Peregrine

ServiceCenter

Database Management and Administration, Volume 1

Release 5.1

PART NO: SCT-5.1-ENG-01012-00233

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 1

Database Management and Administration, Volume 1 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

RDBMS Support Overview

This section 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.

Chapters in this volume include:

- *Introduction to RDBMS Conversion* on page 17 — this chapter introduces the conversion process, discusses its pros and cons, and answers some frequently asked questions.
- *Conversion Strategy* on page 25 — this chapter details the strategy used by ServiceCenter when it maps tables to an RDBMS. It also discusses the implications of the various mapping options available, and the out-of-box mapping, which can be used to help get your data mapped more quickly and efficiently.
- *Data Mapping* on page 71 — this chapter discusses analyzing ServiceCenter files. It also outlines the necessary procedures for configuration of the RDBMS before the conversion of ServiceCenter Data from the native P4 system to an RDBMS can take place.
- *Preconversion Configuration* on page 97 — this chapter explains the configuration necessary before converting ServiceCenter to a Relational Database Management System (RDBMS).
- *Conversion to an RDBMS* on page 161 — this chapter explains the process of converting the P4 file system to an RDBMS.

- *Post Conversion* on page 193 — this chapter discusses the procedure for editing a database after conversion to an RDBMS format.
- *Tuning for Performance* on page 213 — this chapter discusses procedures for enhancing performance for SQL, Oracle, and Sybase RDBMS formats.
- *Troubleshooting* on page 225 — this chapter discusses common RDBMS errors for each database format. It also contains information that can help improve CPU performance, and network performance.
- *Shadowing P4* on page 235 — this chapter discusses the concept and strategy of shadowing. It also includes procedures for shadowing individual files and for stopping shadowing.
- *Converting RDBMS Files Back to the P4 Format* on page 243 — this chapter discusses the procedure for converting data in an RDBMS database back into the ServiceCenter P4 format.

Appendices for this section include:

- *Characteristics of ServiceCenter Files* on page 245 — this appendix contains information necessary to configure ServiceCenter files for optimal performance.
- *Initialization Parameters for RDBMS* on page 297 — this appendix contains the current list of SQL initialization parameters used to configure the ServiceCenter sc.ini file.
- *Data Definitions* on page 315 — this appendix contains list of data definitions for ServiceCenter Files.

CHAPTER 1

Introduction to RDBMS Conversion

This chapter was designed to aid managers and administrators in making the decision on whether or not to convert ServiceCenter Data from the native P4 system to an Relational Database Management System (RDBMS). It provides all audiences with an overview of ServiceCenter's Architecture, and a discussion of the conversion process. It also answers some frequently asked questions.

Topics in this chapter include:

ServiceCenter Architecture

ServiceCenter uses the native client libraries of the RDBMS to manage database I/O, rather than another middle-ware product such as ODBC. ServiceCenter is a client/server product with a three–tiered structure:

- client
- application server
- database server

It is not necessary for the application server and the database server to be running on the same computer. For example, a ServiceCenter application server running on Windows could access ServiceCenter data being stored on an HP-UX Oracle server.

ServiceCenter supports RDBMS servers for the following platforms:

RDBMS Server	Platforms Supported		
DB2Universal	Windows	Unix	OS/390
Informix	Windows	Unix	
Oracle	Windows	Unix	
Sybase	Windows	Unix	
Microsoft SQL	Windows		

Deciding Whether to Use P4 or Convert to an RDBMS

By default, ServiceCenter uses its internal file system (P4) to store data, such as Incident tickets, Changes, and Service Requests. However, the data storage in ServiceCenter can be converted from its internal P4 format to a storage location on a commercial RDBMS. Making a decision to use a relational database system, or the P4 system has both technical and cultural components.

From a technical point of view, the P4 file system will take less space and will perform better out-of-box than a relational system. The records used by ServiceCenter applications are stored as a single record within the P4 file system and therefore can be retrieved and presented to the application very quickly. The records used by these applications are pretty complex and

involve arrays of data, structures of data, and arrays of structures. Relational database systems are not designed to handle this type of data as a single table but rather require that the data be normalized. Therefore what can be stored as a single record within the P4 file system, may require multiple tables within a relational database. These tables then have to be joined together to present the logical record back to the application.

The P4 file system does not have any forward recovery or dynamic back out capability. The P4 file system was written so that the possibility of corruption to the file system due to a system failure is minimized. However, such things as power outages at the wrong time can corrupt a P4 file system. Because there is no back out capability, the corruption either has to be patched using P4 file system maintenance utilities, or a backup of the file system has to be restored and lost data reentered. A relational system has a dynamic back out utility to remove in-flight transactions in such situations and also has a forward recovery capability in those cases where a backup must be restored.

From a cultural point of view, there are organizations that have standards in place requiring that a particular relational database system be used. In that case, a fully trained and experienced Database Administration staff is a valuable resource.

To obtain the best performance and to take full advantage of an RDBMS, a fully trained and experienced database administrator (DBA) is required. The P4 file system is easier to handle and will perform better out-of-box, but still requires a fully trained and experienced person. ServiceCenter will run on the popular relational database systems and it will run without a separate RDBMS (i.e. P4 only). The experienced DBA staff currently in place can be utilized, without requiring additional P4 specialists, or P4 specialists can be used without requiring additional DBA staff. With ServiceCenter there is no need to have both.

When making a decision of whether or not to convert to an RDBMS, the possibility of shadowing should not be overlooked. Shadowing allows you to copy your production system to an RDBMS for creating reports while retaining your working file system in P4. When shadowing is in effect, your file system is updated in both places. Shadowing your entire P4 file system is not necessary. The best strategy may be to shadow individual files against which you intend to run reports and leave your application files in the P4 format.

Conversion Process Flow

RDBMS conversion can be accomplished when you install ServiceCenter or at a later date.

Important: While this guide will assist in converting to a Relational Database, to obtain the best performance and to take full advantage of the RDBMS, a fully trained and experienced DBA is required.

Conversion requires the following major steps:

1 Installation and configuration of ServiceCenter.

If ServiceCenter has not yet been installed, install it, referring to the correct guides. During the installation, set the ServiceCenter case sensitivity for sort order so that it does not conflict with that of the RDBMS.

See Database Management and Administration, Volume 2, for information on how to change the case sensitivity for ServiceCenter.

- For installing ServiceCenter on OS/390, see the *Client/Server Installation Guide for OS/390*.
- For installing ServiceCenter on Unix platforms, see the *Client/Server Installation Guide for Unix*.
- For installing ServiceCenter on Windows platforms, see the *Client/Server Installation Guide for Windows*.

2 Installation and configuration of the RDBMS.

If the RDBMS has not yet been installed, install it. During the installation, set the RDBMS case sensitivity for sort order so that it does not conflict with that of ServiceCenter. See the installation and configuration documentation for the RDBMS.

Important: Setting the case sensitivity is particularly important for Microsoft SQL server, because it is insensitive by default, and ServiceCenter is Case Sensitive by default.

3 Mapping RDBMS Data to ServiceCenter. See *Conversion Strategy* on page 25.

4 Setting up the ServiceCenter and the RDBMS for the conversion process. See *Preconversion Configuration* on page 97.

5 Converting the data. See *Conversion to an RDBMS* on page 161.

6 Editing the database after conversion. See *Post Conversion* on page 193.

Frequently Asked Questions

Why does ServiceCenter have a Database Dictionary

We use a data base dictionary (dbdict) layer between the application logic and the database itself to preserve independence to all standard relational database management systems (RDBMS). This is the way we support Oracle, Informix, Sybase, Microsoft SQL server, DB2MVS, and DB2Universal from ServiceCenter. To us, this gives all of our clients maximum independence from any proprietary database technology since it permits them to use the RDBMS of their choice. The mapping that must be done for Oracle is much different from the mapping that is done for SQL server but the dbdict allows these types of differences to be shielded from the applications.

Why does Peregrine recommend initial development in P4 and then conversion to an RDBMS?

The implementation of ServiceCenter often requires adding additional data fields to the workflow objects in our system, such as incident tickets or change requests. It is fast and convenient for us to develop the initial dbdict on top of P4 and then convert the system to an RDBMS once the schema has reached a relatively stable state. This is our standard methodology in creating customer modifications through our professional services group. Once the adapted system is accepted as being what the customer wants, we then convert the changes into the chosen RDBMS. It is not necessary to do the development in this way. The system will automatically generate the DDL for converted files if new fields are added. The system will also automatically detect fields added to tables in the RDBMS and add those fields to the dbdict. However, we believe it is more effective to develop in P4 and then convert. One of the considerations is that there are no maximum fields lengths for character data in P4. During a conversion the system will scan every record in every file and determine the maximum size of each field. The more data that exists in the file, the more accurate these numbers will be. If a file is converted without having any data, then the ServiceCenter Administrator and the DBA must look at each field and determine the maximum size necessary for the field.

Are the IR Expert indexes stored within the P4 file system?

No they are not. The IR Expert indexes are stored in flat files in a proprietary format. They are not P4 files nor are they RDBMS tables. Backup procedures should take into consideration the IR Expert files. With Asynchronous IR it is possible to stop updates to the IR Files during the database backup process and then restart the updates once the backup is complete.

Is ServiceCenter 24X7 relevant in a fully converted system?

No it is not. ServiceCenter's 24X7 (scenter -startlogging and scenter -stoplogging) facility halts I/O at the physical P4 file level. Since there is no P4 I/O on a fully converted system, the 24X7 support is not relevant.

After a complete conversion of all ServiceCenter data to a relational database management system (RDBMS), the P4 file system still exists. What is its purpose, and does it need regular backups after the conversion?

After a complete RDBMS conversion, no production data exists in the P4 file system, so P4 files (scdb*) do not need regularly scheduled backups. Regular backup procedures for the target RDBMS are sufficient.

However, although the P4 file system is not involved in handling production data, it does perform several important functions, discussed below, and cannot be deleted. Immediately after a complete conversion to an RDBMS, you should backup the nearly empty P4 file system and store it safely. If any disaster strikes the P4 file system, it can be restored from this single backup. No other backups are needed unless anything discussed below changes.

The P4 file system serves the following functions even after a complete conversion:

- The P4 file system holds the company-wide time zone record: ServiceCenter supports a company-wide time zone record that defines the initial time zone used by all processes. This record is written into the P4 file system but does not belong to any file and is not converted to the RDBMS. This timezone record is usually established during initial implementation and stays without change.

- The database dictionary description within the P4 file system is used to bring up ServiceCenter: The database dictionary (dbdict) is a special file within ServiceCenter that describes all other files. For ServiceCenter to start, it must read the dbdict file to get file descriptions and the code file to load the applications. Since the mapping of the dbdict file is contained within itself and since the dbdict file may itself be mapped, there is a static, bootstrap dbdict description within the P4 file system that contains the dbdict mapping.
- New files are created initially as P4 files: When you use the Database Dictionary utility to create a new file, that file exists initially as a P4 file. Once a new file has been defined, you use ServiceCenter's SQL Utilities to map the file to the target RDBMS. This new file only exists in P4 until the conversion is done, it typically contains no data prior to conversion, and its definition is a record in the dbdict file which already resides in your RDBMS. Therefore, it requires no P4 file system backup.
- The P4 file system holds temporary files needed when creating certain reports: Reports created by ServiceCenter sometimes need temporary files to manage and sort required report data. These temporary files are created in the P4 file system and are destroyed during the reporting process.

CHAPTER 2 Conversion Strategy

This chapter was designed to explain the strategy used by ServiceCenter when it maps tables to a Relational Database Management System (RDBMS) and the implications of the various mapping options available. Its intended reader is a ServiceCenter or RDBMS database administrator planning to convert a ServiceCenter system to an RDBMS.

Topics in this chapter include:

Overview

ServiceCenter is capable of storing data either in its own internal database format, or in a variety of commercial database servers. Internally, ServiceCenter manages data in complex record formats. These formats are not always structured to permit one–to–one mapping between a ServiceCenter record and a row in an RDBMS table. In many cases, multiple rows in multiple tables are used to store a single ServiceCenter record on a commercial RDBMS.

Important: A well tuned database converts most easily and quickly.

Analyzing ServiceCenter Tables for Conversion

In ServiceCenter, the definitions of tables are stored in files called database dictionaries or **dbdicts**. Each field in a **dbdict** needs to be evaluated to determine if this data is needed in the RDBMS.

All tables can be converted to the RDBMS, but they do not all have to be converted. The determination of which tables should be converted differs from company to company. An example of one way to determine which tables should be converted, is to decide to convert all tables where reporting is done.

After determining which tables will be converted to the RDBMS, analyze which tables will function well together. Tables that function well together should reside within the same the RDBMS tablespace.

An example would be to include the following tables in one tablespace:

- Contacts
- Location
- Category
- Assignment
- Clocks

It is recommended to put the number table in its own tablespace since this table is accessed for each new record in call, problem, request and change. Also, tables that are updated simultaneously in the same module in ServiceCenter, such as problem and probsummary should be put into different tablespaces to improve performance.

Before converting a table to an RDBMS, the following information is necessary for the proper space to be calculated by the DBA for each table as well as the entire tablespace:

- Average Row Length
- Number of Records on Table Initially
- Approximate Growth Rate per month
- Number of Array Tables
- Number of Indexes
- Length of Fields in Index

When reviewing each table's SQL mapping in the dbdict, all the information on the right side corresponds to the attributes that will be created for that table when it is converted to the RDBMS.

Considerations:

- Each index in the RDBMS should be in its own tablespace.
- Make note of the size of each key in each index, by index. This will be used to calculate index space information.

Note: When using ServiceCenter to review this information, use the ServiceCenter EXPRESS client.

What is a ServiceCenter Record?

A ServiceCenter record is nothing more than a discreet unit of related data that can be loaded, manipulated, and saved as a block. ServiceCenter records have defined structures, such as that required for all records containing Incident tickets. The collection of all records with a given structure inside of ServiceCenter is referred to as a *file*. A ServiceCenter file is roughly equivalent to an RDBMS table and contains a large number of similarly structured elements that can be retrieved for manipulation using indexed search methods.

Despite the similarities, ServiceCenter records do differ from rows in an RDBMS table. Generally speaking, an RDBMS row is made up of various columns containing a single element of data, such as a **varchar**(**200**) or a **float**. ServiceCenter records may contain single data elements, just like rows in an RDBMS, but they may *also* include complex sub-structures, or arrays of data elements. Consequently, a ServiceCenter record may contain a field of character data and another field containing an array of thirty-seven numbers. These sub-structures do not have analogous data types on most RDBMS.

Simple Data Record

ServiceCenter stores its record definitions as separate records. The collection of all such record *descriptors* is stored in the dbdict, or Database Dictionary file.

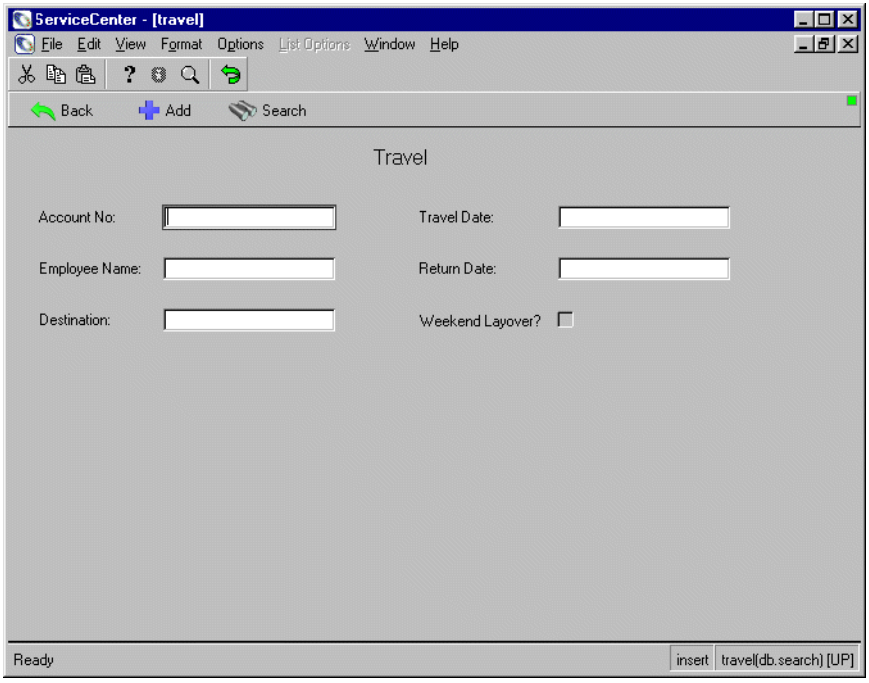


Figure 2-1: Simple ServiceCenter Data Record

The fields consist of the following data types:

Field Name	Input Value	Data Type
Employee Name	name	character
Destination	destination	character
Travel Date	travel.date	date/time
Return Date	return.date	date/time
Account Number.	account.number	number
Weekend Layover?	weekend.layover	logical

The following is the Database Dictionary record for the *travel* file:

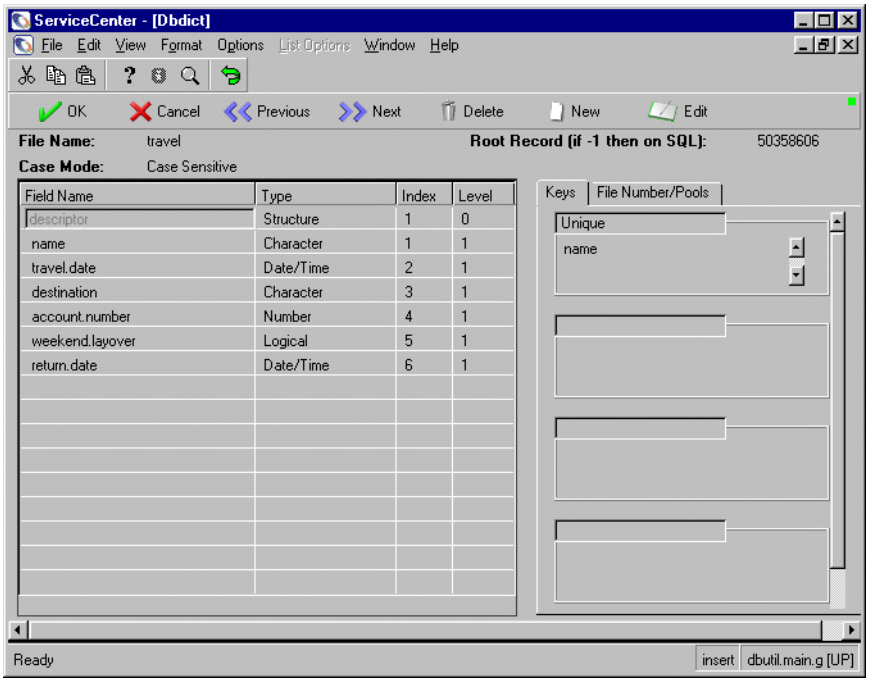


Figure 2-2: Database Dictionary File

Note: P4 actually does not need any unique fields/keys in a **dbdict**, although it is strongly recommended to have one. When converting a P4 file to SQL, a primary key is necessary because SQL requires it. This primary key can be any data type and could be a concatenated key.

