CA Spectrum®

Database Management Guide

Release 9.4

technologies

CA Technologies Product References

This guide references CA Spectrum®.

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Overview

This chapter provides an overview of CA Spectrum database maintenance and discusses both the SpectroSERVER (SS) and Distributed Data Manager (DDM) databases.

This guide discusses CA Spectrum database management in terms of a distributed network environment with multiple SpectroSERVERs. Although distributed computing is an industry standard, the information in this guide can be adapted for non-distributed environments. Distributed database management in CA Spectrum includes several levels of CA Spectrum database maintenance. Various modeling rules and procedures must be followed, and numerous applications can be used to manage a distributed CA Spectrum network and its associated databases.

This guide describes maintaining both the local database that is a part of each SpectroSERVER and the historical databases that represent multiple SpectroSERVERs.

This section contains the following topics:

The SpectroSERVER Database

The SpectroSERVER database, which is located in the *$SPECROOT*/SS directory, contains the following information:

- A modeling catalog (model types and relations), the structure for all network information
- Created models that belong to this SpectroSERVER
- A storage area for network events and statistics

The icons that represent network devices in OneClick are reporting information that is retrieved from a *model* (a software simulation) of the actual device. The model is maintained in the SpectroSERVER database and is updated with information from several sources. Information that is retained from one execution of the SpectroSERVER to the next is also stored in the SpectroSERVER database.

Distributed Data Manager (DDM) Database

Each SpectroSERVER has a Distributed Data Manager (DDM) database to store CA Spectrum events and statistical data for use across multiple landscapes. Client applications, such as CA Spectrum Report Manager and CA Spectrum AlarmNotifier, can then request, receive, and collate the data by landscape. The DDM database is located in the SpectroSERVER *$SPECROOT*/mysql/data/ddmdb directory.

Note: For more information, see the *Distributed SpectroSERVER Administrator Guide*.

Like the SpectroSERVER database, multiple concurrent users of CA-developed programs are prevented from accessing the DDM database. CA Spectrum applies a soft lock file (*$SPECROOT*/SS/DDM/.DDMDB.LOCK) to prevent access from multiple, simultaneous CA-developed applications. Under certain circumstances (for example, when recovering from an abnormal shutdown), the soft lock file can be removed.

Important! When CA-developed applications encounter a database lock file, an error message alerts you. However, non-CA tools and applications may not check for this lock and, therefore, do not generate any message. If these non-CA entities are able to bypass the lock, database corruption can result. As a best practice, before allowing any non-CA tool or application to access a CA Spectrum database, verify that all CA Spectrum processes are shut down.

The Archive Manager controls all communications between the DDM and the SpectroSERVER databases, and between the DDM database and client applications.

More information:

Removing a Database Lock (see page 22)

Archive Manager

Each landscape has an Archive Manager server that retrieves events and statistical data from the SpectroSERVER, compresses them, and stores them in the DDM database. Data compression enables storage of more performance data and decreases the network traffic between the applications and the DDM database.

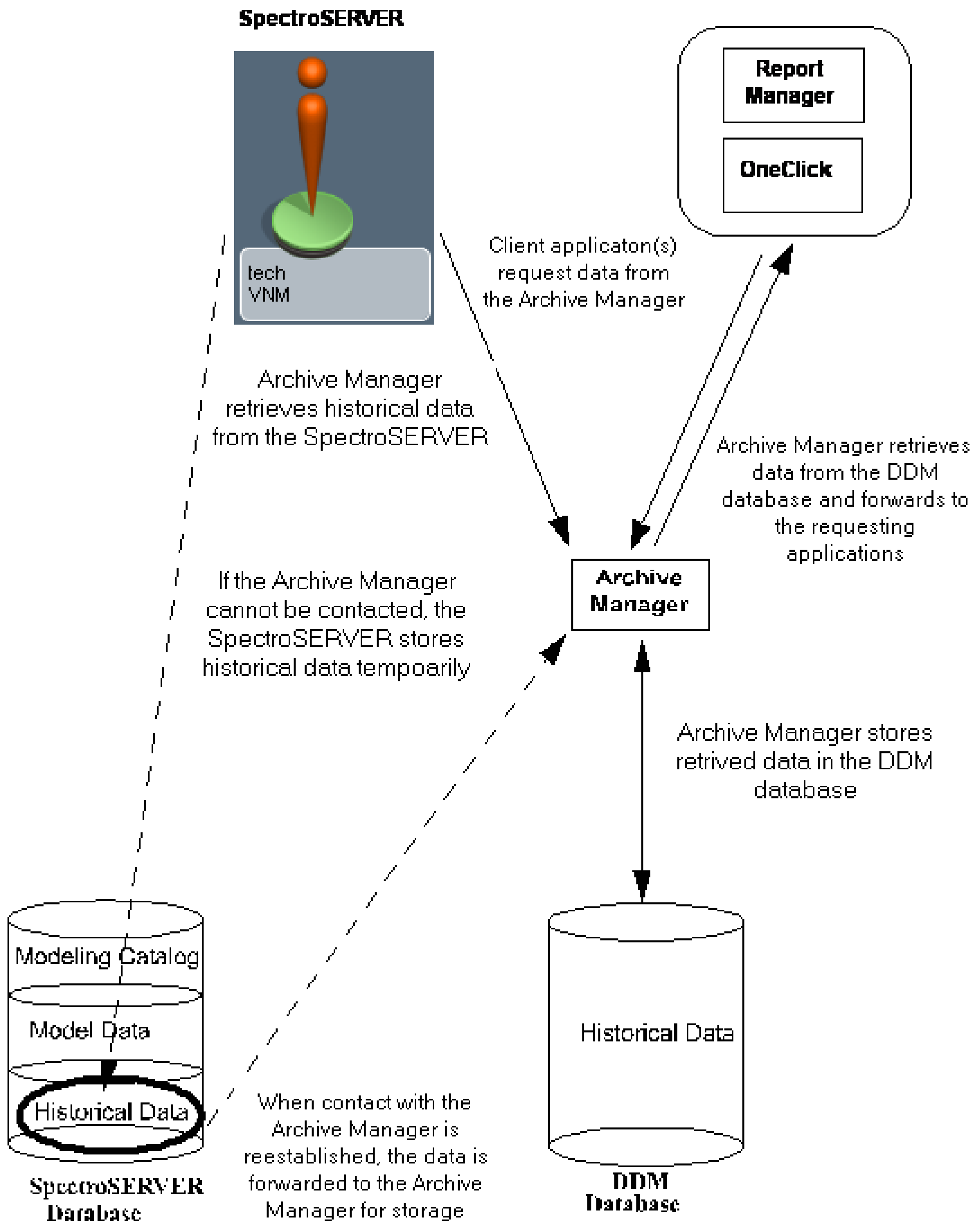
As the diagram shows, if the SpectroSERVER cannot contact the Archive Manager, the SpectroSERVER stores events and statistical data until contact is reestablished. The SpectroSERVER then sends the data to the Archive Manager for storage. The Events and Statistics Archive options in the .vnmrc file determine the amount of data that the SpectroSERVER stores. Options in the .configrc file determine the length of time that historical data is stored in the DDM database.

Note: For more information, see the *Distributed SpectroSERVER Administrator Guide.*

The Archive Manager can also provide the following information in response to requests from client applications:

- A list of landscapes for which information is available
- For each landscape, the time range of the available information and a list of model types for which information is available
- For each model type, a list of models for which information is available
- For each model, a list of attributes for which information is available
- Statistical data in the specified time range
- Event data in the specified time range

The following diagram shows the interaction between the SpectroSERVER and the DDM database.


SpectroSERVER
tech
VNM
Report Manager
OneClick
Client applicaton(s) request data from the Archive Manager
Archive Manager retrieves historical data from the SpectroSERVER
Archive Manager retrieves data from the DDM database and forwards to the requesting applications
If the Archive Manager cannot be contacted, the SpectroSERVER stores historical data tempoarily
Archive Manager
Archive Manager stores retrived data in the DDM database
Modeling Catalog
Model Data
Historical Data
SpectroSERVER Database
When contact with the Archive Manager is reestablished, the data is forwarded to the Archive Manager for storage
Historical Data
DDM Database

Chapter 2: SpectroSERVER Database Maintenance

This chapter describes database maintenance procedures for the SpectroSERVER database. Maintenance procedures for the Distributed Data Manager (DDM) database are discussed separately, in DDM Database Maintenance (see page 61).

This section contains the following topics:

Database Backups

Creating regular backup copies of your database is the foundation of database maintenance. A reliable backup copy of your database can enable you to restore the database following power failures or other system interruptions.

CA Spectrum offers two methods of performing backups of the SpectroSERVER database:

- Automatically with the SpectroSERVER running.
- Manually with the SpectroSERVER shut down.

Important! The database backup methods that are described in this section, Online Backup and the SSdbsave utility, are the only supported methods of backing up the SpectroSERVER database. Use of third-party backup software can result in database corruption.

More information:

Online Backup (see page 12)
Manual Backup (see page 17)

Online Backup

Online Backup lets you create a backup copy of your SpectroSERVER database without having to shut down the SpectroSERVER. Depending on your requirements, you can perform online backups on demand, or you can schedule regular backups to be performed automatically. If disk space is limited, you can configure Online Backup to compress backup files automatically using the CA Spectrum gzip utility.

Online Backup saves the entire database, including the modeling catalog and models. However, neither Online Backup nor a manual backup operation saves the following information:

- Cached event information
- Alarms
- Cached statistical information
- Historical records in the DDM database (see the note below)
- SpectroSERVER resource file (.vnmrc)

Online Backup activity is recorded as events associated with the VNM model. CA Spectrum reports errors that were encountered during backup operations as alarms.

Online Backup performs a save in two major steps:

1. Makes a copy to preserve a “snapshot” of the database files.

   Polling, trap handling, and network management activities are suspended during this first step. The process is relatively short, but as a best practice, consider how often and when to schedule automatic save operations. The time to perform a copy operation depends on your workstation hardware and the database size.

2. Saves the copy (and compresses it when required) using the same format as the manual database save utility, SSdbsave.

The Online Backup does not, by itself, save the DDM database. However, you can configure the post_olb_script to execute automatically and save the DDM database whenever an Online Backup of the SpectroSERVER database is performed.

Note: In a fault-tolerant environment, verify that you are logged on to both the primary and secondary SpectroSERVER as the same user before running Online Backup.

More information:

Online Backup Events and Alarms (see page 16)

Backup File Maintenance

When automatic backups are enabled, backup files can accumulate in your backup directory and deplete the available disk space. To avoid backup failures, delete files occasionally or move backup files to a more permanent storage medium.

Configure Online Backup

When you configure online backup, you can specify the backup interval and the date and time of the first scheduled backup. For more advanced scheduling options, use the StartOnlineBackup application in the *$SPECROOT*/SS-Tools directory to initiate online backups. The StartOnlineBackup application can be launched from either the Task Scheduler or the crontab for the host system. The advanced scheduling options in the StartOnlineBackup application can avoid problems where daylight savings time skews scheduled backups.

Follow these steps:

1. In OneClick, open the Universe Topology view, and then select the VNM model.
2. Click the Information tab, and expand the Online Database Backup subview.
3. Configure the following settings as required:

   Automatic Backups

   If enabled, online backups are automatically performed with the Online Backup feature according to the time interval specified in the Backup Interval setting.

   Backup Interval

   Specifies the interval between automatic backups in hours and minutes. We recommend using the default interval of 24 hours and 0 (zero) minutes so that the database is backed up at the same time every day. Enter any value. For example, 168 hours and 0 minutes (for a one-week interval) and 10,080 minutes (also for a one-week interval) are equivalent.

