MEMORY_SET



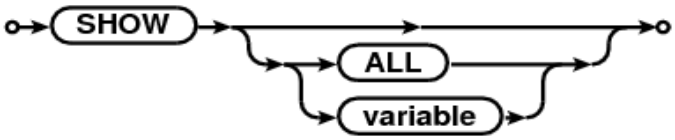
Sets configuration variable in memory only. Use for variables that cannot be set in the DDC DB or if you want the value to be valid for short time only.

MEMORY_RESET



Unsets variable, in memory only.

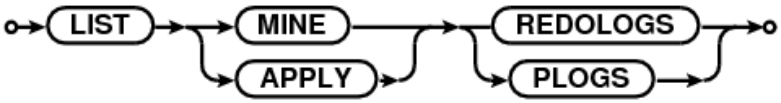
SHOW



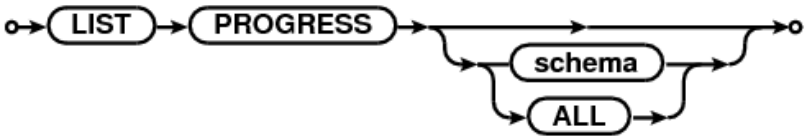
With no parameter or with ALL, shows all configuration variables, as they are seen by current process. It does not reread DDC file nor DDC DB to update the settings.

With variable, shows variable value. (For * and also for all processes.)

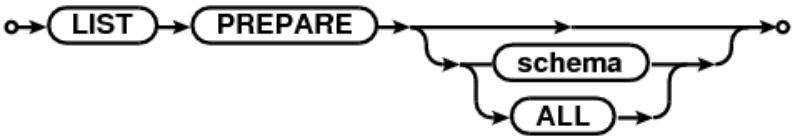
LIST



Connects to database and lists registered redologs / plogs.



Show tabular overview of mine/apply records processed and conflicts. You can optionally specify a schema to filter the results to a particular database schema (this is source schema if renaming is used). Use ALL to see everything including internal tables.



Show list of prepared tables. You can optionally specify a schema to filter the results to a particular database schema (this is source schema if renaming is used). Use ALL to see everything including internal tables.

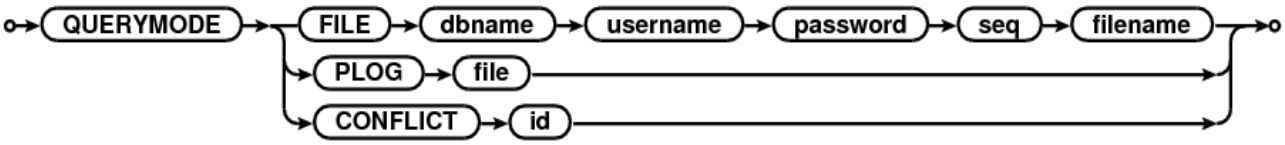
EXIT, QUIT

Exits dbvrep.

WAIT

Waits until killed.

QUERYMODE



FILE: Parses given online/archive redo, generates plog and parses the plog. (Note that to generate traces etc. during query mode, you still have to set the corresponding configuration variables.)

In query mode, incomplete plog will generate only an error, not a fatal error.

Note that this is a local API command, so you have to see the redo log on local filesystem.

No changes are actually applied to apply database.

PLOG: Parses selected plog (usually a conflict error plog) – either by specifying a filename, or by conflict id.

In query mode, incomplete plog will generate only an error, not a fatal error.

Note that this is a local API command, so you have to see the error-log on local filesystem, and have to be able to connect to apply database to query metadata if using conflict id.

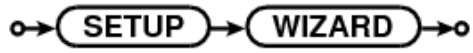
No changes are actually applied to apply database.

CONFLICT: If error plogs are enabled (disabled by default), tries to get error plog for given conflict id and run QUERYMODE PLOG on that file.

COMMENT

Ignored. Just a comment, useful in scripts.

SETUP WIZARD

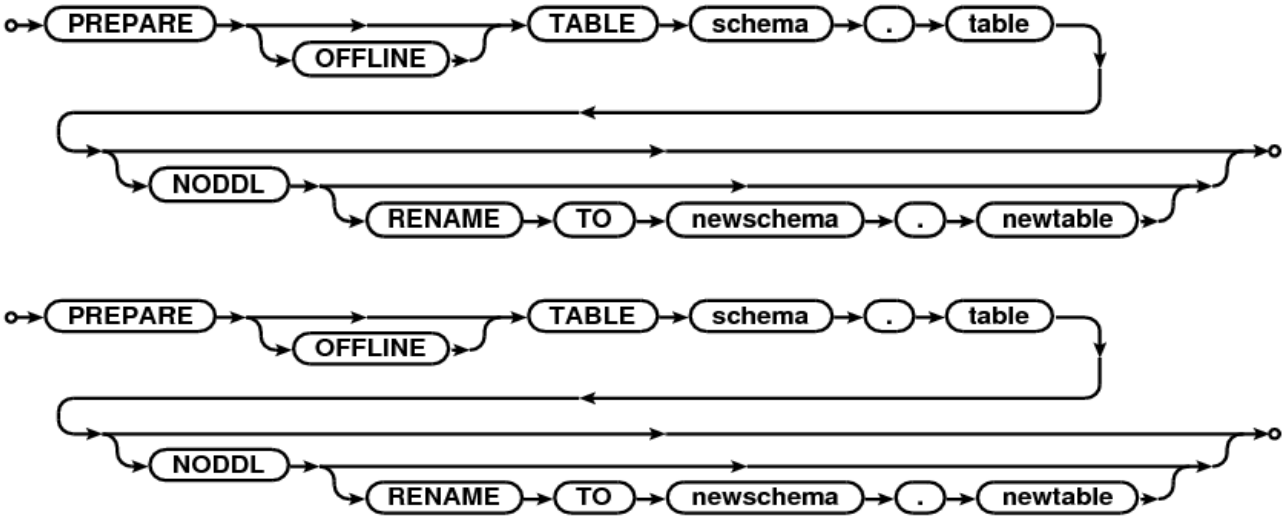


Starts the setup wizard.