Data Record with an Array

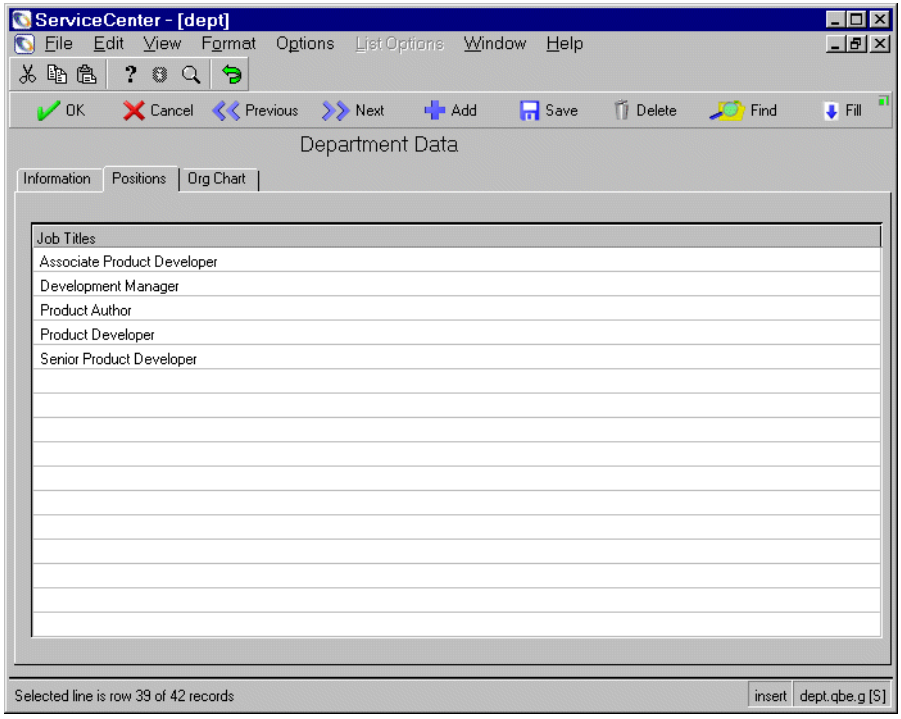


Figure 2-3: Data Record with a Simple Array

The fields consist of the following data types:

Field Name	Input Value	Data Type
Job Titles	title	array

The Database Dictionary for this record looks like this:

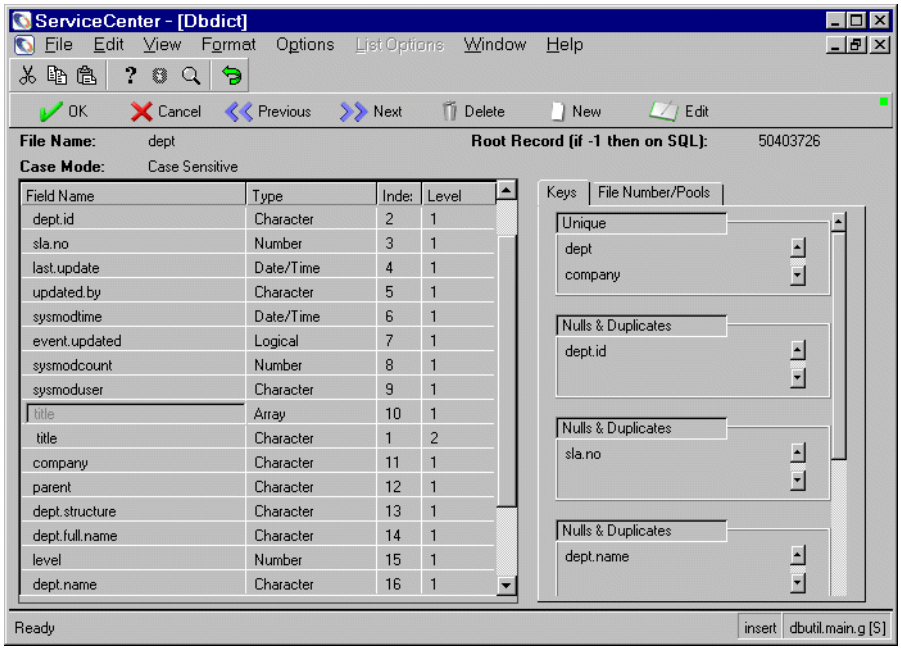


Figure 2-4: Database Dictionary File for a Simple Array

Elemental Data Types in ServiceCenter

All ServiceCenter records are made up of smaller, elemental data types. ServiceCenter supports 11 different types of data that can be stored in a record.

ServiceCenter data types are divided into three categories:

Simple Data Types

The vast majority of operational data in a ServiceCenter system is stored internally in one of the four, *simple* data type. These types should be roughly familiar to anyone who has worked with a commercial RDBMS.

Data Type	Description
Character	Used to store text strings. A field of type *character* may contain up to 32767 characters if running on a single byte system, or 16383 on a double byte version of ServiceCenter. As a rule, most character fields in ServiceCenter are less than 100 characters in length.
Number	Used to store numeric values. All data of type *number* are treated internally as 8 byte floats.
Date/Time	Used to store date/time values. Internally, ServiceCenter tracks dates as the number of seconds that have passed since the year 0.
Logical	Used to store Boolean values. Unlike most RDBMS, ServiceCenter has a four character Boolean alphabet. A *logical* field may be True, False, Unknown, or NULL.

Internal Data Types

ServiceCenter supports a variety of *internal* data types that do not have simple analogs in the SQL world. For the most part, ServiceCenter uses these data types to store internal control information. Little or no data that a ServiceCenter user sees on the screen is ever stored in these formats.

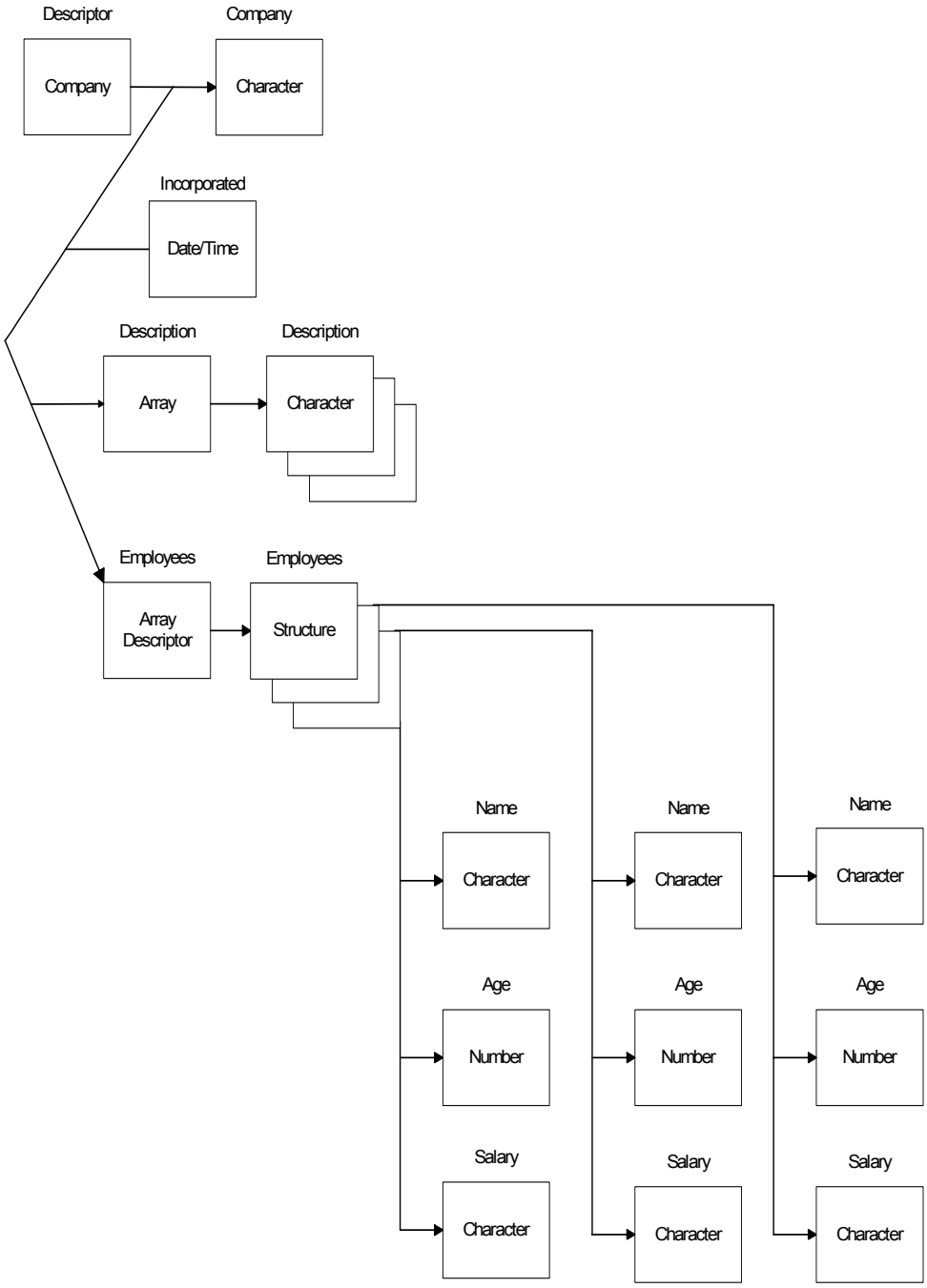
Data Type	Description
Label	Used to store offset information when RAD language programs are being compiled. Labels dictate the flow of RAD programs.
File	Used to pass records between various RAD applications.
Operator	The parsed version of a RAD language expression such as "$name=operator()"
Expression	Used to store operators on file.

Complex Data Types

ServiceCenter also supports two *complex* data types. These data types allow a ServiceCenter record to contain multiple similarly formatted *sub-rows* and/or discreet sub-structures within the main structure.

Data Type	Description
Array	A data type that contains an array of identical data types (e.g., an array of character data types or an array of date/time data types.)
Structure	A data type that contains a collection of other data types (e.g., record within a record.)

Diagram of Complex Data Types in ServiceCenter



RDBMS Mapping Strategy

ServiceCenter presents an interesting challenge when mapping its data to an RDBMS. The ServiceCenter simple data types can be mapped to analogous RDBMS data types, and the ServiceCenter internal data types can be mapped as binary *blob* (binary large object) data; however, there is no easy way to store a ServiceCenter complex data type (array or array of structures) in an RDBMS.

Fortunately, there are a variety of techniques that ServiceCenter can use to map a complete record to an RDBMS table. This section describes the general ServiceCenter RDBMS mapping strategy and then discusses various ways that ServiceCenter can store its complex data types in RDBMS tables.

For information on how to map the data, see*Data Mapping* on page 71.

Simple Mapping

The simplest case of RDBMS mapping involves a ServiceCenter file that contains only the four elemental data types: character, number, date/time, and logical.

Field Name	Input Value	Data Type
Employee Name	name	character
Destination	destination	character
Date Traveled	date.traveled	date/time
Account No.	account.number	number
Weekend Layover?	weekend.layover	logical

ServiceCenter's RDBMS mapping approach is to create a main table on the RDBMS server named <*filename*>m1, where <*filename*> is the name of the file descriptor that ServiceCenter uses internally. All simple data elements from the record are stored in the RDBMS server and translated to their analogous RDBMS data types. For example, ServiceCenter numbers are mapped to a *float* field.

ServiceCenter performs two other operations before it creates the RDBMS table:

- If the field name inside a ServiceCenter record is a reserved word on the RDBMS server in question, ServiceCenter will append a suffix to the name. For example, **select** becomes **select _col** or **select _prgn**.
- ServiceCenter has a different set of legal characters to use in a field name than do most RDBMS Servers. To ensure that the fields it is creating are legal on the RDBMS server, ServiceCenter will replace all period (.), back slash (\), and apostrophe (') characters with an underscore (_). For example, **serial.number** becomes **serial_number**.

When mapping character fields, ServiceCenter has two priorities: allocate a sufficiently long field on the RDBMS server to store all the data in the analogous ServiceCenter records; and to not waste any space on the RDBMS server. To accomplish this, ServiceCenter uses the following formula to determine how long to make a *char* or *varchar* field on the RDBMS server:

Length of RDBMS Field = round (greater of(x,y) + pad length +5)

- Let x = Length of the longest character sting stored inside any field of any record in the ServiceCenter file.
- Let y = Length of longest display format used to display the field on any ServiceCenter format

ServiceCenter first looks at all the records in the file in question and identifies the longest string of characters for a given field. It then looks at all the forms which display that field and identifies the longest display element for that field. It then assumes that the larger of these two numbers represents the true maximum length of the field. As a safety measure, ServiceCenter adds a *pad* to this length and rounds up to the nearest factor of ten.

Field Types

ServiceCenter uses the following field types for mapping simple data types:

ServiceCenter Data Type	Sybase and Microsoft SQL Server	Oracle	DB2	Informix
character	char*	char*	char*	char*
number	float	float	float	float
logical	char(1)†	char(1)†	char(1)†	char(1)†
date/time	datetime	date	timestamp	datetime

* The length of character fields will be set based upon the formula above.
† Char (1) fields will contain "t" for true, "f" for false, "u" for unknown and NULL for null.

From the sample record in *Simple Mapping* on page 36, the *travel* file would be mapped to SQL as follows:

ServiceCenter Field	ServiceCenter Data Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
name	character	travelm1	name	char(60)
destination	character	travelm1	destination	char(40)
account.number	number	travelm1	account_number	float(8)
date.traveled	date/time	travelm1	date_traveled	datetime*
weekend.layover	logical	travelm1	weekend_layover	char(1)

* Assumes Informix, Sybase, or Microsoft SQL server.

Expression

ServiceCenter allows records to store expressions in logical fields. These expressions are evaluated at run time to determine the value of the field. For example, a logical expression containing **gui**() would be true on GUI clients but false on text clients. If ServiceCenter determines that the contents of a specific logical field in a record contain an expression, it will map the logical field as binary data (*Mapping Internal Data Types* on page 39), rather than as a **char**(1).

Mapping Internal Data Types

ServiceCenter's internal data types do not have simple analogs in most RDBMS. In order to store these data types in an RDBMS, ServiceCenter must translate its internal binary format into a platform–neutral binary datum and store it on the RDBMS server as binary data.

ServiceCenter uses the following field types for mapping internal data types:

ServiceCenter Field Type	Sybase and Microsoft SQL Server	Oracle	DB2	Informix
Label	char*	char*	char*	char*
File	image	long raw	long varchar	text
Operator	binary*	raw*	unknown	unknown
Expression	binary*	raw*	unknown	unknown

* Length of the char and binary fields will be determined by the formula used for character data.

Mapping Complex Data Types — Arrays of Characters

ServiceCenter stores long text edit fields, such as the details of an Incident ticket, as arrays of distinct character fields.

When a user types some text into a multi-line edit field, the text is chopped into 60 character pieces. Because the client uses a variable sized font, more characters may fit into the first line of the GUI edit box than would into the equivalent text mode array. The number of characters that appear to be in a line may varies with the font and font size chosen.

For example, if the font size was such that 80 characters appeared on a line, the first 60 characters would be stored in the first array element followed by a TAB character signalling that there is more to come in the next array element. The second element would be generated out of the remaining 20 characters of the first plus 40 out of the second line followed by another TAB, and so on. If the user enters a RETURN somewhere in the text, the array element is not filled up with whatever is in the next line. When such data is read again the client automatically concatenates all the pieces back together and removes the TAB characters it inserted.

ServiceCenter has several different strategies available to it when it maps arrays of character type data to an RDBMS database:

- *Field in the Main Table* on page 41
- *Field in the Alias Table* on page 43
- *BLOB in the Main Table* on page 44
- *BLOB in the Alias Table* on page 45
- *Multi-Row Array Tables* on page 46

See also:

- *LOB Support* on page 50

The example table constructions (except for multi-row array tables) are based upon the following ServiceCenter record:

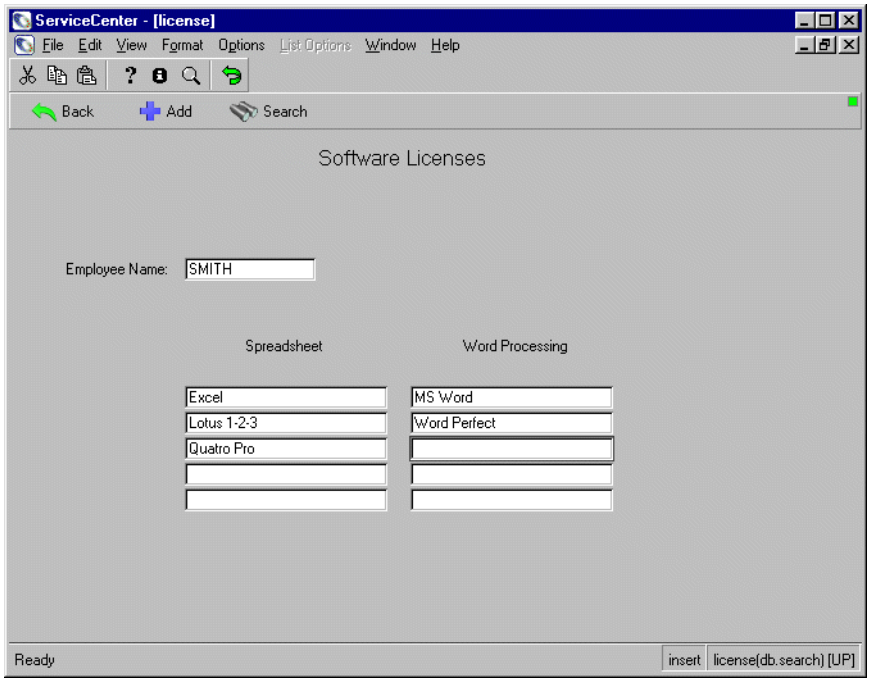


Figure 2-5: Arrays of Characters Example

The fields consist of the following data types:

Field Name	Input Value	Data Type
Employee Name	last.name	character
Spreadsheet	spreadsheet	array of characters
Word Processing	word.processing	array of characters

The Database Dictionary file for this form looks like this:

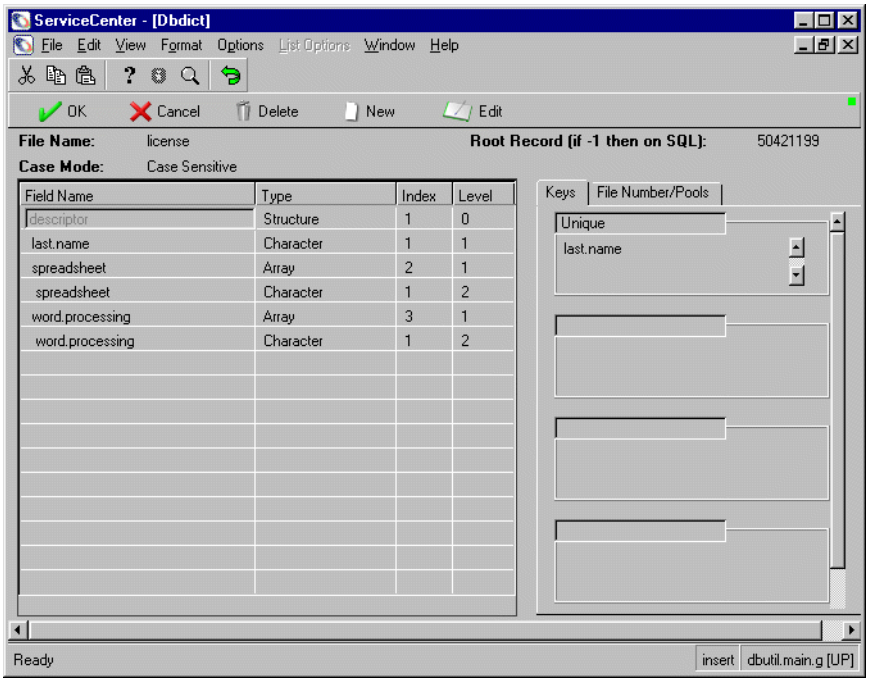


Figure 2-6: Database Dictionary File for Arrays of Characters

Field in the Main Table

This is the simplest option on many RDBMS. This option takes the ServiceCenter array and translates it into a single long text string, separating each line with the new line character. For example, an array of {“a”, “b”, “c”} become a single string reading “a\nb\nc”.

This long text field is then mapped to the appropriate RDBMS data type for long text and stored in the ServiceCenter record's main RDBMS table, as though it were a simple data type.

ServiceCenter uses the following field types for long text fields:

ServiceCenter Field Type	Sybase and Microsoft SQL Server	Oracle	DB2	Informix
long text	text	long	long varchar	text
for bit data	image	long raw	long varchar for bit	byte

Using this strategy, the sample table for the **license** file (Figure 2-6 on page 41) will be mapped as follows:

ServiceCenter Field	ServiceCenter Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
last.name	character	licensesm1	last_name	char(60)
spreadsheet	array of characters	licensesm1	spreadsheet	text*
word.processing	array of characters	licensesm1	word_processing	text*

*Assumes Sybase or Microsoft SQL server. Would be long if Oracle.

Important: Oracle tables only support one field of type *long* per table; therefore, only one array of characters can be mapped to the main table. If a table is to be converted to Oracle, and contains more than one array of characters, only the *first* such array is displayed in the main table. Additional arrays of characters are mapped as fields in an alias table (*Field in the Alias Table* on page 43). In our example, the *spreadsheet* array would be placed in an Oracle main table, while the *word.processing* array would be placed in the alias table.

Field in the Alias Table

This approach is very similar to mapping to the main table, in that it also translates arrays of characters into single strings of long text. However, the mapping strategy stores each string of long text in its own alias table on the server. If this approach is implemented, a single ServiceCenter record will be split into **1+N** tables where **N** equals the number of arrays of character in the ServiceCenter record.

