Peregrine

ServiceCenter

Database Management and Administration, Volume 2

Release 5.1

PART NO: SCT-5.1-ENG-01041-00233

Peregrine SYSTEMS

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 5.1 of the licensed program.

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Contents

Introducing Database Management and Administration, Volume 2

Database Management and Administration, Volume 2 is a set of two volumes covering various aspects of using ServiceCenter with databases. It was designed to aid experienced ServiceCenter system and database administrators who are responsible for installing and implementing the ServiceCenter databases or who will be hosting ServiceCenter data and assisting in database conversion.

Volume one, *RDBMS Support*, was designed to aid ServiceCenter system and database administrators in converting ServiceCenter data from its internal format to a storage location on a commercial Relational Database Management System (RDBMS). It provides technical details on the conversion process and optimization tips.

Volume two is divided into two main sections:

- *PeregrineFour File System Administration* — was designed to provide ServiceCenter system and database administrators with the data necessary for maintaining the ServiceCenter P4 file system.
- *Data Retrieval* — was designed to provide ServiceCenter system and database administrators with information on how to retrieve, edit, and maintain database records.

Sample Screens and Examples

The sample screens and examples included in this guide are for illustration only, and may differ from those at your site.

Knowledge Requirements

While this guide explains various aspects of ServiceCenter system administration, a certain level of knowledge of ServiceCenter is presumed. This manual is designed for a System Administrator or Database Administrator.

For more information on ServiceCenter applications and system administration, please refer to:

- *User's Guide*
- *System Administrators' Guide*
- *Application Administration Guide*

Documentation Web Site

For a complete listing of the current ServiceCenter documentation, see the Documentation pages on the Peregrine CenterPoint Web site

http://support.peregrine.com/

You will need your current login and password to access this Web page.

For copies of the manuals, you can download .PDF files of the documentation using the Adobe Acrobat Reader (also available on the Web site). Additionally, you can order printed copies of the documentation through your Peregrine Systems sales representative.

Contacting Peregrine Systems

For further information and assistance with ServiceCenter in general, contact Peregrine's Customer Support.

Peregrine's CenterPoint Web Site

Current details of local support offices are available through Peregrine's CenterPoint Web site at

http://support.peregrine.com/

To find Peregrine Worldwide Contact Information:

1. Log on with your login User Name and Password.
2. Click Go for CenterPoint.
3. Select Whom Do I Call? in the navigation bar on the left side of the page.

Peregrine worldwide information is displayed for all products.

Contacting Education Services

Training services are available for the full spectrum of Peregrine Products including ServiceCenter.

Current details of our training services are available through the following main contacts or at:

http://www.peregrine.com/education

Address:	Peregrine Systems, Inc. Attn: Education Services 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+1 (858) 794-5009
Fax:	+1 (858) 480-3928

SECTION I

PeregrineFour File System Administration

This section was designed to provide ServiceCenter system and database administrators with the data necessary for maintaining the ServiceCenter P4 file system.

Chapters in this section include:

- *Introduction to the P4 File System* on page 17 — this chapter provides an overview of the ServiceCenter P4 file system.
- *Setting Case Mode for Searching the P4 File System* on page 23 — this chapter explains how to change the case mode used when searching the P4 file system.
- *P4 Backup* on page 31 — this chapter explains how to perform a Cold Backup of the P4 file system (shutting down ServiceCenter and backing-up the database), and how to perform a Hot Backup of the P4 file system using the 24x7 Backup Utility.
- *P4 File System Utility* on page 49 — this chapter provides an introduction to SCDBUTIL, including a description of each Database utility and its usage.
- *Performance and Tuning Tips* on page 91 — this chapter provides tips for tuning the P4 file system.
- *P4 Troubleshooting* on page 99 — this chapter provides known exposures that can corrupt the P4 file system.
- *The ServiceCenter ODBC Driver* on page 113 — this chapter explains how to install and upgrade a current ODBC installation, as well as procedures for creating a data source and for changing data source locations.

- *P4 Glossary* on page 345 — this glossary contains definitions of terms used in the *PeregrineFour (P4) File System Administration* section of this guide.

CHAPTER 1

Introduction to the P4 File System

This chapter was designed to provide ServiceCenter database administrators with an overview of the ServiceCenter P4 file system. The instructions assume a working knowledge of Peregrine Systems ServiceCenter.

Topics in this chapter include:

For definitions of terms used in this chapter, see the *P4 Glossary* on page 345.

Structure of the File System

The ServiceCenter P4 file system is designed specifically for ServiceCenter. It organizes records and reuses all free space. The file system consists of several files (called physical files). On a higher level, physical files can be grouped together into *pools*. Although physical files have a limitation of 2 GB, by grouping several physical files together into pools, this limitation can be avoided.

There are two pool types: *system data pools* and *user data pools*.

- System data pools hold the basic information needed to organize the data stored in the P4 file system.
- User data pools hold all data that can be accessed by the ServiceCenter Database Manager. Although this type of pool is called a *user data* pool, it also stores very important information that ServiceCenter needs, for example, the dbdict file, the format file, the code file, and so on.

System Data Pools

The system data pools each consist of at least one physical file:

Filename	Contents
scdb.fre	(File 0) The free list file contains start pointers to free space within each pool. It also holds the information that tells which physical file belongs to which pool. This file may also be referred to as file 0.
scdb.asc	(File 1) The associator file contains short records called *associators* which describe each data record stored in the P4 file system.
scdb.lfd	(File 2) The descriptor file contains the logical file descriptor records, two for each dbdict record - one for the data logical file, one for the index logical file. These records hold the information in which pool the logical file is stored, usually this is only one pool (for example pool 5 for the probsummary file, or pool 9 for the spool file) but it could be up to 10 different pools.
scdb.dbx	(Files 3-9) These are the User Data Pools. The files contain the actual data records and the index records (keys) for all files stored on P4.

User Data Pools

The following table describes which tables are stored in which pool in the system as shipped, and therefore represents a recommendation by Peregrine Systems. Depending on your needs, you may find a better way of distributing the table over the pools:

Pool	Contents
Pool 3	This pool contains all the data and index records that are not contained in the remaining user data pools (i.e., pool 4 - 9). The data records in this pool are made up of formats, Format Control records, applications, and so forth. This pool consists of the physical file **scdb.db1** in the default ServiceCenter system. New tables or logical files will be put into this physical file by default.
Pool 4	This pool contains the **schedule** table. This pool consists of the physical file **scdb.db2** in the default ServiceCenter system.
Pool 5	This pool contains the **probsummary** table. The latest updates to Incident tickets are stored in this table. This pool consists of the physical file **scdb.db3** in the default ServiceCenter system.
Pool 6	This pool contains the **problem** table. This table contains all the Incident tickets in the system, including all updates. This pool consists of the physical file **scdb.db4** in the default ServiceCenter system.
Pool 7	This pool contains records related to Change Management changes and tasks, i.e., **cm3r**, **cm3t**, **cm3rpage**, **cm3tpage**. This pool consists of the physical file **scdb.db5** in the default ServiceCenter system.
Pool 8	This pool contains the ServiceCenter **device** records created and maintained through Inventory Management in the **device** table. This pool consists of the physical file **scdb.db6** in the default ServiceCenter system.
Pool 9	This pool contains the spool table which ServiceCenter uses temporarily to store reports that are waiting to be printed. This pool consists of the physical file **scdb.db7** in the default ServiceCenter system.

Database Files and IR Expert Files

In addition to the above physical files, there exists a set of related files that are used to maintain indexes used by IR Expert. All IR Expert files have a prefix of ir (for example, ir.probsummary) and are located in the path specified by the ir_prefix parameter of the sc.ini file.

Relationships exist between the P4 file system files and the IR Expert files. All of these files must be treated as a single unit for backup and recovery purposes. For example, it is *not* possible to back up scdb.db1 independently of the other files.

Important: The only reliable ways to back up these files are (1) to copy the files while ServiceCenter is stopped or (2) to use the ServiceCenter 24x7 backup utility (see *P4 Backup* on page 31). No other methods will work because no other method understands the relationships that exist between the files. Other methods may cause corrupted backups.

dbdict

A special file called dbdict holds information describing all files in the database, including:

- file name
- field names and data types
- index key definitions
- index logical file number
- data logical file number

All files in the database consist of fields. Fields can have the following data types:

Field	Purpose
number	used for all numeric type data.
character	used for character strings.
date/time	used for absolute dates and times as well as relative time intervals.

Field	Purpose
logical	used to represent true or false.
array	used to represent multiple fields of the same data type.
structure	used to represent a collection of fields of the same or different data types.
expression	used to represent parsed Rapid Application Development (RAD) language expressions.

Key Definitions

Key definitions allow you to specify the type of key and the names of the fields within the key. There may be more than one field in one key. A key with multiple fields is called a **concatenated key**. The key types allowed are:

Key Type	Description
unique	at least one field in the key must not be null and the value of the complete key must be unique in the index.
no nulls	at least one field in the key must not be null.
no duplicates	the value of the complete key must be unique in the index or the values of all fields must be null.
nulls & duplicates	all fields can be null and the complete key value may be in the index more than once.
IR key	the fields in the key are indexed by IR Expert. Only one IR Key can be used per dbdict record, otherwise, IR searches on that file will not work. However, you can concatenate several fields in an IR key.

P4 Unique keys are no nulls and no duplicates, but in some RDBMSs, such as db2, unique constraints are no duplicates & no null, and in some, such as oracle, unique constraints are no duplicates & null. Since each RDBMS has its own rule on constraints, ServiceCenter only creates Unique constraint on the RDBMS that allows nulls. After conversion, change the unique constraints to primary keys on the RDBMS side (which allow no nulls and no duplicates), if the key will contain no duplicate and no null values.

The following table shows the attributes of the various key types. A unique key, e.g. does not allow null values or duplicate values.

Key Type	No Duplicates	Allows Duplicates
No Nulls	unique	no nulls
Allows Nulls	no duplicates	nulls & duplicates

Note: IR keys are indexed separately and are not part of this table.

CHAPTER 2 Setting Case Mode for Searching the P4 File System

This chapter was designed to help ServiceCenter Database Administrators change the case mode used when searching the P4 file system.

Topics in this chapter include:

Introduction

By default, the P4 file system is keyed for case sensitive searching. ServiceCenter allows database administrators to rekey the P4 file system to allow case insensitive searching, if desired.

Warning: If converting P4 to be case insensitive, and converting P4 to an RDBMS, make sure that the RDBMS also supports case insensitivity.

Case insensitive searching will allow your users to find records, even if they do not know the case in which the data was entered. This is a very useful feature; however, the change is time-consuming and may be problematic.

Note: The case-insensitivity applies only to the database layer. The user interface is still case-sensitive.

Warning: Implement case insensitive searching on a test system and make sure that everything is working before setting it up on a production system.

Preparing ServiceCenter for Case Mode Conversion

By default, ServiceCenter databases are sorted in ASCII or EBCDIC order for searching. Before conversion, the index structure of the contacts database might contain the following entries:

Brown
Hawthorn
falcon

When the indices are regenerated for Case Insensitive sorting, all keys are resorted and the sorting algorithm is changed. The upper and lower case letters are not separated for the case insensitive search mode. After conversion, the index structure of the contacts database might contain the same entries in this order:

Brown
falcon
Hawthorn

This change applies only to the indexes. The original records will not be changed. The databases can be regenerated to be case sensitive again, if necessary.

To prepare your system for the necessary case conversion, make sure all **unique keys** in all records will be unique, regardless of case.

If there are records that contain keys which violate the uniqueness of the supposedly **unique key**, only one record will be accessible. The original records will still be in the database intact and unchanged, but users will not be able to retrieve them.

OS/390 Preparation for 4.0 Applications Only

Before converting the case sensitivity to ServiceCenter running on an OS/390 platform, you add the **case.insensitive** field to the **dbdict** database dictionary and make it a logical field.

Note: This step is not necessary if you are using 5.0 or later applications.

To add the case.insensitive field:

1. Open the **dbdict** file in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the Administration Mode check box.
2. On the database file list select **dbdict**.
3. Use the scroll bar to move to the end of the Names list.
4. In the first available line, add the following:
   - Name: case.insensitive
   - Type: 4
   - Index: 13
   - Level: 1
5. Stop and Restart ServiceCenter
6. Perform the case conversion as documented in *Changing the Case Mode* on page 26.

Changing the Case Mode

The case mode for searching the P4 file system is set from the System Wide Company Definition form **Options** menu. Setting the case mode requires regeneration of all ServiceCenter databases. This can take a very long time. The length of time necessary for the regeneration depends on the number of keys and data records to be converted, as well as your system's CPU, hard drive, and memory.

Warning: Files cannot be accessed while they are being regenerated. Schedule downtime for the conversion, or do it when the shop is closed.

To set the case mode for searching the P4 file system:

1. Shut down all background processes before starting the conversion.
2. Open the system administrator's main menu.

   From the system administrator's main menu, select the **Utilities** tab.
3. Click **Administration**.
4. Click **System Wide Company Record** in the Information panel. The System Wide Company Definition form is displayed.
5. Select the **Misc.** tab.

   The case mode (Case Sensitive or Case Insensitive) will appear on this tab if the case mode has ever been changed. If the case mode is blank, that means that the case has never been set, and the search is currently Case Sensitive.

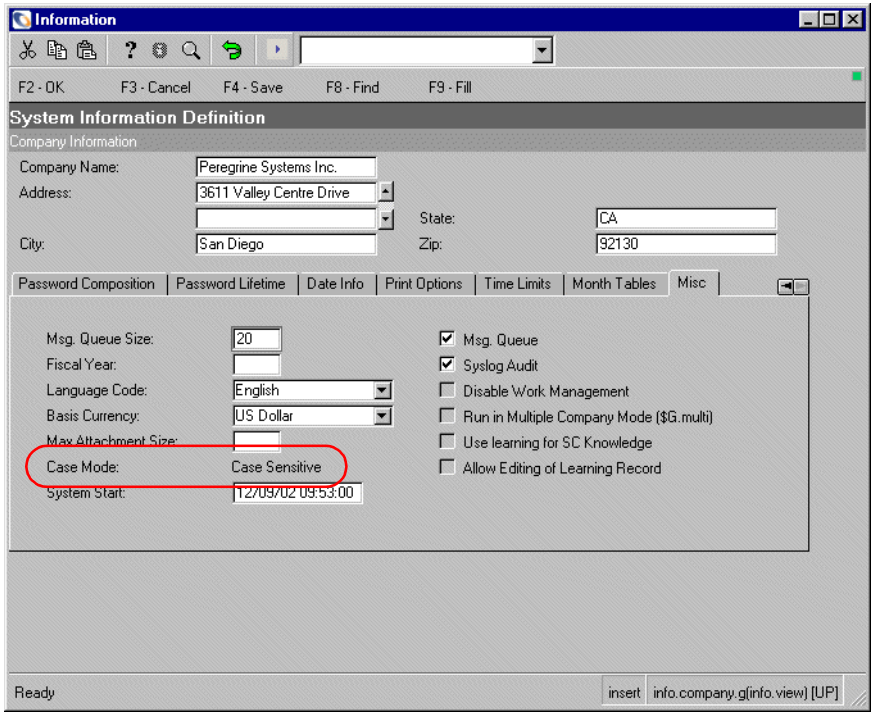


Figure 2-1: System Wide Company Definitions — Misc. tab

6 Open the ServiceCenter **Options** menu. The case mode you can currently switch to will be on this menu.

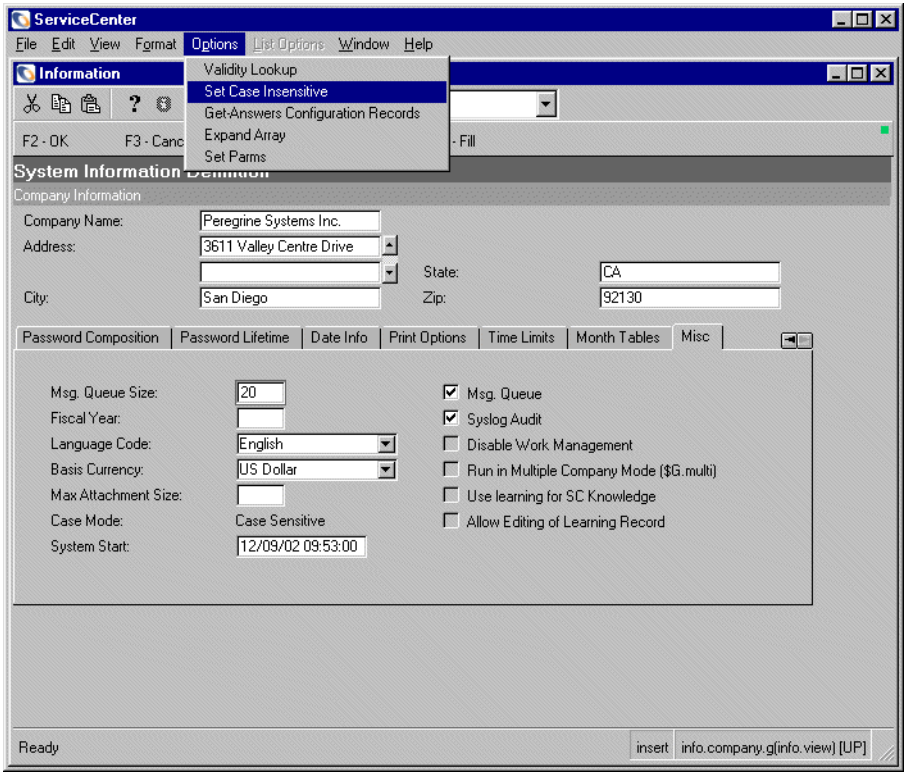


Figure 2-2: System Wide Company Definitions Options menu

7 Set the case mode.

- Select **Set Case Insensitive** to enable case insensitive searching.

  — Or —

- Select **Set Case Sensitive** to disable case insensitive searching.

A message is displayed warning you that this action may take a long time, and asking you to confirm the action.

8 To proceed with the regeneration, click **Yes**.

Warning: This action will start a regeneration of all ServiceCenter databases. This can take a very long time.

9 When the regeneration is finished, view file **sc.log**, located by default in the main ServiceCenter directory on the server. The ServiceCenter log will have any error messages, including duplicate key errors.

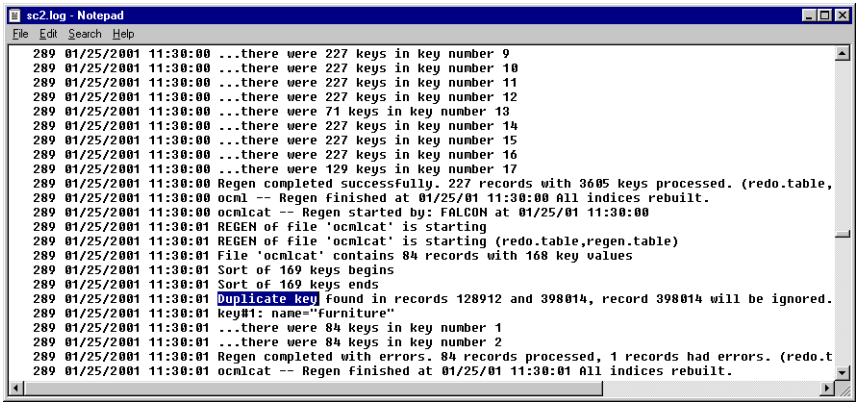


Figure 2-3: Duplicate key error in sc.log

10 There should be no errors. If there are duplicate key errors, correct them by giving each record the appropriate unique keys, or by using one of the other techniques described in *Error Recovery*, next section.

Error Recovery

If the ServiceCenter log indicates duplicate key errors, there are a few choices for recovery.

- Convert the entire set of databases back to the original Case Sensitive Mode. This will take as long as the initial conversion did. All of your records will be accessible again. You can then fix the records and convert the databases back into Case Insensitive mode. This will also take the same amount of time as it did originally.
- Change the key values for the record you can view, making it different than the ones you cannot view. You will have to repeat this for each duplicate but, depending on how many duplicate key errors there are, this can be faster than converting to case sensitive, fixing the records and converting to case insensitive again. When all duplicates have been fixed, regenerate the database. See *Regenerating Database Keys from Database Manager* on page 315 for instructions on how to regenerate the database.

- Change the key type for the record. Case Sensitivity is only a problem for unique and no duplicate keys. Change **unique** keys to **no nulls** and change **no duplicates** to **nulls & duplicates**. (For a list of key types and their definitions, see *Key Definitions* on page 21.) When you have changed all the key types, search for and fix the records in question. Then go back and change the keys back to their original values, if necessary.

3 P4 Backup

CHAPTER

This chapter was designed to aid ServiceCenter database administrators in backing up the P4 file system.

Topics in this chapter include:

Introduction

There are two ways to perform a backup of the P4 file system:

- Cold Backup is performed when the P4 file system is shut down (in other words, the ServiceCenter server is not running).
- Hot Backup is performed when ServiceCenter is running and end-users are potentially updating the P4 file system. This backup method is performed using the 24x7 backup utility. Hot backup allows you to make a copy of the database without having to bring down the system.

Files you may want to back up include: ir.*, scdb.*, sc.ini, sc.cfg, and any RDBMS data files.

All scdb.* files and all ir.* files are related and must be backed up and restored as a unit. Any file located in a directory referred to by either the path or the ir_prefix parameter must be treated as a single unit. Therefore, if your path and ir_prefix parameters do not point to the same directory, you will have to back up the IR directory in the same way and at the same time as you perform a backup of your P4 file system.

Cold Backup

This method is available to back up the database on all platforms. In the out-of-box Windows installations, the P4 data files (scdb.*) and the IR files (ir.*) are in the C:\Program Files\ServiceCenter\Data directory. In the out-of-box Unix installation, the P4 data files (scdb.*) and the IR files (ir.*) are in the /ServiceCenter/Data directory.

To perform a cold backup:

1. Shut down ServiceCenter.
2. Backup the files by copying them to an alternate location, or zipping them using the commands appropriate for your platform.

   For example:

   - Unix

     tar -cvf /device/ scdb.asc scdb.lfd, etc.

     — Or —

     cp scdb.* /alternate directory

- Windows

  Copy the files to a backup location, or zip them.

- OS/390

  The data must be restored with the same definition as before. Therefore, copy the both the data and the DCB to a backup location.

3 Start ServiceCenter when the backups are complete.

This method requires careful scheduling because ServiceCenter is down while the backups are running, which can be over an hour.

A less intrusive method is available for sites that have their disks mirrored.

To perform a cold backup on a site with mirrored disks:

1 Shut down ServiceCenter.

2 Split the disk mirror.

3 Restart ServiceCenter.

4 Run the backups against the unchanging mirror.

5 When the backups are complete, re-sync the disk mirror.

This method requires that ServiceCenter be down for only a few minutes while the mirror is being split.

Recovery

ServiceCenter does not have a journalizing capability. This means that if something happens to the ServiceCenter database, you need to either use the scenter -util command to fix the problems, as directed by customer support, or restore from a backup. In most cases, problems can be fixed without requiring a restore.

To find problems in the database, run a *Database Consistency Check* using the LFSCAN utility *Option 6) Logical file Scan Utility* on page 58. Once the problems are identified, call Peregrine Systems Customer Support for assistance with fixing the problems.

If the errors are too severe to fix using the scenter -util command, you may have to restore the database from a backup. However, you may still be able to unload records out of the broken database that were changed since the backup was run and reload them into the restored copy. Always keep a copy or backup of the broken database before restoring over it.

Hot Backup

The 24x7 Hot Backup Utility allows a ServiceCenter system administrator to make a consistent copy of all the data in the ServiceCenter P4 file system (scdb.fre, scdb.asc, scdb.lfd, and scdb.db*) without having to bring down the system.

Note: IR Expert files are not part of the ServiceCenter P4 file system. If you are using the IR capabilities, additional steps are necessary for backup. See *24x7 Backup and IR Expert Files* on page 39, for details.

The 24x7 Hot Backup process is transparent to the end user. Files can be added and updated during the backup process, without any interruption in service. This method is available to back up the database on all platforms.

General Information

Memory

The 24x7 Hot Backup Utility requires sixteen bytes of shared memory for each update logged during the backup process. These 16 bytes track what has been updated and where the actual update is on the log file. The code does not get 16 bytes at a time. For performance reasons the code gets 1024 blocks per request (in 16 byte blocks). When those 1024 blocks are used, the code gets another block of 1024.

ServiceCenter and RDBMS Backup

You cannot use the 24x7 Hot Backup Utility when operating in the standard RDBMS mode, since the database is external to ServiceCenter. In systems where records are written with the RDBMS interface, the backup function is shifted to the external database product and its utility. In addition, the need for backing up the remaining ServiceCenter files is reduced. The backup of the remaining ServiceCenter files should be scheduled to coincide with the backup of the external database files. While use of the 24x7 Backup Utility is available, it is probably not necessary.

How 24x7 Works

The 24x7 Hot Backup Utility establishes exclusive locks on the ServiceCenter database files at a point of consistency. Once the locks are created, ServiceCenter saves all transactions (new input and updates) in a virtual storage cache and writes them to a log file. When an application attempts to read the data back from the database, 24x7 returns the data from the log if the data is new, or it may read from the file as if the data had been written to the ServiceCenter database.

When the backup process is complete, the logging of update activity is turned off. 24x7 then writes all the data from the log file to the appropriate ServiceCenter database files. The system is now clear to continue the normal updating process (updates are written directly to the ServiceCenter database).

The basic steps for running the 24x7 Backup Utility:

1. *Set Up the Logging Process*, next section.
2. *Start the Logging Process* on page 36.
3. *Verify that Logging is Enabled* on page 36.
4. *Backup the Database* on page 37.
5. *Stop the Logging Process* on page 37.

Set Up the Logging Process

Unix and Windows

On Unix or Windows platforms, there is no special setup required in order to run 24x7 Hot Backup. However, you can change the log file used for storing updates during the backup process. The default log file is scdb.pflog, located in the ServiceCenter RUN directory. Use the filelogging parameter to override the default path and filename. The filelogging parameter must be placed in the SC server's sc.ini file.

Unix example:
filelogging:/var/tmp/scdb.pflog

Windows example:
filelogging:C:\Temp\scdb.pflog

OS/390

These four parameters should be set in the SC PARMS dataset:

Parameter	Definition
filelogging	Required. Fully qualified name with which the scdb.pflog file will be allocated.
pqtylogging	Optional. Primary space allocation for the scdb.pflog file (in cylinders, default is 50)
sqtylogging	Optional. Secondary space allocation for the scdb.pflog file (in cylinders, default is 50)
volserlogging	Optional. Volume (volser) on which space will be allocated for the scdb.pflog file

For a complete list of OS/390 parameters, see the *Technical Reference*.

Start the Logging Process

To start the logging process:

- For Unix or Windows:
  Enter scenter -startlogging in a the operating system's Command prompt.
- For OS/390 (MVS):
  Use /f <jobname>,scenter -startlogging.

A message is displayed, stating:

Startlogging successful, logging mode now enabled, backup can now be started.

Verify that Logging is Enabled

To verify that 24x7 logging is enabled:

- For Windows or Unix:

  Enter scenter -infologging in the operating system's Command prompt to view a status report.
- For OS/390:

  Use /f <jobname>,scenter -infologging. Output is written to the terminal.

The status report is displayed in the following form:

```
Status = On
Current log offset= 18189
Logfile name = scdb.pflog
Cache status= Valid
Shared memory allocated = 20492 bytes
Start timestamp = Wed Mar 5 04:39:22 1997
```

Fields on the Status report form:

Field	Definition
Status	Indicates whether or not 24x7 is enabled.
Current log_offset	Indicates how much of the existing log file has been overwritten by current 24x7 logging.
Logfile name	Identifies the log file to which data is being written during the backup process.
Cache status	Indicates that the cache is active and that 24x7 is enabled.
Shared memory allocated	Indicates the amount of shared memory allocated to the log file.
Start timestamp	Start time of the **startlogging** process.

Backup the Database

Once logging has been enabled, begin backing up your ServiceCenter database files using the appropriate utility for your operating system (*cp* or *tar* in Unix, *Copy* or *Zip* in Windows).

Note: The logging file scdb.pflog should be excluded from the backup, because backup programs often hold exclusive locks that prevent writing to a file during backup.

Stop the Logging Process

To stop the logging process:

- For Windows or Unix:

  Enter scenter -stoplogging in the operating system's Command prompt .

- For OS/390:

  Use /f <jobname>,scenter -stoplogging.

When the stoplogging command is issued, ServiceCenter begins processing the transactions written to the log file during the backup process. Once the contents of the log file have been written to the database, all processes can write directly to the database again.

The backup created is a copy of the database at time the scenter -startlogging command was issued. Updates stored in the log file while a backup is being made are not included in that backup.

Important: Verify that stoplogging was successful. If stoplogging was not successful, the logging file will keep growing, and the next successful stoplogging may take several hours. If that process is stopped before the transactions are finished, data loss will occur.

Recommended Scheduling

Before running any type of system backup, consider the amount of activity on your system.

Important: Although 24x7 is designed to protect an active system during backup, running the utility during periods of high input activity is not recommended.

Schedule your backup procedure during off hours when activity is at a minimum. This reduces the file space required for the log file and keeps any perceivable ServiceCenter performance degradation to a bare minimum.

Note: If you are using an RDBMS to store data, the backup of ServiceCenter data files should be scheduled to coincide with the backup of the external database files.

Log File Size and Maintenance

The size of the log file differs for each ServiceCenter system. Perform a test run of 24x7 (in other words, run 24x7, but do not perform the actual backup) to determine the amount of shared memory required. Once the log file is created, subsequent use of 24x7 overwrites the previous data. If you want to rename the log file, do so at this time.

You do not have to save the log file after completing a successful backup. In the absence of an existing log file, the 24x7 Backup Utility creates and writes to a new scdb.pflog file each time the startlogging command is issued.

24x7 Backup and IR Expert Files

The 24x7 backup utility saves only P4 transactions in an external log file. Any other changes to IR Expert or RDBMS are executed immediately. This means that if you are creating a backup of all P4 file system files and all IR Expert files using 24x7, the IR Expert files in your backup might be inconsistent with the P4 files in your backup.

There are two ways to manage updates to IR files:

- Synchronous IR updates

  You cannot create a valid backup of the IR files with 24x7 hot backup. To create a valid backup of the IR Expert files you must shut down ServiceCenter and create a cold backup. This is the default.

- Asynchronous IR updates

  You can use either cold or hot backup to create a valid backup of the IR Expert files .

  IR updates are written to a temporary file within the P4 file system, and periodically pushed out to the external IR index files by the IRQUEUE background processor. The IRQUEUE process is automatically stopped and restarted during startlogging and stoplogging. The scenter -startlogging command stops any I/O to both the P4 database and IR Expert files.

For information on how use IR Expert including how to select Asynchronous IR updates, see *IR Expert* on page 319.

Note: The IRQUEUE processor does not run during a hot backup. Changes to IR index data are not available to users until after the backup. IR searches will not reflect the newest IR index data until the 24x7 Hot Backup is complete and restarts the IRQUEUE process.

Interruptions

Run-time Errors

In the event of an error (such as insufficient shared memory) while 24x7 logging is enabled, the logging process is terminated. No data is lost, since the logging file is processed upon termination of 24x7 and the logged data is copied to the ServiceCenter database. However, the backup being made is *not* valid.

If this occurs, you have two alternatives:

- Perform another backup when there is less activity on the system.
- Increase the amount of shared memory. (Shared memory is established with the shared_memory parameter normally specified in the sc.ini file.)

Messages

The following **BCKW4000** series of messages indicate that some error has occurred and that the logging process is being terminated. They are always issued together.

```
BCKW4100: WARNING: The LOGGING file has run out of space or
BCKW4101: the ServiceCenter Shared Memory Region has run out of
room!
BCKW4102: The DATABASE SYNCHRONIZATION phase will now start.
BCKW4103: After the completion of this phase the LOGGING option
BCKW4104: will be turned off. After this, your current backups
BCKW4105: may not be in a known state.
BCKW4106: Once the LOGGING option is turned off, all future I/O
BCKW4107: requests will use the ServiceCenter DB files.
BCKW4108: Contact your System Administrator immediately about
this condition!!
BCKW4109: Starting Database synchronization ...
```

The system attempts to process all updates in the logging file. If the update fails, the system issues the following **BCKE8200** series of messages:

```
BCKE8200: Database synchronization (forced) failed!
BCKE8201: Call your DB Administrator immediately!!
```

If the update succeeds, the following **BCKW4300** series of messages is issued and logged:

```
BCKW4300: Database synchronization (forced) completed successfully!
BCKW4301: The LOGGING options has now been turned off!
BCKW4302: Please Call your DB Administrator immediately!!
```

If synchronization succeeds under these conditions, an error has occurred. In either case, the database administrator (DBA) needs to examine the log to determine the cause of the initial failure.

Outages

In the event of a power outage or hardware failure, any transactions written to the log file are compromised and non-retrievable. Database files, which are not being written to during 24x7 logging, are unaffected by the outage.

The backup being created will *not* be valid.

Information Messages

Information messages in 24x7 carry the prefix **BCKI**, identifying them by process and type. The first three letters, **BCK**, signify a *backup* process. The fourth letter, **I**, identifies the message as Informational.

Information messages track the normal progress of the logging process. Message blocks (for example, **BCKI0001**, **BCKI0002**, and **BCKI003**) follow each other in numerical order unless interrupted by a Warning or Error message.

When you see the message below, logging has been successfully started and you can start your backup process.

```
BCKI0001: (startlogging) Initializing pf_block and cache ...
BCKI0002: (startlogging) PF_BLOCK structure initialized ...
BCKI0003: (startlogging) logging mode now enabled, backup can
now be started
```

The following message indicates when each process opens the logging file.

```
BCKI0004: Opened LOGGING file (filename), file descriptor = (value)
```

BCKI0005 is issued each time more shared memory is needed to record a change in the database. It is immediately followed by **BCKI0006** to indicate the amount of shared memory being used.

```
BCKI0005: Allocated (count) bytes for a new cache BLOCK_HEADER
BCKI0006: Current Cache Allocation = (count) bytes
```

BCKI0007 is issued if a stoplogging request is entered when logging is not active.

```
BCKI0007: ServiceCenter LOGGING is not active.
```

If a stoplogging request is entered more than once, **BCKI0008** is issued for all but the first task.

```
BCK10008: Synchronization being handled by another process.
```

The following series of messages is issued to track the progress of a stoplogging request. They follow each other unless some error occurs.

Phase 1 synchronization occurs concurrently with normal user activity.

Phase 2 synchronization prevents concurrent activity while it writes out the final database changes.

```
BCKI0009: ServiceCenter DB synchronization started ...
BCKI0010: Cache is being processed. Please wait ...
BCKI0011: Phase 1 started ... )
BCKI0012: Writing 24X7 log file records to the ServiceCenter
Database
BCKI0001: Phase 2 started ... )
BCKI0002: ServiceCenter exclusive lock has been released ...
BCKI0003: ServiceCenter DB synchronization completed successfully
...
BCKI0004: Normal I/O operations will now resume ...
BCKI0005: 24X7 start time => (start time)
BCKI0006: 24X7 end time => (end time)
BCKI0007: LOGGING file size => (count) bytes
```

BCKI0027 is issued if a **stoplogging** request is made or forced due to some error, but the cache is empty, indicating no updates took place.

```
BCKI0027: Nothing to do (empty cache!)
```

When **Logging** is turned off by something other than a normal completion to a **stoplogging** request, **BCKI0032** is issued, indicating that logging has been disabled.

```
BCKI0032: logging mode now disabled
```

When the **LOGGING** file is created in OS/390, **BCKI0033** is issued, naming the file and displaying the parameters that were used.

```
BCKI0033: Creating file (filename) with parms (characteristics)
```

Warning Messages

Warning messages in 24x7 carry the prefix **BCKW**, identifying them by process and type. The first three letters, **BCK**, signify a backup process. The fourth letter, **W**, identifies it as a warning message.

Warning messages are issued when the logging process detects an unexpected event that is not necessarily an error.

If more than one startlogging request is issued, all but the first request prompts a **BCKW4000** message.

```
BCKW4000: (startlogging) logging has already been started!
```

If more than one stoplogging request is received all but the first request prompts either a **BCKW4001** or a **BCKW4005** message. A **BCKW4001** message will only appear if two people from two terminals entered the stoplogging request at the same time.

```
BCKW4005: ServiceCenter DB synchronization cancelled.
BCKW4001: Logging is already inactive.
```

BCKW4002 is issued if the system is not able to close the logging file.

```
BCKW4002: (stoplogging) Unable to close logfile (filename)
```

If some error occurs, and the logging routines force a synchronization to occur, **BCKW4003** and **BCKW4004** are issued, indicating that a forced synchronization has started.

```
BCKW4003: LOGFILE to ServiceCenter DB forced synchronization started
...
BCKW4004: Cache is being processed. Please wait ...
```

Error Messages

Error messages in 24x7 carry the prefix **BCKE**, identifying them by process and type. The first three letters, **BCK**, signify a *backup* process. The fourth letter, **E**, identifies it as an *error* message.

Error messages are issued when the logging process detects an error that forces termination of the logging process. The logging process attempts to force a synchronization of the database to protect the data. The error will have to be corrected and the backup process restarted.

BCKE8000 is issued if a process does not have write access to the logging file.

```
BKCE8000: Unable to access file (filename)
```

The logging file is created by the process that issues the **startlogging** request. **BCKE8001** is issued if the file cannot be created.

```
BCKE8001: Error (error code) attempting to create LOGGING file
(filename)
```

BCKE8002 and **BCKE8018** are issued if the process cannot read the log file. The P4 record that could not be read is indicated by message **BCKE8018**.

```
BCKE8002: Error attempting read logfile
BCKE8018: Error reading log record:
p_num = (file), rec_num = (record), rec_len = (count),
bytes = (size)
```

If the logging control block was not acquired during system initialization, the **startlogging** request is denied, and **BCKE8003** is issued.

```
BCKE8003: (startlogging) Error pf_block structure was not initialized
```

If a process is not able to open the logging file, **BCKE8004** is issued. *This is the worst case*. The process must stop the logging before database corruption can occur. Since the process cannot open the file, the database is re-synchronized. This error is unlikely to occur.

```
BCKE8004: Error (error code) attempting to open file (filename)
```

During system startup, the logging function attempts to acquire control blocks that are used if logging is requested. **BCKE8005** is issued if the request to acquire storage fails.

```
BCKE8005: Error allocating (count) bytes
```

When the logging file is created during a startlogging request, the permissions on the file are changed so that all other processes have access to the file. **BCKE8006** is issued if the request to change permissions on the file fails.

```
BCKE8006: Unable to change permission on file (filename)
```

At various times, the logging process needs to acquire storage for updates that are being made. **BCKE8008** is issued if storage is not available.

```
BCKE8008: Error allocating BLOCK HEADER of (count)
```

BCKE8010 is issued if an error is returned on an lseek() request to a file. This indicates an error in the file system or a programming error.

```
BCKE8010: Error (error code) attempting to seek to position
(number) in logfile
```

BCKE8011 or **BCKE8012** are issued if an error is returned on a read() request to a file. This indicates an error in the file system or a programming error. **BCKE8011** indicates a total read failure. **BCKE8012** indicates a partial read failure since some data was read but not all the data that was expected.

```
BCKE8011: Error (error code) attempting read logfile
BCKE8012: Invalid read on logfile: Wanted (count) bytes but received
only (count) bytes
```

BCKE8015 and **BCKE8013** are issued if an error occurs while attempting to write data to the logging file. The P4 data that could not be written is displayed in the **BCKE8013** message.

```
BCKE8015: Error (error code) attempting to write to LOGGING file
(filename)
BCKE8013: Error writing log record:
p_num = (file), rec_num = (record), rec_len = (count), bytes =
(size)
```

BCKE8017 is an internal programming error indicating that ServiceCenter is attempting to read a different number of bytes for a P4 record than were written for that record.

```
BCKE8017: bytes > rec_len: bytes = (size), rec_len = (count)
```

During the log synchronization process, records are read from the logging file and applied to the ServiceCenter P4 file system. **BCKE8021** is issued when an error occurs while writing the data to P4.

```
BCKE8021: Error (error code) attempting to write to ServiceCenter DB!
```

When two processes receive errors at the same time and attempt to start a forced synchronization, **BCKE8023** is issued for all but the first process.

```
BCKE8023: ServiceCenter DB synchronization cancelled.
```

The logging routine has a lock for serializing access to the logging file and for controlling blocks. **BCKE8026** is issued when the lock fails. **BCKE8027** is issued when the lock fails to release.

```
BCKE8026: (startlogging) Failed to get 24X7 LOCK, errcode = (error
code)
BCKE8027: (stoplogging) Failed to release 24X7 LOCK, errcode =
(error code)
```

4 P4 File System Utility

CHAPTER

This chapter was designed to provide ServiceCenter database administrators with a description of SCDBUTIL and how to use it.

Topics in this chapter include:

Overview

Use the ServiceCenter database Utility (SCDBUTIL) to check for P4 file system problems or to compress database files. The SCDBUTIL options documented in this guide are based on LFSCAN or LFMAP. The other options are not documented in this guide. Do not use them without direction by customer support.

LFSCAN is a ServiceCenter database maintenance tool, part of the SCDBUTIL suite. The LFSCAN utility checks the P4 file system for errors and inconsistencies. Peregrine recommends running this utility once a week, or at least monthly.

Note: IR files are not part of the P4 file system and cannot be checked by any of the LFSCAN utilities. To check IR files, consult customer support for use of DBEXER.

LFMAP is a ServiceCenter database maintenance tool, part of the SCDBUTIL suite. It repacks data records and index nodes into free space within the same physical file. LFMAP moves data toward the front of the physical file and truncates the file behind the last used record (except on OS/390). It can also be used to print out all allocations in the file system.

How to Run SCDBUTIL in Foreground (Interactively)

SCDBUTL can be run interactively in Windows, Unix and OS/390.

Windows

By default, the SCDBUTIL executable is located in the `\ServiceCenter\RUN` directory.

To run SCDBUTIL interactively in Windows:

1. Open a Windows Command prompt or the Window RUN dialog box.
2. Go to the `ServiceCenter\Run` directory and enter the following command:
   `scenter -util`

For example:

```
C:\Program Files\ServiceCenter\RUN> scenter -util

ServiceCenter Database Maintenance Utility
(Version: 5.1.0.0 Build: 0013)

1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan & fix P4 file system errors (LFSCAN)

x) EXIT

Enter your choice: x

exit
```

Unix

By default, the SCDBUTIL executable is located in the **/ServiceCenter/RUN** directory.

To run SCDBUTIL interactively in Unix:

1. Open a Unix Command prompt.
2. Go to the **/ServiceCenter/RUN** directory
3. To have all input and output captured in a file, enter the following Unix command:
   script scriptname.out

   For example:

```
/home/scdev/hp_10: script scriptname.out
Script started, file is scriptname.out
/home/scdev/hp_10:scenter -util

ServiceCenter Database Maintenance Utility
(Version: 5.1.0.0 Build: 0013)

1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
5) Physical file exerciser (PFEXER)
```

```
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan & fix P4 file system errors (LFSCAN)

x) EXIT

Enter your choice: x
/home/scdev/hp_10: exit
Script done, file is scriptname.out
/home/scdev/hp_10:
```

4 Issue the following command to start the ServiceCenter Database Utility:

scenter -util

5 The script ends when the forked shell exits. (Use control-D to exit the Bourne shell (sh(1)), and exit, logout or control-d (if ignoreeof is not set) for the C-shell, csh(1)).

OS/390

In OS/390 (MVS) there is a CLIST file called SCCLIST included in SAMPLIB dataset. This CLIST can be started from the operating system's Command prompt using this command:

TSO EX '*<FULLY-QUALIFIED-NAME-OF-YOUR-SAMPLIB>*(SCCLIST)'

The same command without the leading TSO can be used within the ISPF option =6. Either way the CLIST asks for additional information. You should be prepared to have all of the following information:

- The fully qualified name of load library
- The path to the database files
- The fully qualified name of PARMS dataset
- You will be asked for the applid, too, but you can enter XXX there.

For example:

```
Menu List Mode Functions Utilities Help
 ------------------------------------------------------------------
        ISPF Command Shell
 Enter TSO or Workstation commands below:
 ===> EX 'SC.V5ROMO.SAMPLIB(SCCLIST)'
 Place cursor on choice and press Enter to Retrieve command
 =>
 =>
```

```
 ENTER RUN TYPE (FOR EXAMPLE: AGEXER,SYEXER OR SCDBUTIL)
scdbutil
 ENTER APPLID TO USE
xxx
 ENTER LOADLIB TO USE (FOR EXAMPLE: 'PREFIX.V2R1M5.LOAD')
'SC.V5ROM0.LOAD'
 ENTER PREFIX FOR DATABASE FILES (FOR EXAMPLE: PREFIX.A9801)
'SC.SC5'
 ENTER PARMS TO USE (FOR EXAMPLE: 'PREFIX.V3ROM0.SAMPLIB(PARMS)')
'SC.V5ROM0.SAMPLIB(PARMS)'
 ALLOCATING FILES
 RUNNING ''SC.V5ROM0.LOAD(KERNEL)' 'P=SCDBUTIL I=XXX'
 00130 15:18:53 ServiceCenter Version 5.0.0 Build 0027 ON CPU ID
 020619
1 05/09/00 15:18:56 Character translation initialized for default
language
00130 15:18:56 Database File SC.SC5.SCDB.FRE Allocated From JCL Using
DD:SCDBFRE
 00130 15:18:56 Database File SC.SC5.SCDB.ASC Allocated From JCL
Using DD:SCDBASC
 00130 15:18:56 Database File SC.SC5.SCDB.LFD Allocated From JCL
Using DD:SCDBLFD
 00130 15:18:56 Database File SC.SC5.SCDB.DB1 Allocated From JCL
Using DD:SCDBDB1
 00130 15:18:57 Database File SC.SC5.SCDB.DB2 Dynamically Allocated
using DD:SCDBDB2
 00130 15:18:57 Database File SC.SC5.SCDB.DB3 Dynamically Allocated
using DD:SCDBDB3
 00130 15:37:46 Database File SC.SC5.SCDB.DB4 Dynamically Allocated
using DD:SCDBDB4
 00130 15:37:46 Database File SC.SC5.SCDB.DB5 Dynamically Allocated
using DD:SCDBDB5
 00130 15:37:47 Database File SC.SC5.SCDB.DB6 Dynamically Allocated
using DD:SCDBDB6
 00130 15:37:48 Database File SC.SC5.SCDB.DB7 Dynamically Allocated
using DD:SCDBDB7
  1 05/09/00 15:37:48 Creating resources for system '12670' with key
3137f00
  1 05/09/00 15:37:48 Initializing ServiceCenter on System Serial
6199672 (0.94.153.120)
  1 05/09/00 15:37:50 Warning! Unable to open IR stop words file:
dsn: SC.SC5.english.stp
 00130 15:37:50 Startup event received

ServiceCenter Database Maintenance Utility
(Version: 5.1.0.0 Build: 0013)
2 05/09/00 15:37:51 Character translation initialized for default
language
  2 05/09/00 15:37:51 Attaching to resources for system '12670' with
key 3137f00
1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
```

```
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan and fix P4 file system errors (LFSCAN)
x) EXIT
 Enter your choice:
 exer: x
 00130 15:49:25 User exer, pid 2 terminated
    ServiceCenter CPU Time = 4.184 Seconds
    Task CPU Time = 0.274 Seconds
    Max Malloc Memory = 940209 Bytes
    Data Base Reads/Read I/O's = 215/18
    Data Base Writes/Write I/O's = 0/0
    Data Base Extends = 0
    Semaphore Locks/Waits/Posts = 1162/0/0
    Page Gets/Frees = 30/17
    Getmains/Freemains = 28/24
    Qsam Opens/Gets/Puts = 0/0/0
    Vsam Opens/Gets/Puts = 0/0/0
    File Opens/Reads/Writes = 0/0/0
    Kernel/IO Calls = 43/1
 00130 15:49:25 I/O task termination
 00130 15:49:25    I/O SUMMARY FOR 1000000 BYTE CACHE
 FILE BLOCKSIZE   EXTENTS   I/O'S   HITS    MISSES  AVOIDED
  0   512         1         1       10      1       90.9%
  1   3860        1         4       19      4       82.6%
  2   3860        2         1       7       1       87.5%
  3   3860        2         12      54      12      81.8%
  4   3860        1         0       0       0       0.0%
  5   3860        1         0       0       0       0.0%
  6   3860        1         0       0       0       0.0%
  7   3860        1         0       0       0       0.0%
  8   3860        1         0       0       0       0.0%
  9   3860        1         0       0       0       0.0%
                            ____    ____    ___     _____
                            18      90      18      83.3%
 00130 15:49:25 Deallocating Files
 00130 15:49:28 semaphore key = 0x3137f00 terminated, total semaphore
waits = 0
  1 05/09/00 15:49:28 Shutdown of ServiceCenter environment has
completed
 00130 16:42:20 User io, pid 1 terminated
    ServiceCenter CPU Time = 4.414 Seconds
    Task CPU Time = 0.473 Seconds
    Max Malloc Memory = 1476019 Bytes
    Data Base Reads/Read I/O's = 0/0
    Data Base Writes/Write I/O's = 0/0
    Data Base Extends = 0
    Semaphore Locks/Waits/Posts = 25/0/0
    Page Gets/Frees = 9/0
    Getmains/Freemains = 27/10
    Qsam Opens/Gets/Puts = 0/0/0
```

```
   Vsam Opens/Gets/Puts = 0/0/0
   File Opens/Reads/Writes = 0/0/0
   Kernel/IO Calls = 71/0
00130 16:42:20 Maximum virtual storage used by servicecenter =
16015k
00130 16:42:20 Kernel Accumulated Accounting Statistics
   ServiceCenter CPU Time = 4.445 Seconds
   Task CPU Time = 0.435 Seconds
   Max Malloc Memory = 1476019 Bytes
   Data Base Reads/Read I/O's = 215/18
   Data Base Writes/Write I/O's = 0/0
   Data Base Extends = 0
   Semaphore Locks/Waits/Posts = 1187/0/0
   Page Gets/Frees = 39/17
   Getmains/Freemains = 55/34
   Qsam Opens/Gets/Puts = 0/0/0
   Vsam Opens/Gets/Puts = 0/0/0
   File Opens/Reads/Writes = 0/0/0
   Kernel/IO Calls = 114/1
00130 16:42:20 Kernel task termination
FREEING FILES
***
```

How to Run SCDBUTIL in Background (Batch)

To start SCDBUTIL in background (for automation of LFSCAN/LFMAP) you will need to create an input file to hold all the commands you would enter interactively via keyboard. This information will be used by prompts that appear while the scan is running. The actual input for each option is found in the appropriate section.

Note: The input file must be a text file. If you save it as a Microsoft Word document, for example, SCDBUTIL will not be able to read it.

Sample Input File

```
6    // call LFSCAN
0    // check all pools
x    // Exit SCDBUTIL
```

In the input file, characters that appear to the right of the // marker in each line, are treated as comments documenting the steps in the batch file. The text on the left-hand side is exactly the same as what you would enter manually.

Windows

By default, the SCDBUTIL executable is located in the \ServiceCenter\RUN directory.

To run SCDBUTIL in background on Windows:

1. Prepare the input file as indicated in *How to Run SCDBUTIL in Background (Batch)* on page 55, using the correct input for the option you want to run. (See the section on each option for this information.)
2. Shut down the ServiceCenter server, if running. (This will terminate the clients as well.)
3. Open the DOS Command prompt or the Window RUN dialog box.
4. Go to the \ServiceCenter\RUN directory and enter the following command:

   scenter -util <filename.in >filename.out

   where filename.in is the name of the input file you have prepared and filename.out is the name of the file holding all the output afterwards.

Unix

By default, the SCDBUTIL executable is located in the /ServiceCenter/RUN directory.

To run SCDBUTIL in background on Unix:

1. Prepare the input file as indicated in *How to Run SCDBUTIL in Background (Batch)* on page 55, using the correct input for the option you want to run. (See the section on each option for this information.)
2. Shut down the ServiceCenter server, if running. (This will terminate the clients as well.)
3. Open a Unix Command prompt.
4. Go to the /ServiceCenter/RUN directory and enter the following command:

   scenter -util <filename.in >filename.out

   where filename.in is the name of the input file you have prepared and filename.out is the name of the file holding all the output afterwards.

OS/390

In your SAMPLIB there is a JCL called SCDBUTIL, which is designed to start SCDBUTIL in background. The input file must be a member in your SAMPLIB. You can use the already existing members LFMAP or LFSCAN for that, however, it is strongly recommended to create a new member. Once you have created an input file, edit the SCDBUTIL job control. Read the instructions included in the JCL carefully. You have to tell the JCL where your input file, your PARMS dataset and your load library are located.

To run SCDBUTIL in background in OS/390:

1. Create a new member NEWNAME in your SAMPLIB data set.
2. Put the input data, indicated in *How to Run SCDBUTIL in Background (Batch)* on page 55, into the new SAMPLIB member, using the correct input for the option you want to run. (See the section on each option for this information.)
3. Shut down the ServiceCenter server, if running. (This will terminate the clients as well.)
4. Make a backup copy of the JCL in the SCDBUTIL member of the SAMPLIB dataset that is provided with ServiceCenter.
5. Edit the copy of the JCL member SCDBUTIL by following the instructions provided in the member, specifying UTILMEM=NEWNAME on the PROC statement in the JCL.
6. Submit the job.

LFSCAN Based Options

The LFSCAN based utilities:

- *Option 6) Logical file Scan Utility* on page 58 — performs a database consistency check on the ServiceCenter P4 file system scanning and reporting any inconsistencies and errors found in a P4 file system. It does not attempt to fix any of the problems found.
- *Option 7) Quick Scan with Minimal Error Checking* on page 70 — performs a limited database consistency check that requires the lowest amount of memory. It is used instead of the LFSCAN in environments with very large file systems or limited memory. It can also delete data records for files without a dbdict record, but it does not fix anything else.

- *Option 8) Remove Invalid Associators* on page 71 — removes (unallocates) associators LFSCAN detected as corrupted.
- *Option A) Scan and Fix P4 File System Errors* on page 74 — detects and repairs problems in the P4 file system.

Option 6) Logical file Scan Utility

SCDBUTIL option **6) Logical file Scan Utility (LFSCAN)** does the following:

1. Validates that the file size headers at the beginning of each physical file are valid and compares it with the operating system file size (not on 0S/390.)
2. Validates the following information in the **dbdict** records:
   - filename
   - data logical file number
   - index logical file number
   - pool numbers for data and index logical file
   - record lengths for data (0 = variable)
   - index logical file (between 256 and 32768 = fixed)
   - root record address
3. Validates the following information in the logical file descriptors:
   - pool numbers into which data records and index nodes are to be written
   - record size for data logical files 0x0000 (= variable)
   - index logical files (= fixed), must be a power of 2 between 256 and 32768
   - index logical files and record size for data logical files match the information stored in the **dbdict** record
4. Validates that index nodes are not corrupted, including checking that:
   - Key numbers in index entries are in ascending order.
   - Key types are consistent.
   - Addresses of child index records are valid.
   - Referred associator records exist and belong to the matching data logical file.
5. Validates the following information in the associator records:
   - logical file number
   - pool number where the record resides

- pool number is defined for this data logical file
- address of data record
- record length (maximum record length is a power of 2 between 8 and 32768)
- current record length is less than allocated record length.
- each associator is included at least once in the index structure of the file

6 Validates the following information in the free list chain:
- All entries on the free list chain point to free space.
- There are no free list loops (no entry in the free list chain points back to another entry in the free list chain).

7 There are no multiple allocations or collisions (no two records in the file are occupying the same space or overlapping space.)

8 Prints a summary of how much space is currently in use for each pool and how much space is unused.

9 Prints a summary of all files and how big the data and indexes are.

When to Run LFSCAN

- Regularly, e.g., weekly, for standard maintenance and to monitor system file sizes.
- Whenever the ServiceCenter system has a hard stop:
  - A power outage occurs while ServiceCenter is running.
  - The ServiceCenter server is turned off while ServiceCenter is running.
  - ServiceCenter is killed while I/O processes are still active.
- If the ServiceCenter system has random signal 11 errors or Dr. Watsons, especially if the frequency or volume is increasing.
- If any P4 physical file (data pool) is approaching the 2GB limit. LFSCAN lets you determine which file(s) within the data pool are causing the growth, so they can be cleaned out and compressed.

When Not to Run LFSCAN

When the sc.log file indicates index errors. The messages in the sc.log file provide sufficient information to regen the affected file and fix it. Running LFSCAN may cause unnecessary additional system downtime without providing any new information.

Note: Make a habit of checking the **sc.log** file regularly, scanning for problems or unusual circumstances.

How to Run LFSCAN

The length of time LFSCAN runs depends on the size of the files, the performance of the computer and the disk, and the memory available. LFSCAN can run for over an hour on a large database.

Important: It is important to capture LFSCAN's output. To check the output, you have to run LFSCAN in background on Windows and OS/390. It is preferable to do the same on Unix, but you also might choose to run it in foreground using the script command. Either way, check the output file for any error messages and send the file to Peregrine Customer Support if necessary.

To run LFSCAN:

1 Shut down ServiceCenter if it is currently running. If ServiceCenter is active when LFSCAN is called, the scan will not run.

2 Run **scenter -util, 6) Logical file scan utility (LFSCAN)**, interactively (Unix only), as directed in *How to Run SCDBUTIL in Foreground (Interactively)* on page 50, being sure to capture the output with the script utility.

The following prompts are displayed:

Enter your choice:

a Enter **6** to run LFSCAN.

Select pools 1 through 9 or enter 0 to select all pools.

Which pool? (0)

b Enter **0** to select all pools. Otherwise, enter the number of the pool you want to select. The number of pools available depends on how many pools are actually defined.

When LFSCAN is finished, the SCDBUTIL menu is displayed again.

c Enter **x** to exit SCDBUTIL and return to the Command prompt.

— Or —

3 Run **scenter -util, 6) Logical file scan utility (LFSCAN)**, in background (OS/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

Use the following example as the basis for your input file.

```
6   // Call LFSCAN
0   // Check all pools
x   // Exit SCDBUTIL
```

4 Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

Explanation of LFSCAN Output

Following you will find a sample LFSCAN output along with information what it means.

```
ServiceCenter Database Maintenance Utility
(Version: 5.0.0 Build: 0027)
```

The title tells you the release used to run LFSCAN.

```
1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan & fix P4 file system errors (LFSCAN)

x) EXIT
```

The title is followed by the main menu of SCBDUTIL.

```
Enter your choice: 6
```

At the prompt you enter 6, the regular LFSCAN.

```
LFSCAN Utility

Select pools 1 through 9 or enter 0 to select all pools.
Which pool? (0)
```

Enter **0** to select all pools. Otherwise, enter the number of the pool you want to select. The number of pools available depends on how many pools are actually defined. Enter a single pool here only if you are running low on memory.

```
Checking pool index file (SCDB.FRE)

pool 1 consists of physical files: 1
pool 2 consists of physical files: 2
pool 3 consists of physical files: 3
pool 4 consists of physical files: 4
pool 5 consists of physical files: 5
pool 6 consists of physical files: 6
pool 7 consists of physical files: 7
pool 8 consists of physical files: 8
pool 9 consists of physical files: 9
```

The first part of the output displays how the pools in the P4 file system are set up. In this example each pools consists of just one physical file. Due to this setup, all pools are limited to 2 GB, which is the physical file size limit.

```
Filesize in file header of physical file 0: 1280
Filesize in file header of physical file 1: 14450688
Filesize in file header of physical file 2: 65536
Filesize in file header of physical file 3: 94208000
Filesize in file header of physical file 4: 5931008
Filesize in file header of physical file 5: 262144
Filesize in file header of physical file 6: 163840
Filesize in file header of physical file 7: 163840
Filesize in file header of physical file 8: 131072
Filesize in file header of physical file 9: 131072
```

These lines of output show you the current size of all physical files, as it is stored in the file headers. LFSCAN also checks that the physical file length of each file matches these values, if not you will find an error message here.

```
Scanning associator file for dbdict records
 20% scanned so far
 40% scanned so far
 60% scanned so far
 80% scanned so far
100% scanned so far

Total of  903168 associators scanned, 495 dbdicts found

Starting sort of 495 dbdicts
Sort is  20% complete
Sort is  40% complete
Sort is  60% complete
Sort is  80% complete
Sort is 100% complete
```

These messages show you the progress while LFSCAN searches for all **dbdict** records in preparation of the upcoming list.

```
Filename  ILF#     Pool    DLF#    Pool    ( root record  )   AscRec#
code      3:       3       4:      3       ( 3:     2/1024)   2
dbdict    2:       3       1:      3       ( 3:     4/ 256)   1
format    296:     3       295:    3       ( 3:     7/1024)   18351
```

Above is the first file list in the LFSCAN output. It shows you the following about every file (including file mapped to SQL) in the P4 file system:

- filename
- file numbers (index and data logical file)
- the pools the file is assigned to
- address of the root record for this file
- associator record number for the **dbdict** record

```
Checking associator file
 20% processed so far
 40% processed so far
 60% processed so far
 80% processed so far
100% processed so far

 903168 total associatorsj checked
```

After the first file list LFSCAN verifies the records being stored in the **scdb.asc** file. If it finds any error, it will issue a message here, otherwise you will just find messages stating the progress being made.

```
Scanning for data records
 20% processed so far
 40% processed so far
 60% processed so far
 80% processed so far
100% processed so far

 903168 total associators checked, 142804 data records found

Checking data records
 20% scanned so far
 40% scanned so far
 60% scanned so far
 80% scanned so far
100% scanned so far
```

After verifying the **scdb.asc** file, LFSCAN first searches for and then checks all data records.

```
Scanning index records ...
  Scanned a total of 495 level 1 records
```

```
  Scanned a total of 959 level 2 records
  Scanned a total of 2687 level 3 records
  Scanned a total of 3606 level 4 records
```

Next LFSCAN validates all index records. Again, if there is no error found, you will only see the above progress messages.

```
Checking free lists

Checking pool 1 of 9
Checked 8 byte records,  0 records.
Checked 16 byte records,  756671 records.
Checked 32 byte records,  1 record.
Checked 64 byte records,  0 records.
Checked 128 byte records,  0 records.
Checked 256 byte records,  0 records.
Checked 512 byte records,  1 record.
Checked 1024 byte records,  0 records.
Checked 2048 byte records,  1 record.
Checked 4096 byte records,  1 record.
Checked 8192 byte records,  0 records.
Checked 16384 byte records,  0 records.
Checked 32768 byte records,  0 records.
Checked emergency records (16 bytes),  10 records.
12113584 bytes in 756685 free space records in pool 1

...

Checking pool 9 of 9
Checked 8 byte records,  1 record.
Checked 16 byte records,  1 record.
Checked 32 byte records,  1 record.
Checked 64 byte records,  1 record.
Checked 128 byte records,  1 record.
Checked 256 byte records,  1 record.
Checked 512 byte records,  1 record.
Checked 1024 byte records,  0 records.
Checked 2048 byte records,  1 record.
Checked 4096 byte records,  7 records.
Checked 8192 byte records,  2 records.
Checked 16384 byte records,  1 record.
Checked 32768 byte records,  0 records.
Checked emergency records (32768 bytes),  2 records.
130040 bytes in 20 free space records in pool 9
```

These messages are generated while LFSCAN checks the free space chains. They show how much free space is available in each pool. They also show how many emergency free space records are available and what size they are.

```
Starting sort of 1099774 records
Sort is  20% complete
Sort is  40% complete
Sort is  60% complete
Sort is  80% complete
Sort is 100% complete
```

These messages show the progress of LFSCAN while sorting. This phase is necessary for the following output.

```
Highest used address in physical file 1 = 0xdc8000 (14450688)
44312 bytes (0.3%) inaccessible in 842 holes in physical file 1
12157896 bytes (84.1%) unused in 88884 bins (40922) in physical file
1

Highest used address in physical file 2 = 0x10000 (65536)
17752 bytes (27.1%) inaccessible in 4 holes in physical file 2
33816 bytes (51.6%) unused in 15 bins (5) in physical file 2

Highest used address in physical file 3 = 0x59d8000 (94208000)
135168 bytes (0.1%) inaccessible in 12 holes in physical file 3
6191704 bytes (6.6%) unused in 16792 bins (11389) in physical file 3

Highest used address in physical file 4 = 0x5a8000 (5931008)
5835256 bytes (98.4%) unused in 550 bins (131) in physical file 4

Highest used address in physical file 5 = 0x40000 (262144)
1024 bytes (0.4%) inaccessible in 1 hole in physical file 5
159736 bytes (60.9%) unused in 62 bins (33) in physical file 5

Highest used address in physical file 6 = 0x28000 (163840)
78840 bytes (48.1%) unused in 13 bins (4) in physical file 6

Highest used address in physical file 7 = 0x28000 (163840)
159736 bytes (97.5%) unused in 25 bins (4) in physical file 7

Highest used address in physical file 8 = 0x20000 (131072)
99064 bytes (75.6%) unused in 14 bins (5) in physical file 8

Highest used address in physical file 9 = 0x20000 (131072)
130040 bytes (99.2%) unused in 14 bins (2) in physical file 9
```

The previous messages give information on the space usage of the physical files. There can be up to three lines for each file.

