DataEX 软件安装配置手册

Version 6.0

2024 年 3 月

TOMcoding.com

前言

DataEX 是一款类似 Oracle GoldenGate 的数据复制软件，抽取进程（extract）从 Oracle 源端数据库的重做日志中分析数据变化，然后转换成逻辑改变记录(LCR)，软件以事务为单位写入跟踪文件(trail)队列，应用进程(apply)从跟踪文件读取事务信息和数据，然后装载到目标端数据库中。

软件运行环境为 Intel x86_64 上的 linux 操作系统，采用集中部署、集中管理的模式。

本文档以实际例子的方式演示软件的安装配置过程。

工作机制和软件结构

软件的抽取进程，读取 Oracle 的重做（redo）日志，优先读取在线重做日志，只有当在线日志被覆盖时，才去读取归档日志。分析日志中的重做记录，采用硬解析的方式翻译重做记录中的数据，结合数据字典，把关注的对象操作类型和操作数据转换成逻辑改变记录(LCR)，然后通过事务集成器按照事务 ID 把 LCR 组合成一个个事务数据包。然后把提交的事务写入跟踪文件（trail）队列中。

软件的应用进程，检测跟踪文件，当有新数据到达时，读取事务数据，根据不同的操作类型，应用到目标数据库中，完成源端到目标端数据库增量数据的复制。

软件采用集中管理的模式，所有软件程序都安装在一台机器上，每个抽取进程对应一个源端 Oracle 实例（instance），每个应用进程对应一个目标数据库。软件系统可以同时启动多个抽取进程和应用进程，源端数据库和目标端数据库是多对多的关系，一个源端数据库的变化数据可以应用到多个目标端，多个源端数据库的变化数据也可以应用到一个目标端。软件通过一个 manager 进程统一管理这些复制线路。复制线路可以进行分组，每一组叫做一个复制单元，方便管理。

软件的部署

软件部署在一台 Intel x86_64 架构的 64 位 linux 操作系统服务器上。抽取进程通过本地（local）和远程（remote）两种方式读取数据库日志。当 DataEX 软件与 Oracle 数据库实例部署在同一台机器上时，抽取进程通过本地读的方式读取日志。当 DataEX 软件与 Oracle 数据库实例部署在不同的机器上，但是通过网络能够把 Oracle 日志挂载（mount）到 DataEX 软件服务器上时，抽取进程也通过本地读的方式读取日志。其他情况抽取进程通过远程读的方式读取日志，这种模式下，需要在 Oracle 的实例服务器上安装一个读取代理程序。抽取进程连接远程的代理进程，通过代理读取日志内容，返回给抽取进程。

本地读的方式，不需要在其他地方安装任何程序。

远程读的方式，需要在 Oracle 实例服务器上安装读取代理程序。

下面以实际的例子来演示安装和配置的过程。

创建 linux 用户

创建一个 linux 用户，DataEX 软件安装在这个用户下，赋予用户权限，能够读取 Oracle 的日志文件。

用 root 用户登录 linux 系统，创建一个用户，例如用户名为 topdx，命令如下：

useradd topdx

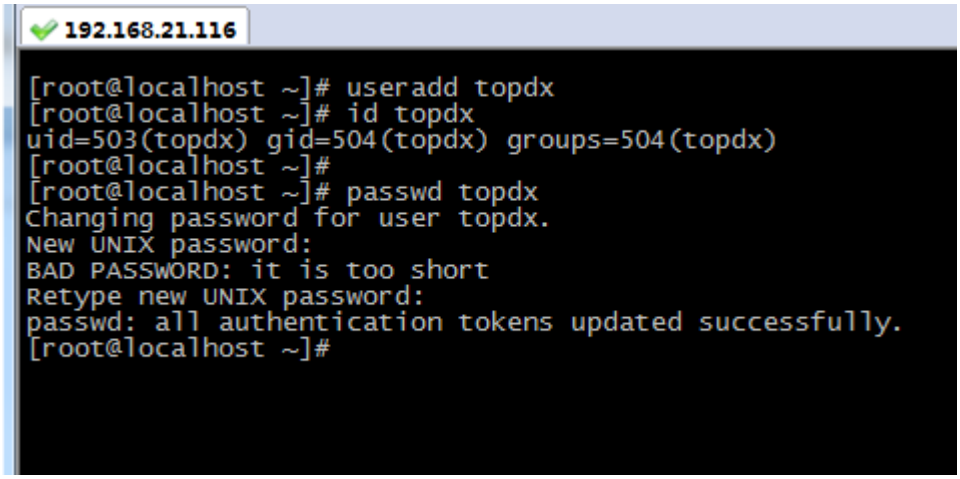
检查用户：

id topdx

修改用户密码，设置密码与用户名相同（这里只做演示用，实际中请设置规范的密码）：

passwd topdx

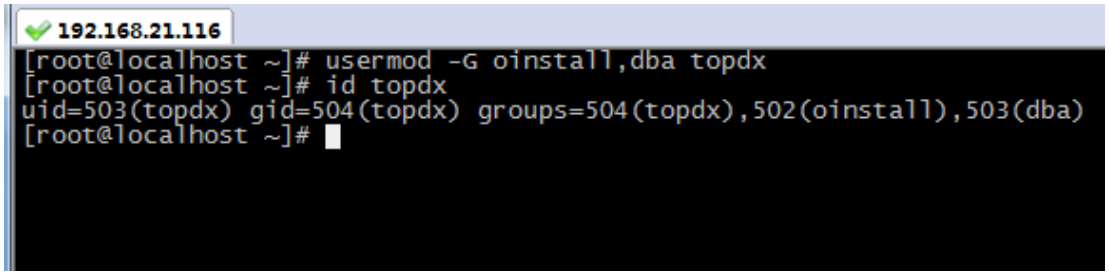
操作过程和结果如下图：



如果软件与 Oracle 服务器部署在一起，还要把用户 topdx 加入 Oracle 用户组，这样 topdx 用户就能够有读取 Oracle 日志文件的权限，如果 topdx 用户所在的机器与 Oracle 安装的机器不是同一台，忽略这一步操作。