Using this strategy, the sample table for the **license** file (see Figure 2-5 on page 40 (file) and Figure 2-6 on page 41 (**dbdict**)) will be mapped as follows:

ServiceCenter Field	ServiceCenter Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
last.name	character	licensem1	last_name	char(60)
spreadsheet	array of characters	licensea1	spreadsheet	text*
word.processing	array of characters	licensea2	word_processing	text*

*Assumes Sybase or Microsoft SQL server. Would be long if Oracle.

Important: ServiceCenter needs a way to associate the row in the main table (m1) with the matching row in the alias table (a1, a2, etc.). It does this by constructing an alias table containing the primary key of the main table and a row counter in addition to the long text field.

With this scheme, our example produces the following RDBMS tables:

```
CREATE TABLE licensem1 (
last_name char (60) NULL
)|
CREATE TABLE licensea1 (
last_name char(60) NULL,
record_number int NULL,
spreadsheet text NULL
)
CREATE TABLE licensea2 (
last_name char(60) NULL,
record_number int NULL,
word_processing text NULL
```

BLOB in the Main Table

This option translates the ServiceCenter array of characters into a single stream of binary data in an internal ServiceCenter format. This *blob* (binary large object) will then be stored in the ServiceCenter record's main RDBMS table as if it were a simple data type. ServiceCenter uses the following data types for long binary data:

ServiceCenter Field Type	Sybase and Microsoft SQL Server	Oracle	DB2	Informix
long binary	image	long raw	long varchar for bit	byte

Using this strategy, the sample table for the **license** file (see Figure 2-5 on page 40 and Figure 2-6 on page 41) will be mapped as follows:

ServiceCenter Field	ServiceCenter Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
last.name	Character	licensem1	last_name	char(60)
spreadsheet	Array of characters	licensem1	spreadsheet	image*
word processing	Array of characters	licensem1	word_processing	image*

*Assumes Sybase or Microsoft SQL server. Would be long raw if Oracle.

Important: Since Oracle tables only support one field of type *long raw* per table, only one array of characters can be mapped to the main table. If a table is to be converted to Oracle, and contains more than one array of characters, only the *first* such array is displayed in the main table. Additional array 1/s of characters are mapped as fields in an alias table (*BLOB in the Alias Table* on page 45). In our example, the **word.processing** array would be placed in an Oracle alias table.

BLOB in the Alias Table

This method also translates arrays of characters into a single stream of long binary data, but stores each binary stream in its own alias table on the server. If this approach is implemented, a single ServiceCenter record is split into 1+N tables where N equals the number of arrays of character in the ServiceCenter record.

Using this strategy, the sample table for the **license** file (see Figure 2-5 on page 40 and Figure 2-6 on page 41) will be mapped as follows:

ServiceCenter Field	ServiceCenter Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
last.name	character	licensem1	last_name	char(60)
spreadsheet	array of characters	licensea1	spreadsheet	image*
word.processing	array of characters	licensea2	word_processing	image*

*Assumes Sybase or Microsoft SQL server. Would be long raw if Oracle.

Important: ServiceCenter needs a way to associate the row in the main table (m1) with the matching row in the alias table (a1, a2, etc.). It does this by constructing an alias table containing the primary key of the main table and a row counter in addition to the long text field.

With this scheme, our example produces the following RDBMS tables:

```
CREATE TABLE licensem1 (
last_name char (60) NULL
)
CREATE TABLE licensea1 (
last_name char(60) NULL,
record_number int NULL,
spreadsheet text NULL
)
CREATE TABLE licensea2 (
last_name char(60) NULL,
record_number int NULL,
word_processing text NULL
```

Multi-Row Array Tables

If you have a spread sheet field, ARRAY, which contains a record over 32K, using Multi-Row Array Table can avoid data truncation. *LOB Support* on page 50 is a possible alternative for Multi-Row Array Tables.

Under this strategy, a separate alias table is created for each array in the ServiceCenter record in which an element of the array is given its own row. Therefore, a given ServiceCenter record spans **1+N** tables where **N** is the number of arrays in the record. Each array in a ServiceCenter record spans **M** rows in its own alias table where **M** is the number of elements in the array.

Note: The RDBMS structure cannot be altered the to change the mappings.

Using this strategy, the sample table for the license file (see Figure 2-5 on page 40 and Figure 2-6 on page 41) mapped as follows:

ServiceCenter Field	ServiceCenter Type	RDBMS Table	RDBMS Field Name	RDBMS Data Type
last.name	character	licensem1	last_name	char(60)
spreadsheet	array of characters	licensea1	spreadsheet	varchar(255)*
word.processing	array of characters	licensea2	word_processing	varchar(255)*

* Each array will have multiple rows in the alias table.

The following example illustrates the mapping of a record in the *license* file.

last.name =“Smith”

Spreadsheet ={“Excel”,“Lotus 1, 2, 3”,“Quatro Pro”}

Word Processing ={“MS Word”,“Word Perfect”}

RDBMS Main Table (licensem1):

last_name char(60)
Smith

RDBMS Alias Table1 (licensea1):

last.name char(60)	record_number int	Spreadsheet varchar(255)
Smith	1	Excel
Smith	2	Lotus 1, 2, 3
Smith	3	Quatro Pro

RDBMS Alias table2 (licensea2):

last_name char(60)	record_number int	word_processing varchar(255)
Smith	1	MS Word
Smith	2	Word Perfect

A single ServiceCenter record has become a total of 8 rows on 3 RDBMS tables.

To reconstruct the original record, ServiceCenter executes the following queries:

- Select * from licensem1 where last_name="smith"
- Select spreadsheet from licensea1 where last_name="smith" order by record_number
- Select word_processor from licensea2 where last_name="smith" order by record_number

Mapping Complex Data Types — Arrays of Structures

Arrays of structures can be thought of as sub-tables with multiple rows, similarly formatted inside a specific ServiceCenter record. An example would be the following:

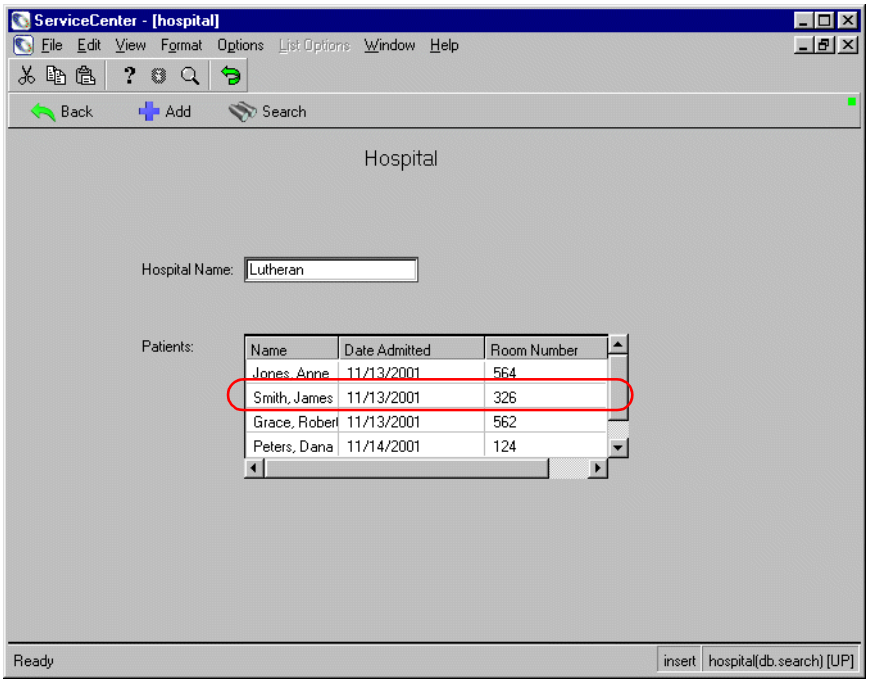


Figure 2-7: Array of Structures

The fields consist of the following data types:

Field Name	Input Value	Data Type
Hospital Name	hospital.name	character
Patients	patients	array of structures
Name	patient.name	character
Date Admitted	date.admitted	date/time
Room Number	room.number	number

The Database Dictionary file for this form is:

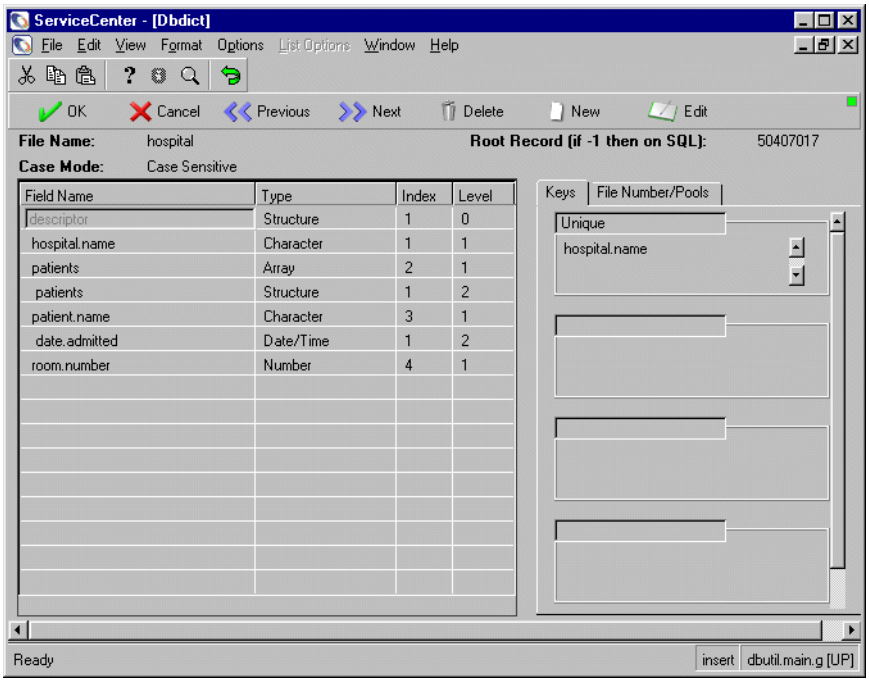


Figure 2-8: Database Dictionary Record for *Hospital* File

The hospital file is mapped to RDBMS in the following form:

```
Hospital.name="Lutheran"
Patients ={
        {["Jones, Anne",'11/13/2001 00',564]},
        {["Smith, James",'11/13/2001 00',326]},
        {["Grace, Robert",'11/13/2001 00',562]}
        {["Peters, Dana",'11/14/2001 00',124]}
        }
```

In ServiceCenter, all arrays of structures are converted to a binary format and stored as *blobs* in the record's main RDBMS table as though they were arrays of characters and mapped with the **Blob in Main Table** option.

LOB Support

This section outlines how to use LOB (Large Object) and datatypes in supported external databases. The LOB data type is supported for the external databases Oracle8 (8.0x or 8I) and DB2Universal.

Topics in this section include:

If you have a spread sheet field of ARRAY which contains a record over 32k bytes, using Multi-Row Array Table can avoid data truncation. *Multi-Row Array Tables* on page 46 are a possible alternative for LOB Support.

CLOB or BLOB Datatypes

CLOB and BLOB (Character LOB and Binary LOB) can be used to replace long and long raw in Oracle, and long varchar and long varchar for bit in DB2Universal. CLOB or BLOB can store up to 2GB data in length per row.

Some advantages of using CLOB and BLOB are:

- Instead of storing the data value in a table for CLOB or BLOB, only a LOB locator is stored. The data itself is stored somewhere else in the database. The size of a LOB locator is 20 bytes for Oracle and from 72 up to 316 bytes for DB2Universal. This feature removes the limit of allowing only one long per table in Oracle RDBMS. Since the size of a LOB locator is much smaller than the real data, a ServiceCenter file will be mapped to fewer tables. For example, the PROBLEM file will be mapped to PROBLEMM1 table with CLOB or BLOB datatype support instead of PROBLEMM1,M2,…, A1, A2,… A31.
- A CLOB or BLOB field can be processed in such a way that a data buffer is not needed to be obtained until the length of the data is known. This is contrasted with a long datatype where the maximum size for data buffer is allocated for each such field to read data.

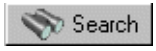
In order to use CLOB or BLOB datatype, some settings need to be made in the sqldbinfo file.

Setting Up LOB Datatypes for Oracle

To set up the sqldbinfo file for Oracle:

1 Open the sqldbinfo file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

The SQL DB Type search form is displayed.



2 Select oracle8 from the SQL DB Type and click **Search** or press Enter.

The sql.db.info form is displayed.

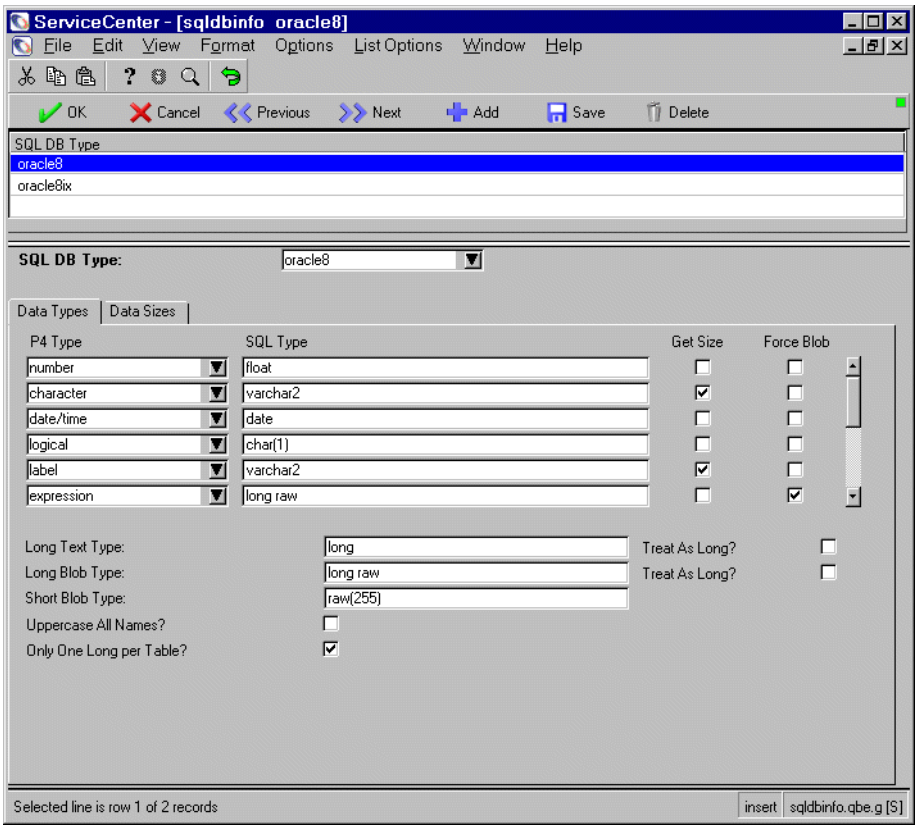


Figure 2-9: sql.db.info form - Data Types Tab

3 In the **Long text type:** text box, Replace **long** with **CLOB**. Uncheck **Treat as Long?** if it is checked as default.

4 In the **Long Blob type:** text box, Replace **long raw** with **BLOB**. Uncheck **Treat as Long?** if it is checked as default.

5 Uncheck the **Only One Long per Table?** check box if it is checked as default.

6 Select the **Data Sizes** tab.

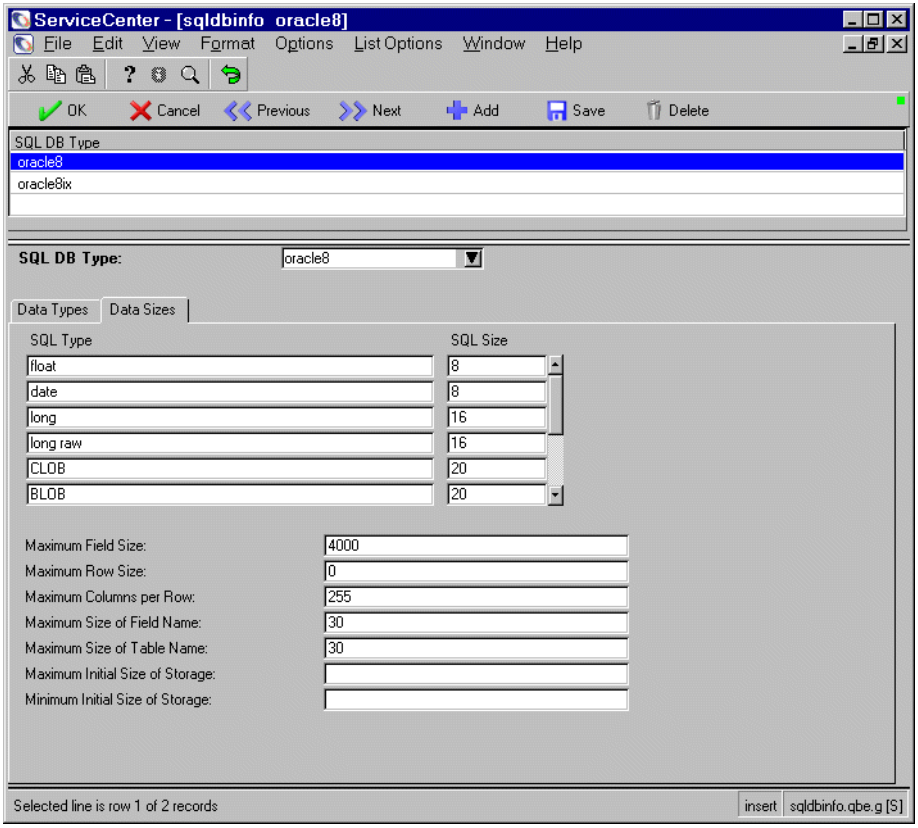


Figure 2-10: sql.db.info form - Data Sizes Tab

7 Add **CLOB** to the **SQL Type** box and **20** to the corresponding **SQL Size** box.

Note: All sizes in the **Data Sizes** tab are in bytes.

8 Add **BLOB** to the **SQL Type** box and **20** to the corresponding **SQL Size** box.

9 Save the settings and click **OK**.

Setting Up LOB Datatypes for DB2Universal

In order to use CLOB or BLOB datatype, some settings need to be made in the sqldbinfo file. When declaring a column of LOB datatype, you must declare its maximum length, which can be anywhere in the range from one byte to two gigabytes.

To do this, use a command based on this example:

BLOB/CLOB(nK/ nM/ nG)

If only BLOB/CLOB(nK/ nM/ nG) is used, then NOT COMPACT and LOGGED are used by DB2 as defaults.

Where:

K is kilobytes (1024 bytes).
M is megabytes (1,048,576 bytes).
G is gigabytes (1,073,741,824 bytes).
If K, M or G is not used, n is the real number of bytes used.

Note: By default, each ServiceCenter record has a maximum size of 64K. Therefore, using BLOB(64K), and CLOB(65K) would be appropriate. If you change the default maximum size, then use BLOB and CLOB specifications that match your new maximum.

You also have the option to control the space storage and data recording for LOB columns. The options are:

COMPACT or NOT COMPACT:

Allow users to control a space-time trade-off in storage of the LOB data in your column.

If COMPACT is specified, the LOB data will occupy minimum space on disk, but there may be a performance penalty for any update that increases the size of a LOB.

If NOT COMPACT is specified, some extra space will be allocated to allow the LOB values room to grow. The default is NOT COMPACT.

LOGGED or NOT LOGGED:

Allows users to control whether updates on your LOB columns are recorded in the system log. In making this decision, you will need to consider the size of your LOB data, how valuable it is, and how easily it can be reconstructed. The default is LOGGED.

If LOGGED is specified, the LOB data in this column is treated exactly like all other data. Whenever the column is updated, the new value is recorded in the system log. This provides the maximum protection for the data, but it is costly both in terms of time and disk space. The System log is needed to restore all the committed transactions while recovering a damaged database.

If NOT LOGGED is specified, changes to the LOB column are not recorded in the system log. Another part of the RDBMS, called shadowing, still remains in effect.

The syntax is:

BLOB/CLOB(nK/ nM/ nG) LOGGED/NOT LOGGED NOT COMPACT/COMPACT.

Here are some examples:

CLOB(2K) NOT LOGGED COMPACT
BLOB(2K) NOT LOGGED COMPACT
CLOB(2M) LOGGED COMPACT
BLOB(2M) LOGGED COMPACT
CLOB(2M) LOGGED NOT COMPACT
BLOB(20M) NOT LOGGED COMPACT
CLOB(4000) LOGGED COMPACT
BLOB(10000) NOT LOGGED COMPACT

To set up the sqldbinfo file for DB2Universal:

1 Open the sqldbinfo file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

The SQL DB Type search form is displayed.