- The first line indicates the highest used address in the physical file, including data records, index records, free space records, etc. This address should match the file size stored in the file header.

- The next line only appears if there are holes in the physical file. A hole is a space in the file that is not referenced and cannot be reused, as it is not part of a free space chain. If a file has a large number of holes, an LFMAP is recommended to reorganize the records and to integrate the space into the free space chain again.
- The last line shows the total amount of unused space in the physical file; which is the sum of free space and holes. The first number is the amount of free space in bytes and in percent of the physical file size. If these numbers are high, an LFMAP is recommended to compress the physical file. The next two numbers in the message tell the number of bins and areas of unused space.

For example, an area of unused space that starts at address 16 and ends at address 48 would be 1 area with 32 bytes of unused space. Bins are the minimum number of records needed to fill an area. The addressing in P4 demands that the starting address of each record must be divisible by its length. In the example, the record length is 32 bytes but the starting address of 16 is not divisible by 32. Therefore, P4 needs at least 2 (16 byte) records to fill this area, one 16 Byte record starting at address 16 and one 16 byte record at address 32.

```
--------Index-----------     -----Data----
D   Recs. Entr. Del   Ks  I Recs. KBytes  Filename             (DLF# ILF#)
3   52    494   7     1     494   611     dbdict               (1    2   )
3   83    2458  0     1     2458  9449    code                 (4    3   )
1   1     2     0     1     2     1       printer              (5    6   )
1   1     1     0     3     1     1       port                 (7    8   )
1   1     23    0     2     12    2       pmstatus             (9    10  )
...
1   1     16    0     2   I 2     3       helptext             (139  140 )
...
1   1     1     0     1     1     1       patcortadmin         (1542 1541)
1   1     3     0     1     3     1       patcorttemplate      (1544 1543)
1   1     0     0     1     0     0       applicationrevision  (1546 1545)
```

The previous lines of output show the final file list, which contains all files found in the P4 file system. Those without a **dbdict** record are marked with a file name starting with ** **dbdict missing**. These messages show the system-wide totals of how much space is used by which file.

Note: Files mapped to an RDBMS are not included they will only show up in the first file list.

Information in the final file list includes:

- Depth of index structure. That means how many levels of index records are used for the file.
- Number of index records.
- Number of index entries.
- Number of deleted index entries.
- Number of keys defined for the file.
- IR Expert flag, if the file has an IR Expert key it will be marked with a capital 'I'
- Number of data records.
- Space used by the data logical file in kilobytes.
- Error flag. This flag is located in front of the filename and can contain the following values:

  * indicates some kind of dbdict or LFD error has been detected.

  - indicates some kind of associator error has been detected.

  + indicates some kind of index error has been detected.

  ? indicates some kind of data error has been detected.

  Additionally, there might be a capital R in front of the error flag, indicating that the file needs to be regenerated.
- the filename
- the logical file numbers

```
80369760 bytes in 143297 variable records
73792 bytes in 2 spanned records with 4 blocks
7892992 bytes in 7747 index records for 495 files
Total of 198256 bytes (0.2%) inaccessible in 859 holes
Total of 24846088 bytes (21.5%) unused in 106369 bins (52495)
```

Finally, LFSCAN issues some overall totals showing how many record it has found in the file system, how much space they need and much free space is available.

Note: If a corruption was found by LFSCAN, you will be notified at this point, so a quick look at the end of the LFSCAN shows if action is required.

```
1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
```

```
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan & fix P4 file system errors (LFSCAN)

x) EXIT

Enter your choice: x
```

Once LFSCAN has completed, the SCDBUTIL main menu is displayed again. Enter **x** to exit.

LFSCAN Condition Codes

LFSCAN returns different condition codes indicating different error situations found during the database integrity check.

To check the condition code:

- on Unix by using the **echo** command. The variable name changes with the shell.
  - for the ksh and bash shells, use **echo $?**
  - for the csh shell, use **echo $status**
- on Windows by using the command **echo %errorlevel%**
- on OS/390 by browsing the job's output

When using condition codes in scripts e.g. to notify the system administrator automatically, be aware that any following command will overwrite the variables, even the **echo** command will change them to a value of zero indicating that the **echo** command completed successfully.

To save the variable's content, the first command after running LFSCAN should be:

- on Unix
  - for the ksh and bash shells, use **export SAMPLE_VAR=$?**
  - for the csh shell use **setenv SAMPLE_VAR $status**
- on Windows, use **set sample_var=%errorlevel%**

Sample Script for Windows 2000/NT:

Create a file called scan.bat containing:

```
@echo off
echo Script starting
scenter- util <lfscan.in >lfscan.out
if errorlevel 1 goto error
goto end
:error
echo Return code was %errorlevel%
:end
echo Script complete
```

Table 4-1: Codes Returned by LFSCAN

Code	Definition
0	No problems were detected in the P4 file system by LFSCAN
4	These warnings cause the condition code 4 to be returned: ■ Data records were found that do not belong to any logical file. ■ Data records that are not referenced by any key and, therefore, are irretrievable without a regen.
8	These categories of errors cause the condition code 8 to be returned: ■ Associator errors ■ Data record errors ■ Index errors
12	These errors cause the condition code 12 to be returned: ■ dbdict errors ■ LFD record errors ■ Multiple allocations (collisions) ■ Free space chain errors
16	If the LFSCAN was terminated early due to one these reasons, the condition code 16 will be returned: ■ The dbdict file is inaccessible ■ Either the scdb.fre, the scdb.asc, the scdb.lfd, or the scdb.db1 file could not be accessed

Option 7) Quick Scan with Minimal Error Checking

SCDBUTIL option 7) Quick Scan with minimal error checking (LFSCAN) performs a database consistency check on the ServiceCenter P4 file system. Although option 6) Logical file scan utility (LFSCAN) is the recommended way of detecting P4 file inconsistencies, it may be necessary to use the Quick Scan option in environments with very large file systems or limited memory.

Option 7 (Quick Scan) differs from option 6 (LFSCAN) in that:

- Quick Scan uses much less memory and enables very large databases to run an LFSCAN. It does not keep track of all the records. Therefore, it does not sort them and is unable to detect multiple allocations.
- Quick Scan allows turning off the check of index records. Although this speeds up the scanning process, it neither checks the index records nor checks if all data records are included in the index structure.
- Quick Scan can automatically delete data records found that no longer belong to any database.

How to Run Quick Scan

The length of time Quick Scan runs depends on the size of the files, the performance of the computer and the disk, and the memory available. LFSCAN can run for over an hour on a large database.

Important: It is important to capture Quick Scan's output. To check the output, you have to run Quick Scan in background on Windows and OS/390. It is preferable to do the same on Unix, but you also might choose to run it in foreground using the script command. Either way, check the output file for any error messages and send the file to Peregrine Customer Support if necessary.

To run Quick Scan:

1. Shut down ServiceCenter if it is currently running. If ServiceCenter is active when Quick Scan is called, the scan will not run.
2. Run option 7) Quick Scan with minimal error checking (LFSCAN) interactively (Unix only), as directed in*How to Run SCDBUTIL in Foreground (Interactively)* on page 50, being sure to capture the output with the script utility.

The following prompts are displayed:

Enter your choice:

a Enter 7 to run Quick Scan.

Should I determine the size of the index in each file? (Y/n).

b Enter y or n depending on whether you want to check the index structure.

Do you want me to remove data not associated with a DBDICT? (y/N).

c Enter n.

Warning: Always enter n for this option unless advised by Peregrine Systems Customer Support to enter y.

When Quick Scan is finished, the SCDBUTIL menu is displayed again.

d Enter x to exit SCDBUTIL and return to the Command prompt.
— Or —
Run option 7) Quick Scan with minimal error checking (LFSCAN), in background (0S/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

Use the following example as the basis for your input file.

```
7    // run Quick SCAN
y    // determine the size of the index in each file
n    // do not remove data not associated with a DBDICT
x    // Exit SCDBUTIL
```

3 Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

Option 8) Remove Invalid Associators

SCDBUTIL option 8) Remove Invalid Associators (LFSCAN) removes (unallocates) associators that were detected as corrupted by the LFSCAN. When option 8 is entered, an LFSCAN is started and whenever LFSCAN detects an error with an associator record, this record will automatically be deleted.

This happens for the following errors:

- Associator 9999 (01/01/00 23:59:59 GMT) for 'ABCD' (DLF#99) points to an invalid physical file (17).

- Associator 9999 (01/01/00 23:59:59 GMT) for 'ABCD' (DLF#99) points beyond EOF: 3:123987546/1024.
- Associator 9999 (01/01/00 23:59:59 GMT) for 'ABCD' (DLF#99): more space used than allocated (35>32).
- Associator 9999 (01/01/00 23:59:59 GMT) for 'ABCD' (DLF#99): allocation 58665 is invalid, must be between 8 and 32768.
- Associator 9999 (01/01/00 23:59:59 GMT) for 'ABCD' (DLF#99): allocation 2556 is invalid, must be power of 2.

How to Remove Invalid Associators

Warning: Once the associators are removed they are gone. Before unallocating you should make sure that either the associators in question are really not needed anymore or that you have a valid backup to regain the information in case of emergency.

The length of time Remove Invalid Associators runs depends on the size of the files, the performance of the computer and the disk, and the memory available. Remove Invalid Associators can run for over an hour on a large database.

Important: It is important to capture Remove Invalid Associators' output. To check the output, you have to run Remove Invalid Associators in background on Windows and OS/390. It is preferable to do the same on Unix, but you also might choose to run it in foreground using the script command. Either way, check the output file for any error messages and send the file to Peregrine Customer Support if necessary.

To remove invalid associators:

1 Shut down ServiceCenter if it is currently running. If ServiceCenter is active when Remove Invalid Associators is called, the scan will not run.

2 Run option **8) Remove Invalid Associators (LFSCAN)** interactively (Unix only), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

The following prompts are displayed:

Enter your choice:

a Enter **8** to Remove Invalid Associators.

When Remove Invalid Associators is finished, the SCDBUTIL menu is displayed again.

b Enter x to exit SCDBUTIL and return to the Command prompt.

— Or —

Run option 8) Remove Invalid Associators (LFSCAN), in background (0S/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Foreground (Interactively)* on page 50.

Use the following example as the basis for your input file.

```
8    // run Remove Invalid Associators
x    // Exit SCDBUTIL
```

3 Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

Option A) Scan and Fix P4 File System Errors

Option A, Scan and fix P4 file system errors (LFSCAN) is the Automatic Repair utility. It is used to detect and repair problems in the P4 file system.

Warning: Since fixing P4 problems often means deleting records, you might lose data that otherwise could have been saved. It is strongly recommended that you run this utility only when being advised by Peregrine Systems Customer Support.

Starting an Automatic Repair does an LFSCAN to gather the necessary pieces of information and to check if there are any file errors before making any repairs.

Once the LFSCAN has been completed, the Automatic Repair fixes detected problems in the following order:

- Resolves multiple allocations/collisions.
- Fixes Database Dictionary errors.
- Fixes any other corrupted records, e.g. data, index, associator or free space records
- Regenerates all files with corrupted index structure.
- Deletes all files that do not have a Database Dictionary record.

Warning: Because of the time involved in regenerating and deleting files, the automatic repair may take some time to complete. The length of time the utility runs depends heavily on the size of the files, the performance of the computer and the disk, and the memory available.

How to Run Automatic Repair

The length of time automatic repair runs depends on the size of the files, the performance of the computer and the disk, and the memory available. automatic repair can run for over an hour on a large database.

Important: It is important to capture Automatic Repair's output. To check the output, you have to run Automatic Repair in background on Windows and OS/390. It is preferable to do the same on Unix, but you also might choose to run it in foreground using the script command. Either way, check the output file for any error messages and send the file to Peregrine Customer Support if necessary.

Under most circumstances, Automatic Repair should be run automatically. Not running it automatically precludes using Batch Mode. During automatic repair, you will be asked *Do you want to run this utility in automatic mode? (y/N)* If you answer n, you will be prompted for every multiple allocation and asked which of the records involved you want to delete to solve the collision. If you enter y, the Automatic Repair will run fully automatically.

Warning: Do not enter n if you want to run the file in batch mode. If you enter n, you will also be asked for a confirmation for every file to be regenerated and for every data logical file to be reset because the dbdict record is missing. These cannot easily be anticipated and put in the batch file.

To run automatic repair:

1. Shut down ServiceCenter if it is currently running. If ServiceCenter is active when Automatic Repair is called, the scan will not run.
2. Run Option A, Scan and fix P4 file system errors (LFSCAN) interactively (Unix only), as directed in *How to Run SCDBUTIL in Foreground (Interactively)* on page 50, being sure to capture the output with the script utility.

   The following prompts are displayed:

   Enter your choice:

   a. Enter a, to run the Automatic Repair.

   Do you want to run this utility in automatic mode? (y/N)

   b. Enter y to have the repair run fully automatically (preferred).

   — or —

   Enter n to be prompted for every multiple allocation and asked which of the records involved you want to delete to solve the collision.

When the Automatic Repair is completed, the SCDBUTIL menu is displayed again.

c Enter x to exit SCDBUTIL and return to the Command prompt.

— Or —

Run option A, Scan and fix P4 file system errors (LFSCAN), in background (0S/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

Use the following example as the basis for your input file:

```
a    //Run the Automatic Repair.
y    //Use automatic mode.
x    // Exit SCDBUTIL.
```

Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

LFMAP Based Options

LFMAP based utilities:

- *Option 4) Logical File Map Utility*, next section — compresses the user data pools (pools 3 and higher).
- *Option 9) Compress of Associator File* on page 86, which compresses pool 1 (the scdb.asc file), but not any other pools.

Option 4) Logical File Map Utility

Warning: Stopping an LFMAP for any reason during the compression process corrupts the database. ALWAYS backup your entire P4 file system before running LFMAP.

When to Run LFMAP

Do not run LFMAP on a regular basis, because it takes a long time to run. Instead, monitor your P4 file system as discussed above, then run LFMAP as needed.

When to run LFMAP:

- Run LFMAP on files approaching the 2GB limit (possibly after moving some logical files to a different data pool if there is not significant empty space).
- Run LFMAP on files that have more than 50MB unused or in holes.
- Run LFMAP on files where a large amount of data has been deleted.
- Run LFMAP if LFSCANs are running unusually slowly. If a file's freelists are long, LFSCANs take much longer than on a file where LFMAP has recently run.

Option 4 (LFMAP) does not allow the ServiceCenter Administrator to compress the **scdb.asc** file. It reorders free space chains in the **scdb.asc** file but does not reorganize the associator records. The Associator Compress Option 9 (Associator Compress) allows you to compress the associator records.

Before repacking data records and index nodes, LFMAP runs a consistency check (LFSCAN) against the P4 data and corrects the dbdict errors and collisions found (multiple allocation errors). If it is not able to resolve all multiple allocations detected, it does not proceed with the compression of the P4 pool. Any other type of P4 corruption, for example, index record errors, data record errors, are ignored by LFMAP since they do not interfere with its task of compressing a P4 pool.

Note: Peregrine Systems recommends that you run the LFMAP utility on an *as needed* basis.

If LFSCAN shows that a large amount of free space or holes can be reclaimed, you should run an LFMAP. You should also run an LFMAP if a message in the ServiceCenter log file similar to the following notifies you of the approaching file size limit for one of your pools:

```
WARNING:Pool <XX> is <XX> percent full.
```

When this message appears, first purge data, then run LFMAP.

The length of time the utility runs depends on the size of the files, the performance of the computer and the disk, and the memory available.

How to Run LFMAP

Important: Before running LFMAP, perform a backup of ServiceCenter.

To run LFMAP:

1 Shut down ServiceCenter if it is currently running. If ServiceCenter is active when LFMAP is called, the scan will not run.

2 Run option `4) Logical file map utility (LFMAP)` interactively (Unix only), as directed in *How to Run SCDBUTIL in Foreground (Interactively)* on page 50, being sure to capture the output with the script utility.

The following prompts are displayed:

Enter your choice:

a Enter `4` to run LFMAP.

Enter Password

b Enter `CARLSBAD`.

Select pools 1 through 9 or enter 0 to select all pools.

Which pool? (3)

c Enter `0` to select all pools. Otherwise, enter the number of the pool you want to select. The number of pools available depends on how many pools are actually defined.

Note: The LFMAP utility does not compress the files `scdb.asc` and `scdb.lfd` (pools 1 and 2). If you select these pools, their free space chains are reordered and all holes in those pools are put back into the free space chains, but they are not compressed and their file size does not necessarily decrease.

Report only? (Y/n)

d Enter `y` to only produce reports. If you enter `y` an LFSCAN is started with some additional options.

— Or —

Enter n to compress database files.

Bypass Free Space Check? (y/N)

e Enter `N` (Preferred) to detect collisions with free space records or general errors with the free space chains. You should answer `N` here unless you are certain that the free space chains are error free.

— Or —

Enter `y` to bypass checks for the free space chains. This makes the LFMAP faster, but it won't be able to detect any collisions with free space records or general errors with the free space chains.

Automatic mode? (y/N)

This options determines how the LFMAP should proceed if it encounters database errors. LFMAP can automatically fix multiple allocations (collisions) and dbdict errors. If it is not able to resolve all multiple allocations detected, it does not proceed with the compression of the P4 pool. Any other type of P4 corruption, for example, index record errors, data record errors, are ignored by LFMAP since they do not interfere with its task of compressing a P4 pool.

f Enter `y` (preferred) to have fix LFMAP try to fix these errors by itself. Since the errors found cannot be predicted, answer y if you are running LFMAP from a script in background.

— Or —

Enter `N` to be prompted on how the error should be fixed.

Are you sure? (y/N)

g Enter `y`.

Full report? (y/N)

h Enter `n` to generate only a partial report. Unless you really want to see where all your records are located, where they will be moved and you have the disk space to hold all the output, you should answer `n`.

— Or —

Enter `y` to get a full report, including a map of all records and all holes being found in the database files. This will generate a huge output.

Report errors only? (y/N)

i Enter `y` (Preferred) to see only the error messages. For a LFMAP it is useful to only report the error messages.

— Or —

Enter `n` to be prompted for different types of messages each of which you can turn on or off.

Do you want the free space chains to start with low addresses after the compress? (Y/n)

j Enter `n` to have the free space chains put in descending order. Future allocations will be done starting at the end of the file.

— Or —

Enter y to have the free space chains put in ascending order. Future allocations will be done from the beginning of the files.

The utility begins the database consistency check. Once space is available to reclaim, the repacking process is started. This might take a while depending the computer's and the hard drive's performance.

When the repacking is completed, some messages stating how space could be regained are displayed, and the SCDBUTIL menu is displayed again.

k Enter x to exit SCDBUTIL and return to the Command prompt.

— Or —

Run option 4) Logical file map utility (LFMAP), in background (0S/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

Use the following example as the basis for your input file:

```
4       // Run LFMAP
CARLSBAD// Password
0       // Compress all pools.
n       // Reports only — or  —  Compress files
n       // Do not bypass free space check
y       // Use Automatic Mode
y       // Confirm Choice
n       // Do not give full report.
y       // Report errors only
y       // Put free space chains in ascending order.
x       // Exit SCDBUTIL
```

3 For OS/390 only, run IDCAMS to copy the compressed ServiceCenter file from its current larger size to a smaller dataset. See *IDCAMS (Used After LFMAP on OS/390 Only)*, next section.

4 Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

IDCAMS (Used After LFMAP on OS/390 Only)

The IDCAMS utility is used to copy a compressed ServiceCenter file from its current larger size to a smaller dataset. Before running IDCAMS you must allocate this smaller dataset. The recommended size of this new file can be determined by using the numbers reported by LFMAP.

For example, if LFMAP indicates that 76% of the current file is needed and the old file allocation was 1000 tracks, then the new file should be at least 76% of that, 760 tracks. To ensure that the new file does not extend too quickly, we recommend that you add approximately 10% to the primary extend, e.g. 840 tracks instead of 760 tracks.

To run the IDCAMS job, ServiceCenter must be shut down, the ServiceCenter file must already have had LFMAP executed against it, and the new ServiceCenter file must have been allocated.

LFMAP automatically generates a JCL step that can be used to start IDCAMS.

This JCL step might resemble the following:

```
//IDCAMS09 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*,HOLD=YES
//SYSUDUMP DD SYSOUT=*,HOLD=YES
//IN09  DD DISP=OLD,DSN=SC5.SCDB.DB7
//OUT09 DD DISP=OLD,DSN=SC5.SCDB.DB7.NEW
//SYSIN DD *
 REPRO INFILE( IN09 ) OUTFILE( OUT09 ) COUNT( 14 )
/*
```

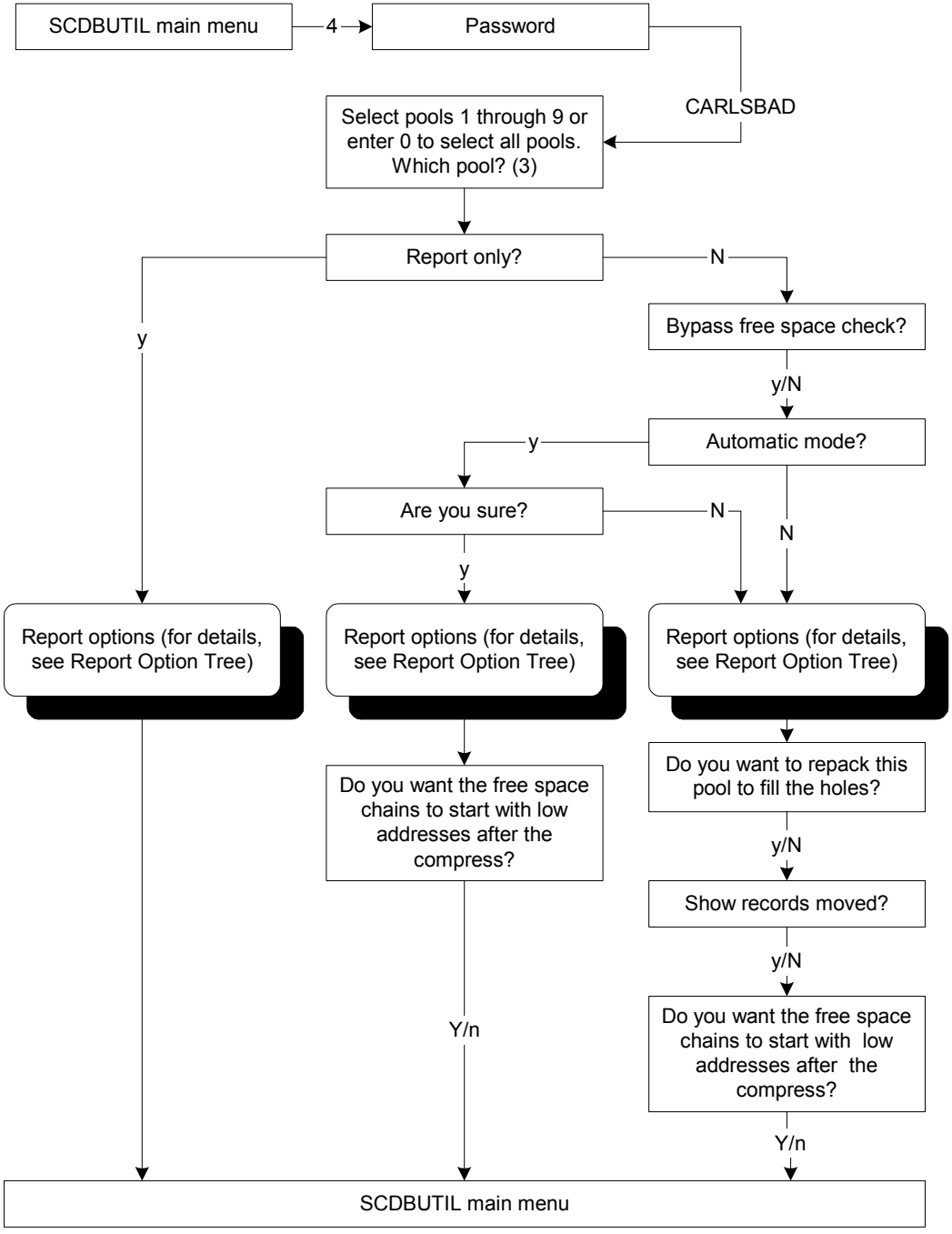
You can use this step to copy the ServiceCenter file, if you add a valid JOB card to the JCL and you named the new, smaller dataset like the original dataset plus an appended '.NEW' qualifier.

The output of the IDCAMS job should look as follows:

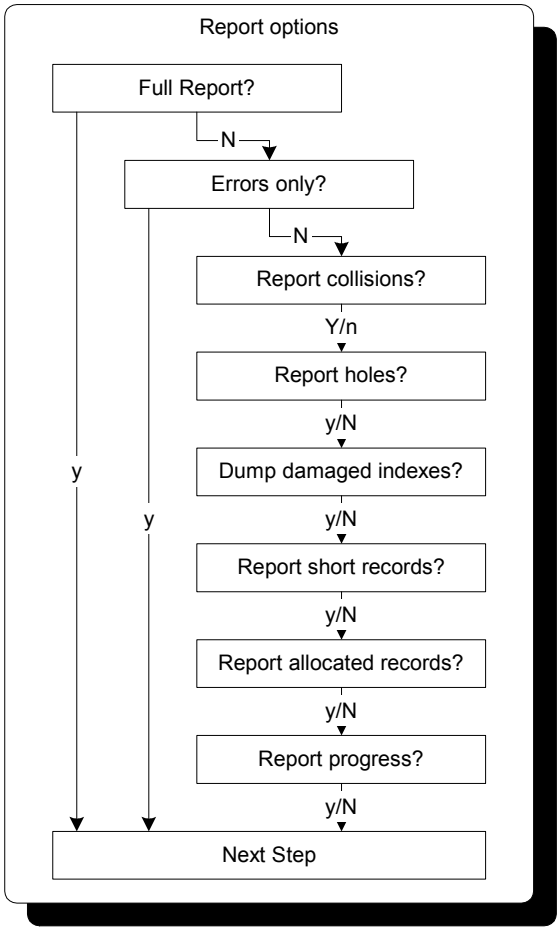
```
IDCAMS SYSTEM SERVICES           TIME: 12:04:23

REPRO INFILE( IN09 ) OUTFILE( OUT09 ) COUNT( 14 )
IDC0005I NUMBER OF RECORDS PROCESSED WAS 14
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Verify that IDCAMS did not indicate any errors, then delete (or rename) the original ServiceCenter file and rename the smaller ServiceCenter file to the original file name. The ServiceCenter file has now been successfully compressed and ServiceCenter is ready for execution.

LFMAP Option Tree



Report Option Tree



Parameters in LFMAP in Compress Mode

In the table below, characters in capital letters mark the default selection.

Parameters	Definition
Enter Password:	This is always **CARLSBAD**.
Select pools 1 through 9 or enter 0 to select all pools. Which pool? (**3**)	Enter the pool number to map, usually **3** to **9**. The number of pools available depends on how many pools are actually defined.

Parameters	Definition
Report only? (Y/n)	y: no compress will be done. n: compress will start.
Bypass Free Space Check? (y/N)	y: LFMAP will run without checking for free lists first. n: LFMAP will check the free lists before running.
The following parameters will only appear if 'Report only' was set to n	
Automatic Mode? (y/N)	y: LFMAP will do all actions (e.g. Repack, fix multiple allocations) automatically. n: LFMAP will ask before taking each action.
Fix files automatically Are you sure? (y/N)	y: LFMAP will do all actions automatically. n: LFMAP will ask before taking each action.
This parameter will always appear	
Full report? (y/N)	y: errors and warnings and informational messages, including allocations, will be reported. n: no informational messages will be reported.
This parameter only comes up if you answered n to 'Full report'	
Report errors only? (y/N)	y: only errors will be reported. n: further options can be chosen for reporting.
The following parameters only appear if 'Report errors only' was set to n	
Report collisions? (Y/n)	y: reports all collisions found. n: doesn't report collision, if found.
Report holes? (y/N)	y: print one line per hole found, stating its address and length. n: no line per hole printed, just a summary of holes.
Dump damaged indexes? (y/N)	y: prints a hex dump of every corrupted index record. n: no dump of index record printed, just error message.
Report short records? (y/N)	y: shows all records that waste space. n: don't show records that waste space.
Report allocated records? (y/N)	y: prints map with ALL records in P4 file system. This will generate a huge output. n: doesn't print map of all records.
Report progress? (y/N)	y: shows detailed progress messages plus file list. n: doesn't print progress messages or files list.
The following two parameters only appear if 'Automatic mode' was set to n	

Parameters	Definition
Do you want to repack pool X to fill the holes? (Y/n)	y: compress will start for this pool. n: compress will not start.
Show records moved? (y/N)	y: LFMAP will show any movements done while compressing n: LFMAP will compress pool without showing movements
This parameter will always appear when compressing	
Do you want the free list to start with low addresses? (Y/n)	y: LFMAP will sort the free lists, so that they point from the lowest address to the highest address n: LFMAP will not sort the free list

Example of an LFMAP Without Compress

```
ServiceCenter Database Maintenance Utility
(Version: 5.0.0 Build: 0027)

1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan and fix P4 file system errors (LFSCAN)
x) EXIT

Enter your choice: 4
LFMAP Utility

Enter Password: CARLSBAD
Select pools 1 through 9 or enter 0 to select all pools.
Which pool? (3) 4
Report only? (Y/n) y
Full report? (y/N) n
Report errors only? (y/N) y
Active options:
Pool:4, Report only:Y, Bypass free space:N, Automatic mode:N
Show collisions:Y, Show holes:N, Dump indexes:N
Show short records:N, Show allocated:N, Show progress:N

Checking pool index file (SCDB.FRE)
Scanning associator file for dbdict records
Checking dbdict records
Checking associator file
Scanning index records ...
```

```
Checking free lists
Checking pool 4 of 6
Starting sort of 8326 records
Highest used address in physical file 4 = 0x100000 (1048576)
282360 bytes unused in 125 bins (54) in physical file 4
--------Index----- -----Data----
D   Recs.    Entr.  Del    Recs.     KBytes    Filename     (DLF# ILF#)
3   46       314    2      314       392       dbdict       (1    2   )
3   82       1892   0      1892      6995      code         (4    3   )
2   14       556    0      278       271       problem      (5    6   )
....
1   1        42     0      42        7         workstation  (1182 1181)
1   1        0      0      0         0         spool        (1191 1192)
2   6        178    1      35        11        schedule     (1194 1193)
323185704 bytes in 95724 variable records
6998528 bytes in 6869 index records for 314 files

Total of 282360 bytes unused in 125 bins (54)

1) Expunge a logical file (LFEXP)
2) File exerciser (DBEXER)
3) Logical file exerciser (LFEXER)
4) Logical file map utility (LFMAP)
5) Physical file exerciser (PFEXER)
6) Logical file scan utility (LFSCAN)
7) Quick Scan with minimal error checking (LFSCAN)
8) Remove Invalid Associators (LFSCAN)
9) Compress of Associator file (LFMAP)
A) Scan and fix P4 file system errors (LFSCAN)
x) EXIT

Enter your choice: x
```

Option 9) Compress of Associator File

SCDBUTIL Option 9), the Compress of Associator file (LFMAP) utility allows you to compress the **scdb.asc** file. It is similar to the LFMAP utility in that it collects all the data necessary, but Associator Compress handles the compressing differently. Before repacking the associator records, the Associator Compress utility runs a consistency check (LFSCAN) against the P4 data and fixes the collisions found. If the utility finds any other errors (associator errors, Database Dictionary errors, or index errors) it stops there.

This utility moves all associators belonging to one logical file and then updates all index records for the same file reflecting the new locations of the associators. In this way, the performance of this utility is optimized.

Warning: There is the risk that if a process is inadvertently terminated, for example, due to a power outage, inconsistencies between the index records and the associators they point to may develop. The data in the logical file that was compressed will be no longer be accessible. It is very important to create a valid backup before compressing the associator file.

The amount of time the utility runs depends on the size of the files, the performance of the computer and the disk, and the memory available. The Associator Compress can run for over an hour on a large database.

The output that is generated by option 9 looks very similar to the LFMAP/LFSCAN output. At the end you will find these messages:

```
Repacking Associator file
clearing pool 1 free lists
Starting to move associators for 'dbdict' file.
Updating index records for file 'dbdict'.
Starting to move associators for 'code' file.
Updating index records for file 'code'.
Starting to move associators for 'syslog' file.
Updating index records for file 'syslog'.
...
Highest used address = 0x13b300 = 1291008
padded to 32768 boundary = 0x140000 = 1310720
old file size = 0x200000 = 2097152
```

The Associator Compress moves the associators, logical file by logical file. Once all associators for one file are moved, the compress reads all index records for that file and updates all indexes to reflect the new associator addresses. After that it proceeds to the next logical file.

How to Run the Associator Compress

Important: Before running an Associator Compress, perform a backup of ServiceCenter.

To run an associator compress:

1 Shut down ServiceCenter if it is currently running. If ServiceCenter is active when LFMAP is called, the scan will not run.

2 Run SCDBUTIL, 9) Logical file map utility (LFMAP) interactively (Unix only), as directed in *How to Run SCDBUTIL in Foreground (Interactively)* on page 50, being sure to capture the output with the script utility.

The following prompts are displayed:

Enter your choice:

a Enter 9 to run the Associator Compress.

Have you created a backup of ALL of your P4 files? (y/N)

b Verify that you have a valid backup of your database files and IR Expert files, then enter y. If you are not sure whether or not you have a backup, enter n and the utility will terminate.

Show records moved?

c Enter n.

The utility now begins the database consistency check. It will fix any collisions it has found and then compress the associator file.

When the compression is completed, the SCDBUTIL menu is displayed again.

d Enter x to exit SCDBUTIL and return to the Command prompt.

— Or —

Run SCDBUTIL, 9) Logical file map utility (LFMAP), in background (0S/390, Unix and Windows), as directed in *How to Run SCDBUTIL in Background (Batch)* on page 55.

Use the following example as the basis for your input file.

```
9    // Run Associator Compress
y    // A backup has been created
n    // Show record moved
x    // Exit SCDBUTIL
```

3 For OS/390 only, run IDCAMS to copy the compressed ServiceCenter file from its current larger size to a smaller dataset. See to *IDCAMS (Used After LFMAP on OS/390 Only)* on page 80 for instructions.

4 Browse the output for error messages. If you see any error messages, call Peregrine Systems Customer Support immediately for assistance with fixing the errors.

Other Options

The other options are not documented in this guide. Do not use them without direction by Peregrine customer support.

- 1) Expunge a logical file (LFEXP) — irrecoverably expunges (deletes) all references for data for a logical file.
- 2) File exerciser (DBEXER) — database maintenance utility.
- 3) Logical file exerciser (LFEXER) — logical file maintenance utility.
- 5) Physical file exerciser (PFEXER) — physical file maintenance utility.

CHAPTER 5

Performance and Tuning Tips

This chapter was designed to aid ServiceCenter database administrators improve the performance of the P4 file system.

Topics in this chapter include:

Improving Query Speed

Most performance problems are noticed during queries. An inefficient query can adversely affect the performance of the whole system. For a query to perform efficiently, the query engine must not be required to search entire files to determine which records match the query.

Some items that affect query speed are:

- *Query Types*, next section
- *Keys* on page 93
- *Fields and Files* on page 96

Query Types

There are five types of queries:

- True
- Fully-keyed
- Partially-keyed
- Non-keyed
- IR

True Queries

A true query is a query which is simply the expression true. The query engine uses the first unique key in the key list to process the query and returns *EVERY* record in the database. If there is no unique key, ServiceCenter will use the first key. Although true queries return results quickly and efficiently, they do not return only the specific records you want to view.

Fully-keyed Queries

A *fully-keyed* query is a query where all of the fields referenced are defined the correct order in a single key that processes the query. A fully-keyed query is very efficient because the query engine determines which records match the query by reading only part of the index tree and none of the data records itself.

Partially-keyed Queries

A *partially-keyed* query is a query where some of the fields referenced are defined in the key that processes the query. The query engine can only partially determine whether a record matches the query using the index. The query engine must read the records that pass the index test before it can determine if they match the query.

In order for a partially-keyed query to be efficient, the first field in the key must be included in the query expression. Otherwise, all index entries in that index must be read in order to determine the query results.

Non-keyed Queries

A *non-keyed* query is very inefficient when used against a large file. The query engine must read all the index entries in an index and all the records to determine which records match the query.

IR Queries

For information on IR Queries, see *IR Expert* on page 319.

Stored Queries

ServiceCenter provides the capability of *storing* a list of popular queries. The user selects which query to execute from the list. The performance of the query should be carefully tuned to ensure that query response time is always fast. For information on stored queries see *Creating a Stored Query* on page 217.

Keys

There are several factors that affect the performance of P4 file system searches. These include key design, key selection algorithms and the number of records. For more information about keys, see *Key Definitions* on page 21, and see *Using Keys in a Search* on page 214.

Designing Keys

Since fully-keyed queries offer the best performance, you could design keys so that every query is fully-keyed. But this is not always practical. To have all queries run fully-keyed can require defining a large number of keys. Too many keys causes performance problems for the add, update, and delete operations.

Instead, you should design keys for the most popular queries. You can also force users to issue fully-keyed queries by not allowing them to issue partially-keyed and non-keyed queries, both of which can be inefficient.

When designing keys, do the following:

- Specify fields that are most commonly used in the query at the beginning of the key.
- Specify fields that have many possible values at the beginning of the key. In other words, a key with a Boolean field at the beginning of the key is not a good key, unless the majority of queries will return only a small number of records. For example, `flag=true` in **probsummary** eliminates 90% of all records.
- Do not use the same field in many keys. If you update a record and change the value of that field, all of the indexes containing that key must be updated.
- Do not define too many keys for one file. The more keys are defined for a file the longer it takes to add, update and delete records in that file. On the other hand if you define too few keys those operations are much faster but the searches on the file might take longer. There is no hard rule of how many keys you should define for a file, however Peregrine recommends defining no more than 25 keys for one file.

Key Selection Algorithm

The key selection algorithm selects a key to perform a query based on the order of the fields in the query expression and the order of the fields in the keys defined in the dbdict. Each key is assigned a weight based on where the fields in the query appear in the key and where they appear in the query and key.

For example, if the first field in a key matches the first field in the query, that key is assigned a higher weight than one which has that field as the second field in the key. This decision is made for all fields in the query and keys.

Example #1

If you had the following query and keys:

```
query:a=1 and b=2 and c=3 and d=4
key1:b,c,d
key2:a,c,d
key3:a,b
key4:a
```

The key selection algorithm would pick key3. Key2 would have the next highest weight, followed by key4 and then key1.

Example #2

Here is an example of the new key selection based on the P4 file location with the keys {location}, {location,state}, {location,city}, {location.name, location}, and {location.code}.

```
query: type=10 location.name#"Pere" and location#"ca"
type=0 key#4 select ed
weightkey#key fields
0.500000   1  type=9 {[12, {"location"}]}
0.500000   2  type=9 {[0, {"location", "state"}]}
0.500000   3  type=9 {[0, {"location", "city"}]}
1.200000   4  type=9 {[0, {"location.name", "location"}]}
0.000000   5  type=9 {[0, {"location.code"}]}
```

For more information on keys (how to add a key, modify a key, delete a key), refer to the *Database Dictionary* section of the *System Tailoring* Guide.

Note: There two special cases: (1) If a sort order is specified and there is a key for this sort order, this key overrides the key selection algorithm. (2) If any part of the query contains an IR query, the IR query is removed, and a key is selected based on the remaining query. All records matching the modified query are selected from the database and matched against the records returned by the IR query for the final result.

Fields and Files

Keep the number of fields in a file low and limit the number of records in a database file.

Number of Fields

By keeping the number of fields in a file low, performance is improved in the following areas:

- Searching for fields in a dbdict.
- Compressing and writing records to disk.
- Decompressing and reading records from disk.
- Operating client/server communications.
- Exercising the Load/Unload option.

Number of Records

Limit the number of records kept in a database file. For example, you would not want to save three years of Incident tickets in a single database file. The more records kept in a file, the larger the index is, which means that all queries run slower.

Database Debug

Database debug can be run by entering the **debugdbquery** parameter in the server's sc.ini file. Use Excel to sort the most time intensive queries.

Entering "**debugdbquery:999**" will show all database access.

Entering "**debugdbquery:n**" will show all queries that exceed n seconds.

Interpreting DEBUGDBQUERY Output

The database debug query output can be found in the **sc.log** file located in the main ServiceCenter directory.

Sample **debugdbquery** output:

```
 223 02/05/2001 17:39:51
DBFIND^F^scmessage(P4)^1^0.000000^F^0^0.000000^"syslanguage="en" and
class="us" and message.id="1""^ ^0.000000^0.000000 ( [ 0]
apm.get.inbox.by.name start )
```

```
 223 02/05/2001 17:39:51
DBQUERY^F^probsummary(P4)^18^1.000000^F^0^0.020000^"hot.tic#true"^
{"category"}^0.000000^0.000000 ( [ 0] sc.manage select )
```

These messages contain several fields, each separated by the ^ character.

Table 5-1: Debugdbquery Output Fields

Field	Description
who	DBFIND or DBQUERY
where	F = foreground or B = background
file	Filename followed by database type (P4 or Oracle/Sybase/DB2/SQL server or LDAP or JOIN), if suffixed by an 'I' case insensitive
key	Number of the key that was selected (If an asterisk follows the key number that indicates that the key was chosen BECAUSE of the sort requirements and not because of the query requirements. The system will always favor using a key that satisfies the sort over the query because then a physical sort of the data is not required).
weight	Calculated weight for the key being selected to be used for that query. See *Key Selection Algorithm* on page 94.
keytype	F=Fully keyed, P=Partially keyed, N=Not keyed, T=True search, I=IR expert search, E=False query
record count	Number of records that have been found that satisfy the query. The DBQUERY entry is put into the log after processing the select panel. The P4 file system will find the first 128 records that satisfies the query before it returns.
seconds till result came back	Amount of time it took to satisfy the query
query	Actual query from the user
sortfields	Sort order in which records were requested
extracttime	Time needed to read data records and extract the key values needed for sorting. This is only necessary if there was no key satisfying the requested sort order

Table 5-1: Debugdbquery Output Fields

Field	Description
sorttime	Time needed to sort all data records matching the query. This is only necessary if there was no key satisfying the requested sort order.
Thread ID	ID of the thread that was active when the query was issued
RAD routine	RAD routine that was active when query was issued
RAD panel	RAD panel that was active when query was issued (Most likely select panel, but could be rio, fdisp, rinit, next, previous, radd, rupdate, rdelete...)

CHAPTER 6

P4 Troubleshooting

This chapter was designed to aid ServiceCenter database administrators solve common ServiceCenter P4 file system problems. It includes the causes of P4 file system corruption.

Warning: Back up your P4 files daily. The ServiceCenter P4 file system does not have a journalizing capability. If a system problem occurs, the problem can cause errors in the database.

Topics in this chapter include:

Corruption Causes

The following events can cause corruption in the database. If any of these events occur, immediately run an LFSCAN to ensure that no corruption exists. The extent of the corruption depends on what the active ServiceCenter processes were doing when the event occurred.

Warning: Do not bring up the system after one of the events below unless LFSCAN shows no errors. If a corrupt system is brought up, more corruption is likely to occur, and data may be lost. In all cases of corruption, contact Peregrine Systems Customer Support immediately.

System or Hardware Crash

If the operating system or hardware crashes, the cached data may not be written to the disk. Since many caching systems do not guarantee the order in which cached blocks are written to disk, it is possible that some blocks are written to disk before other blocks (which were actually updated first).

Use of Operating System commands to shut down the system

- Kill -9 (Unix)
- Cancel (OS/390)
- Task Manager/Processes/End Process (Windows)

These operating system commands cause a process or task to be terminated without allowing the process to finish or even attempt recovery.

System Downtime Causes

The following events can cause system downtime, but should not cause corruption:

- Disk full
- Disk quota exceeded
- Kernel file size limit exceeded

If any of these events occur, ServiceCenter cannot increase the size of a physical file. This causes the current database call to fail and can cause many ServiceCenter applications to fail during database add or update calls.

To solve this problem, shut down ServiceCenter and perform one or more of the following:

- Immediately run LFSCAN to ensure that no corruption exists or to get information on which files need to be cleaned up.
- Delete unneeded data to make more free space.
- After deleting a large amount of data, run the LFMAP utility to pack the data at the beginning of the pool file, effectively shrinking the file and reclaiming space in the database. If you have not freed up some space, there is no point in running LFMAP.
- Increase the disk quota or the kernel file size limit.
- If the disk is full, move the ServiceCenter pool files to another disk or move other files off the current disk.
- Extend the pool (in other words, add a file to the pool).

Warning: Putting ServiceCenter data sets under management of storage products, such as Boole & Babbage's StopX37/II, can have adverse effects or damage the data stored on the files. StopX37/II will attempt to reallocate a ServiceCenter data set which caused the B37. This reallocation can damage the data set to a point where it is unusable by ServiceCenter. If you have any questions, contact Peregrine Systems Customer Support for details.

Extending the ServiceCenter File System Size

ServiceCenter has a limit of two gigabytes per physical file. If your file system needs to increase beyond this limit, the following options are available:

- Extend a pool
- Create a new pool
- Add a pool to a dbdict
- Move a dbdict to another pool

Extending a Pool

Extending a pool offers a quick but short-term solution to working around the ServiceCenter file size limit. It is the solution of choice if a single logical file (for example, *problem*) needs more than two gigabytes.

There is a limit of 38 physical files, regardless of how those files are used by the pools. It does not matter if a pool is extended or a new pool is created, ServiceCenter only supports 38 physical files (including scdb.fre, scdb.asc, and scdb.lfd). There can be 70 GB of data in the 35 files in addition to the scdb.fre, scdb.asc, and scdb.lfd files. Versions earlier than 4.0 have a limit of 10 physical files.

To extend a pool:

1 Run the td.pool RAD application by entering *atd.pool in the ServiceCenter Command line.

Figure 6-1: ServiceCenter Command line

2 Select the **Extend a Pool** option.

3 Select the pool to which you want to add another physical file.

4 For Unix only — Add the following parameter to the server sc.ini file:

max_p4_filesize:65535

Creating a New Pool

Creating a new pool offers a more comprehensive solution to the ServiceCenter file size limit. This solution provides you with increased control over your system's performance and allows you to:

- Separate data and indexes
- Put volatile files on one hard drive, static files on another
- Separate files that are referenced together
- Place spool and schedule files apart from main files

To create a new pool:

1 Run the **td.pool** RAD application by entering ***atd.pool** in the ServiceCenter Command line.

Figure 6-2: ServiceCenter Command line

2 Select the **New Pool** option.

3 Enter the dbdicts you want to add to the new pool. (Optional, can be done later from the Database Dictionary utility.)

What Should Go in a New Pool?

Dbdict tables that Consume the Most Space

Large dbdicts tables are good candidates for putting in their own pool. This option noticeably frees space in your original pool, and allows you to keep better track of when to archive your data.

To determine which dbdicts are exceptionally large:

1 Run LFSCAN as described in *P4 File System Utility* on page 49.

2 At the end of the output generated by LFSCAN you will find a list of files comparable to this one.

```
--------Index-----------                -----Data----
D   Recs. Entr.Del  Ks  I   Recs. KBytes Filename             (DLF#   ILF#)
3   52    494   7   1       494    611   dbdict               (1      2   )
3   83    2458  0   1       2458   9449  code                 (4      3   )
1   1     2     0   1       2      1     printer              (5      6   )
1   1     1     0   3       1      1     port                 (7      8   )
1   1     23    0   2       12     2     pmstatus             (9      10  )
...
1   1     16    0   2   I   2      3     helptext             (139    140 )
...
1   1     1     0   1       1      1     patcortadmin         (1542   1541)
1   1     3     0   1       3      1     patcorttemplate      (1544   1543)
1   1     0     0   1       0      0     applicationrevision  (1546   1545)
```

3 Check the eighth column titled “Data Kbytes.” This column tells you how much space each logical file needs. In the example above, the file **dbdict** needs 611 KB, and the code file needs 9,449 KB.

Files that Change Most Often

Files that are changing should be placed in their own pools. This keeps their constantly changing size from affecting the performance of the other files. Examples: spool, schedule, msglog.

Files that Do Not Change

Put all static files into one pool. These files rarely change and are primarily used as read-only files. Examples: operator, category, model.

What Should Not Go in a New Pool?

Files accessed at the same time, e.g., problem and probsummary, or device and deviceattribute, should be in different pools on different physical hard drives, where possible, to avoid excessive movement of a single head. It's not the size or changeability of the file that matters, it's the size and changeability of the related file, i.e., the actual data records as opposed to the file's structure as defined in the file's dbdict record.

Running ServiceCenter with New Pools

Unix

You can keep your new physical files in the same directory (the Data directory) as your other files. However, to improve performance, or if you are encountering space problems, you can keep your physical files on different hard drives. If you do this, you must set up symbolic links in the ServiceCenter Data directory to reference where you actually keep the physical files. You must also add the allowlinks parameter to your sc.ini file.

OS/390

In your JCL, specify the name of your new physical file along with its data set name (just as with the previous physical files). If you are using the path parameter, you do not have to change your JCL, since the file will be allocated dynamically.

Windows

Files can only be in one directory on one disk.

Adding a Pool to a Dbdict

To add a pool:

1 Run the td.pool RAD application.

2 Select the **Add Pool** option.

3 Enter the dbdicts to which you want to add the selected pool.

The application allows you to add duplicate pools to a dbdict, but internally no duplicates are kept.

Moving a Table to Another Pool

To move a table to another pool, you must first open it using the Database Dictionary Utility.

To open a file using the Database Dictionary Utility:

1 Select the **Toolkit** tab on the ServiceCenter system administrator's main menu. (Figure 9-1 on page 185)

2 Click **Database Dictionary**. The Database Utility screen is displayed.

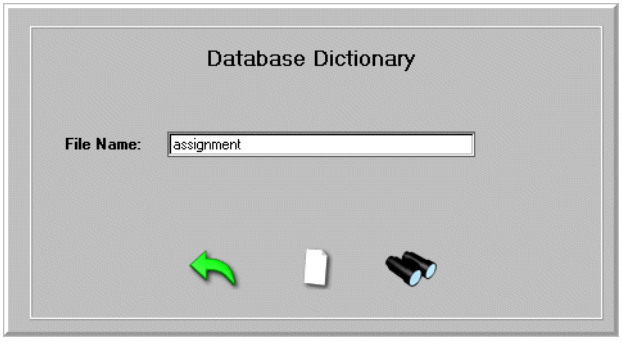


Figure 6-3: Database Dictionary dialog box

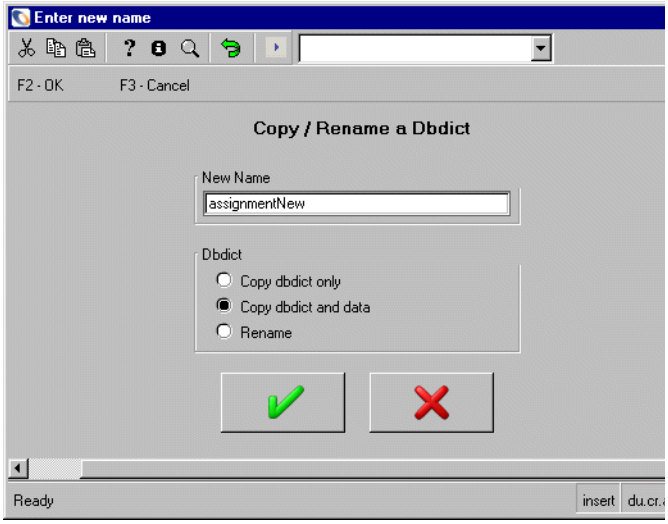
3 Type the database name and click **Search**.

The dbdict definition record(s) with that name is displayed. If more than one filename is displayed, select the correct file to view it.

Once the file is displayed then move the table.

To move a table to another pool:

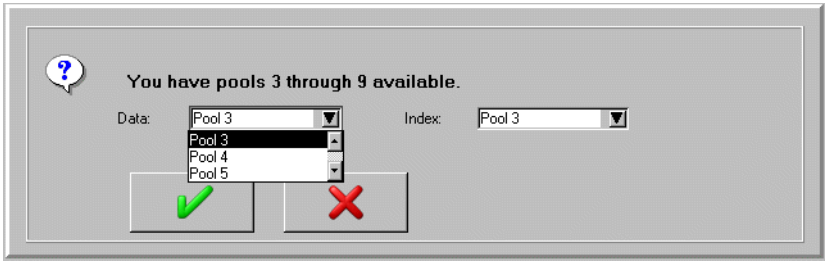
1 Select the **Copy/Rename** from the **Options** menu. The **Copy/Rename a Dbdict** dialog box is displayed.



2 Type the new file name in the New Name text box.

3 Select the **Copy dbdict and data** radio button.

4 Select **Set Data Pools** from the **Options** menu. A dialog box is displayed.



5 Select the data and index pools to which you want the new database written.

6 Click **OK**. You will be returned to the Copy/Rename dialog box.

7 Click **OK** to start copying the database.

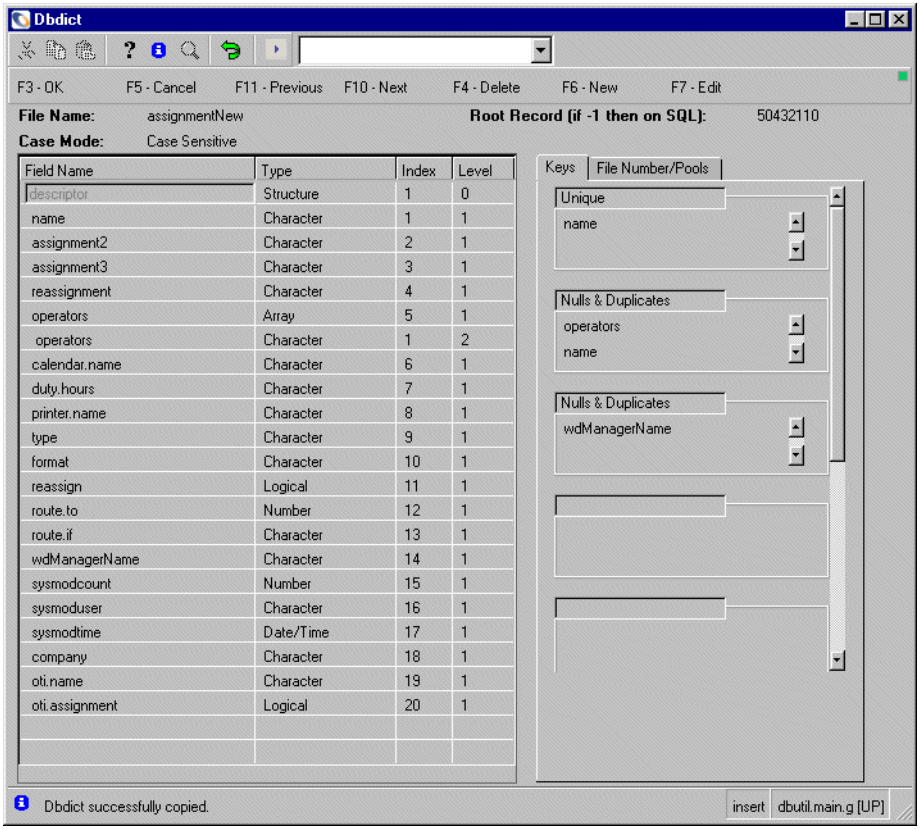
A table will be created with the new filename, and the data from the old table will be copied to it.

You will be returned to the dbdict definition record.

8 Click **OK** to exit.

9 Re-open the file using Database Dictionary.

The new filename is displayed on the list.



10 Click the **View Messages** button to check the messages for errors.

Note: A blue icon indicates a required action, a black icon indicates informational only, and a red icon indicates an error message.

11 If all went well, delete or (preferably) rename the old table.

a Select the old table.

b Select **Copy/Rename** from the **Options** menu.

c Type the desired file name in the New Name text box.

d Select the **Copy dbdict and data** radio button.

e Mark the **Copy/Rename** radio button.

f Click **OK**.

12 Rename the new dbdict to the original dbdict name, with the same steps as 15.

Avoiding Memory Problems During LFSCAN or LFMAP

All of the LFSCAN and LFMAP option in SCDBUTIL (except Quick Scan) need a large amount of memory in order to complete their tasks. This is because all of these options need to track all records stored in your P4 file system: data records, index records, associator records, and free space records. On large databases, the memory needed can exceed 200 MB.

When you run an LFSCAN or LFMAP you will find a message similar to the following in the output:

```
Starting sort of 1357884 records
```

This message gives you the number of all records in your file system. For each of these records, LFSCAN needs 24 bytes of main storage. In the example above, an LFSCAN would need 24 * 1,357,884 = 32,589,216 bytes.

However, if you run out of memory, it will most likely happen before this message is generated. In this case, LFSCAN (or LFMAP) will try to estimate the memory needed and issue messages similar to the following:

```
Memory allocation failure. (status code = 22)
Make sure sufficient swap space is available and retry.
Estimated memory needed for 1686459 records = 45560741 bytes.
Estimated memory needed for 511 databases = 327548 bytes.
Estimated total: 45888289 bytes + shared memory.
```

These messages include a few more data structures which may affect the amount of memory you need. Using this information, adjust your environment so that LFSCAN has enough memory available.

If you still have problems running LFSCAN, use one of the following options:

- Run a Quick Scan instead. Quick Scan does keep track all records; therefore, it does not require as much memory. However, the checks done by Quick Scan are not as complete as a full LFSCAN. You cannot be sure your file system is free of collisions.

- When running LFSCAN, do not check all pools defined in your file system. Instead, select just one pool and run LFSCAN several times with different pools. To do so, type the pool number when prompted, and then press Enter (on Unix) or replace the 0 in the input files by the pool number (on Windows and OS/390).
- Add the parameter lfscan_memory_reclist:NN to your sc.ini file (or your PARMS data set on OS/390) as described in the following section. In this way, the memory needed to complete an LFSCAN can be limited. However, the performance of LFSCAN will be reduced by about 30%. Make sure that there is enough disk space available if you use this parameter.

On Windows, it is possible for ServiceCenter to fail to allocate the shared memory at the default memory address because another DLL is already loaded at the same address, or would overlap with the shared memory. When this happens, the start address of the shared memory can be adjusted to another address until the amount of shared memory wanted can be successfully obtained. Please contact Peregrine Customer Support for detailed instructions.

On Solaris, it is possible that the starting address of the shared memory limits the amount of regular memory LFSCAN can allocate. If LFSCAN or LFMAP fails with at memory allocation failure, and you are certain that the swap space is sufficient, and no user limits were reached, then you can try moving shared memory to a higher address by using the shared_memory_address parameter. Please contact Peregrine Customer Support for detailed instructions.

Allocating Temporary Memory space

When LFSCAN or LFMAP detects the lfscan_memory_reclist:NN in the initialization file (sc.ini or PARMS), or as a Command line parameter, it allocates a temporary file. Instead of holding all record list information in main memory, the information will be written to and be read from this temporary file.

The value (NN) you specify with this parameter determines how many megabytes of main memory will be used to store record list information. If LFSCAN needs more memory, it swaps the oldest record list data out of your main storage into the temporary file and reuses the main storage. If you specify lfscan_memory_reclist without a value, a default of 40 MB is used.