命令如下：

usermod -G oinstall,dba topdx

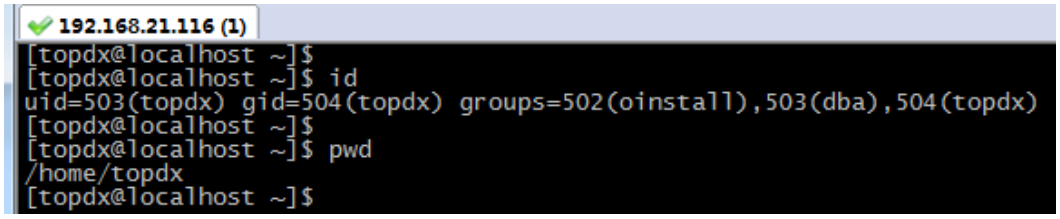


安装软件

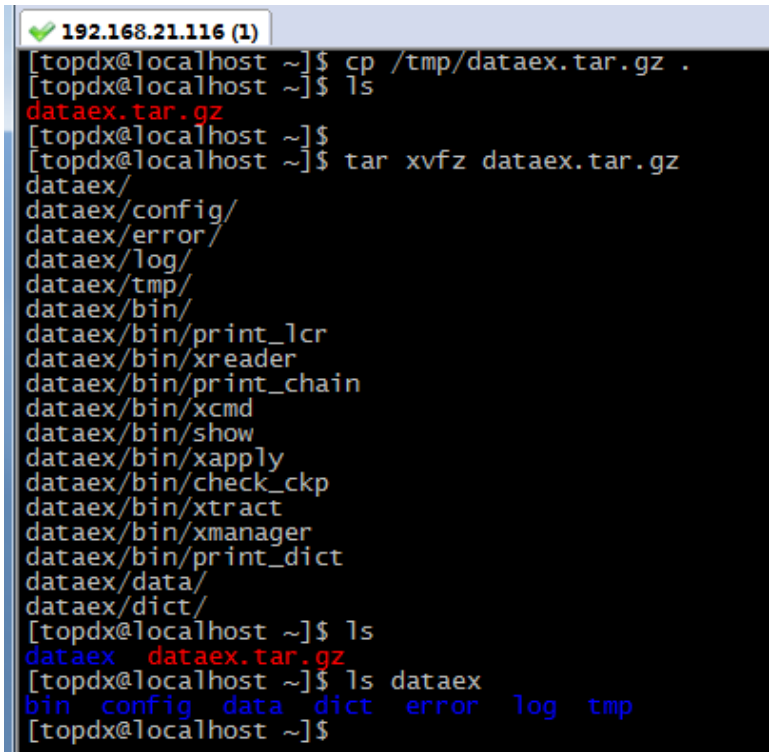
安装 DataEX 软件

下面以 linux 用户 topdx 为例来演示安装的步骤。

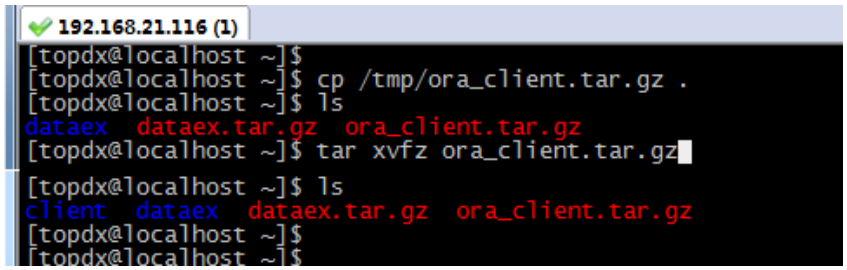
以用户 topdx 登录 linux 系统。



把下载的软件包 dataex.tar.gz 拷贝或传输到/home/topdx 目录下，然后解压。



把下载的软件包 ora_client.tar.gz 拷贝或传输到/home/topdx 目录下，然后解压。



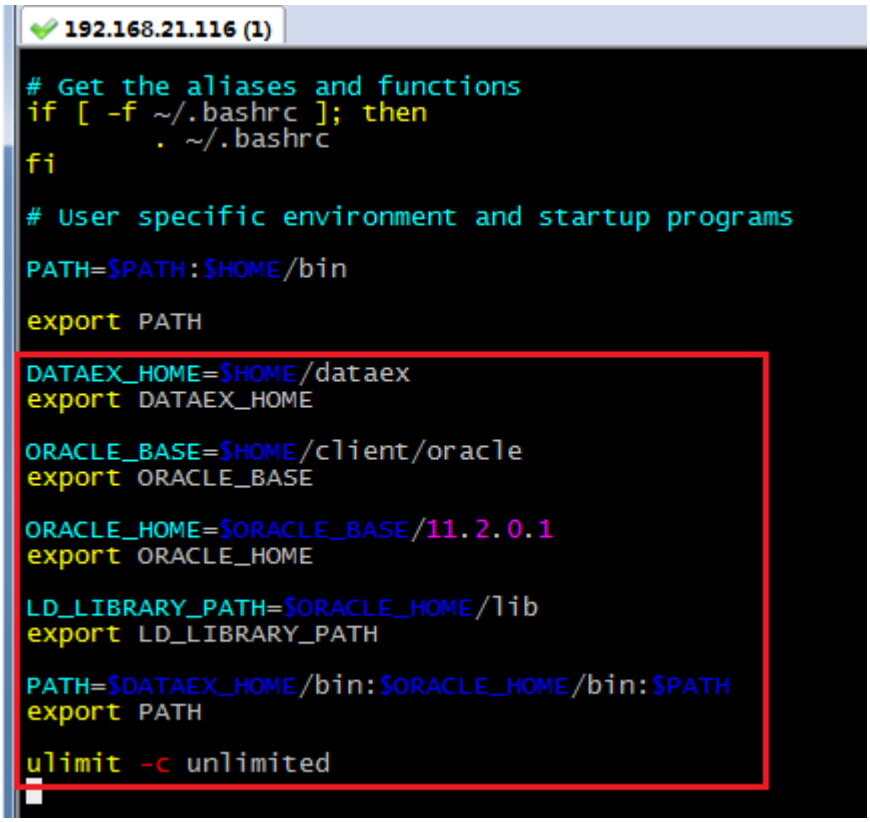
删除掉下载的软件包

rm *.tar.gz

下面设置软件运行的环境变量。

在 topdx 用户目录/home/topdx 下，编辑.bash_profile 文件，添加以下内容：

```
DATAEX_HOME=$HOME/dataex
export DATAEX_HOME

ORACLE_BASE=$HOME/client/oracle
export ORACLE_BASE

ORACLE_HOME=$ORACLE_BASE/11.2.0.1
export ORACLE_HOME

LD_LIBRARY_PATH=$ORACLE_HOME/lib
export LD_LIBRARY_PATH

PATH=$DATAEX_HOME/bin:$ORACLE_HOME/bin:$PATH
export PATH

ulimit -c unlimited
```