2 Select **db2universal** from the SQL DB Type and click **Search** or press Enter.

The sql.db.info form is displayed.

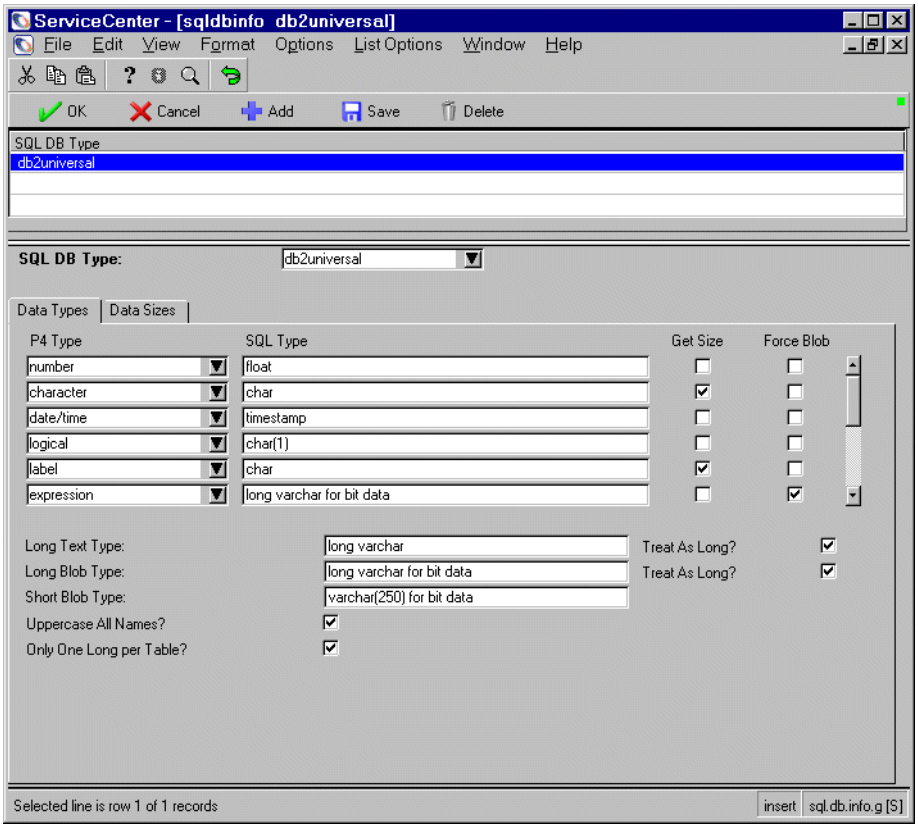
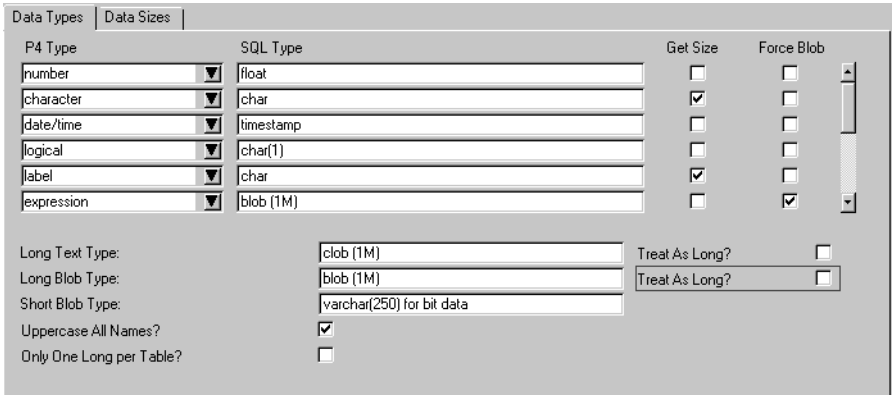


Figure 2-11: sql.db.info form - DB2Universal Data Types Tab

3 In the **Long text type:** text box, replace **long varchar** with **CLOB(nK/ nM/ nG) LOGGED / NOT LOGGED NOT COMPACT/COMPACT**. Uncheck **Treat as Long?** if it is checked as default.

4 In the **Long Blob type:** text box, replace **long varchar** with **BLOB(nK/ nM/ nG) LOGGED / NOT LOGGED NOT COMPACT/COMPACT**. Uncheck **Treat as Long?** if it is checked as default.

5 Uncheck **Only One Long per Table** if it is checked as default. When your are done the screen will look something like this:



6 Select the **Data Sizes** tab.

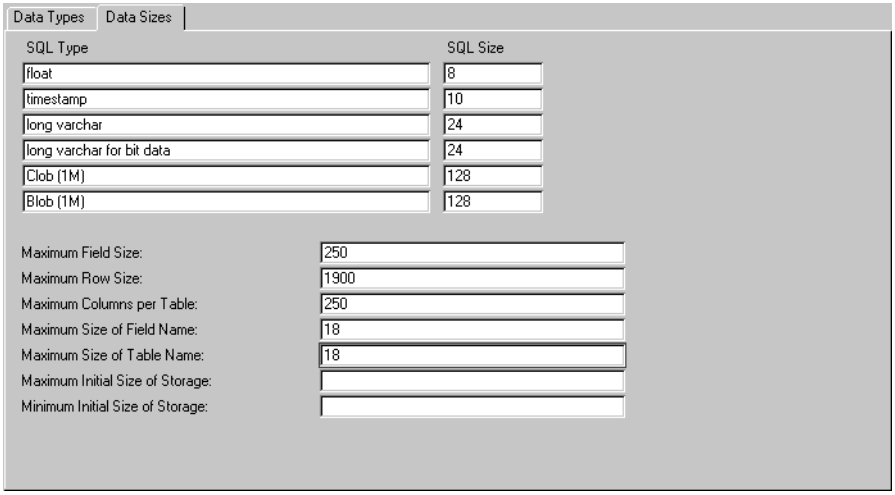


Figure 2-12: sql.db.info form - DB2Universal Data Sizes Tab

7 Add **CLOB(nK/ nM/ nG) LOGGED / NOT LOGGED NOT COMPACT/COMPACT** to the **SQL Type** box and **N** to the corresponding **SQL Size** box. For more information on the SQL size value, see *The SQL Size Value* on page 57.

8 Copy what you entered in step 3, and paste it here. The value added here must exactly match what was entered in step 3.

Note: All sizes in the **Data Sizes** tab are in bytes.

9 Add **BLOB(nK/ nM/ nG) LOGGED / NOT LOGGED NOT COMPACT/COMPACT** to the **SQL Type** box and **N** to the corresponding **SQL Size** box. Copy the you entered in step 4, and paste it here. The value added here must exactly match what was entered in step 4.

10 Save the settings and click **OK**.

The SQL Size Value

The **SQL Size** value is the size of a LOB locator. For DB2Universal, **SQL size** varies according to the maximum length (nK/nM/nG) defined for the LOB column.

The following table shows typical sizes:

Maximum LOB length	LOB Locator Size
1024	72
8192(8K)	96
65,536(64K)	120
524,000	144
4,190,000	168
134,000,000	200
536,000,000	224
1,070,000,000	256
1,470,000,000	280
2,147,483,647	316

Using the table, N will be calculated as:

For **BLOB(2K)** N will be 96
For **CLOB(2M)** N will be 168
For **BLOB(20M)** N will be 200
For **CLOB(4000)** N will be 96
For **CLOB(64k)** N will be 120

The maximum row size and maximum columns per table should be changed based on the largest DB2 page size. We recommend using a 32K page size for ServiceCenter data.

Page Size	4K	8K	16K	32K
Maximum Columns in Table	500	1012	1012	1012
Maximum Row Size	4005	8101	16293	32677

LOB Support for External Databases

The LOB data type is supported for the external databases Oracle and DB2Universal. The user is given the option to choose LOB over LONG and to choose a LOB tablespace that is dedicated to store large objects. The user is also given an option to choose a separate tablespace to store text/image type of data in SQL server.

Separating the large objects from the rest of the data can improve the clustering properties of the table. It can also reduce the disk contention and the number of I/O options needed to scan the table.

The option to separate large objects from the rest of the data is accessed by clicking **Advanced Options** under **Utilities** > **SQL Utilities** > **Move files from SQL to P4**. Put the specified tablespace names in the text boxes Lob Table Space Name and Lob Index Space Name.

Lob Table Space Name applies to Oracle, DB2Universal and SQL server and ignored for the rest of the other databases that ServiceCenter supports.

Lob Index Space Name is only applied to Oracle 8.0.x. and defaults to **Lob Table Space Name** for Oracle8i.

Lob Table Space Name is used to specify the name of a tablespace that has already been created in the database to store the LOBs in Oracle and DB2Universal or text/image in SQL server. It is assigned to the lob or text/image columns during the creation of a table in Oracle, DB2Universal and SQL server.

By issuing a simple SQL command, the text/image columns in Sybase can be easily moved to a separate LOB tablespace after a table is created. Check the Sybase manual for details.

Specifying LOB Tablespace for Oracle

In Oracle 8.0.x the LOB index can be stored separately from the LOB segment. If a tablespace is specified for the LOB segment then the LOB index will be placed in the same tablespace unless a different tablespace is explicitly specified. System generated names are used unless you specify names for the LOB segments.

In Oracle 8I, Even though you can specify a tablespace for the LOB index, Oracle is now ignoring it and placing the index in the same tablespace as the LOB data. The 8.1 manual suggests the LOB index tablespace is ignored if it is different from the LOB tablespace.

When LOB data is used, slowness may be noticed during the conversion of some files from P4 to Oracle. The reason for the slowness is that Oracle inserts a LOB data in two steps. First it selects the LOB locator(s). Then it inserts the data into the selected locator(s). One way to try to speed up the conversion is to increase the number of db_block_buffers or the size of the shared_pool_size in the Oracle initialization file, such as initora.ora on the server side. Consult your Oracle DBA for a solution if a severe slowness is detected during file conversion.

Specifying LOB Tablespace for DB2Universal

In order to use a separate tablespace for LOB data, DB2Universal database also needs a DMS (Database Managed Space) tablespace for the non-LOB data specified in the **Table Space Name** box.

To create the tablespaces by issuing SQL commands:

1. CREATE TABLESPACE dmsRegular MANAGED BY DATABASE USING (FILE 'd:\dms2\dms2.dat' 10000, FILE 'd:\dms2\dms2.dat' 10000);
2. CREATE LONG TABLESPACE dmsLOB MANAGED BY DATABASE USING (FILE 'FILE 'f:\longspace\space1.dat' 50000);

   Assume a table is mapped as Tablem1 (col1 varchar(30), col2 double, clo3 CLOB(1M), col4 BLOB(1M)).
3. Specify dmsRegular in **Table Space Name** box and dmsLOB in **Lob Table Space Name** box.

   ServiceCenter will issue a SQL statement as:

   CREATE TABLE Tablem1 (col1 varchar(30), col2 double, clo3 CLOB(1M), col4 BLOB(1M)) IN dmsRegular LONG IN dmsLOB.

dmsRegular is assigned to store the NON-LOB columns and dmsLOB the LOB columns.

4 For Db2 7.1, do following additional steps:

a Open the db2cli.ini file.

b Add the name of the database that you are using to store LOB under the datasource section, if it is not there.

c Under the database section, add a temp folder to store the temporary files created when working with the LOB data.

For example:

```
[DATASOURCE]
[WDU]
TEMPDIR=D:\db2temp
```

Note: The name of the database used to store LOB data is WDU.

Exceptions to Mapping Rules

System Tables

Certain ServiceCenter tables contain system-only data, such as the RAD programming language. These tables store information in its executable format, or the structures of display formats. When these records are moved to RDBMS, the entire record is translated as a *blob* and stored in the record's main RDBMS table. Any key fields from the ServiceCenter records are, however, mapped normally to enable efficient retrieval.

Indexed Arrays

ServiceCenter's internal data retrieval methods allow the construction of indexes on arrays. It is perfectly legal in ServiceCenter to place an index on an array of characters and retrieve, via that index, only those records whose arrays contain a specific element. In order to allow this same capability in RDBMS, ServiceCenter will not allow you to map an indexed array as a *blob* or a *long text* field. ServiceCenter will *always* map an indexed array to an external array table.

ServiceCenter's Record Retrieval Strategy

Understanding ServiceCenter's record retrieval strategy will help you optimize your indexing and mapping choices. ServiceCenter performs record retrieval in three phases:

- Primary key fetch
- Initial record selection
- Block selection

Note: You may get different results when performing a query for a NULL field value when searching in a system that has been mapped to an RDBMS than you would if the data was stored in P4. This is because no Index entries for fields that contain a NULL value in an RDBMS are created in P4.

Retrieval Phase 1: Primary Key Fetch

The first phase of any ServiceCenter record retrieval is always a query for all primary key values from the main table which satisfy its criteria.

Example

Retrieve the records in the license file from the SQL database:

ServiceCenter Field	ServiceCenter Type	SQL Table	SQL Field Name	SQL Data Type
last.name	character	licensem1	last_name	char(60)
spreadsheet	array of characters	licensea1	spreadsheet	varchar(255)*
word.processing	array of characters	licensea2	word_processing	varchar(255)*

If a ServiceCenter user wants to fetch all users whose last name begins with the letter "J", the user would enter the last name and initiate the search. ServiceCenter would then issue this SQL query:

Select last_name from licensem1 where last_name like "J%" order by last_name ASC

ServiceCenter will fetch up to 500 records from the result set and close the query. For the purpose of this example, assume the following primary keys were returned:

Key Value	Number
Jaams	1
Jaans	2
...	...
Janes	33
...	...
Jones	500

Retrieval Phase 2: Initial Record Selection

After ServiceCenter initiates a query, it will always fully populate the first record in the result set. It does this by selecting data from the main SQL table and any associated alias tables. ServiceCenter will *not* issue a join request. In our example, ServiceCenter issues these queries to complete record selection:

- Select * from licensem1 where (last_name="Jaams")
- Select * from licensea1 where (last_name="Jaams")
- Select * from licensea2 where (last_name="Jaams")

Retrieval Phase 3: Block Selection

As soon as ServiceCenter realizes it will need more than one record in the result set, it initiates a *block selection*. This happens in either of two cases:

- ServiceCenter is asked to display a list of records on the screen.
- ServiceCenter application code attempts to navigate to the next record in the result set.

When a block selection is initiated, ServiceCenter attempts to retrieve a block of records at once to minimize the number of distinct queries it has to make against the RDBMS server. ServiceCenter attempts to fetch records in blocks of 32 from the RDBMS database. It will do so by querying the next 32 records from the system based upon the list of primary keys it has already selected in phase 1.

In our example, ServiceCenter issues the following queries:

- Select * from licensem1 where ((last_name="Jaans" or (last_name="Jab" or (last_name="Janes"))
- Select * from licensea1 where ((last_name="Jaans" or (last_name="Jab" or (last_name="Janes"))
- Select * from licensea2 where ((last_name="Jaans" or (last_name="Jab" or (last_name="Janes"))

Subsequent Retrievals

All subsequent requests for still more records from the query are handled as block fetches described in Phase 3, above. Once ServiceCenter has exhausted its initial request for 500 primary key values, it repeats Phase 1 to query out the next 500 primary key values that satisfy its criteria.

Performance Considerations

ServiceCenter's SQL mapping utilities give you a variety of options for mapping your data to an RDBMS. This section discusses how to optimize your mapping for either of two goals: pure performance, or reporting simplicity.

Optimizing for Speed

Minimize Tables and Rows in a Mapped Record

To optimize your ServiceCenter RDBMS implementation for speed means reducing physical reads on the RDBMS server. Generally speaking, ServiceCenter will fetch a single complete ServiceCenter record from your RDBMS database. To optimize the speed of this process, it is important to place a ServiceCenter record in as few rows in as few tables as possible.

If speed is an important issue, map your arrays of characters either as long text fields or long binary fields in the main RDBMS table. Bear in mind, however, that most commercial RDBMS do *not* actually store long text or binary data on the same page as the rest of a row. Even though a record appears to map to a single table, your RDBMS server must do secondary reads to fetch the long text or binary data from its repository.

Avoid Varchar Data Types

RDBMS usually employ a two phase strategy when asked to fetch specific columns from a table:

- Whenever possible, use an index–only query to fetch the requested columns.
- If the columns in question are not covered by an efficient retrieval index, the RDBMS will fetch the complete rows from the appropriate table and locate the column in question in order to return it.

This process is accomplished by locating the offset within the row itself where the appropriate column is located. If the table contains fixed length columns, this offset can be calculated based upon the table schema. If, however, the table contains variable length columns, then the row must be scanned to determine the actual physical offset of any given column. This scan process slows retrieval.

Avoid the use of *varchar* or *varbinary* data types to map ServiceCenter data to an RDBMS database. Use *char* or *binary* instead.

Index Efficiently

ServiceCenter does not usually place an extremely high update/insert transaction load on an RDBMS server. The majority of its interactions with the RDBMS database involve simple, single row selects. The speed of these selects can be improved by efficiently indexing your RDBMS tables. Avoid the temptation to under-index; this may speed updates or inserts, but will slow down retrieval.

Defining System Files

All ServiceCenter control files are stored as *system* files in the RDBMS. When the P4 file system is converted, only the key fields in these files are converted to unique columns in the RDBMS. The entire *descriptor* structure, containing all data fields, is mapped to a single BLOB. This mapping provides fast access to and extremely fast conversion of this data for use within ServiceCenter.

To be stored in this fashion, these files must be defined as system files in the **systables** record in the **globallists** file prior to RDBMS conversion. The **systables** global list is built from records in the **sqlsystemtables** file.

To define a file as a system file:

1 Open the sqlsystemtables file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

A blank SQL system tables record is displayed.

2 Enter the name of the file you want to designate as a system file in the dbdict **Name** field.

3 Check the **Map as BLOB** check box.

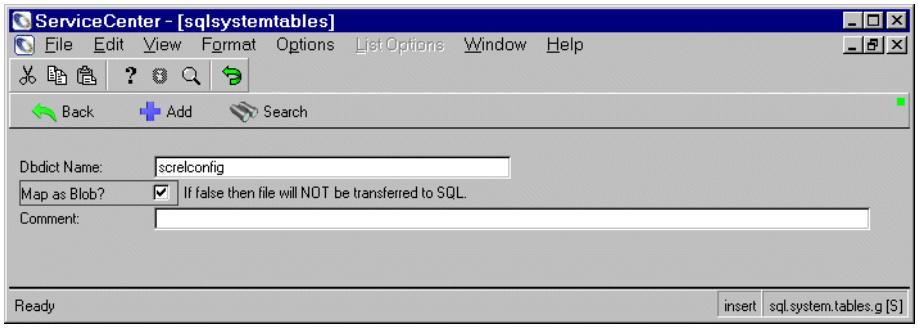


Figure 2-13: Sql System Table Record

4 Click Add.

The record is added to the sqlsystemtables file.

Optimizing for Space

Avoid Long Text or Long Binary Data Types

On most RDBMS, long text and long binary data is stored externally from the main tables as a linked list of small (2k) pages. Whenever data is stored in such a field, *at least one* 2k page is always allocated to hold it. Storing a 5 byte record in a Sybase image field actually takes 2k + 16 bytes (for the offset pointer) on the SQL server. Likewise a record containing 1 byte more than 2k spans a pair of 2k pages and will take 2k + 2k+ 16 bytes, or just over 4k of space on the SQL server.

Avoid the Use of Char And Binary Data Types

These data types always take a fixed amount of space, regardless of the actual quantity of data stored within. A **char(100)** will take 100 bytes, even if it actually only contains the word "hi". Conversely, a **varchar(100)** will use 3 bytes to store the word "hi" (1 byte for the length datum, and two for the data).

Avoid Excessively Long Character Fields

If the longest element of data you will ever store in a field is 20 characters long, do not allocate a 50 character long field to hold it, even if you are allocating a **varchar(50)**. Varchar and varbinary data do not waste space when it stores data in a table, but a **char(50)** always takes up 50 bytes in an index, even if the record being index only contains the word "I".

Optimizing for Reporting

Avoid Binary Data

Binary *blob* translation of ServiceCenter data is very fast from a retrieval standpoint; however, binary data is impossible for most commercial reporting packages to read. Therefore, arrays upon which you wish to report should be stored as either long text fields or as separate array tables. If you choose to use long text fields, check to ensure that your desired reporting package can handle the data in question.

Use Long Text Fields Carefully

ServiceCenter will map an array of numbers as a long text field, but will not perform row aggregate functions against the contents of that array. For example, if you map an array containing 35 price quotes for a PC into a long text field, you will be unable to select out the maximum, the minimum, or the mean price quote using the standard SQL syntax.