   Next Backup Date & Time

   Displays the date and time for the next scheduled backup. You can specify a date and time for the first backup. However, subsequent backups are performed at the interval that is specified in the Backup Interval setting.

   Backup Compression

   If enabled, backup files are compressed using the compression utility before being written to disk. Compressed files are saved with a .gz suffix appended to the filename. If disabled, files are saved uncompressed. The default value is Enabled.

Prefix for Backup File Name

Specifies the user-defined portion of the backup file name. The default prefix is "db_". However, you can specify any character string that creates a legal file name for the system on which you are running SpectroSERVER. If unset, no prefix is added to the file name.

The filename suffix indicates the date and time when the backup was executed and uses the following format: *yyyymmdd_hhmm.SSdb* [*.gz*]

Format	Description
yyyy	4-digit year
MM	month
dd	day of the month 1 through 31
hh	hour of the day - 1 through 24
mm	minute - 00 through 59
gz	indicates a compressed backup (compressed files only)
SSdb	default suffix for all database save files

Backup Directory

Specifies the directory where the backup files are stored. We recommend using a local directory. The default directory is *$SPECROOT*/SS-DB-Backup.

Minimum Required Disk Space (MB)

Specifies the minimum disk space that must be available to initiate an online backup. The default threshold value is 20 MB. If an automatic backup is initiated when the available disk space falls below the threshold, a yellow alarm is generated. The probable cause indicates a backup failure due to low disk space.

4. Click the Begin Backup Now button to initiate a backup on demand using the current settings.

   The online database backup utility first pauses SpectroSERVER operations. Then it copies and saves the database.

Note: The status of an in-progress online backup operation is displayed next to the Begin Backup Now button. If an error occurs, an event and an associated alarm are displayed on the Events tab and Alarms tab. For a list of the events and alarms that can be generated during an online backup, see Online Backup Events and Alarms (see page 16).

More information:

Online Backup Events and Alarms (see page 16)
The StartOnlineBackup Application (see page 15)

The StartOnlineBackup Application

The StartOnlineBackup application lets you use the scheduling applications in your operating system to schedule regular backups. For example, you can use the Task Scheduler in the Windows environment or the crontab in the Solaris environment. Using these applications avoids potential problems with daylight savings time that can skew the scheduled backup. This application is a more sophisticated alternative to using the automatic backup setup.

The StartOnlineBackup application can be launched from either the Task Scheduler or the crontab. It is located in the *$SPECROOT*/SS-Tools directory. StartOnlineBackup uses the following syntax:

```
StartOnlineBackup -lh <landscape handle>
```

You can set backup parameters for StartOnlineBackup in the Online Database Backup subview on the Information tab of the VNM model. The following parameters are exceptions: Automatic Backups, Backup Interval, Next Backup Date & Time.

More information:

Configure Online Backup (see page 13)

Restore Your Database with Online Backup Files

Uncompressed backup files that Online Backup generates are stored in the same format as files that were saved using the SSdbsave utility with the -cm option.

Note: Use the gzip utility that is included with CA Spectrum to restore compressed files that have a .gz suffix. This utility restores the files to the format that the SSdbsave utility uses.

More information:

Restoring Your Database (see page 19)

Online Backup Events and Alarms

The following events and alarms are associated with Online Backup.

Event Code	Event	Event Cause	Create Alarm
0x00010903	OB_EVENT_GEN_FAILURE	Database open failure	RED
0x00010903	OB_EVENT_GEN_FAILURE	Database close failure	RED
0x00010903	OB_EVENT_GEN_FAILURE	Any other reason causing backup failure	ORANGE
0x00010904	OB_EVENT_BACKUP_ON	Automatic backups have just been enabled by the user	No
0x00010906	OB_EVENT_BACKUP_STARTED	Online Backup started	No
0x00010907	OB_EVENT_BACKUP_COMPLETED	Online Backup successfully completed	No
0x00010908	OB_EVENT_NO_FILE_OR_DIRECTORY	File/directory does not exist, or does not have read and execute permissions	YELLOW ORANGE
0x0001090a	OB_EVENT_NO_CREATE_BACKUP_DIR	Cannot create backup directory	ORANGE
0x0001090b	OB_EVENT_LOW_DISK_SPACE	Low disk space, backup failed	YELLOW
0x0001090c	OB_EVENT_COPY_FAILED	Database copy failure	RED
0x0001090e	OB_EVENT_DB_INCONSISTENT	Database inconsistency	ORANGE
0x00010920	OB_EVENT_OFFLINE_SAVE_FAILED	Off-line portion of the save process failed	RED
0x00010921	OB_EVENT_VNM_RESUMED	Resuming normal operation, SpectroSERVER was paused	No
0x00010922	OB_EVENT_BAD_FILENAME	Filename specified is not a valid Posix directory name	ORANGE
0x00010924	OB_EVENT_DBSYNC_FAILED	Attempts to synchronize backup server failed	ORANGE

Manual Backup

You can create a backup of your database manually using the SSdbsave database utility, which is included with CA Spectrum. Your database contains a catalog of template information that is used to create models (model types, relations, and rules) and the models themselves. You can create a backup copy of your database that includes either one or both of these components using SSdbsave. We recommend performing a complete save, containing both the modeling catalog and the models.

More information:

SSdbsave (see page 48)

Create a Complete Backup

You can create a complete backup copy of your database. A complete backup includes both the modeling catalog (model type hierarchy, relations, and rules) and the models themselves.

Follow these steps:

1. Navigate to the *$SPECROOT*/SS directory.
2. Verify that neither the SpectroSERVER nor any other program that accesses the SpectroSERVER database is running.

   Note: When first installed, the utility is located in the *$SPECROOT*/SS-Tools directory. If the location is not set in a system path statement, you must either use a full path, use a relative path, move the file, or link it to the same directory with the SpectroSERVER database.
3. Enter the following command to save both the modeling catalog and models:
   ```
   ../SS-Tools/SSdbsave -cm <save_file>
   ```
   The suggested naming scheme for save files is to incorporate a date stamp and the flags for the save command. In this case, you are using both the “c” (catalog) and “m” (models) flags; therefore, assuming it is October 14, 2013, enter db_20131014_cm as the save file name.

   A file named “db_20001014_cm.SSdb” is created. The “.SSdb” suffix is added automatically.

Create a Split Backup

Although the models in the SpectroSERVER database can change on a daily basis, the modeling catalog changes only when you perform an upgrade installation, add new management modules, or make changes directly using the Model Type Editor. Therefore, if you have a very large database, you can save some time by splitting your backups. With a split backup, you save your models on a regular basis, but save the modeling catalog only after it changes. You then store the backup of the modeling catalog in a safe place. Along with the most recent save of your model information, this backup file provides a complete database backup.

The following procedure describes how to perform a split backup of the database on October 14, 2013, creating one save file for the modeling catalog and a separate file for the models.

Follow these steps:

1. Navigate to the *$SPECROOT*/SS directory.
2. Verify that neither the SpectroSERVER nor any other program that accesses the SpectroSERVER database is running.

   Note: When first installed, the utility is located in the *$SPECROOT*/SS-Tools directory. If the location is not set in a system path statement, you must either use a full path, use a relative path, move the file, or link it to the same directory with the SpectroSERVER database.
3. Enter the following command to save only the modeling catalog:

   ```
   ../SS-Tools/SSdbsave -c db_20131014_c
   ```
4. Enter the following command to save only your models:

   ```
   ../SS-Tools/SSdbsave -m db_20131014_m
   ```

   The modeling catalog is saved in a file named db_20131014_c.SSdb, and the models are saved in a file named db_20131014_m.SSdb.

   Note: Save your models on a regular basis or whenever you make a significant change to your network model. Save your modeling catalog anytime you install a new version of CA Spectrum, add new management modules, or make any catalog changes with the Model Type Editor.

Restoring Your Database

Your CA Spectrum database consists of a modeling catalog that contains the model type hierarchy and relations, models to represent entities in your network, and developer information. The SSdbload database utility program let you perform the following tasks:

- Load previously backed-up database files.
- Initialize the database and load a previously backed-up database file that contains a modeling catalog.
- Load developer information.

Load a Database

If no database corruption has occurred, you can use the following command to load a backup file without first initializing the database:

```
SSdbload -l
```

You can use the SSdbload utility to load the contents of a file that the SSdbsave utility created and uncompressed files that were saved using the CA Spectrum Online Backup tool into a database.

Note: Use the gzip utility that is included with CA Spectrum to restore compressed files that have a .gz suffix. This utility restores the files to the format that the SSdbsave utility uses.

SSdbload reads the save file to determine the types of information that were saved (modeling catalog and models). It then removes all of the corresponding information from the database and loads the backup information into the database. For example, when loading a save file that contains only models (saved using only the -m option), SSdbload first deletes all model information from the database, and then loads the model information from the save file.

Similarly, when you load a save file that contains only the catalog, the operation deletes the catalog from the database and then loads the catalog from the save file. If you created separate backup files, with one for the modeling catalog and another containing model information, load the catalog (model type and relation) information first, and then load the model information.

Important! You *cannot* use the -l option to load additional catalogs or models into an existing database. The -l option does not augment the contents of the database. Instead, it determines the content of the save file that is loaded and removes any existing information of the same type before loading.

Follow these steps:

1. Navigate to the *$SPECROOT*/SS directory.
2. Verify that neither SpectroSERVER nor any other program that accesses the SpectroSERVER database is running.
3. Execute the SSdbload command, providing the name of the file to be loaded into the database structure:

   ```
   ../SS-Tools/SSdbload -l <filename>
   ```

   Note: When first installed, the utility is located in the *$SPECROOT*/SS-Tools directory. If the location is not set in a system path statement, you must either use a full path, use a relative path, move the file, or link it to the same directory with the SpectroSERVER database.
4. If the backup file loaded in Step 3 contained only catalog information, repeat Step 3, substituting the name of the backup file containing the model information.

   The database is now ready for use.

Initialize and Load a Database

You can initialize and load a database with a single operation. Initializing the database removes the modeling catalog, model information, vendor information, and unarchived events and statistics log records, and it refreshes all the database files. (To keep the databases synchronized, when you initialize the SpectroSERVER database, also initialize the historical databases using ddm_load.)

Always initialize the database if you suspect corruption. With the database initialized, the -l option then loads the new database information (modeling catalog and models) according to the SSdbsave options that were used when the backup file was created.

Important! The -i option removes all models, model types, relations, vendor information, and all unarchived events and statistics log records from the database. When running SSdbload with the -i option, you can load the catalog to start the SpectroSERVER with no models. Or you can restore the database completely. Restore it by loading one backup file that was saved using both the SSdbsave -c and -m options, or by loading a split backup (a file that was saved using -c and -m options separately).

If split backups are used, run SSdbload once for each backup file that is loaded. Use the -i option only once, with the first SSdbload. The catalog must be loaded before the models.

To initialize and load with one operation, use the -i option with the -l option. This method is recommended for loading new database information from a file saved using the -c and -m options with SSdbsave.

Follow these steps:

1. Navigate to the directory containing the database.
2. Verify that neither the SpectroSERVER nor any other program that accesses the SpectroSERVER database is running.
3. Use caution if using .after files that are older than the current executables when initializing the database. Information that you are expecting may not be present if the .after files are older than the executables. Refer to SSdbload for an explanation of .after files.
4. Enter the following SSdbload:

   ```
   ../SS-Tools/SSdbload -i -l <filename>
   ```

   The database is initialized and then loaded with the information from the backup file you specify. The database is then ready for use.