安装 reader 软件

reader 软件用于远程读取 Oracle 数据库的在线和归档日志文件。当复制软件部署的节点不在源端数据库上时，用做读取代理。

在安装 Oracle 数据库实例的机器上创建 topdx 用户，把 topdx 用户加入到 oracle 用户组。

以 topdx 用户登录，在 topdx 用户的登录目录下创建 reader 目录。

如果不想创建新用户，可以选择在 oracle 用户下创建 reader 目录。

执行命令：

```
mkdir reader
```

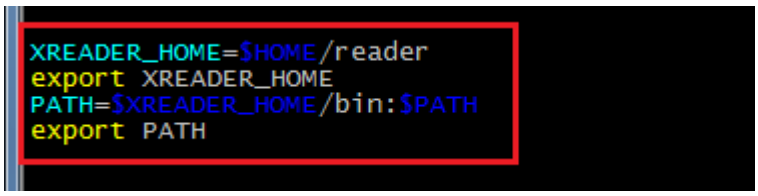
创建子目录 bin 和 log，执行命令：

```
mkdir reader/bin
mkdir reader/log
```

把 dataex/bin/xreader 程序拷贝或传输到 reader/bin 目录下。

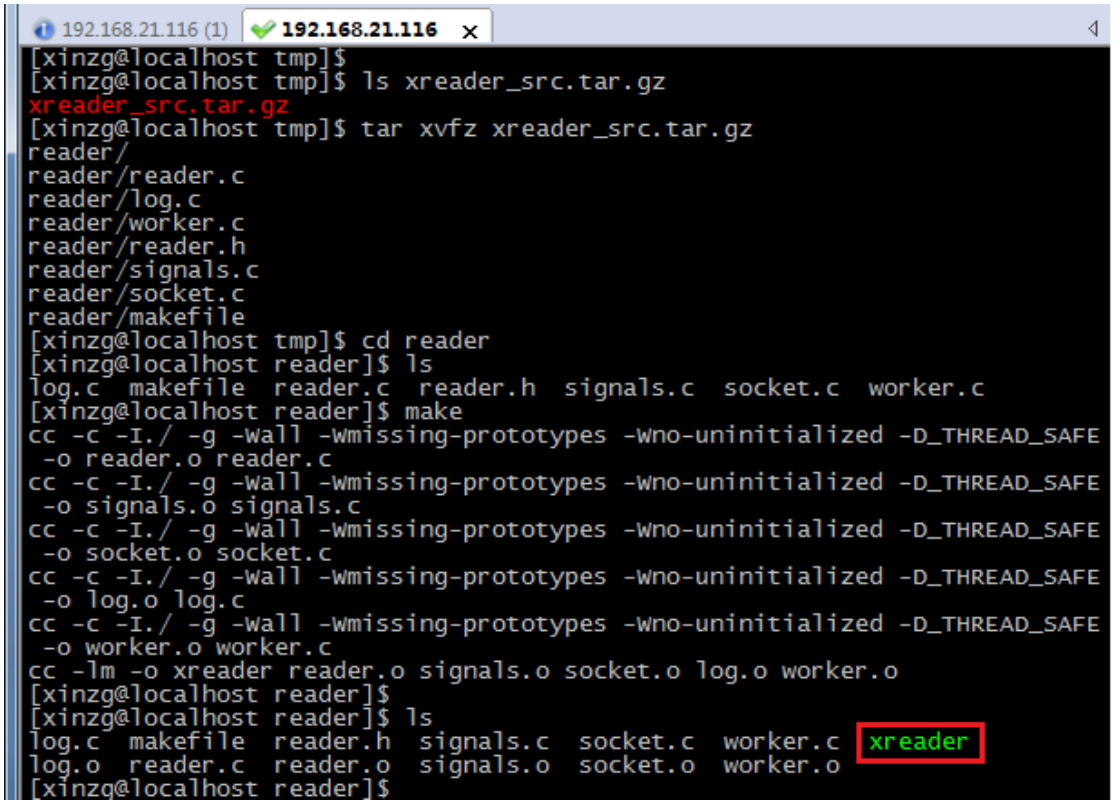
程序运行需要，设置环境变量 XREADER_HOME=/home/topdx/reader，把环境变量添加到.bash_profile 中。

```
XREADER_HOME=$HOME/reader
export XREADER_HOME
PATH=$XREADER_HOME/bin:$PATH
export PATH
```



随程序附带的 xreader 程序是在 Intel x86_64 架构的 linux 系统编译的，只适用于相同的环境。如果是不同的操作系统或硬件架构，请下载源代码，重新编译产生 xreader 程序。

下载 xreader_src.tar.gz 软件包。解压缩到/tmp 目录下，进入 reader 目录，进行编译。



把编译生成的 xreader 拷贝到/home/topdx/reader/bin 目录下。

Oracle 数据库设置

源端创建数据捕捉用户

软件运行时需要在源端 Oracle 数据库中创建一个用户，用于导出数据字典和获取日志的名称。为用户授权最基本的权限，授权用户读取数据字典的权限。以创建 dxcap 用户为例，操作命令如下：

```
create user dxcap identified by dxcap default tablespace users;
```

为 dxcap 用户授权：

```
grant connect, resource to dxcap;
grant select any dictionary to dxcap;
```

如果源端数据库版本是 oracle 12c 以上，上面赋的权限不能读取系统用户表 user$，需要单独授权：

grant select on sys.user$ to dxcap;

```
SQL> create user dxcap identified by dxcap default tablespace users;
User created.
SQL> grant connect, resource to dxcap;
Grant succeeded.
SQL> grant select any dictionary to dxcap;
Grant succeeded.
```

目标端创建数据装载用户

软件运行时需要在目标端 Oracle 数据库中创建一个用户，用于把数据应用到目标端数据库，这个用户需要 dba 权限。以创建 dxapp 用户为例，操作命令如下：

```
create user dxapp identified by dxapp default tablespace users;
```