Generally speaking, you should map arrays on which you will need to perform row–level functions as array tables.

Reusable SQL

Reusable SQL is intended to improve the performance of ServiceCenter when it is mapped to an Oracle or a DB2/Universal database. Reusable SQL changes the SQL queries that are generated by ServiceCenter to allow the RDBMS to cache the results of parsing the query for future use. For example, typical ServiceCenter queries for records within a table might look something like this:

```
SELECT*FROM CATEGORYM1 where name="facilities";
SELECT*FROM CATEGORYM1 where name="abends";
```

This approach requires as many queries as you have categories. Each query is unique, which forces the RDBMS to parse and analyze each query. Eventually, the Oracle buffers that are used to remember previous queries are filled, causing performance degradation.

When reusable SQL is enabled, the query issued changes to the following:

```
SELECT*FROM CATEGORYM1 where name=?;
```

The question mark (?) in this query is called a *placeholder*. Before issuing the query we provide the system with the value that should be substituted for the placeholder. By doing this, Oracle and DB2 only have to parse and analyze the query once.

Analyzing Tables and Indexes

When using reusable SQL with Oracle, all tables and indexes must be analyzed. Failure to do this will cause Oracle to automatically use the COST based optimizer, resulting in performance degradation. The COST based optimizer needs the statistics generated by the ANALYZE command to work properly.

The following commands can be used to create a script that will analyze the tables and indexes:

```
Select 'analyze index '||owner||'.'||index_name|| ' compute statistics;'
from sys.dba_indexes
where owner in (list of owners for ServiceCenter indexes);
Select 'analyze table '||owner||'.'||index_name|| ' estimate statistics sample 20
percent;'
from sys.dba_indexes
where owner in (list of owners for ServiceCenter tables);
```

Parameter

Reusable SQL is enabled in ServiceCenter by default. If you want to disable this function, add the following line to your server sc.ini file:

```
sqlreuseablesql:0
```

Lightweight Directory Access Protocol (LDAP)

ServiceCenter support for Lightweight Directory Access Protocol (LDAP) directories provide a central point to define an organization's infrastructure, e.g., information on users within an organization, including email addresses, phone, fax, user IDs, passwords, and privilege levels. This central point can be accessed by multiple applications, as opposed to duplicating this information in records within each application.

LDAP is not another database, and therefore is not an alternative to P4, SQL server or another RDBMS. An associated database remains a requirement for ServiceCenter even if LDAP is implemented. LDAP allows certain information required for ServiceCenter operations to be located in a common directory database.

The operator file in ServiceCenter has been configured by default to allow it to tie into an LDAP directory. LDAP mappings can also be built for other system files using the LDAP mapping utility and LDAP map templates. Once a file has been mapped, updates in a LDAP directory are immediately seen in the mapped ServiceCenter file. Changes made to mapped field values within ServiceCenter update the values in the LDAP directory, if the user making the update has update rights to the LDAP server.

New records added to mapped files within ServiceCenter create new entries within the LDAP directory, if the user adding the record has add rights to the LDAP server. New entries in the LDAP directory created in this manner will only contain values for those attributes that have been mapped to a ServiceCenter field, i.e., not all file information appearing in the ServiceCenter record will be included in the LDAP directory entry.

Note: Deletions of files or records from within ServiceCenter are not propagated to the LDAP directory. In this way, data potentially required by other applications using LDAP is preserved. Actual deletion of data from LDAP should be performed from LDAP directly.

Refer to the ServiceCenter *System Administrator's Guide* for more details on using the LDAP retrieval interface protocol, including connecting to the LDAP directory, the mapping utility, and map templates.

CHAPTER 3

Data Mapping

This chapter was designed to aid ServiceCenter system and database administrators gain an understanding of mapping in ServiceCenter, and help them to map data more quickly and efficiently.

Topics in this chapter include:

Out-of-Box Mapping

Since SQL mapping is used to create SQL attributes for P4 files, the performance of ServiceCenter in a RDBMS is related to how a file is mapped. In order to simplify the mapping process and assist users in tuning ServiceCenter performance in RDBMS, ServiceCenter includes a standard out-of-box mapping. The standard mapping can be used as is, or modified to suit the needs of your company.

The following database types are pre-mapped:

- oracle8i
- db2u
- sqlserver7
- sqlserver2k
- sybase

The out-of-box mappings use separate tablespaces for INDEX, LOB, and regular tables. The tablespaces in the mappings point ServiceCenter to the specified storage for the tables and indices during ServiceCenter table creation.

To use an out-of-box mapping during conversion, choose one of the available database types from the drop-down list during the conversion process.

Table 3-1: Available Database types and their out-of-box mappings

Database Type	Mapping to Select
oracle8i	oracle8
db2u	db2ux
sqlserver7	sqlserver7x
sqlserver2k	sqlserver2kx
sybase	sybasex

Note: The mappings are subject to change based upon further performance testing and suggestions.

Basic steps in using an out-of-box mapping:

Step 1 Choose the database type to be used during the conversion process. See the table *Available Database types and their out-of-box mappings* on page 72.

Step 2 Create the table spaces. See *Tablespaces* on page 73.

Step 3 Modify the SQL mappings. See *SQL Mapping* on page 74.

Step 4 Select the SQL mapping options. See *SQL Mapping Options* on page 77.

Step 5 Use the out-of-box mapping when doing the conversion process. See *Conversion to an RDBMS* on page 161.

Tablespaces

The mappings are based on the tablespaces in the lists below. If those tablespaces do not exist, create them before doing the conversion.

db2ux

You may use either DMS (database managed system) or SMS (system managed system).

The mapping Db2uxi requires the following tablespaces:

Type of Space	Name	Type of Data
Tablespace	SCTEST	Hosts general ServiceCenter tables.
Tablespace	SCINDEX	Hosts ServiceCenter indices.
LOB tablespace	SCLOB	Hosts LOB data.*

*By default, db2ux maps ServiceCenter binary data or Array of characters as clob/blob. They can be changed back to long/long raw, or long varchar/long varchar for bit by resetting the parameters in file sqldbinfo. If parameters are reset in sqldbinfo not using clob/blob, then SCLOB is not used by ServiceCenter, then do not create it.

sqlserver7x and sqlserver2kx

The mapping **Sqlserver7x** or **Sqlserver2kx** requires the following tablespaces:

Type of Space	Name	Type of Data
Tablespace (file group)	SCTEST	Hosts general ServiceCenter tables.*
Tablespace	SCINDEX	Hosts ServiceCenter indices.
LOB tablespace	SCTEXT	Hosts text/image data.

* Different from system tables

sybasex

The mapping **Sybasex** requires the following tablespaces:

Type of Space	Name	Hosts
Tablespace (segment)	SCTEST	Hosts general ServiceCenter tables. *
Tablespace	SCINDEX	Hosts ServiceCenter indices.

* Different from system tables

SQL Mapping

During SQL mapping, a file's DDL options are saved in the file called **sqlmapping**. ServiceCenter will use the mappings in **sqlmapping** if the option to use existing mappings is chosen during conversion.

The **sqlmapping** file contains the RDBMS attributes that are used by ServiceCenter during table creation. These attributes include table names, data type, and table sqloptions etc. The saved mappings of a file can be viewed and changed in the **sqlmapping** file using Database Manager. The table **sqloptions** contain the table space, index space and object attributes that are used in the CREATE TABLE and CREATE INDEX statements.

The advantage of saving SQL mappings in the sqlmapping file is that it will allow each P4 file to share the same default settings or allow an individual table to have its own table options.

To view or modify an existing mapping:

1 Open the sqlmapping file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

The sqlmapping file is displayed.

— Or —

To update an sqlmapping file:

1 Open the **SQL Utilities Menu**. For information on how to open the **SQL Utilities Menu**, see *To begin conversion:* on page 171.

The **SQL Utilities Menu** (Figure 5-5 on page 171) is displayed.

2 Click **Update Mapping Template**.

3 Enter the name of the file you want to map in the **File Name** field.

4 Select the database type from the **DB Type** field.

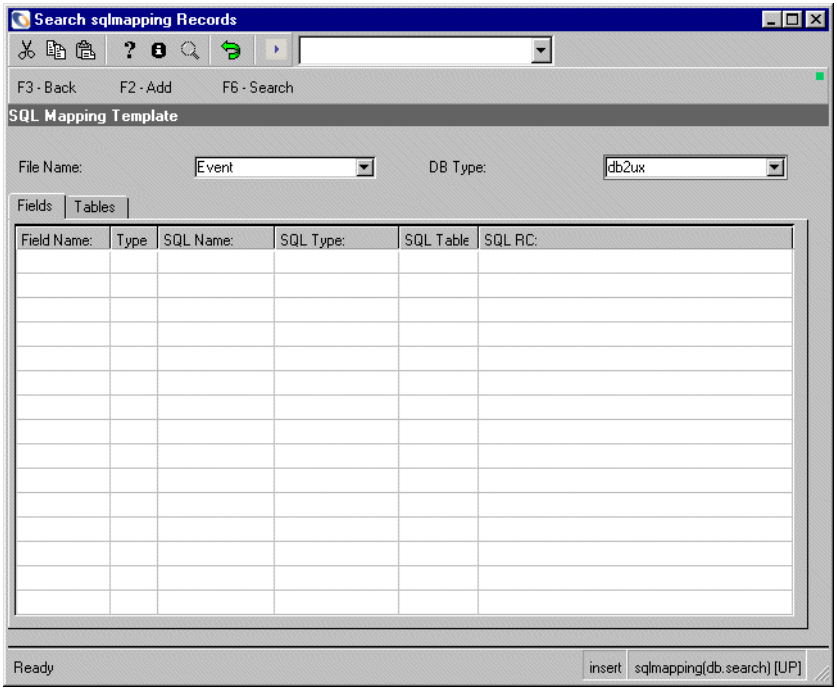
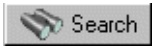


Figure 3-1: Specifying a Map to Edit



5 Click **Search** or press Enter.

The requested data map is displayed.

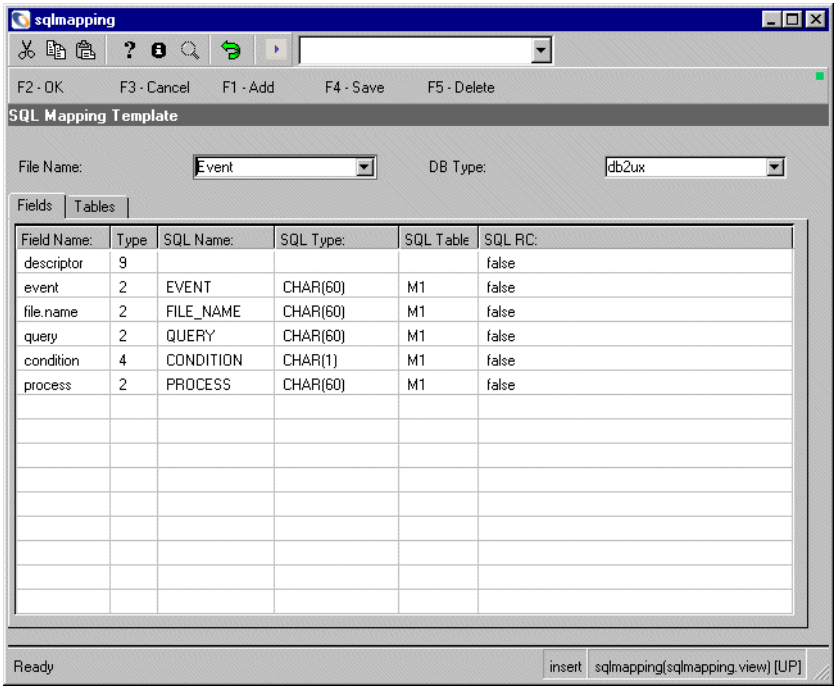


Figure 3-2: Data Map

6 Modify the mapping information for existing fields by clicking in the field and entering the updated values.

Note: New fields may be added through the Database Dictionary application. See *Database Dictionary* in ***System Tailoring*** for more information.

Proceed

7 Click **Proceed** when finished modifying the mapping.

A prompt is displayed asking if you want to save the new mapping.

8 Click **Yes** to save your changes.

Another prompt is displayed informing you that your mapping changes have been saved.

9 Click **OK** and exit the utility.

If there are records in the sqlmapoptions file corresponding to the selected record, the sqlmapoptions form for the selected record can be opened from this location by selecting **Get Mapping Options** from the ServiceCenter **Options** menu, or by clicking the **DB Type** or **Table Name** buttons. See *SQL Mapping Options* on page 77 for more information.

SQL Mapping Options

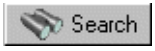
The form sqlmapoptions is used as an interface to allow users to assign the specified DDL options to an individual table. These options will be used during table creation. ServiceCenter saves the input from this interface in the sqlmapoptions file first and then puts it in the sql.table.options field of the sqlmapping file. If a table's DDL options are not assigned from sqloptions, the default options will be used and shared by all tables.

To edit the sqlmapoptions file:

1. Open the sqlmapoptions file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

   The sqlmapoptions file is displayed.

2. Search for the file you want to edit the mappings for.



3. To search a particular record or set of records, enter the DB Type and Tablename, then click **Search** or press Enter.

   If you search with no data entered, all records in the sqlmapoptions file will be returned.

   — Or —

   To open the sqlmapoptions for a particular sqlmapping file, click the **DB Type** or **Table Name** button in the open sqlmapping file.

The sqlmapoptions file is displayed.

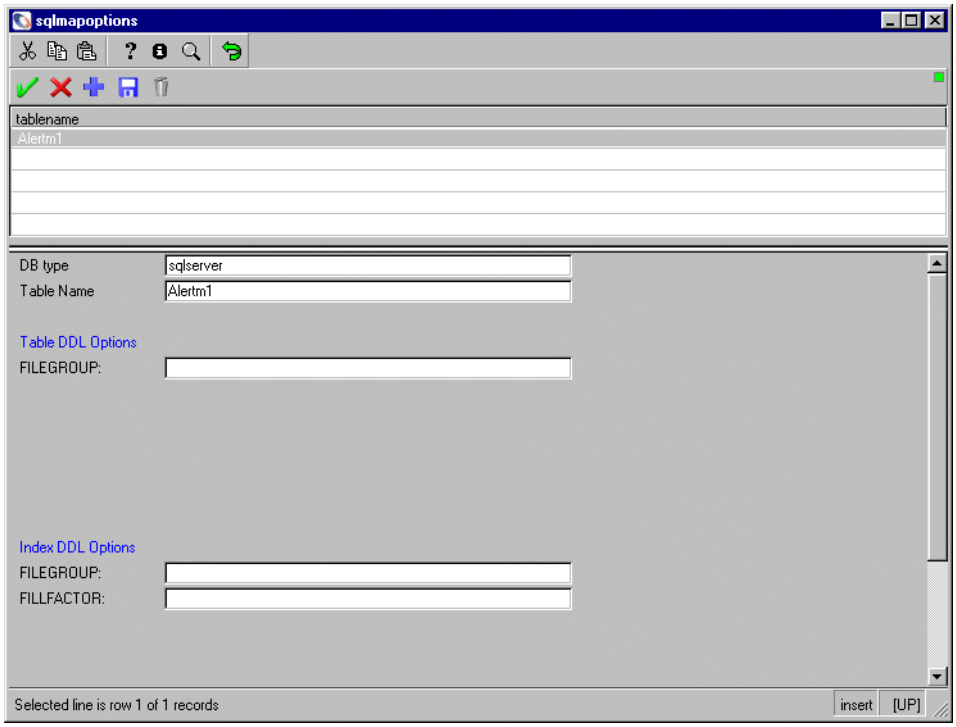


Figure 3-3: *sqlmapoptions* File

4 On the form sqlmapoptions, there are two groups of options, Table DDL and Index DDL, which are used in the CREATE TABLE and CREATE INDEX statements, respectively.

5 Edit the Table DDL and Index DDL options, defined below for the following database types:

- *DB2Universal* on page 91
- *Microsoft SQL Server* on page 93
- *Oracle* on page 94
- *Sybase* on page 95

The options for only one database type will be available. This is a limitation of the sqloptions file and is set by the last SQL conversion that was performed.

Note: Each RDBMS has its own definitions for Table DDL and Index DDL options and users are strongly suggested to refer to the RDBMS's administrative guide for details.

Custom Mappings

ServiceCenter's default mapping options are discussed in *RDBMS Mapping Strategy* on page 36. The mappings we provide are discussed in *Out-of-Box Mapping* on page 72. You may want to map your fields differently.

Changing Mapping of Scalar Fields

ServiceCenter stores its RDBMS mapping options in a file called sqldbinfo.

To access sqldbinfo:

1. Open the sql.db.info form (or the sqldbinfo file) in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)
2. Select the RDBMS from the **SQL DB Type** drop-down list of relational databases supported in ServiceCenter.

The following **sql.db.info** form is displayed for the RDBMS server:

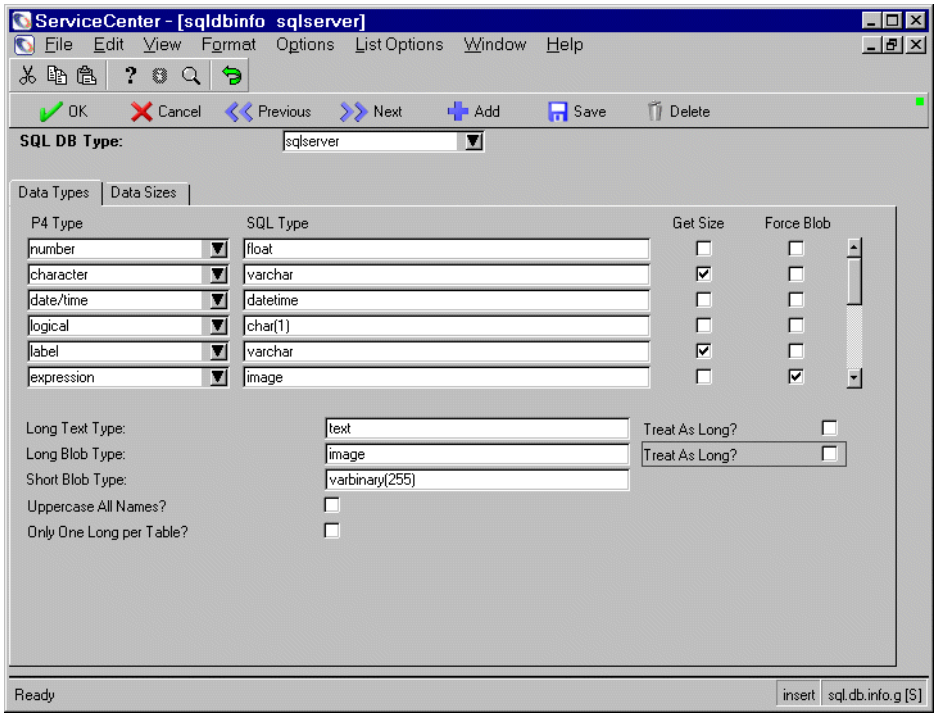


Figure 3-4: RDBMS Database Information

3 To select the Data Type to which a specific ServiceCenter type should be mapped on your RDBMS, enter the new **SQL Type** in the appropriate row.

The flag selected tells the ServiceCenter RDBMS mapping utility what to do when it constructs RDBMS tables to hold ServiceCenter data.

Field	Definition
Get Size	Calculates the size based on data currently in P4. If Get Size is selected, ServiceCenter will automatically affix the appropriate length operator to the field when it is created. For example, char will become char (20) or char (120).
Force Blob	Forces the use of BLOB. If Force Blob is selected, the ServiceCenter data will be stored in the RDBMS field in internal ServiceCenter binary format.