```
# ServiceCenter Initialization file
#
# Copyright (c) 1997-2001 Peregrine Systems, Inc.
# All Rights Reserved
#################################################################################
#
shared_memory:64000000
system:5001

path:E:\SC5
log:..\sc.log

lfscan_detail
lfscan_memory_reclist:120
tmpdirectory:C:\temp\sc
...
```

Figure 6-4: Sample *sc.ini* file

This parameter does not limit the total memory needed to the value you specify. LFSCAN will still need 4 bytes per record additionally for sorting (instead of 24 per record) as well as some more memory for other information including shared memory. If you do not use this parameter, LFSCAN will use virtual memory for an internal record list used to track each record in a ServiceCenter database. It will take about 24 bytes of storage for each entry. This can be excessive for those customers with large file systems and limited virtual memory.

The lfscan_memory_reclist parameter indicates the number of megabytes that should be used to maintain the list in virtual memory. If you do not specify a value, a default of 40 megabytes of virtual storage is assumed. If more than this is required then the data will be written to a temporary file and brought into storage as necessary.

Because the data in main storage is accessible much more quickly than data stored on hard drives, using this parameter will slow down the LFSCAN process. However, it should still be faster than rerunning LFSCAN several times for different pools. This also implies that if you can afford to increase the value specified with this parameter thus increasing the memory available to LFSCAN, LFSCAN will run more efficiently.

Alternatively, call SCDBUTIL from the operating system's Command prompt (or a script which would be basically the same)

```
scdbutil -lfscan_detail -lfscan_memory_reclist:40
```

— Or —

```
scenter -util -lfscan_detail -lfscan_memory_reclist:40
```

Note: If a parameter is entered once in the sc.ini file and is also defined when entering the command at the operating system's Command prompt (here: -path and -lfscan_memory_reclist) the command entered at the prompt overrides the values in the sc.ini.

When you use this parameter, the temporary file will be allocated as defined by your operating system. On Unix systems this will most likely be in the directory /var/tmp. On Windows, the temporary file will be placed in the directory that the environment variable TEMP points to (usually C:\TEMP).

If you do not have enough disk space in the default TEMP location on your system, you can additionally use tmpdirectory:PATH to specify a different location for the temporary file.

In OS/390, you may use the tmpvolser:VOLSER parameter to specify a volume where this data set should be allocated. To use this option, you must also use the tmpdirectory:PATH to specify a location for the temporary file.

Note: For a listing of these and other P4 parameters, see the ServiceCenter *Technical Reference*.

Memory Allocation Failure

When you encounter the memory allocation failure with the LFSCAN, possible fixes are:

- Set the shared_memory_address to a value of
  1073741824
  — or —
  0x40000000
- Decrease shared_memory to 8000000

This helps only if you are short of actual physical memory. To run the Compress of the Associator File option, you will need much more shared memory because this option updates IR Expert files and IR Expert needs lots of shared memory.

Warning: Do not change the shared_memory parameter in your sc.ini file. If you forget to change it back to the original value, there will not be enough shared memory to run ServiceCenter safely.

CHAPTER 7

The ServiceCenter ODBC Driver

ServiceCenter provides an ODBC driver that allows users to generate reports directly from data in the P4 database. The driver is installed automatically with the ServiceCenter installation, but can also be installed separately.

This chapter provides instructions for installing the ODBC driver individually to upgrade a current ODBC installation. It outlines the procedures for creating a data source and for changing data source locations. It includes FAQ and troubleshooting sections for the ODBC driver.

This chapter has been divided into the following topics:

Introduction

In order to allow interactions between outside applications and the P4 data, an appropriate ODBC driver must be installed on each system that will be accessing the P4 data. Typically, this will be the ServiceCenter ODBC driver, although in cases of shadowed data, an ODBC driver native to the shadowed platform is also acceptable.

If you are using the ServiceCenter P4 database, an ODBC driver will automatically be installed when you install ServiceCenter. You can also install it separately from the ServiceCenter CD. For more information, see *Installing the ServiceCenter ODBC Driver* on page 117.

If you are using any other database, refer to that vendor's documentation for instructions for installing the appropriate ODBC driver.

About the ServiceCenter ODBC Driver

The ServiceCenter ODBC driver, **scodbc32.dll** , located in **\winnt\system32** or the equivalent directory for your Windows operating system, is compatible with Win32 (Windows NT, Windows 98, or Windows 2000) operating systems only.

The ODBC driver is essential to the retrieval of ServiceCenter data for reporting purposes. It is designed to be used exclusively with an ODBC compliant application, and will not work with other querying utilities.

These two components are required:

- Microsoft Windows ODBC Administrator, which is installed with your operating system. For information on how to upgrade to a new version of the Windows ODBC Administrator, go to the Microsoft Web site.
- ServiceCenter ODBC driver, which is installed with ServiceCenter. You can also install it separately from the ServiceCenter CD. For more information, see *Installing the ServiceCenter ODBC Driver* on page 117.

Installation and Configuration

If you have installed ServiceCenter, the ODBC driver has already been installed and configured on the workstation. Only run the separate installation described here if you wish to upgrade the ODBC driver, but not ServiceCenter.

Note: Any type of application can be used with the ServiceCenter ODBC driver. The ODBC driver connects directly to the p4Layer no matter which application is used.

Supported Platforms and Operating Systems

Client

The ServiceCenter client must be installed on a WIN32 (Windows NT, 95, 98, or 2000, but not Win 3.1) machine, with ODBC and TCP/IP installed. It does not need to be on the same machine as the ServiceCenter server it connects to.

Note: The ServiceCenter ODBC driver will not run in a Windows session on OS/2, since OS/2 is a 16-bit application, and the ServiceCenter ODBC driver is a 32 bit application.

ServiceCenter Server

The ServiceCenter server can be on any supported platform (OS/390, Unix or Windows) as long as it can be reached from the client port over TCP/IP or APPC.

The server's binaries must be version 2.0 or later. The table below shows which driver to use, depending on the ServiceCenter binary version installed.

ServiceCenter ODBC Driver Version	Compatible ServiceCenter Binaries	Known Issues / Limitations	ODBC Compliance Level
3.00.0	SC3.0.0 SC3.0.1 SC3.0.1A SC3.0.1B SC3.0.1C	Works only with Crystal Reports 7.0 and earlier and ReportCenter 3.0.	Level 2
	SC3.02 SC3.0.2A SC3.0.2B SC3.0.2C SC3.0.2D	No impromptu support.	
	SC3.0.3 SC3.0.4A SC3.0.4B SC3.0.5 SC3.0.6 SC3.0.7 SC3.0.8	Basic query processing.	
4.0.10	All SC3 Versions SC4.0.0 SC4.0.1 SC4.0.2 SC4.0.3 SC4.0.4	Works with Crystal Reports 8.5 and earlier and ReportCenter 4.0.	Level 2
	SC4.0.5 SC4.0.6 SC4.0.8 SC4.0.9 SC4.0.10.1 SC4.0.10.2 SC4.0.10.3 SC4.0.10.4	Impromptu support.	

ServiceCenter ODBC Driver Version	Compatible ServiceCenter Binaries	Known Issues / Limitations	ODBC Compliance Level
4.01.00/5.00.00 *(Same Driver)*	All SC3 Versions All SC4.0 Versions SC4.01.00 SC4.01.01 SC5.00.00 SC5.00.01	Specifying a date range in the Where Clause will return No Records.	Level 3
		Works with any ODBC compliant application.	
		Works with Crystal Reports 8.5 and earlier and ReportCenter 5.	
4.01.01/5.00.01 *(Same Driver)*	All SC3 Versions All SC4 Versions SC4.01.00 SC4.01.01 SC5.00.00 SC5.00.01	Works with any ODBC compliant application.	Level 3
		Works with Crystal Reports 8.5 and earlier and ReportCenter 5.	

Installing the ServiceCenter ODBC Driver

The correct Microsoft ODBC environment (ODBC Administrator) must be installed and set up before the ServiceCenter ODBC Driver is installed.

To check the Microsoft ODBC driver:

1 From the Windows **Start** menu, select **Settings** > **Control Panel**, and double-click the Data Sources (ODBC) icon.

Note: If the icon does not appear, download the ODBC Administrator from the Microsoft Web site.

— or —

From the Windows **Start** menu, select **Programs** > **ODBC** > **ODBC Administrator**.

2 Go to the **Drivers** tab of the ODBC Administrator and look at the version of **ODBC32.DLL**. If it is not version 3 or higher, upgrade it. To upgrade to a new version of the Windows ODBC Administrator, go to the Microsoft Web site.

To upgrade the ServiceCenter ODBC driver:

1 Put the ServiceCenter Installation CD in the CD-ROM drive. The Installation will auto-start.

2 Select the **Install/Upgrade ODBC** button from the installation menu. A welcome message is displayed.

3 Click **Continue** to complete the installation.

The ODBC administrator window is displayed, allowing you to configure your data source. Go to step 3 on page 119 for instructions.

Creating a Data Source

A data source (DSN) stores information that indicates how to connect to a specific database. You can create several data sources pointing toward different databases that all use the same driver. Generally, you will want to create a data source that is visible to all users who need to generate reports from a database. If database security is a consideration, create a data source that is visible only to the user who created it.

To create a data source:

1 On Windows NT:
Click the windows **Start** button, and select **Settings**> **Control Panel**.

On Windows 2000:
Click the windows **Start** button, and select **Settings** > **Control Panel** > **Administrative Tools**.

2 Double-click the Data Sources (ODBC) icon.

The ODBC Data Source Administrator dialog box is displayed.

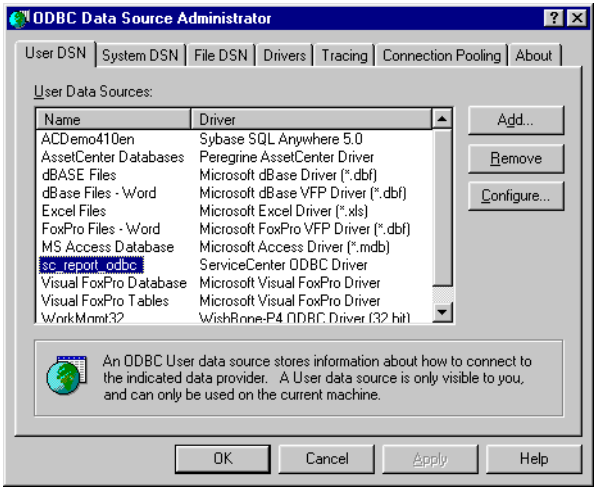


Figure 7-1: The ODBC Data Source Administrator dialog box

3 Select the **User DSN** tab to create a user DSN.

Note: Create a user DSN to safeguard sensitive data. This data source is only visible to the user who created it. In other words, if you create a DSN here and someone else logs on to your workstation, that person will not see it.

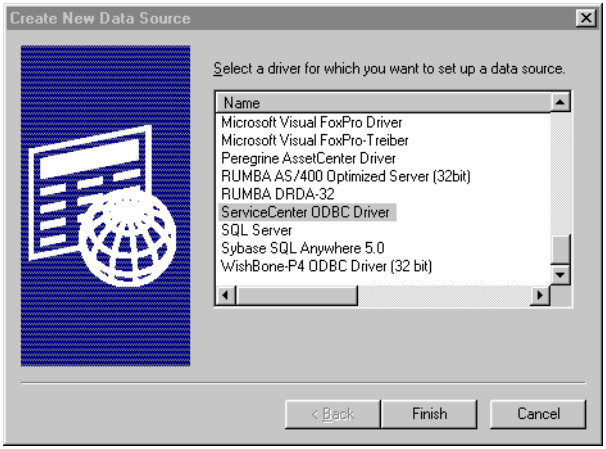
— OR —

Select the **System DSN** tab to create a system DSN.

Note: Create a system DSN to make your data source visible to any user who logs on to the machine on which the DSN was created.

4 Click **Add** to add either type of DSN.

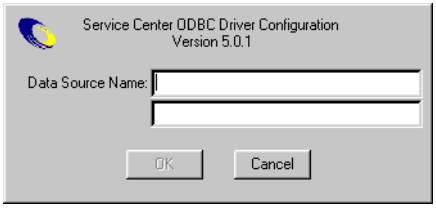
The Create New Data Source dialog box is displayed.



5 From the list of drivers, select **ServiceCenter ODBC Driver**. If your server is mapped to an external database (such as SQL or Oracle), you may use an ODBC driver native to that system.

6 Click **Finish**.

The ODBC Data Source Setup dialog box is displayed.



7 Enter an appropriate data source name in the **Data Source Name** box. The system default is **sc_report_odbc**.

8 In the lower box:

- If your installation of ServiceCenter uses TCP/IP, enter your server and port number separated by a period for example, `jsmith.12670`).
- If you are installing using APPC, enter `SCCPIC14`, and add a line containing "`transport:CPIC`" to the `sc.ini` file.

9 Click **OK**.

Changing the Data Source Location

To change the location of a data source:

1 On the taskbar, click **Start**, point to **Settings**, then click **Control Panel**.

2 Double-click the **Data Sources** (ODBC) icon.

The ODBC Data Source Administrator dialog box is displayed. See Figure 7-1 on page 119

3 In the **User Data Sources** list, double-click on **sc_report_odbc** to display the Data Source Setup dialog box.

4 Change the **Server.Port** information to point toward the new location.

5 Click **OK**.

Configuring the ServiceCenter ODBC driver to work with Windows Applications

The ServiceCenter 5.0 ODBC Driver uses the validateodbcfieldnames parameter with all Windows applications. This parameter configures ServiceCenter to replace all periods in column names with underscores.

To configure the ServiceCenter ODBC driver to work with Microsoft Access and Excel:

1 Locate the ServiceCenter sc.ini file on the ODBC client computer.

Note: There may be more than one sc.ini file. You must edit the file in the working directory for the application you are executing. If the file does not already exist, you must add it in this working directory. For Crystal Reports, for example, edit or add the sc.ini file in Crystal's run directory. For Microsoft Access, edit or add the sc.ini file in the directory that contains the msaccess.exe executable file.

2 Edit the sc.ini file to add this new parameter to the file, on a separate line with no leading spaces:

```
validateodbcfieldnames
```

3 Save your changes.

When importing data into Excel from a new database:

1 Uncheck the “use the query wizard to create/edit queries” checkbox.

2 Once MsQuery opens with the Add Tables Dialog, select the **Options** button. This will cause the table options dialog box to open.

3 Select **Refresh** and close the dialog box.

All Service Center tables will now be displayed correctly in the Add Tables Dialog.

Security Options

ServiceCenter supports multiple security features, each effective in different ways. This document briefly describes some major security options and the security support they provide when accessing the ServiceCenter database through the ServiceCenter P4 ODBC driver.

Securepassword

The securepassword parameter prevents displaying the operator file's password field through both the ServiceCenter client and through ODBC connections. Activate this parameter by adding the line securepassword to your ServiceCenter server's sc.ini file and restarting the ServiceCenter service. If securepassword is not active and no other security is in place, any user with appropriate software and a full client connection can display every username and password in the operator table.

This feature is available in later versions of ServiceCenter 2.1 and all versions of 3.0. Peregrine strongly recommends activating it in any security-conscious environment.

Password randomization and redirection

This category includes third-party utilities, mostly on OS/390 (MVS), which substitute each user's password in the operator table with a programmatically generated password stored in another, more secure location. Commonly used utilities include Top Secret, RACF, and ACF2.

The password can also be hidden if the operator file is mapped to an LDAP server. In this case, LDAP provides authentication and no actual passwords are stored in ServiceCenter database.

Mandanten

Mandanten intercepts data at the database layer, providing filters prior to access by either the ServiceCenter client or ODBC. It has proven effective in shielding data without ill effects. Consult the System Administration for details of configuring Mandanten.

ServiceCenter ODBC Driver FAQ

This section contains a list of questions about the ServiceCenter ODBC driver.

How will the ServiceCenter ODBC driver affect licensing? Does the ODBC connection use a user slot?

Yes. For instance, if you have a ServiceCenter system on Oracle, both ServiceCenter and Oracle will see a user connected when an ODBC connection is established.

The following logic is used to access a user slot:

- If there are active user licenses available, the ODBC driver will use one of these first.
- If no active user licenses are available, the driver will check for named users.
- If the name does not match a named user or the name is already in use, the driver will check for floating licenses and use a floating slot.
- If no floating licenses are left, the login will fail.

Is the ServiceCenter ODBC driver backward compatible with ReportCenter 1.x? Do I still need my data dictionaries?

Yes. ReportCenter will run either type of report (1.0 or later).

The ODBC compliant application report files (*.rpt) shipped with ReportCenter 1.0 and earlier versions use data dictionaries instead of an ODBC driver. The data source type of a report file is embedded in the report and is not changed when you install the new version, therefore you will need to retain the old dictionary files for them to work.

All reports now shipped with ReportCenter are ODBC driver-based. Refer to the *Report Center* guide for other considerations when upgrading ReportCenter.

Some fields/structures are missing when I list the fields of a database in an ODBC compliant application. Why?

The fields in question are in an array. Applications do not always recognize array structures in a table, so the ServiceCenter ODBC driver maps arrays to another table with the same name as the parent table plus the extension "a1" if it is the first array structure, "a2" for the second, and so forth. For example, the contacts table is called "contactsm1," the first array in that table would be mapped to "contactsa1." You can then join the array to the main table by means of a subreport.

Note: Not all arrays will be mapped to tables. Some will remain as fields in the original table.

The criteria for when an array will be mapped externally are as follows:

- The ServiceCenter ODBC driver recognizes only the following data types:
  - character
  - number
  - logical
  - date/time
- An array of characters is displayed as an unlimited string field (memo-type) and not as a separate table, unless it is an odbcchar array.
- An array of array of characters is displayed as a table with just one character field.
- An array of numeric/datetime or logical elements is displayed as a table.
- An array with only one element whose data type is not one of the above recognized data types does not appear as a separate table.
- An array of structures where the first element of the structure is one of the above data types does not appear as a separate table.

Can we do functions like COUNT, AVERAGE, MIN, or MAX in our ODBC driver queries?

The ODBC driver does not support aggregate functions. Some applications, such as Crystal Reports, send the ODBC driver these types of queries by removing the aggregate function and handling it themselves.

For a complete list of supported functions, see *SQL Keywords* on page 132.

Does the ServiceCenter ODBC driver handle foreign languages?

Yes, the ServiceCenter ODBC driver handles all of the languages that ServiceCenter currently provides. In order to use this feature, you need to add the language:<foreign language> parameter to your ODBC sc.ini file.

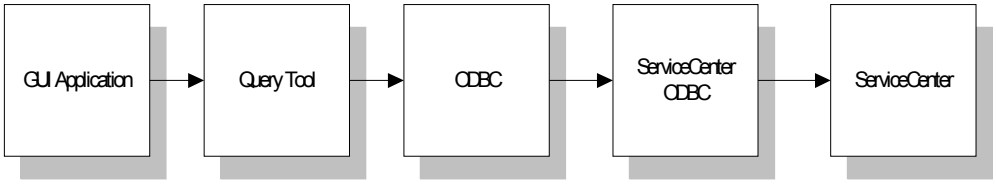
Where should I place the ODBC driver sc.ini file?

Place the sc.ini file in the same location as the sc.log file that the ODBC driver creates during execution. (See *ODBC Driver and SQL Log Files* on page 129 for more information.) If an sc.ini file already exists in that location, then you will only need to edit it.

When connecting via Microsoft products, the sc.ini file is usually placed in the /My Documents directory. When using Crystal Reports it is usually in the /Program Files/Seagate/Crystal Reports directory and when using ReportCenter it is in the /ServiceCenter/RPTCTR directory.

Troubleshooting

To begin troubleshooting, you need to determine that the ODBC driver is the source of your trouble. Several layers are involved and each has its own unique troubleshooting methodology. The overall path of data between ServiceCenter and your reporting tools is shown below.



As a user, you are working with some type of query application. These components translate your commands into calls to the run-time environment (RTE), a set of .dll files and support files on your workstation that do the actual handling of application files. When activated, the application run-time environment makes SQL calls to the Microsoft ODBC environment, which in turn calls the ODBC driver of the database you are using (typically the ServiceCenter ODBC driver). Finally, the ODBC driver calls the database.

Errors at the GUI end come in the form of configuration errors and program bugs. Connection errors and SQL errors happen at the ODBC level. Bad data returned or long lag times may originate at the ServiceCenter end.

Common Problems and Solutions

Here are some possible causes and solutions to the most common errors encountered, as well as some generic troubleshooting steps:

Error message "Cannot Find SQL Server" or "Error Connecting to <Servername>"

These are generic connection failure messages. This means that the RTE attempted to log in to ServiceCenter and received either a failure or no reply.

This can be caused by any of the following:

- The ServiceCenter server is not running.

  Contact your ServiceCenter administrator and make sure the server is up, then attempt to connect using the ServiceCenter Client.
- Your login information is incorrect.

On a failure, you will be prompted to re-issue your login information. Check the information in the fields for accuracy. If the login information specified is incorrect, you may want to go into your program options and set it to the correct login for future use. Also check capitalization. ServiceCenter logins are case-sensitive.

- ServiceCenter has used up its allotted licenses.

  ServiceCenter only accepts a set number of concurrent users, depending on your license. If all available slots are in use, your login is rejected. Contact your ServiceCenter administrator to check this. If your ServiceCenter server is licensed for casual users, we recommend logging in using the casual user account, since the limitations of a casual user do not impact the application, and your logins will not count against the overall total slots.

- ODBC driver error.

  Check the version of your ODBC driver versus the recommended version for your ServiceCenter binaries. If the version is incorrect, it will GPF after login. If the version is correct, but not the most current, you may be experiencing a bug that has been corrected in the current revision.

Error message "Driver Not Capable"

This means that the report you are trying to run is attempting to execute a SQL query that the ODBC driver doesn't support (bad syntax, complex or inappropriate join, use of an unsupported SQL keyword).

To look at the raw SQL being used by the report:

1. Activate SQL tracing in the ODBC Administrator.
2. Run the query in your application.
3. Look for SQL in the ODBC log. For instructions on how to activate SQL tracing and review the log file, see *ODBC Driver and SQL Log Files* on page 129.

If the report uses a ServiceCenter join, compare the join syntax to those accepted by ServiceCenter. See *Joining Multiple Tables* on page 303. Otherwise, search the SQL for unsupported keywords (usually capitalized). See *Unsupported Keywords* on page 132 for a list of unsupported keywords.

If you are still experiencing the same problem after checking this, save your raw SQL or ODBC log and have it ready when you contact technical support.

Error message "Corresponding Join Not Defined"

This indicates the report is attempting to join multiple tables that have not been properly joined in the ServiceCenter database.

When you design reports, linking two tables using the application is NOT sufficient to create a join. ServiceCenter must also be configured equivalently.

Look at the fields that are joined your application. Make sure only the fields you want to join are connected by arrows, and that the arrows point from the main table to the secondary table.

Next, open ServiceCenter (or contact your ServiceCenter administrator) and compare the settings in the JOINDEF and ERDDEF tables to this linking scheme. See *Joining Multiple Tables* on page 303. Make sure that:

- The joins match exactly.
- The join in ServiceCenter is unique (in other words, no other join connects these same tables in a different manner).
- The ServiceCenter server was brought down and restarted after the join was created.

Note: Modifying the JOINDEF and ERDDEF tables in ServiceCenter should be done with caution and only with the help of a ServiceCenter administrator. To be safe, we recommend making an unload of both tables before making modifications, so that you can return to the original configuration if needed.

Queries taking excessively long to execute

There are many things that can affect execution speed. Hardware and network limitations are always a factor. But much of the speed issue is related to report design.

Here are some quick tips on how to speed up the execution of your reports:

- Limit your sorting layers. If your raw SQL has more than two ORDER BY statements, you are greatly slowing the processing time. Each ORDER BY statement exponentially increases execution time.

- Use joins instead of subreports. Subreports are more universal, making for greater portability over different servers, but a subreport executes its SQL query against the database once for every record. Therefore, a report with 100 records with a subreport in the Details section will execute 101 SQL statements. A joined report of equivalent size executes only one.
- Use indexes in ServiceCenter where possible. Typically, the unique key of any table will already be indexed. However, in a join, placing a key on the field being used to join the two tables will help dramatically.

Note: Adding too many indexes to a ServiceCenter table can be detrimental to overall ServiceCenter performance. Adding of indexes should be done only with the help of your ServiceCenter administrator.

- Keep your ODBC driver up to date. New releases often include speed enhancements.

Application Errors when logging in

Your ServiceCenter ODBC Driver is out of sync with your ServiceCenter binaries. See the ODBC Driver FAQ section for the compatibility matrix.

Driver errors when attempting to use with Microsoft Access or Excel

Add the validateodbcfieldnames parameter to your ODBC sc.ini file. When working with Microsoft applications this file is usually located in the /My Documents directory.

ODBC Driver and SQL Log Files

ServiceCenter creates an ODBC Driver log file, sc.log. This log file contains the commands executed by the ODBC driver. The sc.log file often contain important error messages and should be consulted when odd behavior arises.

You can cause ServiceCenter to generate an SQL log file, sql.log.

Enabling SQL Logging

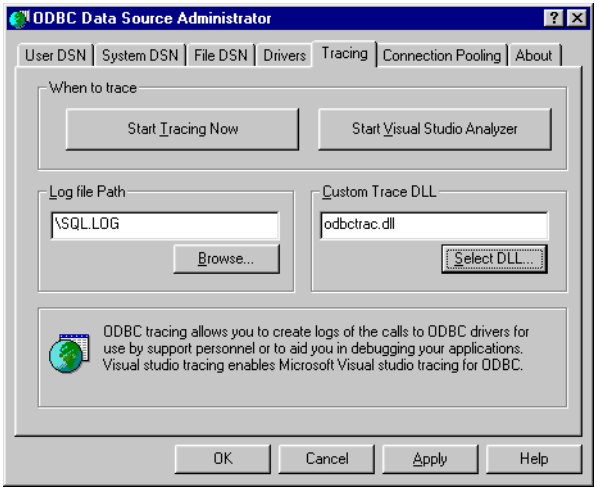
To enable SQL logging:

1 From the Windows **Start** menu, select one of the following (depending on your operating system).

- **Settings > Control Panel**
- **Settings > Control Panel > Administrative Tools**
- **Programs > ODBC > ODBC Administrator**

2 Double-click the **Data Sources (ODBC)** icon.

3 Select the Tracing tab.



4 Either change the name of the output file or rename the existing **sql.log**. This will keep the new logging information separate from existing logging data.

5 Click **Start Tracing Now**.

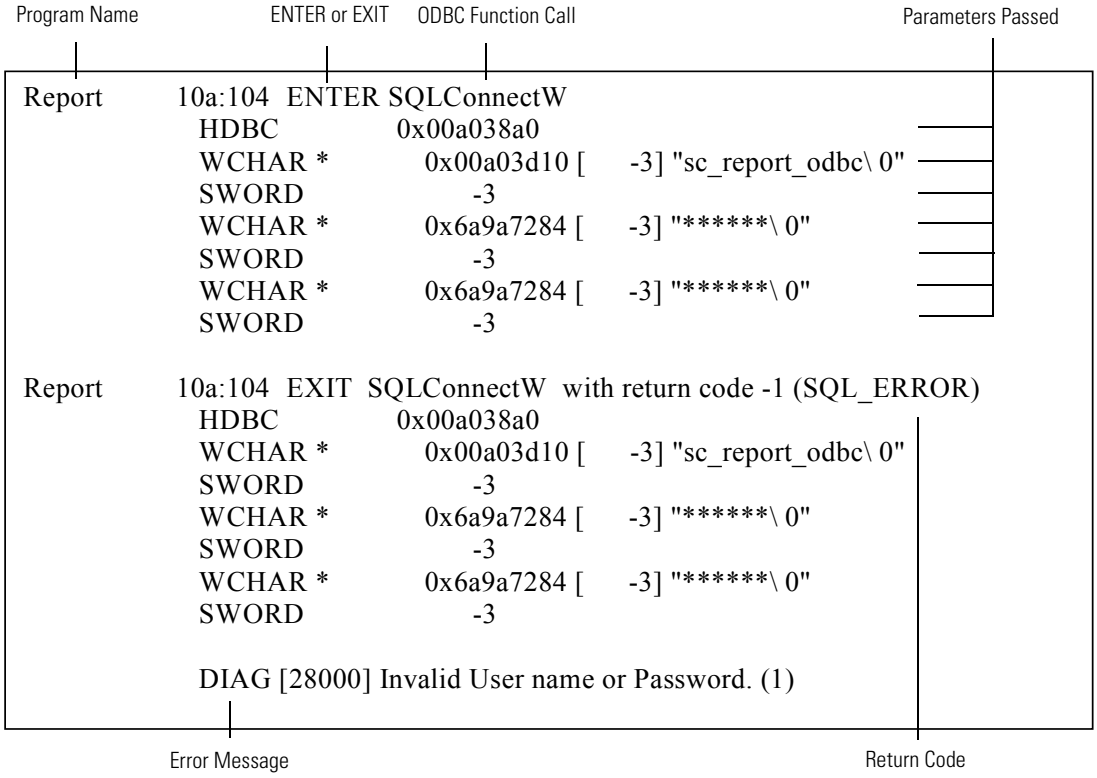
6 Click **Apply**.

Note: You have to click **Apply**. Clicking **OK** without first clicking **Apply** will not activate the tracing feature.

The log file will be created in your root directory.

Reading the SQL Log

The sample file shown below is an example of an SQL log. All items are arranged in ENTER / EXIT pairs. ENTER contains information about what the program passed into the ODBC driver. EXIT contains the return code (whether it was successful or not) and the text of the ODBC error message. Since SQL logs can be very long, do a search for SQL_ERROR to locate the error messages.



When Contacting Customer Support

Before contacting Customer Support for an ODBC Driver issue, collect the following information:

- The version of ServiceCenter binaries you are using (from the About box on the ServiceCenter Client).
- The type and version of the ODBC driver you are using (from the Drivers tab of the ODBC Administrator).

- The version and build numbers of your application (from the About box in your application's main Window).

If this is a design issue, include:

- The SQL statement and selection criteria of the report.

If you are sending Customer Support information by e-mail, please attach:

- The report in question (if not a canned report).
- Any SQL tracing logs you may have generated.

An unload of your JOINDEF and ERDDEF tables (if this issue involves a join report).

SQL Keywords

Supported Keywords

The ServiceCenter ODBC driver supports these keywords:

Table 7-1: Supported Keywords

AND	ASC	BETWEEN x AND y
DESC	FROM	IN
IS IN	LIKE	NOT
NOT LIKE	OR	ORDER BY
SELECT	STARTSWITH	WHERE

There are numerous SQL dialects, and numerous functions that require read-write ability, which are NOT supported by our driver. Unsupported keywords either are ignored by the database or may cause errors, and should be avoided.

Table 7-2: Unsupported Keywords

ABSOLUTE	ACTION	ADA
ADD	ALL	ALLOCATE
ALTER	ANY	ARE
AS	ASSERTION	AT

Table 7-2: Unsupported Keywords

AUTHORIZATION	AVG	BEGIN
BIT	BIT_LENGTH	BOTH
CASCADE	CASCADED	CASE
CAST	CATALOG	CHAR
CHAR_LENGTH	CHARACTER	CHARACTER_LENGTH
CHECK	CLOSE	COALESCE
COBOL	COLLATE	COLLATION
COLUMN	COMMIT	CONNECT
CONNECTION	CONSTRAINT	CONSTRAINTS
CONTINUE	CONVERT	CORRESPONDING
COUNT	CREATE	CROSS
CURRENT	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURRENT_USER	CURSOR
DATE	DAY	DEALLOCATE
DEC	DECIMAL	DECLARE
DEFAULT	DEFERRABLE	DEFERRED
DELETE	DESCRIBE	DESCRIPTOR
DIAGNOSTICS	DISCONNECT	DISTINCT
DOMAIN	DOUBLE	DROP
ELSE	END	END-EXEC
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FOR	FOREIGN	FORTRAN
FOUND	FULL	GET
GLOBAL	GO	GOTO
GRANT	GROUP	HAVING
HOUR	IDENTITY	IMMEDIATE

Table 7-2: Unsupported Keywords

INCLUDE	INDEX	INDICATOR
INITIALLY	INNER	INPUT
INSENSITIVE	INSERT	INTEGER
INTERSECT	INTERVAL	INTO
IS	ISOLATION	JOIN
KEY	LANGUAGE	LAST
LEADING	LEFT	LEVEL
LOCAL	LOWER	
MAX	MIN	MINUTE
MODULE	MONTH	MUMPS
NAMES	NATIONAL	NATURAL
NCHAR	NEXT	NO
NONE	NULL	NULLIF
NUMERIC	OCTET_LENGTH	OF
ON	ONLY	OPEN
OPTION	OUTER	OUTPUT
OVERLAPS	PAD	PARTIAL
PASCAL	PLI	POSITION
PRECISION	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVILEGES
PROCEDURE	PUBLIC	REFERENCES
RELATIVE	RESTRICT	REVOKE
RIGHT	ROLLBACK	ROWS
SCHEMA	SCROLL	SECOND
SECTION	SEQUENCE	SESSION
SESSION_USER	SET	SIZE
SMALLINT	SOME	SPACE
SQL	SQLCA	SQLCODE
SQLERROR	SQLSTATE	SQLWARNING

Table 7-2: Unsupported Keywords

SUBSTRING	SUM	SYSTEM_USER
TABLE	TEMPORARY	THEN
TIME	TIMESTAMP	TIMEZONE_HOUR
TIMEZONE_MINUTE	TO	TRAILING
TRANSACTION	TRANSLATE	TRANSLATION
TRIM	TRUE	UNION
UNIQUE	UNKNOWN	UPDATE
UPPER	USAGE	USER
USING	VALUE	VALUES
VARCHAR	VARYING	VIEW
WHEN	WHENEVER	WITH
WORK	YEAR	

ODBC Driver Functions

Supported Functions

The ServiceCenter ODBC Driver supports these functions:

Function	Description
ConfigDlgProc	Executes Dialog for Data Source Name
ConfigDSN	Adds, modifies, or deletes data sources from the ODBC.INI file.
SQLAllocConnect	Allocates memory for a connection handle within the environment identified by henv.
SQLAllocEnv	Allocates memory for an environment handle and initializes the ODBC call level interface for use by an application.
SQLAllocHandle	Allocates and environment, connection, statement or descriptor.
SQLAllocStmt	Allocates memory for a statement handle and associates the statement handle with the connection specified by hdbc.

Function	Description
SQLBindCol	Assigns the storage and data type for a column in a result set.
SQLBindParameter	Binds a buffer to a parameter marker in an SQL statement.
SQLCancel	Cancels the processing on an hstmt.
SQLCloseCursor	Closes a cursor that has been opened on a statement and discards pending results.
SQLColAttributes	Returns descriptor information for a column in a result set.
SQLColumns	Returns the list of column names in specified tables.
SQLConnect	Loads a driver and establishes a connection to a data source.
SQLCopyDesc	Copies descriptor information from one descriptor handle to another.
SQLDescribeCol	Returns the results descriptor-column name, type, precision, scale and nullability-for one column in the result set.
SQLDescribeParam	Returns the description of a parameter marker associated with a prepared SQL statement.
SQLDisconnect	Closes the connection associated with a specific connection handle.
SQLDriverConnect	Is an alternative to SQLConnect. It supports data sources that require more connection information than the arguments in SQLConnect.
SQLError	Returns error or status information.
SQLExecDirect	Executes a prepared statement using the current values of the parameter marker variables if any exist in the statement. Fastest way to execute a statement for one time execution.
SQLExecute	Executes a prepared statement, using the current values of the parameter marker variables if any exist in the statement.
SQLFetch	Fetches a row of data from a result set.
SQLFetchScroll	Fetches the specified rowset of data from the result set and returns data for all bound columns.
SQLFreeConnect	Releases a connection handle and frees all memory associated with it.

Function	Description
SQLFreeEnv	Frees the environment handle and releases all memory associated with the environment handle.
SQLFreeHandle	Frees resources associated with a specific environment, connection, statement, or descriptor handle.
SQLFreeStmt	Stops processing associated with a specific hstmt, closes any open cursors associated with the hstmt, discards pending results, and optionally, frees all resources associated with the statement handle.
SQLGetConnectAttr	Returns the current setting of a connection attribute.
SQLGetConnectOption	Returns the current setting of a connection option.
SQLGetCursorName	Returns the cursor name associated with a specified hstmt.
SQLGetData	Returns result data for a single unbound column in the current row.
SQLGetDiagField	Returns the current value of a field of a record of a diagnostic data structure (associated with a specified handle) that contains an error, warning, or status information.
SQLGetDiagRec	Returns the current values of multiple fields of a diagnostic record that contains error, warning, and status information.
SQLGetInfo	Returns general information about the driver and data source associated with an hdbc.
SQLGetStmtAttr	Returns the current setting of a statement attribute.
SQLGetStmtOption	Returns the current setting of a statement option.
SQLGetTypeInfo	Returns information about data types supported by the data source.
SQLMoreResults	Determines whether there are more results available on an hstmt containing SELECT, UPDATE, INSERT, or DELETE statements and, if so, initializes processing for those results.
SQLNumParams	Returns the number of parameters in an SQL statement.
SQLNumResultCols	Returns the number of columns in a result set.
SQLParamData	Used in conjunction with SQLPutData to supply parameter data at statement execution time.
SQLPrepare	Prepares and SQL string for execution.

Function	Description
SQLPutData	Allows an application to send data for a parameter or column to the driver at statement execution time.
SQLRowCount	Returns the number of rows affected by an UPDATE, INSERT, or DELETE statement or by a SQL_UPDATE, SQL_ADD, or SQL_DELETE operation in SQLSetPos.
SQLSetConnectAttr	Sets attributes that govern aspects of connections.
SQLSetConnectOption	Sets options that govern aspects of connections.
SQLSetCursorName	Associates a cursor name with an active hstmt.
SQLSetStmtAttr	Sets attributes related to a statement.
SQLSetStmtOption	Sets options related to an hstmt.
SQLSpecialColumns	Retrieves the following information about columns within a specified table: - The optimal set of columns that uniquely identifies a row in the table. - Columns that are automatically updated when any value in the row is updated by a transaction.
SQLStatistics	Retrieves a list of statistics about a single table and the indexes associated with the table.
SQLTables	Returns the list of table names stored in a specific data source.

Aggregate Functions

The ODBC driver does not support aggregate functions. Some applications, such as Crystal Reports, send the ODBC driver these types of queries by removing the aggregate function and handling it themselves.

Unsupported Functions

The ODBC driver does not support these functions. They should not be used.

Function	Description
SQLBrowseConnect	Supports an iterative method of discovering and enumerating the attributes and attribute values required to connect to a data source.
SQLBulkOperations	Performs bulk insertions and bulk bookmark operations, including update, delete, and fetch by bookmark.

Function	Description
SQLCloseCursor	Closes a cursor that has been opened on a statement, and discards pending results.
SQLColumnPrivileges	Returns a list of columns and associated privileges for the specified table.
SQLDataSources	Returns information about a data source.
SQLDrivers	Lists driver descriptions and driver attribute keywords.
SQLEndTran	Requests a commit or rollback operation for all active operations on all statements associated with a connection.
SQLExtendedFetch	Extends the functionality of SQLFetch by returning rowset data, scrolling through result set according to setting of a scroll-type argument.
SQLForeignKeys	Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table.
SQLGetDescField	Returns the current setting or value of a single field of a descriptor record.
SQLGetDescRec	Returns the current settings/ values of multiple fields of a descriptor record.
SQLGetEnvAttr	Returns the current setting of an environment attribute.
SQLGetFunctions	Returns information about whether or not a driver supports a specific ODBC function.
SQLNativeSql	Returns the SQL string as translated by the driver.
SQLParamOptions	Allows an application to specify multiple values for the set of parameters assigned by SQLBindParameter.
SQLPrimaryKeys	Returns the column names that comprise the primary key for a table.
SQLProcedureColumns	Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures.
SQLProcedures	Returns the list of procedure names stored in a specific data source.
SQLSetDescField	Sets the value of a single field of a descriptor record.
SQLSetDescRec	Sets multiple descriptor fields that affect the data type and buffer bound to a column or parameter data.
SQLSetEnvAttr	Sets attributes that govern aspects of environments.

Function	Description
SQLSetPos	Sets the cursor position in a rowset and allows an application to refresh, update, delete, or add data to the rowset.
SQLSetScrollOptions	Sets options that control the behavior of cursors associated with an hstmt.
SQLTablePrivileges	Returns a list of tables and the privileges associated with each table.
SQLTransact	Requests a commit or rollback operation for all active operations on all hstmts associated with a connection.

SECTION II

Data Retrieval

This section was designed to provide ServiceCenter system and database administrators with information on how to use it to retrieve, edit, and maintain database records. Additional reference information can be found in the *Database Dictionary* section of the *System Tailoring* guide.

Chapters in this section include:

- *Federated Database Support* on page 143 — this chapter explains how to save space, and conflict resolution, as well as the backup and synchronization problems by eliminating replication of data.
- *The Database Manager Utility* on page 183 — this chapter gives an introduction to the Database Manager Utility.
- *Record Retrieval* on page 187 — this chapter discusses query based record retrieval.
- *Single Record Functions* on page 237 — this chapter explains how to perform add, update, delete, print functions on individual records within a database.
- *Multiple-Record Functions* on page 247 — this chapter explains how to perform add, update, delete, print functions on multiple records within a database.
- *Database Record Auditing* on page 275 — this chapter explains how to check specified fields within a file in the ServiceCenter database for modifications, when records in that file are updated.

- *Joining Multiple Tables* on page 303 — this chapter explains how to combine multiple tables in a single form using the Database Manager Utility.
- *File Maintenance* on page 311 — this chapter explains how to maintain database files, including resetting a database and regenerating database keys.
- *IR Expert* on page 319 — this chapter explains the concepts and components of IR Expert and Knowledge Engineering and provides IR Expert system-level configuration information.
- *Using Joined Queries* on page 339 — this chapter explains how to write queries that will return data from two different ServiceCenter files.

CHAPTER 8

Federated Database Support

A federated database is a logical database comprised of data from more than one physical database. The discussion in this chapter assumes that people, assets, and locations exist in AssetCenter and it is desirable to share that data with ServiceCenter. However, AssetCenter is just an example; any Relational Database Management System (RDBMS) could be used.

This chapter was designed to help ServiceCenter system and database administrators to save space, avoid the need for conflict resolution, and eliminate the backup and synchronization problems that go along with the replication of data.

Note: This chapter assumes the reader is familiar with ServiceCenter and AssetCenter.

Topics in this chapter include:

Introduction

Federated data is data which resides in, or is owned by another entity. For example, an LDAP directory may own employee or contact data, or AssetCenter may contain asset and location data. The goal of *federated data* is to eliminate replication of data. Using federated data saves space, avoids the need for conflict resolution, and eliminates the backup and synchronization problems that go along with the replication of data.

With ServiceCenter federated databases, data from multiple databases are combined to create a single record. The number of records that can be viewed does not change, but the data within each record is acquired from more than one physical source.

For example, the vital information about an employee might be maintained in one physical database and contains information such as names, dependents, start date, and benefit options, etc. However, additional sensitive data such as current salary is maintained in a separate database. Typically two systems would be required to view that data, since the data is physically separate. However, with a ServiceCenter federated database, those who have access to the sensitive data are able to see all the information at once. A ServiceCenter federated database brings all the information together from the various sources and presents the data to the requestor, as if the data were in a single physical database.

The ServiceCenter federated solution allows you to share data with an external entity. A ServiceCenter record is comprised of data from multiple physical sources but any particular data element (field) will exist in one and only one location. The sharing is done in a non-invasive fashion. It does not require code or schema changes to the external application, and it respects the external system's data policy layer if possible. There are no direct calls against the RDBMS. All calls for external data go through the usual channels for the external application.

ServiceCenter federated database support is platform and database independent. It works with ServiceCenter on OS/390, Unix, or Windows platforms, using any database or other external entity, on OS/390, Unix, Windows platforms.

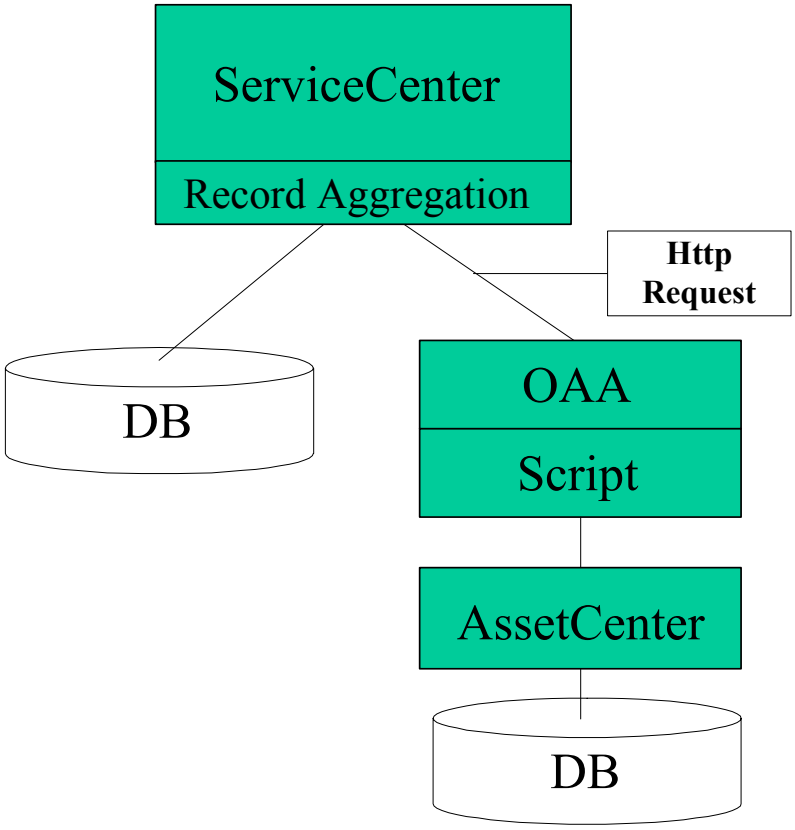
Architecture

From a ServiceCenter point of view, a federated database is the combination of data from the ServiceCenter database with data from an external database. For example, the combination of information about location is maintained in ServiceCenter with the data maintained about locations from AssetCenter. Each individual element of data maintained about a location will only exist within a single source. Some of the elements might be maintained in ServiceCenter and some in AssetCenter. The decision on which elements are maintained in what physical source is discussed later in this chapter.

Throughout this chapter, AssetCenter is used as the external database. However, the federated database was designed to be generic, enabling you to federate ServiceCenter data with another database maintained in your organization.

Federated data is implemented using Open Application Architecture (OAA) technology and the scripting capability within the OAA architecture. Since OAA is accessed using HTTP, the external database can exist anywhere within the organization, and the customer can use the scripting language supported by OAA to provide the interfaces needed by ServiceCenter to extract data from their databases.

The diagram below illustrates the relationship between ServiceCenter, OAA, the script, and an external database. In this case AssetCenter is shown, but a script can be developed to access another external database.



Flow of Data

There are two possible flows of data for federated support, depending on which database (ServiceCenter or the external database) is the primary source of information.

Both flows are described below:

- *When ServiceCenter is the Primary Source* on page 147.
- *When the External Database is the Primary Source* on page 147.

The *primary source* is defined as the source that is accessed *first*, and the source to which all queries are directed.

When ServiceCenter is the Primary Source

With ServiceCenter as the primary source of data, the ServiceCenter database is used as the target of a query. For each record returned from the ServiceCenter database, a query is done against the external database to get the corresponding data, which is merged into the current record. A record will not be returned to the requestor unless that record exists in the ServiceCenter database. The record does not even have to exist in the external database.

When the External Database is the Primary Source

With the external database as the primary source of data, the external database is used as the target of a query. For each record returned from the external database, a query is done against the ServiceCenter database to get the corresponding data, which is merged into the current record. A record will not be returned to the requestor unless that record exists in the external database. The record does not even have to exist in the ServiceCenter database in order for data to be returned. The *SYSDEFAULTS record that exists within the ServiceCenter database will be used to provide default values for any fields that are empty after the merge has taken place. For information about the *SYSDEFAULTS record, see the System Administrator's Guide.

Location of Data within a Federated Database

If we are going to allow data to exist in multiple physical locations, then the decision has to be made as to where each datum should reside. Here are some things to consider in making that decision.

ServiceCenter is the only system that supports federated data. Therefore if some other product expects the data to be in a certain physical database, then that is where the data must be. For example, all the data that AssetCenter needs to do its job must exist within the AssetCenter database. The AssetCenter data cannot exist in the ServiceCenter database, because AssetCenter has no way of getting to the ServiceCenter data.

The data should reside in the physical database of the product that is most likely to update the data. Try to ensure that a single product is responsible for updating the data. If more than one system can update the data then you have to worry about dual updates. The ServiceCenter federated support checks to make sure that it is updating the most current data by checking the date the

record was last modified (if a **dtLastModif** field exists within the external database). The other concern with updates from ServiceCenter to an external database is that the external database or the connection to OAA may not be available at the time of the update. ServiceCenter will queue the update until all components are available, so further updates from ServiceCenter will not be allowed while an entry is queued, but the external source does not know that this has occurred.

The product that is responsible for inserts to the database should be the product that is the primary source for data. (For a definition of the primary source of data, see *Flow of Data* on page 146.) The other systems do not support federated databases. Therefore, new records that have been added by the other product are not automatically updated in ServiceCenter. For example, if a new asset is created in AssetCenter, the same record does not exist inServiceCenter. If ServiceCenter was defined as the primary source then this asset would not be accessible. However, if AssetCenter was the primary source then the asset would show up.

Configuring Federated Databases

The following software documentation may be necessary for the configuration process:

- *ServiceCenter Client/Server Install Guide* for OS/390, Unix, or Windows
- *AssetCenter Installation and Upgrade Guide*
- *Get.It! Quick Start Guide* (if you are installing AssetCenter 3.51or 3.6)
- *Get.Services! Quick Start Guide* (if you are installing AssetCenter 4.1)

Configuration

Use the following steps to configure ServiceCenter and AssetCenter federated databases.

Two examples are given:

- *Configuring ServiceCenter 5.0 and AssetCenter 4.1 Federated Databases*, next section.
- *Configuring ServiceCenter 5.0 and AssetCenter 3.51/3.6 Federated Databases* on page 155.

Configuring ServiceCenter 5.0 and AssetCenter 4.1 Federated Databases

The configuration steps in this section operate under the assumption that you have installed, or are in the process of installing OAA 2.2 with Tomcat 3.2.4. The procedure may vary slightly if you have chosen a different Application server.

To set up ServiceCenter and AssetCenter federated databases:

1 Install ServiceCenter, following the directions in the ServiceCenter *Client/Server Installation Guide* for OS/390, Unix, or Windows (if it is not installed already).

   a Install both ServiceCenter Server and Client.

   b Take note of the server and port number of the server.

2 Install AssetCenter, following the directions in the *AssetCenter Installation and Upgrade Guide* (if it is not installed already).

3 Install OAA for ServiceCenter (if it is not installed already.)

   a Install Java2 SDK.

   b Install Tomcat 3.2.4.

   c Install Peregrine OAA.

4 Configure OAA by doing the following:

   a Create a directory titled ServiceCenter5 in the \jakarta-tomcat-3.2.4\webapps\oaa\WEB-INF\lib directory. There should be other folders with similar titles (ServiceCenter3 and ServiceCenter4) already there.

   b Copy the file sccl32.dll from the ServiceCenter\RUN directory to the ServiceCenter5 directory created in step a.

   c Copy the file scj.dll from the \jakarta-tomcat-3.2.4\webapps\oaa\WEB-INF\lib\ServiceCenter4 directory to the ServiceCenter5 directory created in step a.

5 Copy the file extacoaa.js from the RUN directory where ServiceCenter was installed to the \jakarta-tomcat-3.2.4\webapps\oaa\WEB-INF\apps\common\jscript directory.

6 Add the parameter extoaa on its own line in the ServiceCenter Init file(sc.ini), which is found in the ServiceCenter\RUN directory. The Init file can also be opened from the Windows **Start** menu.

7 In the file archway.xml located at \jakarta-tomkat-3.2.4\webapps\oaa\WEB-INF\default\, set the httpbasicauth tag to from true to false.