VERSION

Shows Dbvisit Replicate version. Can be also invoked by --V switch.

PREPARE



Prepares a schema/table for replication and declares that their content is in sync as of now.

If a schema is prepared and DDL replication is enabled, new tables created in this schema in future will be also prepared and replicated.

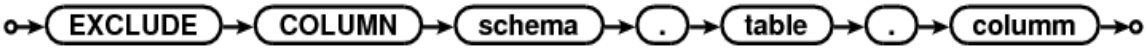
By default, this command connects to running apply and mine and instructs them to replicate the schema/table. If OFFLINE is used, apply/mine will pick the new tables on next start. Use OFFLINE if the apply and mine are not running, as some network configurations may cause long timeouts while PREPARE tries to connect to the apply/mine.

By default, DDL replication is enabled. Use NODDL to disable it (mandatory for non-Oracle databases).

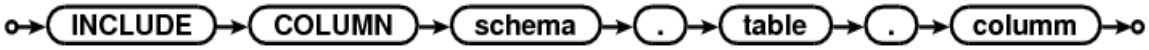
RENAME clause makes the replication to apply the changes to the given schema/table at apply.

Note that the RENAME TO clause requires the NODDL option.

EXCLUDE COLUMN



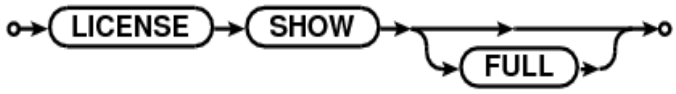
INCLUDE COLUMN



Set a column to be excluded/included in mining. Use this if you don't want to replicate a specific column(s).

Note that PREPARE automatically includes all columns and EXCLUDE thus must follow the prepare.

LICENSE

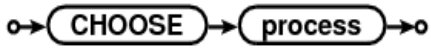


Shows current license, as set by LICENSE_KEY variable.

The optional FULL keyword specifies the verbosity of the command:

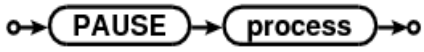
```
dbvrep> license show
Licensed for *, 30-day license (trial license).
dbvrep> license show full
Product:          Dbvisit Replicate
Allowed versions:1.0-2.63
Key-version:      1
License type:     RS1S
Customer id:      1.0 (trial license)
Production:       yes
License-type:     server
Name:             *
Expiry:           30 days
CPU count mine:   0
CPU count apply:  0
Row-count limit:  unlimited
Enabled options:  rac, fetcher, partitions, onetomany, cascade, 2way, ddl, asm,
mysql, mssql, oracle, snmp, mail_notify
Licensed for *, 30-day license (trial license).
```

CHOOSE



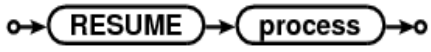
Choose process to work with. This process has to be defined in DDC file (process_name.PROCESS_TYPE).

PAUSE



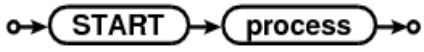
Asks the given process to pause.

RESUME process_name



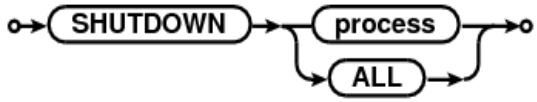
Asks the given process to resume.

START process_name



Starts process.

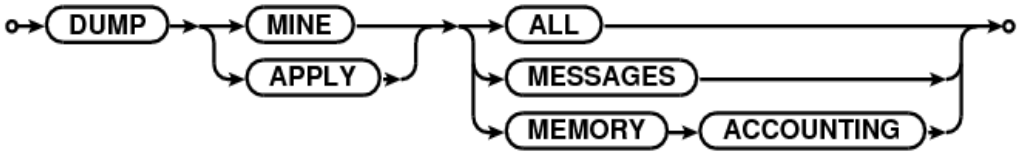
SHUTDOWN process_name



Connect to mine/apply/fetcher and request it to shut down.

ALL means requesting all processes defined in DDC to shut down.

DUMP

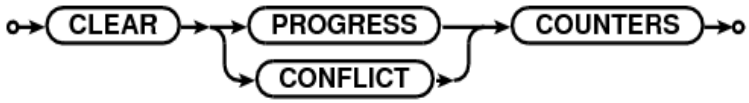


Request mine/apply to dump debug information to log.

DUMP ALL is invoked automatically when exiting due to a untrapped fatal error.

DUMP ALL can be also invoked by sending SIGUSR2 (kill -12) to the process (not available on Windows).

