Explorer Controller Database Considerations

Overview

Introduction

The origin of this document was the following question from a customer:

"Is there a way an operator (or Cisco TAC person) can track or do forensics on the type of SQL being run and at what periodicity by inspecting processes or logs (or additional processes/logs) on the DNCS or EC?"

References

Reference Number	Document
[Ref-1]	Database Defragmentation, Cisco Systems, Inc., August 31, 2015.
[Ref-2]	*Installing and Operating the Replicated Database Package on the DNCS Installation Guide* (part number 78-4034803-01)

Purpose

This document articulates a response to the customer's question for what it would take to evaluate type and frequency of SQL queries made by third-party servers made into the EC/DNCS.

Scope

This document covers the Explorer Controller (EC) under the following System Releases (SRs):

- 4.2.x
- 5.0
- 6.0
- 7.0
- 8.0

Accessing the root and dncs User Accounts

Important:

- Role-Based Access Control is no longer supported in SR 8.0. Please follow the steps below to switch between different user accounts.
- The ecadmin user is used in examples for all Cisco DBDS documents pertaining to EC 8.0.
- Commands run as **root** user are shown with a **#** symbol.

  Example:

  ```
  [root@vodwater ~]#
  ```
- Commands run as a **admin**, **dncs**, or any **Administrator** user are shown with a **$** symbol.

  Example:

  ```
  [admin@vodwater ~]$
  [ecadmin@vodwater ~]$
  [dncs@vodwater ~]$
  ```

Once the application installation is complete, you can only log in with the admin or an Administrator user account.

The ***admin*** account is created by default during the installation, and is granted privileges to access the root user account, as root login is not permitted. These privileges allow the admin user to execute root commands by preceding the command with "sudo". For example, if you want to modify a network configuration file, the command will resemble the following:

Command Example: Executing a root command as Admin user:

```
[admin@vodwater ~]$ sudo vi /etc/sysconfig/network-
scripts/ifcfg-eth0
```

As admin user, you can also change to the root user account by entering the following command.

Important: For any procedure in this guide that states "As root user", you must be logged into a terminal window as admin user and switch to the root user.

Command Syntax: Changing to root user:

```
[admin@vodwater ~]$ sudo –i
```

Any ***Administrator*** account that you create using the useradmin script (see the next section) has privileges to log into the from a terminal window. Administrator accounts do not have privileges to access the root user account, but should be used to access the dncs user account.

Important: Do not access the dncs user account using the root user account.

To switch to the dncs user, type the following command from the terminal window where you are logged in as an Administrative user.

Important: For any procedure that states "As dncs user", you need to execute this command from the terminal window where you are logged in with your Administrator account.

Command Syntax: Changing to the dncs user:

```
[ecadmin@vodwater ~]$ sudo su – dncs
```

Note: Throughout all Cisco DBDS documentation, the **ecadmin** user is used as an example.

Overview:

Terminal Window Logged in as:	Use Account to change to:	Command to execute:
admin	root	sudo -i
[Administrator] **Example:** ecadmin	dncs	sudo su - dncs

Audience

This document is written for system operators. Our engineers may also find this document to be useful.

Document Version

This is the first formal release of this document.

In This Document

Monitoring EC 6.0/7.0 SQL Database Activity

Complications

There are several complications associated with third-party servers accessing the EC database in production that can best be explained by using the newspaper adage: who, what, when, where, and why:

- **Who:** The identity of the third-party servers is NOT known.
- **What:** The information that the third-parties need is NOT known.
- **When:** (periodicity) How often the information is needed is NOT known.
- **Where:** We do know that the target is the EC/DNCS database.
- **Why:** (how used) What the information is being used for is NOT known.

Tracking database activity on the EC requires manual labor. Since this activity is fundamentally outside the EC/DNCS application, there are no operational indicators (for example, logs, SNMP traps, etc.) generated by the EC/DNCS application processes that provide any usable artifacts of this activity.

Additionally, the EC/DNCS platform is NOT designed to solicit or audit external interactions with the EC/DNCS platform that is NOT interacting with EC/DNCS application layer components. This leaves the EC/DNCS resident Informix database engine, and any associated utilities, that can be used to address this issue.

Therefore, what's left is an approach for auditing database (DB) activity. The idea is to audit the DB periodically hoping to catch the database activity at the moment it occurs.

Onstat for the DB Session

The EC-resident Informix engine provides a utility called *onstat* that accepts a parameter called ses (for example, `onstat -g ses)` that displays specific details about the database session and its associated processes (for example, process IDs, PIDs) running against the database.

For example, the following output generated by the `onstat -g ses` command demonstrates the set of current database sessions executing against the database:

```
root@dudley# onstat -g ses


Informix Dynamic Server 2000 Version 9.21.UC3     -- On-Line -- Up 740 days
22:19:58 -- 1356800 Kbytes


session                                        #RSAM    total      used
id       user     tty      pid      hostname  threads  memory     memory
4580497  informix -        0        -         0        12288      7632
4558260  informix -        0        -         0        12288      7632
3651475  root     -        23796    dudley    1        69632      67832
3651474  root     -        23796    dudley    1        61440      57200
3651473  root     -        23796    dudley    1        73728      67560
3650789  dncs     -        4253     dudley    1        53248      47408
3650768  dncs     -        4253     dudley    1        77824      60032
5238     dncs     -        19815    dudley    1        151552     139872
1984     dncs     -        18713    dudley    1        57344      51896
1977     dncs     -        19369    dudley    1        32768      29232
1957     dncs     -        20892    dudley    1        159744     152536
1955     dncs     -        20321    dudley    1        77824      67680
```

Figure 1: Sample output, "onstat –g ses" command, all DB sessions

Note: Although the output above was retrieved from an Informix version 9.21, the onstat command and output is relevant to later versions as well (for example, Informix version 11.7).

It highlights the following information:

- The session ID against the DB
- The process ID (pid) of the process executing the given database query
- The user ID of the user who instituted the session
- The hostname of the system the user is from

Onstat for the DB Session and sess_number

From this information, a lower level of detail can be derived for a specific user.