```
<httpbasicauth label="$$IDS(common,configHttpbasicauthLabel)" type="boolean"
instruction="$$IDS(common,configHttpbasicauthInstr)">true</httpbasicauth>
```

Warning: Since this flag must be false, consider having a version of OAA installed on the same side as the firewall as ServiceCenter, to be used only for ServiceCenter Federated support. This will also aid performance.

8 Add scenter -que:federated on its own line in the sc.cfg file, which is located in the \ServiceCenter\RUN directory. The config file can also be opened from the Windows **Start** menu.

9 Load the ServiceCenter unload files located in the \jakarta-tomcat-3.2.4\oaa\WEB-INF\etc folder.

- portal.unl
- qman.unl
- state.unl

For instructions on how to load the files, see the Importing records in the Foreground section of the ServiceCenter *System Administrator's Guide*.

10 Restart the ServiceCenter server.

11 Start the Webserver and the Application server. To start Tomcat, run the file startup.bat located in the \jakarta-tomcat-3.2.4\bin directory.

12 Start the OAA Administration console. From the Windows **Start** menu, select **Programs** > **PeregrineOAA** > **OAA Administration**.

13 Log in as an administrator. (By default, the Administrator login is Admin.)

The Peregrine Portal Control Panel opens.

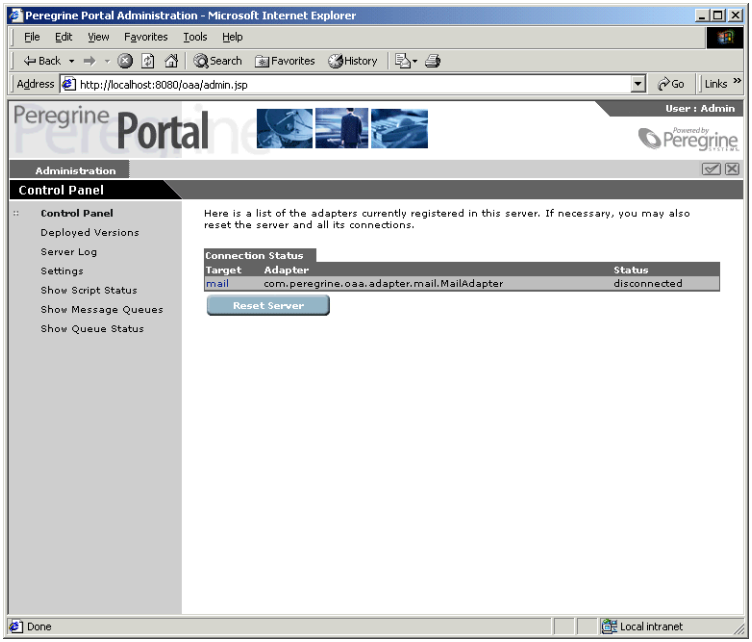


Figure 8-1: Peregrine Portal Control Panel

14 Configure the adapters.

a Select settings from the menu on the left.

b In the List of Target Aliases, located on the Portal tab, specify ac and sc as semicolon delimited entries (ac;sc).

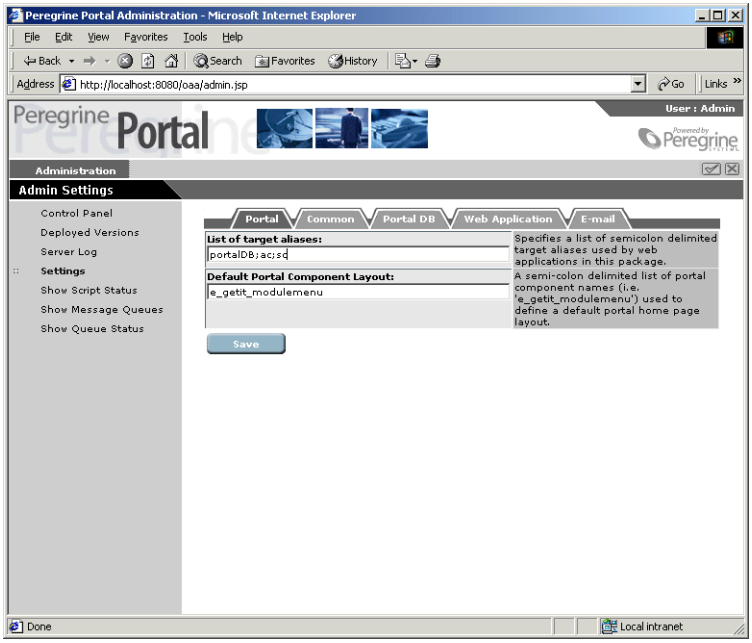


Figure 8-2: Peregrine Portal Admin Settings

c Click Save. You will be returned to the Control Panel.

d Click Reset Server. The Connection Status List should now include the SC and AC connectors.

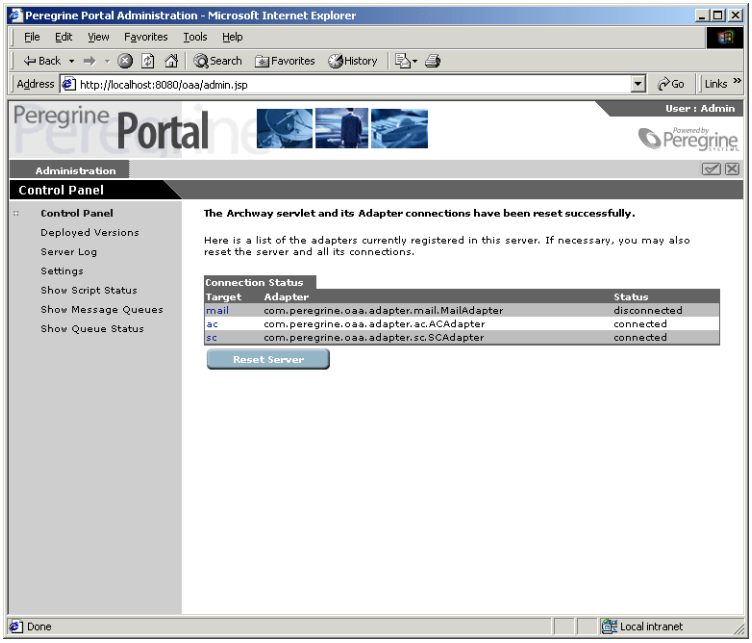


Figure 8-3: Peregrine Portal Connection Status list

15 Connect the AssetCenter database.

a On the Control Panel, click **ac** in the Connection Status list.

The Settings page opens to the **AssetCenter** tab.

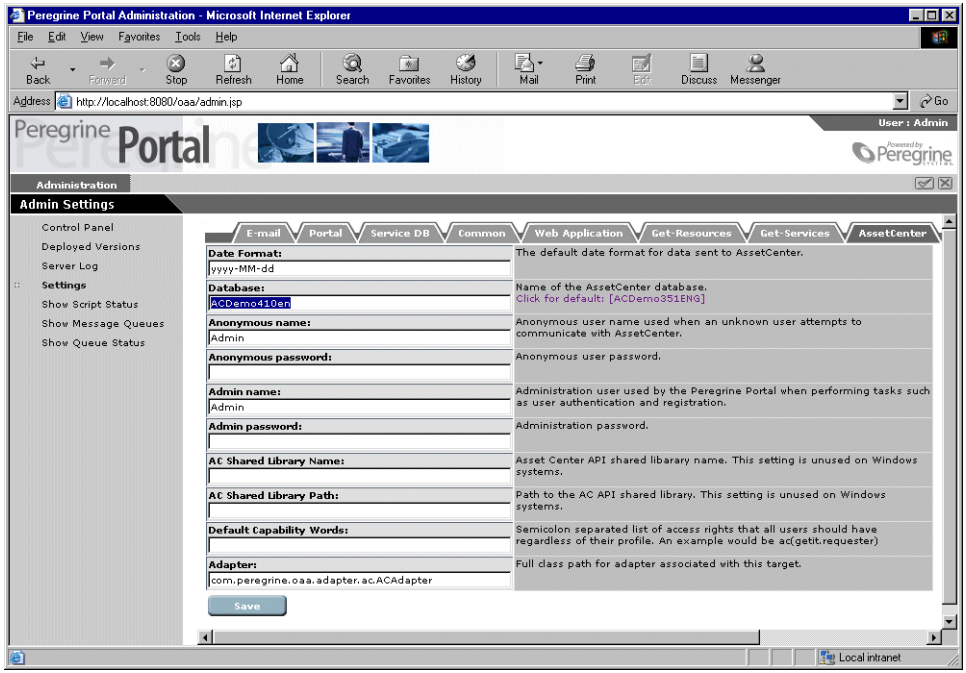


Figure 8-4: AssetCenter Settings tab

b On the **AssetCenter** tab, in the **Database** textbox, type the name of the AssetCenter database you want to connect to. (`ACDemo410en`, in this example).

c Set the User Id and Password under **Anonymous Name**.

Access to AssetCenter from ServiceCenter is done with the AssetCenter User Id specified under **Anonymous Name**.

d Click **Save**.

You are returned to the control panel.

e Click **Reset Server**.

The **ac** Status changes from disconnected to connected.

When ServiceCenter and AssetCenter both show a Status of connected, the setup is complete. Proceed to *The ServiceCenter Mapping* on page 158.

Configuring ServiceCenter 5.0 and AssetCenter 3.51/3.6 Federated Databases

The configuration steps in this section operate under the assumption that you have installed, or are in the process of installing Get.It! 2.0.1 with JRUN 3.1 and IIS 4.0 or 5.0. The procedure may vary slightly if you have chosen different App or Web Servers.

To set up ServiceCenter and AssetCenter federated databases:

1 Install ServiceCenter, following the directions in the ServiceCenter *Client/Server Installation Guide for OS/390*, Unix, or Windows (if it is not installed already).

   a Install both ServiceCenter Server and Client.

   b Take note of the server and port number of the server.

2 Install AssetCenter, following the directions in the *AssetCenter Installation and Upgrade Guide* (if it is not installed already).

   Note: For AssetCenter 3.51, the default port is 80, which is the default Web server port, and will need to be changed. We suggest you use 8080.

3 Start AssetCenter and load the AssetCenter authorization code, when prompted.

4 Install Get.It!, following the directions in the *Get.It! Quick Start Guide* (if it is not installed already).

   a Restart the IIS Admin Service after step 4 of the connection Wizard.

   b After Install configures IIS, you are prompted to install weblications. There are no required weblications.

5 Load the ServiceCenter unload files located in the `\getit\packages\common\common\config` folder.

   - portal.unl
   - qman.unl
   - state.unl

   For instructions on how to load the files, see the Importing records in the Foreground section of the ServiceCenter *System Administrator's Guide*.

6 Configure Get.It! by doing the following:

   a Create a directory titled ServiceCenter5 in the \getit\bin directory. There should be other folders with similar titles already there.

   b Copy the file sccl32.dll from the RUN directory where ServiceCenter was installed to the directory created in the previous step.

   c Copy the file scj.dll from the \getit\bin\ServiceCenter4 directory to the directory created in step a.

7 Copy the file extac.js from the RUN directory where ServiceCenter was installed to the \getit\apps\common\jscript directory.

8 If desired, add the parameter extvalidate on its own line in the ServiceCenter Init file (sc.ini), which is found in the ServiceCenter\RUN directory. The Init file can also be opened from the Windows **Start** menu.

   If this parameter is in the sc.ini file, then the user name and password of the ServiceCenter user will be passed through Get.It! and validated by the external data source (AssetCenter).

   - If the user name or password is not valid in AssetCenter, then the user is denied access, and the query will fail.
   - If the user name and password are valid in AssetCenter, the user is allowed access with whatever permissions have been set up for that user in AssetCenter.

   If the parameter is not included in the sc.ini file, then all queries are passed with the permissions of the Anonymous user, Admin, by default.

9 Type scenter -que:federated on its own line in the sc.cfg file, which is located in the \ServiceCenter\RUN directory.

10 Restart the ServiceCenter server.

11 Start the Peregrine Get.It! Administration jsp.

   To open the Peregrine Get.It! Administration jsp, open the Windows **Start** menu and select **Programs** > **Peregrine Get.It!** > **Administration**.

12 Log in as an Administrator. The Get.It! Control Panel will open.

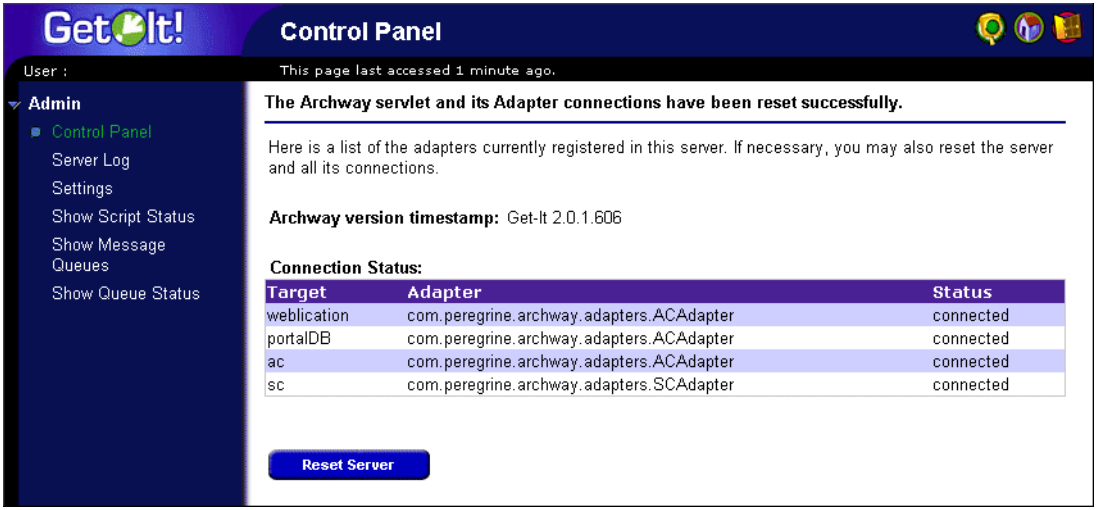


Figure 8-5: Get.It Control Panel

13 Select Settings in the navigation pane.

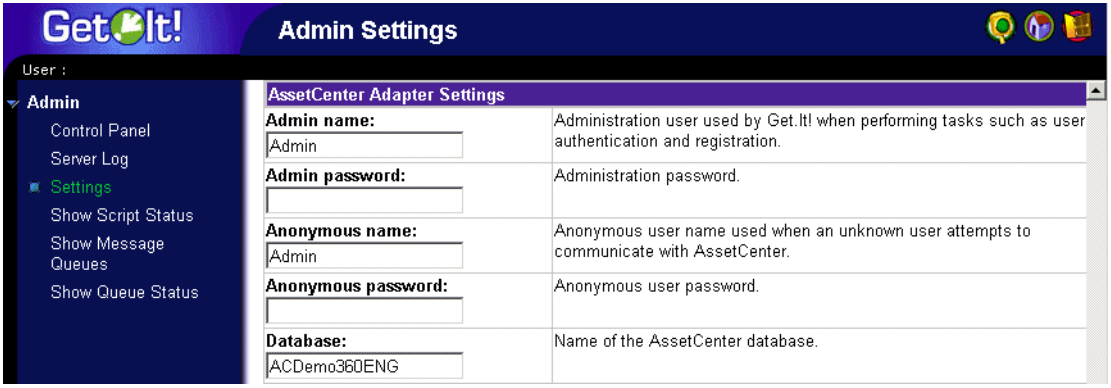


Figure 8-6: Get.It AssetCenter Adapter Settings

14 Scroll to the AssetCenter Adapter Settings section.

a Specify the name of the AssetCenter Database in the **Database** textbox.

b If you added the parameter **extvalidate** to the ServiceCenter Init file, set the User Id and Password under **Anonymous Name**. For more information on **extvalidate**, see step 8 on page 156.

15 Click **Save**.

You are returned to the Control Panel.

16 Click **Reset Server**.

The **ac Status** changes from disconnected to connected.

When ServiceCenter and AssetCenter both show a Status of connected, the setup is complete. Proceed to *The ServiceCenter Mapping* on page 158.

The ServiceCenter Mapping

The mapping describes the relation between fields in ServiceCenter and fields in an external data store. Mapping specification is performed manually. The same technique is used for all types of data stores, including LDAP and AssetCenter.

Data deleted from ServiceCenter will be deleted from the target database system. However, data deleted from the target database system will not be automatically deleted from ServiceCenter.

Important: The Administrator(s) doing the mapping must be familiar enough with both ServiceCenter and the target database system to know which files and fields in ServiceCenter map to which files and fields in the target database system.

Only a partial mapping is necessary. Fields not mapped to an external database reside in the ServiceCenter database. The *SYSDEFAULTS record in the Service-Center database provides default values for external records that do not have a ServiceCenter counterpart. Fields mapped to an external database are always obtained from the external database even when ServiceCenter is the primary source for record sets. Valid/Used data only exists in one database; there is no replication.

Mapping Flow

Step 1 The mapping is read from the ServiceCenter database (datadict table).

Step 2 The OAA URL creates an HTTP connection.

Step 3 The query is built using the fields and tables that have been specified in the mapping.

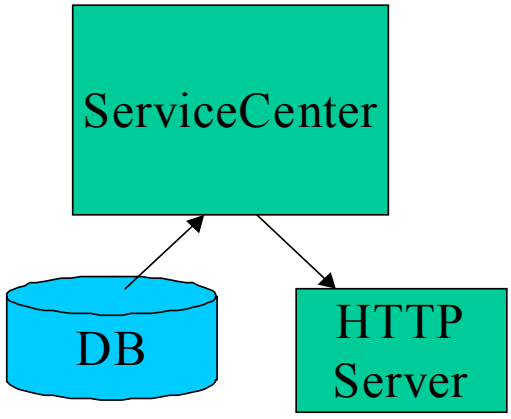
Step 4 OAA executes the script specified in the mapping.

Step 5 The script constructs a query and uses adapter direct query to AssetCenter.

Step 6 Records returned by the adapter are merged with data from ServiceCenter.

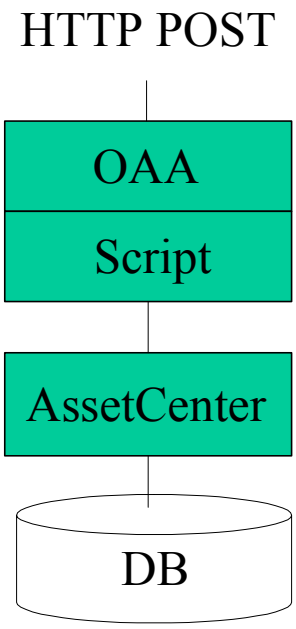
Step 7 The merged record is returned to ServiceCenter applications.

ServiceCenter Flow



- ServiceCenter reads mapping meta-dataServiceCenter issues a database request to external entity
- ServiceCenter issues a database request to the internal the database
- ServiceCenter combines the results

OAA Flow, AssetCenter Example



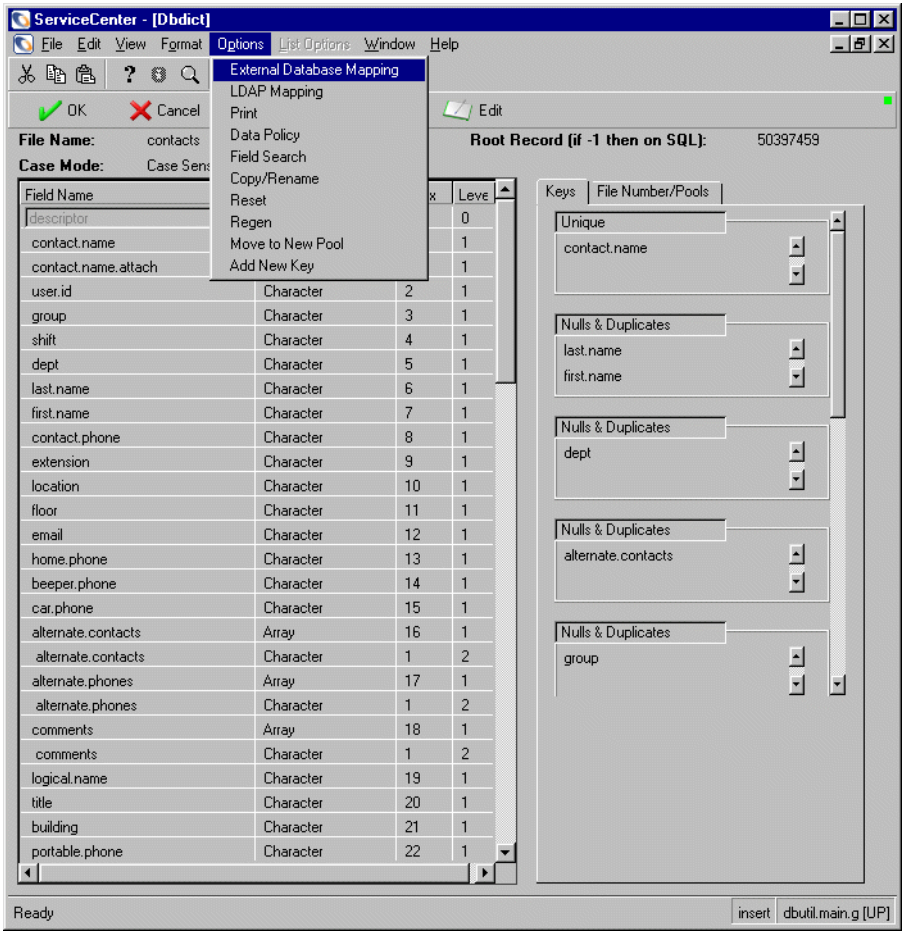
- Receives HTTP Post request
- Loads requested script
- Invokes cardinal database function in script
- Creates result XML document

Mapping the ServiceCenter database to the AssetCenter database

To map the databases:

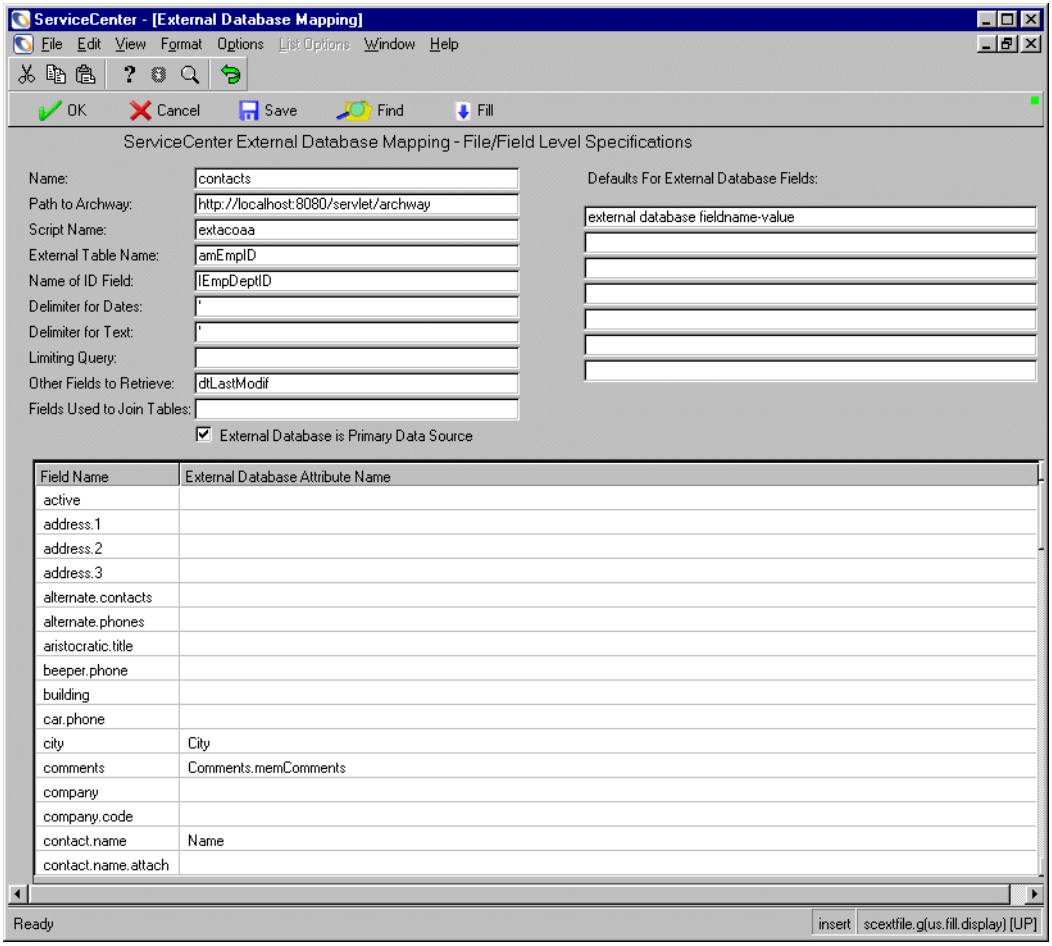
1 Open the Database Dictionary Utility, type the name of the table to be mapped, and press Enter or click the **Search** button. For the example, use contacts. (See *To open a file using the Database Dictionary Utility:* on page 105.)

The Database Dictionary record for your file is displayed.



2 Open the **Options** menu and select **External Database Mapping**.

The External Database Mapping utility is displayed.



3 Fill in the fields according to the table *Fields of the External Database Mapping Utility* on page 163.

Important: If a record includes a linked field, then a text box should be read-only. If you require an editable field, then map to the AssetCenter link directly and use a combo box. For large lists, use a fill box. For more information about creating fill boxes or combo boxes, see the System Tailoring Guide.

Note: In the following tables, the ^ character indicates that an expression is being used.

Table 8-1: Fields of the External Database Mapping Utility

Field	Definition
Path to Archway	The http address for OAA Servlet.
Script Name	The OAA script name that will be loaded and executed by Archway for this external table. This script must be written for the external database system. Peregrine provides an AssetCenter script. This script can also serve as an example of how to write a script for your application. The script name for access to AssetCenter 3.50 or 3.0 is `extac`. The script name for access to AssetCenter 4.1 is `extacoaa`.
External Table Name	The name for the table in the external database.
Name of ID Field	The name of the field that uniquely identifies objects from this table. Optional and if not specified then the fields in the SC unique key are used. This field is required for an AssetCenter mapping as it uniquely identifies the record to AssetCenter.
Delimiter for Dates	The delimiter used for dates in query text.
Delimiter for Text	The delimiter used for text in query text.
Limiting Query	A Query that is added to each user query, that further limits the records that are returned.
Other fields to retrieve	A list of semicolon separated fields that should be retrieved even if not mapped. These fields can be used in expressions even though not explicitly mapped.
Fields Used to Join Tables	The names of the fields in the external database that are used to join the external table with the ServiceCenter file. If none are specified, then the fields mapped to the ServiceCenter unique key are used. This is only needed when ServiceCenter is the primary source for the record list.
Defaults for External Database Fields	During an insert, this establishes any default value that for fields in the external database that are not mapped.
External Database is Primary Data Source	If checked, the external Database is used as the primary source. If not checked, ServiceCenter will be used.

Table 8-1: Fields of the External Database Mapping Utility

Field	Definition
Field Name	The field in the ServiceCenter Database Dictionary.
External Database Attribute Name	The name of the field in the external database to use as source for the ServiceCenter field. - This field may contain an expression if processing is needed to convert the external value into the value wanted or needed by ServiceCenter. - During expression processing, other fields in the file can be obtained by referencing the field in $file - Prefixing the field name with extdbreferences Fields from the external database. For example: ^toupper(extdb.Name in $file+","+extdb.FirstName in $file)

Example Mappings between ServiceCenter and AssetCenter

These are sample mappings. They may not work on your system.

contacts

Access Information:

Prompt	Field Value	Comments
Name:	contacts	
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway	
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.	
External Table Name:	amEmplDept	
Name of ID field:	lEmplDeptId	
Delimiter for Dates:	'	
Delimiter for Text:	'	

Prompt	Field Value	Comments
Other Fields to Retrieve:	dtLastModif	
Limiting Query:	bDepartment=0	AssetCenter stores both contacts and departments in the amEmplDept table. This limiting query ensures that only the contact records are returned.

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions	Comments
contact.name	Name	
contact.phone	Phone	
cost.center	lCostId	
dept	lParentId	
email	Email	
fax.phone	Fax	
first.name	FirstName	
form.of.address	MrMrs	The valid values for this field come from an AssetCenter list. Within ServiceCenter they come from a file called saphrformofaddress. The values need to be consistent. You can see below that we have mapped the saphrformofaddress ServiceCenter file to the proper AssetCenter list.
home.address.1	Location.Address1	Using a link from AssetCenter to get the home address line 1. *
home.address.2	Location.Address2	Using a link from AssetCenter to get the home address line 2. *
last.name	^extdb.Name in $file	Since the Name field has already been used, an expression is used here to get it for the second time.
location	Location	

Field Name in ServiceCenter	Mapping Instructions	Comments
location.code	LLocaId	
manager	Supervisor	
title	Title	The valid values for this field come from an AssetCenter list. Within ServiceCenter they come from a file called saphrttile. The values need to be consistent. You can see below that we have mapped the saphrtitle ServiceCenter file to the proper AssetCenter list.
user.id	IDNo	
valid.from	Costcenter.dStart	
valid.to	Costcenter.dEnd	

* If a record includes a linked field, then a text box should be read-only. If you require an editable field, then map to the AssetCenter link directly and use a combo box. For large lists, use a fill box. For more information about creating fill boxes or combo boxes, see the System Tailoring Guide.

dept

Access Information:

Prompt	Field Value	Comments
Name:	dept	
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway	
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.	
External Table Name:	amEmplDept	

Prompt	Field Value	Comments
Name of ID field:	lEmplDeptId	
Delimiter for Dates:	'	
Delimiter for Text:	'	
Other Fields to Retrieve:	dtLastModif	
Limiting Query:	bDepartment=1	AssetCenter stores both employees and departments in the amEmplDept table. This limiting query ensures that only the department records are returned.

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions
dept	Name
dept.id	IDNo

location

Access Information:

Prompt	Field Value
Name:	location
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.
External Table Name:	amLocation
Name of ID field:	lLocaId

Prompt	Field Value
Delimiter for Dates:	'
Delimiter for Text:	'
Other Fields to Retrieve:	dtLastModif
Fields Used to Join Tables:	Name

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions	Comments
address	Address1;Address2	The two fields from amLocation are used to populate the array field in ServiceCenter
city	City	
comments	Comment.memComment	Using a link from AssetCenter to get the comments. *
country	Country	
level	sLvl	
location	Fullname	
location.code	lLocaId	
location.name	Name	
parent	Parent.Fullname	Using a link from AssetCenter to get the parent. *
state	State	
zip	ZIP	

* If a record includes a linked field, then a text box should be read-only. If you require an editable field, then map to the AssetCenter link directly and use a combo box. For large lists, use a fill box. For more information about creating fill boxes or combo boxes, see the System Tailoring Guide.

vendor

Access Information:

Prompt	Field Value
Name:	vendor
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.
External Table Name:	amCompany
Name of ID field:	lCpyId
Delimiter for Dates:	'
Delimiter for Text:	'
Other Fields to Retrieve:	dtLastModif

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions	Comments
address	Address1;Address2	The two fields from amCompany are used to populate the array field in ServiceCenter.
city	City	
country	Country	
email	EMail	
fax	Fax	
phone	Phone	
sales	Contacts.Name	Using a link from AssetCenter to get the name of the sales person. *
sales.rep.phone	Contacts.Phone	Using a link from AssetCenter to get the phone number of the sales person. *

Field Name in ServiceCenter	Mapping Instructions	Comments
state	State	
type	Qualif1	
vendor	Name	
vendor.id	Code	
vendor.name	^extdb.Name in $file	The **Name** field was already referenced once. Therefore we are using an expression to pick it up the second time.
zip	ZIP	

* If a record includes a linked field, then a text box should be read-only. If you require an editable field, then map to the AssetCenter link directly and use a combo box. For large lists, use a fill box. For more information about creating fill boxes or combo boxes, see the ServiceCenter System Tailoring Guide.

saphrcostcenter

Access Information:

Prompt	Field Value
Name:	saphrcostcenter
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway
Script Name:	If you are using AssetCenter 3.51/3.6, use the script **extac**. If you are using AssetCenter 4.1, use the script **extacoaa**.
External Table Name:	amCostCenter
Name of ID field:	lCostId
Delimiter for Dates	'
Delimiter for Text	'
Other Fields to Retrieve:	dtLastModif

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions
business.code	lCostId
cost.center	Title
valid.from	dStart
valid.to	dEnd

saphrformofaddress

Access Information:

Prompt	Field Value	Comments
Name:	saphrformofaddress	
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway	
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.	
External Table Name:	amItemListVal	
Name of ID field:	lItemListValId	
Delimiter for Dates	'	
Delimiter for Text	'	
Other Fields to Retrieve:	dtLastModif;lItemListId	

Prompt	Field Value	Comments
Limiting Query:	lItemListId=7	amItemListVal is a general table that is used to hold LISTS for values. Each list has an ID value. The ID value that identifies the list of valid forms of address is 7.
Defaults for External Database Fields:	lItemListId=7	When doing an insert of a new form of address, the list ID value must be set to 7.

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions
id	lItemListValId
foadd	Value

saphrtitle

Access Information:

Prompt	Field Value	Comments
Name:	saphrtitle	
Path to Archway:	If you are using AssetCenter 3.51/3.6, use the path: http://localhost/servlet/archway If you are using AssetCenter 4.1, use the path: http://localhost:8080/oaa/servlet/archway	
Script Name:	If you are using AssetCenter 3.51/3.6, use the script extac. If you are using AssetCenter 4.1, use the script extacoaa.	
External Table Name:	amItemListVal	

Prompt	Field Value	Comments
Name of ID field:	lItemListValId	
Delimiter for Dates	'	
Delimiter for Text	'	
Other Fields to Retrieve:	dtLastModif;lItemListId	
Limiting Query:	lItemListId=8	AmItemListVal is a general table that is used to hold LISTS for values. Each list has an ID value. The ID value that identifies the list of valid titles is 8.
Defaults for External Database Fields:	lItemListId=8	When doing an insert of a new title the list ID value must be set to 8.

Mapping Information:

Field Name in ServiceCenter	Mapping Instructions	Comments
abbreviation	lItemListValId	
description	Value	
Object.type	^"Title	This is a constant forced into the field.

Functions Required in the OAA Script

There are six functions that need to be supplied by the OAA script that supports the external database. Three of these functions are used in queries to retrieve records. The other three are used to update, delete, and insert records.

The three types of queries required are:

- A general query which is passed in SQL format.
- A query for a specific record.

- A query for a set of specific records.

  This query is aimed to improve performance. Rather than executing the script multiple times using option 2, the script is executed a single time with a request for multiple records.

The following functions must be provided by each script:

Function	Definition
Extquery	General query to get a record set
Extgetunique	Get a specific record
Extbatchget	Get a batch of records
Extupdate	Update a specific record
Extdelete	Delete a specific record
Extinsert	Insert a new record

Extquery — General Query to Get a Record Set

A general query in SQL format is passed to the script. The script is expected to return KEY fields only, which establish the record set that the requestor will view. As part of the mapping, a limiting query may be specified which is automatically appended to any query the requestor may issue before the query is presented to the script. The delimiters used in the query for text and date fields are provided during the mapping. Only 128 KEY values are expected on the initial query. This is done to limit the amount of data that is actually transmitted. If the requestor asks for the 129th record then a new query will be provided to the script to get more key values. The idea is that the requestor will generally just look at the first few records; therefore, there is no need to gather all the data in the initial request. Here is an example of the document provided to the script for an extquery request and the output expected from such a request.

Notice that the <**count**> field is used to tell the script how many entries to return. The requestor wants to know all the employees whose name starts with "B". The phrase (**bDepartment=0**) is a limiting query that was specified during the mapping process because the person doing the mapping had the

knowledge that both employees and departments are stored in the amEmplDept table and that employees are indicated when the **bDepartment** value is **0**. The select statement includes the list of fields whose value should be returned.

Input — Request to get all employees with a name that start with a "B"

```
<getkeys>
 <query>
  select FirstName,Name,lEmplDeptId
   FROM amEmplDept
   WHERE ((Name LIKE 'B%')) AND (bDepartment=0)
   ORDER BY Name,FirstName ASC
 </query>
 <count>128</count>
</getkeys>
```

Output — A list of amEmplDept entries with the key fields returned

The output from the above query. Notice that the number of items actually returned is indicated with the **_countFound** field and that the **_more** field is used to indicate if there are more records that qualified but were not returned because the **_count** field was exceeded.

```
recordset>
 _count="128"
 _countFound="6"
 _more="0"
 _start="0">
 <amEmplDept>
  <FirstName>Laure</FirstName>
  <Name>Bailly</Name>
  <lEmplDeptId>17269</lEmplDeptId>
 </amEmplDept>
.
</recordset>
```

Extgetunique — Get a Specific Record

Extgetunique is a request to get the detailed information about a single record. The fields that should be returned are defined by the <**fields**> element. The record that is wanted is defined by the <**keyset**> element.

A <**recordset**> should be returned that contains the values for the desired <**fields**>. Once the key values have been returned by the *extquery* request and the requestor has indicated a desire to see a particular record, then an extgetunique request will be issued to get all the data for the desired record. If we believe that the requestor will look at a number of records, extbatchget is issued for a number of records rather than multiple extgetunique requests for individual records. An example of the input and output from extgetunique is shown below.

Input — Request to get entry 17269 from amEmplDept

```
<getunique>
 <table>amEmplDept</table>
 <fields>
  <name>Phone</name>  <name>CostCenter</name>
<name>lParentId</name>
  <name>EMail</name>  <name>Fax</name>  <name>FirstName</name>
  <name>MrMrs</name>  <name>Location.Address1</name>
  <name>Location.Address2</name>  <name>Name</name>
  <name>Location</name>  <name>lLocaId</name>
  <name>Supervisor</name>  <name>Field1</name>
  <name>Title</name>  <name>IDNo</name>  <name>lEmplDeptId</name>
 </fields>
 <keyset>
  <key><lEmplDeptId>17269</lEmplDeptId></key>
 </keyset>
</getunique>
```

Output — An amEmplDept entry with all the requested data

```
<recordset>
 <amEmplDept>
  <Phone>(650) 572-9025</Phone>
  <CostCenter>Common Line</CostCenter>
  <lParentId>17231</lParentId>
  <EMail>gomez@taltek.com</EMail>
  <Fax>(650) 572-9099</Fax>
  <FirstName>Marc</FirstName>
  <MrMrs>Mr.</MrMrs>
  <Location.Address1>5569 Turner Dr.</Location.Address1>
  <Location.Address2>P.O. Box 120</Location.Address2>
  <Name>Gomez</Name>
```

```
  <Location>/San Mateo site/Building 02/Warehouse/003 - Hall/</Location>
  <lLocaId>17413</lLocaId>
  <Supervisor>Bernard, Cathy</Supervisor>
  <Field1></Field1>
  <Title>Maintenance</Title>
  <IDNo>DEMO-M030</IDNo>
  <lEmplDeptId>17235</lEmplDeptId>
 </amEmplDept>
</recordset>
```

Extbatchget — Get a Batch of Records

Extbatchget is a performance enhancement. Rather than doing multiple getunique requests, the requests are batched and presented to the script as one request for multiple records. Input is the same as the extgetunique request, except that the <keyset> consists of multiple entries. Output is a recordset which returns one record for each key that was requested

Input — Request to get series of entries (keyset) from amEmplDept

```
<getunique>
 <table>amEmplDept</table>
 <fields>
  <name>Phone</name>
  <name>CostCenter</name>
  <name>lParentId</name>
  <name>EMail</name>
  <name>Fax</name>
  <name>FirstName</name>
  <name>MrMrs</name>
 </fields>
 <keyset>
  <key><lEmplDeptId>17269</lEmplDeptId></key>
  <key><lEmplDeptId>17223</lEmplDeptId></key>
  <key><lEmplDeptId>17174</lEmplDeptId></key>
  <key><lEmplDeptId>17166</lEmplDeptId></key>
 </keyset>
</getunique>
```

Output — recordset returned with an amEmplDept entry for each key requested

```
<recordset>
 <amEmplDept>
  <lEmplDeptId>17235</lEmplDeptId>
  <Phone>(650) 572-9025</Phone>
  .
</amEmplDept>
 <amEmplDept>
  <lEmplDeptId>17234</lEmplDeptId>
  <Phone>(650) 572-9024</Phone>
  .
</amEmplDept>
 <amEmplDept>
  <lEmplDeptId>17233</lEmplDeptId>
  <Phone>(650) 572-9023</Phone>
  .
</amEmplDept>.
 .
</recordset>
```

Extupdate — Update a Specific Record

When an update is done, only those values that have actually been changed along with their key values are sent to the script. If a field with a name of dtLastModif has been mapped for the table, then a query is done first to make sure the record has not been modified by someone else on another system.

Input — Updating the address for a location

```
<amLocation>
 <Address1>Right Down Town</Address1>
 <Address2>P.O. Box 1288</Address2>
 <lLocaId>17369</lLocaId>
</amLocation>
```

Output — The record should be returned

```
<amLocation>
 <Address1>Right Down Town</Address1>
 <Address2>P.O. Box 1288</Address2>
```

```
 <lLocaId>17369</lLocaId>
</amLocation>
```

Extdelete — Delete a Specific Record

Delete is invoked when the requestor has indicated that the record should be deleted. The request simply identifies the table name and the key value for the record that should be deleted. No output other than a clean response is expected from this request.

Input — Request to delete entry 17369 from the amLocation table

```
<amLocation>
 <lLocaId>17369</lLocaId>
</amLocation>
```

Extinsert — Insert a New Record

When an insert is done, all of the fields that have been mapped are provided. Additionally all of the fields identified in the "default values" section during the mapping are also provided.

Input — request to add a new record to the amLocation file

```
<amLocation>
 <Address1>3333 Turner Dr.</Address1>
 <Address2>P.O. Box 1288</Address2>
 <City>Santa Clara</City>
 <Country>United States</Country>
 <sLvl>3</sLvl>
 <Fullname>/San Mateo site/Building 02/Warehouse/007 - Raw material
stock/</Fullname>
 <Parent.Fullname>/San Mateo site/Building 02/Warehouse/</Parent.Fullname>
 <State></State>
 <ZIP>CA 53879</ZIP>
</amLocation>
```

Output — The output from the request should be the complete record

```
<amLocation>
 <Address1>3333 Turner Dr.</Address1>
 <Address2>P.O. Box 1288</Address2>
 <City>Santa Clara</City>
 <Country>United States</Country>
```