More information:

SSdbload (see page 43)

Load Developer Information

Registered developers are permitted to load developer information (a file containing your developer ID) into the SpectroSERVER database using SSdbload with the -d option. You must reload the developer information after any initialization of the database, but you can only load this information once between initializations. Attempting to execute a second SSdbload using the -d option produces an error message and leaves the developer information unchanged.

Important! Load developer information into only one database, and perform all modeling catalog editing on that database. Modifications can then be propagated to other databases using SSdbload if required. However, loading the same developer information on multiple databases can result in duplicate model type, attribute, or relation handles. Duplicates make it impossible to correlate historical data across landscapes.

Follow these steps:

1. Navigate to the *$SPECROOT*/SS directory.
2. Verify that neither the SpectroSERVER nor any other program that accesses the SpectroSERVER database is running.

   Note: When first installed, SSdbsave is located in the *$SPECROOT*/SS-Tools directory. If its location is not set in a system path statement, you must use a full path, a relative path, move it, or link it to the same directory with the SpectroSERVER database.
3. Enter the following command:

   ```
   ../SS-Tools/SSdbload -d <developer filename>
   ```

Removing a Database Lock

Whenever a CA-developed program accesses a CA Spectrum database, it creates a lock file within that database directory. The lock file in the SpectroSERVER database is called .VNMDB.LOCK. The lockfile in the DDM database is called .DDMDB.LOCK. The lock file serves as an indicator that the database is currently in use and cannot be accessed by another CA-developed program.

The following CA-developed programs create a lock file:

- converter
- dbtool
- lh_set
- mte (Model Type Editor)
- reports
- SpectroSERVER
- SSdbdelete
- SSdbload
- SSdbsave

When a CA-developed program reaches normal termination, it removes the lock file. However, occasionally a CA-developed program is abnormally terminated, and the lock file is left behind. If not removed, the lock file inhibits execution of other CA-developed programs that access the database. When a lock file exists and another CA-developed program is started, the following message appears:

```
Database already locked by:<user id>,
by process:<process name>,
with process ID:<process id>,
on network node:<node name>,
which started at:<date/time stamp>
```

You can manually remove a .VNMDB.LOCK or .DDMDB.LOCK file that you believe was left behind by an abnormally terminated program, but first you should verify that the program is not running. If it is still running (and preventing a higher priority program from accessing the database), first attempt to terminate the program normally. If that is not possible, you can stop the program using a UNIX kill command (with no options) or the End Process button in the Windows Task Manager. Any of these methods removes the lock file.

If the program was abnormally terminated, you can remove the lock file manually by navigating to the *$SPECROOT*/SS or *$SPECROOT*/SS/DDM directory, as appropriate, and entering one of the following commands:

- rm .VNMDB.LOCK (to remove a lock from the SS directory)
- rm .DDMDB.LOCK (to remove a lock from the DDM directory)

Important! To maintain the integrity of your database, you must restore both your SpectroSERVER and DDM databases anytime you are forced to remove a lock file from either the *$SPECROOT*/SS or *$SPECROOT*/SS/DDM directories. The SpectroSERVER does not restart until you perform a database restore operation.

Also remember that .VNMDB.LOCK and .DDMDB.LOCK files do not prevent database access by non-CA-developed programs, and no error message is generated when such programs encounter these locks. Therefore, before allowing any non-CA tools or applications to access a CA Spectrum database, you should make sure that all CA Spectrum processes accessing that database are shut down.

More information:

Restoring Your Database (see page 19)

Import and Export Model Types

The dbtool utility allows you to import and export model types and associated objects (attributes, relations, and meta-rules) from the SpectroSERVER database.

More information:

dbtool (see page 28)

Recovering from Database Corruption

Several conditions, including hardware failures and power interruptions, can result in a corrupted database and generate an error message such as the following:

```
Sep 18 15:42:39 ERROR at CsSSDbRp.cc(642):
table open failed @ TableImpl.cc:674(0x2)
Db::open: No such file or directory
Could not open the database. VNM exiting.
Landscape not initialized. VNM exiting.
```

Database corruption may also have occurred when the SpectroSERVER or other applications generate error messages not readily attributable to specific causes such as the lack of a user model, the presence of a database lock, and so on.

If you have maintained a schedule of regular backups, you can usually recover from database corruption with minimal loss of information simply by using the SSdbload utility to initialize your database and reload the last known “good save” of your database-that is, one that includes both the modeling catalog and the models, and that was created prior to any indications of corruption. If you do not have a known good save of your database, or if application errors persist after you reload what you believe to be a good save, contact your CA Support representative.

More information:

SSdbload (see page 43)

SpectroSERVER Database Troubleshooting

This section describes some of the more common problems reported to CA Support by customers.

Using SSdbload to Load Old Objects after an Install/Upgrade

Using SSdbload to load old models, model types, or relations after an install or upgrade can cause serious database problems. The best way to avoid these problems is to understand how SSdbload operates. When you attempt to load a file using the -l option, SSdbload looks at the file to determine the types of objects it contains (which depend on the option flags that were used with SSdbsave to create the file). For each object type in the file (modeling catalog and/or models), SSdbload first removes any existing objects of that type from the database. For example, if you execute:

```
SSdbload -l somefile_c.SSdb
```

with the intention of updating a newly upgraded database with your old modeling catalog, you would replace the modeling catalog in the database with one from the save file. This could leave you with an unusable database since the upgraded modeling catalog may be needed to operate with the new version of CA Spectrum. Also, any modifications made with the Model Type Editor would be lost.

Inappropriate Use of the -i Option of SSdbload

Using the -i option of the SSdbload tool involves some potentially serious repercussions. When this option is used, all existing objects in the database are removed. The objects that are removed include model types, models, relations, attributes, rules, unarchived events, and unarchived statistics. Thus, if you execute the following command:

```
SSdbload -i -l somefile_m.SSdb
```

and the somefile_mSSdb file contains only models, the load fails. If you execute the same command with a file that contains only a modeling catalog (that is, one saved with the -c option only), the resulting database contains only the model types and relations that are loaded from the file. All other objects that previously existed in the database are lost, including all user-created models.

Copying Database Files between Operating Systems/Platforms

The database files (*.db, *.ix, log.*) are operating system and platform-specific. Copying a database from one operating system or platform to another is not supported. The only supported method of moving the data that is contained in these files is by running SSdbsave on the source computer and SSdbload on the destination computer. As long as these tools exist on each computer, and they were compiled for their respective servers, transfer of data succeeds.

Loading Order of the Modeling Catalog and Models

It is important to understand two facts about CA Spectrum models:

- Models are instances of model types, which are defined within the modeling catalog.
- Models and model types have a dependency relationship.

If the model type for a given model is not present or is not the correct version, an attempt to load a file that contains that model fails. Always load any prerequisite modeling catalog before loading models that were created from that modeling catalog.

Adding Modeling Catalog Objects to Multiple Databases with the Same Developer ID

When modeling catalog objects (model types, relations, and attributes) are added to a database using the Model Type Editor, they are assigned a unique handle. The handle comprises the active developer ID plus the next available sequential number for that object type. However, each database assigns and maintains these sequential numbers independently. Therefore, if new objects are added to another database using the same developer ID, the same handle can represent different objects in each database. In a distributed SpectroSERVER environment, this kind of conflict makes it impossible to correlate historical data across landscapes.

To avoid a problem with duplicate handles, only modify modeling catalog information in one database. Save the changes using SSdbsave with the -c option, and then propagate the changes to other databases as required using SSdbload with the -l option.

UNIX File Access Permissions on Database Files and Directory

For proper operation of the SpectroSERVER and database tools, you must have write permission to all database files (*****.db, *****.ix, log.*****) and to the directory that contains those files. If you receive a message from one of the database tools or the SpectroSERVER that indicates that the program was "Unable to open lock file," check the permissions on the database files and directory.

CA Spectrum Database Tools

This section describes the following database tools, utility programs, or scripts:

- db_remove
- dbtool
- HostUpdate

- MapUpdate
- reports
- SSdbload
- SSdbsave
- Database Model Conversion Tool (DBconv)

Note: When first installed, all of the tools listed above are located in the *$SPECROOT*/SS-Tools directory. If location of a tool is not set in a system path statement, either use a full path, or move it into the same directory with the SpectroSERVER database.

db_remove

This utility removes obsolete model types (and their associated originating attributes and meta-rules) from the modeling catalog of the SpectroSERVER database.

Note: This tool is not intended for general usage. Do not run db_remove unless you have specific instructions or technical bulletins that were issued by CA. For more information, contact a CA Support representative.

This utility has the following format:

```
db_remove [-debug] [<MTH_FILE>]
```

-debug

Enables verbose informational message output.

<MTH_FILE>

Specifies the text file containing a whitespace-separated list of the model type handles to remove from the database.

Errors

Warning: Model Type 0x???????? - "*model type name*" could not be removed.

You see this message in the unlikely event that you (or another vendor or partner) have derived new model types from the model types that are being removed. In that situation, run the Model Type Editor to see what is derived from this model type. Then move the derived model type or types to a different, non-obsolete derivation point. Finally, rerun the db_remove tool with the original MTH_FILE.

Warning: Model Type 0x???????? - "*model type name*" is currently referenced in a default Attribute value. The reference occurs in Attribute 0x???????? - "*attribute name*", at Model Type 0x???????? - "*model type name*".

You see this message if you (or another vendor or partner) have added a reference from a default value of an attribute (of MODELTYPE_HANDLE or list-of MODELTYPE_HANDLE) to the value of the model type handle that is being removed. This message does not prevent the removal of the model type. However, investigate the default value of the attribute. Depending on how it is used, either remove the reference or replace it with an appropriate, non-obsolete model type handle.

dbtool

As your network grows, you typically add model types to your database. In some cases, you may want to add the new model types without installing a new database. You can use the dbtool utility to accomplish the following related tasks:

- Export model types from the permanent catalog in the SpectroSERVER database.
- Import model types into the permanent catalog in the SpectroSERVER database.
- Display (dump) the contents of an export file.

  The dump function sends the output to the standard output for the workstation, normally your display screen. However, you can also send the output to a file or to a printer.

The Model Type Editor includes similar functions for exporting and importing model types. However, the dbtool utility lets you specify multiple files as command-line arguments. As a result, it is more useful for batch-processing a set of files.

Note: For more information, see the *Model Type Editor User Guide*.

Depending on the function (export, import, or dump), the dbtool utility uses one of the following types of files:

- A model type list file with file extension .m. This file contains a list of model type ID codes. Each ID appears on a separate line (separated by return characters) or is separated from others on the same line by spaces. You can create these files with your preferred shell text editor.
- An extract file with file extension .e. You can create these binary files, also referred to as catalog files, using the dbtool export function or using the Model Type Editor. They contain all the information in the database that is relevant to the model types that are listed in the associated *.m file (if one is produced by dbtool). Or, if the Model Type Editor produces the catalog file, they contain the information that is relevant to all of the selected model types.

  Catalog files are the means of transferring model types from one system to another using media such as email files, DVDs, and others.

Important! Before you run dbtool, shut down the SpectroSERVER and any other program that accesses the SpectroSERVER database, including third-party programs. Always run dbtool from the directory that contains the SpectroSERVER database.

The dbtool utility loads the database so that symbolic names can be used when possible.

This utility has the following format:

```
dbtool          [dump <file>.e [ <file>.e ...] ]
dbtool          [dump_mt <file>.e [ <file>.e ...] ]
dbtool          [import <file>.e [ <file>.e ...] .xml [ <file>.xml ...] ]
dbtool          [export <file>.m [ <file>.m ...] ]
```

dump

Displays the contents of the specified .e file (catalog file) in readable form. Unlike the dump_mt argument, this argument includes not only model type information in the output, but also attribute, relation, and meta-rule information. You can specify multiple catalog files to dump if desired.