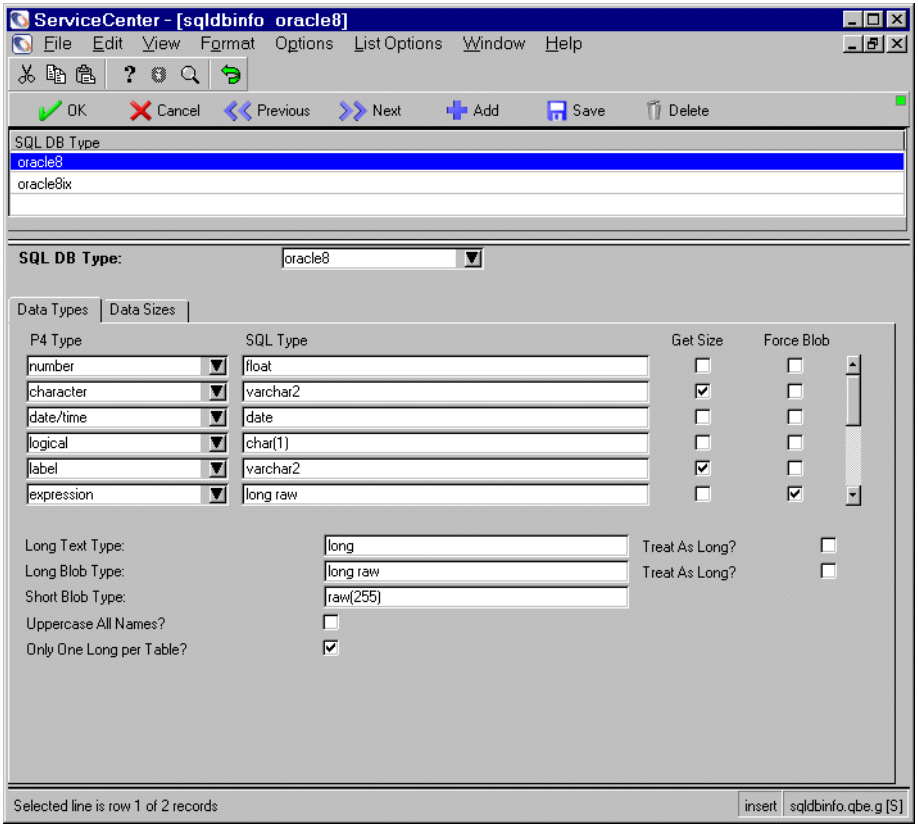


Figure 3-5: RDBMS Database Information

Unless you are absolutely certain of what you are doing, Peregrine recommends you *do not* alter the defaults for anything except to the four simple data types (character, number, date/time and logical).

Mapping Fields to a Null Table

Since ServiceCenter applications and files can be customized, some fields may be unused. To save storage space and reduce processing overhead, unnecessary fields can be mapped to a table called **nulltable**. This is not an actual table, but rather a keyword that causes the SQL interface to ignore all fields mapped to that table alias. The SQL interface does not create columns for fields mapped to **nulltable**, and SQL statements generated by the SQL interface do not reference such fields.

Note: Before you can do the mapping, you will need to do the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for details.

To map fields to nulltable:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To open the SQL Menu:* on page 170.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click **Move Files to P4 from SQL**. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

The P4 to SQL Conversion Console is displayed.

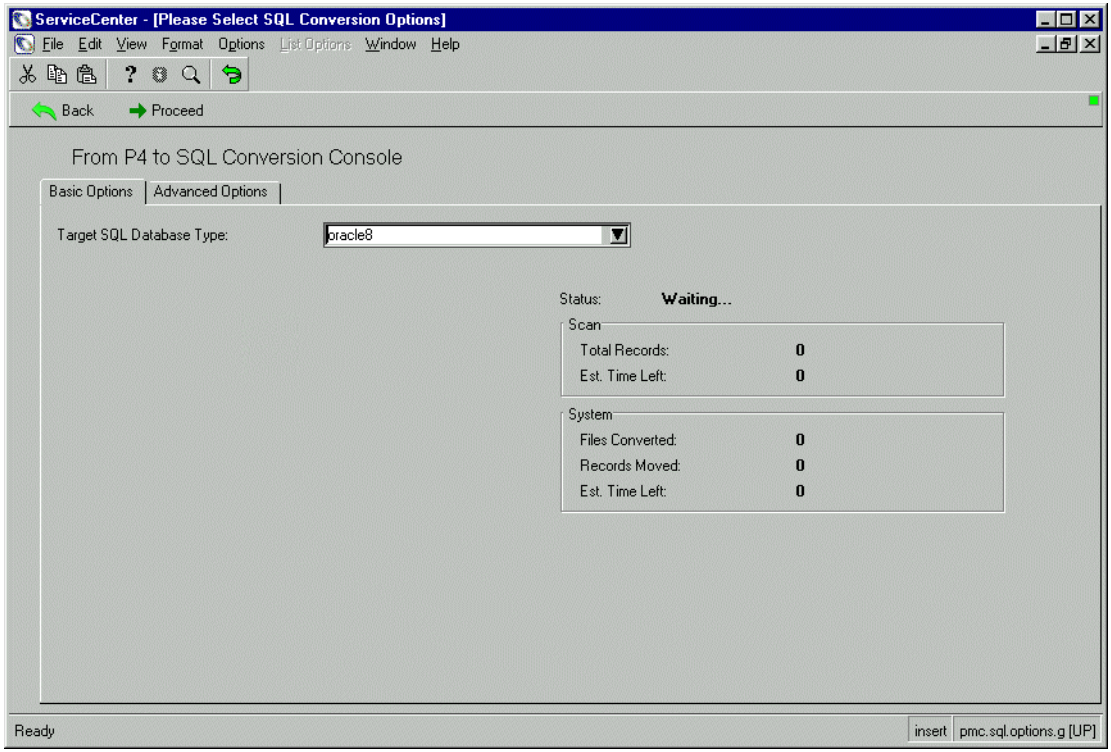


Figure 3-6: SQL Conversion Console

3 Select **Options** > **Single File** from the menu bar.

The **File to Convert** field is displayed, allowing you to name a file.

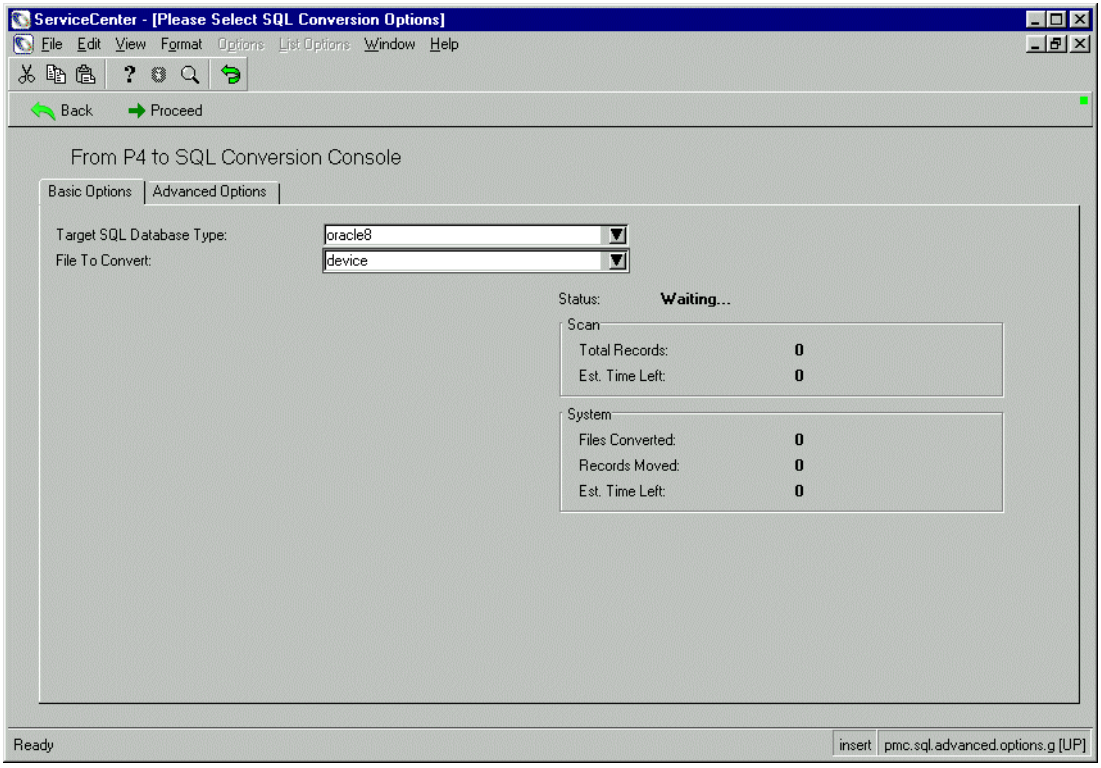


Figure 3-7: SQL Conversion Console for Mapping a Single File

4 Select the file you want to map from the drop-down list.

For this example, we have chosen the device file.

5 Select your target RDBMS from the drop-down list in the **Target SQL Database Type** field.

6 Select the **Advanced Options** tab.

7 Select the following Final Objective options:

- **Map Tables**
- **Manually Review Maps?**

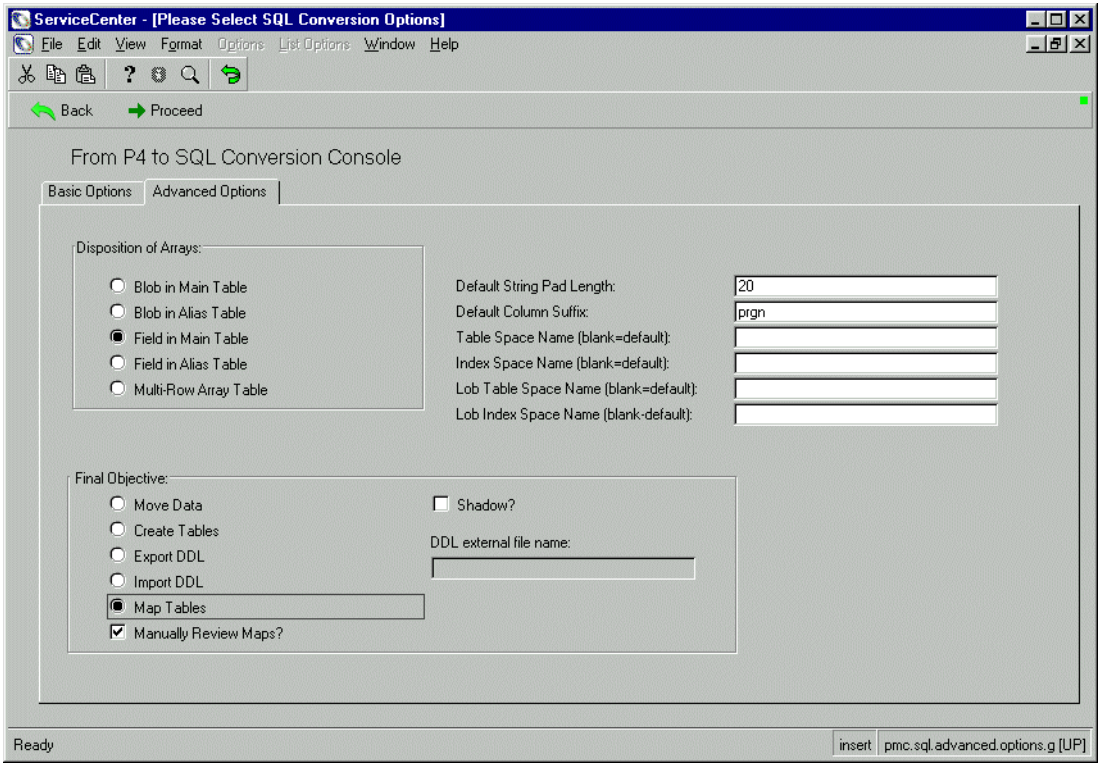


Figure 3-8: SQL Mapping Options

8 Click **Proceed**.

If the file you selected has already been mapped, you are given the option to use the existing map or create a new one.

9 Click **Yes** to use the existing map or No to remap the file.

The desired map is displayed.

For this example, suppose that the **container**, **end.point.1**, and **end.point.2** fields were unnecessary.

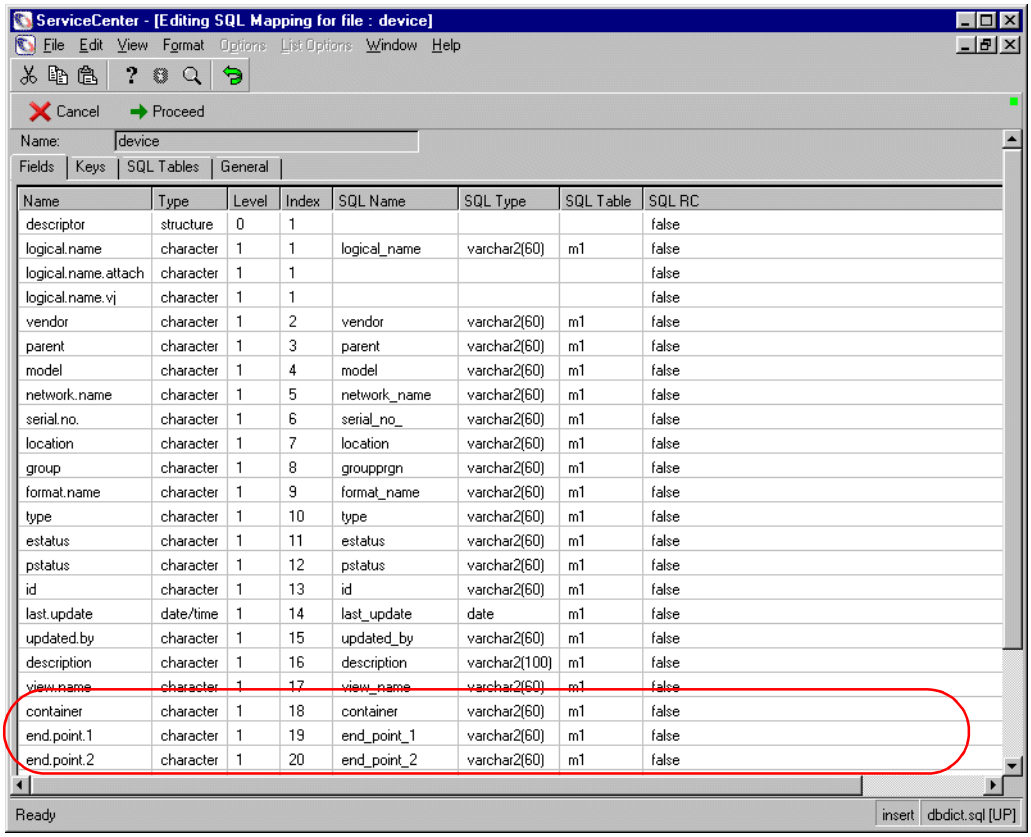


Figure 3-9: Fields Mapped to an Oracle Database

10 Select the **SQL Tables** tab.

11 Enter the following values:

Field	Value
Alias	n1
Name	nulltable
Type	*<target RDBMS name>* For example, **oracle8**

The alias of `n1` is an arbitrary designation. You may select any letter to define your nulltable.

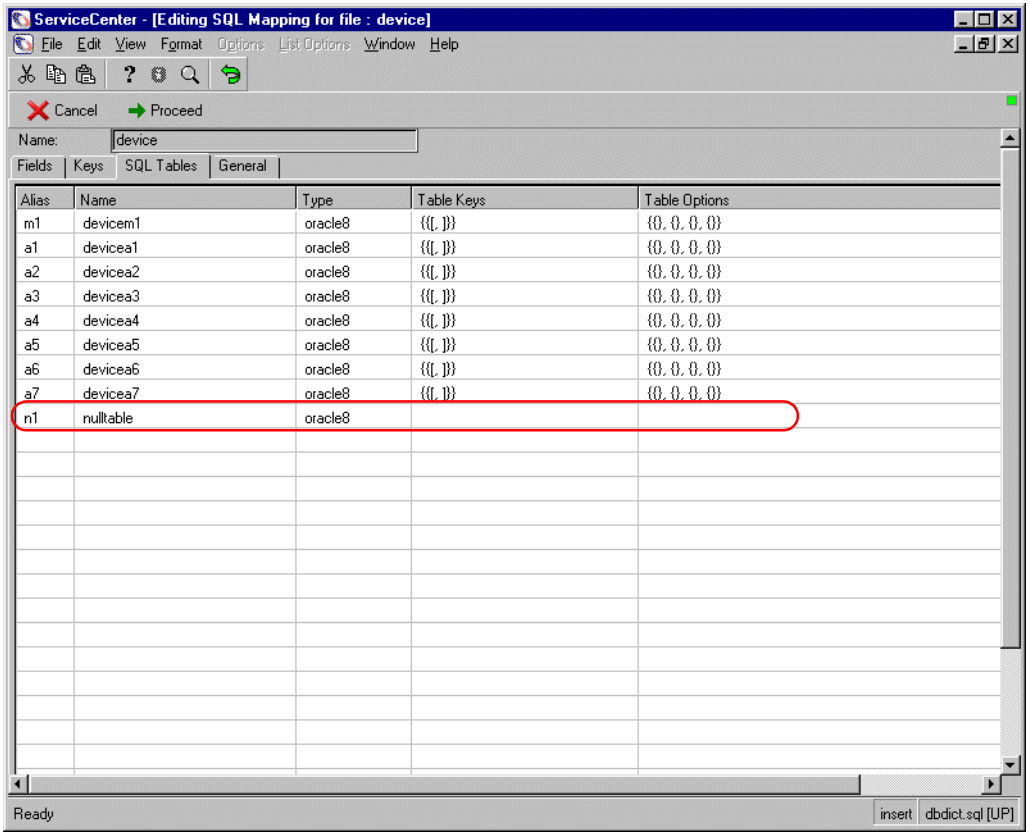
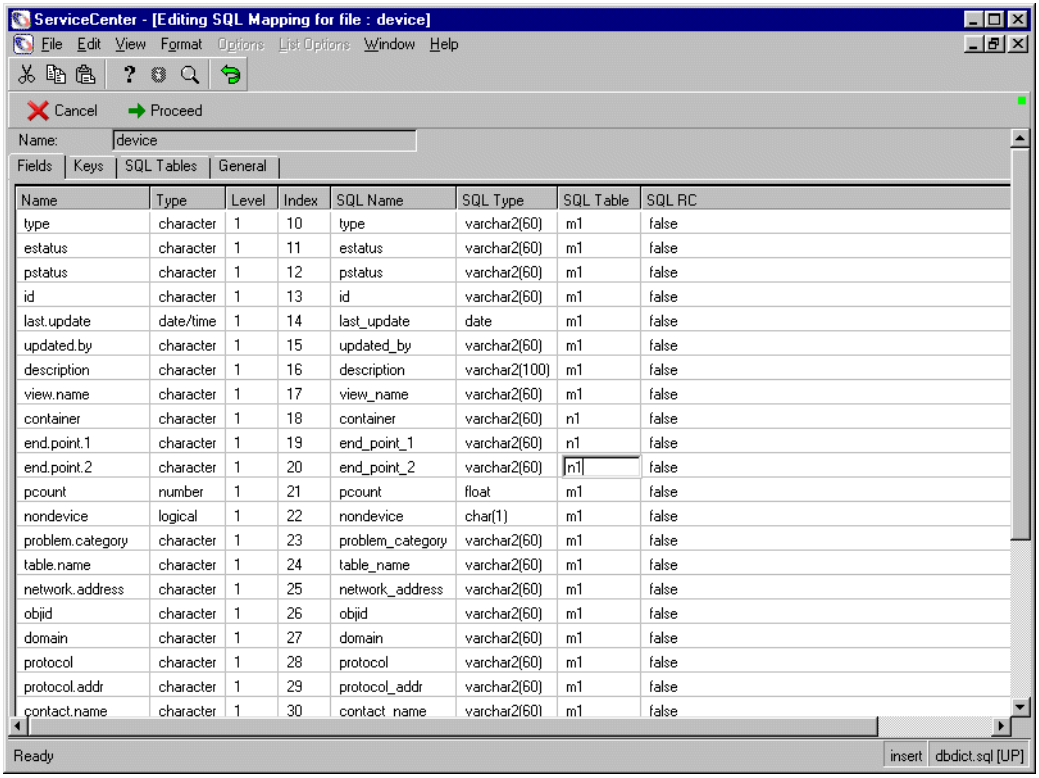


Figure 3-10: SQL Table Definitions

12 Select the **Fields** tab.

13 Change the value in the **SQL Table** field to `n1` for each field you want to map to the nulltable.



Name	Type	Level	Index	SQL Name	SQL Type	SQL Table	SQL RC
type	character	1	10	type	varchar2(60)	m1	false
estatus	character	1	11	estatus	varchar2(60)	m1	false
pstatus	character	1	12	pstatus	varchar2(60)	m1	false
id	character	1	13	id	varchar2(60)	m1	false
last.update	date/time	1	14	last_update	date	m1	false
updated.by	character	1	15	updated_by	varchar2(60)	m1	false
description	character	1	16	description	varchar2(100)	m1	false
view.name	character	1	17	view_name	varchar2(60)	m1	false
container	character	1	18	container	varchar2(60)	n1	false
end.point.1	character	1	19	end_point_1	varchar2(60)	n1	false
end.point.2	character	1	20	end_point_2	varchar2(60)	n1	false
pcount	number	1	21	pcount	float	m1	false
nondevice	logical	1	22	nondevice	char(1)	m1	false
problem.category	character	1	23	problem_category	varchar2(60)	m1	false
table.name	character	1	24	table_name	varchar2(60)	m1	false
network.address	character	1	25	network_address	varchar2(60)	m1	false
objid	character	1	26	objid	varchar2(60)	m1	false
domain	character	1	27	domain	varchar2(60)	m1	false
protocol	character	1	28	protocol	varchar2(60)	m1	false
protocol.addr	character	1	29	protocol_addr	varchar2(60)	m1	false
contact.name	character	1	30	contact_name	varchar2(60)	m1	false

Figure 3-11: Fields Mapped to Nulltable

14 Click **Proceed**.

15 Click **OK** to complete the process and return to the P4 to SQL Conversion Console.

DDL Options

DB2MVS

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
TABLESPACE *string*	Specifies the tablespace on which the table is created. If tablespace is specified, the table is created in the named tablespace. The tablespace must exist within the database. If it is not specified, the table is stored on the default tablespace.
AUDIT *string*	Identifies the types of access to this table that causes auditing to be performed. One of the following values can be chosen. CHANGES ALL NONE If it is not specified, NONE is used by the RDBMS as the default value.
OBID *integer*	An OBID is the identifier for an object's internal descriptor. If it is defined, Identifies the OBID to be used for this table. The OBID keyword is required if the database for the table was defined as ROSHARE READ.
DATA CAPTURE *string*	Specifies whether the logging of SQL INSERT, UPDATE, and DELETE operations on the table is augmented by additional information. One the following values can be used. NONE CHANGES If it is not specified, NONE is used by the RDBMS as the default value.