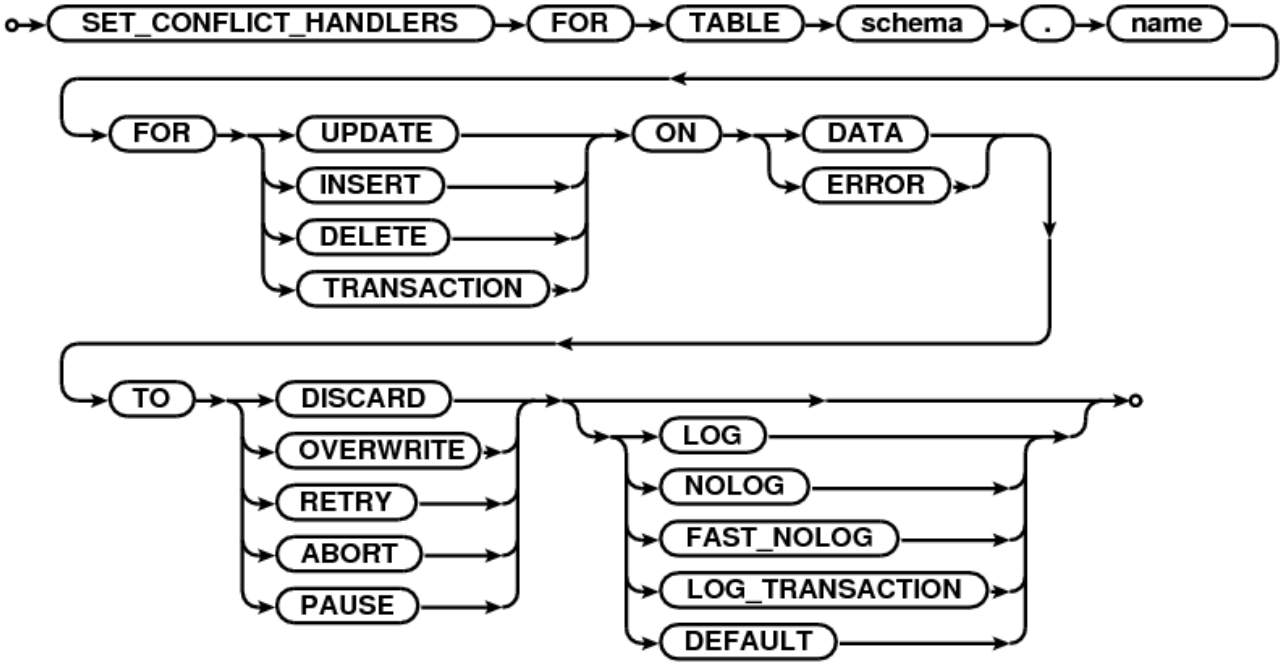
CLEAR

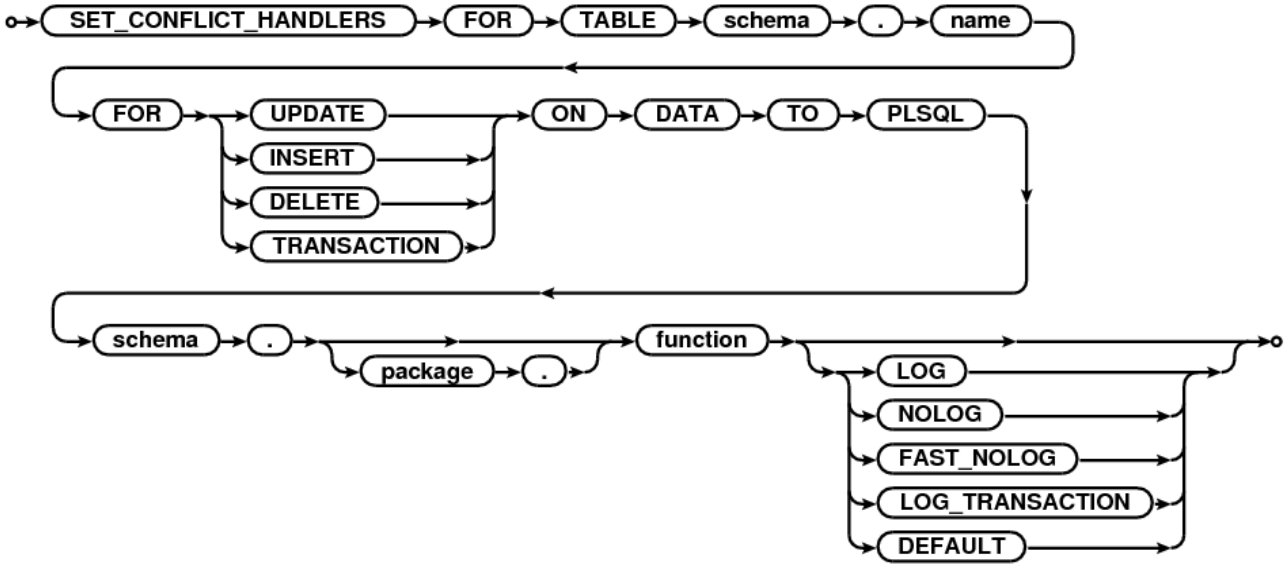


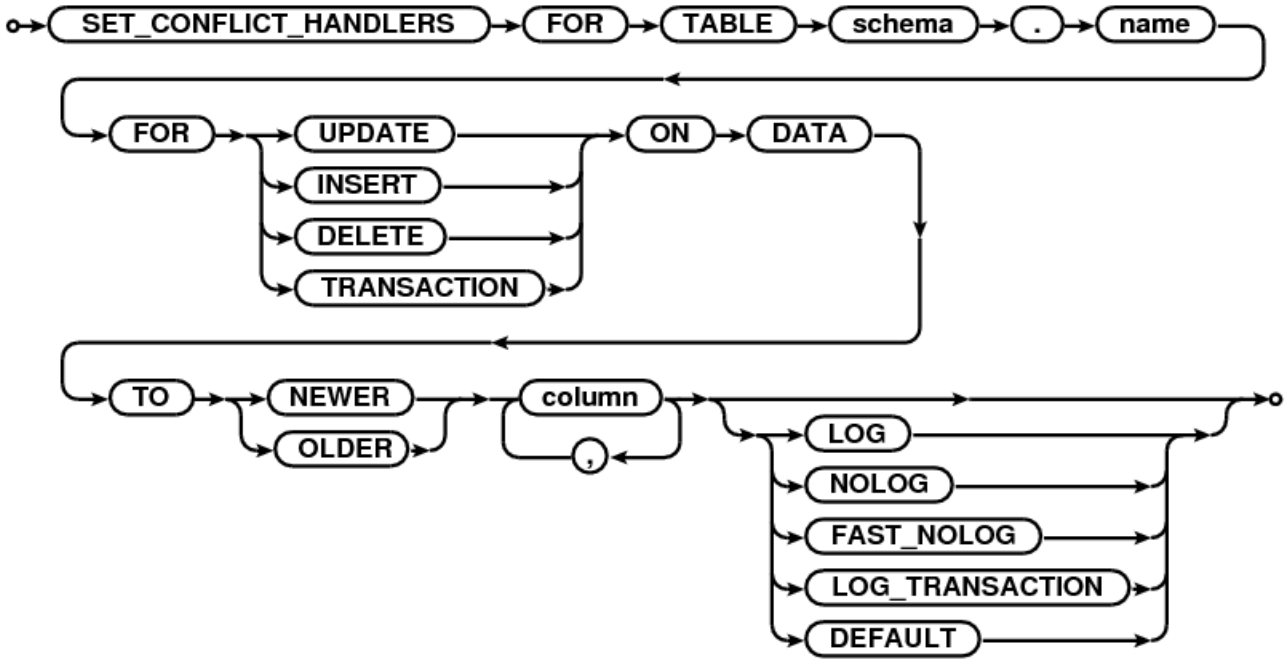
PROGRESS: Asks APPLY to reset conflict counters.

