# 1. 整理一下linux下开发需要的工具，并打包

# 2. 准备一台良好运行的linux虚拟机

## 2.1 /tek/Stacks 结构

### 2.1.1 mount

/tek/stacks 是从另外一台机器的/vol/tools/stacks mount过来的：

tekcb-dbe.shpd.tek.com:/vol/tools/stacks on /tek/stacks type nfs (rw,addr=134.64.201.121)

如果要求这台虚拟机可以独立运行，必须把mount的目录以及其下的所有子目录，文件和链接关系完整的拷贝到虚拟机的对应目录上/tek/stacks。

### 2.1.2 /tek/stacks 列表

/tek/stacks下的每一个子目录就是一个stack的版本. Datacast 只依赖ACE的lib，并且ACE的版本在stack库基本不变。所以为了减少空间，我们只需要一个stack版本就可以了，如V6.15.3.0.1b5。

iris@cdc-dc-linux1:/tek/stacks$ ls -ltr

total 52

lrwxrwxrwx 1 1028 3400 5 Jan 19 2011 home -> /home

drwxr-xr-x 7 1028 3400 4096 Jun 21 2013 V6.13.1.1

lrwxrwxrwx 1 1028 3400 9 Jun 25 2013 V6.13.1.1b10 -> V6.13.1.1

drwxr-xr-x 6 1028 3400 4096 Aug 11 17:19 VSLEEPER

drwxr-xr-x 8 1028 3400 4096 Sep 7 02:23 V6.15.3.0.1b4

drwxr-xr-x 8 1028 3400 4096 Sep 23 07:31 V6.15.3.0.1b5

### 2.1.3 in version

每一个stack版本，都有适合不同平台的程序变体，在linux下，我们使用x64-linux-deb-6

iris@cdc-dc-linux1:/tek/stacks/V6.15.3.0.1b5$ ls -l

total 24

drwxr-xr-x 16 1028 3400 4096 Sep 23 04:23 sparc64-sunos-solaris-10

drwxr-xr-x 33 1028 3400 4096 Sep 23 06:56 sparc-sunos-solaris-10

drwxr-xr-x 17 1028 3400 4096 Sep 23 06:41 x64-linux-deb-6

drwxr-xr-x 17 1028 3400 4096 Sep 23 04:38 x86\_64-sunos-solaris-10

drwxr-xr-x 32 1028 3400 4096 Sep 23 05:18 x86-sunos-solaris-10

### 2.1.4 in platform

一般我们需要bin, etc, include, lib, lib64,libexec

iris@cdc-dc-linux1:/tek/stacks/V6.15.3.0.1b5/x64-linux-deb-6$ ls -l

total 120

drwxr-xr-x 2 1028 3400 16384 Sep 23 06:45 bin

drwxr-xr-x 4 1028 3400 4096 Sep 23 06:35 doc

drwxr-xr-x 2 1028 3400 4096 Sep 23 06:45 etc

drwxr-xr-x 24 1028 3400 8192 Sep 23 06:45 include

drwxr-xr-x 2 1028 3400 4096 Sep 23 06:43 info

drwxr-xr-x 6 1028 3400 53248 Sep 23 06:45 lib

drwxr-xr-x 2 1028 3400 4096 Sep 23 06:43 lib64

drwxr-xr-x 3 1028 3400 4096 Sep 23 05:58 libexec

drwxr-xr-x 6 1028 3400 4096 Sep 23 06:33 man

drwxr-xr-x 6 1028 3400 4096 Sep 23 06:38 oracle

drwxr-xr-x 17 1028 3400 4096 Sep 23 06:45 share

drwxr-xr-x 6 1028 3400 4096 Sep 23 06:03 ssl

drwxr-xr-x 2 1028 3400 4096 Sep 23 05:56 var

## 2.2 /sea 结构

### 2.2.1 mount /mnt/sea

shdcnas02.global.tektronix.net:/vol/tools/sea on /mnt/sea type nfs (rw,addr=134.64.201.121)

### 2.2.2 mount /mnt/sea to /sea

/mnt/sea/local/x86-linux-centos-4/sea/local on /sea/local type none (ro,bind)