Index DDL Options

The following parameters are used in the CREATE INDEX statement.

Parameter	Definition
SUBPAGES *integer*	Gives the number of subpages for each physical page. (The subpage is the unit of index locking.) The default value of 4 is used if it is not specified. Refer to DB2MVS documentation for details.
BUFFERPOOL *string*	Identifies the buffer pool to be used for the index. It must identify an activated 4KB buffer pool and the privilege set must include SYSADM or SYSCTRL authority or the USE privilege for the buffer pool. The default is the default buffer pool of the database. If the default buffer pool of the database is a 32KB page buffer pool, the default is BP0.
CLOSE *string*	Specifies whether or not the data set is eligible to be closed when the index is not being used and the limit on the number of open data sets is reached. One of the following values can be chosen. ■ YES ■ NO If it is not specified, YES is used by the RDBMS as the default value.
VCAT *string*	Specifies name of the data set that is cataloged in an integrated catalog facility catalog. If this field is specified, it means that the first data set for the index is managed by the user, and that following data sets, if needed, are also managed by the user.
PRIQTY *integer*	Specifies the minimum primary space allocation for a DB2-managed data set. The primary space allocation is at least *n* kilobytes, where *n* is: ■ 12 If *integer* is less than 12 or PRIQTY is omitted ■ integer If *integer* is between 12 and 4194304 ■ 4194304 If *integer* is greater than 4194304
STOGROUP *string*	Specifies a storage group that exists at the current server and the privilege set must include SYSADM authority, SYSCTRL authority, or the USE privilege for the storage group.

Parameter	Definition
SECQTY *integer*	Specifies the minimum secondary space allocation for a DB2-managed data set. The secondary space allocation is at least *n* kilobytes, where *n* is: ■ 12 If SECQTY and PRIQTY are omitted ■ 131068 If *integer* is greater than 131068 ■ integer If *integer* is not greater than 131068 ■ If *integer* is 0, no data set for the index can be extended.
ERASE (NO/YES)	Indicates whether the DB2-managed data sets are to be erased when they are deleted during the execution of a utility or an SQL statement that drops the index. Refer to *DFSMS/MVS: Access Method Services for VSAM Catalogs* for more information. ■ **NO** Does not erase the data sets. Operations involving data set deletion will perform better than ERASE YES. However, the data is still accessible, though not through DB2. This is the default. ■ **YES** Erases the data sets. As a security measure, DB2 overwrites all data in the data sets with zeros before they are deleted.
FREEPAGE *integer*	Specifies how often to leave a page of free space when index entries are created as the result of executing a DB2 utility or when creating an index for a table with existing rows. One free page is left for every *integer* pages. The value of *integer* can range from 0 to 255. The default is 0, leaving no free pages.
PCTFREE *integer percentage*	Determines the percentage of free space to leave in each nonleaf page and subpage when entries are added to the index or index partition as the result of executing a DB2 utility or when creating an index for a table with existing rows. The first entry in a page or subpage is loaded without restriction. When additional entries are placed in a nonleaf page, the percentage of free space is at least as great as *integer*. When additional entries are placed in a leaf page, the percentage of free space is at least a great as *integer*/*m*, where *m* is the number of subpages.

DB2Universal

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
TABLESPACE *string*	Identifies the table space in which the table will be created. The table space must exist, and be a REGULAR table space. If it is not specified, the table is stored on the default tablespace.
DATA CAPTURE *string* (NONE/CHANGES)	Indicates whether extra information for inter-database data replication is to be written to the log. This clause cannot be specified when creating a sub table. NONE indicates that no extra information will be logged. CHANGES Indicates that extra information regarding SQL changes to this table will be written to the log. This option is required if this table will be replicated and the Capture program is used to capture changes for this table from the log. If it is not specified, NONE is used by the RDBMS as the default value.
INDEX SPACE:string	Identifies the index space in which the index will be created. The index space must exit. If it is not specified, the index will share the same tablespace with tables.

Index DDL Options

The following parameters are used in the CREATE INDEX statement.

Parameter	Definition
PCTFREE *integer percentage*	Specifies what percentage of each index page to leave as free space when building the index. The first entry in a page is added without restriction. When additional entries are placed in an index page at least *integer* percent of free space is left on each page. The value of *integer* can range from 0 to 99. However, if a value greater than 10 is specified, only 10 percent free space will be left in non-leaf pages. The default is 10.

Informix

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
DBSPACE *string*	Specifies the dbspace on which the table is created. If dbspace is specified, the table is created in the named dbspace. The dbspace must exist within the database. If it is not specified, the table is created on the default dbspace.
FRAGMENTATION *string*	Specifies fragmentation scheme to be used during table creation. One of the following fragmentation schemes can be chosen. They are: ■ ROUND ROBIN ■ HASH ■ EXPRESSION The general goal of fragmentation is to balance I/O across multiple disks and distribute data evenly. If **FRAGMENTATION** is not specified, no fragmentation is used.
LOCK LEVEL *string*	Specifies the level on which data item is locked. One of the following lock levels can be chosen. They are: ■ PAGE ■ ROW ■ TABLE ■ DATABASE
EXTENT SIZE *integer*	Specify the initial amount of disk space that you want to allocate for row storage.
NEXT SIZE *integer*	Specify the amount of contiguous disk space that you want to allocate for additional row storage.

Index DDL Options

The following parameters are used in the CREATE INDEX statement.

Parameter	Definition
DBSPACE *string*	Same as under Table DDL Options.

Parameter	Definition
FRAGMENTATION string	Same as under Table DDL Options.
FILLFACTOR *integer percentage*	Specifies a percentage that indicates how full Informix server should make the leaf level of each index page during index creation. If it is not defined, 90 is used by Informix server as the default value.

Microsoft SQL Server

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
FILEGROUP *string*	Specifies the filegroup on which the table is stored. If filegroup is specified, the table is stored in the named filegroup. The filegroup must exist within the database. If it is not specified, the table is stored on the default filegroup. This parameter is available for sqlserver7 and higher.

Index DDL Options

The following parameters are used in the CREATE INDEX statement

Parameter	Definition
FILEGROUP *string*	Specifies the filegroup on which the index is created. The filegroup must exist in the database. This parameter is available for sqlserver7 and higher.
FILLFACTOR *integer percentage*	Specifies a percentage that indicates how full SQL server should make the leaf level of each index page during index creation. When FILLFACTOR is specified, SQL server rounds up the number of rows to be placed on each page.

Oracle

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
TABLESPACE *string*	Specifies the tablespace in which Oracle creates the tables.
PCTFREE *integer percentage*	Specifies the percentage of space in each data block of the table that is reserved for future updates to the table's rows. The value of PCTFREE must be a value from 0 to 99. The default value is 10 set by the RDBMS during installation.
PCTUSED *integer percentage*	Specifies the minimum percentage of used space that Oracle maintains for each data block of the table. The value of PCTUSED must be a value from 0 to 99. The default value is 40 set by the RDBMS during installation. The combination of PCTFREE and PCTUSED determines where new rows will be inserted into the existing data block or into new blocks. The sum of PCTFREE and PCTUSED must be less than 100.
INITTRANS *integer*	Specifies the initial number of transaction entries allocated within each data block allocated to the table. This value ranges from 1 to 255 and defaults to 1.
MAXTRANS *integer*	Specifies the maximum number of concurrent transactions that can update a data block allocated to the table. This limit does not apply to queries. This value can range from 1 to 255 and the default is a function of the data block size.
STORAGE CLAUSE *string*	Specifies the storage characteristics for the table. Storage should be allocated to minimize dynamic allocation of additional space. The input on this field should be in format of the storage clause in a CREATE TABLE statement. For example: (initial 5K next 5K minextents 2 maxextents 50 pctincrease 0). Refer to Oracle documentation for details.

Index DDL Options

The following parameters are used in the CREATE INDEX statement

Parameter	Definition
TABLESPACE *string*	Specifies the name of the table space to hold the index. If this clause is omitted, Oracle creates the index in the default tablespace of the owner of the schema containing the index.
PCTFREE *integer percentage*	Specifies the percentage of space to leave free for updates and insertions within each of the index's data blocks.
INITRANS *integer*	Same definition as under Table DDL Options.
MAXTRANS *integer*	Same definition as under Table DDL Options.
STORAGE CLAUSE *string*	Same definition as under Table DDL Options.

Sybase

Table DDL Options

The following parameters are used in the CREATE TABLE statement.

Parameter	Definition
SEGMENT_NAME *string*	Specifies the name of the segment in which to create a table.

Index DDL Options

The following parameters are used in the CREATE INDEX statement

Parameter	Definition
SEGMENT_NAME *string*	Specifies the name of the segment in which to create an Index. The **SEGMENT_NAME** must exist in the database.
FILLFACTOR *integer percentage*	Specifies the extent to which index pages should be filled (excluding the root) as indexes are created.

4 Preconversion Configuration

CHAPTER

This chapter was designed to explain the necessary preconversion configuration of a Relational Database Management System (RDBMS) to ServiceCenter system and database administrators.

Topics in this chapter include:

Server Configuration

This chapter builds from the premise that ServiceCenter and the RDBMS have already been installed.

General Space Requirements

Unlike most commercial RDBMS vendors, the ServiceCenter P4 file system stores data in a compressed format. Consequently, between four and five times as much storage space is required in a commercial RDBMS to contain converted data from the original P4 format.

If you are establishing a new ServiceCenter system, Peregrine recommends you allocate data space of at least 1 GB for a test system and 4 GB for a production system. This calculation is very conservative and normally produces a space large enough to hold your data.

If this estimate does not produce adequate room, use the following formula to calculate the necessary size of your RDBMS:

Data Space Size = Size of P4 data x 5

Note: Peregrine highly recommends placing all ServiceCenter data in a dedicated tablespace within a single instance of your RDBMS. This tablespace must contain ServiceCenter data only. Multiple instances will consume more system resources than a single instance solution.

Server Connections

Every ServiceCenter process, foreground or background, requires a connection to your RDBMS server. ServiceCenter background processes need a total of 13 connections to run. When you configure your database, be sure to allocate enough additional connections for all your users. Refer to the RDBMS documentation for configuration details.

Employing Multiple Databases with Oracle Call Interface

The ServiceCenter Oracle Call Interface (OCI) features allow data to be split and managed between multiple databases. This configuration enables data owned and maintained by different groups on different databases to be shared and accessed as if it were on the same database. One group of files can be stored on a database1, and other groups of files on database2, database3, etc.

You can map a single instance of ServiceCenter to different RDBMS tables. For example you may want to have your HR data and your ServiceCenter files in an Oracle table and your device inventory data in a DB2 table. This is accomplished by specifying a database section name in the sc.ini file.

See *Oracle Call Interface* on page 114 for more on enabling OCI.

RDBMS Connections

Login ID

You must create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. This login must have database administrator (DBA) authority in the target database. The login you provide must use the tablespace created earlier (see *Tablespaces* on page 73) as its default. When ServiceCenter logs in, it always creates its table in the default tablespace defined for that login ID.

Rebuild the Master Database

ServiceCenter no longer requires that any third-party relational database (RDBMS) be setup to use case-sensitivity for searching and sorting. However, the default installation of P4 is case-sensitive and the default installation of Microsoft SQL server is case-insensitive. Therefore, a default Microsoft SQL server installation cannot be successfully used for the ServiceCenter conversion, unless you first set up the P4 file system for case-insensitive searching.

You may either set up the ServiceCenter's P4 file system to be case-insensitive and keep MS SQL case-insensitive, or set up Microsoft SQL server to be case sensitive and keep P4 case sensitive; both must have the same sensitivity.

Warning: Rebuilding a Microsoft SQL server's master database will destroy any other tablespaces running on that server.

Limit the Transaction Log Size

During the conversion, ServiceCenter will place an extraordinarily high insert transaction load on your SQL server. To prevent the transaction log from growing disproportionately large, set the **Truncate Log on Checkpoint** option for the target database on your SQL server.

Oracle

The majority of tables on an Oracle server hold less than 50K of data. Peregrine now sets the initial storage space size when creating the SQL tables.

When manually creating a new Oracle instance for ServiceCenter on a Unix or Windows system, create the database with a block size of 8K or a multiple of that. It is also recommended to set *MAXDATAFILES* in the *CREATE DATABASE* statement to 100 or more.

Create a separate tablespace for the ServiceCenter data, and make this the default tablespace for the ServiceCenter user.

Set the *TEMPORARY* tablespace for the ServiceCenter user to an appropriate temporary tablespace.

Setting the Oracle environment variable

The files scenter.oracle and scenter.oraoci are built with oracle shared lib which needs an oracle RDBMS installed first in order to start scenter. After you install an oracle RDBMS, you may still need to reset your oracle environment. If you haven't installed oracle yet, please install it and then do the following.

To set your Oracle environment variable:

1. Find the path where libclntsh.so.8.0 or libclntsh.so.1.0 is located, for this example, ORACLELIB.
2. Set the environment variables as indicated below.

   On Solaris

   C shell: setenv LD_LIBRARY_PATH ORACLELIB
   Korn shell: export LD_LIBRARY_PATH = $ORACLELIB

On HP-11

C shell: setenv SHLIB_PATH ORACLELIB
Korn shell: export SHLIB_PATH= $ORACLELIB

Sybase

During the conversion, ServiceCenter places an extraordinarily high insert transaction load on your RDBMS server. To prevent the transaction log from growing disproportionately large, set the **Truncate Log on Checkpoint** option for the target database on your RDBMS server.

Setting up Time Zones for RDBMS Reporting

If you plan to report on ServiceCenter data using RDBMS tools, the sqltz: parameter should be set in the initialization file (sc.ini or PARMS) before conversion.

For information about using the sqltz parameter, see *Appendix B, Initialization Parameters for RDBMS*.

To change this parameter after conversion, you will need to convert all SQL converted tables back to P4 (see *Converting RDBMS Files Back to the P4 Format* on page 243), change the settings, and then reconvert them to your RDBMS. (See *Conversion to an RDBMS* on page 161.)

Important: If you use different time zone settings than the ones set before conversion, the dates in the reports made by your RDBMS utility may be off.

Windows Dynamic RDBMS Conversion DLLs

ServiceCenter supplies one *scserver*, *scautod*, and one *scenter* module that support all third party relational databases and P4. A DLL file is included with the ServiceCenter installation, which provides the ability to connect with specific RDBMS vendors. Database DLLs are dynamically loaded at the time of conversion, based on the RDBMS specified. No changes need to be made to your client installations.

Note: The ServiceCenter RDBMS conversion DLL replaces the need to use the **SCSwitch** application.

Unconverted Files

If the Direct SQL Interface is used before an SQL file is accessed, the user may need to add a new parameter (sqldirect) to the end of the sc.ini file for the server. In this case, the server does not know where to direct the SQL request. The sqldirect parameter must be used to tell the system which SQL provider to call. For example, sqldirect:oracle, indicates that the Oracle interface should be used.

The value to be used with sqldirect should be the one selected from the Target SQL Database Type dropdown list on the P4 to SQL Conversion Console. (See Figure 3-6 on page 82.)

Changing Unix Binaries

ServiceCenter supplies different sets of server applications for each commercial RDBMS to which you want to connect. Each set of unique binaries is optimized for a particular RDBMS vendor. In order to connect to a specific RDBMS vendor, you must configure ServiceCenter to use the appropriate build of the ServiceCenter binaries. No changes need to be made to your client installations.

Your ServiceCenter RUN directory contains multiple copies of the scenter file. These copies might be named:

```
Scenter
Scenter.sybase
Scenter.oracle
...
```

To enable a particular RDBMS, you must rename scenter.<rdbms> to scenter.

For example:

Issue the following commands from your ServiceCenter RUN directory to enable Sybase connectivity:

```
mv scenter scenter.p4
mv scenter.sybase scenter
```

OS/390 Configuration

ServiceCenter stores its data in flat files until you decide to move the structures and data into DB2. These flat files were created at the time of initial installation and may be displayed with ISPF option 3.4.

The file names follow a simple format of high level qualifier (chosen during installation), application version, and then file name as seen in the following example:

```
 DSLIST - Data Sets Matching SC51.SC51.SC51          Row 13 of 22
 Command ===>                                     Scroll ===> CSR
 Command - Enter "/" to select actionMessage        Volume
------------------------------------------------------------------------
 SC51.SC51.SC51.SCDB.ASC                            PRODQ0
 SC51.SC51.SC51.SCDB.DB1                            PRODQ0
 SC51.SC51.SC51.SCDB.DB2                            PRODQ0
 SC51.SC51.SC51.SCDB.DB3                            PRODQ0
 SC51.SC51.SC51.SCDB.DB4                            PRODQ0
 SC51.SC51.SC51.SCDB.DB5                            PRODQ0
 SC51.SC51.SC51.SCDB.DB6                            PRODQ0
 SC51.SC51.SC51.SCDB.DB7                            PRODQ0
 SC51.SC51.SC51.SCDB.FRE                            PRODQ0
 SC51.SC51.SC51.SCDB.LFD                            PRODQ0
 ********************** End of Data Set list **********************
```

These datasets contain logical files. The ServiceCenter product uses over 100 files stored in the flat files.

Modifying the ServiceCenter.x Files

Modification procedure:

1. Identify the DB2 subsystem into which you wish to install your ServiceCenter files. You will need to know the DSN of this subsystem.
2. Identify the user ID ServiceCenter uses to connect to that subsystem. Peregrine recommends using the same user found in the ServiceCenter jobcard. This user may need RACF authority to connect to DB2.
3. Modify your PARMS member contained in the SAMPLIB PDS created during ServiceCenter installation.