For example, to acquire the details of which SQL statement is being executed against the Informix engine by session ID 1955 for PID 20321, we enter the command `onstat –g ses sess_number` as follows:

```
root@dudley# onstat -g ses 1955


Informix Dynamic Server 2000 Version 9.21.UC3     -- On-Line -- Up 740 days
22:26:50 -- 1356800 Kbytes


session                                        #RSAM    total      used
id       user     tty      pid      hostname  threads  memory     memory
1955     dncs     -        20321    dudley    1        77824      67680


tid      name     rstcb    flags    curstk   status
1998     sqlexec  50129948 Y--P---  1648     50129948 cond wait(netnorm)


Memory pools    count 1
name         class addr     totalsize freesize #allocfrag #freefrag
1955         V     51a56020 77824     10144    145        15


name           free       used           name           free       used
overhead       0          1648           scb            0          96
opentable      0          5184           filetable      0          912
log            0          2152           temprec        0          6216
keys           0          160            ralloc         0          29248
gentcb         0          1240           ostcb          0          2520
sqscb          0          11992          sql            0          40
rdahead        0          448            hashfiletab    0          280
osenv          0          1904           sqtcb          0          2680
fragman        0          960


Sess  SQL            Current            Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database           Lvl Mode       ERR  ERR  Vers
1955  SELECT         dncsdb             CR  Wait 300   0    0    9.03


Current statement name : c_byoidval


Current SQL statement :
  SELECT oidval, certificatedata, datalen, distinguishedname, optimctrl FROM
    pdkeycertificate WHERE pdkeycertificate.oidval = ? ;


Last parsed SQL statement :
  SELECT oidval, certificatedata, datalen, distinguishedname, optimctrl FROM
    pdkeycertificate WHERE pdkeycertificate.oidval = ? ;
```

Figure 2: Sample output, “onstat –g ses” command, specific DB session

Safely Accessing the Database (Playing Nice with Others)

One *best practice* is knowing the proper method to use for accessing the database to not adversely affect the core EC/DNCS application processing.

To that end, the default database lock employed when a process accesses the database is usually greater than *dirty read*. If so, then other EC/DNCS processes must WAIT for the offending access routine to complete its work against the database engine before they can continue the core EC/DNCS work that was interrupted by the non-EC/DNCS routine.

By contrast, when an application sets the database isolation level to *dirty read* as part of its entrance parameters into the Informix engine, the program never waits and never makes another program wait.

Therefore, Cisco's recommendation is to apply the *dirty read* isolation level, especially when accessing large tables such as the hct_profile, or secure_micro, or emm tables. The syntax for the proper method to set the isolation level to *dirty read* is shown in the following example:

```
dbaccess dncsdb -  << EOF
set isolation to dirty read;
select *
 from sm_pkg_auth
 where sm_serial_num='00:19:47:8B:C1:6D'
EOF
```

The connection established by the dbaccess utility against the database retains the isolation level set to *dirty read* for the duration of the associated database connection. *And* select statements executed do not lock any resources, including when accessing large tables, as previously mentioned.

Database Feature Evolution

Predicated on the fact that the platform for EC-resident services is a CISCO Unified Computing System (UCS) C240M3, the EC-specific database allocates up to 200 Gigabytes (GB) of space for data for the traditional database sizing stratum, dataspace1 is in scope.

Additionally, there are 12 GB allocated for the rootdbs, tempspace1, and logspace constructs.

Following this line of reasoning, and extrapolating capacity to house set-top box (STB) devices based on them, the following table shows sizing comparisons between previous releases of the system and the EC:

Criteria	SR 4.2	SR 6.0/7.0
Database Sizing	Total: 62 GB EMM tbl: 33M rows	Total: 200 GB EMM tbl: 128M rows (4X)
DB hardening	Power outage usually requires recovery from back up media	~1000 incremental Database layer bug fixes Power outage scenario does not require restore from backup media
DB defragmentation (defrag)	Full database basis	Per table basis
Space reconfiguration	Manual, static re-configuration	Auto-grow to 200 GB as needed
Platform roadmap	V890/V880: Hardware end-of-service (EOS) DB version, end of support SUN/SPARC/Solaris Vendor lock-in	UCS Virtualized (VMWare) for hardware abstraction Solaris x86 Supported DB version Evolving to vendor agnostic

Table 1: Database feature evolution

This leads to support for the following mix of unstaged versus staged device volume capacity:

Unstaged	Staged
33,600,000	0
30,220,800	1,000,000
28,531,200	1,500,000
26,841,600	2,000,000
25,152,000	2,500,000

23,462,400	3,000,000
21,772,800	3,500,000
20,083,200	4,000,000
18,393,600	4,500,000
16,704,000	5,000,000
15,014,400	5,500,000
13,324,800	6,000,000
11,635,200	6,500,000
9,945,600	7,000,000
8,256,000	7,500,000
6,566,400	8,000,000
4,876,800	8,500,000
3,187,200	9,000,000
1,497,600	9,500,000
0	10,000,000

Table 2: Unstaged versus staged device volume capacity

Replicated Database Best Practices (per Ref-2)

Data replication allows a copy of the database from a primary server to be maintained on a secondary server. When activated, the primary database server continuously replicates data between itself and the secondary server by sending copies of the logical-log transactions to the secondary database server.

For filesystem level synchronization, the *rsync* utility allows a copy of file systems from a primary server to be maintained on a secondary server. When activated, the rsync utility periodically synchronizes files and directories from the primary server to the secondary server.

The Replicated Database (RepDB) package consists of the following components:

- Data Replication
- Remote File Copying

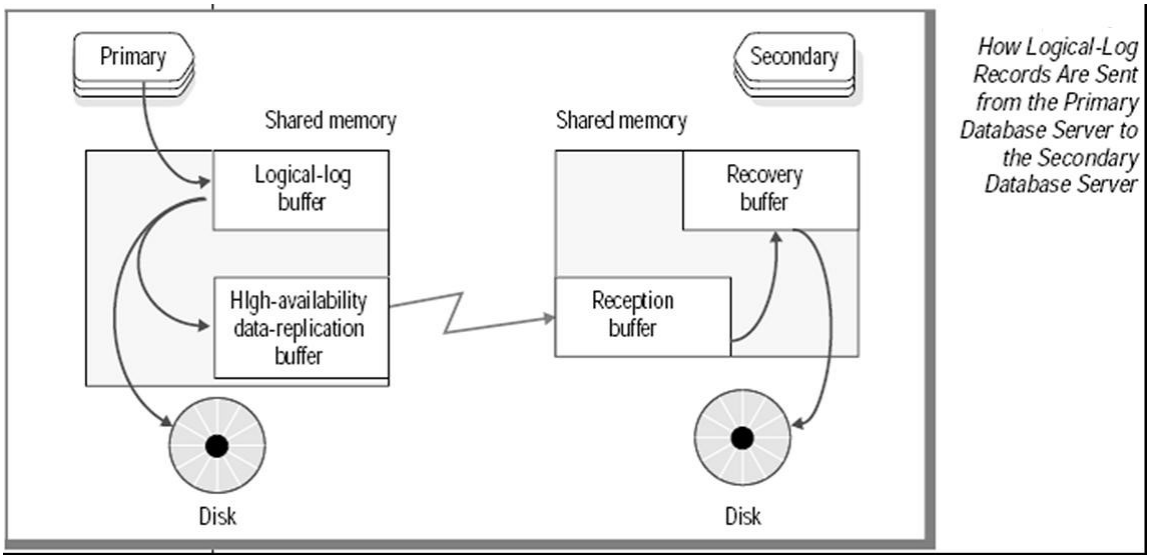

Figure 3: Logical log record process

RepDB Pros and Cons

The following table summarizes the pros and the cons of the RepDB approach.