/mnt/sea/dev/x86-linux-centos-4/sea/dev on /sea/dev type none (ro,bind)

/mnt/sea/compilers/x86-linux-centos-4/sea/compilers on /sea/compilers type none (ro,bind)

### 2.2.3 /sea/dev

iris@cdc-dc-linux1:/sea/dev$ ls -la

total 16

drwxr-xr-x 4 1028 3400 4096 Apr 7 2011 .

drwxr-xr-x 5 root root 4096 Dec 31 2014 ..

lrwxrwxrwx 1 1028 3400 1 Feb 12 2010 1.0 -> .

lrwxrwxrwx 1 1028 3400 20 Jan 15 2011 bin -> .versions/LATEST/bin

lrwxrwxrwx 1 1028 3400 1 Feb 12 2010 dbs -> .

lrwxrwxrwx 1 1028 3400 20 Jan 15 2011 etc -> .versions/LATEST/etc

lrwxrwxrwx 1 1028 3400 1 Feb 12 2010 imake-cpp -> .

drwxr-xr-x 2 1028 3400 4096 Apr 8 2011 legacy

lrwxrwxrwx 1 1028 3400 1 Feb 12 2010 levelize -> .

lrwxrwxrwx 1 1028 3400 1 Feb 12 2010 ual -> .

drwxr-xr-x 4 1028 3400 4096 Oct 18 07:09 .versions

## 2.3 建立交叉编译环境

### 2.3.1 建立/tek/stacks和/sea

由于虚拟机需要独立于实验室环境运行，因此我们不能通过mount方式来获取这两个目录，因为mount的target机器在实验室环境中，如果虚拟机运行在家庭环境下就不可能mount到这些目录。

所以，我们需要从真实的linux机器上scp 这两个文件夹上关于linux的部分的所有文件，并要保持它们的目录结构和链接（link）结构。

Linux下的文件结构如下：



为此，需要准备两个脚本程序。

一个运行在real linux机器上，用于读取/tek/stacks和/sea的目录和文件的结构，包括link关系，并保存到一个或多个txt文件里面。

另一个运行在virtual linux机器上，用于根据读取的txt文件在virtual linux上创建相应的目录结构，从real linux上scp所有实体文件存放于相应目录，并在这些目录下建立一样的link关系。

#### 2.3.1.1 读取文件夹的结构信息的脚本

**方式1：**

将所有信息放于一个文件中，

用 D dir\_name [path] {} 表示一个文件夹，文件夹内的信息都放于{}内。其中path可以不存在，事实上只有top文件夹需要path

用 L link\_name target\_name表示一个链接（link）

用 F file\_name表示一个实体文件，例如/usr/lib有如下文件：

iris@cdc-dc-linux1:/usr/lib$ls –l

drwxr-xr-x 2 root root 12288 Sep 18 05:24 ldscripts

-rwxr-xr-x 3 root root 4096 Dec 31 2014 libaccount-plugin-1.0

lrwxrwxrwx 1 root root 30 Mar 15 2014 libaccount-plugin-1.0.so.0 -> libaccount-plugin-1.0.so.0.0.0

-rw-r--r-- 1 root root 48416 Mar 15 2014 libaccount-plugin-1.0.so.0.0.0

可以转化为如下结构

D lib /usr

{

D ldscripts

{

}

F libaccount-plugin-1.0

L libaccount-plugin-1.0.so.0 libaccount-plugin-1.0.so.0.0.0

F libaccount-plugin-1.0.so.0.0.0

}

**方式2：**

对每个文件夹分别处理，

用一个文件来存放本文件夹下所有的所有链接（link）信息，暂时将文件取名为linkInfo.txt, 其内部信息包括link\_name 和target\_name，每一条信息占用一行

用一个文件来存放本文件夹下所有的所有实体文件信息，暂时将文件取名为realFileInfo.txt, 其内部信息包括file\_name，每一条信息占用一行

对每一个文件夹，在其父文件夹中创建一个同名的文件夹，将linkInfo.txt和realFileInfo.txt文件放入其中。对于top文件夹，创建文件夹的时候要更改文件夹的名称，将path信息也放入文件夹名称之中，可以简单将path中的”/” 替换为‘’\_”，如/tek/stacks可以更改为\_tek\_stacks