```
 <sLvl>3</sLvl>
 <Fullname>/San Mateo site/Building 02/Warehouse/007 - Raw material
stock/</Fullname>
 <Parent.Fullname>/San Mateo site/Building 02/Warehouse/</Parent.Fullname>
 <State></State>
 <ZIP>CA 53879</ZIP>
</amLocation>
```

Frequently Asked Questions

Is Federated Data only Supported Between ServiceCenter and AssetCenter?

No, the federated support is a general interface so that data can be federated between ServiceCenter and any external database. However, the script used in OAA to access AssetCenter data is provided. Federating other data would require that a script be written to interface with the external database.

What Prevents Duplicate Updates of the Data?

When an update is attempted, ServiceCenter first checks to make sure that the date last modified (dtLastModif) for the record has not changed. If the date has changed, then the user is notified that the record has been modified. Also if queue entries exist for the record indicating that the external database was not available, then the update is rejected and will continue to be rejected until the external database is available and the queue entries are processed.

What Happens During Updates/Deletes/Inserts When the External Database or Connection to the OAA Server Is Down?

The action is queued to the prgnqman file (a file that handles general queuing within ServiceCenter). The queue is processed by a background task that is started with scenter -que:federated. The HTTP request and the XML document are written to the queue.

The following fields are used in the prgnqman file:

- messageID = The tag value for the record that is being processed
- destination = URL that identifies the OAA server, the script and the action requested
- queueName = SCFEDERATED
- type = The external table name that is being processed
- text = XML document that is being passed to the script

- correlationID = Internal ROW ID used in ServiceCenter for the record being processed

Do All Files Within ServiceCenter Support Federated Data?

Yes, any file in ServiceCenter can support Federation of data. Simply use the **External Database Mapping** option from the Data Dictionary Utility and indicate which of the fields in the file should be mapped to an external source. No application changes are required as the Federation of data takes place at the database layer of the ServiceCenter Run-time Environment.

When Federating ServiceCenter Data with AssetCenter Data, Why Does All the Data that AssetCenter Needs Have to be in AssetCenter?

Currently the architecture of AssetCenter does not allow that product to actively participate in the federated data scheme. AssetCenter is simply not able to access any data that is not directly under its control.

How Are the Business and Data Rules of the AssetCenter Data Invoked for Write Operations?

The OAA script for AssetCenter uses the AssetCenter provided API for accessing the data. This is the same API that is used by the Get.It! product line. This API enforces all the business and data rules.

How is the Federated Solution Different from a Connect.It! Solution?

Connect.It! is used to transfer data from one data source to another and is used in the replication of data. With a federated database there is no replication of data. The data exist in a single place.

Will virtual joins work with Federated data?

Yes, federated data is implemented in the Run-time Environment of ServiceCenter and therefore virtual joins will work as always.

Can Federated Data be Used to Populate Drop Downs on a ServiceCenter Form?

Yes because the Run-time Environment is doing the Federation of the data, the data is available for use in drop downs automatically. The Global List processor may also be used to keep lists current, and access federated data.

Can Multiple ServiceCenter Servers Connect to the Same External Database?

The is up to the external database, but as long as the external database is SQL based, there would not be any problem.

Can One ServiceCenter Connect to Multiple External Sources?

The source of the data is determined when a file is mapped, and it is possible for each file to have a different source. For an individual file there can only be one external source. If multiple sources are required, then the OAA script would have to handle that.

Can Data Manipulation be Applied Between the External Source and the ServiceCenter Form?

There are no restrictions on what can be done with the data just because it is federated. The applications within ServiceCenter are not aware that Federation has taken place and therefore all manipulations that are available in ServiceCenter are also available when using federated data.

Will Format Control and Data Policy Apply on the Way Out from a ServiceCenter Form to the External Source?

Yes, Format Control and data policy will be invoked as they would for any data.

Will Federated Data be Available to Dynamic View Dependencies?

Yes, there are no restrictions on the use of federated data.

Will Federated Data be Controllable via Mandanten?

Yes, data that is federated can be the target of Mandanten protection.

Can one file in ServiceCenter be mapped to multiple tables in the external source?

Not directly. However, when a mapping is being done to an AssetCenter system, all of the links provided by AssetCenter can be used. Notice in the example mappings of the ServiceCenter **vendor** file to the AssetCenter **amcompany** table that mappings were done to **contacts.name** and **contacts.phone**. These links between tables in AssetCenter are used to effectively pull data from multiple AssetCenter tables into a single ServiceCenter file.

9 The Database Manager Utility

CHAPTER

This chapter was designed to provide ServiceCenter system and database administrators with an introduction to access and edit ServiceCenter database files with the Database Manager Utility.

Topics in this chapter include:

Overview

The ServiceCenter Database Manager runs in two modes, standard and administrative.

- In **standard mode**, behavior is determined by whatever security you have in place. Request management, for example, would use Request security. The standard database would use Format Control. In standard mode, an administrator does not necessarily see all options. Options that are potentially troublesome for the system, mass-updates, regenerating keys, etc., are kept out of sight, so as to prevent accidental usage.
- In **Administration Mode**, (the checkbox selected) a system administrator will have rights to ALL options.

Administration Mode can be thought of as similar to root privileges on Unix systems. Administration Mode is powerful in that you can make changes that affect the system as a whole, and as a result, can be dangerous. Peregrine recommends that day to day administration work be done in standard mode.

Only users with the SysAdmin capability word defined in their operator file will see the **Administration Mode** checkbox.

Placing the capability word AlwaysAdmin in a user's operator file puts that user in **Administration Mode** by default.

Accessing a Record from the Database Manager Utility

The first step is to open the Database Manager Utility.

To open the Database Manager Utility from the Toolkit tab:

1 Log in as an administrator. Use FALCON in the default system.

2 Select the **Toolkit** tab on the ServiceCenter system administrator's main menu.

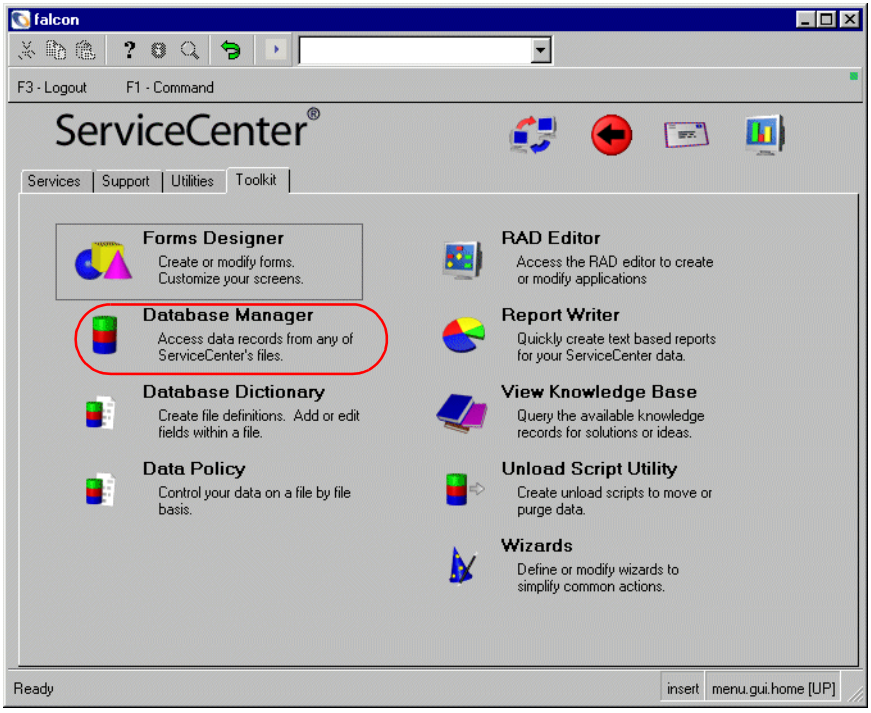


Figure 9-1: System Administrator's Home Menu — Toolkit tab

3 Click **Database Manager**.

— Or —

To open the Database Manager Utility from the Command line:

▶ Type db on the Command line and press Enter.

Figure 9-2: ServiceCenter Command line

The Database Manager dialog box is displayed.

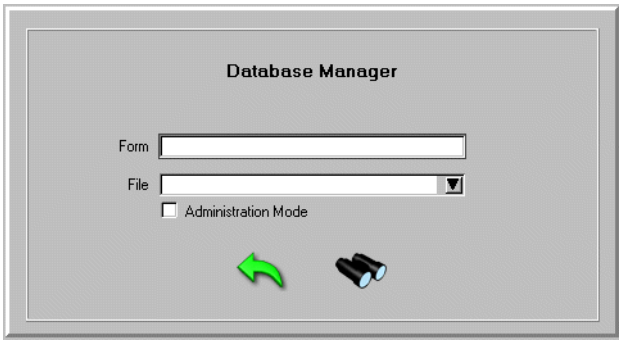


Figure 9-3: Database Manager dialog box

The next step is to open the record.

To open a record:

1 Type the name of a form in the **Form** field or a file in the **File** field. You also can leave the fields blank to return a list of all records.

2 Click **Search** or press Enter.

If you left the fields blank or more than one record matches the search criteria, a QBE list is displayed.

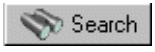
- Select one of the listings.

A blank form is displayed.

3 Enter any information that you might know.

For example, you might enter known information in key fields such as names of operators, names of devices, or problem numbers.

Note: Searches against fields that are not keyed will slow your search.



4 Click **Search** or press Enter.

A record list is displayed along with the highlighted record. If multiple records are available, click on the record that you want to see.

10 CHAPTER Record Retrieval

This chapter was designed to aid ServiceCenter system and database administrators retrieve ServiceCenter database records based on particular user-provided search criteria (query).

It is divided into the following sections:

Overview

Searching is case sensitive by default, unless your database has been set up for case insensitive searching. To set up the database for case insensitive searching, see *Setting Case Mode for Searching the P4 File System* on page 23.

For information on how to improve query speed, see *Improving Query Speed* on page 92.

Three methods are used to query records from a ServiceCenter database:

- Simple queries are defined via the QBE (Query-by-Example) method.
- More complex queries are defined via the Query Window method.
- other queries can be performed using the IR Expert utility (Information Retrieval engine).

Queries are logical expressions that evaluate to either *true* or *false* against each record (i.e. the record matches the search criteria or not).

If true, the record is retrieved.

If false, the record is not retrieved. If no record is retrieved, an error message is displayed.

Database Manager searches for records that match the query (using either method) and displays a list of matching records on a QBE list.

A QBE list is a specific form displaying pertinent field values from each record found in the search. If a customized QBE list format for this list is not created by the user, ServiceCenter will create one (<filename>.qbe) based on key fields defined in the database dictionary for this file. The user then selects a record from this list by positioning the cursor on the desired record and double-clicking or pressing Enter.

If the search criteria uniquely defines only one record, that record is displayed using the form used to define the search, or the one defined by a form name in the record depending on Format Control.

If the search criteria does not find any records, the message: *No records found to satisfy QBE search argument(s)* is returned.

Relational and Logical Operators

Both the QBE Method and the Query Window Method require the use of *Relational Operators*, next section, and *Logical Operators* on page 190.

Relational Operators

A relational operator makes a comparison, then generates results based on whether the comparison is true or false. The relational operators are defined in the table below:

Relational Operator	Definition
#	starts with Starts with is used if no other relational operator is specified. (#) is the default query.
¬# (0S/390)	does not start with
~# (Unix and NT)	does not start with
=	equal to
¬= (0S/390)	not equal to
~= (Unix and Windows)	not equal to
<>, ><	not equal to
<	less than
<= or =<	less than or equal to
>	greater than
>= or =>	greater than or equal to
isin	is element in
like	is similar to This operator is used only when querying character fields and enables wildcard searches on these fields.

Queries using the *equal to* relational operator are more efficient than queries using other relational operators and should be used whenever possible. Queries using the *equal to* relational operator are valid on all types of fields, although the *equal to* operator is seldom useful when querying date/time fields. Use the *greater than (>)* operator when querying on date/time fields since these values include seconds which make the *equal to (=)* operator impractical.

For example, if you are querying for records in the problem file that were opened at 11 a.m. on June first. It is unlikely a record was opened exactly at 11 a.m. (i.e., 11 hours, 00 minutes, 00 seconds), so an equal to query is not advisable. However, querying with `>06/01/92 11:00` in the open.time field retrieves every record opened after 11 a.m., starting with 1 second after 11 a.m.

Logical Operators

A logical operator evaluates one or two boolean (true/false) expressions and determines whether the entire expression is *true* or *false*, based on the operator.
Logical Operators, syntax choices, and their symbols are as follows:

Logical Operator	Syntax	Symbol
AND	value AND value	&
OR	value OR value	|

See *Using the AND and OR Logical Operators* on page 198.

Keys

ServiceCenter uses keys identifying fields in files to select data efficiently. If a partially-keyed or non-keyed query is detected in Database Manager, the operator is warned so that the search can be abandoned. The ability to perform partially-keyed queries and the ability to modify search interval parameters is controlled via `partial.key` system capability words.

Table 10-1: Capability Words

Capability Word	Description
partial.key	Allows operator to execute a partially keyed or non-keyed query.
partial.key.msg.skip	Allows operator to bypass the partial key warning screen.
mod.time.limit	Allows operator to modify the default time limit for partially keyed or non-keyed database searches.

Used in combination, these capability words help control performance degradation due to inefficient queries.

Refer to the ServiceCenter *System Administrator's Guide* for more information on capability words.

Retrieving Records via the P4/QBE Method

P4 (PEREGRINE FOUR)/QBE (Query-by-Example) is a standard feature for accessing records in a database. It is most frequently encountered at prompts where a form or file name is required.

To retrieve records via a QBE search:

- Type the desired values on any form displaying a Filename, Fieldname or other prompt and press Enter.

  The query will be processed looking for similar examples, based on the values the user enters at the prompt fields. These QBE search values can be made more efficient through the use of relational operator characters recognized by the P4 Database.

Using the *starts with (#)* Relational Operator

Database Manager automatically assumes that values entered into character type fields are intended for a starts with query and prefixes the value entered with the # symbol. However, this is not true if the value that has been entered contains any wildcard characters (" * " or " ? "). In that case a like query is generated. If you want to do a starts with query containing a wildcard character, enter a # character in front of the value.

The following example demonstrates retrieving all capability records with a capability name that begins with the letter p.

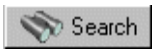
Note: This search will not return records that begin with upper case P, if any were there, unless your database has been set up for case insensitive searching. To set up the database for case insensitive searching, see *Setting Case Mode for Searching the P4 File System* on page 23.

To retrieve records using the *starts with (#)* relational operator:

1. Open the capability form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

   A blank capability form is displayed.

2. Type p in the **Capability** field.



3 Click **Search** or press Enter.

Note: Database Manager performs the *starts with* search and displays the record list of matching records, if any, using the capability form.

Notice that all values listed under the Capability heading in Figure 10-1 on page 192, start with **p**.

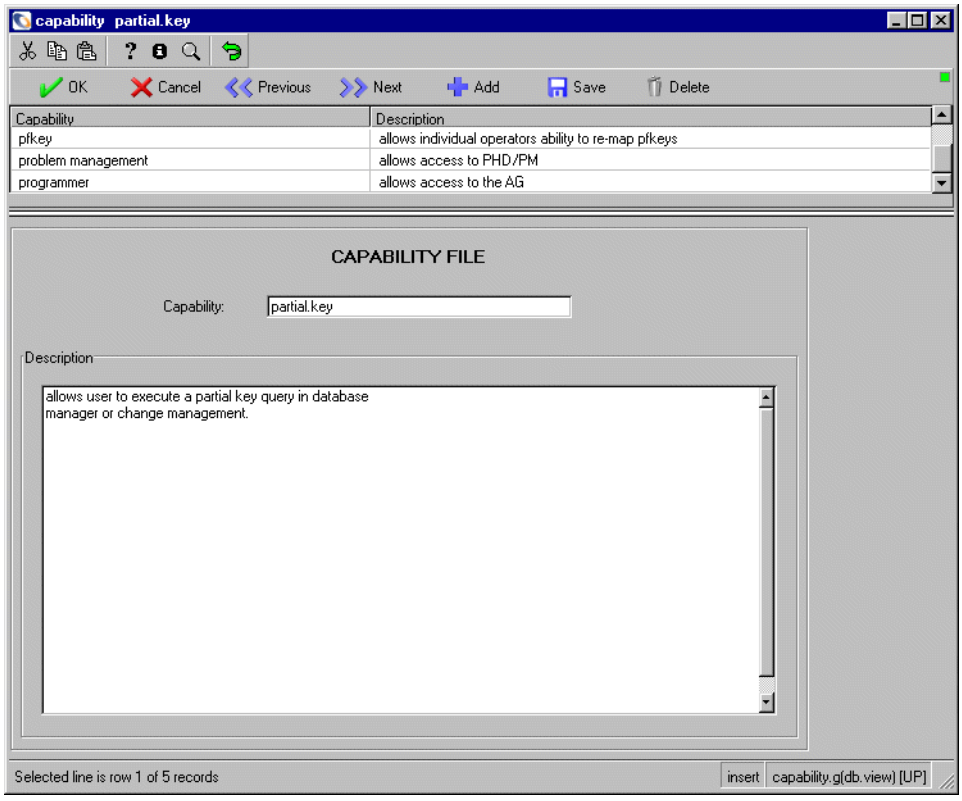


Figure 10-1: Results of "starts with" query

4 Click on the listing for the record that you want to see.

Using the *equal to (=)* Relational Operator

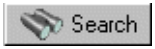
Queries using the *equal to* relational operator are more efficient than queries using other relational operators and should be used whenever possible. Queries using the *equal to* relational operator are valid on all types of fields, although seldom useful when querying date/time fields.

The following example demonstrates retrieving all device records where the location field is *equal to* ACME HQ.

To retrieve records using the *equal to (=)* relational operator:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =device in the **Form** field.
3. Click **Search** or press Enter.

   The device form is displayed.
4. Type =ACME HQ in the location field.



5. Click **Search** or press Enter.

   Database Manager performs the *equal to* search and displays the record list of matching record(s), if any, using the device form.

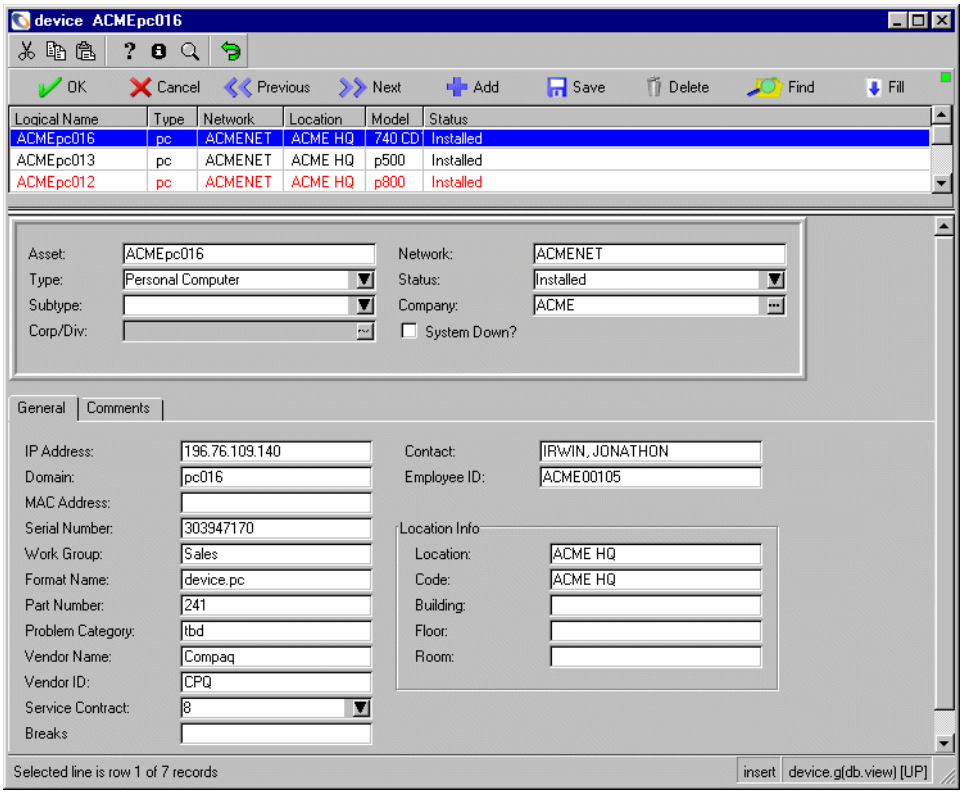


Figure 10-2: Results of "equal to" query

Note: Since the *equal to (=)* operator was prefixed to the value being searched for (ACME HQ), the Database Manager searches the device file for any records that have the EXACT value entered (ACME HQ). Notice, all values listed under location have the EXACT value of ACME HQ.

6 Click on the record that you want to view.

Using the *greater than (>)* Relational Operator

The *greater than (>)* operator can be used for any field type. It is most frequently used when querying on date/time fields because these values include seconds which make using the *equal to (=)* operator impractical.

For example, if you are querying for records in the problem file that were opened at 11 a.m. on June first, it is unlikely that a record was opened exactly at 11 a.m. (i.e. 11 hours, 00 minutes, 00 seconds). An *equal to* query therefore is not advisable. However, querying with `>06/01/01 11:00` in the open.time field retrieves every record opened after 11 a.m., starting with 1 second after 11 a.m.

The following example demonstrates retrieving all device records with the logical.name field value greater than a character value of `d`.

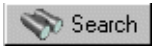
To retrieve records using the *greater than (>)* relational operator:

1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type `=device` in the **Form** field.

3 Click **Search** or press Enter.

The device form is displayed.



4 Type `>d` in the Asset field. Click **Search** or press Enter.

Database Manager performs the *greater than* search, and displays the QBE list of matching record(s), if any, using the device form.

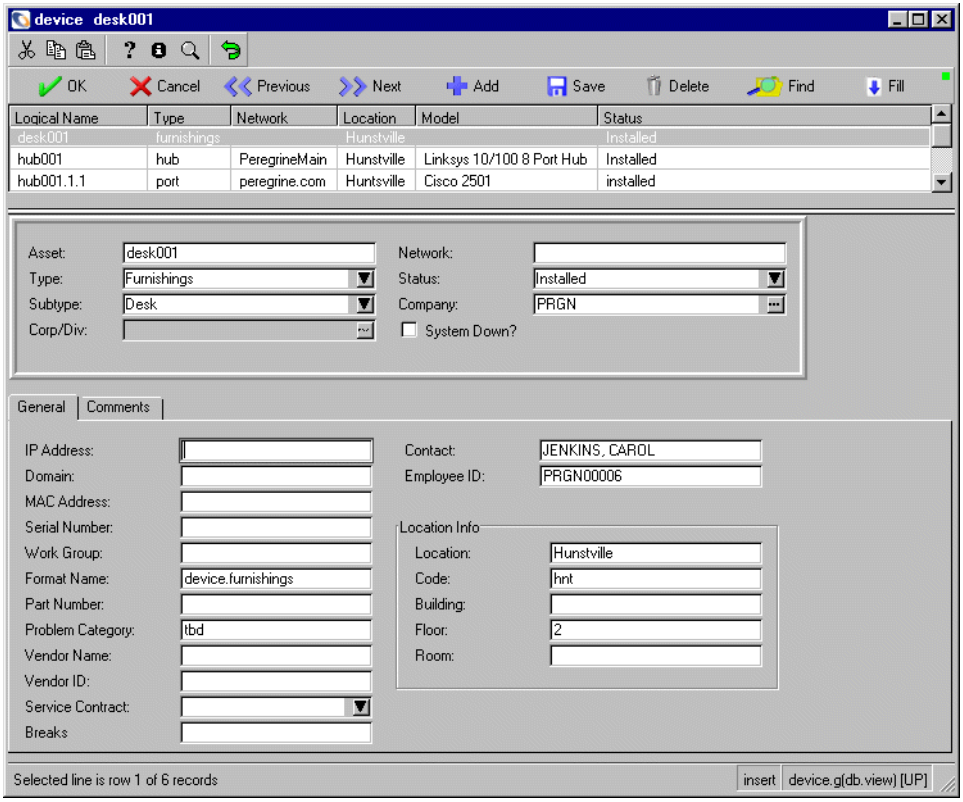


Figure 10-3: Results of "greater than" query

5 Click on the record that you want to view.

Using the *less than (<)* Relational Operator

The following example demonstrates retrieving all device records where the IP Address field value is less than a character value of *196*.

To retrieve records using the *less than (<)* relational operator:

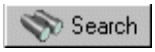
1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type `=device` in the **Form** field.

3 Click **Search** or press Enter.

The device form is displayed.

4 Type **<196** in the **IP Address** field.



5 Click **Search** or press Enter.

Database Manager performs the *less than* search and displays the record list of matching record(s), if any, using the **device** form.

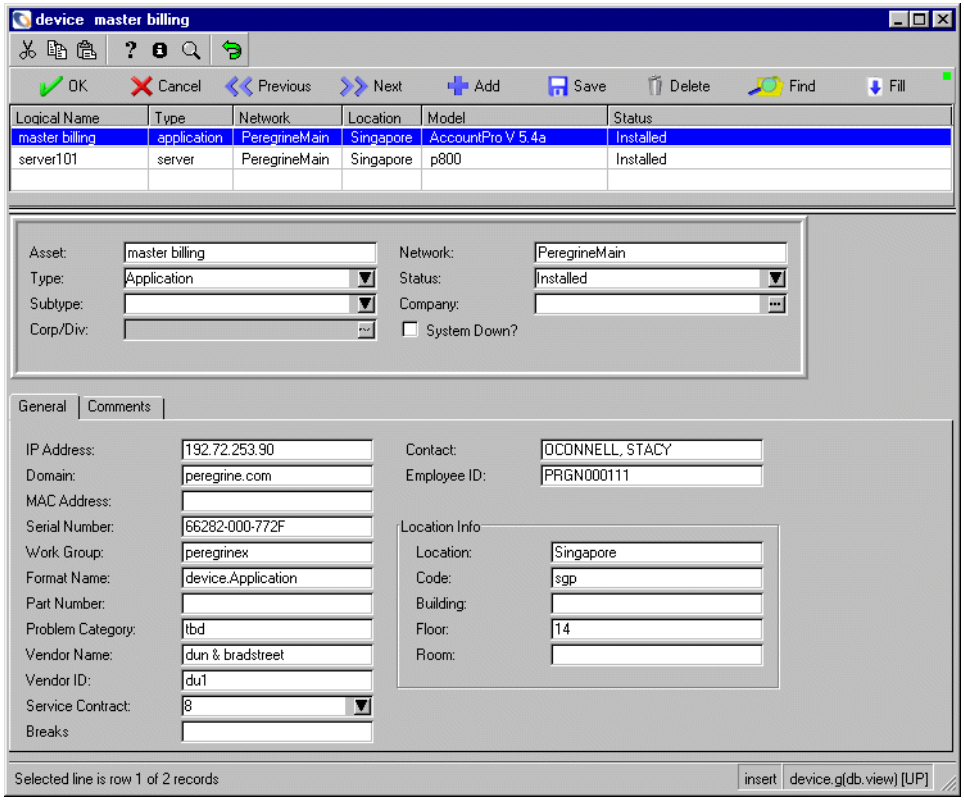


Figure 10-4: Results of "less than" query

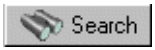
6 Click on the record that you want to view.

Using the *like* Relational Operator

The following example demonstrates retrieving all **contacts** records that have an **e** as the second letter of their last name.

To retrieve records using the *like* relational operator:

1 Access the **contacts** file from the **Support** tab in the system administrators main menu.



2 Type **?e*** into the field labelled **Last Name**. Click **Search** or press Enter.

The database manager performs the search and displays the record list of matching records. In this case the records matched the query: **last.name like "?e*"** See Figure 10-5 on page 197.

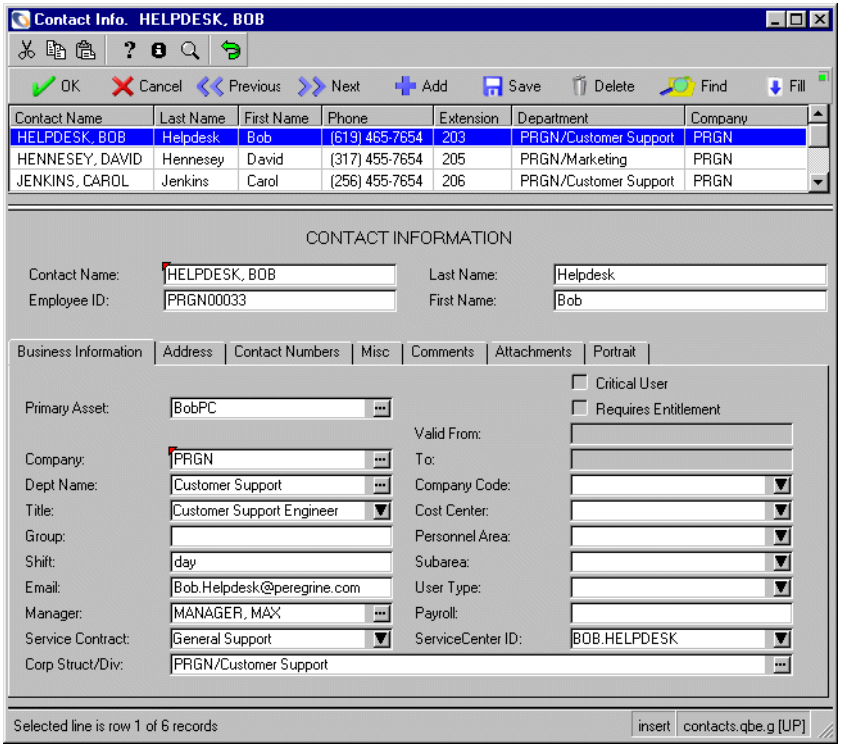


Figure 10-5: Wildcard query using the "like" operator

Because the database manager found two wildcard characters in the last name that was entered, it automatically generated a like query. If you want to find all records that start with **?e*** you have to enter **#?e***. The leading **#** character indicates that you want to perform a starts with query.

The wildcard characters * and ? represent any character. So if you want to search for these characters literally, you have to prefix these characters with the \ character. For example, if you want to find all records that start with any

character followed by an e and then followed by a * character, you have to enter ?e* into the field labelled **Last Name**. This still generates a like query (because the ? is a wildcard character) but verifies that the last name ends with a literal *.

From the above record list no record in this example would be displayed. Instead, records with a last name of Ae*, be*, ?e* or *e* would be retrieved and displayed by the database manager.

A *like* query does not automatically perform a *starts with* query as well. If you want to see all records that start with any character followed by e* and then have any number of any characters you have to enter ?e**. Such a query would display records with a last name of Ae*, he*e, fe*, Be*bcdefg.

The characters *, ? and \ as they are described here can be modified to any other character by using the parameter wildcardcharacters in the sc.ini file residing on the ServiceCenter server.

Using the *AND* and *OR* Logical Operators

A range can be determined by using the logical operators between the relational operators. Allowable values are AND and OR. The relation must be spelled out. The substitute characters for AND and OR (i.e. &, |) cannot be used. The range is defined by the conjunction of the less than and greater than operators.

Syntax:

- <value AND <value
- <value OR <value
- >value AND <value
- >value OR <value

Note: To do a range query for numeric fields, enter the query in the query window. See *Retrieving Records via the Query Window* on page 207.

The following example demonstrates retrieving all device records where the Serial Number field value is more than a character value of 500 *and* less than a character value of 1000.

To retrieve records within a date or time range:

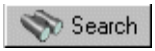
1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type `=device` in the **Form** field.

3 Click **Search** or press Enter.

The **device** form is displayed.

4 Type `>500 AND <1000` in the **Part Number** field.



5 Click **Search** or press Enter.

Database Manager performs the *less than* search and the *greater than* search and displays the record list of matching record(s), if any, using the **device** form.

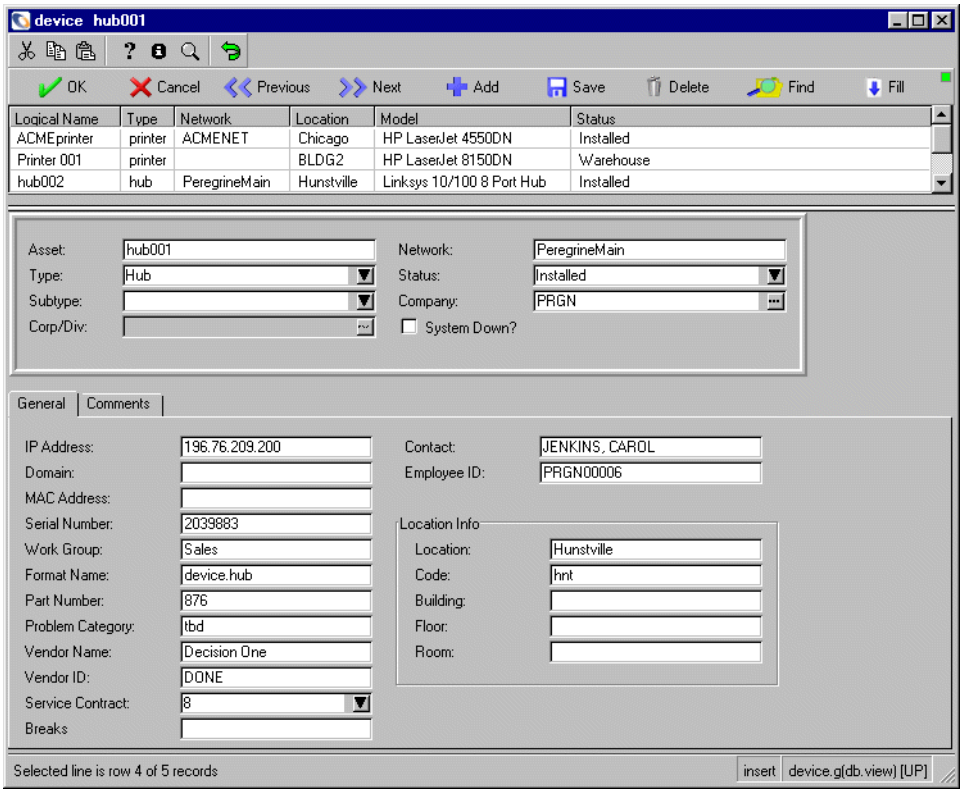


Figure 10-6: Results of "less than" AND "greater than" query

6 Click on the record that you want to view.

Using the *Not* Symbol with Logical or Relational Operators

To exclude certain records from a query, the *not* symbol can be used with any of the logical or relational operators.

For example:

- ¬=i (OS/390)
- ~=i (Unix and Windows)

— Or —

- ¬#i (OS/390)
- ~#i (Unix and Windows)

The following example demonstrates retrieving all device records except those with a value in the vendor name field starting with the character C.

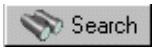
Note: This search will return records that begin with lower case c, if any were there, unless your database has been set up for case insensitive searching. To set up the database for case insensitive searching, see *Setting Case Mode for Searching the P4 File System* on page 23.

To exclude records by using *not* with a relational operator:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =device in the **Form** field.
3. Click **Search** or press Enter.

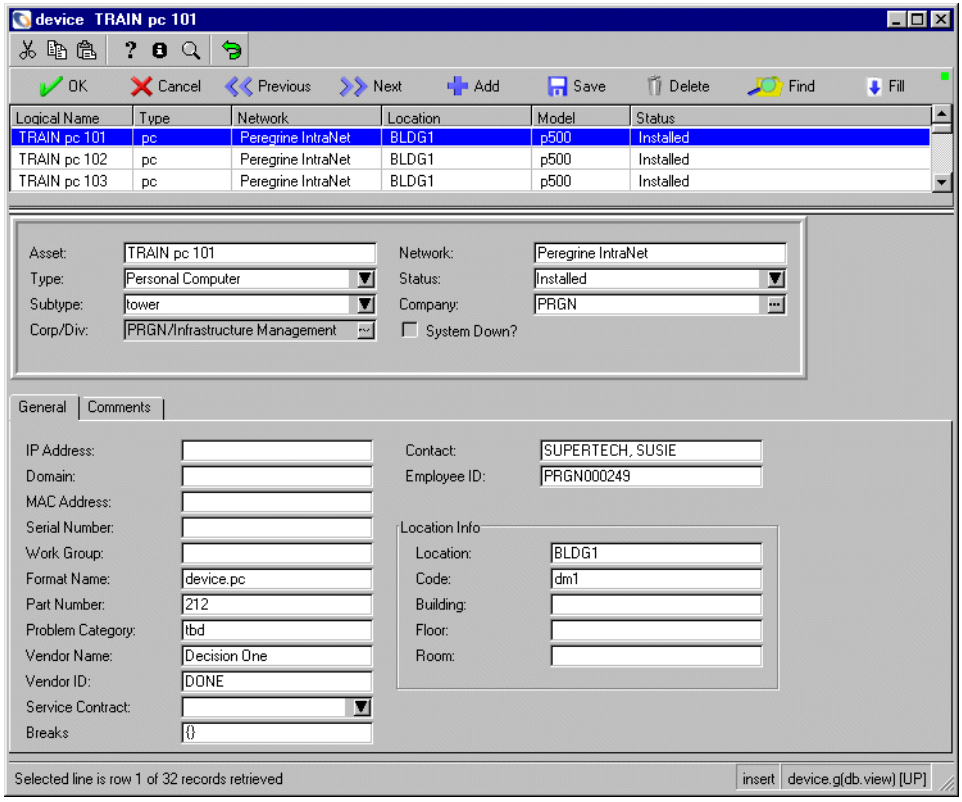
   The device form is displayed.
4. Type ¬C or ~C (depending on your system) in the **Vendor Name** field.

   Note: If equal to (=) was not entered in the query, the search assumes the query is *not starts with*.



5. Click **Search** or press Enter.

   Database Manager performs the *not starts with* search, and displays the record list of matching record(s), if any, using the device.qbe format.



6 Click on the record that you want to view.

Retrieving All Records in a Database

The following example demonstrates a *true* query. All records in the **device** file will match the search criteria.

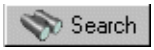
To retrieve all device records:

1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type **=device** in the **Form** field.

3 Click **Search** or press Enter.

The device form is displayed.



4 Click **Search** or press Enter with no values in any of the fields.

Database Manager performs the search and displays the record list of the records using the **device** format. In this example, the list contains all device records.

When Search is run from a BLANK form, a true query is performed, which returns a list of all records for that file.

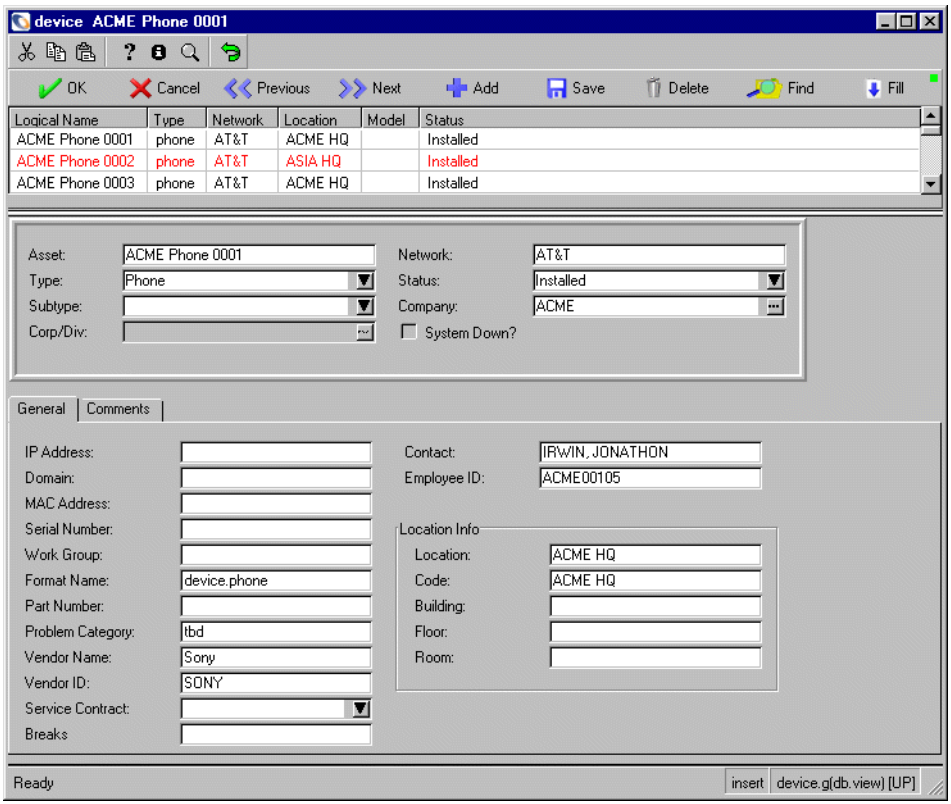


Figure 10-7: Results of a true query

Click on the record that you want to view.

Using More than One Field

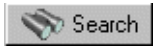
The following example demonstrates retrieving all device records with a logical.name (asset) starting with the letter H and with an IP address greater than 196.76.209.

Note: This search will not return records that begin with lower case h, if any were there, unless your database has been set up for case insensitive searching. To set up the database for case insensitive searching, see *Setting Case Mode for Searching the P4 File System* on page 23.

Important: Database Manager always forms QBE queries on scalar (non-arrayed) fields with the logical operator AND.

To retrieve records using more than one field:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =device in the **Form** field.
3. Click **Search** or press Enter.

   The device form is displayed.
4. Type the value H in the **Location** field.
5. Type the value >196.76.209 in the **IP Address** field.
6. 

Click **Search** or press Enter.

   When a partially-keyed query is executed, ServiceCenter searches the file for a specified interval or until the screen buffer is filled, then stops and displays the records retrieved so far.

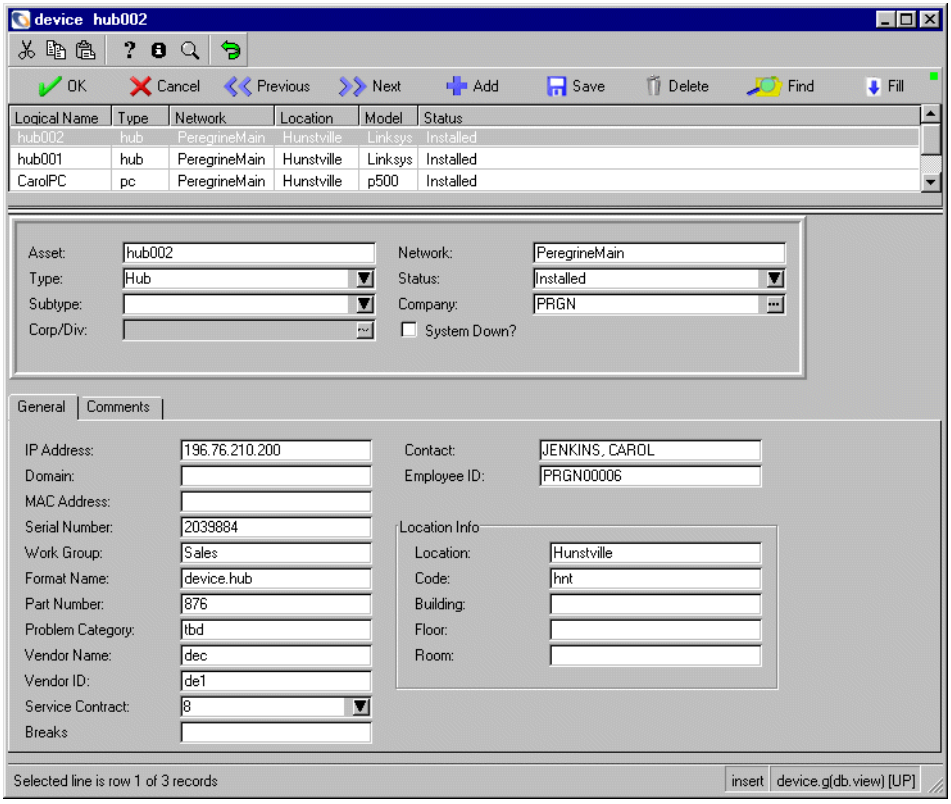


Figure 10-8: Results of a multiple field query

7 Click on the record that you want to view

Note: A warning screen, indicating an attempt to initiate a partially-keyed query may be displayed, depending on your operator and user profile, which in some cases restricts users from initiating inefficient queries.

If you receive a warning screen regarding an inefficient query:

1. Type in a time interval.
2. Click **Search** again to continue the search.

Database Manager performs the search and displays the record list of matching records, if any, using the device format. The list contains device records which met the search of both criteria.

Using Array Fields

The following example demonstrates retrieving all operator records with the execute capability words of *SysAdmin* or *problem management*.

Important: Database Manager always forms QBE queries on array fields with the logical operator OR.

To retrieve records using array fields:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =operator in the **Form** field.
3. Click **Search** or press Enter.

   A blank operator form is displayed.
4. Select the Startup tab to display the operator start-up parameters.
5. Type =SysAdmin in the first field of the **Execute Capabilities** array.
6. Type =problem management in the second field of the **Execute Capabilities** array.

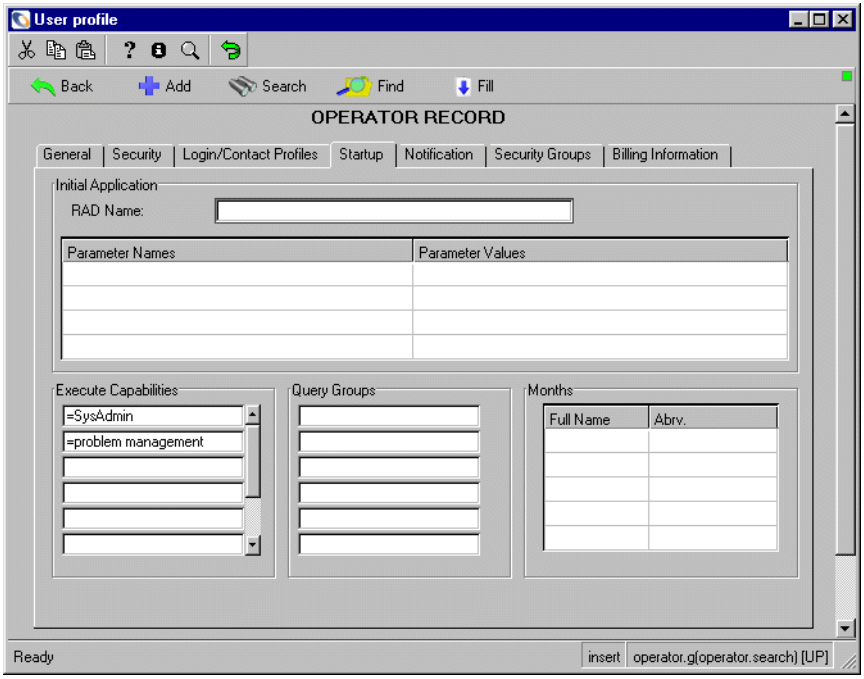
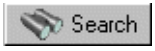


Figure 10-9: Operator Record Startup tab



7 Click **Search** or press Enter.

Database Manager performs the search and displays the record list of matching record(s), if any, using the operator.qbe format. In this example, the records found met the search of cap.exec="SysAdmin" or cap.exec="problem management".

Note: The position of an element (any of the input lines) within an array is irrelevant when it comes to queries. Even though the preceding query was formed with =SysAdmin in element 1 and =problem management in element 2, it retrieves operator records with either value in any position in the array.

The record list is displayed containing the records that match the search criteria. See Figure 10-10 on page 207.

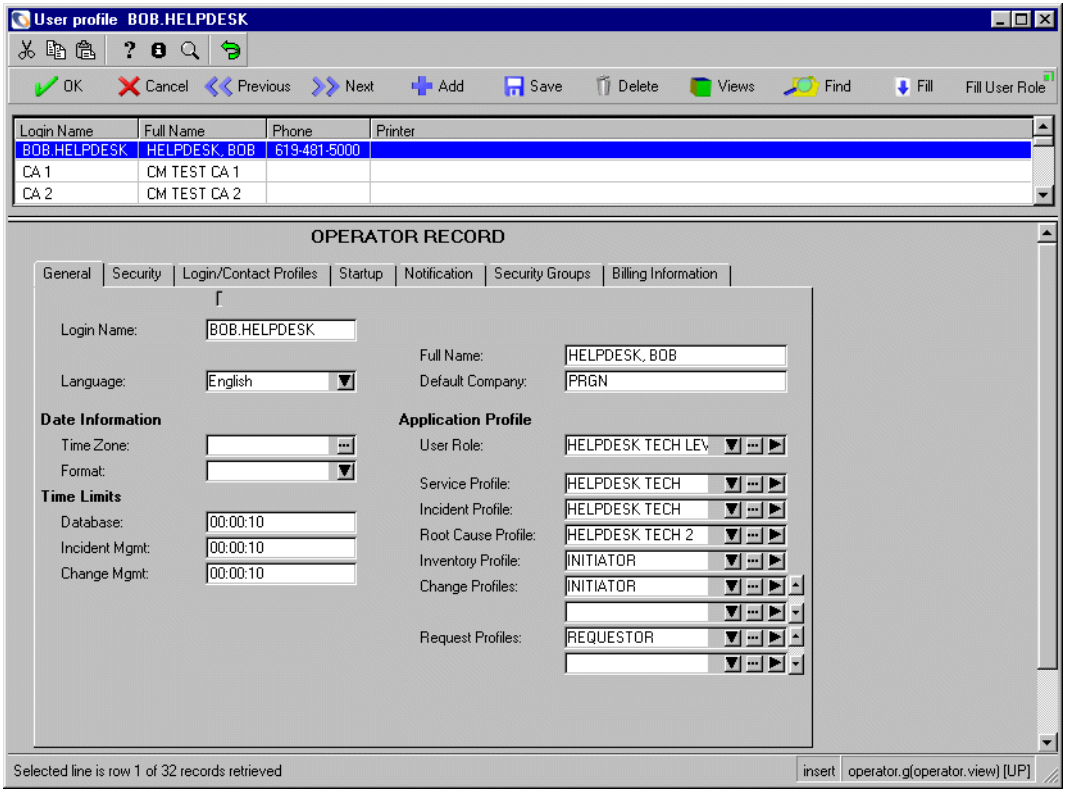


Figure 10-10: Operator record list

Retrieving Records via the Query Window

The Query Window method permits you to retrieve records by entering a query expression directly into the query pop-up window and pressing Enter. Database Manager searches for records that match your query and displays a list of matching records. If Database Manager displays a single record instead of a list, then the search criteria uniquely defined only one record.

The expression entered into the query window is a logical expression — one that has a value of either *true* or *false*. Its value is calculated for each record examined by the query. It retrieves only those records whose expressions evaluate to *true*.

Note: Access to the query window is controlled by one or more capability words. Refer to the ServiceCenter *System Administrator's Guide* for detailed information on capability words.

Accessing the Query Window

The query window is accessed by selecting **Advanced Search** from the ServiceCenter **Options** menu while a form or file is open in Database Manager.

The following example demonstrates how to access the query window from the operator record.

To access the query window:

1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type `=operator` in the **Form** field.

3 Click **Search** or press Enter.

The operator form is displayed.

4 Select **Options** > **Advanced Search** from the menu bar.

Several options are available, depending on the user's execution capabilities:

- If the user has query window access, the query window pops up immediately at the bottom of your screen.
- If the user has stored query access and stored queries have been defined, a list of them is displayed.
- If the user has access to both, the query window pops up and buttons defining query options are displayed on the main form.

The query window is shown below.

Figure 10-11: The Advanced Search Query window

Using the Query Window

The following example demonstrates retrieving all device records with logical name starting with the character value A.

To retrieve all device records using the query window:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =device in the **Form** field.
3. Click **Search** or press Enter.

   The device form is displayed.
4. Open the query window by selecting **Options** > **Advanced Search** from the menu bar. (For more information, see *Accessing the Query Window* on page 208.)

   A blank query.prompt format is displayed.
5. Type the query expression logical.name#"A".
6. Click **Search**.

Figure 10-12: The Advanced Search Query window with query

Database Manager performs the *starts with* search and displays the QBE list of matching record(s), if any, using the **device.qbe** format.

Notice all values listed under the **logical.name** heading start with A.

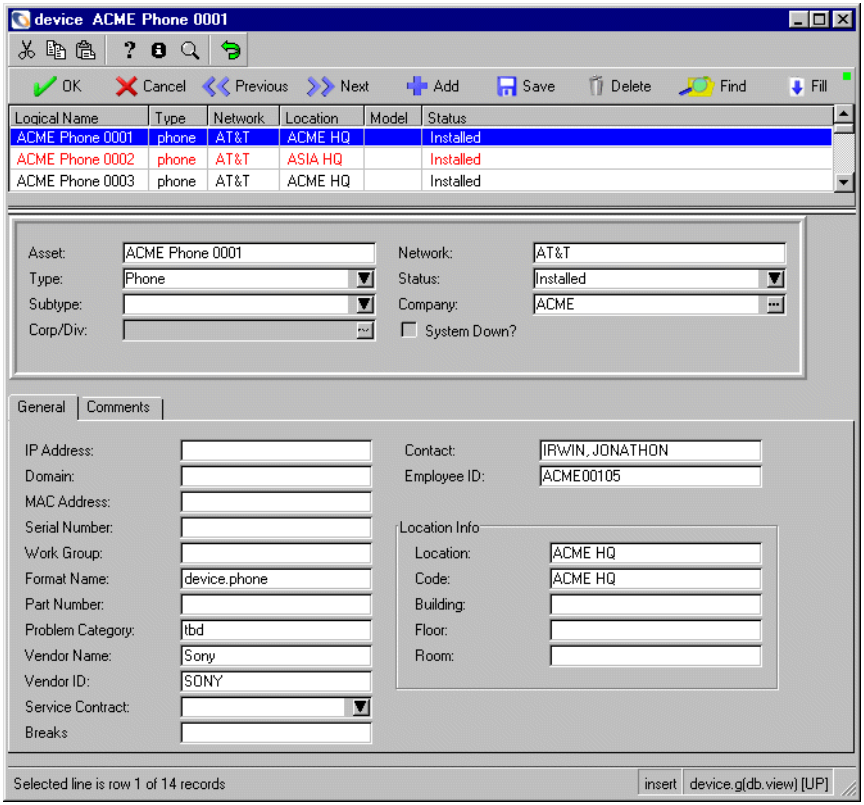


Figure 10-13: Results of Advanced Search query

7 Click on the record that you want to view.

Using a Simple Query Expression

The Query Window method is most useful when performing simple queries in situations where a QBE query cannot be performed, and when performing range queries. For example, the desired field exists in the database dictionary record of the file, but it is not displayed on the form, or the input field on the format is not long enough to contain the desired value.

The query.window application used in Database Manager, Incident Management, and Inventory Management allows specification of sort fields.

Important: To use this function, a user must have query.window (capability to open the Query Window) defined in the operator record.

The example below illustrates searching the operator record for a list of records with the application name menu.manager, sorted by full name and then by name.

To retrieve records with a simple query expression:

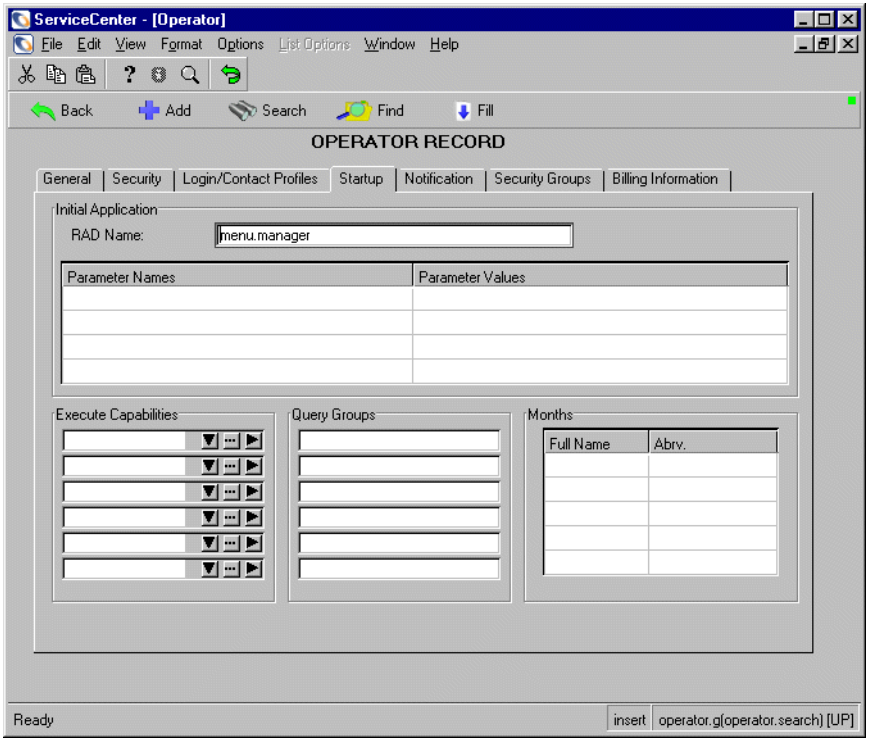
1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type the form name in the **Form** field. For this example =operator).

3 Click **Search** or press Enter.

The form is displayed.

4 Access the **Startup** tab.



5 Type in your query criteria. For this example, type menu.manager in the **Initial Application RAD Name** field.

6 Open the query window by selecting **Options** > **Advanced Search** from the menu bar. (For more information, see *Accessing the Query Window* on page 208.)

The query.prompt format is displayed with the query in the Query text box, For this example, application.name#"menu.manager" (application.name field input value starts with menu.manager).

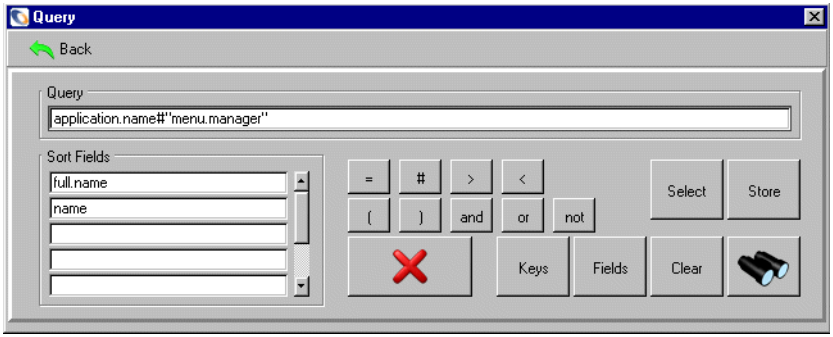


Figure 10-14: Operator Form with Advanced Search

7 Type in the sort criteria. For this example, type full.name as the first sort field and type name as the second sort field.

8 Click **Search**.

The list of records with the search criteria will be displayed, in the search order requested. In this example, application name menu.manager will be retrieved, sorted by full name and then by name.

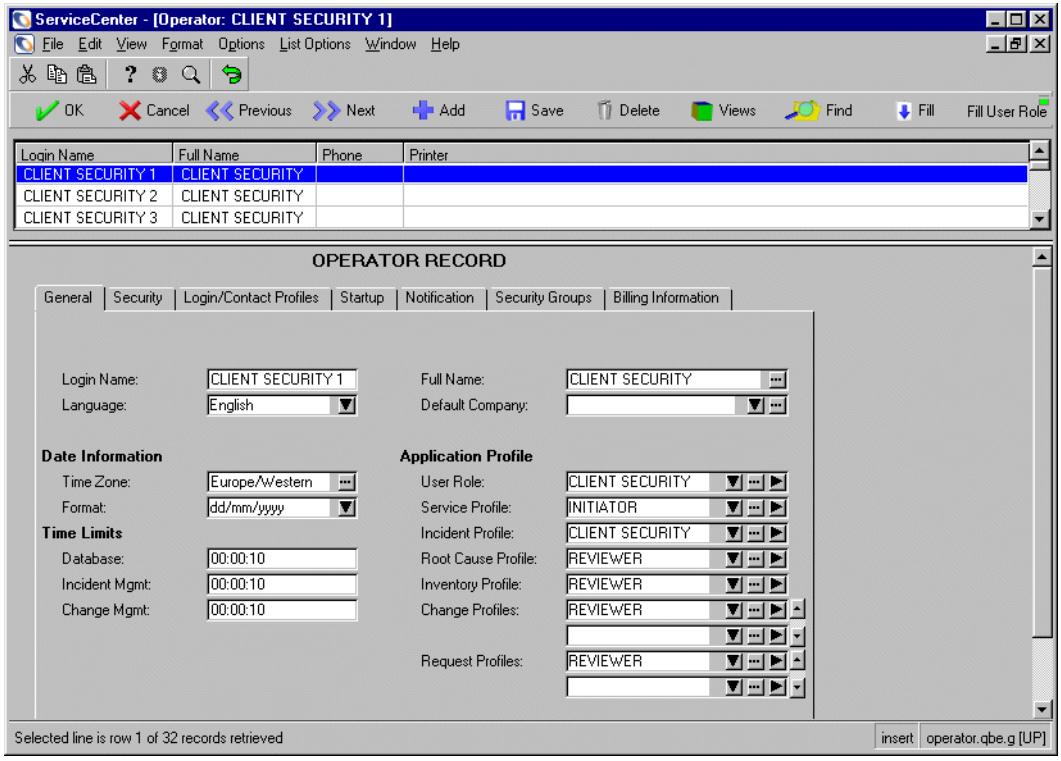


Figure 10-15: Results of Advanced Search with Simple query

9 Click on the record that you want to view.

Using Keys in a Search

Keys can be used to search and to create stored queries. This feature is particularly useful in ensuring fully-keyed queries.

The example below illustrates searching the operator record for a list of records with the application name menu.manager, using a fully keyed query.

To search using a key to ensure a fully keyed query:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =operator in the **Form** field.
3. Click **Search** or press Enter.
4. The operator form is displayed.
5. Access the **Startup** tab.
6. Type menu.manager in the **Initial Application RAD Name** field.
7. Open the query window by selecting **Options** > **Advanced Search** from the menu bar. (For more information, see *Accessing the Query Window* on page 208.)

   The query.prompt format is displayed with the query application.name#"menu.manager" (application.name field input value starts with menu.manager).
8. Click **Keys**.

   The Key Window (the keylist form) is displayed with a list of keys defined for the selected file. See Figure 10-16 on page 215.
9. Type the **Key Number** (3 for this example). Selecting 3 places the *application.name* key in the query.

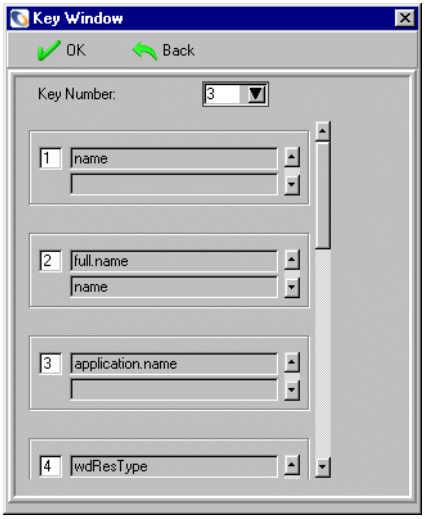


Figure 10-16: The Key Window

Note: Entering 2 in this example would pass the sort command on the same query as entered in *Using a Simple Query Expression* on page 211, step 7, returning full.name and name.

10 Click **OK**.

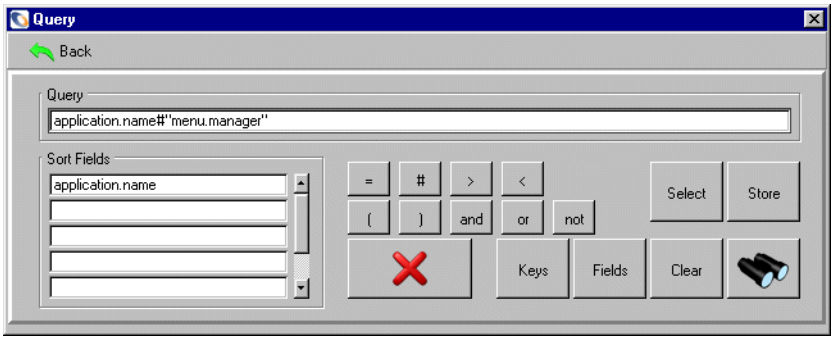


Figure 10-17: Search Window with Sort Field

You are returned to the query window. The Sort Fields field in the Query window change to agree with the key definition selected from the keylist form, application.name in this example.

11 Click **Search**.

The list of records with the application name menu.manager will be retrieved, sorted by application name.

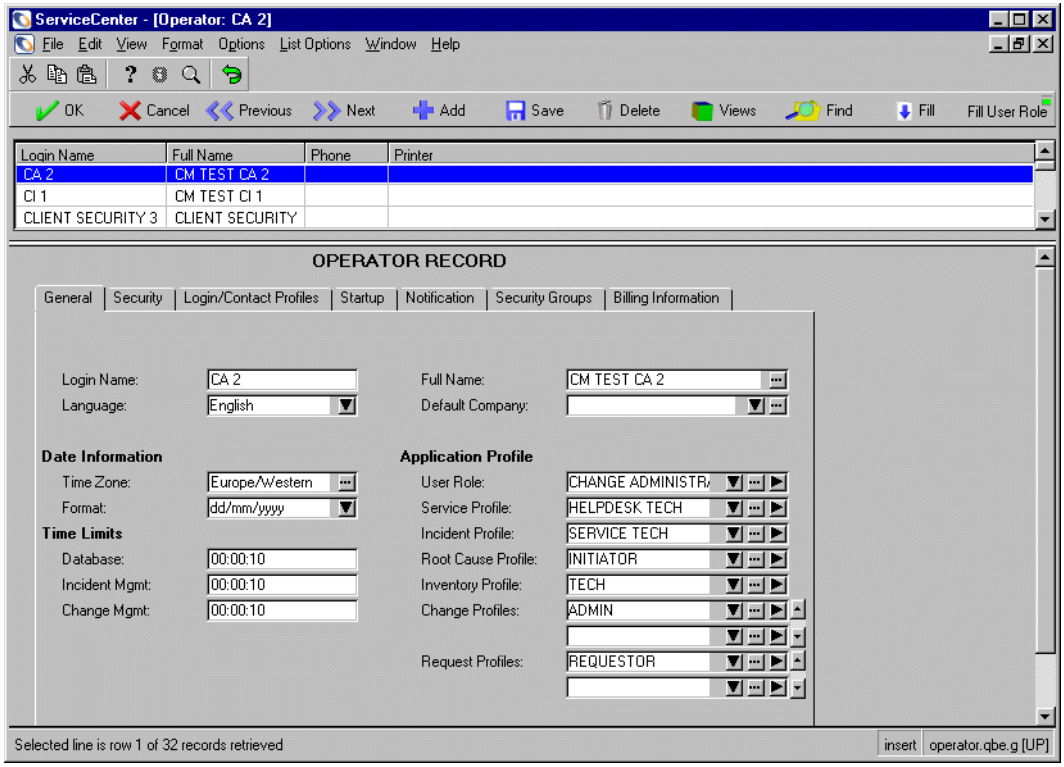


Figure 10-18: Results of Keyed query

Creating a Stored Query

The option of creating a stored query available to users with access privilege capability words of QueryAdmin or query.stored.mod in their operator profile.

The example below illustrates creating a stored query from a the operator record for a list of records with the application name menu.manager, sorted by a key.

To create a stored query:

1 Follow steps step 1 to step 10 from *Using Keys in a Search* on page 214.

You are returned to the query window.

2 Click the **Store** button on the main form to store this query directly in the querystored file. The query was built by entering information in fields in the form.

Note: The cursor remains busy (shows an hour-glass) when moved from the Advanced Query form to the menu bar.

3 Type in the appropriate **Access List** (**Query Group of Operator Name**) information. The ServiceCenter users or groups entered in this list are allowed to use this stored query. If the list is left blank, all users can use this query.

4 Type an appropriate name.

5 Type an appropriate description.

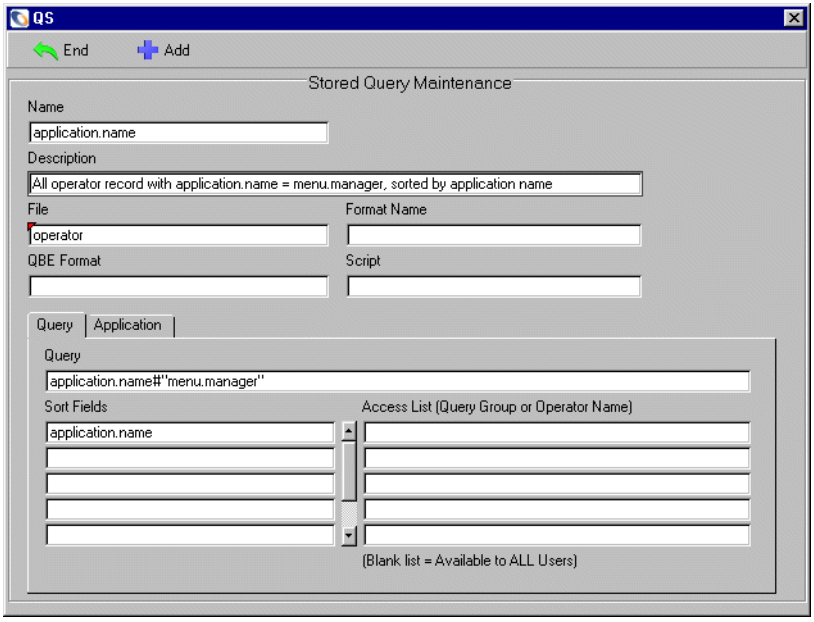


Figure 10-19: Stored Query Maintenance form

6 Click **Add** to add the query to the **querystored** file.

You are returned to the **query.prompt** format with the message: *Query added to querystored file* at the bottom of the screen.

7 At this point, you can click **Search** or press Enter to run the query you have created, or use one of the other options to modify or select another query.

To update a stored query:

1 Open the Query Stored list by clicking **Select** on the Query window. (See Figure 10-17 on page 216).

2 Double click on the query name to select it from the list. For this example, select operator.SysAdmin. See Figure 10-20 on page 219.

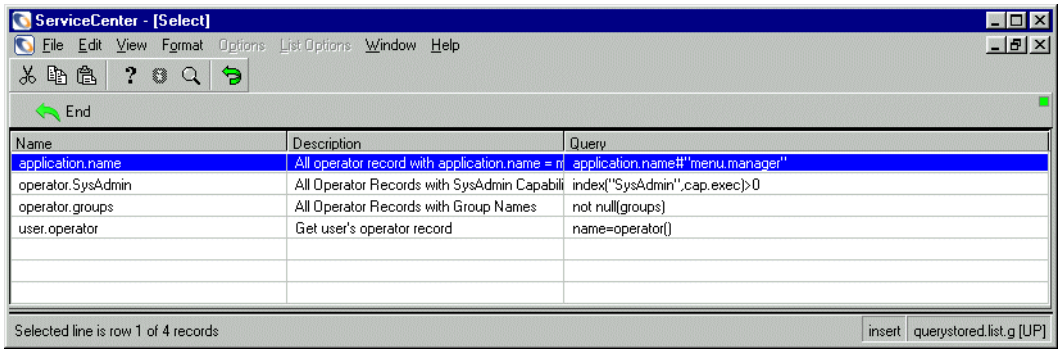


Figure 10-20: Stored Query list

The stored query record is displayed.

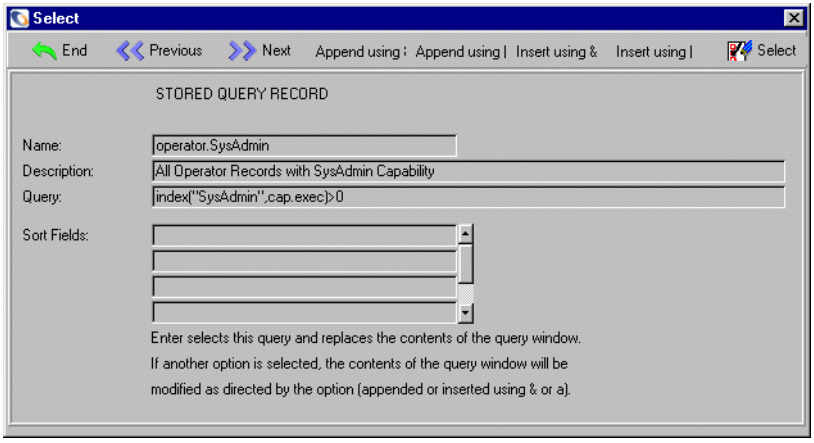


Figure 10-21: Stored Query Record

This form provides additional options that allow the user to ensure a fully-keyed query is established.

3 Click **select** from the toolbar to choose this query to run as it appears against the opened operator form.

4 You can also scroll through the records for the other stored query records by using **Next** and **Previous**.

5 The **Append** and **Insert** options, also displayed in the toolbar, provide you with the option of tailoring your query or using multiple stored queries at the same time. See *Stored Query Options* on page 220 for details.

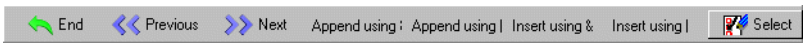


Figure 10-22: Stored Query Option buttons

Stored Query Options

The options available for a stored query are:

Option key	Function	Definition
F1	Append using &	This option adds another query to the end of your original query with the word AND separating the two queries.
F2	Append using |	This option adds another query to the end of the original query with the word OR separating the two queries.
F3	Insert using &	This option inserts another query at the beginning of the original query with the word AND separating the two queries.
F4	Insert using |	This option inserts another query at the beginning of the original query with the word OR separating the two queries.

Using Complex Query Expressions

OR/AND Statements

The Query Window method must be used to perform queries that involve combinations of logical operators that are not available using the QBE query method.