Pros (Advantages)	Cons (Disadvantages)
Service-impacting events are reduced on the primary server by allowing third-party database query tools to access the secondary database server.	The Replicated Database is read-only. The Replicated Database cannot be used for tape or disk backups because a database backup is considered a write process.

The secondary server provides a flexible platform for developing new tools. Furthermore, if the secondary server has access to the Digital Broadband Delivery System (DBDS), the secondary server can be used by third-party tools that require both database and network access.	There is a minor time delay between changes made to the primary database and those changes being reflected on the secondary server.
The secondary server, under certain configurations, can be converted to the primary server, if needed.	Automatic failover — the ability to re-route users and applications to the Replicated Database with minimal interruption — is not supported. Failover requires manual intervention.
The Replicated Database can be used as the core for future development of high-availability products.	Regular backups of the primary database server are still required. Database corruption in the primary server, if it occurs, will be copied to the secondary server if the Replicated Database is active.

Table 3: Pros and cons of RepDB

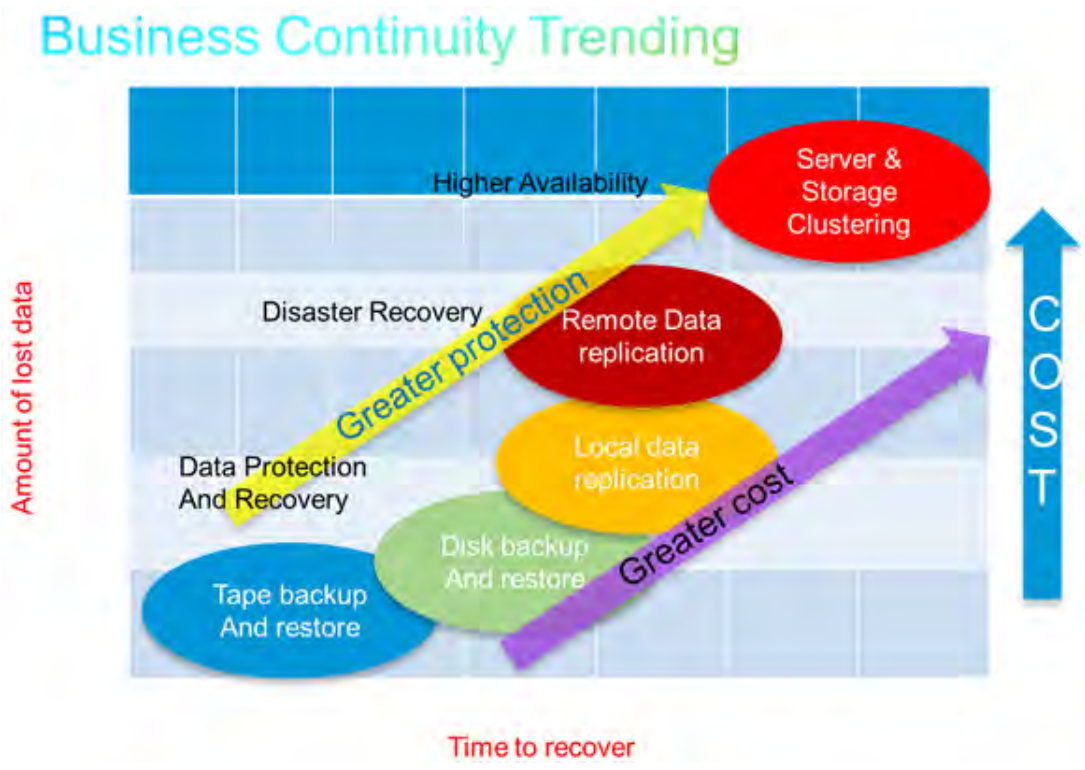

Figure 4: Business continuity trending

RepDB Perequisites

Make sure that you meet the following requirements:

Prerequisite	Description
Identical software installs	Make sure that the primary server and the secondary server have identical software. If the secondary server is to act as a failover for the primary server, then both servers must have identical hardware and software configurations. There are two ways you can make sure that the software is identical: ■ Perform a fresh install on both servers, using identical software settings. ■ Back up the primary server file system and restore it to the secondary server. **Important:** If you choose to use the backup and restore method, and if the primary server has been previously configured for the Replicated Database, make sure that you remove the Replicated Database configuration from the primary server by deconfiguring the Replicated Database before backing up the file system.
Directory considerations	The `/usr/local/backup_restore` directory MUST exist on both servers. If the `/usr/local/backup_restore` directory does not exist on both servers, insert the DNCS installation CD into the CD drive of both servers and copy the `/cdrom/cdrom0/backup_restore` directory to `/usr/local/backup_restore`. **Example:** `cp -R /cdrom/cdrom0/backup_restore /usr/local`
Config requirements	Make sure that you have obtained all configuration information for both servers. Specifically: ■ Hostname for the network interfaces used by each one of the servers when communicating with each other as the primary, and secondary systems. ■ The unique IP address for the primary and secondary. ■ Netmask. ■ Gateway. ■ -Existing network interfaces other than the ones that will be used for the repdb.

Table x: RebDB prerequisites

Monitoring RepDB

Follow the procedures discussed per Ref-2 to verify that Data Replication from the primary server to the secondary server is functioning properly (PrimaryServer□SecondaryServer).

SARA AppServer Considerations

This section is included for context and background.