Note: This option reports the attribute names for only those attributes that originate in the model type being exported. It does not provide output for changes to extended flags, OID Prefix values, or OID Reference values.

This argument does not operate on the database.

dump_mt

Displays information about the model types that are listed in the specified .e file in readable form. You can specify multiple catalog files to dump if desired. Unlike the dump argument, this argument does not include attribute, relation, or meta-rule information in the output.

This argument does not operate on the database.

import

Imports the model types, attributes, relations, and meta-rules in the specified .e or .xml file into the database. You can specify multiple catalog files to import if desired.

Importing model types that are already present in the database produces a warning message about that model type being redefined. This warning message can be ignored.

export

Uses the specified .m file to create a catalog file that contains the model types that are specified by model handle in the .m file. You can specify multiple .m files to create multiple catalog files.

The model type handle entries in the .m files must be hexadecimal integers that are preceded by “0x” (zero followed by lowercase “x”). In addition, the entries must be separated by at least one space character or by a newline character.

help

Displays usage information about the command.

More information:

Direct the Contents of a Catalog File to an Output Device (see page 33)

Export Model Types Using dbtool

You can export model types using the dbtool utility.

Note: You can only export the model types, attributes, and relations (and associated meta-rules) that were created using the developer ID that is currently loaded in the SpectroSERVER database. That developer ID is the “owner” of these objects. If you are not the owner of a model type or other object that is included in an export operation, the operation terminates with an error message.

A catalog file (.e file) that the export process produces contains the following information:

- The attribute descriptors that originated in the model types being exported.
- The attribute descriptors that have been specialized (for example, by specifying a default value to override an inherited one).
- The relations and associated meta-rules in which the model types and any ancestor model types participate as an antecedent or a predicate.

“Fringe” model types have at least one base model type that is not exported in the same *.e file. These types differ in that they include the attribute values and extensions that are inherited from the base model type that is not included. The inclusion ensures the availability of those values and extensions.

Follow these steps:

1. Change your working directory to the directory that contains your database.
2. Verify that the dbtool utility is either in the current directory or in your system search path. Or, when you later invoke the dbtool command (step 4), use an appropriate relative or absolute path name.
3. Create the .m file (see page 31) that specifies the model types to export.
4. Export the model types using the following command:

   ```
   ../SS-Tools/dbtool [export<filename_11>.m [<filename_2>.m … ] ]
   ```

   The following command serves as an example:

   ```
   ../SS-Tools/dbtool export smart_hub.m smart_router.m
   ```

   The command in this example exports the model types that are defined in smart_hub.m and writes the resulting output to a file named smart_hub.e. It then processes smart_router.m in the same way.

More information:

dbtool (see page 28)
Create the .m File (see page 31)

Create the .m File

The export function of the dbtool utility uses one or more *.m files to specify the model types to be exported by model type handle. Before running an export using dbtool, create one or more of these files using your preferred shell text editor.

Typically, the list of model types to export includes any base model types that are required by the model types being exported and that do not exist in the destination database. However, dependencies normally are limited to certain commonly used base model types that are contained in one or more “core” catalogs. These catalogs are included as part of the basic CA Spectrum system.

Follow these steps:

1. Enter model handle entries as hexadecimal integers preceded by “0x” (zero followed by lowercase “x”). 0x should be followed by 8 digits that consist of your 4-digit developer ID followed by a 4-digit sequence number.
2. Separate model handle entries, either by at least one space character or by a newline character.

3. Name the file that lists the handles with the .m file extension.

   For example, the following list specifies five model types (created using the default developer ID) for export:

   ```
   0xffff0003 0xffff0008
   0xffff0017 0xffff0023
   0xffff0045
   ```

   This example would produce a *.e file that contains the database information for the 3rd, 8th, 17th, 23rd, and 45th model types created under the currently loaded developer ID (in this case, the default developer ID).

Important! Database access must be limited to a single application at a time. When dbtool is in use, all other applications (including OneClick and the Model Type Editor) are denied access. While CA-developed programs automatically lock out other CA products, database corruption can occur if this caution is bypassed by third-party applications.

Import Model Types Using dbtool

You can use the dbtool import function to import model types into the SpectroSERVER database from one or more catalog files (.e files) that you previously created using the dbtool export function or using the Model Type Editor.

Database access must be limited to only one application at a time. When dbtool is in use, all other applications (including OneClick and the Model Type Editor) must be denied access. While CA-developed programs automatically lock out other CA products, corruption of the database can occur if this caution is ignored or bypassed with respect to any third-party application programs.

Follow these steps:

1. Back up the SpectroSERVER database.
2. Change your present working directory to the directory that contains your database.
3. Verify that the model type files to import are either in the current directory or in your system search path.

4. If your database is initialized (files with .d and .k extensions), proceed to the next step. Otherwise, initialize the database and load the core model type derivation, using the SSdbload -i -l utility command.
5. Import the model types using the following command:

   ```
   ../SS-Tools/dbtool import [<filename_1.e> … <filename_n.e>]
   ```

   The following command serves as an example:

   ```
   ../SS-Tools/dbtool import rmon1.e rmon2.e
   ```

   After the contents of the last source file are imported, a message indicates that the operation is complete.

More information:

Database Backups (see page 11)
SSdbload (see page 43)

Direct the Contents of a Catalog File to an Output Device

When you run the dbtool utility with the dump argument or dump_mt argument, the utility sends the output to the workstation's standard output device, which normally is your display screen. If you want to redirect the output to a file or printer directly, provide standard UNIX piping commands in the command line to redirect the output as desired.

Example 1

The following command dumps the output of a catalog file named rmon.e to the workstation's standard output device:

```
dbtool dump rmon.e
```

The output would appear on your workstation as a one-time display, and, if it were too long, you would only see the end of the output (that is, however many lines of text your display is capable of showing).

Example 2

The following command dumps the output of the same catalog file as the preceding example, but the output is sent to your workstation screen as an incremental display file, showing you the first screen:

```
dbtool dump rmon.e | more
```

To increment through successive lines while viewing the screen, you would press Return; to increment to the next screen, you would press the spacebar.

Example 3

The following command dumps the output of the same catalog file as the preceding example, but the output is written to an ASCII file name filesave.out:

```
dbtool dump rmon.e > dumpouts/filesave.txt
```

The file is created in the *$SPECROOT*/SS/dumpouts directory, which you must have created previously.

Example 4

The following command dumps the output of the same catalog file as the preceding example, but the output is sent as a print file to the printer designated by <ptr>:

```
dbtool dump rmon.e > lpr -P<ptr>
```

Troubleshoot dbtool

The following are errors that you might encounter when running dbtool:

database open failed

The SpectroSERVER database is not present in current directory.

database files are missing are missing read and/or write permissions

You do not have your developer ID loaded into the database, or you are not a valid user with respect to the specified database.

Database already locked by:<user id>
by process:<process name>
with process ID:<process id>
on network node:<node name>
which started at:<date/time stamp>

The database is locked by another process.

More information:

Load Developer Information (see page 21)
Removing a Database Lock (see page 22)

MapUpdate

MapUpdate is a utility used to modify and display the landscape map. This program is located in the *$SPECROOT*/SS-Tools directory and performs the following tasks:

- Removes a landscape entry from a landscape map
- Displays the current landscape map

If you are removing a secondary SpectroSERVER, be sure to run MapUpdate to remove the secondary SpectroSERVER from the list of loaded landscapes on the primary SpectroSERVER. If you are using a timeout value for your landscape entries, run MapUpdate -remove before the landscape entries on the secondary SpectroSERVER time out. Otherwise, you may not be able to properly remove the secondary SpectroSERVER from the primary SpectroSERVER's list of loaded landscapes.

Note: In previous CA Spectrum releases, landscape entries timed out after one hour by default and were removed automatically from the landscape map. Starting in CA Spectrum 9.2.2, landscape entries do not time out by default. You must remove an entry from a landscape map manually, using MapUpdate. You can use a timeout value for a landscape entry. For more information, see the *Distributed SpectroSERVER Administrator Guide*.

This command has the following format:

```
MapUpdate [-remove LANDSCAPE_HANDLE] [-precedence PRECEDENCE] [-view]
```

-remove *LANDSCAPE_HANDLE*

Specifies the handle of the landscape to remove. You must first shut down the SpectroSERVER that you want to remove. In addition, if you are removing a secondary SpectroSERVER, the primary SpectroSERVER must be running.

Default: 0x400000

-precedence *PRECEDENCE*

The precedence value of the landscape to remove.

-view

If supplied, displays the current landscape map.

HostUpdate

The HostUpdate utility, located in the SS-Tools directory, lets you remove all landscape map entries that are partitioned by host for the host that you specify.

```
HostUpdate [-remove HOSTNAME] [-view]
```

-remove

Removes all entries for the host that you specify by entering the hostname.

Note: Host entries automatically time out from the landscape map. Therefore, this option is probably unnecessary. However, you can use this option if the automatic timeout mechanism fails, or if you want to remove the entry before the timeout interval.

-view

Displays all entries that are partitioned by host.

reports

This command-line utility, which is located in the SS-Tools directory, lets you display a listing of selected objects in the current modeling catalog. Run reports from the directory where the SpectroSERVER database is installed.

The command locks the database during a report generation sequence. As a result, you can run only one report at a time.

Note: Before executing reports, you must shut down the SpectroSERVER and any other program that accesses the SpectroSERVER database, such as a VNM, the Model Type Editor, or any third-party utilities. Otherwise, database corruption can occur.

This utility has the following format:

```
reports [-mtype <name_pattern>] [-relation <name_pattern>] [-handle <handle>]
[-attrflags <defglmprsvw>] [-fields <cdefgimnotvGE>] [-types <bierdtcgmMRlaoIAOTU>]
[-recursive]
[-invisible] [-lists] [-nolists] [-groups] [-help]
```

-mtype

Specifies the model types to include in the report. All model types with names that contain the specified text string are included. For example, if the model type is IRM, the report lists sections for Hub_CSI_CIRM, Hub_CSI_IRM2, Hub_CSI_SIRM, and all other model types whose names contain the “IRM” character string.

Note: You can substitute a period (.) as a wildcard for “all applicable” entries. Use the wildcard option sparingly, as it takes a long time to display the report. Run the following command from the *$SPECROOT*/SS directory to display a report on all model types without the attribute information:

```
../SS-Tools/reports -mtype . -fields e
```

-relation

Specifies the relation to include in the report. The output lists how one model type relates to another.

Note: You can substitute a period (.) as a wildcard for “all applicable” entries. Run the following command from the *$SPECROOT*/SS directory to display a report on all relations:

```
../SS-Tools/reports -rel .
```

-handle

Specifies the hexadecimal handle of the model type(s) to include in the report, with or without the preceding “0x” prefix. All model types with handles that contain the specified text string are included. This argument is similar to the mtype argument except that it accepts a model type handle rather than a name string.

For example, if the specified handle is 0x180027, the report includes sections for the 27th model type that was created under the 0x180000 Developer ID. If the handle is 180, conversely, the report includes not only all model types that were created under that developer ID, but also any other model types with the same three digits anywhere in their handles.

-attrflags defglmprsvw

Filters the report to include only the attributes for the model types in the report that have one of the specified flags set.

d

Database

e

External

f

Global

g

Guaranteed

l

Logged

m

Memory

p

Polled

r

Readable

s

Shared

v

Preserve

w

Writable

Note: Entering two consecutive single-quote or double-quote characters instead of a list of flags is equivalent to entering all of the flags. If you include this argument without specifying a value, the reports utility fails with an error. The help file is displayed. For detailed descriptions of these attribute components, see the *Model Type Editor User Guide*.