The following example demonstrates making a complex query, using logical operators, against the contacts file where the **Company** is either ACME or GENERICOM and the **Contact Name** starts with "B".

To retrieve records using the logical operator *or/and*:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank contacts form is displayed.

2 Select **Options** > **Advanced Search** from the menu bar.

3 Type the query expression (company="GENERICOM" or company="ACME") and contact.name#"B" in the query window as shown in Figure 10-23 on page 221.

4 Click **Search** or press Enter.

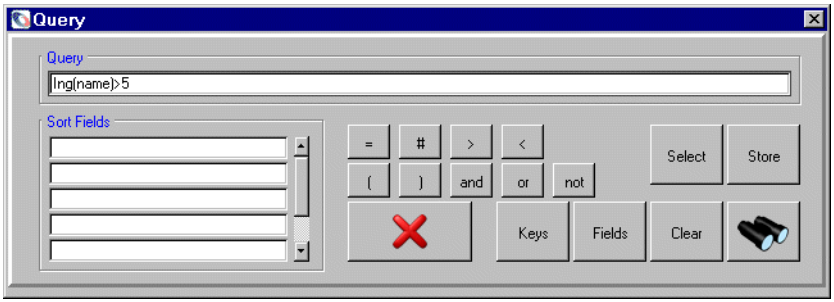


Figure 10-23: Search window with complex query expression

Note: Logical operators are executed in the following order: NOT, AND, OR. When the operators have equal precedence, they execute from left to right.

This query may result in a partially-keyed search.

a If so, type in a time limit for running the query.

b Click **Search**.

Database Manager performs the *starts with* search and displays the record list of matching records, if any, using the contacts.qbe format.

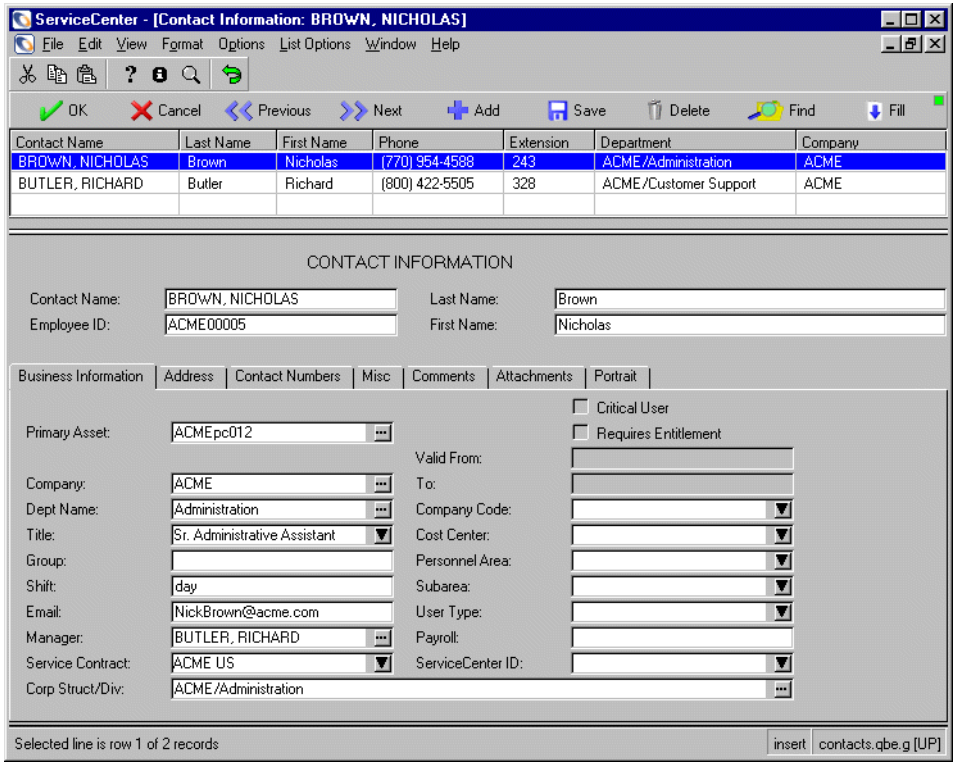


Figure 10-24: Results of "OR/AND" query in step 3

Only records for ACME are returned, because GENERICOM has no contacts that start with "B".

5 Select the contacts record that you want to view.

If the same expression is added without parentheses:

company="GENERICOM" or company="ACME" and contact.name#"B"

Since the AND operator takes precedence over OR, this query would retrieve records satisfying a condition very different from the above example:

company is "GENERICOM"

— Or —

company is "ACME" and **contact.name** starts with "B"

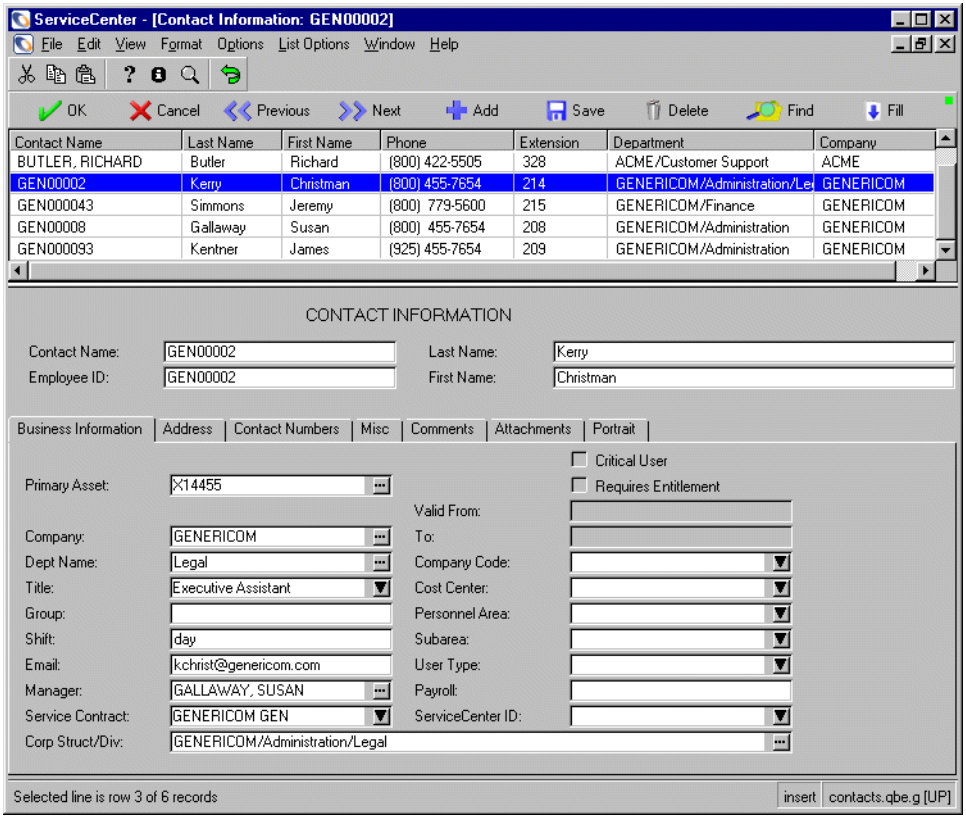


Figure 10-25: Results of "OR/AND" Query in step 5

Records with a value of "GENERICOM" are returned with this query, because it is less restricted that the query entered in step 3.

A range can be determined by using the logical operators between the relational operators. Allowable values are **AND** and **OR**. The relation must be spelled out. The range is defined by the conjunction of the *less than* and *greater than* operators.

A range includes everything between two values:
field>lesser.value and field<greater.value

For example:
open.time>'1/1/2003' & open.time< '1/5/2003')

Isin Statements

The following example demonstrates making a complex query against the contacts file where **Company** is either ACME or GENERICOM and the **Contact Name** starts with B using *isin*.

To retrieve records using the logical operator *isin*:

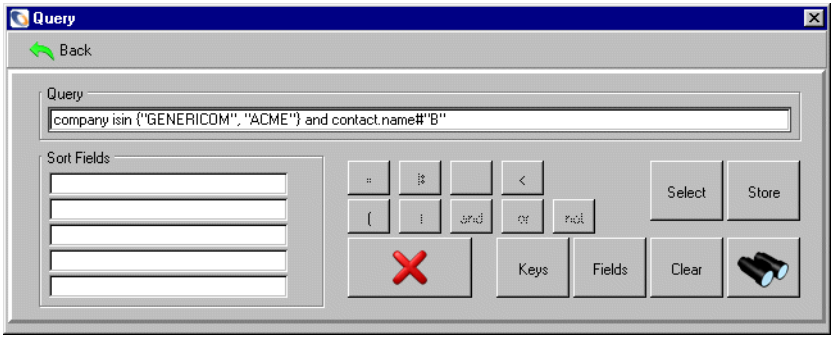
1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank contacts form is displayed.

2 Select **Options** > **Advanced Search** from the menu bar.

3 Type the query expression company isin {"GENERICOM", "ACME"} and contact.name#"B" in the Query text box.

Click **Search** or press Enter.



The database manager displays all records whose logical name starts with "B" and whose company is either **GENERICOM** or **ACME**. See Figure 10-24 on page 222. This query expression is equivalent to the first query in the entered in *OR/AND Statements* on page 220, step 3.

Not Statements

The following example demonstrates the retrieval of all contacts records with a manufacture value other than ibm when the "*not*" symbol is not available on the keyboard.

To retrieve records using the logical operator *not*:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank contacts form is displayed.

2 Select **Options** > **Advanced Search** from the menu bar.

3 Type the query expression not (company="GENERICOM") in the Query window. Click **Search** or press Enter.

Important: Leave a space between "not" and the left parenthesis " (".

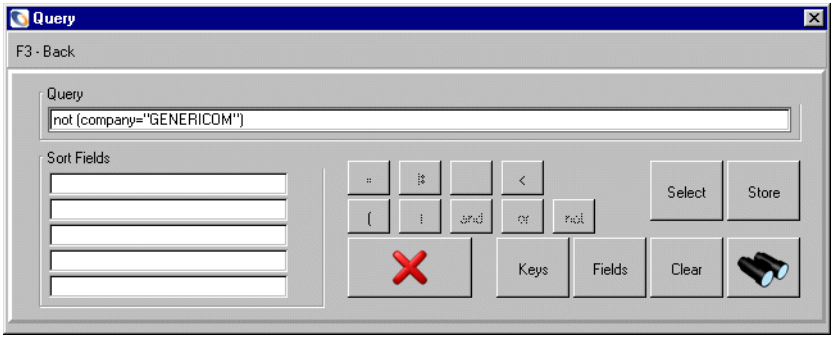


Figure 10-26: Search window Query using "not"

Database Manager performs the *not equal to* search and displays the record list of matching record(s), if any, using the contacts.qbe form.

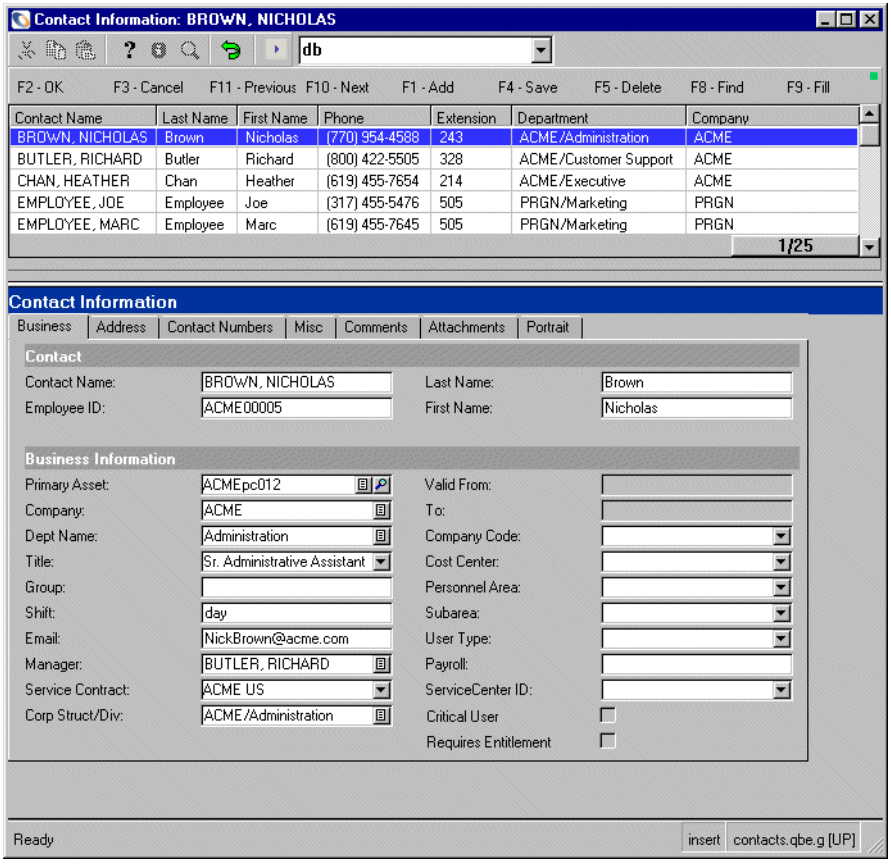


Figure 10-27: Results of "not" query

4 Click on the record that you want to view from the list.

Like Queries

Like queries can be used for identification of characters anywhere in a field.

Note: You can use a *like* query in combination with a function.

To retrieve records containing a particular character string:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type `=device` in the **Form** field.
3. Click **Search** or press Enter.

   The device form is displayed. Do not type any additional information into any of the fields on the form.
4. Select **Options** > **Advanced Search** from the menu bar. The Query window is opened.
5. Type the query expression `logical.name like "*001"` in the Query field. Click **Search** or press Enter.

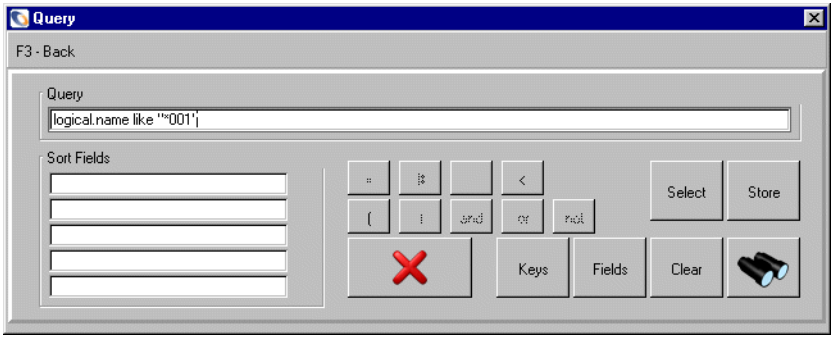


Figure 10-28: Query window — searching for a character string

Database Manager performs the search and displays the record list of matching records, if any, using the `device.qbe` format.

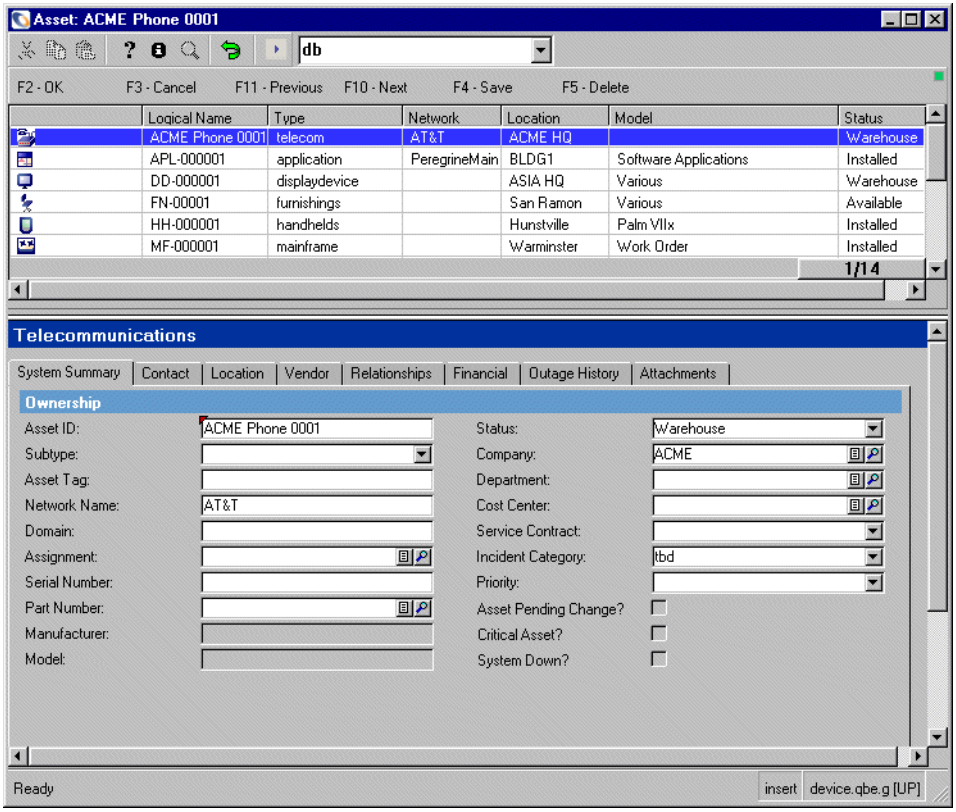


Figure 10-29: Results of "like" query

6 Click on the record that you want to view from the list.

Using Functions in a Query

You must use the query window to perform queries that involve functions. Refer to *System Language* in the *System Tailoring Guide* for a list and description of available functions.

Possible use of three sample functions is described here.

- index — *Using the index function in a like query* on page 229.
- tod — *Using the tod function in a >/< query* on page 231.
- lng — *Using the lng function in a query* on page 233.

For a complete description of theses functions, and the other functions available, see the *System Language* chapter of the *System Tailoring* guide.

Using the *index* function in a *like* query

The following example demonstrates retrieving records with the character string ACME occurring anywhere in the device name, and that also end in 002. We will use the index function and the like query to do this.

Wildcards are only valid in *like* queries. If you type in a query using a function the wildcard will not work as a wildcard but instead will be read as a literal character.

- logical.name like "*002" and index("ACME", logical.name)>0 is the correct query.
- index("ACME*002", logical name)>0 would not work, because the index function would search for the asterisk character literally and not as a wildcard.

For more information on like queries, see *Like Queries* on page 227.

To retrieve records containing a particular character string:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =device in the **Form** field. Click **Search** or press Enter.

   The device form is displayed. Do not type any additional information into any of the fields on the form.
3. Select **Options** > **Advanced Search** from the menu bar.
4. Type the query expression logical.name like "*002" and index("ACME", logical.name)>0 in the **Query** field. Click **Search** or press Enter.

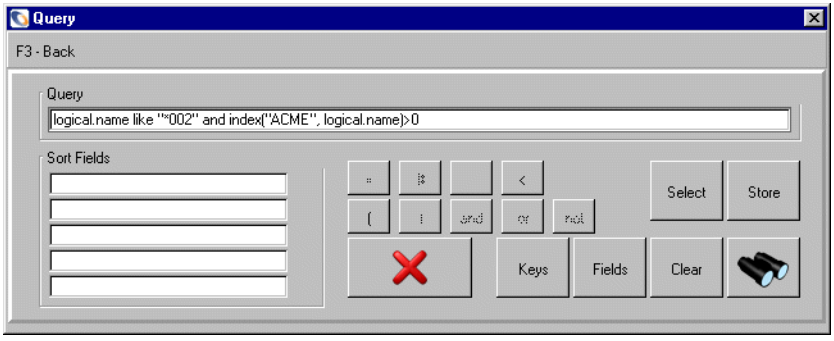


Figure 10-30: Query window, querying for logical.name containing "ACME" and ending in 002

Note: If you were to use `=0` instead of `>0` at the end of the query, the query would return the records that did not have `ACME` in the name.

Database Manager performs the search and displays the record list of matching records, if any, using the `device.qbe` format.

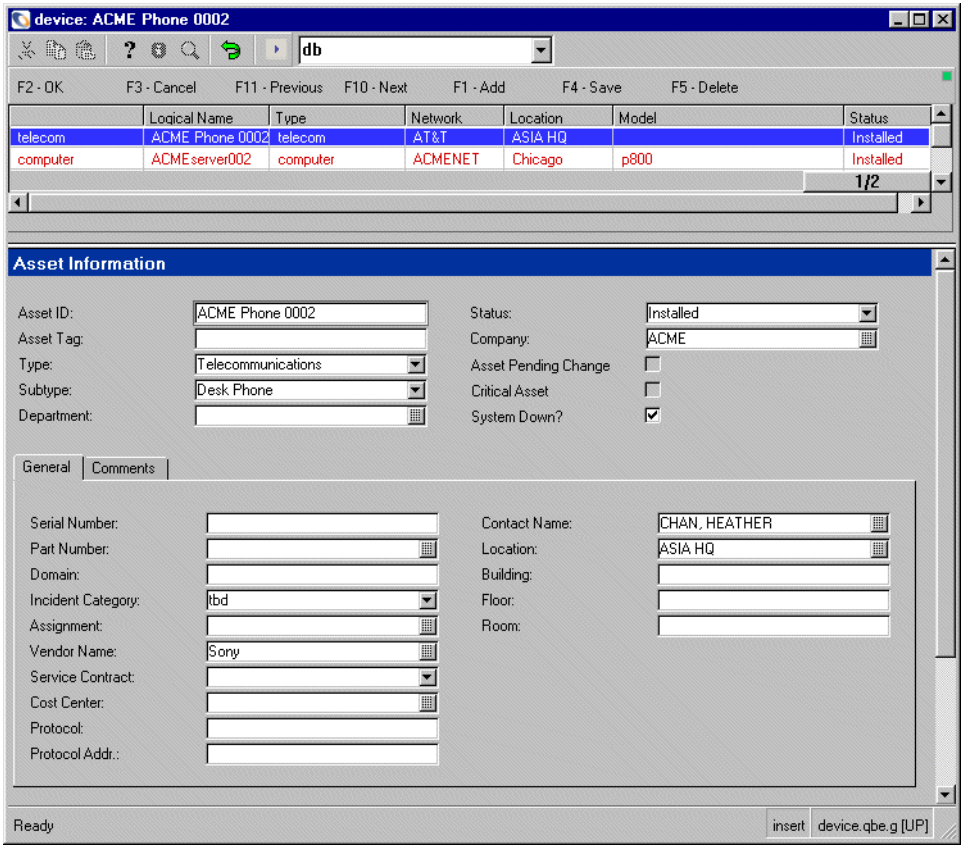


Figure 10-31: Results of *"like"* query using index function

5 Click on the record that you want to view from the list.

Using the *tod* function in a >/< query

The following example demonstrates retrieving all operator records with passwords updated within the past 100 days. We will use the >/< relational operators and the *tod* function to do this.

To retrieve records using the *greater than/less than* (>/<) function:

1 Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Type =operator in the **Form** field.

3 Click **Search** or press Enter.

The operator form is displayed. Do not type any additional information into any of the fields on the form.

4 Select **Options** > **Advanced Search** from the menu bar.

5 Type the query password.date>tod() - '100 00:00' in the **Query** window. Click **Search** or press Enter.

Note: The same query could be done in the opposite direction for all operators who have not change their password within the last 100 days by using the query password.date<tod() - '100 00:00' instead.

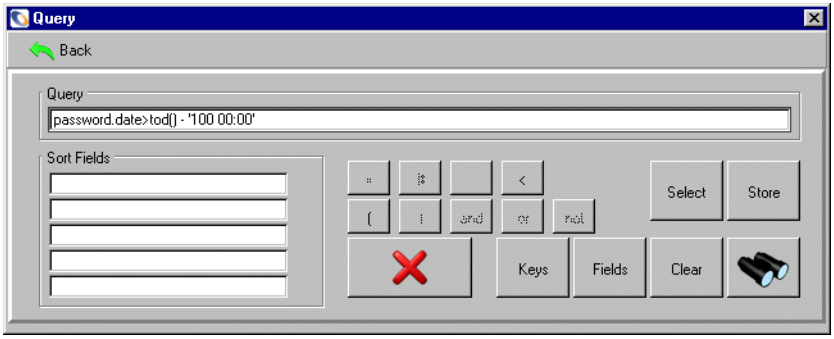


Figure 10-32: Query window specifying password updated within 100 days

Database Manager performs the search and displays the record list of matching record(s), if any, using the **operator.qbe** format.

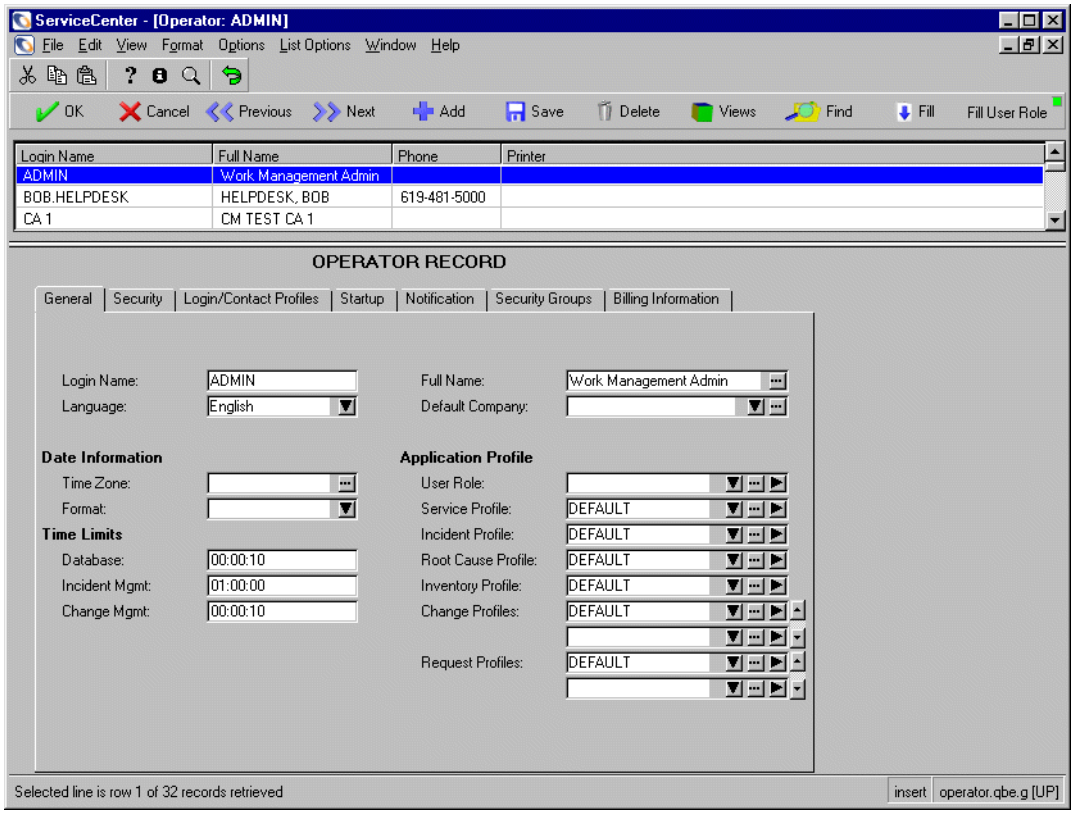


Figure 10-33: Results of >< query using tod function

6 Click on the record that you want to view.

Using the lng function in a query

The following example demonstrates retrieving all operator file records with names longer than 5 characters. The lng function is used to specify length of character strings in a query.

To retrieve records with names of a specified length:

1. Access the Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Type =operator in the **Form** field.
3. Click **Search** or press Enter.

   The operator form is displayed. Do not enter any additional information into any of the fields on the form.
4. Select **Options** > **Advanced Search** from the menu bar.
5. Type the query lng(name)>5 in the query window (That is a lowercase "L" at the beginning of the query, not an upper case "I".)
6. Click **Search** or press Enter.

Note: "lng(name)>5" is equivalent to "name like "??????*".

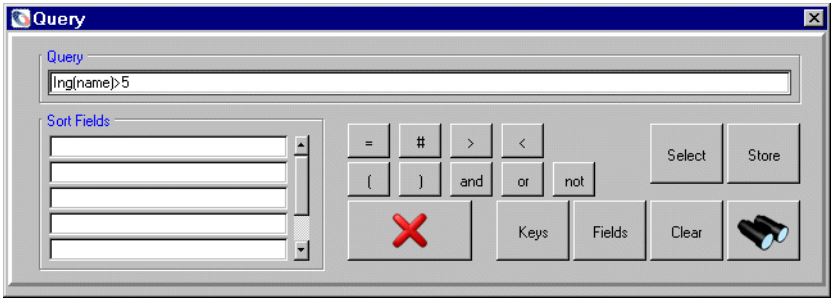


Figure 10-34: Query window specifying character string length >5

Database Manager performs the search and displays the record list of matching record(s), if any, using the **operator.qbe** format.

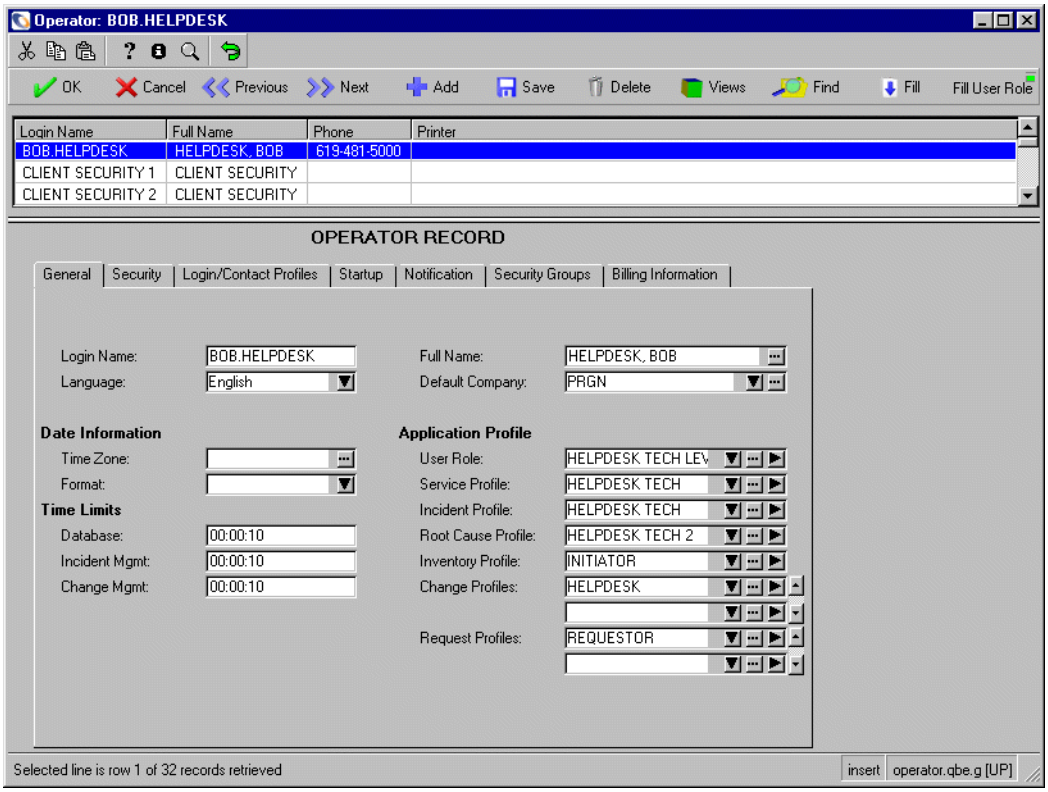


Figure 10-35: Results of string length query

7 Click on the record that you want to view from the list.

Performing IR Expert Queries

To run a more intelligent query against the database, a query can be more accurately defined within Database Manager by using the IR Expert utility (Information Retrieval engine), via the IR Query option.

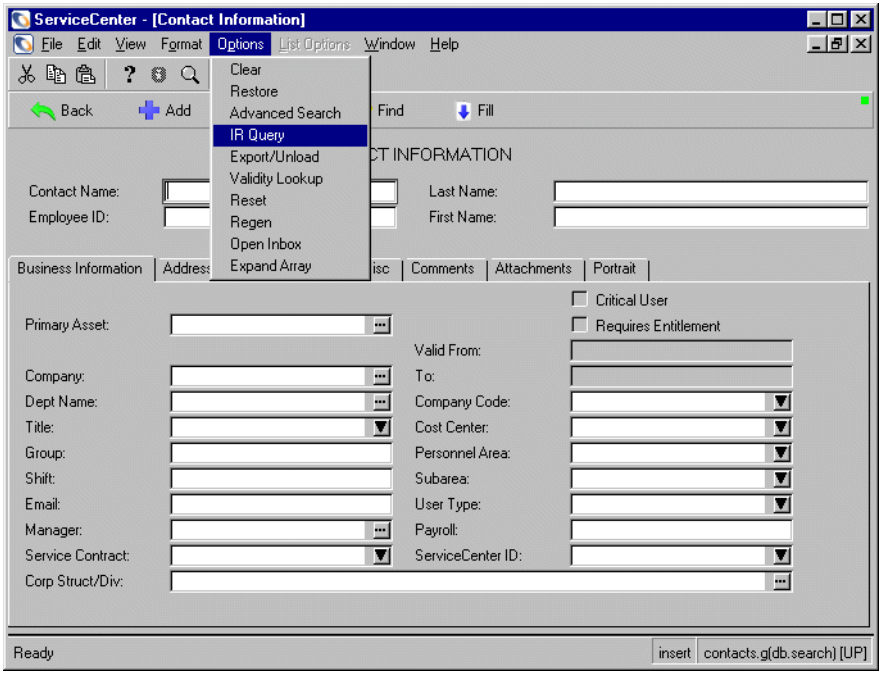
For more information on IR Expert, refer to *IR Expert* on page 319, and to the IR Expert chapter in the ServiceCenter User Guide.

Note: This option is not available for all forms.

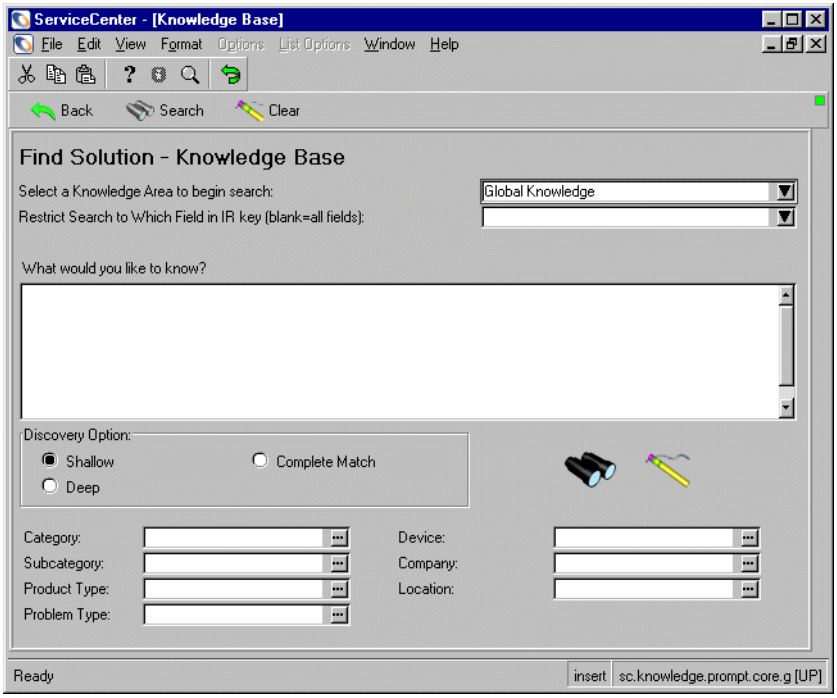
To exercise IR Expert to pass a query:

1 Open the **contacts** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank **contacts** form is displayed.



2 Select **IR Query** from the **Options** menu.



3 Type in your search criteria and click **Search**.

4 Click **Back** or **Exit** to return to the original form.

11 CHAPTER Single Record Functions

This chapter was designed to aid ServiceCenter system and database administrators in performing including add, update, delete, print functions on individual records within a database.

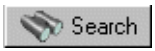
Topics in this chapter include:

Adding a Record

The following example demonstrates adding a new record, *Bob Hoskins*, to the **contacts** file.

To add a record:

1 Open the **contacts** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

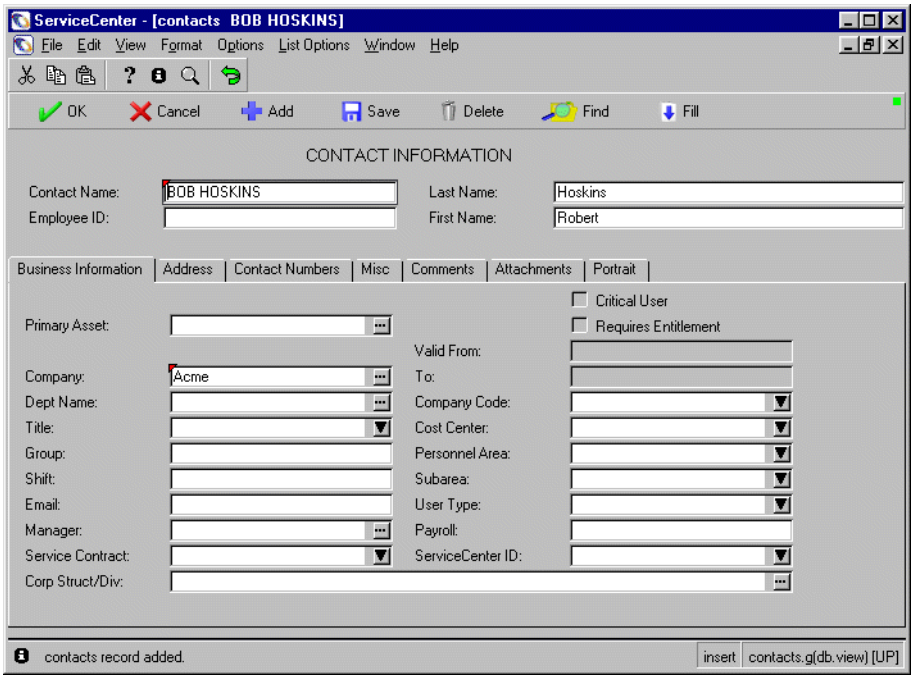


2 Select the **contacts** file from the record list and click **Search** or press Enter.

3 Enter the following required values on the **Business Information** tab, and other values as needed.

Field	Record Value
Contact Name	HOSKINS,BOB
Company	Acme

4 Click **Add**.



Database Manager adds the record, retains the input from the screen displayed and responds with the following message: *Contact Information record added.*

Duplicating an Existing Record

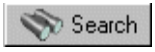
The following example demonstrates adding a new record to the contacts table that is a near duplicate of an existing record. For this example, all information except the Primary asset and Contact Name will be the same.

To duplicate an existing record:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Select the contacts file from the record list and press Enter.

A blank contacts file is displayed.



3 Click **Search** or press Enter. Select BUTLER, RICHARD from the returned list.

4 Select the **Business Information** tab.

5 Enter a new name in the **Contact Name** field.

6 Enter a new Primary asset for this record.

7 Delete the Employee ID. You can either add your own at this point or leave this field blank.

8 Click **Add**.

9 The new contact file is added to the database.

Database Manager adds the record, retains the input from the screen displayed and responds with the following message: *Contact Information record added.*

Updating an Existing Record

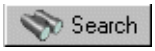
The following example demonstrates updating an existing contacts record for Heather Chan who has a Primary asset of *Acmepc014* and changing the value to *Acmepc012*.

To update an existing record:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Select the **contacts** file from the record list and press Enter.

A blank **contacts** file is displayed.



3 Click **Search** or press Enter. Select CHAN, HEATHER from the returned list.

4 Select the **Business Information** tab.

5 Replace the current value in the **model** field with Acmepc012.

6 Click **Save**.

Database Manager saves the record and responds with the following message: *Contact Information record updated.*

Deleting a Record

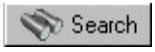
The following example demonstrates deleting the contact record we added for Richard Butler (*Adding a Record* on page 238.)

To delete a record:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Select the **contacts** file from the record list and press Enter.

A blank **contacts** file is displayed.



3 Query on the existing record by entering BUTLER, RICHARD in the **Contact Name** field, and click **Search** or press Enter.

4 Select the record from the returned list.

5 Click **Delete**.

A prompt is displayed asking you to confirm the action.

6 Click **Yes** to continue with the record deletion.

The following message is displayed in the status bar: *Contact Information record deleted.*

Printing a Record

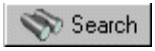
The following example demonstrates printing an existing **contacts** record for Richard Butler.

To print a record:

1 Open the **contacts** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

2 Select the **contacts** file from the record list and press Enter.

A blank **contacts** file is displayed.



3 Click **Search** or press Enter. Select Richard Butler from the returned list.

4 Select **Options** > **Print** from the menu bar.

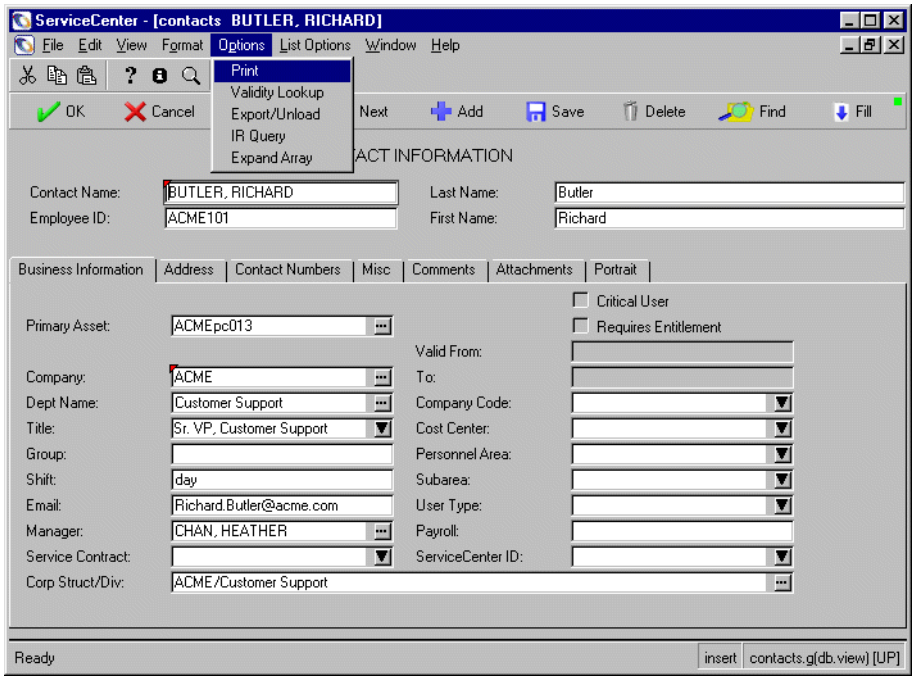


Figure 11-1: Printing a record

Database Manager responds with a print prompt message if your Printer Settings specify you are using the CLIENT printer.

If you are using the SERVER printer you may be prompted to select the specific printer to use. A spooling message is then presented at the bottom of

the screen, e.g., *Rpt spooled and sched as no: nnnnn (ServiceCenter Print Job)*. See the *Printer Setup* section in the *System Administrator's Guide* for information on using CLIENT or SERVER printers.

The entire record is then printed on the printer defined in the operator record for the current operator.

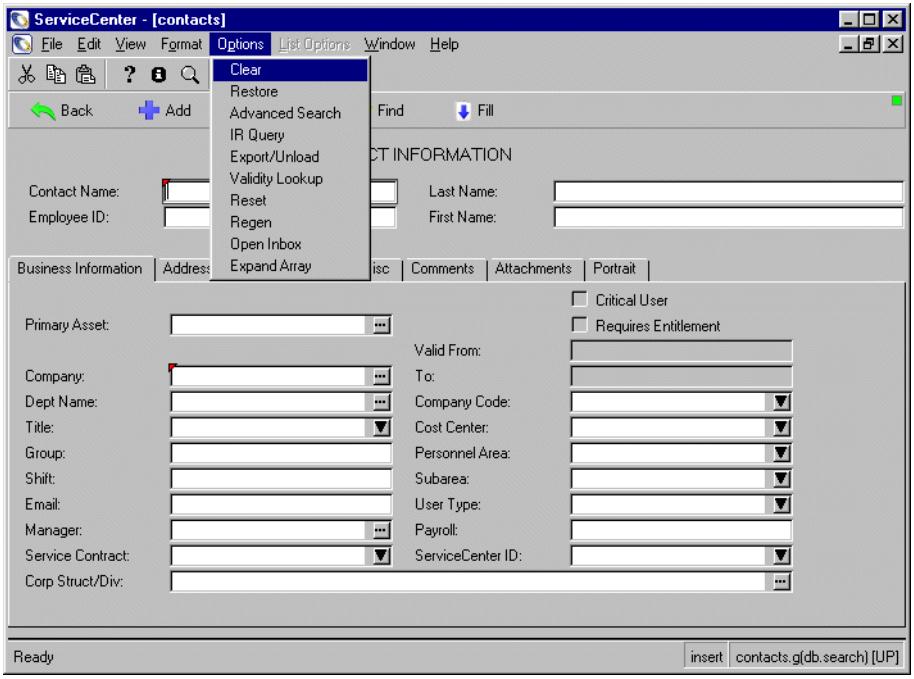
If Active Notes in enabled, the following message is displayed: *Report* <nnn> *printed; (ServiceCenter Print Job)*.

Clearing an Initial Record

If you need to redo an incorrect record or need otherwise to clear all data from an initial record form, this data can be quickly and completely removed with a Clear command.

To clear all data from an initial screen:

1. Open the form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Select the record from the record list.
3. Pull down the **Options** menu and select **Clear**.



Advanced Operations

Recovering from Record/Key Conflicts

The following table lists possible record/key conflict errors, a description of each one, and recovery actions.

Error	Description	Recovery
Record contains invalid duplicate key.	Occurs during Add or Update operations. The **dbdict** record for the database defines at least one key of the type no duplicates or unique. The record being added or updated contains values for these keys that are already used by another record in the database.	Either rename the record or select the existing record and update it. If duplicates should be accommodated on the database, the keys must be changed within the database dictionary record for this file to allow duplicates in the file for the field(s) in question. Once the dictionary has been changed, you may add or update the record.
Record contains invalid null key.	Occurs on add and update functions when a key is defined in the database dictionary as no nulls or unique and that key for the record being added or updated is null (blank). (See *Key Definitions* on page 21.)	Enter data into the field(s) defined as no nulls or unique keys and add or update the record. If nulls should be accommodated on the database, the keys must be changed within the database dictionary record for this file to allow nulls in the file for the field(s) in question. Once the dictionary has been changed, you can add or update the record.
This record has been changed since you selected it.	Occurs on update and delete functions when the selected record is no longer current, i.e., the record was updated by another task after you selected it and before you attempted the update or deletion.	Reselect the record and proceed with the update or deletion.
The record has already been deleted.	Occurs on update and delete functions when the selected record has been deleted, i.e., the record was deleted by another task after you selected it and before you attempted the update or deletion.	If caused by a delete function, the record has already been deleted and no further action is needed. If caused by an update, the record can be re-added by pressing Add. The record will be added just as it appears on your screen.

Record Level Options

Several advanced user options exist once a record has been selected. These options are available from the **Options** menu.

Expand Array — expands an array field. It opens a window and expands the array, enabling line editing options. This option is most useful in non-GUI versions of ServiceCenter.

To expand an array:

1 Locate a field to expand (**comments** in the **contacts** form is used in this example).

Note: The **Expand Array** option only works with array fields. Attempting to expand all other fields will return an error message. Use the **Magnifying Glass** button to expand non-array fields.

2 Place the cursor in the array field.

3 Select **Options** > **Expand Array** from the menu bar.

A new form is displayed with an extended version of the selected field.

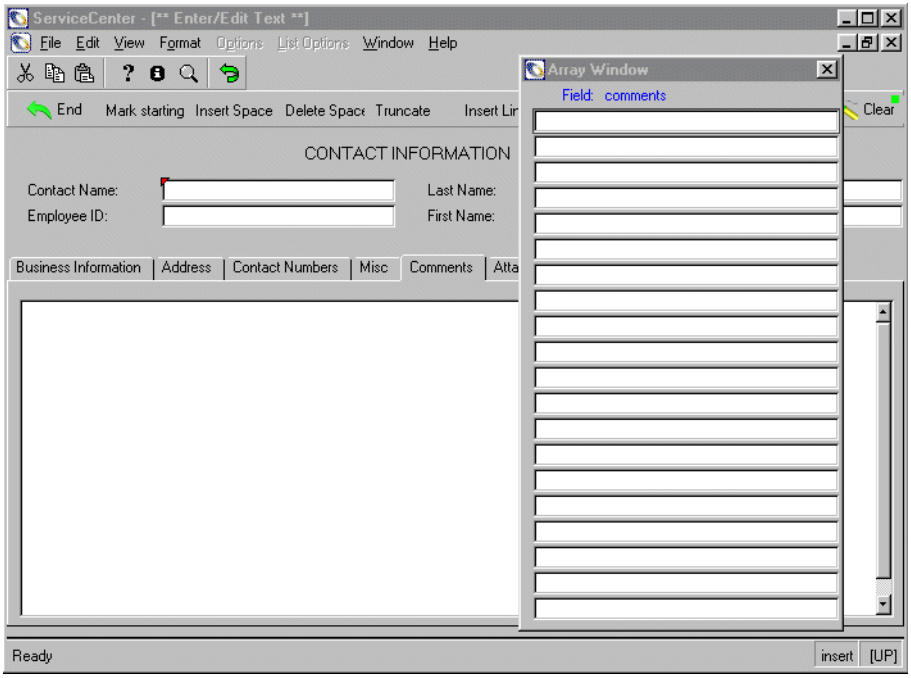


Figure 11-2: Expanded Array form

4 The user can now make any necessary additions to the data list or edit data relevant to the particular field.

Note: When in an application, the **Magnifying Glass** can also be used to expand array and scalar (non-array) field alike. This button expands any field (array or scalar), and is more useful with the multi-line text box construction of the GUI versions of ServiceCenter.

Table 11-1: Expand Array Options

Option	Purpose
End	checks for differences and prompts for confirmation.
Mark starting line	used to mark the first line of a block to be moved, copied, or deleted; re-displays window to mark last line, then enables move, copy to the line the cursor is on, or delete; move in this case removes the original lines.
Insert space	inserts a space at the start of the line the cursor is on.
Delete space	deletes the first space on the line the cursor is on.
Truncate	removes all line values below the line the cursor is on.
Insert line	inserts a blank line above the line the cursor is on.
Delete line	removes the line the cursor is on from the array.
Copy	selects the line the cursor is on and prompts to insert or
Move	same as Copy, except it removes the original value from the selected line.
Clear	removes all lines in array.

Format Control Settings

The options and contents of the **Options** menu in this and all applications are determined by a combination of the RAD application currently executing, Format Control, Additional Options, (only in Database Manager and selected other applications) and Display settings. Options can be created to allow for extended user capabilities and data gathering, including additional views and find/fill functions.

CHAPTER 12

Multiple-Record Functions

This chapter was designed to aid ServiceCenter system and database administrators in performing add, update, delete, and print functions on multiple records within a database.

It is divided into the following sections:

Overview

The Mass functions (Mass Add, Mass Update, Mass Delete) become available whenever a query produces a record list, i.e., the desired function acts upon the records in the QBE list.

Note: The Mass functions may require use of the Administrative Mode check box when starting Database Manager.

The Record List

Depending on whether you have the record list option active, through the **Record list** option in the **View** menu, you will either receive a QBE list of records prior to viewing the contents of a record, or the records returned will be listed in a table above the format displaying the contents of the first record in the list.

- If the **Record list** option in the **View** menu is not active, the Mass functions are available only from the QBE in the **Options** menu.
- If the **Record list** option in the **View** menu is active, the Mass functions are available at anytime while viewing records in the **List Options** menu.

Adding Multiple Records

Having accessed a list of records from a ServiceCenter file, the **Mass Add** function allows you to add a block of records to the database. The new records are exact duplicates of those in the original record (or QBE) list. Exact duplicates can only be added to a database which has keys defined as *nulls & duplicates* or *no nulls* in the database dictionary, otherwise an invalid duplicate key error message is received for every record. (For a list of key types and their definitions, see *Key Definitions* on page 21.)

To avoid this error when executing a Mass Add against a file with *unique* or *no duplicates* type keys, processing statements should be executed during the Mass Add to manipulate the contents of the new records. Mass Add processing statements reference $file to access fields on records in the QBE list. For example, name in $file or 1 in action in $file. Refer to the System Language section in the *System Tailoring Guide* for examples of processing statements.

Note: Format Control, triggers and macros, if present for the current form, will be executed for ADD=true processing. Refer to the *System Tailoring Guide* for detailed information.

Mass Adding Records Using a Literal Value

The following example demonstrates duplicating all contact records for Genericom, retaining the same information in each record with the exception of changing the **Location** to New York.

Since Contact Name (contact.name) is a *unique* key and Employee ID (user.id) is a *no duplicates* key in the contacts file, unique contact.name and user.id values must be assigned to the new records. (See *Key Definitions* on page 21.) For this example, the value -NY is appended to the contact.name and user.id values to maintain uniqueness.

To add records using a literal value:

1 Open the Contact Information (contacts) form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the **Administration Mode** check box.

A blank contacts form is displayed.

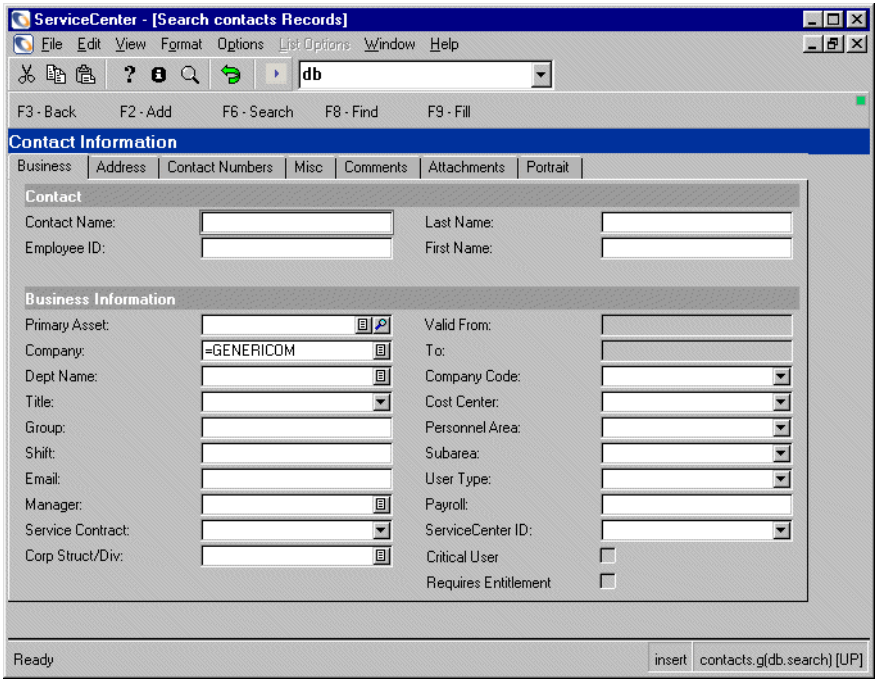
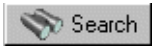


Figure 12-1: Querying for records to duplicate

2 Enter the desired **Company** value. For this example, select GENERICOM from the **Company** list.



3 Click **Search** or press Enter.

Database Manager performs the *equal to* search and displays the record list of matching record(s), if any. If the query produces no matching records, a message is displayed.

Note: To show the Mass function buttons, use a query that will return more than one record.

4 Select one of the listed records.

5 If **Record list** is turned on (See *The Record List* on page 248.), select **List Options** > **Mass Add** from the menu bar to start the process of duplicating the records shown in the list.

— or —

If **Record List** is turned off, equivalent options are displayed in the **Options** > **Add** menu of the search list.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Add** menu of the Search list.

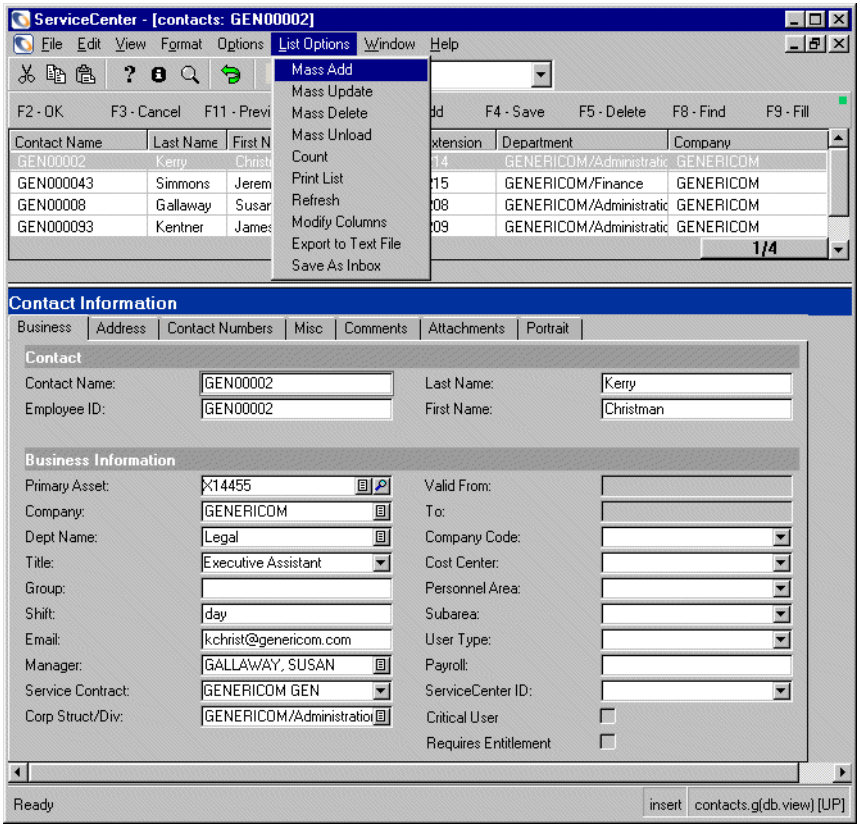


Figure 12-2: Calling Mass Add action

Database Manager displays the Mass Add/Update Instruction screen.

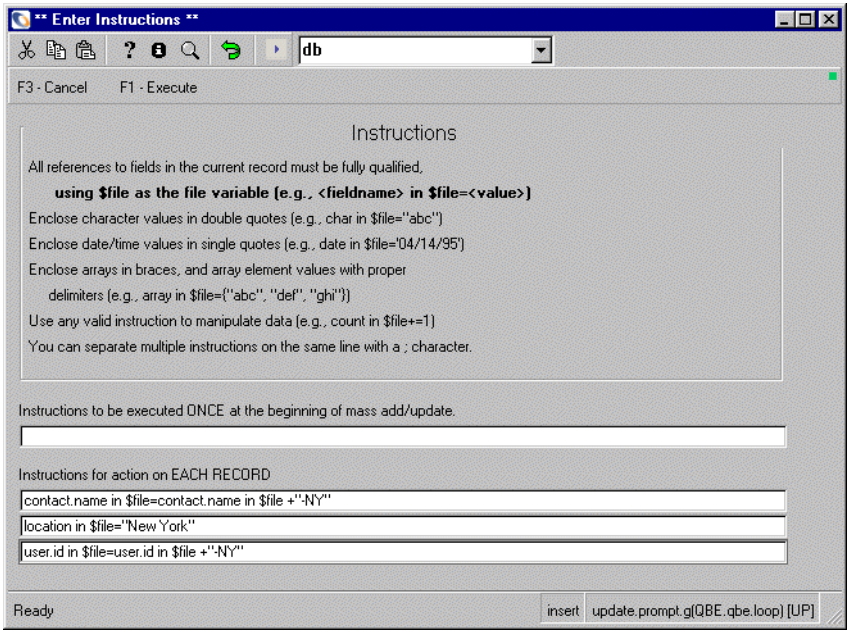


Figure 12-3: Mass Add action specifications

6 To proceed with the mass add, leave the first input field blank. Statements in this field are executed once at the beginning of the add.

7 Enter the following assignment statements on the **Instructions for action on EACH RECORD** text boxes:

Statement	Purpose
contact.name in $file=contact.name in $file +"-NY"	Appends -NY to the end of all contact.name values.
location in $file="New York"	Changes the location value to New York.
user.id in $file=user.id in $file +"-NY"	Appends -NY to the end of all user.id values.

These statements will be executed against every record as it is added.

8 Click **Execute**.

Note: The **Mass Add** function is performed in foreground, which means the session is devoted to the add task until completed. If a large number of records is being updated, this can take a while.

Upon completion, terminal control is returned to you and the following message is displayed in the status bar: *<nnn> records added to the contacts file* where nnn is the number of records added.

9 Click the **View Messages** button to check the messages for errors.

Note: A blue icon indicates a required action, a black icon indicates informational only, and a red icon indicates an error message.

10 Close the Messages window to return to the **contacts** form.

To see the updated record list:

▶ Select **Refresh** from the **List Options** menu to see the added records, or of Record List is turned off, click **F2-Refresh**.

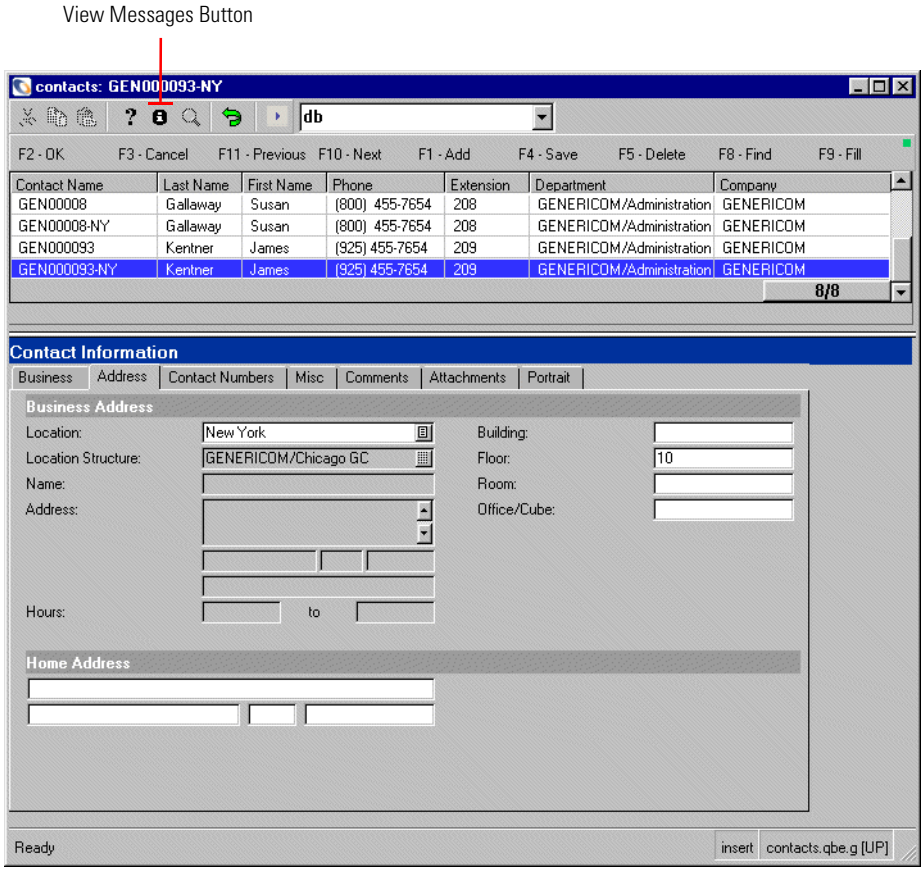


Figure 12-4: Record duplicates with modified contact.name and location values

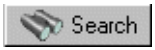
In the new records, the Contact Name (contact.name) and Employee ID (user.id) values are the same as the original records with -NY appended, and the Location (location) values are now New York. The original records listed before the Mass Add remain in the contacts file with no changes to the original data.

To search for the newly added records:

1 Click **OK** to return to the contacts form.

2 Enter GENERICOM in the **Company** field.

3 Enter New York in the **Location** field.



4 Click **Search** or press Enter.

Database Manager performs the *equal to* search and displays the record list of matching record(s), if any, using the contacts.qbe format.

Mass Adding Records Using a Variable Value

The following example demonstrates duplicating all contacts records with a company.name value of GENERICOM with all the same information in each record, except changing the **Company** to NEWGEN, creating sequential user.id values, and creating a unique key for each record.

Since **Contact Name** (contact.name) is a *unique* key and **Employee ID** (user.id) is a *no duplicates* key in the contacts file, unique contact.name and user.id values must be assigned to the new records. (See *Key Definitions* on page 21.) For this example the user.id values in the new records become are suffixed with a sequential number starting with 1000 (e.g., 1000, 1001, etc.).

Mass Add processing statements reference $file to access fields on records in the list. For example, name in $file or 1 in action in $file. Refer to the System Language section of the *System Tailoring Guide* for examples of processing statements.

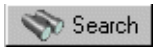
To add records using a variable value:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the **Administration Mode** check box.

A blank contacts form is displayed.

2 Enter the desired **Location** value. For this example, enter New York.

3 Click **Search** or press Enter.



Database Manager performs the *equal to* search and displays the record list of matching record(s), if any. If the query produces no matching records, a message is displayed.

Note: To show the Mass function buttons, use a query that will return more than one record.

4 Select **List Options** > **Mass Add** from the menu bar to start the process of duplicating the records shown in the record list. See Figure 12-2 on page 251.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Add** menu of the search list.

Database Manager displays the Mass Add/Update Instruction screen.

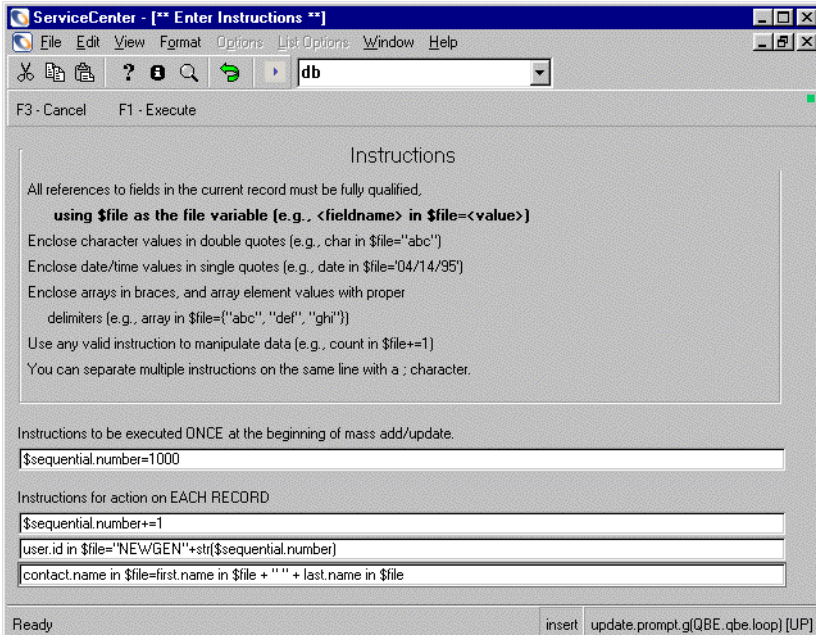


Figure 12-5: Mass Add instructions

5 To proceed with the Mass Add, enter the following assignment statement in the **Instructions to be executed ONCE at the beginning of Mass Add/update** text box.

Statement	Purpose
$sequential.number=1000	Sets the start point for the variable at *1000* and will be executed only once for the entire function.

6 Enter the following statements on the **Instructions for action on EACH RECORD** text boxes:

Statement	Purpose
user.id in $file="NEWGEN"+str($sequential.number)	Sets the User Id to NEWGEN with a sequential number as a suffix.
$sequential.number+=1	Causes one (1) to be added to all future numbers created from this variable.
contact.name in $file=first.name in $file + " " + last.name in $file	Makes the contact name the same as the first and last name.

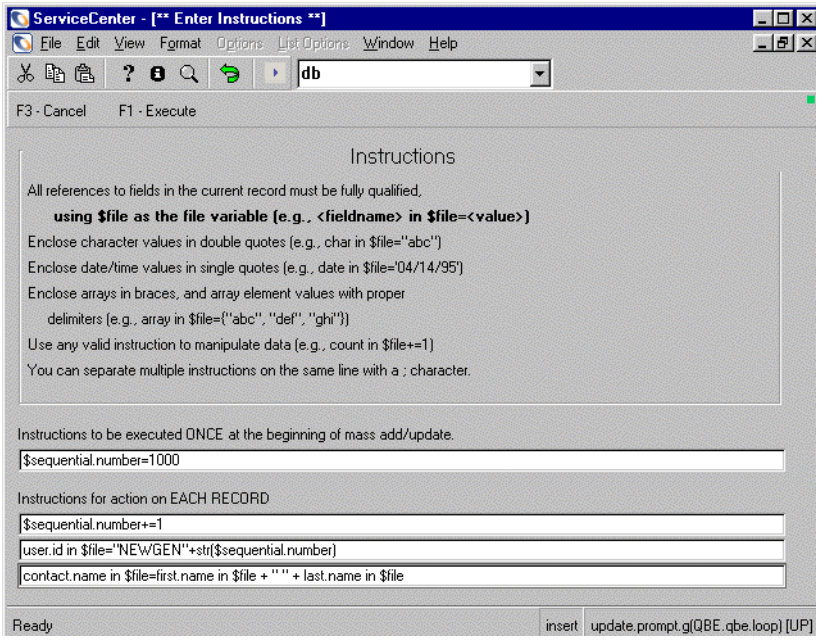


Figure 12-6: Mass Add instructions

These statements are executed against every record as it is added.

7 Click **Execute**.

Note: The *Mass Add* function is performed in foreground, which means your session is devoted to the add task until completed. If a large number of records is being updated, this can take a while.

Upon completion, terminal control is returned to you and the following message is displayed in the status bar: *<nnn> records added to the contacts file* where *nnn* is the number of records added.

8 Click the **View Messages** button to check the messages for errors.

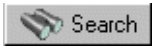
Note: A blue icon indicates a required action, a black icon indicates informational only, and a red icon indicates an error message.

9 Close the Messages window to return to the **contacts** form.

To see the updated record list:

▶ Select **Refresh** from the **List Options** menu to see the added records, or of Record List is turned off, click **F2-Refresh**.

To search for the newly added records:



1 Enter NEWGEN in the Employee ID (user.id) field, and click **Search** or press Enter.

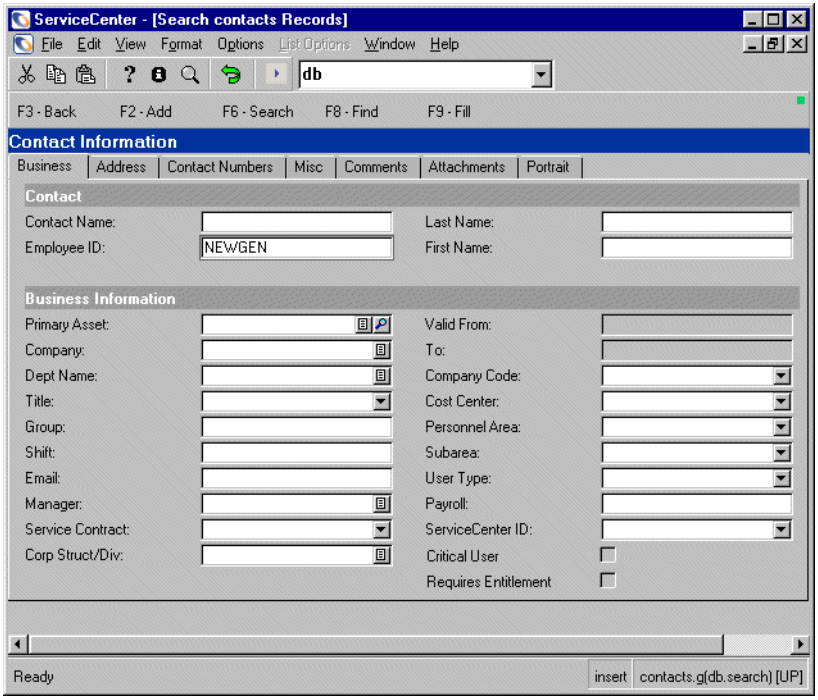


Figure 12-7: Querying for new records

The records you have just added are displayed in a record list.

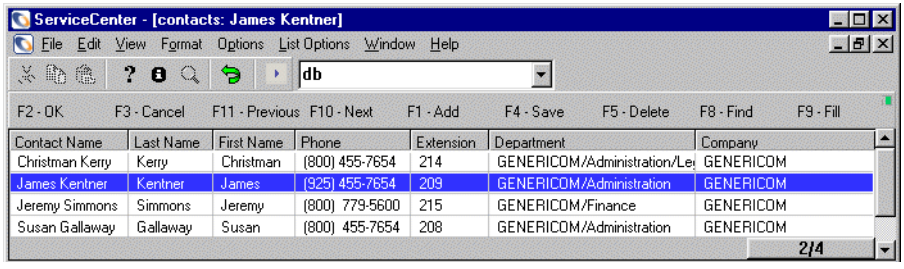


Figure 12-8: Duplicated record with modified logical.name and location values

The Employee ID values consist of Genericom + 1000, 1001, etc., and the Contact Name values the first and the last names. The original records listed before the Mass Add remain in the contacts file with no changes to the original data.

Updating Multiple Records

Having accessed a record list of records from Database Manager, the **Mass Update** function allows the user to enter one or more processing statements which modify the contents of each record in the list.

Mass Update processing statements reference `$file` to access fields on records in the list. For example, `name in $file` or `1 in action in $file`. Refer to the System Language section of the *System Tailoring Guide* for examples of processing statements.

Note: Format Control, triggers and macros , if present for the current form, will be executed for ADD=true processing. Refer to the *System Tailoring Guide* for detailed information.

The following options are available when doing mass updates:

Table 12-1: Options for Mass Update

Option	Action
Retry	Causes everything to be re-executed including the Format Control.
Skip	Causes the record to be skipped.
Force	Causes the record to be updated with whatever information that you entered, regardless of Format Control.

Updating Multiple Records with a Literal Value

The following example demonstrates modifying the Service Contract field value to GENERICOM GEN for the contacts records for all GENERICOM employees in New York.

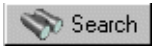
To update multiple records with a literal value:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the Administration Mode check box.

A blank contacts form is displayed.

2 Enter the desired Company value. For this example, select GENERICOM from the Company list.

3 Enter the desired Location value. For this example, enter New York.



4 Click **Search** or press Enter.

All matching records are displayed in a record list.

Note: If the query produces no matching records, a message is displayed. In order to receive the Mass function buttons, use an appropriate query to produce a record list of more than one record.

5 Select **List Options** > **Mass Update** from the menu bar to start the process of updating the records shown in the record list.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Update** menu of the search list.

The initial format is re-displayed with new option buttons.

6 Enter the desired changes. For this example, select GENERICOM GENfrom the **Service Contract** drop-down list.

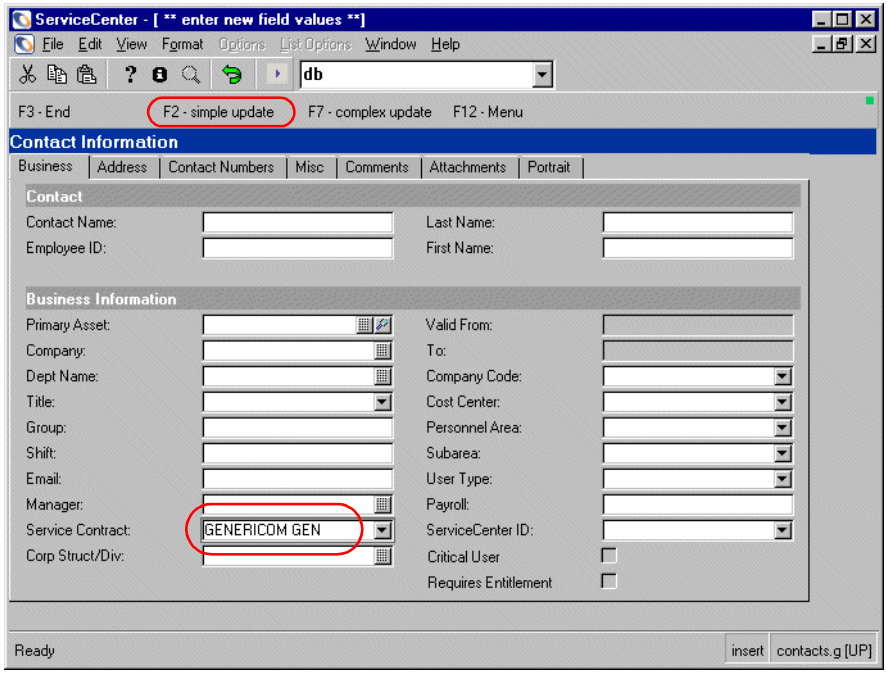


Figure 12-9: Defining mass update field value

7 Click **Simple Update**.

Terminal control is returned to the user, and the message *nnn records updated in the contacts file*, where *nnn* is the number of records updated, is displayed in the Status bar. The newly updated records are displayed in the record list.

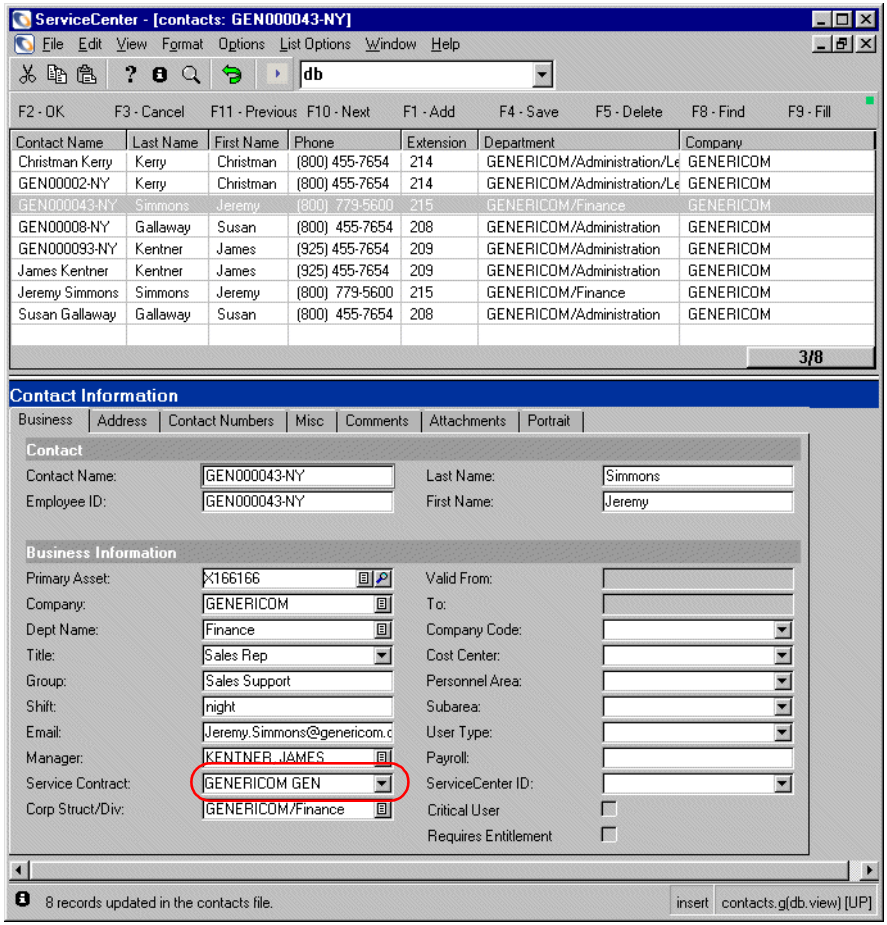


Figure 12-10: Records with mass updated Service Contract value

Updating Multiple Records with a Variable Value

This example illustrates changing the email address for all GENERICOM **contacts** records for people located in New York to follow the pattern: Firstname.Lastname@GENERICOM.com.

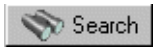
To update multiple records with a variable value:

1 Open the **contacts** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the Administration Mode check box.

A blank **contacts** form is displayed.

2 Enter the desired **Company** value. For this example, select GENERICOM from the Company list.

3 Enter the desired **Location** value. For this example, enter New York.



4 Click **Search** or press Enter.

Database Manager performs the full search and displays the record list of matching records, if any, using the contacts.qbe form.

5 Select **List Options** > **Mass Update** from the menu bar to start the process of updating the records listed.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Update** menu of the search list.

The initial format is re-displayed, with different option buttons.

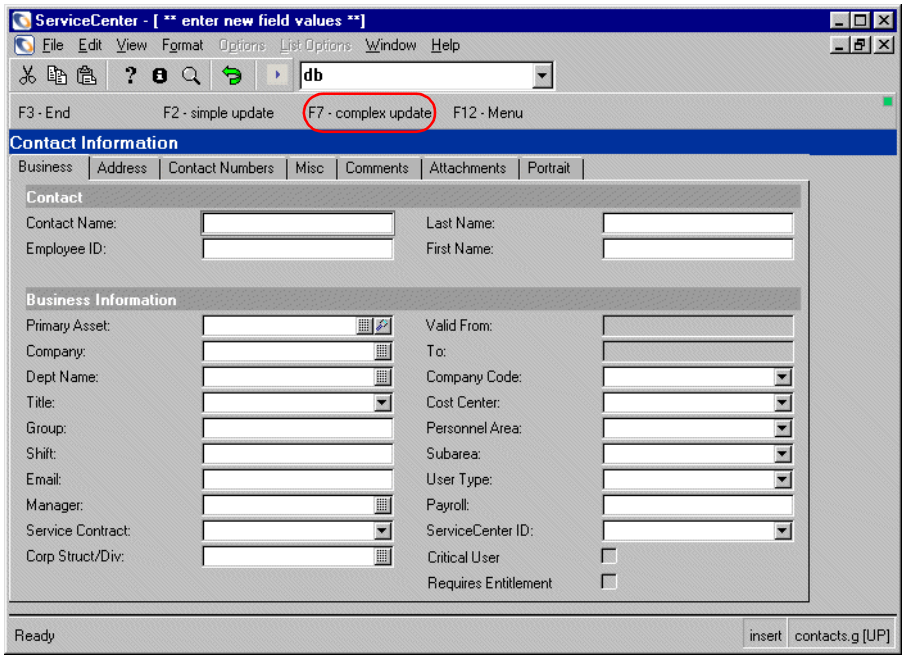


Figure 12-11: Displaying records with mass updated field value

6 Do not enter values in any field, and click **Complex Update**.

Database Manager displays the Mass Add/Update Instruction screen.

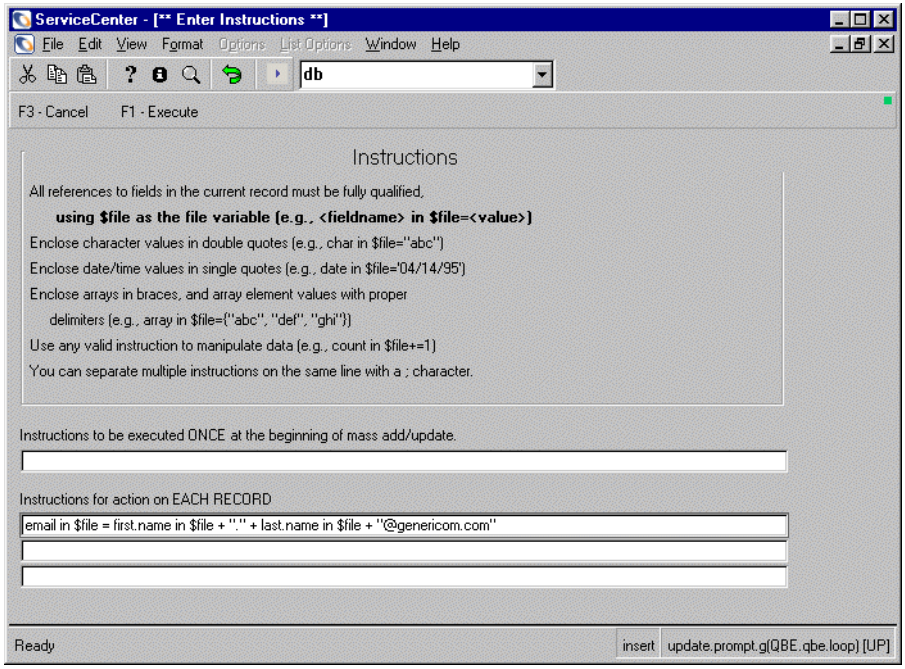


Figure 12-12: Defining Mass Update action

7 Enter the following assignment statement in the first **Instructions for action on EACH RECORD** text box, as shown.

Statement	Purpose
email in $file = first.name in $file + "." + last.name in $file + "@genericom.com"	Sets the email address to firstname.lastname@genericom.com.

This instruction will be executed for each record updated.

8 Click **Execute**.

Note: The **Mass Update** action is performed in foreground, which means the session is devoted to the update task until completed.

Upon completion, terminal control is returned to the user with the message *nnn records updated in the contacts file*, where *nnn* is the number of records updated.

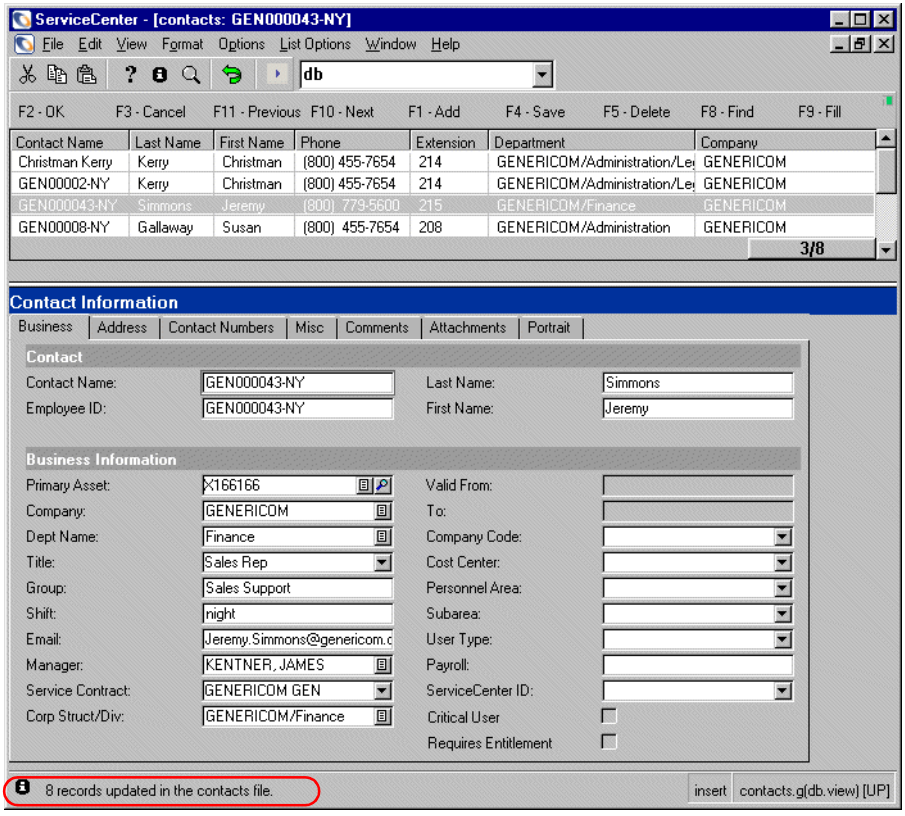


Figure 12-13: Updated Record List message

Mass Add/Update Function Errors

If an error is encountered in any **Mass Add** or **Mass Update** function, the change is not made and the user is prompted to fix the error or skip to the next record.

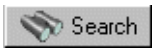
Invalid Duplicate Or NULL Key Errors

Records cannot have duplicate *unique* or *no duplicates* keys. This example is formulated to show one such error and how to overcome it. See *Key Definitions* on page 21.

For example:

1 Open the **contacts** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the Administration Mode check box.

A blank **contacts** form is displayed.



2 Enter NEWGEN in the **Employee ID** field, and click **Search** or press Enter.

3 Select **List Options** > **Mass Update** from the menu bar to begin updating the records shown in the record list.

The initial form is re-displayed. (Figure 12-14 on page 266)

4 On the displayed form, enter NEWGEN in the **Contact Name** field.

5 Click **Simple update**.

The first update is accepted because the contact name is unique. The second is rejected since changing it would create a duplicate contact name.

The record is displayed for modification, with an error message in the status bar.

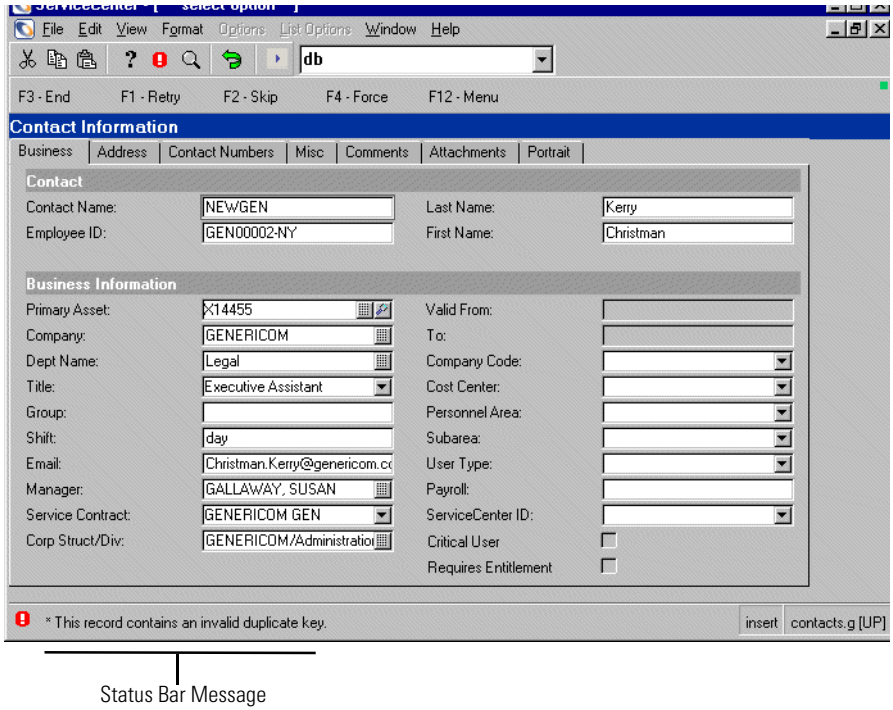


Figure 12-14: Record update error

6 Click the **View Messages** button to check the messages for errors.

Note: A blue icon indicates a required action, a black icon indicates an informational message, and a red icon indicates an error message.

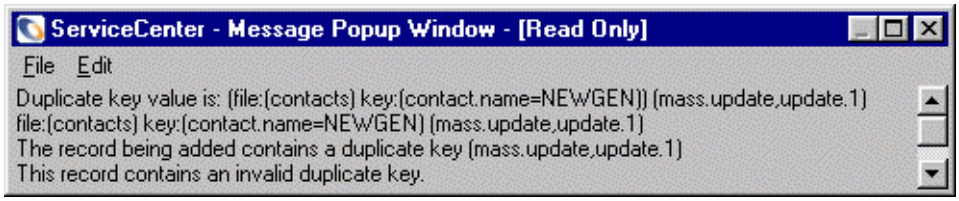


Figure 12-15: Message Popup Window

7 At this point, there are several options concerning the outcome of this process.

- You can enter a unique key and click **Retry** for each item on the list separately. This will cause everything to be re-executed including the Format Control.
- You can click **Skip** — This will cause the record to be skipped.
- You can click **Force** — This will cause the record to be updated with whatever information that you entered, regardless of Format Control.

For this example, modify the **Contact Name** field value to make it unique for each item. (NEWGEN2, NEWGEN3, NEWGEN4, etc.).

If the update is successful, the following message is displayed in the status bar: <n> *records updated in the contacts file.*

Deleting Multiple Records

This example illustrates deleting the contacts for GENERICOM with a variant of NEWGEN as the contact name.

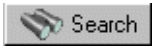
To delete all records with a specified value:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.) Be sure to select the Administration Mode check box.

A blank contacts form is displayed.

2 Enter the desired value. For this example enter NEWGEN* in the **Contact Name** text box.