为 dxapp 用户授权：

```
grant connect, resource, dba to dxapp;
```

```
SQL> create user dxapp identified by dxapp default tablespace users;
User created.
SQL> grant connect, resource, dba to dxapp;
Grant succeeded.
```

Oracle 11g 设置归档模式

软件运行时，需要把源端数据库设置为归档模式，这样当数据库比较繁忙时，可以保证日志数据不丢失。否则当日志被覆盖，而软件分析滞后时，数据会丢失，软件不能正常运行。

如果数据库运行平稳，不会在短时间内产生大量日志，软件也可以在非归档模式下运行。

设置归档模式的步骤如下：

1. 创建归档日志保存目录。如果是文件存储，先在操作系统层创建一个归档目

录。如果是 ASM，在磁盘组中创建一个目录。

2. 启动数据库到 mount 状态
3. 修改归档路径位置
4. 修改数据库为归档模式
5. 启动数据库到 open 状态

下面以数据库为 ASM 存储，磁盘组为 DG1，来演示设置归档模式。

以系统用户连接到 oracle 数据库，查看归档模式。

```
SQL> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            /ora_app/oracle/product/11.2.0/dbs/arch
Oldest online log sequence     16
Current log sequence           18
```

看到数据库不是归档模式，归档路径也不是 ASM。

先修改归档路径，查看磁盘组和数据库 sid。

```
SQL> select name from v$asm_diskgroup;

NAME
------------------------------
DG1

SQL> select instance_name from v$instance;

INSTANCE_NAME
----------------
ora11gasm
```

修改磁盘组增加归档路径。

```
alter diskgroup DG1 add directory '+DG1/arch';
```

修改数据库的归档路径。

```
alter system set log_archive_dest_1='location=+DG1/arch'
scope=spfile sid='ora11gasm';
```

关闭数据库。

```
shutdown immediate;
```

数据库启动到 mount 状态。

```
startup mount;
```

修改数据库为归档模式。

```
alter database archivelog;
```

打开数据库。

```
alter database open;
```

查看数据库归档属性。

```
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            +DG1/arch
Oldest online log sequence     17
Next log sequence to archive   19
Current log sequence           19
```

数据库已设置为归档模式。

Oracle 12c 设置归档模式

Oracle 12c 以上的版本，归档模式只能在 CDB 一级开启，开启后，所有 PDB 都启用归档模式。

修改归档模式的步骤与 Oracle 11g 一样。下面以 pdb 名称为 pdb_dx，来演示设置归档模式的步骤。

```
SQL> startup mount;
```

在 CDB 一级查看归档模式。

```
SQL> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            /u01/oracle/product/19.3.0/db_1/dbs/arch
Oldest online log sequence     8
Current log sequence           10
```

切换到 PDB 一级查看归档模式。

```
SQL> alter session set container=pdb_dx;

Session altered.

SQL> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            /u01/oracle/product/19.3.0/db_1/dbs/arch
Oldest online log sequence     8
Current log sequence           10
```

切换回 CDB，然后开启归档模式。

```
SQL> alter session set container=cdb$root;
```

设置归档路径，在 linux 下创建归档目录。

```
mkdir /u01/oracle/arch1

SQL> alter system set log_archive_dest_1='location=/u01/oracle/arch1' sid='*';
```

开启归档模式。

```
SQL> alter database archivelog;
```

打开数据库。

```
SQL> alter database open;
```

打开 pdb 数据库。

```
SQL> alter pluggable database pdb_dx open;
```

添加补充日志

在更新数据行时，为保证数据的唯一性，需要在日志中记录额外的信息，这部分信息叫做补充日志（supplemental log）。DataEX 软件的运行需要在源端数据库中添加补充日志。

添加数据库最小补充日志，这一步是后续添加主键或 UK 补充日志的先决条件。

```
alter database add supplemental log data;
```

在数据库层级添加主键和 UK 补充日志。

```
alter database add supplemental log data (primary key, unique) columns;
```

如果不想在数据库层级添加补充日志，可以在表级添加补充日志。

```
alter table table_name add supplemental log data (primary key, unique) columns;
```

查看补充日志情况。

```
SQL> select supplemental_log_data_min min, supplemental_log_data_pk pk,
     supplemental_log_data_ui ui from v$database;

MIN      PK  UI
-------- --- ---
YES      YES YES
```

对于 12c，也是在 CDB 层面添加补充日志，然后会自动添加大 PDB 数据库中。

如果从 PDB 层添加补充日志，会报错，如下：

```
ORA-65040: operation not allowed from within a pluggable database
```

强制写日志

强制写（force logging），是强制一些 nologging 表，操作也必须记录到重做日志中，避免丢失数据。

打开强制写日志模式。

```
alter database force logging;
```

查看强制写日志状态。

```
SQL> select force_logging from v$database;

FOR
---
YES
```

对于 12c，也要在 CDB 层打开强制写日志。

配置软件

配置软件的命令在控制台命令行程序 xcmd 中执行。以用户 topdx 登录 linux 系统，执行 xcmd 命令，进入控制台命令行界面。

添加源端数据库

Oracle 数据库在 12c 以后采用了多租户的概念，DataEX 软件的每个抽取进程只针对一个数据库，对于多租户系统，只针对一个 PDB（pluggable database），因此在添加数据库时也有差别，下面分别以 11g 和 12c 数据库来演示。

Oracle 11g 添加数据库

假设数据库名称为 source_db，存储方式为 ASM，远程读取，以此为例添加源端数据库命令如下：

```
add db source_db
(
  storage_asm=yes,
  remote=yes,
  user=dxcap,
  password=dxcap,
  (instance_name=inst1,
    thread=1,
    service=ora11g_asm,
    host=192.168.16.130,
    port=1521,
    agent_port=9001
  )
);
```