-fields cdefgimnotvGE

Filters the report to include only the specified attribute components (fields).

c

Creator model type

d

OID Reference

e

No attribute info printed

f

Flags

g

Polling Group

i

Attribute ID

m

Manifest Constant Name

n

Name with developer ID

o

OID Prefix

t

Type

v

Default value

G

Group ID

E

Enumerated Value List

Note: Entering two consecutive single-quote or double-quote characters instead of a list of flags is equivalent to entering all of the flags. If you include this argument without specifying a value, the reports utility fails with an error. The help file is displayed. For detailed descriptions of these attribute components, see the *Model Type Editor User Guide*.

-types bierdtcgmMRlazoIAGOTU

Filters the report to include only attributes of the specified types:

b

Boolean

i

Integer

e

Enumeration

r

Real

d

Date

t

Time

c

Counter

g

Gauge

m

Model Handle

M

Model Type Handle

R

Relation Handle

l

Landscape Handle

a

Attribute ID

z

Text String

o

Object ID

I

IP Address

A

Agent ID

O

Octet String

T

Tagged Octet

U

64- bit Unsigned Integer

-recursive

Includes descendant (child) model types in the report.

-invisible

Includes the following model types in the report:

- All visible model types (Visible model type flag is set to true) that match the search pattern
- All invisible model types (Visible model type flag is set to false) that were created by the currently loaded developer ID and that match the search pattern.

-lists

Includes only attributes that allow multiple values in the report.

-nolists

Does not include attributes that allow multiple values in the report.

-groups

Includes model type groups in the report.

-help

Displays usage information on the command.

More information:

Run a Model Type Report (see page 41)

Run a Model Type Report

A model type report lists the attribute information for a particular model type. To run a model type report, use the following syntax:

```
../SS-Tools/ \
reports [-mtype <model type>][-attrflags w][-fields][-invisible]
```

or

```
../SS-Tools/ \
reports [-handle <model handle>][-attrflags ""][-fields][-invisible]
```

As an example, the following command generates a report on the HUB_CSI_IRM2 model type:

```
reports -mtype HUB_CSI_IRM2 -attrflags e -fields don -invisible
```

The report lists all attributes with the External flag set. For each attribute that is included, it identifies the OID Reference, OID Prefix, and Name (with Developer ID). The report also includes the complete list of base model types (parent model types) for the specified model type; the base model types are listed in order of inheritance.

The report is sent to the standard output device for the workstation.

More information:

Direct Reports to an Output Device (see page 41)

Direct Reports to an Output Device

The reports utility sends the output to the standard output device for the workstation, normally the display screen. To redirect the output to a file or to a printer directly, provide standard UNIX piping commands in the command line.

Example 1

The following command generates a report on the HUB_CSI_IRBM model type:

```
reports -mtype HUB_CSI_IRBM -attrflags e -fields n
```

The report includes the standard header section to identify the model type developer, its name and handle, the state of the six attribute flags, and the identity of direct base model types. The remainder of the report is restricted to external-flagged attributes, and the line items in the report list only the attribute names.

The report appears on your workstation screen as a one-time only display. If the report is too long, you only see the end of the report (the maximum lines of text that your display can show).

Example 2

The following command generates the same report, but sends it to your workstation screen as an incremental display file, showing you the first screen:

```
reports -mtype HUB_CSI_IRBM -attrflags e -fields n | more
```

The Return key lets you increment through successive lines while viewing the screen; the spacebar lets you see the next screen.

Example 3

The following command generates the same report, writing it to an ASCII file named REPORT_1 in your current directory:

```
reports -mtype HUB_CSI_IRBM -attrflags e -fields n > REPORT_1
```

If the named file exists in the current directory, it is overwritten by the new report.

You can precede the report filename with a directory path.

Example 4

The following command generates the same report, sending it as a print file to the printer that is designated by <ptr>:

```
reports -mtype HUB_CSI_IRBM -attrflags e -fields n > lpr -P<ptr>
```

Run a Relation Report

A relation report lists how one model type relates to another according to the selected relation. To run a relation report, use the following syntax:

```
 ../SS-Tools/reports -rel <relation>
```

where <relation> is the name of the relation to include in the report. Specify a single relation at a time.

As an example, the following command generates a report on the Connects_to relation:

```
../SS-Tools/reports -rel Connects_to
```

The report lists each sequential rule in the Connects_to relation, and it is sent to the standard output device for the workstation. For information on writing reports to a file or sending them to a printer, see Direct Reports to an Output Device (see page 41).

More information:

Direct Reports to an Output Device (see page 41)

SSdbload

This utility program, located in the *$SPECROOT*/SS-Tools directory, lets you restore a SpectroSERVER database with previously created backup files, load developer ID information, or set the precedence value for a SpectroSERVER in a fault tolerant environment.

Note: For more information about establishing fault tolerance, see the *Distributed SpectroSERVER Administrator Guide*.

You must shut down the SpectroSERVER and any other program that accesses the SpectroSERVER database before executing SSdbload.

This utility has the following format:

```
SSdbload [-quiet] [-initialize] [-developer <DEV_INFO_FILE>] [-load] [-models]
[-catalog] [-replace <PRECEDENCE>] [-add <PRECEDENCE>] [-port <PORT_NO>] [-showmap]
[-version]
[-extension] [-new_primary <NEWHOSTNAME>]
[<SAVE_FILE>]
```

Where the first letter of an argument name appears in bold type, you can use the letter only, rather than entering the entire string.

-quiet

Disables prompting (interactive mode). Useful for running load commands from within a script.

-initialize

Initializes the database by removing the modeling catalog, all of the models, and all unarchived Events and Statistics Log records.

Important! If you use the -i (initialize) option, you must restore at least the catalog. If split backups are used, you must run SSdbload once for each backup file that is loaded. The -i option must only be used once, specifically, with the first execution of SSdbload. Load the modeling catalog before the models.

-developer

Loads the developer information file you specify using the <DEV_INFO_FILE> variable.

Note: Load developer information into only one database, and perform all modeling catalog editing on that database. Using the same developer information on more than one database can result in duplicate model type, attribute, and relation handles.

The -d option can be used to load developer information, but only once. Attempting to execute a second SSdbload using the -d option produces an error message and leaves the developer information unchanged.

-load

Loads the database with objects from the save file that you specify using the <SAVE_FILE> variable.

Important! The -l (load) option is not designed to add models or modeling catalog components into an existing database. It does not augment the contents of the database; rather, it first removes any existing information of the same type as that contained in the save file, and then loads the contents of the savefile. For example, if the save file were created using the SSdbsave -m option and contained only models (and not the modeling catalog), the SSdbload -l option would not affect the existing modeling catalog. However, it would remove any existing models and replace them with models contained in the save file.

-models

Loads models from the save file you specify using the <SAVE_FILE> variable.

-catalog

Loads the modeling catalog (model types, relations, and rules) from the specified save file.

-replace

Used only in fault tolerant environments to replace the precedence value currently assigned to a particular SpectroSERVER with a new value you specify using the <PRECEDENCE> variable.

-add

Used only in fault tolerant environments to assign a precedence value to a particular SpectroSERVER using the <PRECEDENCE> variable.

-port

Used with either the -add or -replace arguments to specify the port number for the SpectroSERVER. If you do not specify a particular port using the <PORT_NO> variable, the port specified for the comm.port resource in the .vnmrc file is used by default).

-showmap

Prints landscape map information showing which landscapes are loaded on which servers and at which precedence levels.

-version

Displays the version of SSdbload and the version of the saved file that you are attempting to load. If these versions are incompatible, an error message is displayed (even if you are operating in quiet mode), and the file is not loaded.

-extension

Disables file extension enforcing.

-new_primary

Must be used when you are loading a database that was saved on another landscape. Use the <NEWHOSTNAME> variable to name the landscape using the name of the host where you are loading the database. Otherwise, the landscape is given the host name from the SpectroSERVER where it was originally saved.

<SAVE_FILE>

The name of the backup file to be loaded. The backup can be a file that was saved using either the SSdbsave utility or the Online Backup feature. It can also be one of the save files created by the CA Spectrum installation program.

Each successful execution of this program creates two savefiles in the *$SPECROOT*/SS directory, each containing a copy of the modeling catalog being installed. The first is a date-stamped file with the .after extension. A copy of the .after file is then created and named “legacy.SSdb”, overwriting any previously existing legacy.SSdb file.

Use the legacy.SSdb file to re-initialize your database with the most recently installed modeling catalog. Use the .after files as necessary to restore the catalog associated with a particular installation.

A sequential counter following the date stamp in the .after file name lets you distinguish among multiple files created on the same day, for example:

```
db_20001014,1.after.ssdb
db_20001014,2.after.SSdb
db_20001014,3.after.SSdb
```

If the .after file used to initialize your database is older than the current executables, expected information may not be present in the database and is not accessible through OneClick.

Example

To initialize a database and load the modeling catalog (model types and relations) and model information from a previously saved file named “db_950318_cm”, you would use the following command:

```
SSdbload -il db_950318_cm
```

Errors

Can't open database.

The SpectroSERVER database is not present in the current directory.

Database already locked by: *<user id>*
by process: *<process name>*
with process ID: *<process id>*
on network node:*<node name>*
which started at: *<date/time stamp>*

The database is locked by another process.

Save file version:<version_number>.
SSdbload version: <version_number>.The save file CANNOT be loaded by this version of SSdbload.
This save file cannot be loaded by version <version_number> of CA Spectrum. It was saved as version <version_number>.

SSdbload does not let you load the savefile if a version incompatibility is detected. This error message is generated if you have specified the [-version] argument. It is displayed when appropriate even if you specify the -quiet option.

This save file cannot be loaded by version <version_number> of CA Spectrum. It was saved as version <version_number>.

SSdbload does not let you load the savefile if a version incompatibility is detected. This error message is generated if you have specified the [-version] argument. It is displayed when appropriate even if you specify the -quiet option.

Note: If nonfatal attribute descriptor errors are detected during the load, SSdbload creates a log file named SSdbload.log in the directory containing the database. The following is an example of a nonfatal attribute entry in this file:

Important! <SSdbload path>/SSdbload Can't read attribute 10004 in model 400000

More information:

Removing a Database Lock (see page 22)

Changing Host Names

If you change the host name of a single SpectroSERVER host, you do not have to use SSdbsave before you make the change and then use SSdbload afterward. The database is automatically changed to reflect the new host name, and the following message appears in the Control Panel when you restart the SpectroSERVER:

```
This database was previously loaded on <old hostname> port <old port number>, but is
now being loaded on <new hostnames> port <new port number>.
```

However, in a fault-tolerant environment that includes one or more backup SpectroSERVERs, the servers recognize their relationship to one another by a *precedence* value that is associated with their host names. Therefore, to preserve the fault-tolerant relationship, use SSdbsave and SSdbload in the following order to change the host name of the primary (or secondary) SpectroSERVER:

1. Save the database using SSdbsave with the -cm option.
2. Change the host name.

3. Reload the database with the save file that was created in Step 1 by running SSdbload with the -il and -replace options. The reload lets the database associate the new host name with the existing precedence value:

   ```
   SSdbload -il -replace <precedence> <save file>
   ```

More information:

SSdbsave (see page 48)
SSdbload (see page 43)

SSdbsave

This utility program, located in the *$SPECROOT*/SS-Tools directory, allows you to create a backup copy of an existing SpectroSERVER database's modeling catalog (model types and relations) and/or the actual models and associated data it contains.

Before executing SSdbsave, you must shut down the SpectroSERVER and any other program that accesses the SpectroSERVER database.