3 Click **Search** or press Enter.



Database Manager performs the *begins with* search and displays the QBE list of matching record(s), if any, using the contacts.qbe format.

Note: If the query produces no matching records or only one matching record, then a QBE list is not displayed. In order to receive the Mass function buttons, use an appropriate query to return a list of more than one record.

4 Select **List Options** > **Mass Delete** from the menu bar to remove the records shown in the QBE list from the database.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Delete** menu of the search list.

Database Manager confirms the delete request with a prompt screen.

5 To confirm the delete, click **Yes.** (To cancel, click **No**.)

Database Manager deletes all records in the list.

The Mass Delete function is performed in the foreground, which means your terminal is devoted to this function until complete. If a large number of records is being deleted, this can take a while.

Upon completion, terminal control is returned to the user with a blank contacts form with the message: *nnn records deleted from the contacts file*, where *nnn* is the number of records deleted.

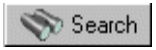
Printing Multiple Records

This example illustrates printing all contact records. For more information on printing, see the System Administrator's Guide.

To print multiple records:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank contacts form is displayed.



2 Do not enter any values, and click **Search** or press Enter.

Database Manager performs the *true* search and displays a complete list of all the contact records.

Note: If the query produces no matching records or only one matching record, then a QBE list is not displayed. In order to receive the Mass function buttons, use an appropriate query to return more than one record.

3 If **View** >**Record List** is turned on, select **List Options** > **Print List** from the ServiceCenter menu bar to print the records shown in the list.

Note: In this example, **Record List** has been selected from the ServiceCenter **View** menu (see *The Record List* on page 248). When **Record List** has not been selected, equivalent options are displayed in the **Options** > **Print** menu of the Search list.

One of the following printing dialog boxes is displayed.

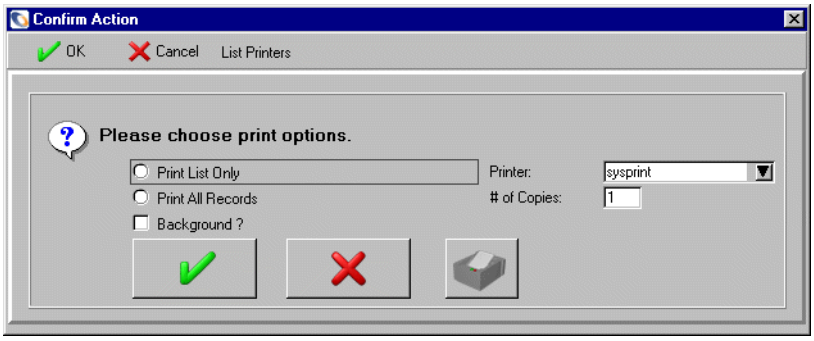


Figure 12-16: Print confirmation window for server printer settings

— Or —

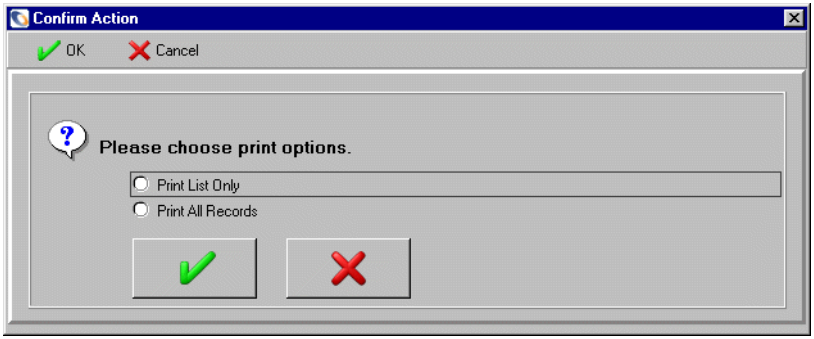


Figure 12-17: Print confirmation window for client printer settings

4 Select from the following printing options.

The options are all present if your printer setup specifies that you are to use the server printer. Only the first two are present if your settings indicate that you are using a client printer.

Table 12-2: Print Options

Option	Definition
Print List Only	Print the list exactly as shown with a total count at the bottom.
Print All Records	Print the corresponding records of all items listed.
Background	Print the selection in the background, leaving terminal session free for other operations. Server printer only.
Printer	Select the printer to be used. Server printer only. (A client printer prints to the default printer for the client computer.)
# of Copies	Enter the desired number of copies. Server printer only.

5 Select a printer from the drop-down list or click the **Printer** button to display a list of system printers.

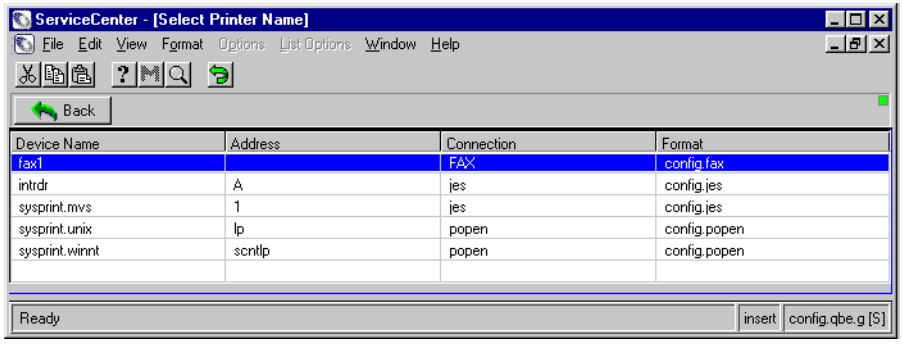


Figure 12-18: Available system printers

6 Double-click a printer to select it.

7 Select the number of copies you want to print.

8 Click **OK**.

Note: This function, when performed in the foreground, means your terminal session is devoted to printing the records.

Upon completion, terminal control is returned to you with a blank contacts format with the message: *Report spooled as no. nnn. (ServiceCenter Print Job).*

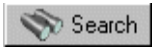
Counting Records

The **Count** menu option is a convenient, quick way to determine the number of records in a QBE list. By selecting any record and clicking **Count** the user can get a clear picture of how many records are contained in a particular list.

To determine the number of records in a QBE list:

1 Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank contacts form is displayed.



2 Leave all fields blank and click **Search** or press Enter.

- If **Record List** has been selected from the **View** menu, the contacts format is displayed with a record list at the top, and the **Count** option in the **List Options** menu.
  - Select **List Options** > **Count** from the menu bar. A prompt is displayed asking whether all records in the list should be counted.
  - Confirm by clicking **Yes**. Cancel the count call by clicking **No**.

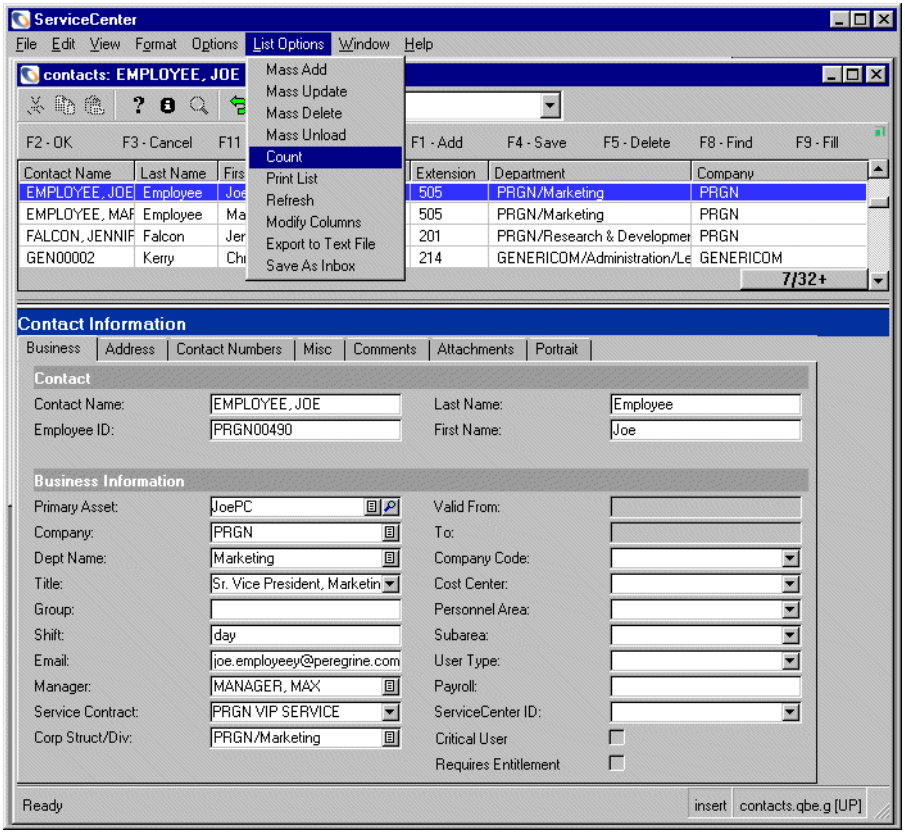


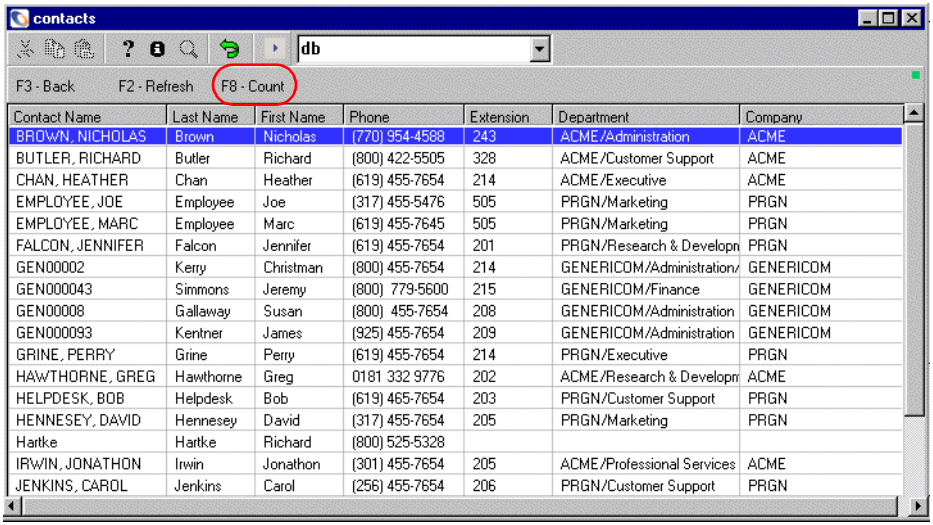
Figure 12-19: Record Count option

After confirming the process, the records are counted and the following message is displayed: *There are* <n> *records in this list.*

Database Manager performs the full search and displays the record list of matching record(s), if any.

- If Record List has not been selected from the **View** menu, a **Count** button (F8) will be displayed on the list of records.

  Click the **Count** button or press F8 to get the record count.



Contact Name	Last Name	First Name	Phone	Extension	Department	Company
BROWN, NICHOLAS	Brown	Nicholas	(770) 954-4588	243	ACME/Administration	ACME
BUTLER, RICHARD	Butler	Richard	(800) 422-5505	328	ACME/Customer Support	ACME
CHAN, HEATHER	Chan	Heather	(619) 455-7654	214	ACME/Executive	ACME
EMPLOYEE, JOE	Employee	Joe	(317) 455-5476	505	PRGN/Marketing	PRGN
EMPLOYEE, MARC	Employee	Marc	(619) 455-7645	505	PRGN/Marketing	PRGN
FALCON, JENNIFER	Falcon	Jennifer	(619) 455-7654	201	PRGN/Research & Developn	PRGN
GEN00002	Kerry	Christman	(800) 455-7654	214	GENERICOM/Administration/	GENERICOM
GEN000043	Simmons	Jeremy	(800) 779-5600	215	GENERICOM/Finance	GENERICOM
GEN00008	Gallaway	Susan	(800) 455-7654	208	GENERICOM/Administration	GENERICOM
GEN000093	Kentner	James	(925) 455-7654	209	GENERICOM/Administration	GENERICOM
GRINE, PERRY	Grine	Perry	(619) 455-7654	214	PRGN/Executive	PRGN
HAWTHORNE, GREG	Hawthorne	Greg	0181 332 9776	202	ACME/Research & Developn	ACME
HELPDESK, BOB	Helpdesk	Bob	(619) 465-7654	203	PRGN/Customer Support	PRGN
HENNESEY, DAVID	Hennesey	David	(317) 455-7654	205	PRGN/Marketing	PRGN
Hartke	Hartke	Richard	(800) 525-5328			
IRWIN, JONATHON	Irwin	Jonathon	(301) 455-7654	205	ACME/Professional Services	ACME
JENKINS, CAROL	Jenkins	Carol	(256) 455-7654	206	PRGN/Customer Support	PRGN

13 CHAPTER Database Record Auditing

This chapter was designed to aid ServiceCenter system and database administrators check specified fields within a file in the ServiceCenter database for modifications, when records in that file are updated.

Topics in this chapter include:

Introduction

Auditing allows the user to check specified fields within a file in the ServiceCenter database for any modifications, when records in that file are updated. It tracks record updates when paging (i.e. creating a unique record or page for every update) is not available, and provides an alternative to Paging.

Field modifications are detected by comparing the field input values in the original version of a record to the same field values in the updated version of that record. When modifications are detected, an Audit Log entry is generated showing:

- The name of the modified field(s).
- The old and new version of the data.
- The date/time of the modification.
- The userid of the operator who modified the record.

The Audit Specifications File

The Audit Specifications file provides the instructions on how and when to perform an audit. It defines files and fields to be watched by the Audit application. There is one specification record for each ServiceCenter Database Dictionary (dbdict) file.

To access the Audit Specifications file from the Tools menu:

1 From the system administrator's main menu, select the **Utilities** tab.

2 Click **Tools**.

The ServiceCenter **Tools** menu is displayed.

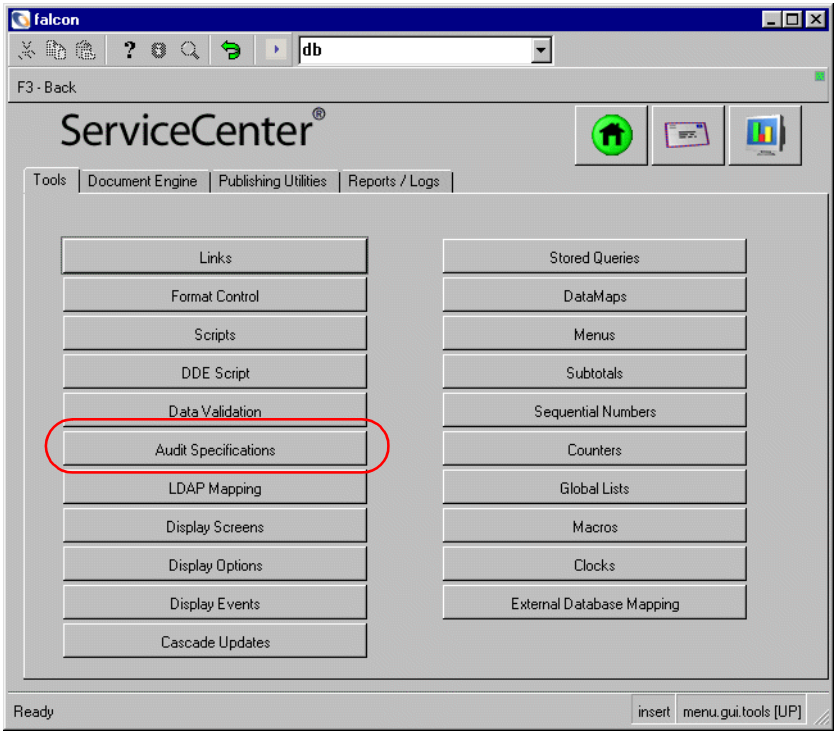


Figure 13-1: System Administrator's Tools menu — Tools tab

3 Click **Audit Specifications**.

A blank Audit Specifications table will be displayed. See Figure 13-3 on page 278.

Important: Figure 13-19 on page 300 shows a record that does not exit in the default system. Create it as displayed for the examples in this section. See *Defining an Audit Specifications Entry* on page 283

To access the audit specifications file from the Command line:

1 Type audspec on the Command line and press Enter.

Figure 13-2: ServiceCenter Command line

2 Type audspec on the Command line and press Enter.

A blank Audit Specifications table will be displayed. See Figure 13-3 on page 278.

To open the Audit Specifications file from Database Manager:

- Open the **auditspecs** form. For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.

  A blank Audit Specifications table will be displayed. See Figure 13-3 on page 278.

Audit Specifications File Description

The Format Control record associated with the **auditspecs** format executes a routine which validates entries in the **Filename** and **Field Name** fields. This validation routine is executed when Audit Specification records are added or updated. It prevents invalid fields or file values from being entered into the system and controls unpredictable run-time results. This routine is described in *Defining an Audit Specifications Entry* on page 283 and *Field Name Verification* on page 285.

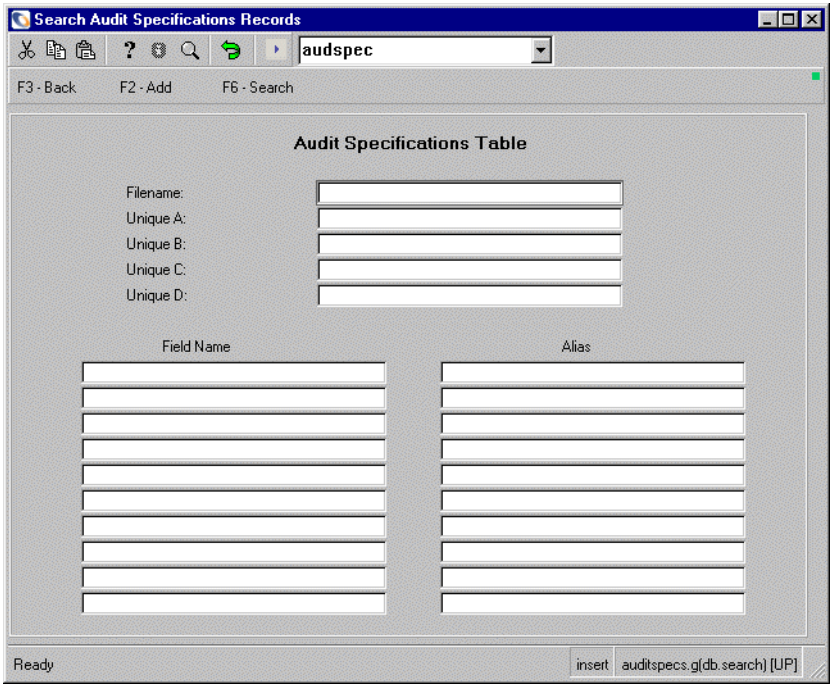


Figure 13-3: Audit Specifications Table

Table 13-1: Fields in the Audit Specifications Table

Field	Definition
File Name	The name of a valid ServiceCenter Database Dictionary file upon which Audit will be performed. This is a **required** field on the table.
Unique A - D	Used to parallel data records in the Audit File to data records in the source file. Under most circumstances, the field(s) specified in Unique A through D are the same field(s) that are defined as *unique* keys in the specified Database Dictionary file. These values need to be unique identifiers for the specific file. **Unique A** is the only **required** entry. Additional entries for **Unique B through D** are optional. For example, the unique key for the contact file is contact.name. In this case, Unique A is defined as **contact.name** and Unique B **through** D are left blank (NULL). When an Audit Log record is recorded for a contact John Miller, **Filename** in the Audit Log is recorded as contacts, **Unique A** as *JOHN MILLER*. This allows for all contacts Audit Log records to be uniquely associated with each contacts record. A new Audit Log can be generated based on data found in the previous log record for this device. It is possible for **Unique A through D** to use the same fields as a *no nulls* or a *no duplicates* key. However, this should only be done when a unique key is not available. If *no nulls* keys are used, Audit Records could be related to different source records, making it difficult, if not impossible, to determine to which specific source record a particular Audit Record is related. (See *Key Definitions* on page 21.) Do not use *nulls & duplicates* keys to define **Unique A through D,** because if the field(s) in the key are NULL, then Audit records are created which do not relate to any records in the Source File. The fields **Unique A through D** need not be defined as key(s) in the Source File. However, you must take the necessary steps to ensure any **non-key** field(s) used are defined as unique identifiers, in order to avoid the potential problems outlined above. See *Database Dictionary* in ***System Tailoring*** for more information on keys and other aspects of files. When defining Audit Specifications for the Problem and Change Management files, the only field that needs to be paralleled is **number**. It is not necessary to parallel the **last** or **page** fields. See *Database Dictionary* in ***System Tailoring*** for information on identifying fields in files.

Table 13-1: Fields in the Audit Specifications Table

Field	Definition
Field Name	Defines the name of the field to be checked for modifications. The fields specified for Auditing can be of any data type except arrayed structures or fields within arrayed structures. Any number of fields can be specified; however, Auditing overhead increases as the number of fields increase. It is not recommended to specify all fields within a file for Auditing. Rather, analyze the fields within the file to determine which are critical for the management of data records in the file. The following fields in the contacts file are considered critical: ■ contact.name ■ first.name ■ last.name ■ dept.name ■ email ■ location The recommended maximum number of fields to audit is 20. Performance degradation occurs when this number is exceeded. Under most circumstances, the recommended maximum will not impair the management of a file.
Alias	Defines the alias of a **field name** that will be used in the audit log. When entries are recorded in the audit log (audit.summ.g form), the default is to record the actual field name. This is overridden by specifying an alias. For example, if the widgets file has a field named **fd.ast.no**, it may be more meaningful to define the alias field name **fixed asset number**.

Note: A one-to-one correlation exists between the **Field Name** and **Alias** input fields. Due to processing considerations, these are independent arrays. Therefore, when one is scrolled, the other must also be scrolled to keep the definitions synchronized.

The Audit Log File

The Audit Log File is the repository for comparative data gathered during the audit. It displays the old and new input field data, as well as which user made the revisions and when. The log is a record of the update transaction for the specified source file. One log record is added per source record update, when one or more of the source record input fields defined in the Audit Specifications file are updated.

Note: The following procedure will display a list of audit log records only if you have made modifications to **contacts** records after creating the audit specifications record as displayed in the previous section. Otherwise, you receive an empty log record and no list.

To open the Audit Log file from the Maintenance tab:

1. From the System Administrator's main menu, select the **Utilities** tab.
2. Click **Maintenance**.
3. Select the **Logs** menu tab.
4. Click **Audit Log**. If log records exist, they are displayed.

To open the audit log file from the Command line:

- Type audlog on the Command line and press Enter.

Figure 13-4: ServiceCenter Command line

To open the Audit log file from Database Manager:

- Open the audit.summ form.

  For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.

Audit Log File Description

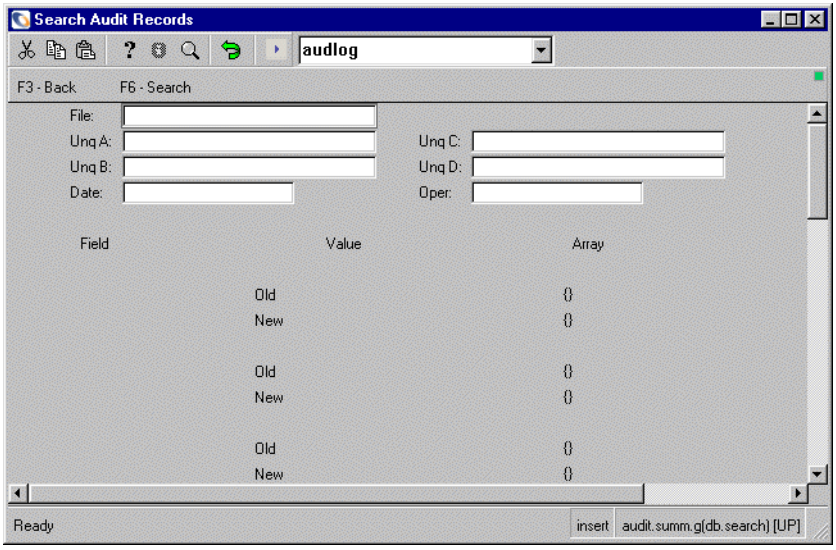


Figure 13-5: Audit log file

Table 13-2: Audit Log Fields

Field	Definition
File	The file name with which this Audit Log record is associated.
Unq A through D	The contents of the source record fields defined as Unique A through D in the Audit Specifications file.
Date	The creation date and time that the Audit Log entry was created.
Oper	The *userid* of the operator who modified the source record, causing the Audit Log entry to be generated.
Field	The field name, as defined in the Audit Specifications file, or its Alias which has been modified. The old and new version of each **Value** (i.e., scalar field) or **Array** (i.e., arrays of simple data types) is recorded. All of these fields are defined in **Format Designer** as *read/display only*, and therefore cannot be modified by the user. These fields are scrollable and synchronized so when one is scrolled all related fields move as well.

Defining an Audit Specifications Entry

The Filename and Field Name values are validated anytime an existing record is updated or a new record added. The Audit utility safeguard system prevents records with misspelled and incorrect file names or field names from being processed. Such errors potentially could cause faulty communication within the database.

Note: The name of the file and one field name of the following example have been entered incorrectly to illustrate the error-correction process built into the Audit utility.

To enter data in the Audit Specification file:

1 Open the Audit Specifications Table following the instructions given in *The Audit Specifications File* on page 276.

A blank Audit Specifications Table form is displayed.

2 To select the file to create the specifications for, enter information in the **Filename** and the **Unique A** through **D** fields as necessary to parallel the unique key in the source file with the Audit Log (refer to *Audit Specifications File Description* on page 278 for more information on input fields). For this example, enter contacts in the **File Name** text box, and click **Search** or press Enter.

3 Define those fields you wish to be audited and any aliases. If an field name is invalid, a list will open up allowing you to copy a valid name and use it to replace the invalid name. See *Field Name Verification* on page 285. For this example, enter:

Field Name	Alias
contact.name	Contact Name
user.id	Employee ID
first.name	First Name
last.name	Last Name
dept.name	Department
email	Email Address
location	Location

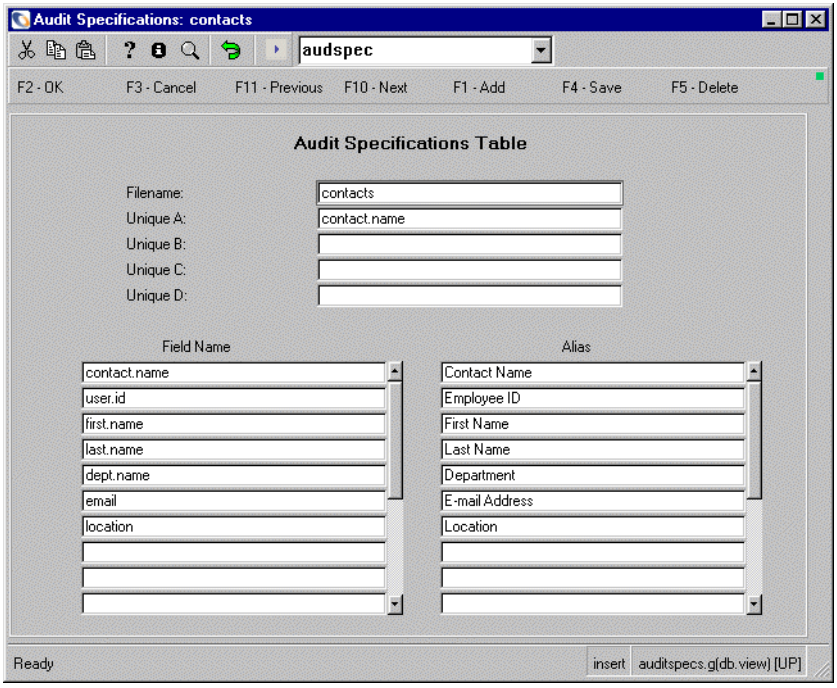


Figure 13-6: Audit Specifications Table

4 Click **Add** to retain this record and commit it to the database.

Important: Making changes to the contacts file to cause it to invoke auditing will cause two audit records to be generated when a file is updated. To prevent this, do a backup and restore the original file when finished with the example, or create a different file to practice on.

File Name Verification

When defining an audit specifications entry, if the File Name value is invalid, The message "*The filename 'filename' is not valid. Select one from the list.*" is issued, and a QBE list is displayed, showing valid ServiceCenter Database Dictionary file names.

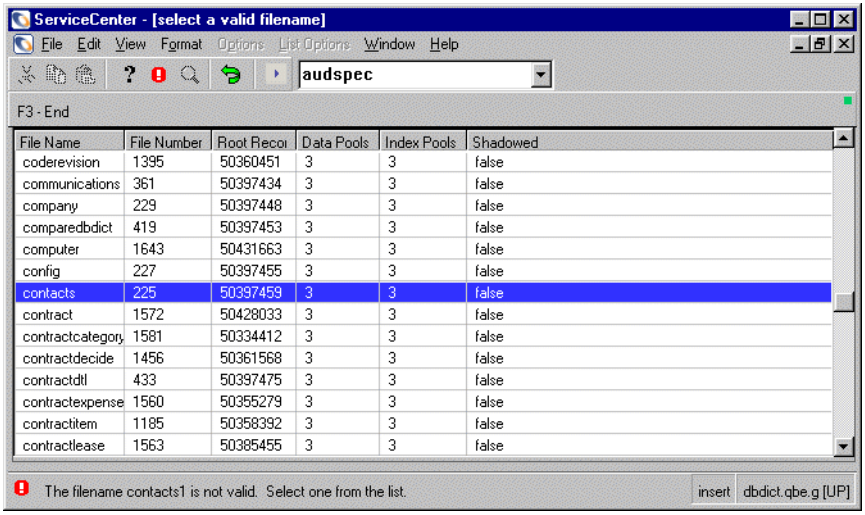


Figure 13-7: Qbe with invalid system filename message

To correct the filename:

- Select an entry in the list (**contacts** for this example), and press Enter.

  The correct file name is copied to the **File Name** field.

Field Name Verification

Similar to the Filename values, all field names defined in the record (Unique A through D and all Field Names) are validated against the selected Database Dictionary file.

When an invalid field name is found, the message "*The Field 'field name' is invalid. Select one from the list.*" is issued, and a pop-up window is displayed showing a list of valid field names for the specified Database Dictionary.

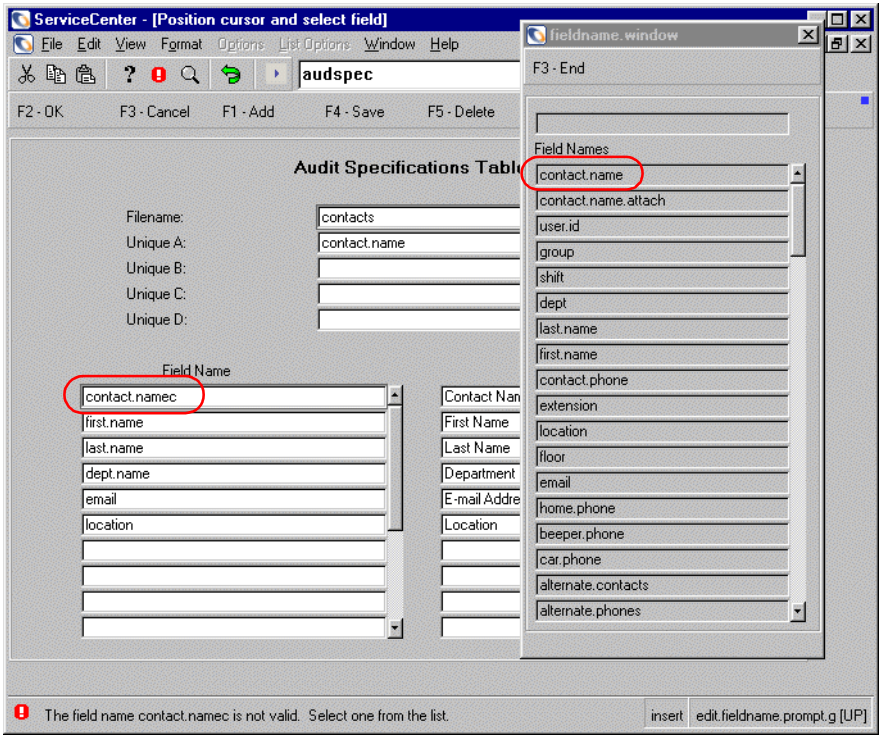


Figure 13-8: Audit Specification Field Names window

To correct a field name:

1 Double click on the correct Field name in the **fieldname** window, and click **End.**

The selected field name is copied to the appropriate field.

— or —

Copy the correct field name click **End**, and paste it into the field.

The message *Record added to the auditspecs file* is issued, which means that all fields have been validated and that the record has been saved.

2 Re-query the record before attempting to make any modifications.

- **a** Exit the form.
- **b** Re-open the **auditspec** form.
- **c** Query for the **contacts** specifications record.

Maintenance of the Audit Specifications file follows normal ServiceCenter Database Manager procedures. See *File Maintenance* on page 311 for more information).

Invoking Audit Processing

Important: If you are invoking Auditing from Format Control or a RAD call, as a general rule, you should invoke Auditing *after* Validity Table processing but *before* the record add or update.

For some files, such as **contacts**, the out-of-box Format Control specifies that every time an update occurs, the audit.compare application is called. If you invoke auditing for a file that already has auditing being called by the Format Control, an audit records will be generated by Format Control AND by trigger.invoke.auditor, making two audit records when a file is updated.

To determine if a file already has audit processing:

1. Open the **triggers** file in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)
2. Click search. A list of all files with triggers will appear.
3. Search for the filename on the list.

There are two methods to invoke auditing. The second method is valid for Change Management only.

- *Setting Up Auditing from Format Control* on page 287
- *Setting Up Auditing from the File in Database Manager (CM only)* on page 292

Setting Up Auditing from Format Control

To invoke Auditing from Format Control:

1. Create an Audit Specifications record for a particular file. Refer to *Defining an Audit Specifications Entry* on page 283 for details.
2. Access the Format Control record associated with the form and the file for which you created the audit specifications record in step 1.
   a. From the System Administrator's main menu, select the **Utilities** tab.
   b. Click **Tools**.

3 Click **Format Control**.

A blank *Format Control Maintenance* form is displayed.

Important: Ensure the **Save Copy** parameter checkbox on the Main Information screen is checked (set to *true)*

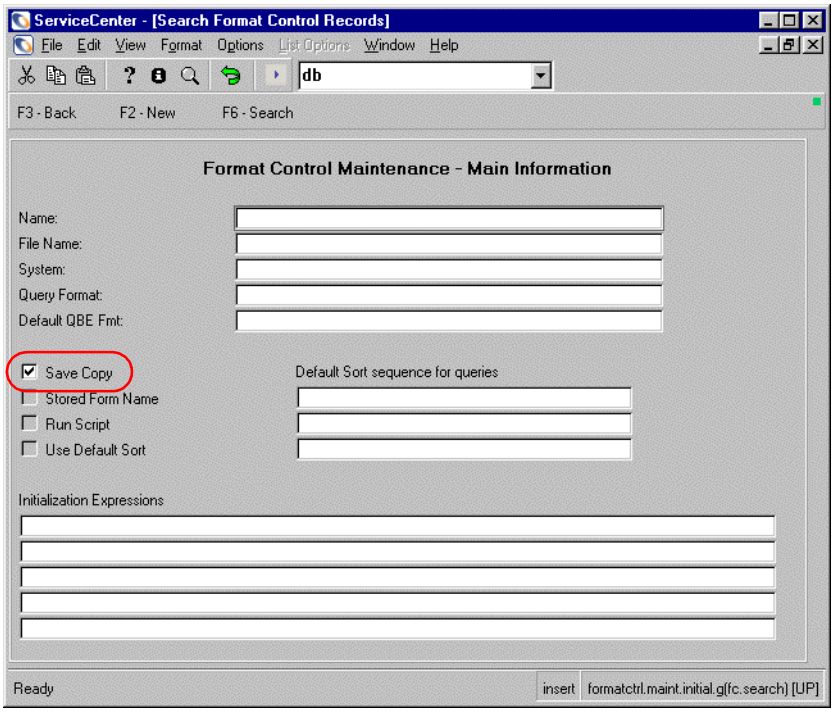
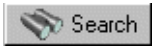


Figure 13-9: Format Control Maintenance — Main Information screen



4 Enter the name of the form you wish to view (in this example, **contacts**), or leave all fields blank, and click **Search** or press Enter.

A list of active forms is displayed.

a If you left all fields blank, select a specific form from the list of records.

The selected form is displayed, showing the Main Information screen.

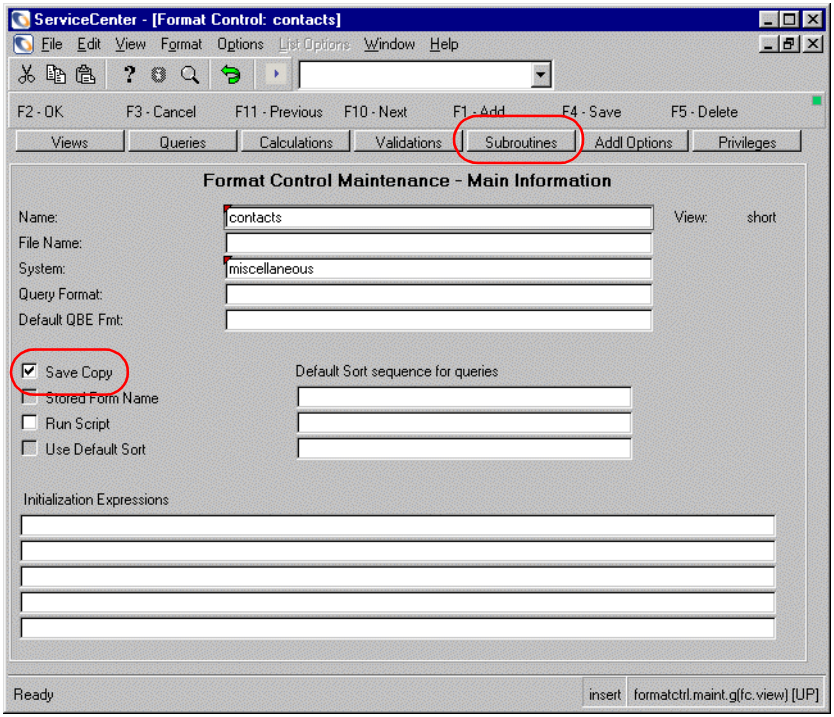


Figure 13-10: Activating history tracking options

5 Make sure that the **Save/Copy** option is selected.

6 Click **Subroutines** to display the **Subroutines** form, or select **Options** > **Subroutine** from the menu bar.

The Subroutines panel is presented, as shown in Figure 13-11 on page 290.

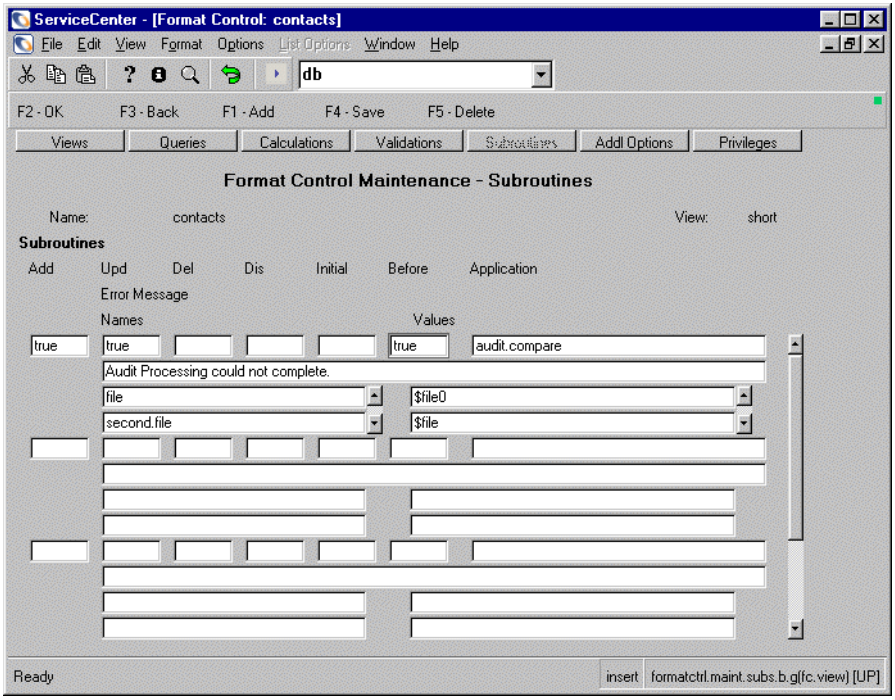


Figure 13-11: Setting up Format Control subroutines

7 Enter the desired format control. See the Table 13-3 on page 291 for field definitions. For this example enter the following.

Field	Enter
Add	true
Upd	true
Before	true
Application	audit.compare
Error Message	Audit Processing could not complete.

Field Definitions

Each field in the Subroutines form needs to be considered in terms of the final outcome you are seeking.

Table 13-3: Fields in the Subroutines Form

Field	Definition
Add	This field needs to be *true*, since you want audit data to be written whenever a new record is *added*.
Upd	(update) This field needs to be *true* to allow audit data to be written whenever there is a modification to an existing record.
Del	(delete) This field is empty, as that would write an audit record whenever a record is deleted.
Dis	(display) This field is empty.
Before	This setting on the subroutine call can be set to *true* or *false*. - If it is *true*, and a key error is detected, then an Audit record exists, but the update never really occurs. - If it is *false*, then you receive the *Record Updated* message before the *Audit Recorded* message.
Application	This field states the application that compares the fields in the old and new data records, and if necessary, adds a record to the Audit Log file.
Error Message	This field provides the error message, which is displayed if the process cannot be completed as planned.
Names	This field states which files are handled by the subroutine. The *Old* (or original) version of the data record must be passed to the file parameter. The Save Copy option on the Format Control Maintenance form must be activated before this file variable is available. The *New* (or updated) version of the data record must be passed to the second.file parameter
Values	Values in this field correspond to each parameter. These values state to which variables data is passed.

Setting Up Auditing from the File in Database Manager (CM only)

This process works for Change Management only.

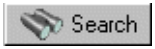
To invoke auditing from Database manager:

1 Open the cm3rcatphase.main form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank cm3rcatphase.main form is displayed.

2 Enter the name of any phase in the form.

Analysis is used in this example.



3 Click **Search** or press Enter.

4 Set the **Audit Records** field value to *true*.

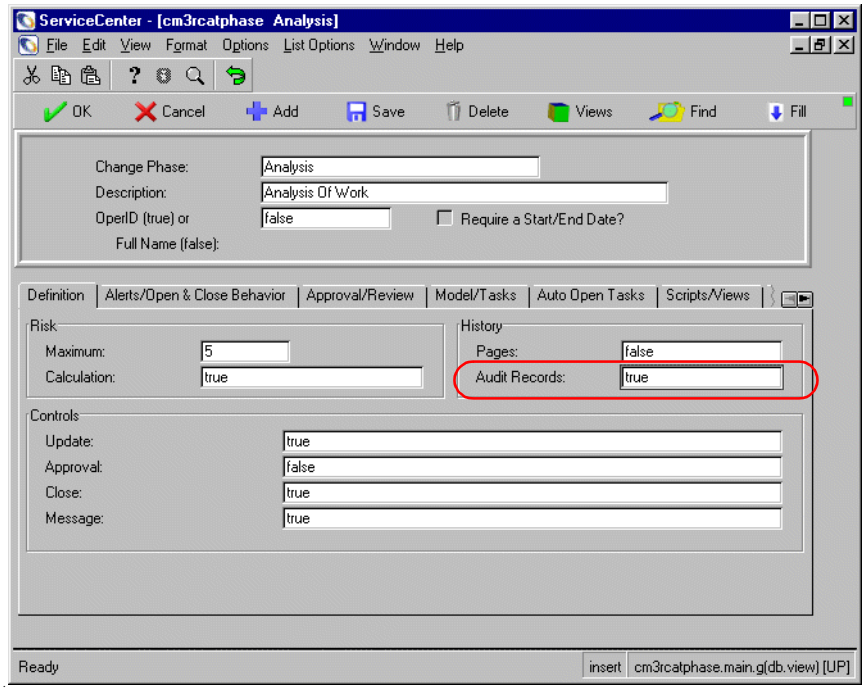


Figure 13-12: Activating History tracking option

5 Click **Save**.

Trigger Setup

Event triggers have been integrated into the system auditing function. A trigger can be set up to invoke the auditing application, audit.compare.

Note: If an audit is needed for ICM, do not use triggers. Instead, use format control to call the audit.compare application.

Set up the audit.compare trigger as follows:

1 Open the triggers form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank triggers form is displayed.

2 Enter the name of the new trigger, e.g., example.trigger.audit.update.

Note: Each trigger performs one action; therefore, when naming the trigger, you may wish to incorporate the trigger type (when the trigger is to fire) into the name of the trigger. For example, trigger.audit.add could be the name of a trigger that fires whenever a record is added.

3 Enter the name of the file which will be audited in the **Table Name** field. In this case it is the contacts file.

4 Select the type of trigger, according to the number legend displayed beside the field. Use 4 in this case.

5 Enter trigger.invoke.auditor in the **application** field.

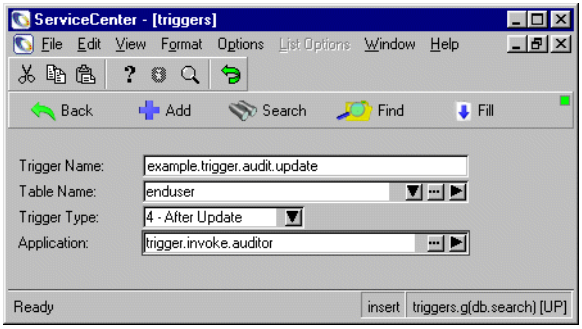


Figure 13-13: Trigger invoker

6 Click **Add** when ready to add this record to the **triggers** file.

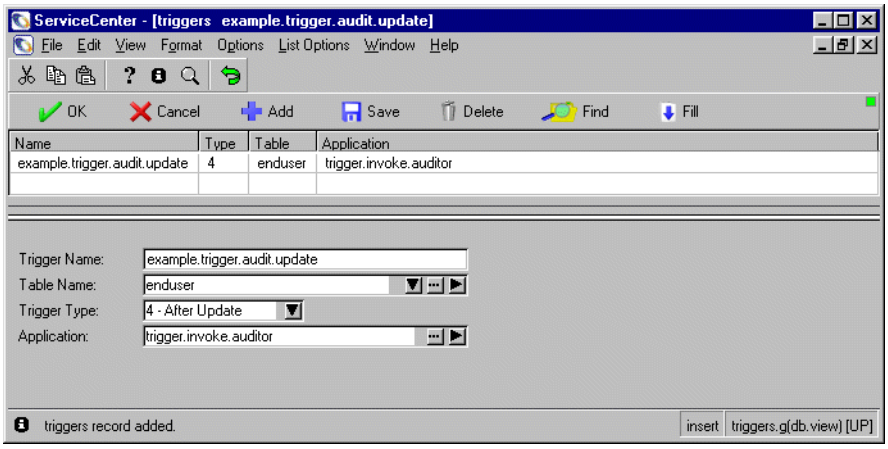


Figure 13-14: Trigger Record for updates to contacts file

Looking Up Audit Log Entries

Some files may not be configured for Audit Lookup. If that is the case, Audit Lookup functionality must be added before Audit Lookup will work.

Adding Lookup Functionality to Format Control

Before using Audit Lookup, be sure that the Audit Specifications include all fields that need auditing. See *Defining an Audit Specifications Entry* on page 283.

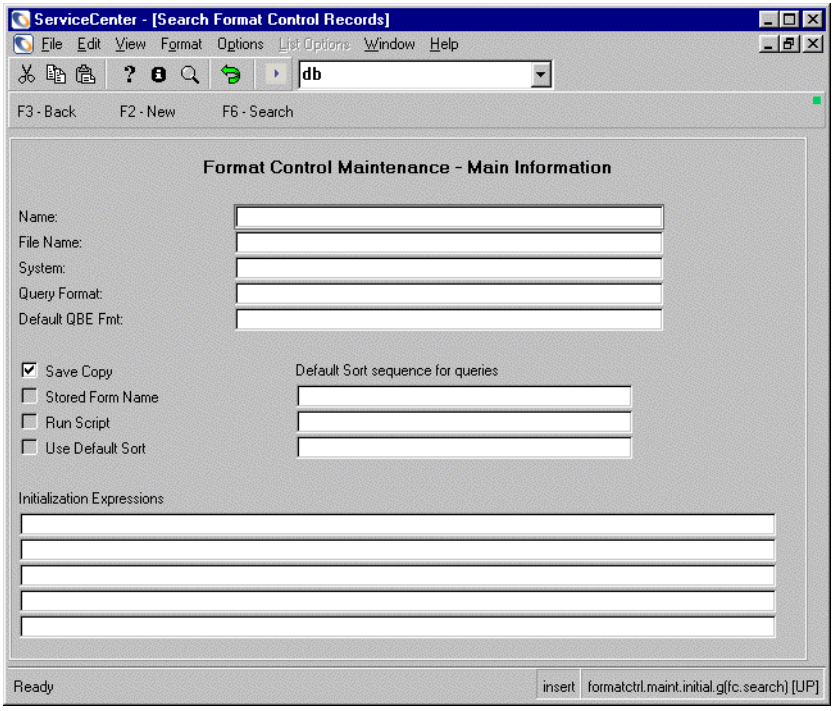
To add Lookup Functionality:

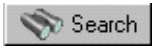
1 Access the Format Control record associated with the form and the file.

a From the System Administrator's main menu, select the **Utilities** tab.

b Click **Tools**.

2 Click **Format Control**. A blank *Format Control Maintenance* form is displayed.





3 Enter the name of the form you wish to update (in this example, enter **contacts**), or leave all fields blank, and click **Search** or press Enter.

a If a list of records is displayed, select a specific record. If you entered a specific file name, the selected form is displayed, showing the Main Information screen.

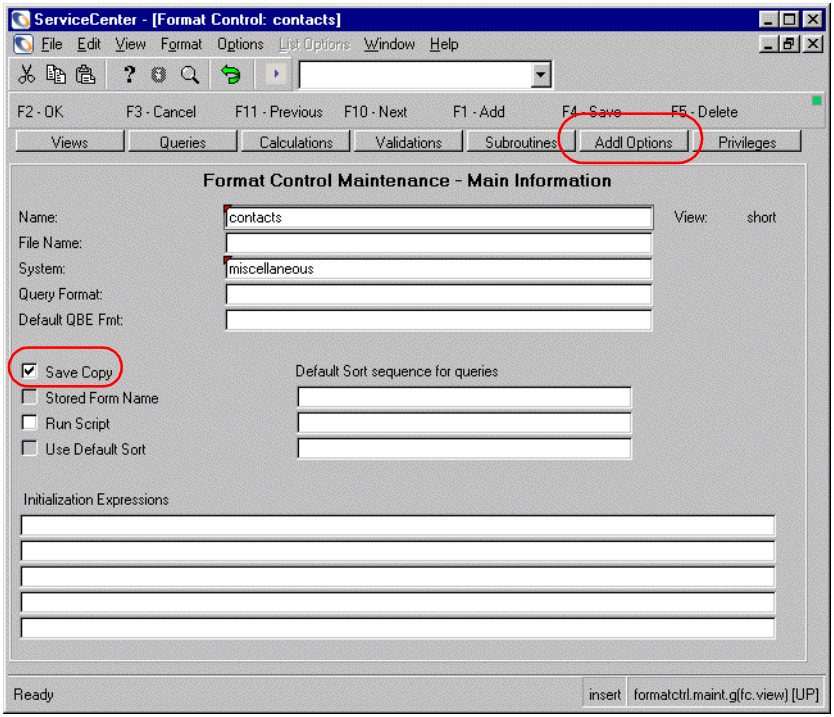


Figure 13-15: Activating History tracking option

4 Make sure that the **Save/Copy** option is selected.

5 Click **Addl Options** to display the **Subroutines** form, or select **Options** > **Additional Options** from the menu bar.

The Additional Options panel is presented.

6 Activate Lookup Functionality by adding the following specifications:

Field	Contents
Option	1
Desc	Audit Lookup
Condition	true
Application	audit.lookup
Comment	Audit Lookup
Message	Could not call Audit Lookup application.
Names	file
Values	$file

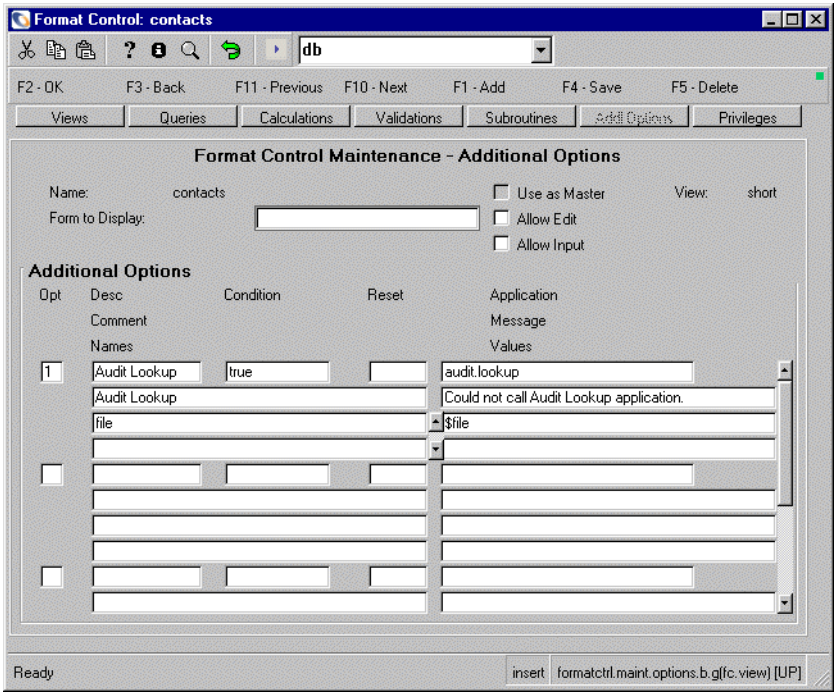


Figure 13-16: Adding Audit Lookup functionality

The ServiceCenter Options menu will now contain the Audit Lookup menu item.

Figure 13-17: Audit Lookup menu item

Run-time Example

Using the definitions provided in the Format Control example in *Setting Up Auditing from Format Control* on page 287, Audit Processing is invoked when a record is added or updated in the **contacts** file.

Note: Before using Audit Lookup, be sure that the Audit Specifications include all fields that need auditing. See *Defining an Audit Specifications Entry* on page 283.

For example:

1 Add a record to the **contacts** file using the following values. For instructions on how to add a record, see *Adding a Record* on page 238.

Field	Value
Contact Name	JOHN MILLER
Employee ID	GEN00003
Last Name	Miller
First Name	John
Company	GENERICOM
Email	john.miller@genericom.com
Dept Name	Documentation

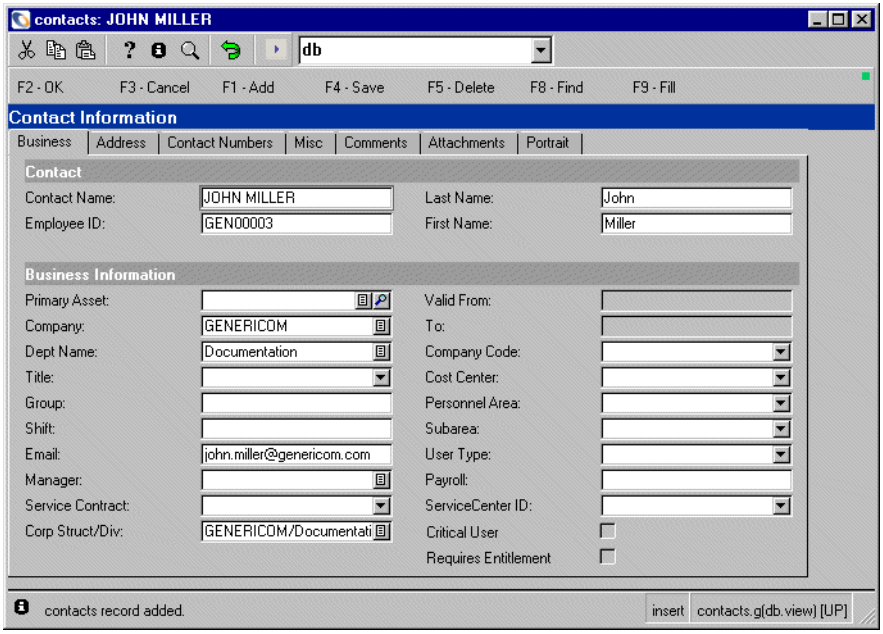


Figure 13-18: New Contact record

Note: The following message is displayed in the status bar: *contacts record added*. The audit process is invoked. No audit record will be displayed until a change is made from the current field values.

All fields specified in the Audit Specifications record have been recorded. All Old values are NULL or contain no data because there is no previous version of the record; this is a new record.

Note: Audit Log entries are created in response to any changes made to the current values in this record. These audit records are displayed in the audit.summ form, when accessed via the Format Control option created earlier in *Setting Up Auditing from Format Control* on page 287.

If you check Audit Lookup now the following message will appear in the status bar.

2 To complete the example, modify the record created in the previous steps by entering new values in the fields of the form that were specified as Unique A-D fields in the audit specifications file (See *Defining an Audit Specifications Entry* on page 283.). For this example, change the Employee ID to GEN00004.

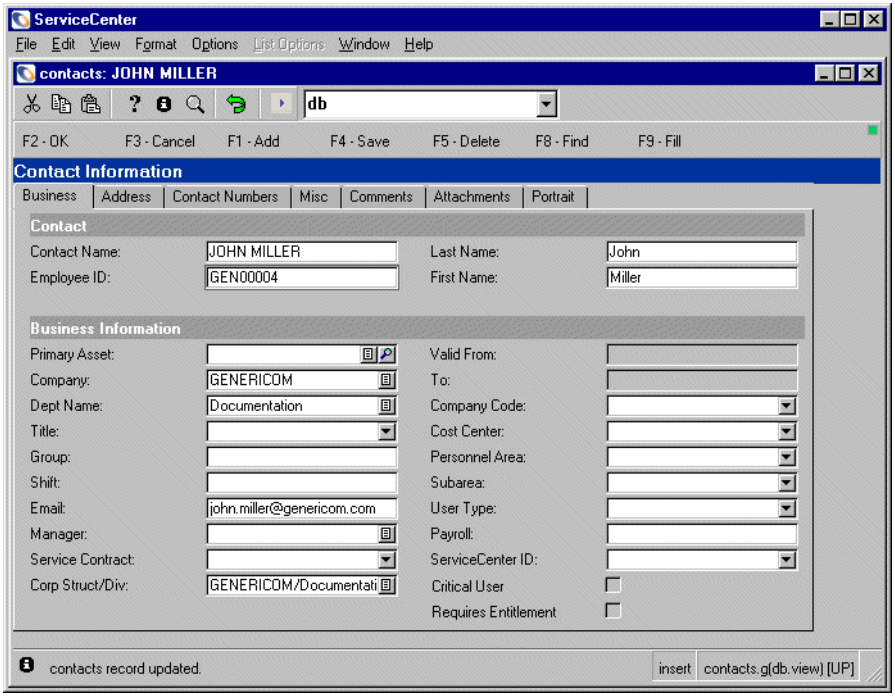


Figure 13-19: Modified fields in contacts table

3 Click **Save** to commit the changes to the database.

Note: The following message is displayed in the status bar: *Record updated in the contacts file. Audit Record successfully recorded and added* is issued when the record is saved and an Audit Log is created.

4 To view the audit log record generated by this action first deactivate the Record list option by deselecting **View** > **Record List** from the menu bar.

5 Select **Options** > **Audit Lookup** from the menu bar.

A QBE list is displayed.

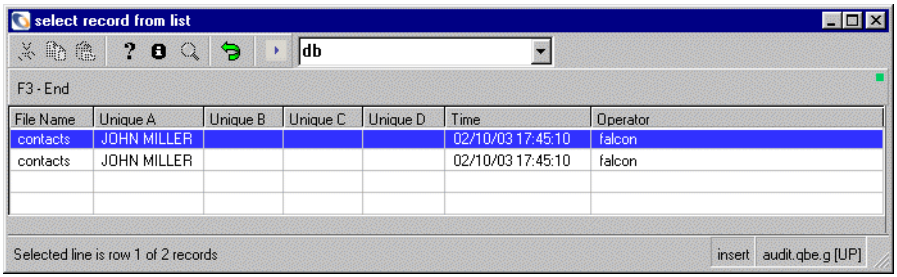


Figure 13-20: Audit Record QBE

Note: The updates you see may vary, depending on how you have your triggers set up for the file. See *Trigger Setup* on page 293.

6 Select the desired record from the list.

The Audit Log (**audlog**) record generated by the previous record update is shown in Figure 13-21 on page 301.

Note: Only those fields which were modified are recorded. Both the Old and the New versions of each modified field are displayed in the audit log record. If none of the fields defined in the Audit Specifications Table are modified, an audit log entry is not generated for the Database Dictionary file.

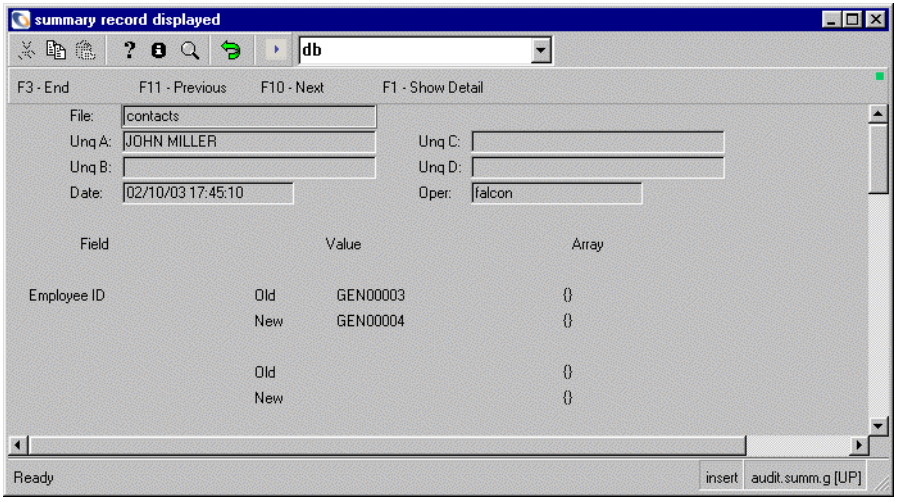


Figure 13-21: Audit Record summary

7 For further information regarding the modifications to the **contacts** file, click **show detail**.

The same record is re-displayed in an expanded format (**audit.g**), which shows more detailed information for the arrayed fields, as well as scrollable fields.

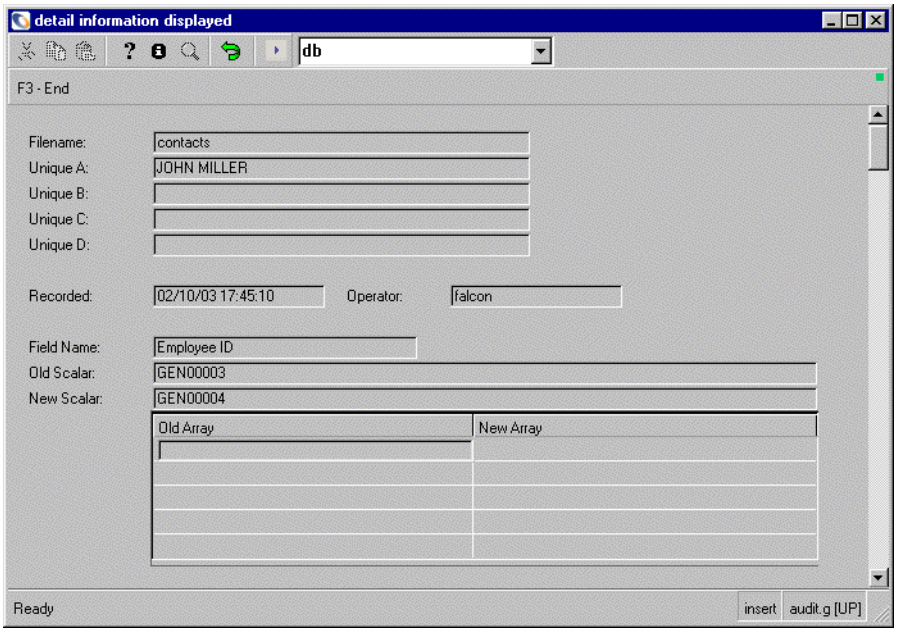


Figure 13-22: Audit Record detail

8 Click **End** to return to the summary form.

9 From the summary form, click the **Prev** and **Next** buttons to review any additional audit records in the log for this contacts record.

10 Click **F3-End** to return to the Contacts record.

14 CHAPTER Joining Multiple Tables

The data from multiple tables can be combined into a single form. For example, you want a form that displays the details of all open incident tickets, and also the contact information of the contacts who reported the incidents. The *probsummarym1* table contains all the incident details, but contains only the name and phone number of the contact. Additional information about the contact is stored in the *contactsm1* table. A join between the two tables will allow you to create a report containing information about the tickets from *probsummarym1* and the detailed contact information from *contactsm1* in a single report.

This chapter was designed to help ServiceCenter and database administrators use the Database Manager Utility t o work with multiple tables. For more information on virtual joins see the *System Tailoring Guide*.

Topics in this chapter include:

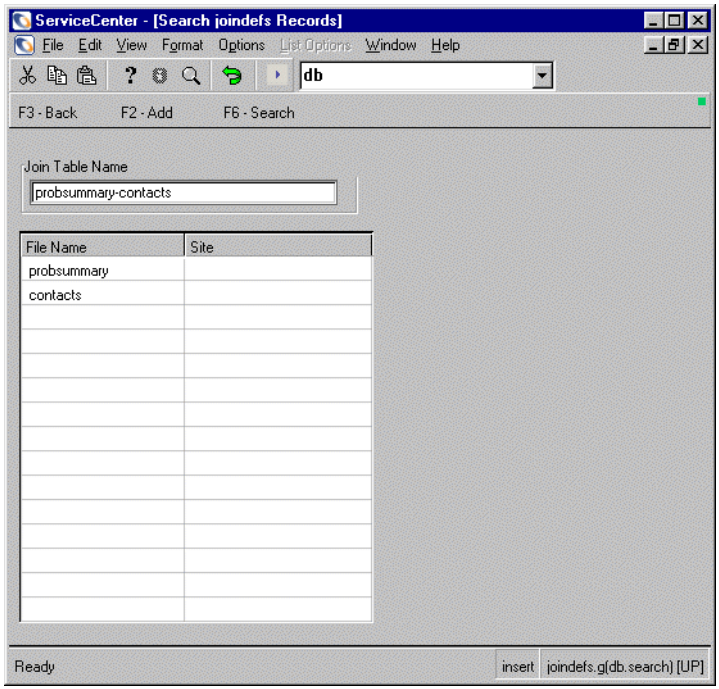
Searching for a Join

Before you create a join, you should perform a database search to make sure that the join does not already exist.

To search for an existing join:

1 Start your ServiceCenter client and open the Database Manager utility.

2 Enter joindef in the **Form** field and click **Search**.



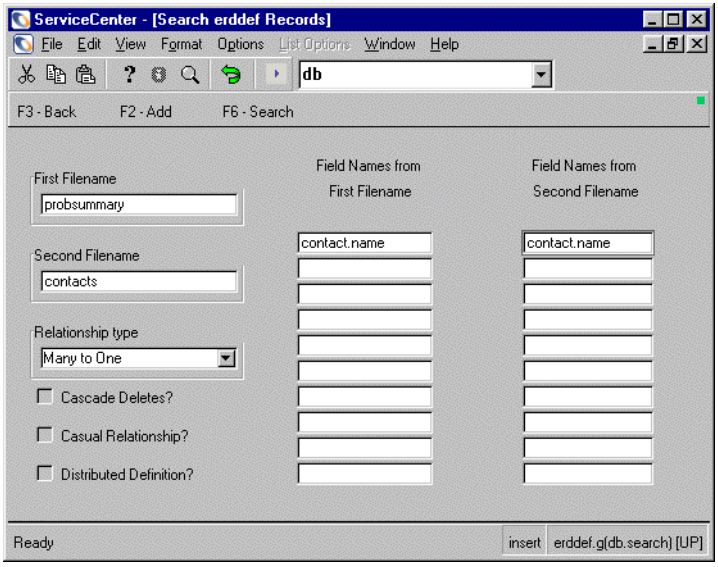
3 Search for the join you want to create by typing the name you want to give the join. Join Definition files are typically named for the tables being joined. In this example, you are looking for a join called probsummary-contacts or contacts-probsummary.

Creating a Join

To create a join:

1. Enter the name you want to use for the join in the **Join Table Name** field (keeping with the common naming convention).
2. Add the names of the tables to be joined in the fields below, starting with the upper-left field and working down in columns.
3. Click **Add**.
4. Back out to the main Database Manager selection window. Enter erddef in the **Form** field. Click **Search**.
5. Search for the ERD (Entity Relationship Diagram) Definition you are about to make. In this case, you want to search based on the *First Filename*, *Second Filename*, and *Relationship Type*.

Field	Description
First Filename	Name of the table you want to be the parent table in this join, in other words, the table that will show all of its records. In the example, this will be probsummary, since the primary goal is to show all open incidents.
Second Filename	Name of the dependent table, in other words, the table that will only show records related to each record of the parent table. In the example, the name would be contacts, since the contact information displayed is determined by the contact name for each individual incident record.
Relationship Type	**One To Many**, **Many To One**, or **One To One**. This refers to the number of times the record of one table can be connected to records from another table. For the example, the relationship would be **Many-To-One**, since a single contact can appear in multiple incident records, but an incident summary can contain only one contact name.



6 If no match is found, create an ERD definition.

a Fill in the **First Filename**, **Second Filename**, and **Relationship Type** as described above. In the **Field Names from First Filename** box, enter the field(s) on which you want to create the join (in other words, the field in the first table that is equivalent to the field in the second table). Do the same for the **Field Names from Second Filename** box. In the example, the contact.name field in the probsummary table contains the same data as the contact.name field in the contacts table.

b Click **Add**.

7 Cycle your ServiceCenter server.

Note: The join does not take effect until you shut down and restart your ServiceCenter server.

The join results in the following:

- Your report prints every record in the probsummary table that meets your selection criteria.
- For each record, the system looks in the contacts table for a record containing the same name in the **contacts.contact.name** field as contained in the **probsummary.contact.name** field.
- All requested contact information for that record is printed.

Join Types

In most relational databases, there are three major types of joins between tables. For the following illustration we will use an unfiltered search on two tables PROBSUMMARY and CONTACTS, joined on the field contact.name which is shared by both tables:

PROBSUMMARY

number	contact.name
PM1001	Falcon
PM1002	Max.manager
PM1003	(null)

CONTACTS

contact.name	dept
Falcon	Engineering
Max.manager	Sales
Susie.supertech	Customer Support

Inner Join

A join that returns only rows where both tables contain data in the shared column. The result is a subset of data from both tables. In our example, any PROBSUMMARY records with a contact.name value that is blank, null, or does not exist in the CONTACTS table will not appear. Likewise, any CONTACTS records containing a contact.name that is blank, null, or does not exist in any PROBSUMMARY record will not appear.

Hence, the result is

Prob.number	Contacts.contact.name	Contacts.dept
PM1001	Falcon	Engineering
PM1002	Max.manager	Sales

Left Outer Join

A join that displays all records from the first (left) table but only related records from the second (right) table. The result is a set of records equal to the number of records in the first table, each attached to zero, one, or more relevant records from the second table. In our example, an unfiltered search would return all records from the PROBSUMMARY table and, for each of those records, the associated record from the CONTACTS table. If the contact.name field in the PROBSUMMARY record is blank, null or contains a contact name that does not exist in the CONTACTS table, the PROBSUMMARY record will still be displayed, but related CONTACTS table information will be missing.

Prob.number	Contacts.contact.name	Contacts.dept
PM1001	Falcon	Engineering
PM1002	Max.manager	Sales
PM1003	(blank)	(blank)

Right Outer Join

The same as a left outer join, except that the second (right) table is the primary table and the first (left) table is the secondary table. The result is a set of records equal to the number of records in the second table, each attached to the relevant records (if any) from the first table. In our example, an unfiltered search would return all records from the CONTACTS table, and for each of those records, the relevant data from the PROBSUMMARY record matching the CONTACTS.contact.name field. If the CONTACTS.contact.name field is blank, null or contains a contact name that does not exist in the PROBSUMMARY table, the CONTACT record will still be displayed, but no PROBSUMMARY data will appear.

Prob.number	Contacts.contact.name	Contacts.dept
PM1001	Falcon	Engineering
PM1002	Max.manager	Sales
(blank)	Susie.supertech	Customer Support

Important: The ServiceCenter ODBC driver, as of version 3.00.0000, supports only left outer joins. All joins will be treated as left outer joins. Report designers wishing to simulate other join types should use linked subreports instead of joins to achieve the desired result.

Join Syntax

As in the case of SQL keywords, there are many different dialects for join query syntax. As of version 3.00.0000, the ServiceCenter ODBC driver supports two types of join syntax, the two defaults used by Crystal Reports Professional 6.0 and 7.0. Details are provided below. Examples assume the same join as above, between the PROBSUMMARY and CONTACTS tables on the shared field contact.name.

Syntax #1