CONFLICT: Asks MINE and APPLY to reset progress counters.

SET_CONFLICT_HANDLERS





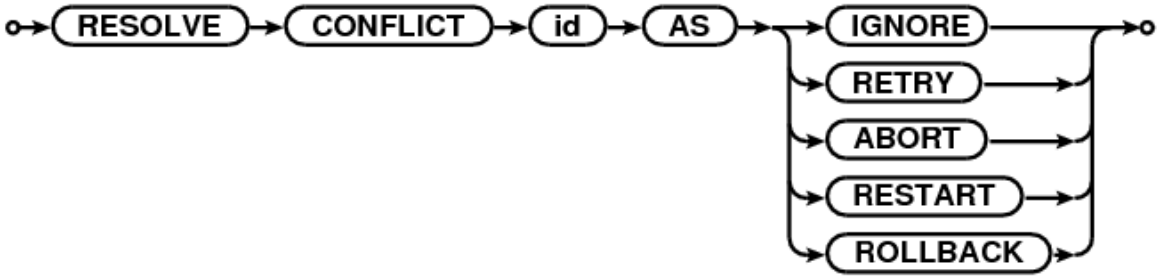


Gets object id as of current mine progress and sets conflict handlers for this table (schema.name refers to current table name at mine).

SHOW_CONFLICT_HANDLERS

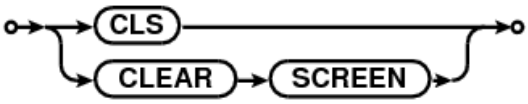
Shows current setting of conflict handlers for given table (schema.name refers to table at mine).

RESOLVE



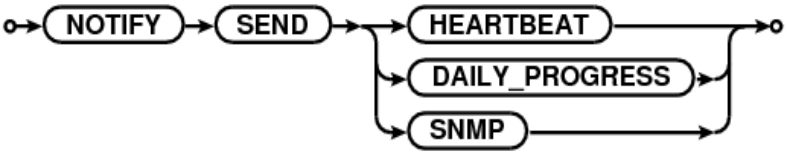
Contacts apply and resolves conflict as IGNORE, RETRY, ABORT, RESTART.

CLEAR SCREEN, CLS



Clear console screen.

NOTIFY



Sends sample email/SNMP trap – use to test your SMTP/SNMP configuration.

CREATE SERVICE process

Create Windows service for given process.

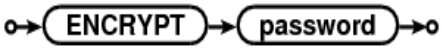
DELETE SERVICE process

Delete Windows service for given process.

START_SERVICE process

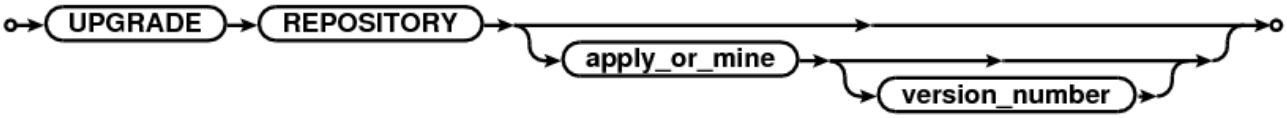
Start Windows service for given process. Same as Windows command NET START.

ENCRYPT password



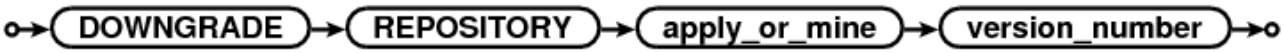
Encrypts the password, so the value can be used in the DDC file, where all passwords are stored encrypted.

UPGRADE REPOSITORY



Upgrade the Dbvisit Replicate repository to a newer version.

DOWNGRADE REPOSITORY



Dbvisit Replicate Configuration Variable Reference

Per-process setting

Usually, variables are set globally, for all processes. This is done by specifying no process name, or *.

If the variable value needs to be set different for a process, use SET with a process name, e.g.

```
SET APPLY.LOG_FILE=/home/oracle/log/%N.%E
```

If the variable needs to be set back to the global "*" setting, use RESET command.

Variable reference

APPLY_CASE_CONVERT

When a table is prepared, should the name from mine be kept as-is, made uppercase, or lowercase? This is honoured for case-sensitive databases only (MySQL and MSSQL).

APPLY_DATABASE

Connection definition for the apply database. This is TNS connection string for Oracle, connection string for MySQL or DSN (Data Source Name) for MS SQL.

APPLY_LISTEN_INTERFACE

Network interface on which apply listens for commands (hostname:port).

APPLY_PASSWORD

Encrypted apply database password. (Use ENCRYPT command to get encrypted password from the plain text password.)

APPLY_PEER

What is the corresponding apply for this process? If SIMPLE_CONFIG = YES, this is simply set to APPLY.

APPLY_RDBMS

Apply database type: Oracle, MySQL, MSSQL.

APPLY_REMOTE_INTERFACE

Where to connect to reach this apply process. This is usually same as APPLY_LISTEN_INTERFACE, but may be changed if the name resolution works differently among the machines, listen binds to selected interfaces only, etc. Note that this settings is used by the console, too.

APPLY_SET_TRIGGER_FIRE_ONCE

How to make triggers to not fire in apply sessions (YES/NO/AUTO).

Yes = enable the set-trigger-firing property (to honor dbms_ddl.set_trigger_firing_property) and abort if fail.

Auto = try to enable it, if possible.

No = disable it, even the package/variable way.

APPLY_SCHEMA

Apply database schema (Oracle, usually same as APPLY_USER), schema=database (MySQL), database (MS SQL).