In the past, the Cisco SARA Application Server (AppServer) has been deployed as a stand-alone application on its own platform, something similar to:

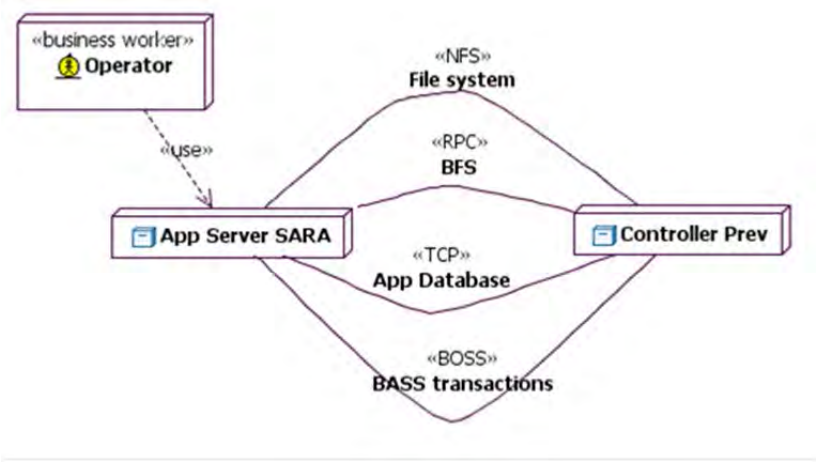

Figure x: Stand-alone SARA server example

Also, the Cisco SARA AppServer is integrated on DNCS/EC and deployed on the same platform, similar to:

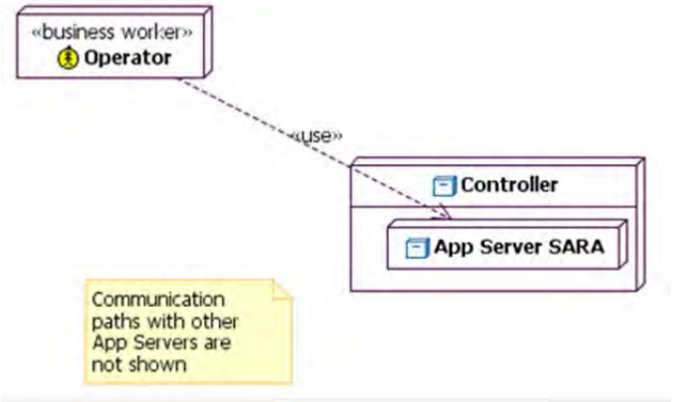

Figure x: Integrated SARA server example

AppServer Process List

The following table lists the processes on the Application Server.

Process	Description
DHCT (set-top) Configuration Server	Generates the files containing the global, hub-specific global, and staging configuration values for SARA. Places those files on the BFS.
Interactive Program Guide Server	Generates the IPG files for a specific language and places that file on the BFS.
Pay-per-view Server	Receives PPV event definitions from the billing system and the PPV UI and stores them in the database. It also notifies the PPV File Server process when it is time for the PPV File Server to generate updated files.
Pay-per-view File Server	Generates PPV files for SARA and places those files on the BFS.
PPV XML Server	Receives NDS-related PPV events and generates Future ECMs (fECMs). This server is only specific to installations of a particular MSO.
Virtual Channel Server	Places the files for all configured virtual channels on the BFS.

Table x: Processes on the Application Server

Operator Interaction

The operator can configure settings on the integrated Cisco SARA AppServer as follows:

Note: The integrated solution allows the operator to enable/disable the functionality of the Cisco SARA AppServer using a feature flag.

- Administer set-top configuration for all set-tops
- Administer configuration for a single set-top
- Administer set-top configuration for a hub
  - Create hub-specific item list
  - Add new hub-specific configuration
- Set up the language for PPV and IPG
  - Administer IPG Server for a language
  - Administer IPG Collector for a language
  - Administer IPG Service

AppServer Database Concerns

The AppServer, whether deployed on its own stand-alone platform or integrated with the DNCS/EC, shares the IBM Informix database engine and allocated space as configured and deployed on the DNCS/EC.

Specifically, the AppDB is allocated as another DB instance serviced by the same Informix engine facilities as the DNCS/EC data store. From that perspective, for allocating, deallocating, configuring, defragging, etc., the AppServer's database impacts the use and footprint of the Informix database on the DNCS/EC.

Database Fragmentation (per Ref-1)

Over time, due to high database activity, tables and indexes might become fragmented. Database fragmentation occurs normally as the system is continually creating, deleting, and modifying data in the database.

Tables and indexes that are spread over a number of extents can cause significant performance degradation due to increased disk head movement and increased contention for head positioning with other objects in the database. Such fragmentation should be avoided. Defragmenting the table extents brings data rows closer together in contiguous, merged extents, which in turn, brings efficient data access.

Table space is allocated in blocks of disk called *extents*. When a table is created, Informix, by default, creates the table with an initial first extent size. Additional extents are added as needed with a size determined by the next extent size. The default extent size for Informix is 8 pages.

Contiguity of physical pages is important to performance. When pages of data are contiguous, the time used to access data on disk is minimized and the database can read rows sequentially. If tables have too many extents, it is very likely that those extents are interleaved, which could harm performance a great deal.

When to Defrag

If any large tables (more than 10,000 rows) have 30 or more extents, with the exception of specific pre-sized tables, you should consider rebuilding them to consolidate the extents.

In DNCS release 4.X (Informix 9.2), and our later releases of EC starting with 6.x and up (Informix 11.7), we do pre-size with an initial extent and a next extent size all the large database tables.

On the pre-sized database tables it is fine to have extents, since we are sizing them to a maximum average based on usage.

If any of the tables showing more than 30 extents are not in the list of pre-sized tables, you should consider performing the defrag exercise.

This is a list of pre-sized tables in the DNCS/EC.