**比较：**

方式1 结构比较简单，但是用脚本获取这些信息会比较麻烦点

方式2结构比较复杂，但是用脚本获取这些信息会比较简单

#### 2.3.1.2 在目标机器上创建一样的文件结构的脚本

**方式1：**

**方式2：**

读取top文件夹名称，并将它转化为path+dirName的格式，可以简单将path中的‘’\_”替换为”/”,转换后的文件名称取名为topdirName

创建top目录，通过命令mkdir –p topdirName

登陆到此目录，通过命令cd topdirName

遍历topdir下的realFileInfo.txt文件的每一行，

Begin

Scp 远程机器的这个文件到本地 （scp keyu@remoteIP: topdirName/file\_name）

End

遍历topdir下的linkInfo.txt文件的每一行，

Begin

创建链接 （ln –s target\_name link\_name）

End

遍历topdir下的所有子目录，

Begin

执行类似于topdir的操作

End

### 2.3.2 设定PATH环境变量

建立.x86env.sh用于设置运行的环境变量，并在.bash\_profile中调用这个脚本。

将如下行加入文件.x86env.sh

#################################################################

### PATH , will be updated later . take care the order

#################################################################

PATH=.

for modulename in /sea/local/bin /sea/dev/bin /usr/local/bin /bin /usr/bin /sbin /usr/sbin /usr/ucb /usr/openwin/bin /usr/dt/bin /etc /usr/etc /usr/ccs/bin /opt/sfw/bin /sea/compilers/native/gcc-4.4.5/V1.3/bin

do

[ -d ${modulename} ] && PATH=${PATH}:${modulename}

done

export PATH

############################################################################

## PATH

############################################################################

#export GCCCHAIN=`cd ${DATACAST\_HOME}/src && ( make info | grep COMPILER\_AREA | cut -d : -f 2-)`

export GCCCHAIN=

PATH=${PATH}:${GCCCHAIN}/bin:${ORACLE\_HOME}/bin

#PATH=${PATH}:${BIA\_HOME}/bin64

PATH=${PATH}:${IRISGEN\_HOME}/install/${PLATFORM\_TYPE}/bin64

PATH=${PATH}:${PLATFORM\_HOME}/install/${PLATFORM\_TYPE}/bin64

PATH=${PATH}:${PLATFORMDR\_HOME}/install/${PLATFORM\_TYPE}/bin64

PATH=${PATH}:${DATACAST\_HOME}/install/${PLATFORM\_TYPE}/bin64

#PATH=${PATH}:${IRISGEN\_HOME}/bin

#PATH=${PATH}:${PLATFORM\_HOME}/bin64

#PATH=${PATH}:${PLATFORMDR\_HOME}/bin64

#PATH=${PATH}:${DATACAST\_HOME}/bin64

export PATH

### 2.3.3 设定LD\_LIBRARY\_PATH环境变量

#################################################################

### LD\_LIBRARY\_PATH, will be updated later . take care the order

#################################################################

LD\_LIBRARY\_PATH=.

for modulename in /sea/local/lib /sea/dev/lib /usr/local/lib /lib /usr/lib /libx32 /usr/lib32 /usr/libx32 /usr/ucb /usr/openwin/bin /usr/dt/bin /etc /usr/etc /usr/ccs/bin /opt/sfw/bin /sea/compilers/native/gcc-4.4.5/V1.3/bin

do

[ -d ${modulename} ] && LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${modulename}

done

export LD\_LIBRARY\_PATH

####################################################################################

# LD\_LIBRARY\_PATH

####################################################################################

DC\_STACK\_VERSION=V6.15.3.0.1b5

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/tek/stacks/${DC\_STACK\_VERSION}/${PLATFORM\_TYPE}/lib

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${IRISGEN\_HOME}/install/${PLATFORM\_TYPE}/lib64

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${PLATFORM\_HOME}/install/${PLATFORM\_TYPE}/lib64

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${PLATFORMDR\_HOME}/install/${PLATFORM\_TYPE}/lib64

LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${DATACAST\_HOME}/install/${PLATFORM\_TYPE}/lib64