APPLY_SCHEMA2

Apply database schema (MS SQL only).

APPLY_STAGING_DIR

Directory to store received plogs on apply server.

APPLY_TRACE

OFF (default) or pattern for trace file name (use sequence number %S). The trace file contains all SQL applied as well as further debugging info.

APPLY_USER

Apply database login username.

DDC_BACKUP_DIR

If DDC DB, on every startup of a process a backup of all DDC settings is made into this directory as a backup, e.g. for use with CREATE DDCDB FROM DDCFILE, or just for reference.

DDC_DATABASE

TNS identifier for the database when the DDC DB is stored. (Setup wizard sets it to mine database for this by default.) Note that just setting this variable will not create the DDC DB tables, these are contained in every dictionary.

DDC_ID

Unique id of the DDC. This must not change after configuration is complete. Default: 1.

DDC_NAME

Name of the configuration (%d in filemask). This can change as you want, as internally only the DDC_ID matters, but beware of all %d references.

DDC_PASSWORD

Encrypted ddc db database password. (Use ENCRYPT command to get encrypted password from the plain text password.)

DDC_SCHEMA

DDC database schema, usually same as DDC_USER.

DDC_USER

DDC DB database login username.

DEBUG_LEVEL

Set debugging level. Contact support for possible values.

FETCHER_DATABASE

Mine database TNS (when connecting from fetcher).

FETCHER_ENABLED

Fetcher (downstream capture) enabled, this means mine waits for redologs from fetcher.

FETCHER_LISTEN_INTERFACE

Network interface on which fetcher listens for commands (hostname:port).

FETCHER_PASSWORD

Encrypted fetcher database password. (Use ENCRYPT command to get encrypted password from the plain text password.)

FETCHER_PEER

What is the corresponding fetcher for this process? If SIMPLE_CONFIG = YES, this is simply set to FETCHER.

FETCHER_REMOTE_INTERFACE

Where to connect to reach this fetcher process. This is usually same as FETCHER_LISTEN_INTERFACE, but may be changed if the name resolution works differently among the machines, listen binds to selected interfaces only, etc. Note that this settings is used by the console, too.

FETCHER_SCHEMA

Schema of mine repository (when connecting from fetcher). Virtually always same as MINE_SCHEMA.

FETCHER_THREADS

On RAC, specify a subset of redo log threads that a particular fetcher should handle; it is necessary to send all threads to mine, but it is possible to use multiple fetchers. Use “ALL” (default) or colon-separated list of threads (e.g. “1:3:5”). Default: ALL.

FETCHER_USER

Mine database login username (when connecting from fetcher).

CHECKVARS

Disable/enable checking of variable values. Can be: ON (default), OFF, or comma-separated list of variables NOT to check.

LICENSE_KEY

License key, as obtained by your purchase.

LOG_FILE

General process log file location template.

LOG_FILE_TRACE

General trace file location template.

LOG_FILE_COUNT

How many general log file copies to keep (default 2 = 1 active log and 1 copy).

LOG_FILE_DATE_ROTATE

If not OFF (default), specifies how often rotate the general log files. Allowed values:

yyyy-MM: every month

yyyy-ww: every week

yyyy-MM-dd: every day at midnight

yyyy-MM-dd-a: every day at noon

yyyy-MM-dd-HH: every hour

yyyy-MM-dd-HH-MM: every minute

LOG_FILE_SIZE

Approximate maximum log size in bytes, exceeding this size causes logs to rotate.

LOG_OBSOLETE_AGE_PLOG

When plogs are applied and no longer needed even after apply restart, they are eligible for deletion. In addition, they must be at least LOG_OBSOLETE_AGE_PLOG days old.

LOG_OBSOLETE_AGE_RLOG

When redo logs are mined and no longer needed even after mine restart, they are eligible for deletion. In addition, they must be at least LOG_OBSOLETE_AGE_PLOG days old. NOTE: this applies only for redolog copies shipped to mine from fetcher – if mine reads them directly, they are left to be managed by database administrator, using RMAN, backup scripts etc.

MAILCFG_AUTH_PASSWORD

OFF/password: set the password to be used if the SMTP server requires authentication.

MAILCFG_AUTH_USER

OFF/username: set the username to be used if the SMTP server requires authentication.

MAILCFG_FROM

The From address to be used in outgoing emails

MAILCFG_PORT

The SMTP port to be used (default is the usual 25; however, the usual value for SSL-enabled SMTP server is 465).

MAILCFG_SMTP_SERVER

Sets SMTP mail server to be used for sending emails.

MAILCFG_USE_SSL

Use SSL protocol for the SMTP server (yes/no, default no).

MEMORY_LIMIT_MINE_MB

Memory limit for mine, specified in megabytes. NB that these figures does not include overhead of Perl and of system malloc(), thus do not set it to consume all of your available memory. It does not use ulimit or other OS-based limits. Settings this too low will cause mine to abort.

MEMORY_LIMIT_APPLY_MB

Memory limits for apply, specified in megabytes. NB that these figures does not include overhead of Perl and of system malloc(), thus do not set it to consume all of your available memory. It does not use ulimit or other OS-based limits. Setting this low may cause performance degradation, setting it too low will cause apply to abort.

MINE_ASM

SID for ASM (usually +ASM). Note that on RAC, this must be set to the local ASM node name (e.g. +ASM2).

MINE_DATABASE

Mine database TNS connection string.

MINE_LISTEN_INTERFACE

Network interface on which mine listens for commands (hostname:port).

MINE_PASSWORD

Encrypted fetcher database password. (Use ENCRYPT command to get encrypted password from the plain text password.)

MINE_PEER

What is the corresponding mine for this process? If SIMPLE_CONFIG = YES, this is simply set to MINE.

MINE_PLOG

Template for plog files generated on mine.

MINE_REMOTE_INTERFACE

Where to connect to reach this mine process. This is usually same as MINE_LISTEN_INTERFACE, but may be changed if the name resolution works differently among the machines, listen binds to selected interfaces only, etc. Note that this settings is used by the console, too.