- atm_connection
- atsc_mod_type
- authorization
- available_imagesid
- bfselement
- bfselement_params
- bfsserver
- hct_profile
- home_account
- home_devices
- hub
- ip_association
- ip_cluster
- ip_localization
- pdmpegport
- pdoshct
- pdpasession
- pdrfparameters
- pdsegment
- pdsernummap
- pdservercarsr

- bfsservertosource
- bfssource
- camexsession
- county
- davic_qpsk
- davic_qpsk_demod
- dcm
- displaychannels
- eam
- eis_parameters
- elementary_stream
- emm
- entitlement_unit
- ethernetinterface
- event
- event_bigdata
- event_pub
- event_sub
- fips
- fips_oob_bridge
- gige_transport
- hct_ip_params
- ip_source
- ippv_mode
- ippv_purchase
- linked_home_accounts
- mpeg_program
- name
- ne_alarm_parms
- netcrypt
- opencable_id
- opencasqamparameters
- package
- package_member
- pcg
- pdauthorization
- pdcaqam
- pdclientcarsr
- pdheadendrsr
- pdkeycertificate
- pdmpegconnector
- pdmpeggraph
- pdmpeggraphnode
- pdservercauserid
- pdsource
- pdsourceident
- pdsourcesecurity
- pdsrcinfo
- pdvodgraph
- pdwire
- phys_channel
- podhostpair
- ppvevent
- preallocated_rsr
- qpsk_docsis
- qpsk_mcast_params
- rbfs_file
- samservices
- sdbrsr
- secure_micro
- session
- shared_resource
- sm_auth_profile

Table x: Pre-sized tables in the DNCS/EC

System Failure

If you experience a DBDS system failure involving a Solaris panic or an Informix assertion failure, you should contact Cisco Services for assistance. Cisco engineers will defragment your tables and rebuild your indexes using the dncsDbData and formatDbSpace.sh utilities. If these recovery steps are not performed, you may experience other database failures in the future.

Solaris panics are logged in the /var/adm/messages file, and Informix assertion failures are logged in the export/home/informix/online.log. To determine whether you should contact Cisco Services, look for messages in the /export/home/informix/online.log stating that Informix performed work during fast recovery.

Example: Defrag Not Necessary

The following example indicates that your tables do not need to be defragmented. Messages indicate that no work was needed or performed during the recovery. This example shows what you will see upon startup after a graceful shutdown.

```
15:16:36  Physical Recovery Started.
15:16:36  Physical Recovery Complete: 0 Pages Restored.


15:16:36  Logical Recovery Started.
15:16:36  20 recovery worker threads will be started.
15:16:39  Logical Recovery Complete.
          0 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks
```

In this case, you should restart the system and verify that there are no other problems, then check the log files once again for errors.

Example: Defrag Necessary

The following example indicates that you should contact Cisco Services. Messages indicate that 1776 pages were restored, 2995 records were committed, and 14 records were rolled back. This example shows what you will see upon startup after an uncontrolled shutdown.

```
12:09:11  Physical Recovery Started.
12:09:12  Physical Recovery Complete: 1776 Pages Restored.


12:09:12  Logical Recovery Started.
12:09:12  20 recovery worker threads will be started.
12:09:17  Logical Recovery Complete.
          2995 Committed, 14 Rolled Back, 0 Open, 0 Bad Locks
```

In cases like this, you should contact Cisco Services to defragment your tables and rebuild your indexes.

Monitoring for Database Fragmentation

Run and analyze the output of the Doctor Report regularly. Pay attention to the headings in the report called **Database Table Extents** for dncsdb and Database Table Extents for appdb.

The number of extents (more than 30) associated with a few specific tables provides you with some warning that the database is becoming fragmented, and that there might be a need to defrag the database.

The exception to this rule is if the table is in the Pre-Sized table list shown in the previous topic.

Eliminating Database Fragmentation

You should follow this procedure to complete a full defragmentation of the DNCS Informix database.

Operations Alert Bulletin: Working with Cisco Services to Recover From a DBDS System Failure (part number 78-4021636-01) was written to inform customers of the need for this procedure and should be referenced for the scheduling of the SURF. This activity is generally scheduled using SURF and performed in the maintenance window.

Please note if the customer is using Disaster Recovery. If so, there are additional steps required to update the Informix Database following this procedure. Those steps have been incorporated into this document. The approach to the management of database fragmentation includes the following:

- Pre-maintenance Window Tasks
- Maintenance Window Tasks
- Post Defrag Tasks
- RepDB and database defragmentation

Pre-Maintenance Window Tasks

Since the result of this procedure is the complete reformatting of the Informix database, it is vital to have a recent copy of the Informix database available to use as a recovery point. The database backup can take approximately one hour to execute on larger systems.

This procedure requires the **dncs** and **root** passwords. Make sure you have been provided with both before proceeding.

1. Verify that DBDS Utilities version 6.1 or greater is installed on the Cisco controller.
2. Verify that a recent database backup has been executed and saved. This backup can be to tape or to an auto-backup server.
3. Log into the controller.
4. Change to the dncs role.
5. Run a full Doctor report.
6. Review the Doctor report for basic errors. This ensures that the system is in a good state to complete a defrag and be compared to a Doctor Report later in the procedure.
7. Review the Doctor report to determine how the Cisco SARA Application Server is used (if at all). If the site does NOT use the Cisco SARA AppServer, you can skip all the tasks for the Cisco SARA AppServer. However, in the case of the use of a third-party AppServer, there may be some interaction with the third-party AppServer that you must consider (for example, stopping/starting the server at the proper time).
8. Make there is sufficient disk space available to complete the defrag:

```
df -k /dvs/backups
```

```
$ df -k /dvs/backups
Filesystem            kbytes    used   avail capacity
Mounted on
/dev/md/dsk/d510     61682748 14930923 46134998    25%
/disk1
```

Note: Check the avail for the /disk1 slice. Make sure that there is a least 10 GB free to complete the defrag; otherwise, you will need to acquire additional space by removing unnecessary files. DO NOT remove files from the customer's system if you are uncertain they are needed.

9 Create the following directories :

```
$ cd /dvs/backups
/dvs/backups
$ mkdir DbBackups
$ cd DbBackups
$ pwd
/dvs/backups/DbBackups
$ mkdir DNCSDB_<DATE>
$ mkdir APPDB_<DATE>
$ ls -ltr
total 4
drwxr-xr-x   2 dncs     dncs         512 Apr 29 06:28
DNCSDB_04292012
drwxr-xr-x   2 dncs     dncs         512 Apr 29 06:28
APPDB_04292012
```

10 Save a copy of the current onconfig file to ensure continuity after the defrag is complete:

```
$cd /export/home/informix/etc
$cp –p onconfig onconfig.predefrag.<date>
```

11 Make sure that there are no third-party applications accessing the system throughout the entire procedure (for example, business or operations support systems (BSS/OSS), billing, third-party AppServer, etc.).