```
SELECT
   <field1>,<fieldco2>,etc…
FROM
   { oj <t11able1>.<join field> = <table2>.<join field> }
WHERE
   <selection criteria>
```

Example:

```
SELECT
   probsummarym1.number, probsummarym1.open.time,
   probsummarym1.close.time, contactsm1.contact.name, contactsm1.first.name,
   contactsm1.last.name, contactsm1.dept
FROM
   {oj probsummarym1.contact.name=co‘ntactsm1.contact.name}
WHERE
   probsummarym1.status="open" AND probsummarym1.category="hardware"
```

Syntax #2:

```
SELECT
   <field1>,<field2>,etc…
FROM
   <table1>,<table2>
```

```
WHERE
    <table1>.<join field> = <table2>.<join field>, <selection criteria>
```

Example:

```
SELECT
    probsummarym1.number, probsummarym1.open.time,
    probsummarym1.close.time, contactsm1.contact.name, contactsm1.first.name,
    contactsm1.last.name, contactsm1.dept
FROM
    probsummarym1, contactsm1
WHERE
    probsummarym1.contact.name=contactsm1.contact.name,
    probsummarym1.status="open" AND probsummarym1.category="hardware"
```

15 CHAPTER File Maintenance

This chapter was designed to aid ServiceCenter system and database administrators in maintaining their database files.

Topics in this chapter include:

Resetting a Database file from Database Manager

In some scenarios you may need to remove all records from a ServiceCenter database file. For example, you may want an empty **production** file when moving a file from development into production, or you may need to maintain the size of files which continually grow (such as **syslog** or **msglog**). Use the **Reset** option to remove all records from a file.

Warning: Reset removes all records in a file. Therefore caution should be used when choosing this option.

The following example demonstrates resetting the **syslog** file.

To reset the syslog file:

1 Open the **=syslog** form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank **syslog** form is displayed.

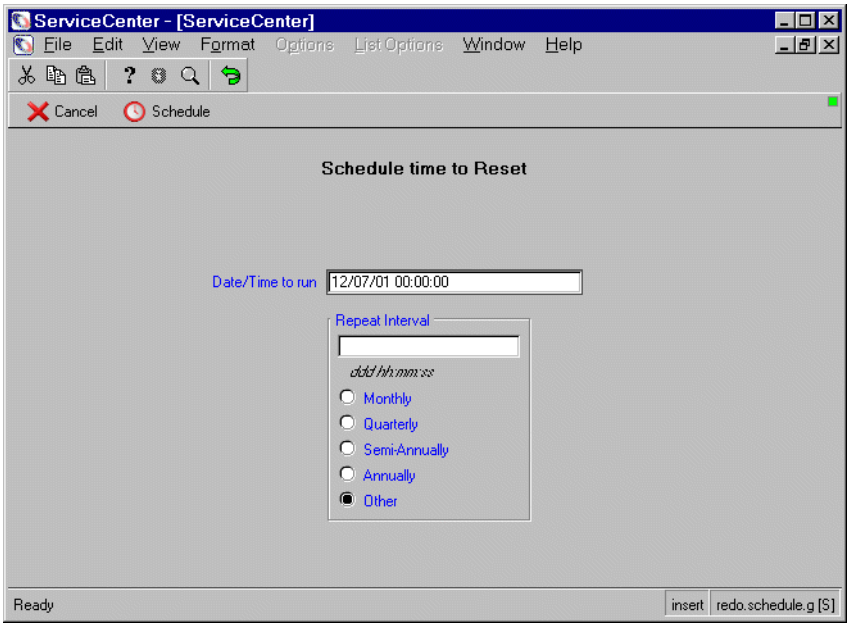


Figure 15-1: Blank System Log file

2 Select **Options** > **Reset** from the menu bar.

A prompt is displayed, asking you to confirm the action and allowing you to schedule the Reset.

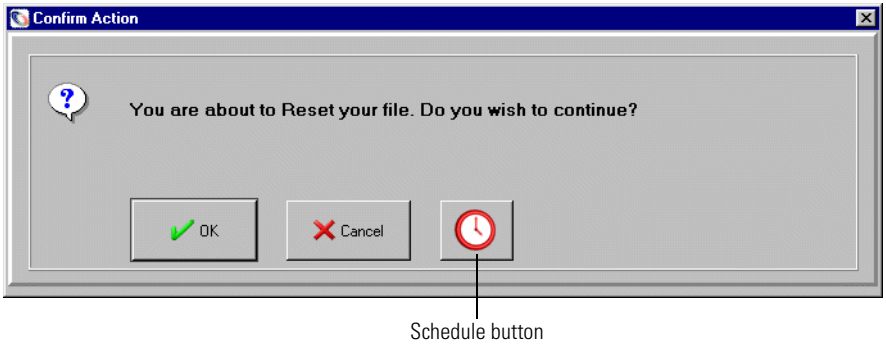


Figure 15-2: Reset prompt

3 From the prompt, you can:

- Click **OK** to reset the syslog file.
- Click **Cancel** to leave the file intact and return to the blank syslog format.
- Click the **Schedule** button to schedule a time to run the file reset operation, either once or repeatedly at a set interval.

Note: Scheduling a reset operation is discussed in the following section.

Scheduling a Reset

If you select **Schedule** from the reset prompt (Figure 15-2 on page 313), the schedule form is displayed.

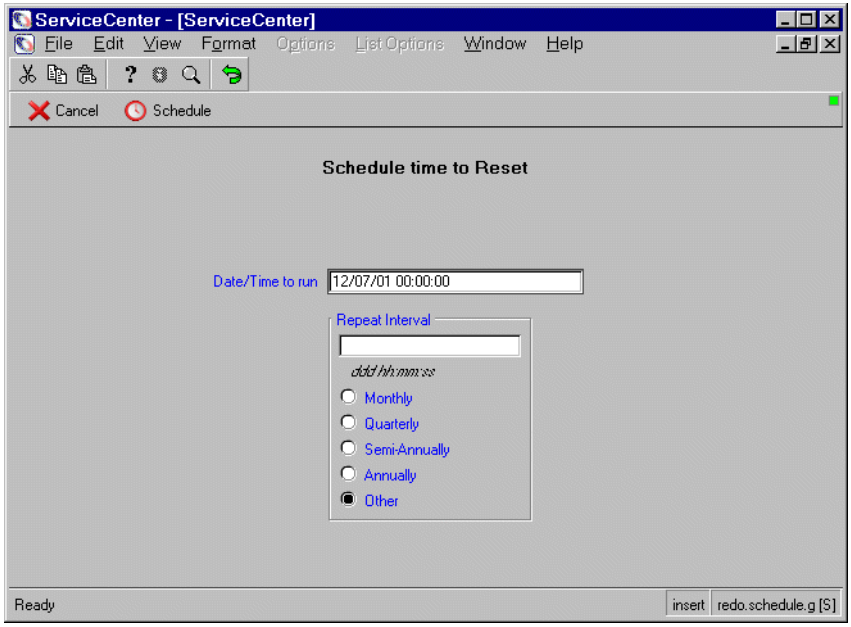


Figure 15-3: Scheduling form for resetting a file

1 Provide a date in a *DD/MM/YYYY* format in the **Date/Time to run** field.

2 Provide a time in an *HH:MM:SS* format the **Date/Time to run** field.

3 Select a **Repeat Interval** option if the file needs to be reset on a regular basis. This setting is optional. The interval period starts from the time and date set in steps 1 and 2

- **Monthly** — reset once a month.
- **Quarterly** — reset every three months.
- **Semi-annually** — reset every six months.
- **Annually** — reset once a year.
- **Other** — reset in the specified number of days, at the specified time. Use the *ddd hh:mm:ss* format, where *ddd* is the number of days from the initial date and time, and *hh:mm:ss* is the time of day at which the reset is run. For example, `26 10:00:00` schedules the reset to run at 10 a.m., 26 days from the initial reset date and time set in the Date/Time to run field.

4 Click **Schedule** to confirm this reset action.

Regenerating Database Keys from Database Manager

Keys provide efficient, organized access to records in a table. They define a hierarchical tree of indexes associated with the actual data records. Key regeneration means that the existing index tree for a table is discarded and then regenerated from scratch. The regeneration process involves examining every data record in the table and adding indexes to the recreated index tree for each record.

Key regeneration is a time consuming process taking up to several hours for very large databases.

Under most circumstances, keys are regenerated from within the Database Dictionary Utility whenever keys are added or modified in the Database Dictionary record. The key regeneration function is also available from within Database Manager.

If you want to regenerate the keys to the contacts file, be sure that other users are not accessing the file; the regen procedure will interrupt all activity in progress on the contacts file.

Note: Regeneration of all files can also be done from the Database Dictionary Utility. See the *System Tailoring* for details.

To regenerate the keys to the contacts file:

1. Open the contacts form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

   A blank contacts form is displayed.

2. Select **Options** > **Regen** from the menu bar to start the regen for the contacts file.

   A prompt is displayed, asking you to confirm the action and allowing you to schedule the Regen.

3. From the prompt, either:
   - Click **OK** to continue with the file/key regen.
   - Click **Cancel** exit this screen without doing a regen.
   - Click the **Schedule** button to schedule a time to run the file regen operation, either once or repeatedly at a set interval.

   Note: Scheduling a regen operation is discussed in the following section.

Scheduling a Regeneration

If you select Schedule from the regen prompt:

(The regen prompt is shown in Figure 15-2 on page 313.) The schedule form is displayed.

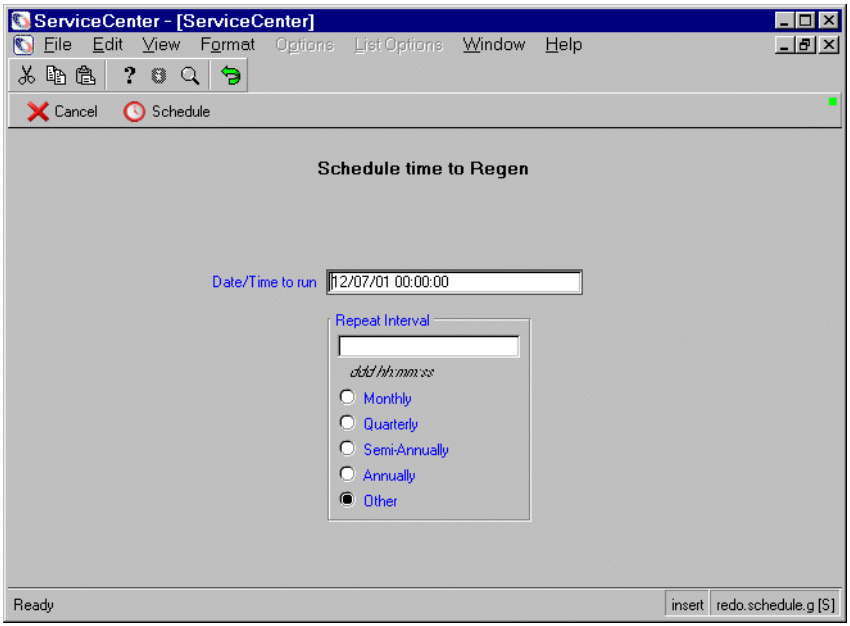


Figure 15-4: Scheduling a regen time

1. Provide a date in a *DD/MM/YYYY* format in the **Date/Time to run** field.
2. Provide a time in an *HH:MM:SS* format in the **Date/Time to run** field.
3. Select a **Repeat Interval** option if the database needs to be regenerated on a regular basis. This setting is optional. The interval period starts from the time and date set in steps 1 and 2
   - **Monthly** — regenerate once a month.
   - **Quarterly** — regenerate every three months.
   - **Semi-annually** — regenerate every six months.

- **Annually** — regenerate once a year.
- **Other** — regenerate in the specified number of days, at the specified time. Use the *ddd hh:mm:ss* format, where *ddd* is the number of days from the initial date and time, and *hh:mm:ss* is the time of day at which the regen is run. For example, `26 10:00:00` sets the regen to run at 10 a.m., 26 days from the initial regeneration set in the **Date/Time to run** field.

4 Click **Schedule**, to confirm this file/key regeneration action.

16 CHAPTER IR Expert

This chapter was designed to provide system and database administrators information on IR Expert setup, configuration and implementation.

Topics in this chapter include:

Overview

IR Expert is an intelligent, concept-based information retrieval (IR) engine that searches the ServiceCenter (SC) database for similar/related information based on a simple, natural language query. Instead of relying on exact-match keywords to select like incidents from the ServiceCenter Incident Management database, for example, the description of an Incident is used to locate similar incidents via IR Expert. The records called up are then assigned a probability of relevance, allowing retrieved documents to be ranked in order of relevance to the original query.

Any ServiceCenter file is a candidate for IR queries if the information in the file is stored in array or extended text fields. Some suggested files are:

File	IR Keys
probsummary	action, update.action, resolution, key.words
msglog	msg.text
help	brief, detail, keywords
abendcodes	explanation, system.action, operator.response, programmer response
application	comments, comments.more

Note: Certain files are already set up with IR keys in ServiceCenter. These include **probsummary**, **incidents**, **probcause**, and **knowledge**.

Special Considerations

Several issues are critical to the implementation and administration of IR Expert:

- IR Expert relies heavily upon shared memory. Refer to the *Installation and Technical Reference Guide* for additional information.
- IR Expert files must be kept in sync with ServiceCenter database files **scdb.fre**, **scdb.asc**, **scdb.lfd**, and **scdb.db1**. For example, if you restore IR Expert database files from a weekly backup, you must restore the ServiceCenter database files from the same weekly backup.

- The IR Expert database files are not currently portable across different system architectures. When moving from one platform to another, you must rebuild IR indexes on the new platform using IR-Only Regen.
- When configuring security application such as RACF, ACF2, and TOPSECRET, take into consideration that the ServiceCenter job or started task modifies the IR Expert files.

How IR Expert Ranks the Documents it Finds

IR Expert queries return results based on relevance to the query. To do this, IR Expert looks at each term used in an IR query and gives a ranking to the term, based on its commonness in the stored documents.

A term that is used in a lot of documents has a lower weight than a term that is only used in a few documents. For example if all of the problems being reported involve the windows operating system then the term "window" is probably used in each and every document and therefore would have a very small weight. A term that is used in only one document would have a higher weight.

Once the terms are assigned their weights then the stored documents are given a weight based on how many of the terms used in the query are used in the document and how often a term is used in the document. A document that contains a given term twice will have a greater weight than one that only contains the term once. The terms in the document are then compared with the terms in the query to see if there is a "phrase" match and if so then the document is given a higher weight. Everything else being equal, the document that was updated most recently will be the most relevant.

Operational Tasks

Term operations are enforced on all queries run on the ServiceCenter database enabling automatic indexing based on key terms. These include:

- Lexical analysis — process of converting an input stream of characters into a stream of or words or tokens.
- Stemming — automated process of combining related words, based on the reduction of words to their common root, stem, form.

- Pruning stop words — list of words considered to have no indexing value, used to eliminate potential terms appearing in query statements. If a word from the stop list appears in a query, that word is ignored in the search.
- Spelling correction — automated process of verifying and correction of the spelling of query terms, ensuring the query is processed correctly.

Lexical Analysis

Symbol	Description
digits	Numbers do not make good index terms, and are not usually included as tokens. In some instances however, (e.g. File Number) query statements consisting of only digits need to be passed. IR Expert therefore indexes digits along with alphabetic characters.
hyphens	Words broken at the end of lines or including hyphens can result in multiple word fragment tokens. To avoid this IR Expert considers hyphenated terms as a single token and does not break them apart.
other punctuation	Similar to the use of dashes/hyphens, other punctuation, including periods (.), commas (,) and underscores (_) are often used as parts of terms. IR Expert allows apostrophes (’), dashes (-) and periods (.) to appear *within*, but not at the *beginning* or *end* of a token.
case	Though case distinctions are important in some cases, e.g. programming languages, for the purposes of IR Expert, operating on the ServiceCenter database, all terms are converted to lower-case. IR Expert is case in-sensitive, as a result.

Stemming

Stemming allows the user to find the variants of a term, while reducing the size of the index file. Since single stems typically correspond to several full terms, storing stems instead of full terms, allows a compression factor of over 50 percent.

IR Expert uses an English suffix removal stemming algorithm. A simple example is seen in the removal of plurals:

- If a word ends in *ies*, but not *eies* or *aies*, then change *ies* to *y*.
- If a word ends in *es*, but not *aes*, *ees*, or *oes*, then change *es* to *e*.
- If a word ends in *s*, but not *us* or *ss*, then remove s.

IR Expert also employs a more complex algorithm from Porter (1980), consisting of a set of condition/action rules. The conditions fall into three classes: conditions of the stem, conditions of the suffix and conditions on the rules.

Pruning Stop Words

The use of the stop words file naming convention in IR Expert does away with the need for an extensive list of English words, where both the most frequently occurring and least useful for intelligent data retrieval are identified. This utility is explained in greater detail in *File Management* on page 324.

Spelling Correction

The primary issue with spelling correction is to identify when an input string is significantly close to one of a set of given strings. When a user enters a query, after lexical analysis and the other conditions are performed, IR Expert attempts to identify words in the index that are close to the unrecognized word. The problem is to attain good selectivity while still exploring large databases in a timely manner. This is achieved with IR Expert via two tests.

First, all existing index terms are compared to any unidentified query terms, taking into account order of letters. This is referred to as a *shallow match*, and relies upon the identification of same letters in both words, ideally seeking a distance of zero (0) between the words/letters. Only words with a distance of two (2) or less are passed to the second test.

Secondly, a test prunes obviously different words with merely similar letter arrangement, e.g. bushland for husband. *Deep matching* is enacted in this test, verifying letter order within words. Words with the lowest distance are considered as corrections for any unrecognized words.

File Management

The following table summarizes information about the IR Expert files.

Description	Purpose	Naming Convention
database index	Dynamically-allocated system files maintained exclusively by IR Expert. One database index file exists for each ServiceCenter file containing an IR Key. Contains word and record entropies and relationships.	[ir_prefix]irnnnnn.dc where ir_prefix corresponds to the start-up parameter, and nnnnn is the internal ServiceCenter file number.
database map	Dynamically-allocated system files maintained exclusively by IR Expert. One database map file exists for each ServiceCenter file containing an IR Key Contains mappings between ServiceCenter and IR Expert record numbers.	[ir_prefix]irnnnnn.map where ir_prefix corresponds to start-up parameter, and nnnnn is the internal ServiceCenter file number.
stop words	Required user file maintained by the system administrator. Contains words that have little or no value to the information retrieval process. For example, the preposition *with* is a stop word. Stop words can be added or deleted as necessary. Changes take affect when ServiceCenter is restarted and indexes are regenerated.	[ir_languagefiles_path]language.stp where ir_languagefiles_path and language correspond to start-up parameters.
stem dictionary	Required system file for languages other than *english* and *german*. Contains word stems from which derivative words are formed, allowing IR Expert to match closely related words. Maintained exclusively by IR Expert.	[ir_languagefiles_path]language.stm where ir_languagefiles_path and language correspond to start-up parameters.

Description	Purpose	Naming Convention
suffix dictionary	Required system file for languages other than *english* and *german*. Contains suffix templates used in stemming. Maintained exclusively by IR Expert.	[ir_languagefiles_path]language.suf where ir_languagefiles_path and language correspond to start-up parameters.
normals dictionary	Required if the language employs special keyboard characters. Normalization characters can be added or deleted as necessary. Changes take affect when ServiceCenter is restarted and indexes are regenerated. The excerpt below shows a typical normalization file. The first two characters of each line become substitutions for the following character or comma-separated characters (in decimal notation). ae 132,142 oe 148,153 ss 225 ue 129,154	[ir_languagefiles_path]language.nor where ir_languagefiles_path and language correspond to start-up parameters.

Updates to IR Files

IR files can be updated in two modes:

- **synchronous mode**

  The default mode. The IRQUEUE processor is not used. All IR Expert updates are immediately written into the IR Expert files.

- **asynchronous mode**

  Changes made to the IR key are placed in a queue and are not processed immediately. A separate background process (IRQUEUE), writes the accumulated updates into the IR index files. The changes are then available for searches.

Query response time will be faster when using Asynchronous IR. In asynchronous mode, updates to files that have an IR key do not have to wait for the completion of IR queries that are executing at the same time.

Note: If the IRQUEUE process is not running for any reason (including during a hot backup), changes to IR index data are not available to users. IR files do not reflect the newest IR index data and therefore will not retrieve newly added data. However, searches will work against existing IR data.

Starting IR Asynchronous Mode

IR Asynchronous Mode is turned on by adding the ir_asynchronous parameter to the sc.ini file of the ServiceCenter server, and starting the IRQUEUE process.

The IRQUEUE process checks if the queue processor is already running. If it finds the process, then a new queue process is not started.

Important: If the IRQUEUE process is not running, the IR files will not be updated.

To have the IRQUEUE process started at system startup:

1 In Unix, add the following line to the scstart script:

```
scenter -que:ir -forceque &
```

In Windows and OS/390 add the following line to the configuration file (the sc.cfg file or the CONFIG data set):

```
scenter -que:ir -forceque
```

Note: The forceque parameter forces the process to start, whether or not it is already running. If you do not want to force the queue to start, you do not include this parameter.

2 Restart the ServiceCenter server.

Refer to *24x7 Backup and IR Expert Files* on page 39 and the *Installation and Technical Reference* for more information on this and other ServiceCenter parameters.

IR Files and Hot Backup

Using Hot Backup does not prevent writes to the ir.* files. Writes to the ir.* files can result in a corruption of these files if they are backed up while I/O is occurring on them. To prevent this, the ir_asynchronous parameter should be turned on if Hot Backup is used. Refer to *Hot Backup* on page 34 and *Starting IR Asynchronous Mode* on page 326 for more information.

Note: The IRQUEUE process is automatically stopped when a Hot Backup starts and is restarted when the Hot Backup is complete.

Customizing IR Expert for Foreign Languages

IR Expert can be set up to perform efficiently in any language with the use of stop words, and normals, stem and suffix dictionaries. The following files contain these items:

The stop word file, [ir_languagefiles_path]language.stp, contains words used so frequently in the language they should not be indexed since they do not help identify documents. Each word should be entered on a separate line in this file.

The words in the file go through a stemming process, which eliminates the need to enter all the forms for the word. For example, in English, if you did not want to index **go,** you would not have to enter both *go* and *going* into the stop words (.STP) file since the stemming algorithm changes *going* to *go* anyway. The only word which needs to be entered in the .STP file is the term *go*.

- The **normals dictionary**, [ir_languagefiles_path]language.NOR, is only involved when there are characters in the language that need to be transformed into other characters. For example, in the German language the umlauted characters are changed, the **ä** (a-umlaut) is changed into **ae**. You many want to do this to make setting up the stem (.STM) and suffix (.SUF) dictionary files easier.

- The **stem dictionary**, [ir_languagefiles_path]language.STM, contains the stem which is that part of the a term used in the IR indices. Each word is considered to have a *stem* (defined in the .STM file) and many possible suffixes (defined in the .SUF file). In the .STP explanation above there was an example of *go* and *going;* **go** is the stem and **ing** is the suffix. Entries in the .STM file consist of the stem word (*go*) followed by a blank, then an index into the suffix file (.SUF), e.g. *go 1*. This index indicates which suffix values are acceptable for the stem word.
- The **suffix** file, [ir_languagefiles_path]language.SUF, contains a series of lines, each a list of valid suffix values. The .STM file indicates which line in the .SUF file should be used as the possible suffixes for any given stem word, e.g. for the stem **go** suffixes might be *ing, es, ne*.

File Management Example

For the following example it will be said a particular user only wants to cover stemming for *take, ride* and *walk*.

- The acceptable forms of **take** are *take, taken, taking*.
- The acceptable forms of **walk** are *walk, walking, walked*.
- The acceptable forms of **ride** are *ride, ridden, riding*.

The stem dictionary (.STM file) might contain the following set up:

- tak 1 (words with this stem will use the first suffix option)
- rid 1
- walk 2 (words with this stem will use the second suffix option)

The suffix dictionary (.SUF file) would contain:

- e, en, ing, den
- ing, ed

With the above files:

- *take, taken, taking* would result in tak.
- *walk, walking, walked* would result in walk.
- *ride, ridden, riding* would result in rid.

These files are not perfect. For example *takden* would be changed to tak because the same suffix index was used for tak and rid.

The configuration could be changed so that the stem dictionary (.STM file) contained:

- tak 1
- rid 3 (words with this stem will use the third suffix option)
- walk 2

The suffix dictionary (.SUF file) contained:

- e, en, ing
- ing, ed
- e, den, ing

Important: Setting up these language support files requires a considerable amount of time. It should only be undertaken by someone fluent in the language, and knowledgeable of word components and pronunciation.

Implementing Foreign Language Files

Use the following steps to enter foreign language files into the IR system. (Spanish file are used in this example.)

To implement a foreign language file:

1. Once the proper management files have been created, place ir_language:spanish into the sc.ini file.
2. Place *spanish.stp* (stop words), *spanish.stm* (stem dictionary), *spanish.suf* (suffix dictionary) and *spanish.nor* (normals dictionary) in a unique directory.
3. Point the ***ir_languagefiles*** parameter at the directory containing these language files.

Accessing IR Query

The IR Query application can be directly accessed when browsing, opening, updating or closing incidents, service management calls, and when using the Database Manager.

Refer to the ServiceCenter *User's Guide* for detailed information on accessing and using Service Management and Incident Management. Refer to the ServiceCenter *User's Guide* for instruction on accessing IR Query when browsing, opening, updating or closing incidents, service management calls. For detailed information regarding the Database Management applications, see *Data Retrieval* on page 141.

- From the Service Management (Help Desk) main menu the **Search Knowledge Base** option is available as a button.
- From the Call Queue, the IR Query parameters are available on the search form called with the **Search Calls** option in the **Options** menu, or the Search button on the main portion of the screen.
- From the call open, update and close screens, IR Query is available through the **Find Solution** option in the **Options** menu.
- From the Incident Management main menu the **Search Knowledge Base** option is available as a button.
- From the Incident Queue the IR Query parameters are available on the search form called with the **Search Incidents** option in the **Options** menu, or the **Search** button on the main portion of the screen.
- From the Incident open, update and close screens, IR Query is available through the **Find Solutions** option in the **Options** menu.
- From the Database Manager application, **IR query** is available from any form as an option in the **Options** menu.

Once IR Expert has been accessed from a record, you must manually choose which database on which to perform the query, and on a separate screen insert the specific query to run.

Accessing IR Query from the Database Manager

To perform an IR query from any form:

1 Open the form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

A blank form is displayed.

2 Select the **IR Query** option in the **Options** menu.

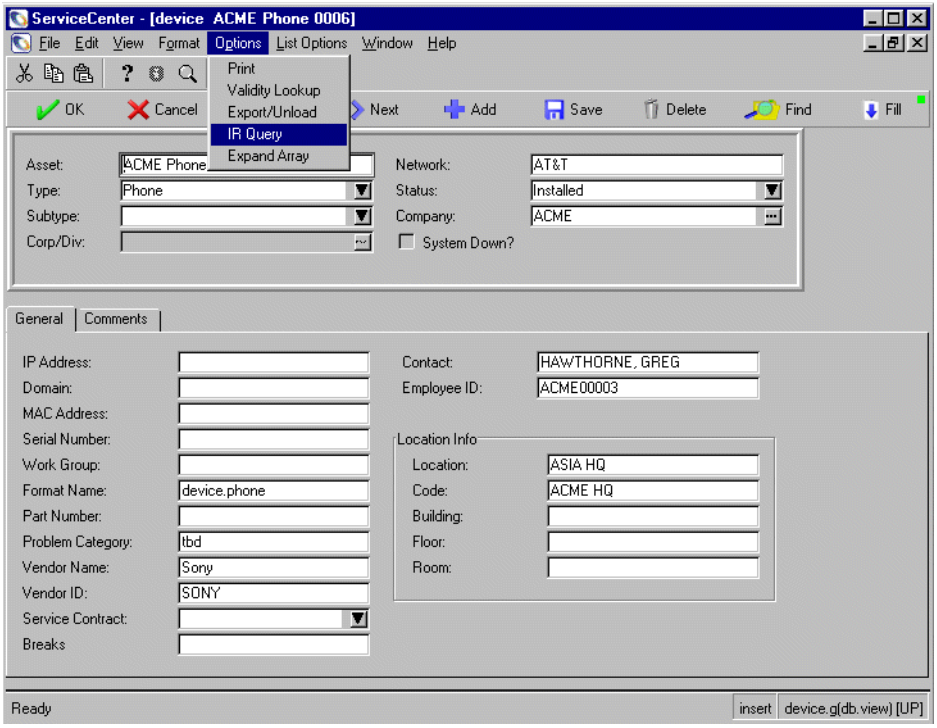


Figure 16-1: IR Query option in Database Manager

There is no default IR file defined in Database Manager; IR Query will prompt for a file name and a query statement.

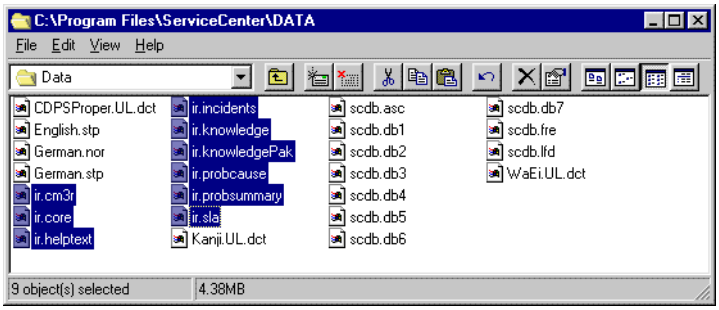
Creating an IR File

An IR file is a standard ServiceCenter file that contains information captured in other ServiceCenter files — specifically in multiple-page files like **problem**, **cm3r** and **cm3t** — and is used only for information retrieval.

The IR file is created just like any other ServiceCenter file. However, it must contain an IR key in addition to the key that is used in the link query. The field names do not have to be identical, but in the IR files accessed by the **probsummary**, **cm3t**, and **cm3r** files a **number** field must be the unique key. (See *Key Definitions* on page 21.) A sample **ir.probsummary** file is contained in the Incident Management module. This file, and all **ir.<filename>** files,

appear in the ServiceCenter **Data** directory. These files are created automatically when you perform an IR Regen on Database Dictionary files which contain IR keys and when records are added or updated to files containing IR keys.

For complete information on building ServiceCenter files, refer to the *System Administrator's Guide*.



Building An IR Key

You define an IR key using the standard Database Dictionary utility in ServiceCenter. An IR key is composed of one or more array or scalar text fields. IR keys that combine array and scalar fields should define an array field as the first element of the key.

Only text fields should be used in an IR key. A text field is an array or scalar field that contains arbitrary information that would not be used for traditional queries. For example, the device file contains a scalar field called description. This field contains descriptive text about a device, such as this PC's display doesn't support the new drawing package or this modem needs ventilation on the top and sides.

The assignment file, on the other hand, contains an array field called operators. The operators array contains the exact login name of ServiceCenter operators, and is not a good candidate for an IR key since IR searches are relevance searches.

When defining IR keys, use the following two rules:

- If a query contains a field that is part of an IR key, an IR search is always performed. Records are selected and presented based upon relevance.
- If a query contains a field that is part of an IR key, any sort criteria you supply is ignored in version 1.4, but sort criteria is honored in version 2.0 and later. IR queries always sort by relevance.

IR queries can be combined with traditional queries to specifically limit the answer set. In early versions of ServiceCenter it was also necessary to modify the IR parameters to return more records in its answer set if you routinely used combination queries.

Refer to *Data Retrieval* on page 141 in for detailed information on manipulating keys and maintaining the Database Dictionary.

Database Dictionary and IR Expert

The Database Dictionary utility has three differences when IR expert has been installed in a ServiceCenter system:

- The key type *IR Key* is available in the key list.
- The regen type *IR Regen* is available when keys in a file that contains an IR key have been modified.
- The **Type** field in the **datadict.g** record is updated to reflect the presence of an IR key in a file.

For information on how to do a regen, see *System Tailoring*.

Note: An IR Regen does not necessarily rebuild any indices other than IR indices; if any key other than an IR key is modified, ServiceCenter may not recognize the change unless a full regen has been performed. IR Regens are several times slower than P4 regens; the average IR Regen time is 1 minute per 2,500 records.

Data Policy information (**datadict** file) is a critical factor in the IR Query application. It stores details on which files contain IR keys and which fields within those files are keyed for IR queries. Access the **datadict** file by clicking Data Policy on the **Toolkit** tab of the system administrator's main menu. (Figure 9-1 on page 185)

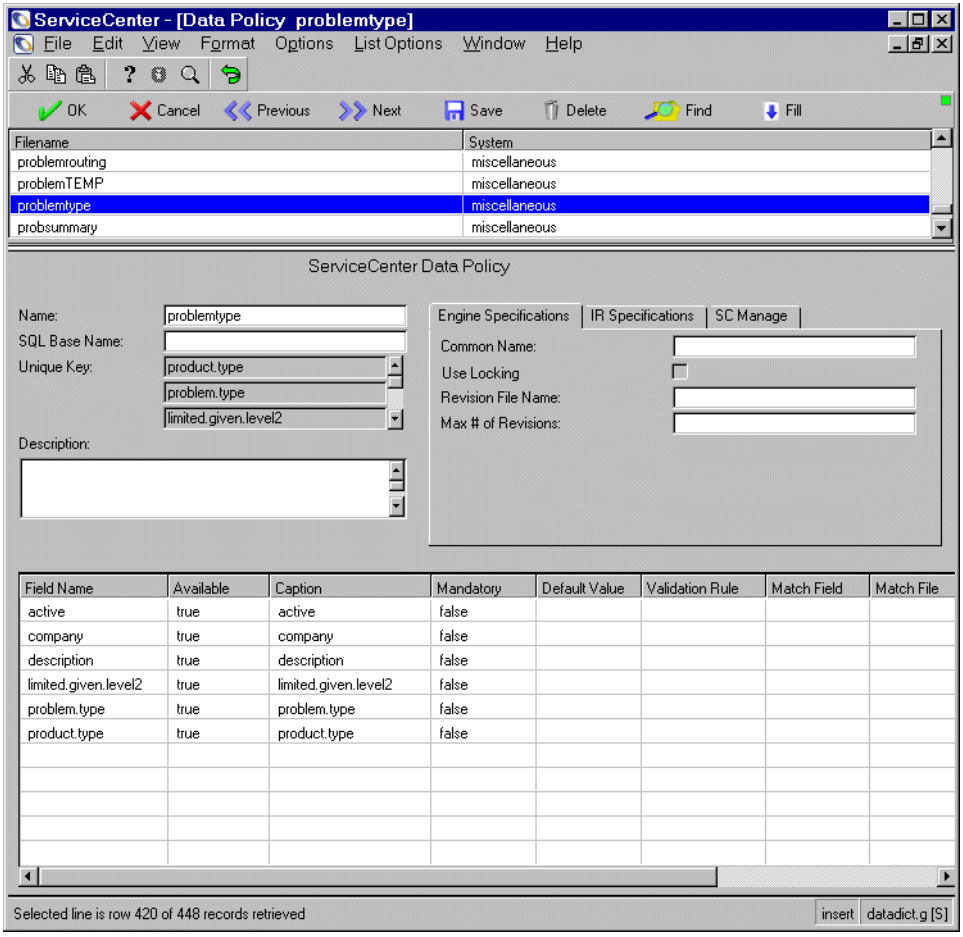


Figure 16-2: Data Policy record

IR/~IR keys

A ServiceCenter file can contain only one IR key, and must contain a non-IR key. All files that contain an IR key must also contain another key that is *unique*, with *no nulls*, *no duplicates* or *nulls & duplicates*. (See *Key Definitions* on page 21.) An attempt to build an index for a file that contains only an IR key results in a regen error. The external IR indices are built, but the IR key is removed from the ServiceCenter index.

Loading Data Files with IR Expert

Any files containing an IR key that are created when loading data into ServiceCenter (such as when moving files from a test system to a production system) will require an IR Regen in the production system.

If the file already exists in the production system and the file contains at least one record, you can add the IR key and perform the IR Regen before loading the records from your test system. Then the IR indices are updated as each record is added during the load.

Multiple Files Containing IR Keys

The current release of ServiceCenter uses shared memory, and does not specify a practical limit on the number of files you could have containing IR Keys. The most frequently referenced data is cached, making the number of files that have IR keys less of an issue.

If you want multiple fields within the IR key, there is no real impact to IR performance. ServiceCenter takes all fields defined as part of the IR key and concatenates them together for IR processing. It takes more time to concatenate five fields, than one field, but the difference is negligible.

If you want multiple files with IR keys, they will compete for use of the shared memory cache. In this scenario, you would want to allocate more shared memory as you increase the number of files with IR keys.

The best approach however, is to use the Knowledge Engineering Application. This allows all Knowledge data in the corporation to be placed within a single file (core) for IR processing. With this, you no longer need to know which file you should search to find a solution to your problem.

Find Solution

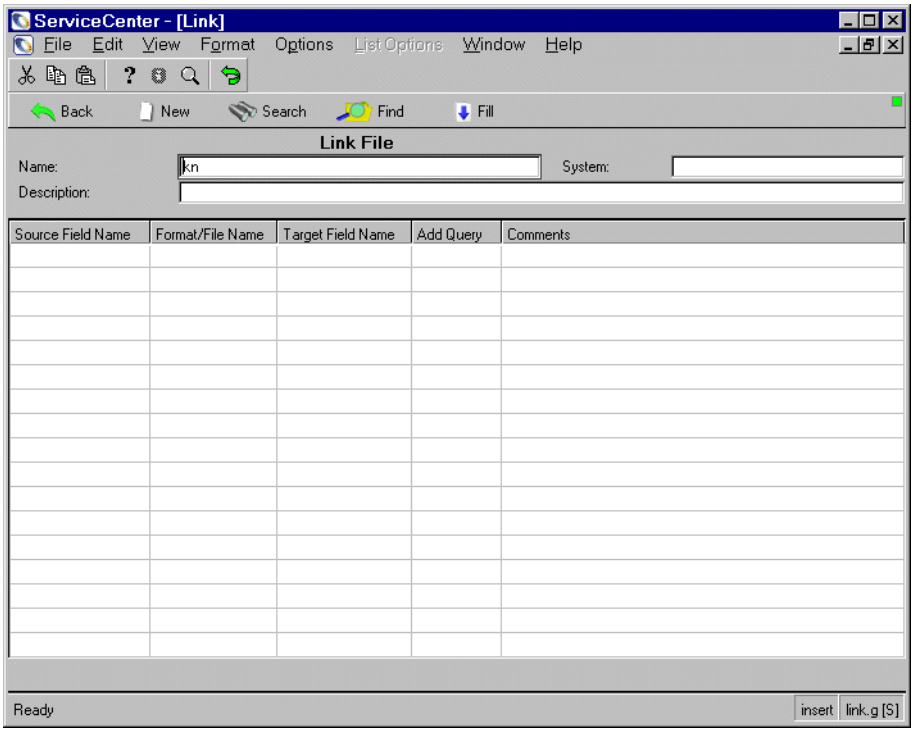
Find solution is an interface to Known knowledge. It is a global repository for information regarding ServiceCenter. Information can be input, retrieved and flagged for specific importance.

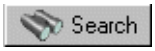
There are predefined fields to be used when querying in find solution. These fields are set in the link file. The queries can be used out-of-box, or refined by an administrator.

To edit queries for Find Solution:

1 Open the link form in Database Manager. (For instructions, see *Accessing a Record from the Database Manager Utility* on page 185.)

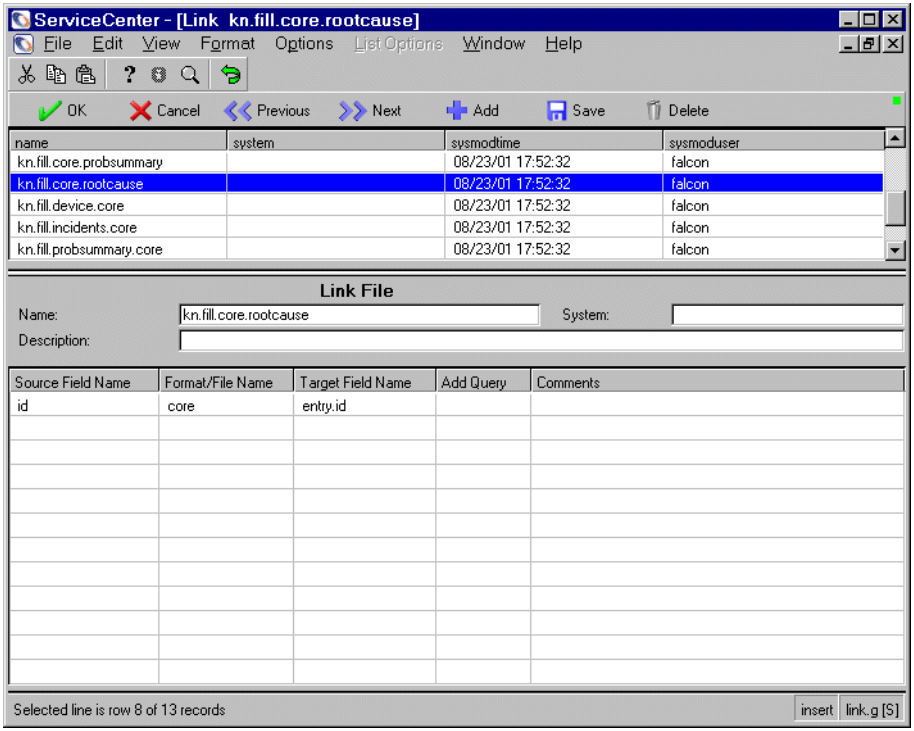
A blank link form is displayed.





2 Enter kn in the **Name** textbox and click **Search** or press Enter.

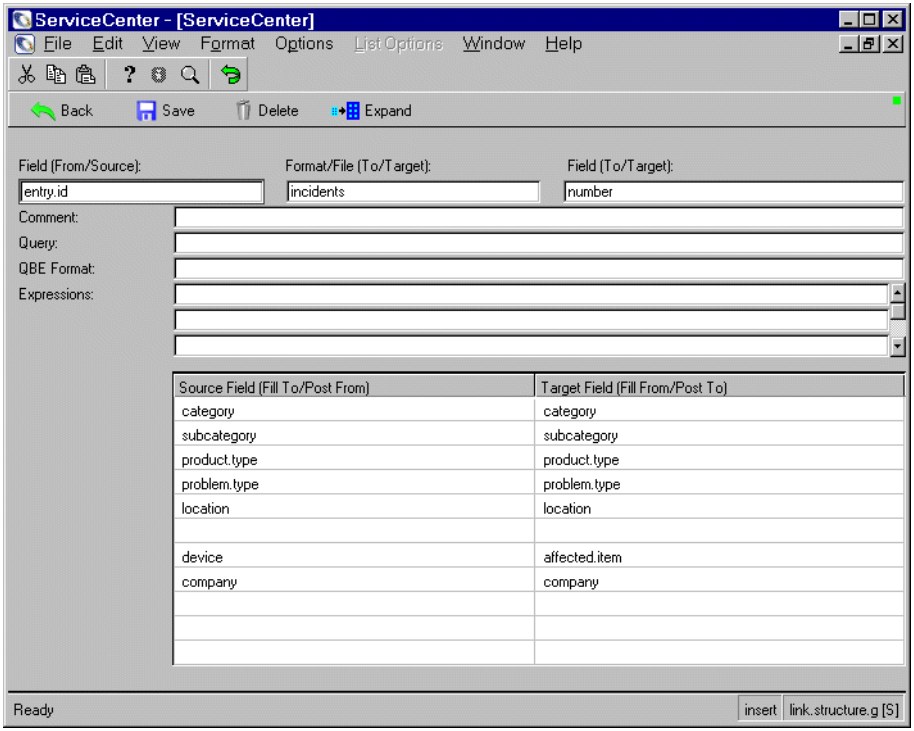
All records starting with kn will be returned.



3 Select the link to be edited.

4 Place your cursor in the Source Field Name text box.

5 Select **Line** from the ServiceCenter **Options** menu. The link structure file for the selected link record is displayed.



6 Enter the appropriate information to edit the query. For more information on Links, see *System Tailoring*.

Field	Data
Field (From/Source)	Unique key in core file (See *Key Definitions* on page 21.)
Format/File (To/Target)	Name of file to be filled from or to
Field (To/Target)	Unique key in change file
Source Field (Fill To/Post From)	Query
Target Field (Fill To/ Post From)	Query

7 Save the query and click **Back** to exit.

17 CHAPTER Using Joined Queries

This chapter was designed to aid advanced ServiceCenter administrators, implementers, or developers implement joins in ServiceCenter.

Topics in this chapter include:

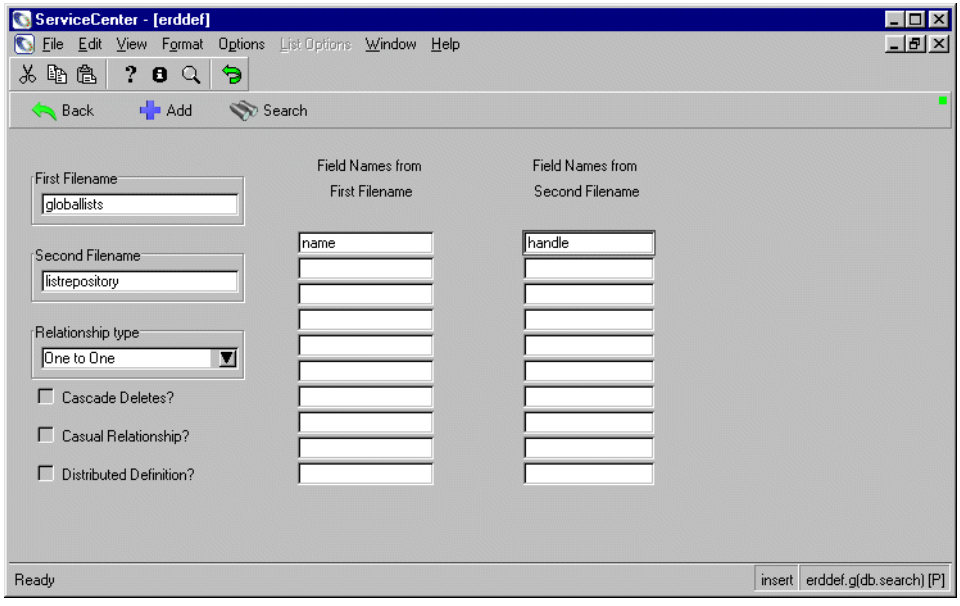
Introduction

A joined query is a query that is executed against, and returns data from, two different ServiceCenter files. For example, in ServiceCenter's inventory system, each device in inventory is stored in two separate files. Part of the data is stored in the **device** file. Other data is stored in the category-specific **attribute** file, for example, the **PC** file or the **server** file.

If you want to find all PCs located in Ohio (stored in the **device** file), which have 32 MB of memory (stored in the **attribute** file), you have two possible courses of action. The first is to execute a query to find all records in **device** which are PCs and are located in Ohio. You could then query all matching records from the PC file and test to see if they have 32 MB of memory. This, however, is a slow manual process. The second way to find this information is with a joined query. With ServiceCenter's joined queries, you can execute one query that will be evaluated against both files.

Defining a Relationship

The first step in defining a joined query is to define a relationship between the files that you want to query. This is done by adding a record to the **erddef** file. A typical record is shown below.



This record indicates a relationship between the globallists and the listrepository files (globallists is entered in the **First Filename** field and listrepository is entered in the **Second Filename field**).

The two files share a one-to-one relationship (Select One-to-One from the drop-down list in the **Relationship Type** field.), which means that one record in the globallists can be uniquely associated with one record in listrepository.

The two files are joined based on the fact that the **name** field in the **First Filename field** (globallists) matches the **handle** field in the **Second Filename field** (listrepository). The remaining fields on the form are used for distributed ticketing and are not used in joined queries.

Joined queries can included several files. If you make a joined query that includes N, files, you will need to make N-1 erddef files, with one file as the central file defining the relationships to all others. For example, to make a joined query with 3 files, A, B, and C, 2 erddef files must be created, defining the relationship between A->B and A->C.

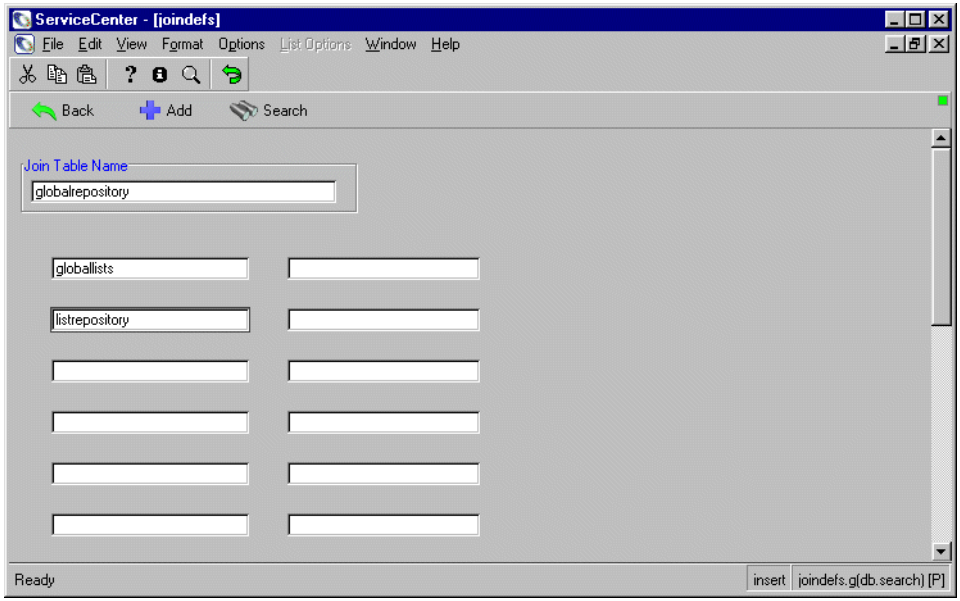
Defining a Joinfile

The second step in defining a joined query is to create a joinfile. A joinfile is a programmatic way to refer to your joined result set.You define a joinfile by adding a record to the joindefs file. The first line of the form stores a unique name for this joinfile.

All joinfiles must have unique names, and no joinfile can have the same name as a dbdict. The left-hand array is used to stores the filenames that make up the join. In this case, a join is established between globallists and listrepository.

The join file will be dynamically created internally; therefore it does not need a dbdict entry.

The examplebelow shows a joinfile named globalrepository.



Note: The files listed here must have a relationship defined in the erddef file.

Querying Out of a Joinfile

To access the joinfile, create a format with input fields from both files.

Input property on the format has the syntax:

file.<filename>, <fieldname>
— Or —
file.<second filename>, <fieldname>.

You can then execute a query normally. The result set that returns will contain matching criteria from both files. The structure of this file will be artificially constructed and consist of one structure containing two sub-structures. Each sub-structure will contain fields from one of the two files.

For example:

File A:
{["Brown",12,'12/12/95']}
File B:
‘{["Brown","Can’t print to network printer","but I can print to my local printer",true]}

A joinquery against the two would return:

{[{["Brown",12,'12/12/95']}, {["Brown","Can’t print to network printer”,"but I can print to my local printer",true]}]}

Referencing Fields in a Joined Result Set

You can reference fields out of a joined result set using this syntax:

file.<filename>, <fieldname> in $file

For example:

file.globallists,list.variable in $L.file

where $file stand for the file variable used.

Some examples of file variables are:

$file: Format Control
$L.filed: display application

P4 Glossary

This table contains terms used in this guide.

Term	Definition
Data Logical File	The Data Logical File consists of all data records in a P4 file and is defined by the data logical file descriptor. The Data Logical File must be assigned to one or more pools (pool 3 or higher).
dbdict	The **dbdict**, or better the dbdict record, describes the file. It contains field definitions like the data types and key definitions. Additionally the numbers of the data and index logical files are stored in the **dbdict**. These numbers refer to records in the **scdb.lfd** (logical file descriptor) file. Another field in the **dbdict** holds the root record number. This root record number points to a record in the physical file where the root index record is stored. Every database, even an empty one, contains at least one index record, this root index record.
Emergency Free Space Chain	There is one special free space chain in each pool, the emergency free space chain. This chain is used when every other way of allocating space has failed, meaning there is no other free space available and the attempt to extend any of the associated physical files to this pool has failed. In such a case the allocation is done using this emergency free space. After this allocation ServiceCenter flags this pool as full internally and denies every further allocation, until either a reset is done on a database stored in the same pool or the system is shut down and a LFMAP is run.

Term	Definition
Free Space Chain	ServiceCenter reuses the space in the pools by organizing free space in chains. That means whenever a record somewhere in a pool is deleted, it is put into a chain of free space. The next time a record of the same size needs to be allocated, the space of the previously deleted record is taken out of the free space chain and is reused. This also means that the size of a physical file does not automatically decrease by deleting records. To actually shrink the files you need to run the LFMAP utility. This utility reorganizes the whole physical file, moving all records to the front and truncating the file after the last record. This prevents the P4 pools (and corresponding physical files) from growing as rapidly if the ServiceCenter administrator purges logical files like **syslog**, **msglog**, **mail**, **eventin**, or **eventout** records.
Index Logical File	The Index Logical File consists of all index records in a P4 file. One index record can consist of one or more index entries. These index entries may later be referred to as indexes. The index logical file is defined by the index logical file descriptor and is assigned to one or more pools (pool 3 or higher). These pools can be different from those of the Data Logical File.
Physical File	The physical file is what you can actually see from the operating system's point of view using commands/utilities like NT's explorer, Unix's **ls** command or MVS's (now OS/390) 3.4 option. In the ServiceCenter canned system there are 10 different physical files.
Pool	A pool is a layer between logical files and physical files. Several (up to 21) physical files can be grouped together to a pool. A logical file can be assigned to up to 10 different pools. In the ServiceCenter canned system, Peregrine Systems ships P4 with 10 pools, each of which consists of 1 physical file. Some information in the dbdict record is also stored in the corresponding LFD record (e.g. pools to be used for logical file, record length) and this information should always match. However, the system actively uses the LFD values and the dbdict values are for reference only. There currently is a limitation that P4 can handle at most 38 physical files. This limitation is due to the coding and not to any data structures and therefore might be increased in some future release.

Index

G

H

I

P

Q

R

S

T

U


Peregrine
SYSTEMS