上例中 source_db 是数据库在 DataEX 软件系统中的名称，唯一标识一个数据库。

storage_asm 属性指示数据库存储方式是否 ASM，如果是取值 yes，否则取值 no。

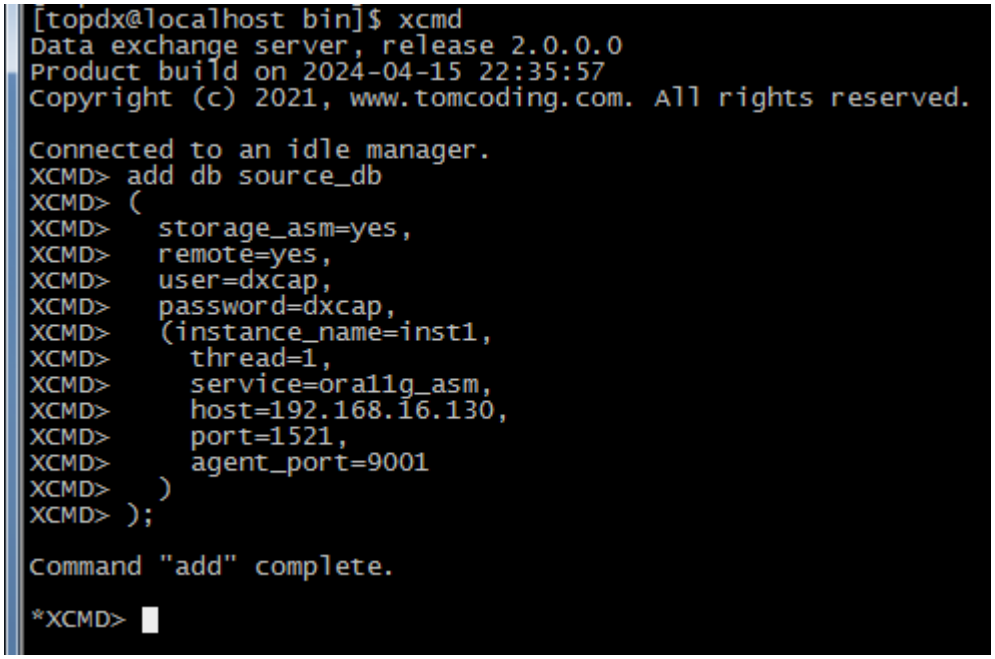
remote 属性指示抽取进程读取日志的方式，本地读取为 no，通过代理读取为 yes。

user 和 password 是连接数据库时的用户名和密码，是前面创建的数据捕捉用户。

instance_name 和 thread 是数据库实例的名称和线程号。

service，host 和 port 是连接数据库的网络参数，与 TNS 配置中的意义相同。

agent_port 是读取代理程序的 socket 端口，在远程读取模式必须配置。本地读取可以忽略。

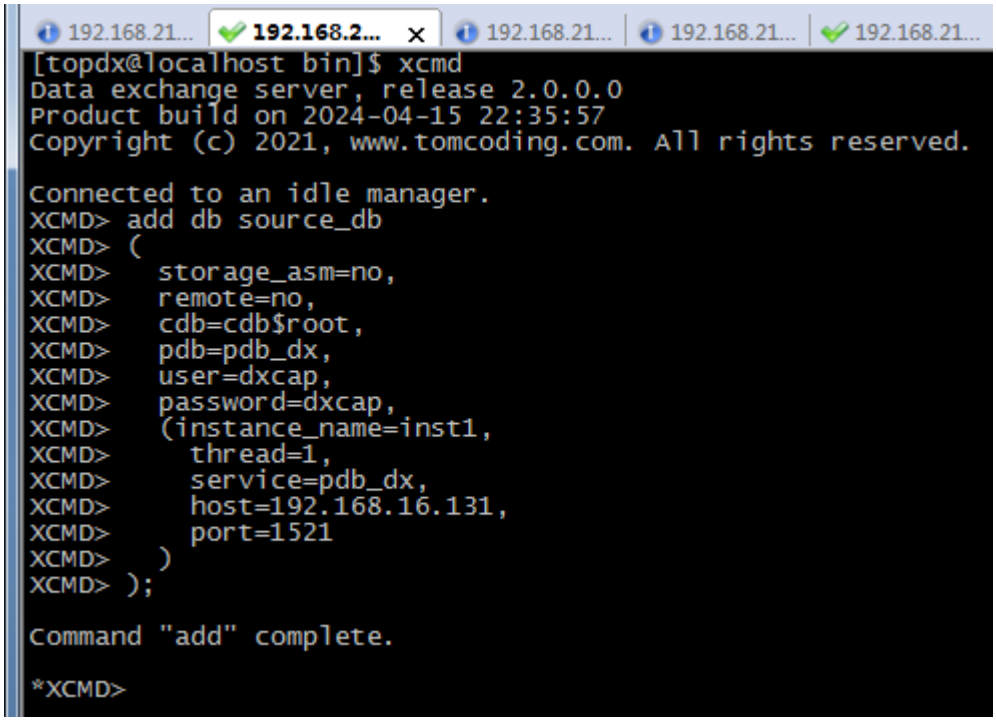


Oracle 12c 添加数据库

假设数据库名称为 source_db，CDB 名称为 cdb$root，PDB 名称为 pdb_dx，存储方式为文件，本地读取，以此为例添加源端数据库命令如下：

```
add db source_db
(
  storage_asm=no,
  remote=no,
  cdb=cdb$root,
  pdb=pdb_dx,
  user=dxcap,
  password=dxcap,
  (instance_name=inst1,
    thread=1,
    service=pdb_dx,
    host=192.168.16.131,
    port=1521
  )
);
```

在 12c 以上的多租户系统中，必须指定 pdb 名称。



添加目标端数据库

Oracle 11g 添加数据库

假设数据库名称为 target_db，存储方式为文件，以此为例添加目标端数据库命令如下：

```
add db target_db
(
  storage_asm=no,
  remote=no,
  user=dxapp,
  password=dxapp,
  (instance_name=inst1,
    thread=1,
    service=ora11g,
    host=192.168.16.130,
    port=1521,
    agent_port=9001
  )
);
```