12 You can make sure that there are no database session against the database by running the following command as root user:

```
# onstat -g ses
```

This shows you whether there are any sessions running against the database for any user. If there are no sessions running, as the example below shows, then you can continue with the pre-maintenance tasks:

```
[root@bja-ec3-node ~]# onstat -g ses

IBM Informix Dynamic Server Version 12.10.FC4W1XK -- On-
Line -- Up 64 days 02:24:52 -- 2442964 Kbytes

session #RSAM total used dynamic
id user tty pid hostname threads memory memory explain
26021 informix - 0 - 0 16384 12440 off
```

```
25209 informix - 0 - 0 16384 12440 off
25207 informix - 0 - 1 45056 41184 off
25206 informix - 0 - 1 45056 41200 off
17305 informix - 0 - 0 16384 14032 off
40 informix - 0 - 1 897024 530960 off
39 informix - 0 - 1 905216 529096 off
38 informix - 0 - 1 663552 485080 off
37 informix - 0 - 1 98304 84888 off
11 informix - 0 - 0 16384 12440 off
10 informix - 0 - 0 16384 14032 off
9 informix - 0 - 0 16384 12440 off
8 informix - 0 - 0 16384 12440 off
6 informix - 0 - 0 16384 14032 off
5 informix - 0 - 0 16384 14032 off
4 informix - 0 - 0 16384 12440 off
3 informix - 0 - 0 16384 12440 off
2 informix - 0 - 0 16384 12440 off
```

13 **(SP 4.2.X Systems Only)** - There may be instances where it becomes necessary to block certain traffic from accessing the DNCS controller. For example, you need to prevent an external application from accessing the database while maintenance is in progress. You can do this with the *ipfilter* command in Solaris 10. The following steps allow you to block traffic from any port trying to reach the Informix database.

Note: To run these commands, you must be logged in as **root** (superuser).

a Is the site running the SARA AppServer?
- If **yes**, go to the next step.
- If **no**, go to step g.

b Is the site using an integrated SARA AppServer?
- If **yes**, go to the step e.
- If **no** (the SARA AppServer is installed on a separate platform from the DNCS controller), go to step c.

c On another terminal or xterm log into the SARA AppServer as the **dncs** user.

d Verify that the AppServer can reach the DNCS and the appdb.

```
ping dncsatm
```

e Access the SARA AppServer's database instance by executing the following command:

```
dbaccess appdb -
```

Result: The command should return a **>** prompt

f Press **CTRL + C** (control key and the C key simultaneously) to exit the database shell. Leave the terminal window open.

g Log into the DNCS/EC as **root :**

```
su - root
```

h Enable ipfilter:

```
svcadm enable network/ipfilter
```

i Create an ipf file with ruleset:

```
echo "block in quick from any to any port = 3010" >
/tmp/ipf.conf
```

j Place the ruleset into effect:

```
ipf -Fa -f /tmp/ipf.conf
```

k View and verify which rules are in effect:

```
ipfstat -io
```

14 Make sure that any Disaster Recovery, Auto Backup or Basic Backup applications are suspended for the duration of the maintenance window.

15 During the maintenance window, perform the tasks detailed in the next topic.

Maintenance Window Tasks

This section requires the processes to be stopped. Inform the customer that no new VOD sessions or billing hits will process once you begin. If any of these tasks fail to complete, proceed with normal troubleshooting or escalation.

1 Install and run the CFET-provided screen binary in the dncs home directory.

a SPARC architecture:

```
install sol10_screen_sparc
```

b x86 architecture:

```
install sol10_screen_x86
```

c Rename sol10_screen_sparc:

```
mv sol10_screen_sparc screen
```

d Add execute permission:

```
chmod 775 screen
```

2 Execute the screen utility:

```
$screen
```

3 Verify with the customer that all systems and operations are prepared for the controller processes to be stopped (for example, check with customer operations).

4 If this is a Disaster Recovery customer (running the Cisco replicated database solution (RepDB)), follow the procedure to disable the RepDB per Ref-2, Chapter 4, *Advanced Features for Data Replication, Disable Data Replication*.

5 Stop the DNCS/EC processes:

```
$ dncsStop
Are you sure you want to stop the DNCS (Y/N)? Y
$ dncsKill
If the Cisco SARA AppServer is in use;
Stop the Cisco SARA AppServer processes:
$ appservatm
$ appStop
$ appKill
```

6 Stop the cron jobs on the Cisco SARA AppServer and DNCS/EC.

If the Cisco SARA AppServer is in use and deployed on its own platform (you should still be on the Cisco SARA AppServer platform):

```
$ su
```

```
password: (enter the root password)
#svcadm disable cron (Solaris 10)
```

Or

```
# /etc/init.d/cron stop (Solaris 8)
# pgrep -fl cron
#
 (/usr/sbin/cron should NOT be running at this point)
#exit
$exit
Remote connection closed
$uname -a  (you should be back on dncs)
$ su -
password:  (enter the root password)
#svcadm disable cron (Solaris 10)
```

Or

```
# /etc/init.d/cron stop (Solaris 8)
# pgrep -fl cron
#
 (/usr/sbin/cron should NOT be running at this point)
```

7 Source the dncs profile as root user:

```
# . /dvs/dncs/bin/dncsSetup
Working directory is /dvs/dncs
Database is dncsdb
-o: bad option(s)
```

8 Verify the metadb's.

```
metadb –i
flags           first blk       block count
a m  p  luo       16              8192
/dev/dsk/c1t0d0s7
a    p  luo       16              8192
/dev/dsk/c1t1d0s7
a    p  luo       16              8192
/dev/dsk/c1t4d0s7
a    p  luo       16              8192
/dev/dsk/c1t5d0s7
a    p  luo       16              8192
/dev/dsk/c1t8d0s7
```

9 Detach the DNCS/EC Mirrors.

```
# metadetach d520 d720
d520: submirror d720 is detached
```

10 Check for active database session connections:

```
# showActiveSessions
```

```
Database selected.
1 row(s) unloaded.
Database closed.
INFORMIXSERVER  dncsDbServer is idle.
```

11 Did the output from the previous step show that the DB server is **idle**?

- If **yes**, go to the next step.
- If **no**, follow these steps to use killActiveSessions to remove active DB session connections:

```
# killActiveSessions
Database selected.
0 row(s) unloaded.
1 row(s) unloaded.
Database closed.
INFORMIXSERVER  coxdncsDbServer is idle.
```

12 Change to the directory to begin the DNCS/EC Database unload:

```
# cd /dvs/backups/DbBackups/DNCSDB_<DATE>
# pwd
/dvs/backups/DbBackups/DNCSDB_<DATE>
```

13 Rename the database by doing the following (Remember the ! is part of the complete command):

```
dbaccess sysmaster <<!
rename database dncsdb to defragdb;
!
Database selected.
Database renamed.
Database closed.
Begin the DNCS/EC Database unload:
# dncsDbData -d defragdb -u
                  ————UNLOAD DATABASE————
```

14 If using tapes, insert the DATA tapes. Remember to label them. Tape-based exports and imports do not enjoy the dncsDbData enhancements associated with row count verification and should be avoided if at all possible.

```
Enter destination (Deflt:    Data Cartridge) ..:
    1 -        Data Cartridge
    2 -        Local Hardrive
***Select 2   for Local HD
Is the above information correct (Y/N): ........:    Y
Enter backup directory (Default: Current Directory) .....:
***Press "Enter" for current directory
Is the above information correct (Y/N): ........: Y
```

System Response:

```
Please wait while unloading the data to disk.....
```

The unload begins and can take several hours to complete for large systems.