#DC\_STACK\_VERSION=V6.13.2.2.1

#LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/tek/stacks/V6.13.2.2.1/${PLATFORM\_TYPE}/lib

#LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${IRISGEN\_HOME}/lib

#LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${PLATFORM\_HOME}/lib64

#LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${PLATFORMDR\_HOME}/lib64

#LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:${DATACAST\_HOME}/lib64

export LD\_LIBRARY\_PATH

### 2.3.4 设定其他环境变量

#################################################################

####################################################################################

######You must modify the following variable

####################################################################################

export CVSROOT=:pserver:134.64.2.102:/home/spiuser/cvsRepository

PLATFORM\_TYPE=x64-linux-deb-6

export BIA\_HOME=${HOME}/gitdev

####################################################################################

#activate\_dcast\_preproc\_all need the following declare

####################################################################################

export JAVA\_HOME=/home/iris/datacast/deploy/jdk

export BIA\_SERVER\_HOME=${HOME}/server ###serverTrunk

export PROCESS\_MONITOR\_HOME=${BIA\_SERVER\_HOME}/pm ##/export0/home/bia/ngbase/pm1

export NGBASE\_PM\_HOME=${BIA\_SERVER\_HOME}/pm #/export0/home/bia/ngbase/pm1

####################################################################################

### BIA

####################################################################################

export DATACAST\_HOME=${BIA\_HOME}/datacast

export IRISGEN\_HOME=${BIA\_HOME}/irisgen

export PLATFORMDR\_HOME=${BIA\_HOME}/platform\_drutils

export PLATFORM\_HOME=${BIA\_HOME}/platform

#export DRGEN\_HOME=${BIA\_HOME}/xdrGen

# CVS

#################################################################

export CVS\_RSH=ssh

export CTIME\_ROOT=/opt/ctime/

###################################################################################

# alias

###################################################################################

. ./.setAlias.sh

### 2.3.5 建立自定义命令

.setAlias.sh文件

## aliases for all shells

alias psg='ps -ef | grep \!\* | grep -v grep'

alias ll='ls -l'

alias la='ls -la'

alias lg='ls -la | grep -i'

alias his='history'

alias psme='ps -ef | grep -v grep | grep $USER'

alias psg='ps -ef | grep -v grep | grep -i'

alias gdb='/tek/stacks/V6.13.2.2.1/x86\_64-sunos-solaris-10/bin/gdb'

alias xps='/usr/ucb/ps awwx'

alias pssw='/usr/ucb/ps awwx | grep -v grep | grep -i'

## Set up some convienient aliases.

alias ff='find . -name \!\* -print '

alias fcc='find . -name "\*.cc" -print | xargs grep -i '

alias fh='find . -name "\*.h" -print | xargs grep -i '

alias cls='clear'

alias fstrcc='find . -name "\*.cc" -exec grep -l \!\* {} \;'

alias fstrh='find . -name "\*.h" -exec grep -l \!\* {} \;'

alias fstr='find . -name "\*.\*" -exec grep -l \!\* {} \;'

## cd dir

alias gen='cd ${IRISGEN\_HOME}/etc'

alias site='cd ${BIA\_SERVER\_HOME}/site'

alias bia='cd ${BIA\_HOME}'

alias dat='cd ${DATACAST\_HOME}'

alias pla='cd ${PLATFORM\_HOME}'

alias pdr='cd ${PLATFORMDR\_HOME}'

alias iris='cd ${IRISGEN\_HOME}'

##compile

alias make1='cd ${PLATFORM\_HOME} ; mkmf; make -j 32 '

alias make2='cd ${PLATFORMDR\_HOME} ; mkmf; make -j 32 '

alias make3='cd ${DATACAST\_HOME} ; mkmf; make -j 32 '

alias make4='cd ${IRISGEN\_HOME} ; mkmf; make -j 32 '

##run

alias run1='cd ${BIA\_SERVER\_HOME}/site;it7\_process -config dp\_g10.keyu.xml -V -debug'

alias run2='cd ${BIA\_SERVER\_HOME}/site;drRcvr -xdr\_port 0 -idr\_port 0 -hdr\_port 19200 -dump yes'

alias run3='cd ${IRISGEN\_HOME}/etc; ../bin/irisGen -f sip\_Invite.trf -p 32100'