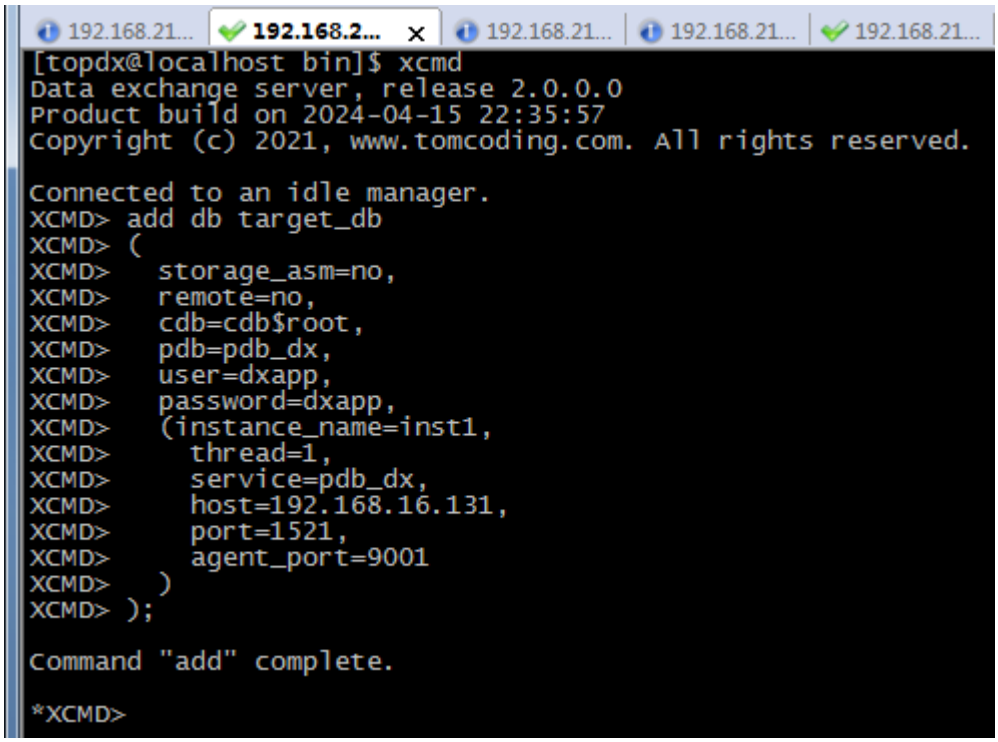
目标端添加数据库，storage_asm，remote，agent_port 参数可以忽略，这些参数指定读取日志的方式，对目标端数据库没有影响。

```
*XCMD>
*XCMD> add db target_db
*XCMD> (
*XCMD>   storage_asm=no,
*XCMD>   remote=no,
*XCMD>   user=dxapp,
*XCMD>   password=dxapp,
*XCMD>   (instance_name=inst1,
*XCMD>     thread=1,
*XCMD>     service=ora11g,
*XCMD>     host=192.168.16.130,
*XCMD>     port=1521,
*XCMD>     agent_port=9001
*XCMD>   )
*XCMD> );

Command "add" complete.

*XCMD>
```

Oracle 12c 添加数据库

假设目标端数据库名称为 target_db，PDB 名称为 pdb_dx，以此为例添加目标端数据库命令如下：

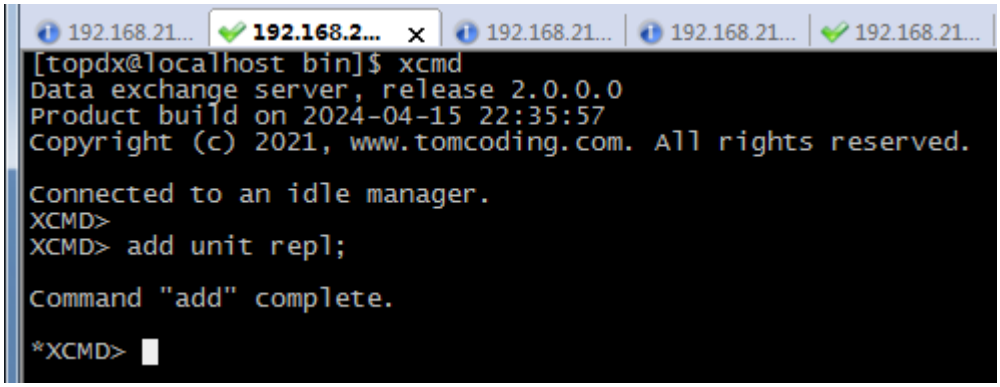
```
add db target_db
(
  storage_asm=no,
  remote=no,
  cdb=cdb$root,
  pdb=pdb_dx,
  user=dxapp,
  password=dxapp,
  (instance_name=inst1,
    thread=1,
    service=pdb_dx,
    host=192.168.16.131,
    port=1521,
    agent_port=9001
  )
);
```



创建复制单元

每个组件都隶属于唯一一个复制单元，复制单元是组件的容器。所以首先要创建一个复制单元。假设复制单元名称为 repl，添加复制单元的命令如下：

```
add unit repl;
```



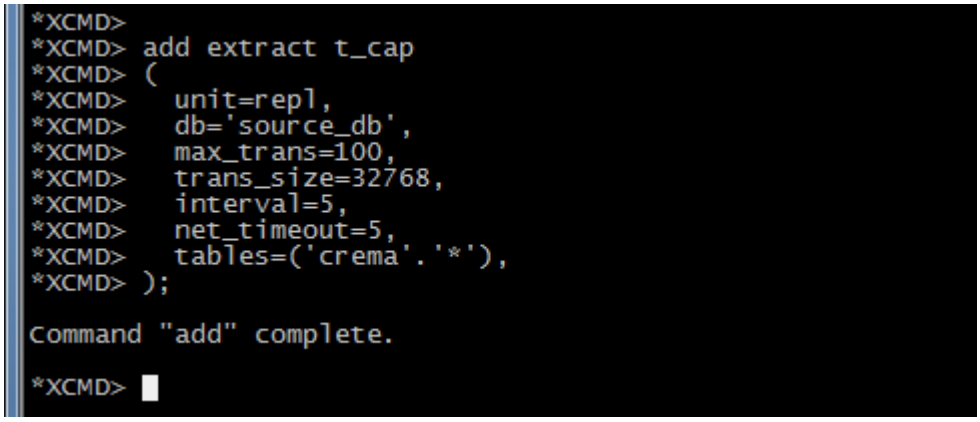
创建抽取组件

抽取组件从 Oracle 日志中捕获变化的数据，解析出需要的内容，每行数据生成一个 LCR，输出到跟踪文件（trail）中。假设抽取组件名称为 t_cap，复制单元名称为 repl，抽取的数据库名称为 source_db，要捕获的数据库用户名称为

crema，添加抽取组件的命令如下：

```
add extract t_cap
(
  unit=repl,
  db=’source_db’,
  max_trans=100,
  trans_size=32768,
  interval=5,
  net_timeout=5,
  tables=(’crema’.’*’),
);
```