This utility has the following format:

```
SSdbsave [-quiet] [-extension] [-version]
[-catalog] [-models] <SAVE_FILE>
```

Where the first letter of an argument name appears in bold type, you can use the letter only, rather than entering the whole name.

-quiet

Disables interactive/verbose mode.

-extension

Disables file extension enforcement.

-version

Displays the version of SSdbsave. The version number is included with the save file. When using this argument while saving a file, a message indicates the version of SSdbload that can be used to load the saved file.

-models

Includes models in the save file.

-catalog

Includes the modeling catalog (model types, relations, and rules) in the save file.

<SAVE_FILE>

Specifies the name of the destination file for the saved database. The suggested naming scheme for save files is to incorporate a date stamp as well as the option flags that are used for the save. For example, a file named db_20121014_cm would indicate a backup performed on October 14, 2012 using both the “c” (catalog) and “m” (models) options.

Sample Output

The following is an example of the output generated when running SSdbsave using the -models and -catalogs arguments (../SS-Tools/SSdbsave -mc Mar16_2013DB). The name of the database file to be saved is Mar16_2013DB. The last line of this output indicates the version number of SSdbload that can be used to load this saved file.

```
Number of Model Types saved: 3493
Number of Relations saved: 92
Number of Models saved: 103
SSdbsave has successfully saved the database model and catalog information as
'Mar16_2013DB.SSdb'.
This file can be loaded with version 7.0.0.000 of SSdbload.
```

Errors

Can't open database

The SpectroSERVER database is not present in current directory.

Database already locked by: *<user id>*
by process: *<process name>*
with process ID: *<process id>*
on network node: *<node name>*
which started at: *<date/time stamp>*

The database is locked by another process.

SSdbsave: Warning. Expected attr: 0x999999 not found, for model: 0x999999, of Model Type: 0x999999 Cs Whatever MT. Processing continues

SSdbsave displays this message when attempting to save a model attribute value for which there is no corresponding attribute descriptor. This informational message appears if a user or a developer removes an attribute.

More information:

Removing a Database Lock (see page 22)
SSdbload (see page 43)

Database Model Conversion Tool (DBconv)

This utility program, located in the *$SPECROOT*/SS-Tools directory, lets you convert a set of models within a CA Spectrum database from one model type to another. DBconv can also be used to rediscover the applications and reconfigure the interfaces in a set of models.

The utility has the following format:

```
DBconv [-file=]<Input File Name>
[-src_mth=]<Source Model Type Handle>
[-dest_mth=]<Destination Model Type Handle>
[-landscape=]<Landscape Handle>
[-rediscover=]Reason:<d><i>
[-all_landscapes] [-test=]<Test Level> [-quiet] [-debug]
```

DBconv and Configurations

DBconv can take its configuration from an input file (specified on the command line) or from the command line itself. To get a list of command line options, run DBconv with no options.

-file

Specifies the input file to be used.

-src_mth

Specifies the model type handle from which to convert. If -rediscover is specified, specifies the type of models that are rediscovered.

-dest_mth

Specifies the model type handle to which to convert. Overrides the setting in the input file. This option is ignored if -rediscover is specified.

-landscape

Specifies the landscape handle to search for models.

Default: 0x400000

-all_landscapes

Specifies that all landscapes are searched for models.

-test

Test level 0 (the default) means the conversion actually takes place. Test level 1 means that DBconv stops processing just after validating the command line and input files. Test level 2 means that the old models are not deleted, and new ones are not created.

-quiet

Specifies that there should be no output except for error messages.

-debug

Specifies that there should be extra output.

-rediscover

If this option is specified, models that are found are not converted. Rather, one or more of the following specified actions are performed on each found model:

d

Destroy all application models for each device model found.

r

Send a Rediscover Application action to each device model found.

i

Send a Reconfigure Interfaces action to each device model found.

Note: The actions above can be specified in any order on the command line. However, they are always executed in the order shown here.

More information:

Examples of DBconv Command Line Usage (see page 51)
Using DBconv with an Input File (see page 52)

Examples of DBconv Command Line Usage

The following are examples of typical uses of the DBconv tool on the command line:

- Run a conversion using only the specified input file:

  ```
  $ DBconv -file=config.dbc
  ```

- Convert all GnSNMPDev models to Smart Switch Routers, in all known landscapes:

  ```
  $ DBconv -src_mth=0x3d0002 -dest_mth=0x2c60000 -all_landscapes
  ```

- Convert all GnSNMPDev models to Smart Switch Routers, in landscape 0x400000:

  ```
  $ DBconv -src_mth=0x3d0002 -dest_mth=0x2c60000 -landscape=0x400000
  ```

- For every SmartSwitch Router in every landscape, destroy all applications, rediscover the applications, and then reconfigure the interfaces:

  ```
  $ DBconv -src_mth=0x2c60000 -rediscover=rdi -all_landscapes
  ```

- For every SmartSwitch Router in landscape 0x400000, reconfigure the interfaces:

  ```
  $ DBconv -src_mth=0x2c60000 -rediscover=i -landscape=0x400000
  ```

Note: When it runs, DBconv goes through a short initial phase and a longer second phase, noting in the window which phase is in effect. Models that cannot be contacted are not converted; in each case, an error message appears. Some possible causes of these error messages are:

- Devices represented by specified model types have lost contact
- Specified model types do not exist in the database
- The SpectroSERVER is not responding
- The SpectroSERVER does not have a model for your user ID

Using DBconv with an Input File

DBconv can be used with or without an input file. If you want to use an input file, use the template.dbc file in the <*$SPECROOT*>/SS-Tools directory as a starting point.

The input file consists of several sections that allow you to customize the conversion with far more precision than the command line arguments allow. It is made up of several sections, all optional except for the Configuration section. Use the format of the sections and case as specified here. Blank lines and lines starting with a '#' character (pound sign) are ignored.

Elements of a DBconv Input File

The following section describes the required and optional contents of an input file to be used with DBconv.

Configuration (Required)

This section contains the single configuration items, described below.

Note: Only the Source_Model_Type, Destination_Model_Type, and Landscape_Handle fields in the Configuration section of the input file are mandatory. All other fields are optional.

Example:

```
Configuration {
Source_Model_Type = 0x3d0002
Destination_Model_Type = 0x2c60000
Reconnect_Sleep_Time = 90
Dont_Change_Discovery_Attributes = true
Models_To_Convert = 0
Landscape_Handle = all
Is_Obsolete_Model_Type = false
Relation_Section_Ignores = false
}
```

- Source_Model_Type = <old model type handle>

  The model type handle to convert from. It must be hexadecimal and prefixed with “0x”.

- Destination_Model_Type = <new model type handle>

  The model type handle to convert to. It must be hexadecimal and prefixed with “0x”.

- Landscape_Handle = <landscape handle>

  Specifies the landscape where the conversion is done. If 'all' is specified, then DBconv switches to enterprise mode and converts model types from all landscapes. If 'selection' is specified, then all of the landscapes specified in the Landscapes { } section are used.

- Models_To_Convert = <max models to convert>

  If this line is specified with a value greater than zero, database conversion converts only the specified number of models of the old type. If the value is 0, all models are converted.

- Reconnect_Sleep_Time = <sleep time>

  Allow for a configurable sleep period, in seconds, between the time Model_State changes to active and port reconnections are attempted. The default value is 60 (seconds).

- Reconnect_Interval = <seconds>

  After the Reconnect_Sleep_Time has run, DBconv attempts to reconnect the interfaces and ports. If a model after this time has still not gone active (some device models do not go active until all modules and applications have been created), then DBconv waits for the specified amount of time before trying again. The default value is 30 (seconds).

- Reconnect_Interval_Count = <count>

  This option controls the number of times that DBConv attempts to reconnect interfaces and ports. The default value is 30 (times).

- Dont_Change_Discovery_Attributes = <true/false>

  If this is false, the conversion program modifies Discovery-related attributes in the old model type. This is to force Discovery to use the new model type instead of the old one. This defaults to true if (a) the old model type is GnSNMPDev, or (b) both the old model type and the new model type are the same. The default value is false.

- Is_Obsolete_Model_Type = <true/false>

  This switch converts models where the Obsolete flag in the model type is set (when this is set, you cannot read any attributes). In this case, no attributes are read/validated/written during the conversion. No interfaces or ports are accessed. However, the model's relations are processed. This means that if a device model is converted with this flag, then the new model is created without Name, IP Address, and so on. The default value is false.

- Relation_Section_Ignores = <true/false>

  This switch interprets the Left_Relationships and Right_Relationships sections. If the value is false, then only the associations specified in the Left/Right sections are restored to the device model (none if both the Left/Right sections are empty). If the value is true, then all associations except for those specified are restored. This does not affect CONNECTS_TO on the left of the device model because DBconv always tries to restore this to maintain in which view the device is modeled.

- Convert_Scm_Configs = <true/false>

  This switch is to determine if SCM Configurations are converted along with the model types. If this is set to true, then SCM configurations that were on behalf of models/model types to be converted are converted as well. The shared SCM configurations are converted to the destination model type, and the non-shared SCM configurations are converted to the destination model handle and model type.

  Note: Host SCM configurations are transferred effectively, but attribute configurations may be lost during conversion.

- Convert_Sanm_Policies = <true/false>

  This switch determines if SANM Policies are converted along with the model types. If this is set to true, then any reference to the source model type name in any policy is converted to the destination model type name.

- Landscapes

  List of landscapes to convert. For DBconv to take any notice of this section, the Landscape_Handle entry in the Configuration section should be set to 'all'.

  Example:

  ```
  Landscapes {
  0x400000
  0x80c00000
  }
  ```

- Model_Handles

  List of models to convert. If you want only a selection of models to be converted, enter their model handles in this section. All other models are ignored.

  Example:
  ```
  Model_Handles {
  0x80c00be6
  0x80c00be8
  }
  ```

- Transfer_Attributes

  List of attributes to transfer from old models into new models. Make sure that these attributes are valid for old as well as new model types. Note that the Network Address and Community Name attributes are always transferred so they need not be specified in this section.

  Example:
  ```
  Transfer_Attributes {
  0x001006e
  0x00010024
  }
  ```

- Transpose_Attributes

  List of attributes to transpose from old models into new models. Make sure that these attributes are valid for old as well as new model types. The old and new attributes *must* be of the same type.

  Example:
  ```
  Transpose_Attributes {
  0x001006e = 0x001884
  0x00010024 = 0x777533
  }
  ```

- Set_Attributes

  List of attributes into which to force data. Verify that these attributes are valid, and that the data is valid for the type of attribute.

  Example:
  ```
  Set_Attributes {
  0x001006e = Router 42
  0x00010024 = public
  }
  ```

- Transfer_Port_Attributes

  List of attributes to save from ports of old models into ports of new models.

  Example:
  ```
  Transfer_Port_Attributes {
  0x00011564
  }
  ```

- Right_Relationships

  A list of relations whose associations with old models on the right side are ignored or included. (See **Relation_Section_Ignores** switch in this section for details.)

  Example:
  ```
  Right_Relationships {
  0x10004
  0x230000
  }
  ```

- Left_Relationships

  A list of relations whose associations with old models on the left side are ignored or included. (See **Relation_Section_Ignores** switch in this section for details.)

  Example:
  ```
  Left_Relationships {
  0x10004
  0x230000
  }
  ```

- Object_ID_Exists

  A list of MIB objects that must exist on each target device. If all of the MIB objects exist and are readable, the target device can be converted.