MINE_SCHEMA

Database schema containing the repository tables (usually same as MINE_USER).

MINE_STAGING_DIR

Directory where to store redologs received from fetcher.

MINE_TRACE

OFF (default) or trace file name template. Trace file can be compared with trace file generated by Oracle alter system dump logfile.

MINE_UNIQUE_ID

Unique ID of mine, regenerated every time the mine dictionary is recreated, e.g. when scripts created by setup wizard are re-run. This ID uniquely identifies the plogs, so they do not get mixed up between different replications.

MINE_USER

Mine database login username.

NETWORK_QUALITY

(wan/LAN) – autoconfigure network timeouts, compression, transfer block size, etc. to suit slow or fast network.

NETWORK_TRAFFIC_KEY

Common key for network authorization among fetcher, mine, apply and console.

NOTIFY_ALERT_EMAIL

The error notification emails (=all except list progress and heartbeat) are sent to these addresses. Separate multiple addresses by comma.

NOTIFY_ALL_EMAIL

All notifications emails are sent to these addresses. Separate multiple addresses by comma.

NOTIFY_CONFLICT_THRESHOLD

If the number of conflicts on apply exceeds this threshold, an SNMP trap and/or email is sent.

NOTIFY_DAILY_LIST_PROGRESS_TIME24

Times when the overall progress email should be sent. Use 24-hour time format, separate multiple times by colon, e.g. 0700 or 0800:2000. Note that times near to midnight may be skipped, if no check gets scheduled till midnight (see also NOTIFY_INTERVAL_BETWEEN_CHECKS).

NOTIFY_EXCEED_CYCLE_NUM

How many times must be a notification condition be met before the SNMP trap / email is actually sent. (default:2)

NOTIFY_INTERVAL_BETWEEN_CHECKS

How often the notification checks are performed. (Specify seconds, minutes, hours, days, as needed, e.g. 5m or 1h20m30s).

NOTIFY_PEER_DOWN

ALL/colon-separed list: if the specified peer is down, sends an SNMP trap and/or email.

NOTIFY_PROGRES_DIFFERENCE_PERC

If the lag between apply and mine for any table exceeds this threshold, an SNMP trap and/or email is sent. (This checks the percentage as shown by the LIST PROGRESS command.)

NOTIFY_SCN_DIFFERENCE

SCN difference: if the lag between apply and mine exceeds this threshold, an SNMP trap and/or email is sent.

NOTIFY_SEND_HEARTBEAT_TIME24

Times when the overall heartbeat email should be sent. Use 24-hour time format, separate multiple times by colon, e.g. 0700 or 0800:2000. Note that times near to midnight may be skipped, if no check gets scheduled till midnight (see also NOTIFY_INTERVAL_BETWEEN_CHECKS).

NOTIFY_SEQUENCE_DIFFERENCE

Number of redo logs / plogs: if the lag between apply and mine or mine and fetcher exceeds this threshold, an SNMP trap and/or email is sent.

NOTIFY_SUCCESS_EMAIL

The list progress and heartbeat emails are sent to these addresses. Separate multiple addresses by comma.

ON_ERROR

Should errors be treated fatal? Usually used in DDC file so that failed variable checks are clearly pointed out. SKIP (default): treat normally, EXIT: fatal error.

ON_WARNING

Should warnings be treated fatal? SKIP (default): treat normally, EXIT: fatal error.

ORACLE_HOME

Set this variable for Dbvisit Replicate with the same affect as setting them in the shell environment. Note that ASM connection use bequeath connection and thus need same ORACLE_HOME as that when ASM actually resides.

ORACLE_SID

Set this variable for Dbvisit Replicate with the same affect as setting them in the shell environment for any Oracle client.

PLOG_TRACE_SQL_FORMAT

Format of SQL in conflict log table: BIND / NOBIND / BOTH (show SQL statements with bind variables, show SQL statements with literals, show both).

PROCESS_TYPE

Set type of process to mine/fetcher/apply (this must be set per-process). With SIMPLE_CONFIG, APPLY process is set to APPLY, MINE to MINE and if fetcher is enabled, FETCHER to FETCHER.

PROFILER

OFF / profiler file name. Use only as instructed by support.

RETRY_TIME

Seconds between retries if conflict handling is RETRY.

SETUP_SCRIPT_PATH

Path to setup scripts. Used when packing scripts for support.

SIMPLE_CONFIG

If set to YES, configures processes MINE, FETCHER (if fetcher_enabled) and APPLY. Use for a single one-way configuration.

SNMP_INDEX

Disables/enables SNMP subagent. Also sets the index in the process table presented by the SNMP agent. (Set to 0 to disable SNMP, set to 1 if there is just one process of that particular type on the machine, set to unique values for each process type if there are multiple processes of the same type)

On Windows, no SNMP agent is supported and thus the only value supported is 0.

SNMP_TRAP_COMMUNITY

Sets community (=password) for sending of SNMP traps.

SNMP_TRAP_DESTINATION

OFF/hostname/hostname:port. Sets destination for SNMP traps (notifications).

STATUS_BAR

Show status bar in console

OFF: no status bar (default)

ON or STATUS: show SCNs and time (default setting by setup wizard)

LIST: show “list progress”

LISTALL: show “list progress all”

STATUS+LIST: show both STATUS and LIST

STATUS+LISTALL: show both STATUS and LISTALL

STATUS_BEAT_LCR

How often should be a log message written to log showing number of processed LCRs (in LCR count) (use 0 to disable).

STATUS_BEAT_TIME

How often should be a log message written to log showing number of processed LCRs (in seconds) (use 0 to disable).

STATUS_SORT

Set for each process to specify sort order in status bar. Simple config does this automatically, setting the order FETCHER.STATUS_SORT=01, MINE.STATUS_SORT=02, APPLY.STATUS_SORT=03.

TNS_NAMES

Set this variable for Dbvisit Replicate with the same affect as setting them in the shell environment for any Oracle client.