15 Detach the screen session.

```
screen –d
```

16 Monitor the following for errors:

```
/dvs/dncs/tmp/dncsDbData_dncsdb.err
dncsDbDataVerify_dncsdb_export.out
```

17 When completed, the system shows the following message:

```
Process is completed.
```

18 If the following ERROR is displayed, go to Appendix A for corrective actions:

```
ERROR: Row counts do not match and/or other errors
detected; see lines marked with an asterisk
```

19 Resume the screen session.

```
screen –r
```

20 Change to the directory to begin the Cisco SARA AppServer Database unload:

```
# cd /dvs/backups/DbBackups/APPDB_<DATE>
# pwd
/dvs/backups/DbBackups/APPDB_<DATE>
# dncsDbData -d appdb -u
```

21 Repeat all responses entered for the DNCS/EC unload directly above. The unload begins and usually takes 5 to 15 minutes to complete.

22 Detach the screen session.

```
screen –d
```

23 Monitor the following for errors:

```
/dvs/dncs/tmp/dncsDbData_appdb.err
dncsDbDataVerify_appdb_export.out
```

24 When completed, the system shows the following message:

```
Process is completed.
```

25 Resume the screen session.

```
screen –r
```

26 If the following ERROR is displayed go to Appendix A for corrective actions:

```
ERROR: Row counts do not match and/or other errors
detected; see lines marked with an asterisk
```

27 Run the following to initialize the Informix Database:

Note: If you are unloading and reloading to fix your extents, skip to step 15.

```
# /export/home/informix/bin/formatDbSpace.sh
```

Notes:

- The system displays output while the Informix Database is initialized.
- This step usually takes 15 to 20 minutes to complete.

28 Open a second xterm window to monitor cron to make sure it does not restart during the next steps of reloading the database.

Important:

- To monitor, either use `top` or `ps -ef | grep cron`.
- If cron does restart, kill it immediately.

29 Reload the DNCS/EC Database: (import):

```
# cd /dvs/backups/DbBackups/DNCSDB_<DATE>
```

```
# pwd
/dvs/backups/DbBackups/DNCSDB_<DATE>
# dncsDbData -d defragdb -l  -c /dvs/dncs/etc/dncsDbConfig
```
(lower-case L)

30 Select the option to reload from the current directory. The system displays output while the DNCS Database is reloaded.

31 When completed, the system shows the following messages:
```
Turning buffered loggin ON started
Turning buffered loggin ON ended
LOAD DATABASE ended
You can find these lines can be found in
/dvs/dncs/tmp/dncsDbData_defragdb.log.
Process is completed.
```

32 Reload the Cisco SARA AppServer Database:
```
# cd /dvs/backups/DbBackups/APPDB_<DATE>
# pwd
/dvs/backups/DbBackups/APPDB_<DATE>
# dncsDbData -d appdb -l -c
/dvs/appFiles/dbConfig/appDbConfig
```
(lower-case L)

33 Select the option to reload from the current directory. The system displays output while the Cisco SARA AppServer Database is reloaded.

34 When completed, the system shows the following messages:
```
Turning buffered loggin ON started
Turning buffered loggin ON ended
LOAD DATABASE ended
(These lines can be found in
/dvs/dncs/tmp/dncsDbData_appdb.log)
Process is completed.
```

35 Before continuing, check the output files for any errors and address the errors to ensure the tables, indexes, and keys are built.
```
dncsDbDataVerify_dncsdb_import.out.
dncsDbDataVerify_appdb_import.out.
```

36 Is the current onconfig file the same as the previous onconfig file?
```
$cd /export/home/Informix
$ diff onconfig onconfig.predefrag.<date>
```
- If **yes**, go to the next step.
- If **no**, copy the previous over the current file:
  ```
  $cp -p onconfig.predefrag.<date> onconfig
  ```

37 Type the following to rename the database to dncsdb (Remember the ! is part of the complete command):
```
dbaccess sysmaster <<!
rename database defragdb to dncsdb;
!
```

38 If this is a Disaster Recovery" customer (running the Cisco replicated database solution (RepDB)), follow the procedure to enable the RepDB per Ref-2, Chapter 2 Ins*talling and Configuring the Replicated Database Software, Activate Data Replication*.

Post-Defrag Tasks

This section will restart crontab, the processes, and verify the system following the defrag operation.

1 Restart the cron jobs:

```
#svcadm enable cron (Solaris 10)
```

Or

```
# /etc/init.d/cron start (Solaris 8)
# pgrep -fl cron
 5324 /usr/sbin/cron
```

2 Is the customer using the Cisco SARA AppServer deployed on its own platform?

- If **yes**, repeat this task on the Cisco SARA AppServer platform.
- If **no**, go to the next step.

3 Disable the ipfilter

```
#svcadm disable network/ipfilter
```

4 Is the customer using the Cisco SARA AppServer deployed on its own platform?

- If **yes**, verify that the Cisco SARA AppServer can reach the DNCS/EC and the appdb:

  ```
  ping dncsatm
  dbaccess appdb –
  ```

  Result: The command should return with a **>** prompt
- If **no**, go to the next step.

5 Verify that the database reloaded:

```
#exit
$admincon
```

6 Open the Source GUI. If the Sources appear with no errors, go to the next step.

7 Start the processes on the DNCS/EC:

```
$dncsStart
Loading ASI driver...
Database engine is On-Line ...
  Setting table [hct_profile] MEMORY RESIDENT
  Setting Index [hct_profile_idx] MEMORY RESIDENT
  Setting Index [hctnsapidx] MEMORY RESIDENT
  Setting Index [hctseidx] MEMORY RESIDENT
…….
DNCS Applications started.
```

8 Is the customer using the Cisco SARA AppServer deployed on its own platform?

- If **yes**, start the processes on the Cisco SARA AppServer:

  ```
  $appservatm
  ```

```
$appStart
$exit
```

- If **no**, go to the next step.