### 2.3.6 建立server目录

buildServerEnv

此函数定义在.x86env.sh文件中，其定义如下

function buildServerEnv()

{

mkdir -p ${BIA\_SERVER\_HOME}/site/format\_dir

mkdir ${BIA\_SERVER\_HOME}/bin

mkdir ${BIA\_SERVER\_HOME}/etc

mkdir ${BIA\_SERVER\_HOME}/logs

mkdir ${BIA\_SERVER\_HOME}/pm

chmod 777 -R ${BIA\_SERVER\_HOME}

cp -r ${DATACAST\_HOME}/etc/\* ${BIA\_SERVER\_HOME}/etc

cp -r ${PLATFORMDR\_HOME}/etc/\* ${BIA\_SERVER\_HOME}/etc

cp -r ${PLATFORM\_HOME}/etc/\* ${BIA\_SERVER\_HOME}/etc

cd ${BIA\_SERVER\_HOME}/etc/iris-uacn

./activate\_dcast\_preproc\_all standard

}

### 2.3.7 下载代码

Step 1.首先创建public key

ssh-keygen -t rsa -C "kevin.yu@tekcomms.com"

Step 2.然后拷贝的内容

cat ../.ssh/id\_rsa.pub

Step3并将它添加到stash的SSH keys

stash->view profile->manage account->SSH keys->add key

Step 4 下载code

downloadDCCodes

此函数定义在.x86env.sh文件中，其定义如下

function downloadDCCodes()

{

cd ${PLATFORM\_HOME}

git clone ssh://git@134.64.2.102:7999/~keyu/platform.git

cp ${PLATFORM\_HOME}/customize.imake.example ${PLATFORM\_HOME}/imake-config/customize.imake

cd ${PLATFORMDR\_HOME}

git clone ssh://git@134.64.2.102:7999/~keyu/platform\_drutils.git

cp ${PLATFORMDR\_HOME}/customize.imake.example ${PLATFORMDR\_HOME}/imake-config/customize.imake

cd ${DATACAST\_HOME}

git clone ssh://git@134.64.2.102:7999/~keyu/datacast.git

cp ${DATACAST\_HOME}/customize.imake.example ${DATACAST\_HOME}/imake-config/customize.imake

cd ${IRISGEN\_HOME}

git clone ssh://git@134.64.2.102:7999/~keyu/irisgen.git

cp ${IRISGEN\_HOME}/customize.imake.example ${IRISGEN\_HOME}/imake-config/customize.imake

##git clone ssh://git@134.64.2.102:7999/~keyu/probe.git

}

Step 5 build

alias make1='cd ${PLATFORM\_HOME} ; mkmf; make -j 32 '

alias make2='cd ${PLATFORMDR\_HOME} ; mkmf; make -j 32 '

alias make3='cd ${DATACAST\_HOME} ; mkmf; make -j 32 '

alias make4='cd ${IRISGEN\_HOME} ; mkmf; make -j 32 '

### 2.3.8 运行程序

alias run1='cd ${BIA\_SERVER\_HOME}/site;it7\_process -V -debug -config dp\_g10.xml'

alias run2h='cd ${HOME}/dr;drRcvr -xdr\_port 0 -idr\_port 0 -dump yes -output\_dir . -write\_binary yes -hdr\_port '

alias run2i='cd ${HOME}/dr;drRcvr -xdr\_port 0 -hdr\_port 0 -dump yes -output\_dir . -write\_binary yes -idr\_port '

alias run2x='cd ${HOME}/dr;drRcvr -hdr\_port 0 -idr\_port 0 -dump yes -output\_dir . -write\_binary yes -xdr\_port '

alias run3='cd ${IRISGEN\_HOME}/etc; irisGen -p 17100 -f '

### 2.3.9 自动化测试 (DaRTS)

### 2.3.10 自动化编译 （JENKINS）

# 3. tek third party stack 代码下载和原理

## 3.1 irisStacks

### Install path

remote-master.rich.tek.com /uadev/iris\_stack/LATEST-7.16.1.0/install/x64-linux-centos-5

### Code

<http://tekcomms-stash:7990/projects/IAO/repos/irisstack/>

### irisStacks tarball

<http://tekcomms-artifactory.global.tektronix.net:8081/artifactory/webapp/#/artifacts/browse/tree/General/platform-local/3rdParty-src/protobuf-2.3.0.tar.bz2>

### build steps

1. download the code.