TWO_TASK

Set this variable for Dbvisit Replicate with the same affect as setting them in the shell environment for any Oracle client.

WATCHDOG_TIMEOUT

Timeout for waiting on lock to be released when applying (most notably for TM/TX locks, i.e. waiting for other sessions modifying the same rows). (Waits ½ of this time before aborting due to internal deadlocks etc., waits full time for other users.)

File templates

Where a template is required, specify a full filename including path, utilizing following placeholders:

- %S – sequence
- %T – thread
- %E – extension (hardcoded per file type – plog, log, ...)
- %P – process type (mine, apply, fetcher)
- %N – process name
- %D – DDC_NAME
- %I - process ID (PID)
- %U - six random characters (letters, numbers, underscore)

Internal variables

Additionally, internal variables exist and are denoted by _ (underscore character), e.g. “_INT_SETTING”. Use only as instructed by support.

Dbvisit Replicate Tips and Tricks

Locations

The following are examples of locations and log files needed during setup:

```
set MINE_PLOG /home/oracle/dbvrep/mine/%S.%E
```

File name and location of the Plog files on the source server.

```
set APPLY_STAGING_DIR C:\app\oracle\dbvrep
```

Location of the Plog files on the target server.

```
set LOG_FILE dbvrep_%P_%D.%E
```

Name of log files on both source and target server.

```
%S is sequence, %T thread, %F original filename (stripped extension),
%P (process),   %E default extension.
```

Handy Tips

Show real time replication on status bar:

```
set STATUS_BAR status+list
```

Show progression during Dbvisit Replicate setup. This will show the internal Dbvisit Replicate tables being replicated. This is the default setting.

```
list progress all
```

Run healthcheck of Dbvisit Replicate.

```
healthcheck
```

Check connectivity of mine and apply. These commands will display the version id of the processes if functioning correctly

```
engine mine send engine id
engine apply send engine id
```


Dbvisit Replicate Trouble Shooting

If Dbvisit Replication fails to replicate or the mine process is not able to connect to the apply process and vice-versa, there are a number of steps that can be taken to diagnose the issue.

General check

1. Check that only one instance of each process is running for a given replication. Always kill all old processes.
2. Pay attention to all errors reported during the run of the setup script.
3. Check that the TNS configuration points to correct databases on all machines involved (mine, apply, console). Ensure there are valid TNS entries so that the apply database can connect to the mine database, and the mine database can connect to the apply database.
4. Ensure firewall ports are open to the default replicate ports (7890, 7891)
5. Ensure there is enough memory to start the mine and apply processes. The default memory allocation is 500MB. This is set by MEMORY_LIMIT_APPLY_MB and MEMORY_LIMIT_MINE_MB.
6. If the dbvrep process suddenly dies, it may mean that there is not enough free memory on the server and the OS will terminate the dbvrep process without warning. This can be verified by running “top” in Linux/Unix while the dbvrep apply or mine process is started.

Apply (or mine) cannot connect

```
|MINE IS running. Currently at plog 1071 and SCN 9832904 (06/10/2011 23:10:19).
Could not connect to APPLY process. Process not started or connection refused.
```

Solution:

1. Check if the mine process is able to connect through the network to the apply process.

```
dbvrep> engine apply send engine status
-1: Connection refused
```

This indicates that there is a network issue. Check to ensure the firewall is not blocking the Dbvisit Replicate ports.

2. Run a health check command:

```
dbvrep> healthcheck
ERROR: Could not connect to APPLY, it is down or unreachable.
(Connection refused)
Connectivity test for APPLY: FAILED
ERROR: Could not connect to APPLY, it is down or unreachable.
(Connection refused)
Connectivity test for MINE: PASSED
2 processes checked.
```

3. Try to ping the unreachable process server; try the ping from the mine machine, the machine console (if it is different from mine machine) and the apply machine itself. Check that the IP addresses reported by the ping are the same. (It definitely should not be 127.0.0.1 unless the replication is completely running on one machine.)
4. If the processes does not seem to see each other, try running “telnet *machine port*” on each of the machines to check connectivity to each other machine. The connection should not be refused. Use exactly same names as in your DDC file (use copy-paste if possible). This checks various DNS problems, typos in names/IPs, closed firewall ports etc.
5. Ensure there is enough free memory for the dbvrep process to start. f the dbvrep process suddenly dies, it may mean that there is not enough free memory on the server and the OS will terminate the dbvrep process without warning. This can be verified by running “top” in Linux/Unix while the dbvrep apply or mine process is started.

Apply does not replicate the changes

Check that the particular table exists in the Dbvisit Replicate data dictionary table. The following query on the run as the Dbvisit Replicate owner on the target database can be run to verify:

```
SQL> select name, obj_ from DBRSOBJ$ where obj_ in (select SOURCE_OBJECT_ID from
DBRSAPPLY_DICT_TABLES);
NAME                            OBJ_
------------------------------ ----------
DBRSCOL$                       78949
DBRSOBJ$                       78951
DBRSTAB$                       78953
DEPT                           74209
EMP                            74211
BONUS                          74213
SALGRADE                       74214
DBRSUSER$                      78955
DBRSV_$DATABASE                78957
```

Only the tables listed in this table will be replicated. To replicate other tables use the PREPARE command.

Contacting support

Dbvisit Replicate support is reachable at http://www.dbvisit.com/support/service_desk/, the single point of contact for all Dbvisit products.

To help collect all the information needed by the support, Dbvisit Replicate contains a built-in trace file packaging facility. Use command SUPPORT PACKAGE *process* to create a zip file containing the files for the specified process, running this command in turn on fetcher, mine and apply servers.

```
$ ./dbvrep --ddcfile LINA.ddc support package MINE
Initializing.....done
DDC loaded from database (124 variables).
Dbvisit Replicate version 2.0.01.740
Copyright (C) Avisit Solutions Limited and Dbvisit Software Limited.
All rights reserved.
DDC file LINA.ddc loaded.
Packaging into ZIP file: dbvisit_support_dbvrep_MINE.7IPy.zip for process MINE,
 addinfo []
Trying retrieving +DATA/src/onlinelog/group_1.276.748600819 from ASM. If this
fails, restart with NOASM added to the SUPPORT command.
Packaging data from database. If this fails, restart SUPPORT PACKAGE with NODB
option.
====> If instructed by support, run as SYSDBA the script  get_logtrace.jTZ4.sql and
upload the generated tracefile (script will show the name) to Dbvisit Support as
well.
```

The created file includes the process name and a unique identifier, so multiple runs of the packaging command won't overwrite past zip files.