上面的命令中 unit 指定了抽取组件所属的复制单元。db 指定了抽取组件分析的数据库名称。max_trans 定义了数据库并发事务的个数。trans_size 定义了每个事务内存的大小，单位是字节。interval 定义了日志中没有新数据时，抽取组件再次检查日志的时间间隔，单位为秒。net_timeout 定义了远程读取日志时，网络超时的时间，单位为秒。tables 定义了抽取组件要捕获的用户名称和表名，’ *’ 表示所有的表，多个表中间用逗号隔开，如果用户名和表名区分大小写，用双引号包括起来。



创建应用组件

应用组件读取上一个组件产生的跟踪文件（trail），然后把事务数据组合成 SQL 语句，装载到目标数据库中。假设应用组件名称为 t_app，复制单元名称为 repl，要装载的目标数据库为 target_db，添加应用组件的命令如下：

```
add apply t_app
(
  unit=’repl’,
  db=’target_db’,
```

```
  prev='t_cap',
  interval=5,
  net_timeout=5,
  tables=('crema'.'*'),
  mapping=(crema.*=jake.*)
);
```

上面的命令中 unit 指定了应用组件所属的复制单元名称。db 指定了装载目标数据库的名称。prev 定义了上一个组件的名称，上一个组件是通过跟踪文件与应用组件相连的组件，上一组件产生一个与自己名称相同的跟踪文件，应用组件根据上一组件的名称来确定读取的跟踪文件名称。interval 定义了跟踪文件中没有新数据时，应用组件再次读取跟踪文件的时间间隔，单位为秒。net_timeout 定义了装载过程中网络超时的时间，单位为秒。tables 定义了要装载的用户名和表名称。mapping 定义了用户名和表名的映射关系，上例中就是把源数据库中的用户 crema 映射到目标数据库中的用户 jake 中。

```
*XCMD>
*XCMD> add apply t_app
*XCMD> (
*XCMD>   unit='repl',
*XCMD>   db='target_db',
*XCMD>   prev='t_cap',
*XCMD>   interval=5,
*XCMD>   net_timeout=5,
*XCMD>   tables=('crema'.'*'),
*XCMD>   mapping=(crema.*=jake.*)
*XCMD> );

Command "add" complete.

*XCMD>
```

保存配置

使用 save config 命令，保存创建的配置内容到文件中。

```
*XCMD>
*XCMD>
*XCMD>
*XCMD> save config;

Command "save" complete.

XCMD>
```

保存配置后，命令行前表示数据变更的星号（*）消失了，说明内存中的数据与配置文件中的一致。

初次同步

软件配置完成，在开始运行之前，首先要保证源端数据库的数据和目标端相关数据是一样的，这样在源端数据库数据变化时，通过软件同步到目标端，才能保证目标端数据与源端一致。为保证首次的数据一致性，需要通过工具把源端数据同步到目标数据库。可利用的工具有 Oracle 自带的实用程序 exp 和 expdp，或者利用第三方工具进行同步。

首先要得到一个没有当前事务的 SCN 号，作为导出数据的基准。

```
select to_char(current_scn) from v$database union select status from v$transaction;
```

如果显示类似下面的结果，表示没有活动事务。

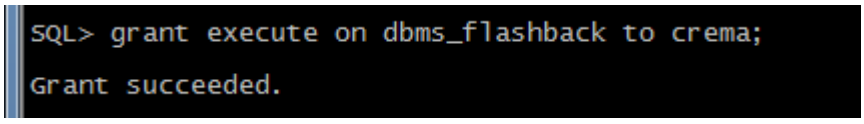
```
TO_CHAR(CURRENT_SCN)
----------------------------------------
3987830
```

如果显示类似下面的结果，表示有活动事务。

```
TO_CHAR(CURRENT_SCN)
----------------------------------------
3987124
ACTIVE
```

假设同步的用户为 crema，密码也为 crema，导出文件名为 crema.dmp。导入用户名为 jake，密码也为 jake，数据导入到用户 jake 下，导出数据和导入数据的命令分别为：

```
grant execute on dbms_flashback to crema;
exp crema/crema owner=crema file=crema.dmp flashback_scn=3987830

imp jake/jake fromuser=crema touser=jake file=crema.dmp
```



```
[oracle@localhost ~]$
[oracle@localhost ~]$ exp crema/crema owner=crema file=crema.dmp flashback_scn=3987830

Export: Release 11.2.0.1.0 - Production on Mon Apr 15 23:17:01 2024

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Producti
on
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Export done in US7ASCII character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user CREMA
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user CREMA
About to export CREMA's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export CREMA's tables via Conventional Path ...
. . exporting table                MY_BAS_INROW_BLOB          6 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                MY_BAS_INROW_CLOB         23 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                MY_BAS_MOUT_CLOB          4 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                MY_BAS_MULTI_LOB          1 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table               MY_BAS_OUTROW_CLOB         11 rows exported
EXP-00091: Exporting questionable statistics.
```

还以上面的用户名为例，使用数据泵导出和导入数据的命令分别为：

```
create directory data_pump_dir_crema as '/home/oracle/dump';

grant read, write on directory data_pump_dir_crema to crema;

expdp crema/crema schemas=crema dumpfile=crema.dmp flashback_scn=3987830

impdp jake/jake schemas=crema remap_schema=jake dumpfile=crema.dmp
```

```
SQL>
SQL> create directory data_pump_dir_crema as '/home/oracle/dump';

Directory created.

SQL> grant read, write on directory data_pump_dir_crema to crema;

Grant succeeded.
```

```
[oracle@localhost ~]$
[oracle@localhost ~]$ expdp crema/crema schemas=crema dumpfile=crema.dmp flashback_scn=3987830

Export: Release 11.2.0.1.0 - Production on Mon Apr 15 23:32:50 2024

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Database Directory Object has defaulted to: "DATA_PUMP_DIR_CREMA".
FLASHBACK automatically enabled to preserve database integrity.
Starting "CREMA"."SYS_EXPORT_SCHEMA_01":  crema/******** schemas=crema dumpfile=crema.dmp flash
back_scn=3987830
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 29.25 MB
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/CLUSTER/CLUSTER
Processing object type SCHEMA_EXPORT/CLUSTER/INDEX
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "CREMA"."MY_TCR"                            7.353 MB     796 rows
. . exported "CREMA"."MY_SEC_INROW_CLOB"                 172.0 KB      14 rows
. . exported "CREMA"."MY_SEC_OUTROW_CLOB"                151.5 KB      14 rows
. . exported "CREMA"."MY_SEC_INROW_BLOB"                 24.49 KB       5 rows
. . exported "CREMA"."MY_BAS_INROW_CLOB"                 137.2 KB      23 rows
```