  Example:
  ```
  Object_ID_Exists {
  1.3.6.1.4.1.52.2501.1.270.4.1.1.5
  1.3.6.1.4.1.52.2501.1.1.5
  }
  ```

- Filter_Attributes

  A list of attributes to use to select models. When the attribute_value of the attribute_id in any old models found is equal to the value given in the Filter_Attributes line, the model is selected. For text_string attributes, the attribute_value need only be a sub-string of the string that is returned from the old model.

  Example:
  ```
  Filter_Attributes {
  0x10053 = 1.3.6.1.4.1.49.2.3.5
  0x10b5a = Network Administrator
  0x1154f = false
  }
  ```

- Advanced_Transfer_Attributes

  Enables the transfer of attributes from any number of models that are related to the device model by a known relation path. Any number of these sections can be included within the input file.

Syntax:

```
Advanced_Transfer_Attributes {
Path = <Relation Path>
Identifier = <Identifier Attribute>
If <Attribute ID> = <Value>
If ModelType = <Model Type ID>
Transfer = <Attribute to transfer>
Transpose <Dest Attribute ID> = <Src Attribute ID>
}
```

Explanation:

Path = *<Relation Path> [Mandatory]*

Specifies how to find the models to be transferred. The relation path specifies a '.' delimited set of relation names. There is no limit to the number of elements in a relation path.

Examples:

Path = HASPART

Finds all models on the right of a HASPART relation to the device model, that is, all of the interfaces.

Path = Contains.HASPART

Scans all models on the right of a Contains relation. For each of these models, scans the models on the right of a HASPART relation, and these found models are those that the transfer acts upon.

- Identifier = <Identifier Attribute> [Mandatory]

  When the set of models specified by the relation path has been found, a method is required for matching these models to the models in the converted database, as the model handles will have changed. DBconv reads the values of the attributes specified by Identifier. It then tries to match this value with the converted models. If a match is found, DBconv considers this model the equivalent of the preconverted model. A common attribute for this entry would be Component_OID.

Example:

```
Identifier = 0x1006a
```

If <Attribute ID> = <Value>

This is a filter entry. It only transfers or transposes the specified attributes if the value of <Attribute ID> equals <Value>.

Example:

The following three lines would match all interfaces, identified by Component_OID, only when ifType equals 6:

Path = HASPART

Identifier = 0x1006a

If 0x1134c = 6

If ModelType = <Model Type ID>

A filter entry that only transfers or transposes the specified attributes if the model type matches <Model Type ID>.

Example:

The following matches all SSR_PortIf interfaces on an RS-8000.

Path = HASPART

Identifier = 0x1006a

If ModelType = 0x2c60006

- Transfer = <Attribute to transfer>

  Specifies the attributes to be transferred. There is no limit to the number of entries. Either one Transfer or one Transpose entry is required.

  Example:

  Transfer = 0x11564

- Transpose <Dest Attribute ID> = <Src Attribute ID>

  Specifies the attributes to be transposed. It takes the value specified in <Src Attribute ID> and writes it to <Dest Attribute ID>. The types of both of these attributes should be the same. Either one Transfer or one Transpose entry is required.

  Examples:

  The following copies in the VLAN name on an RS-8000 and writes it into the Notes field.

  Transpose 0x11564 = 0x2c604dd

  The following entry copies Notes and PollingStatus from all SS-PortIf models on an RS-8000:

```
Advanced_Transfer_Attributes {
Path = HASPART
Identifier = 0x1006a
If ModelType = 0x2c60006
Transfer = 0x11564
Transfer = 0x1154f
}
```

- Transfer_Notes

  Enables the copying of the notes fields from the specified areas. Each of the lines is translated as an **Advanced_Transfer_Attributes** (described earlier in this section).

  Example:

  ```
  Transfer_Notes {
  Device
  Interfaces
  Applications
  Modules
  Ports
  }
  ```

  One or more fields can be specified to copy the notes field for all models in the specified class (Device, Interfaces, and so on.).

Chapter 3: DDM Database Maintenance

This section contains the following topics:

Database Security

To enhance the security of the Distributed Data Manager database, perform one of the following platform-specific procedures after CA Spectrum installation.

On Windows:

1. Log in as Administrator.
2. Use Windows Explorer to navigate to the *$SPECROOT*/SS/DDM directory.
3. Right-click the .configrc file and select Properties.
4. Click the Security tab, examine the Permissions panel, and click the Advanced button.
5. Make sure that only "Administrator" or CA Spectrum Users" groups are listed.

   If you see entries for "Everyone" or any other person or group, delete them. It might be necessary to clear the setting that allows inheritable permissions from the parent to propagate to child objects. This setting is under the Advanced options for permissions.

On Solaris:

1. Log in as root.
2. Navigate to the *$SPECROOT*/SS/DDM directory.
3. Enter the following command:

   ```
   chmod 600 .configrc
   ```

Database Size Management

Managing the size of the DDM database can help it run more efficiently. To control the size of the DDM database, you can limit event and statistics logging to critical components of your network. To manage database size, you can take the following steps:

- Limit logged data by disabling the logging of statistics. Set the stats_logging_enabled parameter in the .vnmrc file to FALSE.

  Notes:

  - The default value for stats_logging_enabled is FALSE. However, if you have upgraded from a previous release, verify this setting.
  - For more information on the .vnmrc file, see the *Distributed SpectroSERVER Administrator Guide*.

- Limit the amount of event data that is stored. Disable the 'E' logging flag in EventDisp files for any events whose history is not significant.

  Note: For more information, see the *Event Configuration User Guide*.

- Disable or reduce logging for models that you are not actively monitoring. You can disable logging on an individual model by setting the Log Ratio (the number of polls per log) to zero in the Model Information view. Or use the Attribute Editor to disable logging for all models of a certain type. For example, if you are not interested in reporting traffic statistics for user ports, set the Polls to Log Ratio to zero on all user ports.

  To reduce logging frequency, increase the log ratio using either of these methods.

  Note: For more information, see the *Modeling and Managing Your IT Infrastructure Administrator Guide*.

Database Backup

Like the SpectroSERVER database, the DDM database requires regular backups to enable full recovery in the event of a hardware or operating system failure. Ideally, both of these databases are backed up daily to ensure the availability of current data and to keep the databases synchronized.

However, the DDM database is typically much larger than the SpectroSERVER database and thus requires more time for backups. Also, the Archive Manager must be shut down during the backup. As a result, a considerable backlog of historical data must often be processed when it is restarted. Backlog processing can adversely affect SpectroSERVER performance. For these reasons, we typically recommend backing up the DDM database weekly and backing up the SpectroSERVER database more frequently, preferably daily.

Note: The DDM and SpectroSERVER databases can fall out of synchronization when you restore one without restoring the other. For example, you back up both databases on Sunday. You create a model on Monday, then experience a system crash and restore your SS database with the Sunday save file. The restored SS database no longer includes the model that you created, but historical records for that model are in the DDM database. Moreover, if you recreate the model, you cannot then reconcile the DDM records from the original model with the new records that were generated for the recreated model. Avoid this kind of conflict by always restoring the DDM database whenever you restore the SpectroSERVER database.

You can perform a DDM database backup at any time using the ddm_save (see page 64) or ddm_backup (see page 64) tools. However, you can also have the DDM database backed up automatically using the Online Backup feature. Online Backup uses the post_olb_script. Because it is critical to keep the SpectroSERVER and DDM databases synchronized, this script is the preferred method for DDM backups.

Important! The tools that are described in this section (post_olb_script, ddm_backup, and ddm_save) are the only supported methods of backing up your DDM database. Use of third-party backup software can result in database corruption.

post_olb_script

This script file lets you coordinate the execution of custom scripts with SpectroSERVER database backups that are performed using Online Backup. The script is located in the *$SPECROOT*/SG-Support/CsScript directory. When you run it with the default settings, post_olb_script backs up the DDM database immediately following any Online Backup that completed on the specified day (the default is Sunday).

To run the post_olb_script file with its default settings, copy the file from *$SPECROOT*/SG-Support/CsScript to *$SPECROOT*/custom/CsScript. Use a text editor to activate the final 11 lines of the file by removing the pound sign (#) from the start of each line. The DDM database backup is scheduled after any Online Backup executed on a Sunday (target day=0). Thus, if you schedule Online Backup to run daily, your DDM database is automatically backed up weekly.

Note: During a CA Spectrum upgrade installation, the copy operation is performed automatically. Before copying, verify that the file does not already exist to avoid reverting to default settings.

If you require more frequent DDM database backups, uncomment only the last five lines of the post_olb_script file. No “target day” is then specified. Therefore, your DDM database is backed up after *every* backup of your SpectroSERVER database using Online Backup.

More information:

Online Backup (see page 12)

ddm_save

The ddm_save utility, located in the *$SPECROOT*/SS/DDM directory, performs a full save of the historical database to the file you specify. Use the Control Panel to shut down the Archive Manager before executing the command, and then restart it when the save is completed.

Note: The ddm_save utility does not remove any data from the DDM database; it copies it to the designated file.

The utility has the following format:

```
ddm_save [-extension] [-quiet] <SAVE_FILE>
```

Where the first letter of an argument name appears in bold type, you can use the letter only, rather than entering the whole name.

-extension

By default, ddm_save assigns a .tgz file extension to the file. When this argument is specified, the file is saved without a file extension.

-quiet

Disables interactive/verbose mode.

<SAVE_FILE>

Specifies the name of the destination file for the saved database.

Note: The ddm_save command saves the necessary database files to a gzipped tar file format with a default file extension of .tgz.

ddm_backup

The ddm_backup utility, located in the *$SPECROOT*/SS/DDM directory, shuts down the Archive Manager, executes the ddm_save command, and restarts the Archive Manager when the save has completed.

This utility has the following format:

```
ddm_backup <SAVE_FILE>
```

<SAVE_FILE>

Specifies the name of the destination file for the saved database. Supply a fully qualified file path with the destination file name to save the file in that location. If a fully qualified file path is not specified, the file is saved in the DDM directory.

More information:

ddm_save (see page 64)

Points To Remember About Backups

- The more frequently you perform backups, the less data you risk losing.
- Backing up historical data does *not* remove that data from the DDM database.
- Historical records are purged (permanently removed) from the DDM database according to the MAX_STAT_DAYS and MAX_EVENT_DAYS settings in the .configrc file. For more information, see Archive Manager Resources (see page 70).

  The default setting is 45 days. Once any records attain the specified age, the purge occurs daily thereafter. As a result, the oldest day's data is purged daily.
- If you use the ddm_save method, you must restart Archive Manager using the Control Panel once the backup is completed.

Database Restoration

In the event of a hardware or operating-system failure, restore the DDM database by loading a save file that was created using any of the supported backup methods:post_olb_script, ddm_backup, or ddm_ save.

Save files are loaded using the ddm_load tool, which is described in the following section.

ddm_load

This utility, which is located in the *$SPECROOT*/SS/DDM directory, lets you restore the DDM database. First, it initializes (removes all data from) the DDM directory, and then it loads the save file that you specify. Initialization occurs automatically if you are loading a save file or directory. The -initialize option is only required if you want to clean out the DDM directory and start from nothing. With that option, the landscape handle for the associated SpectroSERVER is also required.

Before you execute the command, shut down the Archive Manager from the Control Panel. Then restart it when the restoration has completed.

This command has the following format:

```
ddm_load [-quiet] [-initialize <LANDSCAPE_HANDLE>] [-events_init]
[-stats_init][<SAVE_FILE>]
```

Where the first letter of an argument name appears in bold type, you can use the letter only, rather than entering the full string.

-quiet

Disables prompts (interactive mode). Useful for running load commands from within a script.

-initialize

Removes all data from the DDM database. Used only with the LANDSCAPE_HANDLE argument.