2. download the tarballs from above link, put them at tarballs directory

3. modify Imakefile if need

4.mkmf

5.make

6. make packages

## 3.2 tek stacks

### Install path

remote-master.rich.tek.com /tek/stacks/V6.16.1.0.1b2/x64-linux-deb-6/

### Code

### <http://tekcomms-stash:7990/projects/EXT/repos/stacks/browse>

### build steps

1. download the code.

git clone --depth=1 ssh://git@134.64.2.102:7999/~keyu/stacks.git [--depth=1 means just download the top level version code]

2. modify Imakefile if need

3.mkmf

4.make

5. make packages

## 3.3 build one open source code project-protobuf

take protobuf as an example

### 1. get the source code

### 2. untar

Tar –xzvf protobuf-2.3.0.tar.gz

### 3. config

Cd protobuf-2.3.0

./configure --prefix=/export0/home/keyu/self/GPB/protoDir/

### 4.make

It report an error: “./google/protobuf/io/coded\_stream.h:1047: error: 'INT\_MAX' was not declared in this scope”

Change the file of google/protobuf/io/coded\_stream.h, like this:

+

+#ifdef \_\_sun\_\_

+#include <limits.h>

+#endif

+

### 5. make patch

If some error found, we must correct it and then make a patch file.

Diff -rNu sourceDir destDir

### 6. patch the changing

Patch –p1 < ../protobuf-2.3.0.patch

Make again, it passed.

### 7. make install

Make install

It will put the output to directory /export0/home/keyu/self/GPB/protoDir/

### 8. use built files

Put the file in install directory to /tek/stack in certain place.

If /tek/stack is mounted from other place, copy it to some place writable.

-bash-3.2$ cp /tek/stack/ V6.16.1.0.1b5/x86-sunos-solaris-10 /export0/home/keyu/tekStacks

And change the variable $STACK\_ROOT\_DIR in imake-config/xxx-rules.imake, xxx could be uaplat, datacast or drutils.

Take drutils-rules.imake as an example:

//STACK\_ROOT\_DIR ?= /tek/stacks/$(STACK\_VERSION)

STACK\_ROOT\_DIR ?= /export0/home/keyu/tekStacks/$(STACK\_VERSION)

-bash-3.2$ cd /export0/home/keyu/tekStacks/V6.16.1.0.1b5/x86-sunos-solaris-10

-bash-3.2$ mv bin/protoc protoc.bak/bin

-bash-3.2$ cp ~/self/GPB/protoDir/bin/protoc bin

-bash-3.2$ mv include/google protoc.bak/include

-bash-3.2$ cp ~/self/GPB/protoDir/ include/google include

-bash-3.2$ mv lib/libproto\* protoc.bak/lib

-bash-3.2$ cp ~/self/GPB/protoDir/ lib/ libproto\* lib

## 3.4 build tek stacks in pure linux

If you want to make stacks in local machine, please make below changings:

1) add "#define EnableNotBuildInDBS YES" to imake-config/customize.imake, it tell the compiler to make in local

2) make the below changing to build/scripts/build-package, it will skip the check of NATIVE\_ARCH

#----------------------------------------------------------------------

# Get the architecture of the current machine.

#----------------------------------------------------------------------

$main::NATIVE\_ARCH = `/sea/local/bin/arch-name`;

+if ($main::NATIVE\_ARCH =~ /x64-linux-\*/)

+ {

+ $main::NATIVE\_ARCH = "x64-linux-centos-5";

+ }

chomp($main::NATIVE\_ARCH);

# 4. open resource code learn

## 4.1 zookeeper

## 4.2 memory to disk

06/28/2016:11:39:37, bia, DataCast\_G10, INFO, (105) MapQosRepository: Save the MapRepository to disk for 0 entry in 1 msec (file=

/export0/home/iris/server/logs/dp\_g10\_mappingtable).