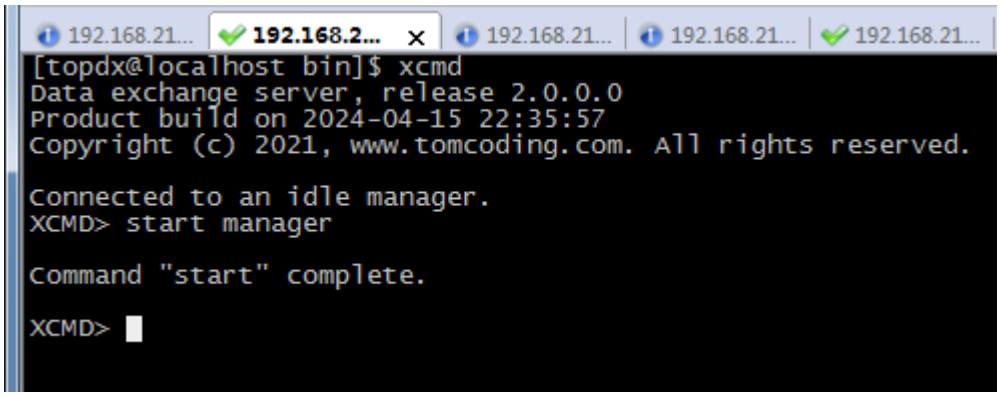
启动软件

xcmd 是 DataEX 软件的命令行控制台，通过 xcmd 输入用户命令，启动软件。

启动 manager

```
start manager
```

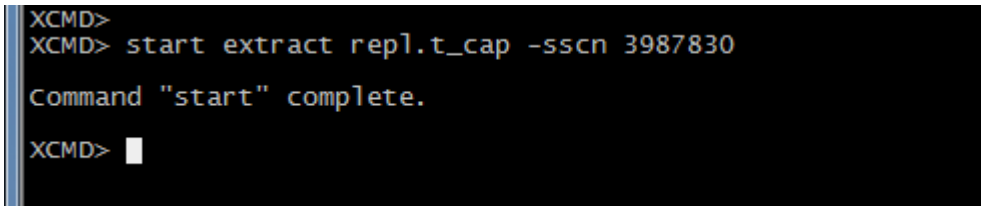
启动管理进程后，系统分配内存，装载配置文件，为运行其他组件准备环境。



启动抽取组件

以前面配置的抽取组件为例，启动命令如下：

start extract repl.t_cap -sscn 3987830



启动应用组件

以前面配置的应用组件为例，启动命令如下：

start apply repl.t_app

如果应用组件已经装载过数据，现在需要从头装载，指定-reset 选项。

start apply repl.t_app -reset

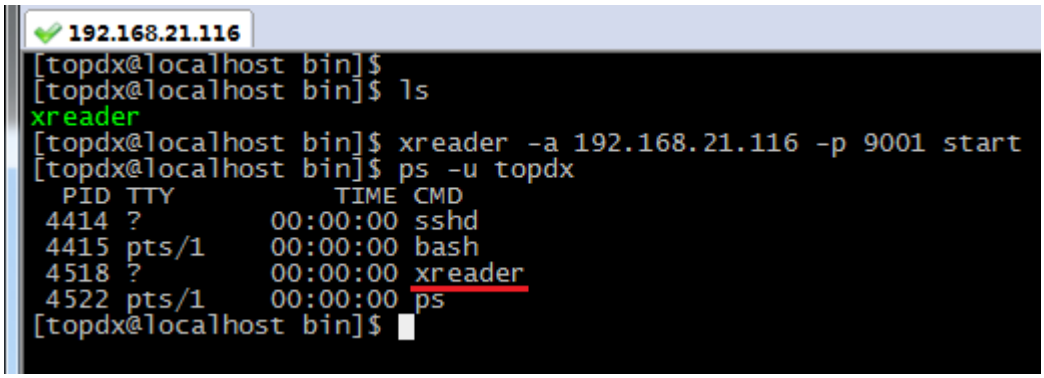
启动读取代理

读取代理程序不在命令行中启动，需要单独启动。以 topdx 用户登录 linux 系统，进入$XREADER_HOME/bin 目录，执行启动命令：

xreader -a 192.168.16.130 -p 9001 start

读取代理默认的端口为 9001，监听所有的 IP 地址，如果使用默认值，启动命令为：

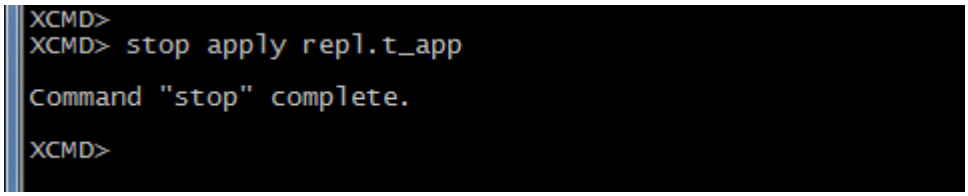
xreader start



停止软件

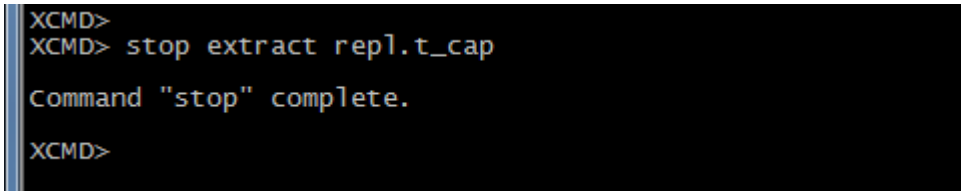
停止应用组件

stop apply repl.t_app



停止抽取组件

stop extract repl.t_cap



停止 manager

stop manager

```
XCMD>
XCMD> stop manager
Stop manager success.

Command "stop" complete.

XCMD>
```

停止读取代理

停止读取代理程序不在命令行中执行，需要单独停止。以 topdx 用户登录 linux，执行命令：

xreader stop