<LANDSCAPE_HANDLE>

Specifies the landscape handle of the SpectroSERVER whose database is being initialized.

-events_init

Initializes (removes) *only* the event records in the DDM database.

-stats_init

Initializes (removes) *only* the statistical records in the DDM database.

<SAVE_FILE>

Specifies the name of the backup file to load. The ddm_load utility supports both the new gzipped tar file format and the legacy, proprietary file format.

Points To Remember About Restorations

- To keep the databases synchronized, the DDM database must be restored each time that the SpectroSERVER database is restored.
- Some of the records in a given save file are likely to have reached their purge date by the time you reload them. As a result, they are purged as soon as the Archive Manager is restarted.
- Restart the Archive Manager when the restoration has completed.

Database Maintenance and Optimization

The following sections discuss scripts that you can run to keep the DDM database running efficiently.

db_maintenance.pl

Located in the *$SPECROOT*/SS/DDM/scripts directory, this script removes all unreferenced statistical records from the DDM database. Unreferenced records are typically created when models are deleted. Over time, their associated records are purged, leaving some unnecessary remnants in the database.

Important! The db_maintenance.pl script can only be run with the Archive Manager down. Carefully weigh the potential benefit of running this script against the drawbacks of taking the Archive Manager down.

Do not schedule this script to run regularly. Run it occasionally, at times when an extremely large number of models have been deleted from a landscape. The performance gains that are realized by running this script are not noticeable. Run this script only for general housekeeping if a great many models have been destroyed (such as an entire landscape).

We recommend running the db_optimize.pl script before running this script. In addition, we recommend running this script at a time that minimizes its impact on production environments. The script can run very slowly on a large, unoptimized database or on a database with many event records.

The script has the following format:

```
./db_maintenance.pl[-q]
```

-q

(Quiet) Disables prompting (interactive mode).

More information:

db_optimize.pl (see page 67)

db_optimize.pl

This script is located in the *$SPECROOT*/SS/DDM/scripts directory. It provides an easy way to optimize all of the tables in the DDM database.

Optimizing tables has two major benefits:

- The speed of queries that are sent to the DDM database is increased.
- Disk space that is available from purged records is recovered.

You can schedule this script to run weekly or monthly, depending on your performance requirements. Keep in mind that this script can take a considerable amount of time to run, depending on the size and the level of fragmentation of the DDM database.

Note: The db_optimize.pl script requires free disk space that exceeds the size of the largest file in the *$SPECROOT*/mysql/data/ddmdb directory. We recommend backing up your database file before running this script.

The script may be run with the Archive Manager up. However, it creates a 10-minute delay between table optimizations to allow the Archive Manager to recover from the database being locked.

This script has the following format:

```
./db_optimize.pl[-q]
```

-q

(Quiet) Disables prompting (interactive mode).

DDM Database Queries

The queries provided in this section can help you determine what events are being generated and what models are generating these events. We have provided queries that can help you determine whether a specific device is generating excessive events or whether a change is required to the number of days that statistical data is stored. These queries can take a while to run, depending on the size of the DDM database event table.

Note: For information on setting the MAX_STAT_DAYS parameter value, which controls the number of days that data is stored, see Archive Manager Resources (see page 70).

To log in to MySQL:

At a command prompt, enter the following command in *$SPECROOT*/mysql/bin:

```
mysql -uroot -proot ddmdb
```

On UNIX, also pass in the following argument:

```
-S $SPECROOT/mysql/tmp/mysql.sock
```

The mysql> prompt appears.

To query the DDM (ddmdb) database:

At the mysql> prompt, enter any of the following queries:

- To display table statistics:

  ```
  SHOW TABLE STATUS LIKE "event";
  ```

- To get a count of the total number of events:

  ```
  SELECT COUNT(*) FROM event;
  ```

- To get a count of the number of events that occurred after a particular date:

  ```
  SELECT COUNT(*)
    FROM event
    WHERE utime >= UNIX_TIMESTAMP("yyyy-mm-dd");
  ```

- To get the top ten events most commonly generated:

  ```
  SELECT HEX(type), COUNT(*) AS cnt
    FROM event
    GROUP BY type
    ORDER BY cnt DESC
    LIMIT 10;
  ```

- To get the top ten models with the most events:

  ```
  SELECT HEX(e.model_h), m.model_name, COUNT(*) AS cnt
    FROM event e, model m
    WHERE e.model_h=m.model_h
    GROUP BY e.model_h
    ORDER BY cnt DESC
    LIMIT 10;
  ```

- To get the top ten high-volume days for events:

  ```
  SELECT DATE(FROM_UNIXTIME(UTIME)) AS x, COUNT(*) AS cnt
    FROM event
    GROUP BY x
    ORDER BY cnt DESC
    LIMIT 10;
  ```

- To get the last ten days of volume of events:

  ```
  SELECT date(from_unixtime(utime)) AS x, COUNT(*) AS cnt
    FROM event
    GROUP BY x
    ORDER BY x DESC
    LIMIT 10;
  ```

Other Database Utilities

CA Spectrum provides three MySQL database utilities, all of which are installed in the *$SPECROOT*/mysql/bin directory.

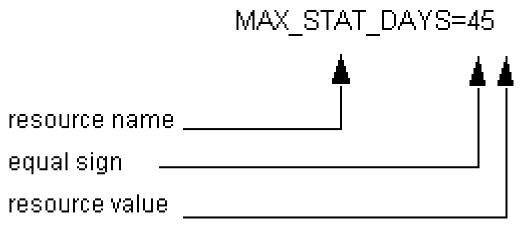
- myisamchk lets you check and repair MyISAM tables while the MySQL server is stopped.
- mysqlcheck is similar to myisamchk. However, it does not require you to stop the server to check or repair tables. This utility also optimizes and analyzes tables.
- mysqladmin enables you to perform administrative operations, such as checking the server configuration or dropping a database.

Note: For more information on using these MySQL utilities, see the documentation provided at http://www.mysql.com.

Archive Manager Resources

Archive Manager resources are settings that control how historical records are processed. These resources also enable the Archive Manager to communicate with the SpectroSERVER and DDM databases.

Archive Manager resources are defined in the .configrc file, which is located in the *$SPECROOT*/SS/DDM directory. The resources are listed in the form "resource = resource_value", as shown here.



The .configrc file defines the following Archive Manager resources:

ARCH_MGR_SOCKET_NUMBER

Identifies the port where the Archive Manager listens for requests from the VNM and SSAPI clients.

Default: 0xbafe

AUTO_REPAIR_DB

Controls whether the Archive Manager automatically attempts to repair a corrupt DDM database. By default, this parameter is not listed in the .configrc file and, therefore, it is not enabled. To enable the auto-repair functionality, add AUTO_REPAIR_DB=TRUE to the .configrc file. To disable the auto-repair functionality, set the value to FALSE or remove the entry from the .configrc file.

Default: Disabled

DDM_DATABASE_NAME

Required for communications between the Archive Manager and the DDM database. The default value is read in automatically when CA Spectrum is installed. Changing this value is not recommended.

Default: ddmdb

DDM_DATABASE_PORT

Specifies the port used to connect to the MySQL database server.

DDM_DATABASE_HOSTNAME

Required for communications between the Archive Manager and the DDM database. The default value is set automatically when CA Spectrum is installed. Changing this value is not recommended.

Default: local host

DDM_DATABASE_PASSWORD

Required for communications between the Archive Manager and the DDM database. The default value is read in automatically during installation. Changing this value is not recommended.

Default: CA Spectrum Installation Owner user name password

DDM_DATABASE_USERNAME

Required for communications between the Archive Manager and the DDM database. The default value is read in automatically during installation. Changing this value is not recommended.

Default: CA Spectrum Installation Owner user name

DDM_SOCKET_NUMBER

Deprecated. This resource will not appear in future releases.

LANDSCAPE_PRECEDENCE

Currently unsupported resource.

MAX_DB_CONNECTIONS

Specifies the maximum number of simultaneous MySQL connections that the Archive Manager can use to service requests.

Default: 25

MAX_EVENT_DAYS

Specifies the maximum number of days that event data is stored. When MAX_EVENT_DAYS is exceeded, the older data is purged.

Default: 45

MAX_STAT_DAYS

Specifies the maximum number of days that statistical data is stored. When MAX_STAT_DAYS is exceeded, the older data is purged.

Default: 45

TIME_TOLERANCE_IN_SECONDS

Specifies the maximum allowable variance in seconds between the timestamp that is reported for logged data and the actual logging time. The higher the time tolerance value, the more effectively data can be compressed, conserving CPU and disk resources. However, accuracy is sacrificed as this value increases.

Default: 300

DDM and Archive Manager Troubleshooting

This section identifies probable causes and solutions for problems that are associated with the Archive Manager and the DDM database.

Error Messages in ARCHMGR.OUT

Symptom

I saw the following message:

```
<date/time> ERROR at ArchMgr.cc(591): Failed to open connection to SpectroSERVER at
<hostname>, 0xbeef. The error indicates that the SpectroSERVER is down. Will try again
in 60 seconds.
```

Solution

The SpectroSERVER is not ready. The Archive Manager automatically starts when the SpectroSERVER is started but cannot connect until the SpectroSERVER is ready. This message always appears on initial startup and anytime that the SpectroSERVER goes down.

Symptom

The following message appears once the Archive Manager has connected or reconnected:

***<date/time>* : ArchMgr has successfully registered with DDM name service - *<ip_address>* (<machine_name>).**

Symptom

When I attempted to start the Archive Manager, I saw the following message:

***<date/time>* : ArchMgr is shutting down...**

Solution

The Archive Manager has been shut down because the SpectroSERVER does not contain a CA Spectrum User model for the user who attempted to start it.

Symptom

I saw the following message:

***<date/time>* : ArchMgr has successfully shut down.**

If this message persists, check the Users tab in the OneClick Console to verify that a user exists for the owner of the ArchMgr process.

Symptom

I saw the following message:

```
<date/time> : ArchMgr started as user '<user_name>'
<date/time> : ArchMgr validating database
Database corruption detected:
    ddmdb.statistic_ul64 - record delete-link-chain corrupted
    ddmdb.statistic_ul64 - Corrupt
Error opening the DDM database. One or more tables are corrupt.
Recovery options:
1. Run `ArchMgr -repair` to attempt recovery
2. Load a valid DDM savefile using `ddm_load <savefile>`
3. Initialize the DDM database using `ddm_load -i <LANDSCAPE_HANDLE>`
<date/time> : ArchMgr invalid database error.
```

Solution

The DDM database has been corrupted and cannot be loaded. You can use the ddm_load command to load a saved DDM database or initialize the DDM database. You can also attempt to repair the existing database file using the ArchMgr -repair command.

To run the ArchMgr -repair command, navigate to the *$SPECROOT*/SS/DDM directory and enter:

./ArchMgr -repair

Once the repair has successfully completed, the following message appears:

“ArchMgr successfully repaired database.”

To enable the Archive Manager to automatically attempt to repair a corrupted DDM database, use the auto_repair_db option in the .configrc file. For more information,see Archive Manager Resources (see page 70) .

More information:

ddm_load (see page 65)
Archive Manager Resources (see page 70)

No Events Are Listed

Symptom

No events are listed in on the Events tab and the Servers (Connection Status) button is flashing yellow. When you double-click the button, the Connection Status dialog shows a status of “Down” for the Events and View services, but “Up” for the Landscape service.

Solution

The Archive Manager is not running or is not connected to the SpectroSERVER. Check the Control menu in the Control Panel to verify that the Archive Manager service has been started. If it has been started, check the *$SPECROOT*/SS/DDM/ARCHMGR.OUT file for Archive Manager errors.

Index