The command can take further arguments, forcing it to skip some files – use it only if packaging otherwise fails.

NLS considerations

Dbvisit Replicate relies on the target database client settings for handling of NLS issues. For an Oracle target database, set the NLS_LANG environment variable to “AMERICAN_AMERICA.*source_db_charset*”. The AMERICAN_AMERICA part ensures unified number and date formats and specifying the source database charset ensures that the client libraries handle any non-ASCII characters properly.

Setup how-to

Install the software on all machines (optional fetcher, mine, apply, console).

Run the setup wizard on mine (it is run only once for the replication, do not run it again on apply).

Edit and review the scripts, especially:

- Check directories in -onetime.ddc: The wizard creates the directories on mine and you have to create them on fetcher and apply. This is the ideal time to change the directory settings if needed, e.g. if the machines have different directory structure. Use the per-process SET syntax in such case. (e.g. set APPLY.LOG_FILE=c:\dbvrep\log\dbvrep_%D_%N.%E)
  Check TNS_ADMIN and ORACLE_HOME, too.
- If the machines uses a mix of Linux (Unix) and Windows, change the scripts to correctly start the processes, as this differs between Windows and other platforms.
- Copy the.ddc file to fetcher/apply/console. (No need to copy the -onetime.ddc, this is used only once during the setup.dbvrep script and it's content is then stored in DDC DB in database).

After testing, consider adding the starting of dbvrep processes to init scripts on Linux/Unix to automatically start them on server reboot. (This is not needed on Windows, as processes are registered as services by default).

Windows services, Linux/Unix start

On Linux/Unix, the processes are started using the parameter --daemon:

```
$ ./dbvrep --ddcfile LINA.ddc --daemon start MINE
```

The --daemon parameter causes the process to detach from the console and run in the background.

The -all.sh script generated by the setup wizard starts the processes this way.

On Windows, processes are registered as services and started by Windows automatically.

The -all.bat script deletes any old service for the process/ddc, creates a new one and starts it:

```
$ ./dbvrep --ddcfile LINA.ddc delete service MINE
$ ./dbvrep --ddcfile LINA.ddc create service MINE
$ ./dbvrep --ddcfile LINA.ddc start_service MINE
```

The start_service command does exactly the same as pressing start button in Services console or using standard Windows utility “NET START”.

DDC file and DDC DB

All the persistent variable settings are stored in the DDC, which stands for “Dbvisit Database Configuration”. This comes in two flavors:

- DDC file: a text file stored on the filesystem
- DDC DB: a table in an Oracle database, by default the mine database is used

The DDC DB is optional; if it is used, DDC file contains only credentials to the database:

- DDC_DATABASE
- DDC_USER
- DDC_PASSWORD
- DDC_SCHEMA
- TNS_ADMIN
- ORACLE_HOME
- and instruction to load the rest from DDC DB (LOAD DDCDB command).

If the DDC DB is not used, the DDC file contain all variable settings.

The main purpose of DDC DB is to enable easier setting changes – issuing a SET command updates the setting in database, obliterating the need to edit the DDC files on every machine.

The DDC DB can contain any variable, except those listed above that must be known in order to connect to the DDC database.

When DDC DB is used, the SET command automatically updates the DDC DB. If this is not desired, MEMORY_SET command can be used to set the setting in memory only. This is useful for temporary variable setting in scripts, when the variable change is not meant to be persistent.

Note that MINE_UNIQUE_ID variable is set by the -setup.dbvrep script and if DDC DB is not used, DDC file must be manually updated with the new value. The purpose of this variable is to distinguish between repeated runs of setup scripts.

The dbvrep parameter --no-ddcdb (or --noddcdb) disables the LOAD DDCDB command. This is used for the first time startup, when the DDC DB is yet to be created and thus LOAD DDCDB would fail.

DDC Commands

A DDC file can be created from the DDCDB. The command is:

```
CREATE DDCFILE filename FROM DDCDB
```

The DDC file will be created in the filename specified.

A DDC DB can be created from a DDC file. The command is:

```
CREATE DDCDB FROM DDCFILE
```

This takes the active DDC file that dbvrep has been loaded with (through readddc or --ddfile commands) and loads the settings into the DDC DB.

DDC_BACKUP_DIR

On every startup of a process a backup of all DDC settings is made into the DDC_BACKUP_DIR directory as a backup, e.g. for use with CREATE DDCDB FROM DDCFILE, or just for reference.

Network encryption

All network commands among fetcher, mine, apply and console are sent encrypted. The key serves both as authentication and encryption.

On WAN networks, even the redo/plog contents is sent encrypted.

The key is defined by variable NETWORK_TRAFFIC_KEY. As this can be considered as sensitive information, there are more ways how to specify this value:

- DDC file (use memory_set, so it is not stored in DDC DB automatically)
- DDC DB
- the .ssh way: create a private directory (chmod 0700), store a file with a single command “memory_set NETWORK_TRAFFIC_KEY …” and include it from your DDC file (standard READ command)
- specify the key as dbvrep parameter: --netkey

Dbvisit Replicate Dictionary tables

Usually, there is no need to directly access most of the tables. Nevertheless, some of the tables can be interesting:

DBRSAPPLY_CONFLICT_LOG

List of conflicts encountered, including failing SQL.

DBRSAPPLY_CONFLICT_HANDLERS

Conflict handler settings. Note that the key is object id on source database.

DBRSAPPLY

Includes last commit SCN (current_scn), so it can be used to monitor apply progress if SNMP or log monitoring is not an option.