9 Verify VOD is working by running `sesstail`.

10 Run a Doctor Report and review for errors.

11 Reboot a DHCT and verify VOD/IPG/PPV all work correctly.

12 Perform a qtail bossServer to make sure that billing is communicating correctly.

13 Complete an Informix database backup.

14 Attach the DNCS/EC mirrors.
`mirrState.ksh –a`

15 Verify customer is satisfied that the system is functioning normally.

16 Monitor the state of the mirrors until all devices are synchronized.
`syncwait.ksh`

17 Remove the files and directories that were created from `/dvs/backups/DbBackups` and the screen utility that was installed.

18 End the call and close the SURF.

RepDB and Database Defragmentation

Note: These steps have been integrated into the procedures in the previous sections.

1 Disable the RepDB. Follow the procedure to disable the RepDB per Ref-2, Chapter 4, *Advanced Features for Data Replication, Disable Data Replication*.

2 Perform the Defrag procedure as previously described.

3 Enable the RepDB. Follow the procedure to enable the RepDB per Ref-2, Chapter 2, *Installing and Configuring the Replicated Database Software, Activate Data Replication*.

Appendix A: Defrag Procedure Error Recovery

Note: The following procedure is the same for either the dncsdb or appdb, the only difference are the output log file names:

- For dncsdb: `dncsDbDataVerify_defragdb_export.out`
- For appdb: `dncsDbDataVerify_appdb_export.out`

The dncsdb is used in the example. We recommend that you export both databases before performing the corrective actions.

1. Determine the problem table(s) by performing the following command (in this example, the emm table is reported although multiple tables may be returned):
   ```
   # grep "*"dncsDbDataVerify_defragdb_export.out
   emm                    24,089,891        24,089,891
   24,089,887*
   ```
2. Go into the export directory.
   Note: This file is typically has an .exp file extension (for example, defragdb.exp or appdb.exp). The naming convention used for the exported tables names is [`(dbtablename).unl.gz i.e. emm.unl.gz`]
3. Perform the following to determine the appropriate number:
   ```
   # gzcat emm.unl.gz|wc -l
   24089887
   ```
4. Return to the parent directory (`# cd ..`) and open a text editor to edit the output file listed in step 1.
5. Scroll to the line for the table name, emm in this instance.
6. Change the numbers so they are the same for all 3 columns.
7. Remove the * at the end of the line
8. Using a text editor, edit the file `.TableNames`
   Note: The leading **.** is part of the file name, scroll to the line of the impacted table(s) and modify the line to reflect the correct number as noted above:
   `emm|24089891.0|20885935.497|` (before modifying)
   `emm|24089887.0|20885935.497|` (after modifying)
9. Return to the import portion of the defrag procedure.

Appendix B: Defrag Log Files

This section describes the log files used during the defrag procedure, assuming that you used the naming convention suggested earlier in the procedure.

Exporting defragdb (dncsdb)

- `/dvs/backups/DbBackups/DNCSDB_<DATE>/.TableNames`
  Created during the export phase as a pipe delimited file with the table name, size, and checksum.
- `/dvs/backups/DbBackups/DNCSDB_<DATE>/dncsDbDataVerify_defragdb_export.out`
  This file is used during the db parsing and verifies the row and table counts during the export.
- `/dvs/dncs/tmp/dncsDbData_defrag.log`
  This file is used during the actual table export and should specify UNLOAD as each table is successfully unloaded with output similar to:

```
Tue Mar 20 03:07:07 EDT 2012<> UNLOAD DATABASE from disk
started
.
.
.
Tue Mar 20 03:07:07 EDT 2012<> UNLOADING [  24089891] rows
in table emm
```

Importing defragdb (dncsdb)

- `/dvs/backups/DbBackups/DNCSDB_<DATE>/.TableNames`
  This file is referenced during the import phase to insure proper table naming and sizes.
- `/dvs/backups/DbBackups/DNCSDB_<DATE>/dncsDbDataVerify_defragdb_import.out`
  This file is used during the db parsing and verifies the row and table counts during the import.
- `/dvs/dncs/tmp/dncsDbData_defrag.log`
  This file is the same file used during the export and is overwritten with the actual table import data. It will specify LOAD as each table is successfully loaded with output similar to:

```
Tue Mar 20 03:07:07 EDT 2012 <> LOAD DATABASE from disk
started
.
.
.
```

```
Tue Mar 20 03:07:07 EDT 2012<> LOADING [  24089891] rows in
table emm
Tue Mar 20 04:28:17 EDT 2012 <> Turning buffered loggin ON
started
Tue Mar 20 04:28:27 EDT 2012 <> Turning buffered loggin ON
ended
Tue Mar 20 04:28:27 EDT 2012 <> LOAD DATABASE ended
```

Exporting appdb

- `/dvs/backups/DbBackups/APPDB_<DATE>/.TableNames`
  This file is created during the export phase as a pipe delimited file with the table name, size, and checksum.
- `/dvs/backups/DbBackups/APPDB_<DATE>/dncsDbDataVerify_appdb_export.out`
  This file is used during the db parsing and verifies the row and table counts during the export.
- `/dvs/dncs/tmp/dncsDbData_appdb.log`
  This file is used during the actual table export. It specifies UNLOAD as each table is successfully loaded with output similar to the load of defragdb as listed above, except it will specify the appdb-specific table names and sizes.

Importing appdb

- `/dvs/backups/DbBackups/APPDB_<DATE>/.TableNames`
  This file is referenced during the import phase to insure proper table naming and sizes.
- /dvs/backups/DbBackups/APPDB_<DATE>/dncsDbDataVerify_appdb_import.out
  This file is used during the db parsing and verifies the row and table counts during the import.
- /dvs/dncs/tmp/dncsDbData_appdb.log
  This file is the same file used during the export and is overwritten with the actual table import data. It will specify LOAD as each table is successfully loaded with output similar to the load of defragdb as listed above, except it will specify the appdb specific table names and sizes.

For Information

If You Have Questions

If you have technical questions, contact Cisco Services for assistance. Follow the menu options to speak with a service engineer.

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA

http://www.cisco.com
Tel: 408 526-4000
800 553-6387
Fax: 408 527-0883

February 2016 Part Number TP00131-01