Insert the following three case sensitive lines into the PARMS member:

- sqldb - the DB2 name identified in step 1
- sqllogin - the user identified in step 2
- sqlplan - the plan for connecting to DB2/OS/390 (optional)

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
  ------------------------------------------------------------------------
  EDIT  SC51.SC51.V5R1M0.SAMPLIB(PARMS) - 01.03            Columns 00001 00072
  Command ===>                                            Scroll ===> CSR
  000045 #
  000046 #  mvstcp specifies the TCP/IP port name to be used for MVS server.
  000047 #mvstcp:t3271
  000048 #mvstcpexpress:t3272
  000049 mvstcp:3056
  000050 mvstcpexpress:3057
  000051 scauto:3058
  000052 debugscauto:999
  000053 #
  000054 #  mvstcp_prefix specifies the dataset prefix of the TCP/IP system
  000055 #  datasets. TCPIP is the default value for this parameter.
  000056 #mvstcp_prefix:TCPIP
  000057 #
  000058 #  mvstcp_addrspc specifies the name of the TCP/IP address space.
  000059 #  TCPIP is the default value for this parameter.
  000060 #mvstcp_addrspc:TCPIP
  000061 #
  000062 triggers:1
  000063 #
  000064 parentvars:1
  000065 sqllogin:EJC1
  000066 sqldb:db41
  000067 sqlplan:SCPLAN1
  000068 sqllimit:600
  ****** ************************ Bottom of Data ***************************
```

For more information on these parameters, see *Parameters* on page 298.

4 Concatenate the DB2 **dsnload** and **dsnexit** libs into the STEPLIB of the ServiceCenter job contained in the SAMPLIB PDS.

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
//*************************************************************
EDIT  SC51.SC51.V5R1M0.SAMPLIB(SC) - 01.01Columns    00001 00072
Command ===>                              Scroll ===> CSR
000028 //*********************************************************************
000029 //*
000030 //SC        PROC PREFIX='QATEST2.SC51',
000031 //          PARMMEM=PARMS,
000032 //          LOADLEV=V5R1M0,
000033 //          RUNPARMS='P=SYEXER C=500000'
000034 //*
000035 //*********************************************************************
000036 //SC        EXEC PGM=KERNEL,
000037 //          PARM='&RUNPARMS'
000038 //STEPLIB   DD  DSN=&PREFIX..&LOADLEV..LOAD,DISP=SHR
000039 //*         DD  DSN=DSN230.DSNEXIT,DISP=SHR
000040 //*         DD  DSN=DSN230.DSNLOAD,DISP=SHR
000041 //PARMS     DD  DSN=&PREFIX..&LOADLEV..SAMPLIB(&PARMMEM),DISP=SHR
000042 //SYSPRINT  DD  SYSOUT=*                      STANDARD OUTPUT
000043 //SYSTERM   DD  SYSOUT=*                      STANDARD ERROR
000044 //SYSUDUMP  DD  SYSOUT=*                      SYSTEM DUMP
000045 //SNAP      DD  SYSOUT=*                      SERVICE CENTER SNAP DUMP
000046 //          PEND
000047 //*********************************************************************
```

Preparing the DB2 Subsystem

This phase requires the expertise of a DB2 DBA.

To prepare the DB2 subsystem:

1. Create a DB2 STOGROUP called **PRGNSTO**. The name can be changed after the installation is complete if required by the DBA staff. The STORGROUP will contain volumes where the DB2 datasets are allocated.
2. Create a DB2 DATABASE called **PRGNDB** using the following command.

   CREATE DATABASE PRGNDB BUFFERPOOL BP32K STOGROUP PRGNSTO;

   ServiceCenter uses the 32K buffer pool for all its data. The default 32K buffer pool size (24 buffers) is too small. The DBA should monitor this pool while ServiceCenter is running to make sure the pool is effectively utilized. Peregrine uses a size of 2000 buffers during its testing. This is a significant increase over what most sites currently have in the 32K buffer pool.

   The current size of the buffer pool can be obtained using the command **DB2 DISPLAY BUFFERPOOL (BP32K)**, then observing the value for the Virtual BufferPool Size. This size of the buffer pool can be changed at any time with the **DB2 Alter BufferPool(BP32K) VPsize(nnnn)** command. These commands are documented in the *IBM DB2 for OS/390* manuals.

3. Grant the appropriate authorities to the userid for the PRGNDB database:
   a. DBADM is customary in a test/acceptance DB2 region. This is a matter of DBA policy.
   b. The user will need CREATE authority in the BP32K bufferpool where the install process creates indexes.
4. Verify that the user has appropriate authorizations by using the IBM utility **SPUFI** to create a test table in the PRGNDB database. (Please refer to the IBM DB2 documentation for instructions.)
5. Ensure that the bufferpool used in Step #2 has adequate resources for the transaction load. Typically, the DBA will be responsible for this resource.
6. Ensure that the DB2 EDMPOOL can accommodate the DBD created by the installation process. In a test system at Peregrine Systems, the PRGNDB had a length of 940844.
7. BIND the SCPLAN1 PLAN. Use the SQDB2MVS member in the DBRM PDS as the member.

```
                         BIND PLAN                          SSID:DB41
 Command ===>

 Enter DBRM data set name(s):
 1   MEMBER .......... ===>SQDB2MVS
 2   PASSWORD......... ===>
 3   LIBRARY.......... ===>'EJC2.V5R1M0.DBRM
 4   ADDITIONAL DBRMS? ===>NO             (YES to include more DBRMS)

 Enter options as desired:
 5   PLAN NAME ..................... ===>SCOLAN1(Required to create a plan)
 6   CHANGE CURRENT DEFAULTS ...... ===>(NO or YES)
 7   ENABLE/DISABLE CONNECTIONS? .. ===>(NO or YES)
 8   INCLUDE PACKAGE LIST? ........ ===>(NO or YES)
 9   OWNER OF PLAN (AUTHID) ....... ===>(Leave blank for your primaryID)
 10  QUALIFIER ..................... ===>(For tables views and aliases)
 11  CACHESIZE ..................... ===>(Blank, or value 0-4096)
 12  ACTION ON PLAN ................ ===>(REPLACE or ADD)
 13  RETAIH EXECUTION AUTHORITY ... ===>(YES to retain user list)
 14  CURRENT SERVER ................ ===>(Location name)

PRESS: ENTER to processEND to save and exitHELP for more information
```

8. Grant EXECUTE authority on the PLAN to PUBLIC or to the user specified in Step 2.

Enabling Connectivity

In order to connect ServiceCenter's application server to your RDBMS, the following information is necessary:

- The name of the RDBMS to which it is to connect.
- Which login and password to use in order to connect to your SQL server. This login and password should be those you created in the section on generating a login ID for ServiceCenter (See *Login ID* on page 99).

Add the following two lines to the sc.ini file in your ServiceCenter RUN directory:

sqldb:<*dbname*>

sqllogin:<login>/<*password*>

Only ServiceCenter servers connect to the SQL database; ServiceCenter clients do not connect directly. Consequently, the parameters described in this section are applicable only to the ServiceCenter server, and do not need to be specified on the ServiceCenter clients' sc.ini files.

Example

To connect to a database named *datatrax1* with a login of *sc_login* and a password of *topsecret*, add the following lines to your sc.ini file:

sqldb:datatrax1

sqllogin:sc_login/topsecret

Important: After you make this change, cycle your ServiceCenter application server.

Microsoft SQL Server

ServiceCenter uses native Microsoft SQL server db–lib calls to manage its connectivity to your SQL server. In order for your application server to connect to your SQL server, they both must be members of the same *domain*, or a trust relationship must exist between your domain and the SQL server's domain.

The general connectivity rules are:

- The database name entered in your *sc.ini* file (above) must correspond to the SQL server's SERVER name as broadcast to the domain.
- The ServiceCenter application server must have the SQL client/server libraries installed on it.
- The SQL client/server libraries must be in the path of the service that launches ServiceCenter.

Example

The SQL client/server libraries are normally installed to:

<header>\MSSQL\binn

— Or —

<header>\MSSQL\bin

Sybase

Only ServiceCenter *servers* connect to Sybase; ServiceCenter clients do not directly connect to Sybase. Consequently, the parameters described below are applicable only to the ServiceCenter server, and do not need to be specified on the ServiceCenter clients.

Parameter	Definition
sqldb	The sqldb parameter in the ServiceCenter sc.ini file specifies the name of a Sybase SQL server. The connection name is defined in the *sql.ini* file on Windows Platforms, and in the interfaces file on Unix platforms. This file is specified in the Sybase program and is not an element of ServiceCenter. Server Parameters are also set in the sc.ini file. (See *Sample connections* on page 109.) The Unix interfaces file is located via\ the SYBASE environmental variable. (See *Sybase interfaces file entry specifying the scsyb server description:* on page 110.) The Windows *sql.ini* file must be in the path of the UID that will launch the ServiceCenter application server. (See *Sybase sql.ini file entry specifying the scsyb server description* on page 110.) For detailed information about the *sql.ini* or interfaces file see the *Sybase System Administration Guide Supplement* for your platform.
sqllogin	The sqllogin parameter specifies the Sybase userid and password used to connect to the database.

In the sample files below, ServiceCenter connects to the Sybase instance named **scsyb**. The **scsyb** Sybase SQL server is defined in the interfaces file for Unix, and in sql.ini for Windows.

Sample connections

The sc.ini file is the ServiceCenter configuration file that must be present on the ServiceCenter server RUN directory. The sql.ini file is a Sybase configuration file for Window Platforms. The interfaces file is a Sybase configuration file for Unix platforms.

Sc.ini file with sqllogin and sqldb parameters:

```
# database connects to scsyb
sqldb:scsyb
sqllogin:ed/ed
system:7111
auth:xxxxxxxx F2B3FBA0 48D7A4CD 23AE831B
```

Sybase interfaces file entry specifying the scsyb server description:

This entry in the interfaces file describes a connection name, scsyb, to a Sybase server using the TCP protocol. The connection's host name is bear, the listening port number (for the Sybase server) is 1521.

```
scsyb 5 5
    query tcp ether bear 1521
    master tcp ether bear 1521
```

The interfaces file is usually located with the SYBASE environmental variable. An example of setting the environmental variable using the *C* shell would be:

```
setenv SYBASE /dba/admin
```

With the Bourne or Korn shell you would enter:

```
SYBASE=/dba/admin
export SYBASE
```

If the SYBASE variable has not been set, Sybase looks in the home directory for an interfaces file.

Sybase sql.ini file entry specifying the scsyb server description

This entry in the sql.ini file describes a connection name, scsyb, to a Sybase server using the TCP protocol. The connection's host name is bear, the listening port number (for the Sybase server) is 1521.

```
scsyb 5 5
    query tcp ether bear 1521
    master tcp ether bear 1521
```

ServiceCenter uses native Sybase libraries to connect to Sybase. These libraries require the presence of a file named **sql.ini** to establish a relationship between a logical database server name (e.g., **scscb**) and a physical IP address and port number. The **sql.ini** file must be present on the application server and must be in the path of the UID that will launch the ServiceCenter application server.

Oracle

Oracle support can only be achieved after a complete ServiceCenter installation. Complete the following procedures prior to converting your database to Oracle support:

- Modify the ServiceCenter **sc.ini** file.
- Set the Oracle environmental variable.
- Prepare the Oracle instance for ServiceCenter.
- Log in with an express connection.

Note: An Oracle DBA may be required to assist with these preparations.

SQL*Net

Since ServiceCenter uses SQL*Net to access the Oracle data, the Oracle instance may be on the same or a different computer than the ServiceCenter application server, and may use a variety of protocols. The ServiceCenter application server binary code contains all the necessary code to access Oracle via SQL*Net.

Only ServiceCenter servers connect to Oracle, ServiceCenter clients do not directly connect to Oracle. Consequently, the parameters described below are applicable only to the ServiceCenter server, and do not need to be specified on the ServiceCenter clients.

Server Parameters

Parameter	Definition
sqldb	The **sqldb** parameter in the ServiceCenter sc.ini file specifies the name of an Oracle database connection. The connection name is defined in the tnsnames.ora file. Server Parameters are also set in the sc.ini file. (See *Sample connections* on page 112.) On Unix platforms, the tnsnames.ora file is located using the TNS_ADMIN environmental variable. (See *Setting the environmental variable on Unix platforms* on page 113.) On Windows platforms, the tnsname.ora file is located in the Oracle Home directory. (See *Setting the environmental variable on Windows platforms* on page 114.)
sqllogin	The **sqllogin** parameter specifies the Oracle userid and password used to connect to the database.

In the sample files below, ServiceCenter connects to the Oracle instance named **scora.world**. The **scora.world** parameter is defined in the tnsnames.ora file (a description of which can be found in the *Oracle SQL*Net* documentation).

Note: Only **scora** must be specified in the ServiceCenter *sc.ini* file to connect to **scora.world**. The **world** domain is defined in the Oracle sqlnet.ora configuration file.

Sample connections

The sc.ini file is the ServiceCenter configuration file that must be present in the ServiceCenter server RUN directory. The tnsnames.ora and sqlnet.ora files are Oracle configuration files.

ServiceCenter *sc.ini* **file with sqllogin and sqldb parameters:**

```
# database connects to scora.world (world appended automatically by
Oracle)
sqldb:scora
sqllogin:ed/ed
system:7111
auth:xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
```

Note: Define a default table space for the conversion to land in. If you do not, the conversion will land in the system table space, which may result in corrupt data.

Oracle *tnsnames.ora* file entry specifying the *scora* connect descriptor:

This entry in the tnsnames.ora file describes a connection name **scora** to an Oracle server using the TCP protocol. The connection's host name is **bear**, the listening port number (for the Oracle server) is **1521**, and the Oracle System ID (SID) is **bear1**.

```
scora.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (PROTOCOL = TCP)
          (Host = bear)
          (Port = 1521)
        )
    )
    (CONNECT_DATA =
       (SID = bear1)
    )
  )
Oracle sqlnet.ora file - defining the world domain:
##################
# Filename......: sqlnet.ora
# Name..........: manatee_tcpcom.world
# Date..........: 26-FEB-96 18:35:15
##################
AUTOMATIC_IPC = ON
TRACE_LEVEL_CLIENT = OFF
#TRACE_LEVEL_CLIENT = ADMIN
SQLNET.EXPIRE_TIME = 0
NAMES.DEFAULT_DOMAIN = world
NAME.DEFAULT_ZONE = world
```

Setting the environmental variable on Unix platforms

The tnsnames.ora file is usually located with the TNS_ADMIN environmental variable.

Example of setting the environmental variable using the C shell:

```
setenv TNS_ADMIN /dba/admin
```

Example of setting the environmental variable using the Bourne or Korn shell:

```
TNS_ADMIN=/dba/admin
export TNS_ADMIN
```

If the TNS_ADMIN variable has not been set, ORACLE looks first in the /etc directory. If that fails, it then looks in the $ORACLE_HOME/network/admin directory.

Setting the environmental variable on Windows platforms

The tnsnames.ora file is located in the ORACLE_HOME/network/admin directory where ORACLE_HOME gets established when the Oracle products are installed on NT. To find the current Oracle_Home directory on a computer, use the regedit program and look for the key ORALCE_HOME in the registry folder "HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE".

Oracle Call Interface

The Oracle Call Interface (OCI) enables ServiceCenter to map data files to multiple Oracle databases, with some files becoming tables in one database, and others becoming tables in a separate Oracle instance. Before this can be accomplished multiple new database types must be created in ServiceCenter (one for each additional database), and these types must be associated with the corresponding Oracle instances in the *sc.ini* file.

Important: The current OCI module is supported with ORACLE 8.x only, and does not work with ORACLE7.x or lower. OCI related executables use names suffixed with *.oraoci*. For example, *scenter.oraoci* and *scserver.oraoci* have been created to be distinguished from *scenter.oracle* and *scserver.oracle*.

In Windows, ServiceCenter detects the version of the Oracle database being used and loads the appropriate Oracle dll. The version of a database is searched before the dll is loaded. The searching orders are first oracle8i, then oracle8.0.x, oracle7.3 and finally oracle7.2. In this way, scor8cli.dll or oradll73, or oradll72.dll is loaded if oracle8.x (or oracle7.x) is found.

ServiceCenter 3 SP3 and later use oradll73.dll, oradll72.dll and scor8cli.dll, ServiceCenter 3 SP2 and earlier versions of ServiceCenter use oradll73.dll, oradll72.dll and oradll80.dll.

The following example demonstrates the process:

Assume an administrator has two databases `datab1` (with connection parameters of uid *prgn1* password *prgn1*) and `datab2` (with connection parameters of uid *prgn2* password *prgn2*).

Two database types are created using the ServiceCenter SQL utility for datab1 and datab2, e.g., `oradb1` and `oradb2`. (All Oracle database types are prefixed with **ora** to distinguish Oracle from other databases.)

Two sections are then added to the `sc.ini` as follows:

```
[oradb1]
sqldb:prgn1
sqllogin:prgn1
[oradb2]
sqldb:prgn2
sqllogin:prgn2
```

System Requirements

- ServiceCenter running on a supported server.
- Multiple databases on the same server. For example, you can map to SQL server and Oracle on the same server, but not to different versions of the same RDBMS (Oracle 7.2 and Oracle 8.)

  — Or —

  Separate servers for two instance of the same RDBMS. For example, you can map data to an Oracle 8.0 database on one server and map a different file from the same ServiceCenter instance to another Oracle 8.0 database on another server.

Creating New ServiceCenter Data Types

New database type record must be created to begin the process. These type records will be used by the system when referring to the separate Oracle databases.

The database type must match the name of a **SQL DB Type** field in the sqldbinfo file.

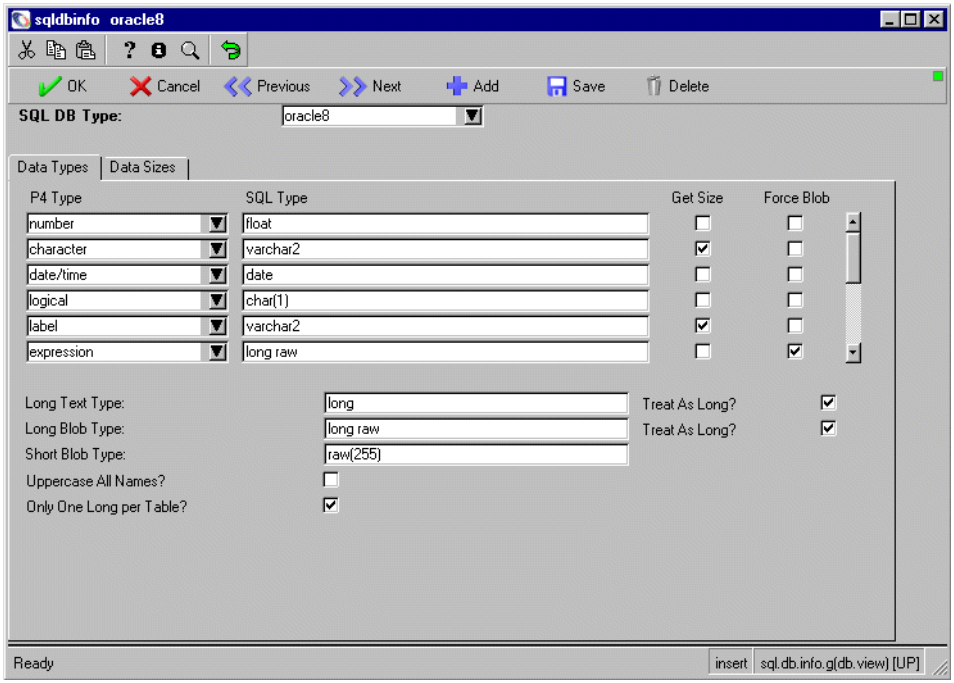


Figure 4-1: SQL Database Information Form

This same database type must also appear in the conversion utility prior to converting a P4 file system to an RDBMS.

Sample

```
[oracle8]
sqldb:scora8
sqllogin:scuser/scpass
[db2universal]
sqldb:TESTDB2
sqllogin:scuser/scpass
```

To create database types:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 From the **SQL Utilities** menu, click **SQL DB Types Utility**.

3 At the prompt screen, enter the name of the new database type.

The name must start with *ora*.

4 On the **Based on this SQL DB type** line, select the SQL database type upon which this database is to be modeled, from the drop-down list.

For this example *oracle8* is used.

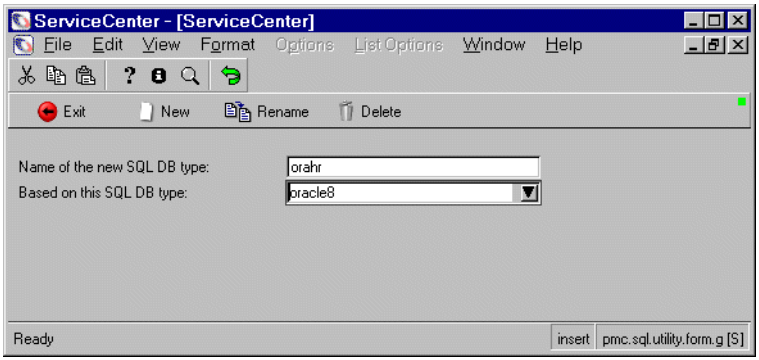


Figure 4-2: Creating a New Database Type

5 Click **New** to build the new database type.

A prompt is displayed, telling you that the new type was added.

6 Click **OK** to proceed.

7 Repeat these steps to create two types, one for each Oracle database.

Associating Database Types with Oracle Instances

Once created, the database types need to be associated with the actual Oracle databases. This is done through Initialization File Parameters.

Define a section in your ServiceCenter server **sc.ini** file for each RDBMS to which you want to connect, adding 3 lines for each database. The first line contains the new data type, in square brackets [ ]. The next two lines are the same sqldb (Oracle SID) and sqllogin (Oracle account and password) parameters ordinarily used to connect to an Oracle database.

This is an example of the Syntax:

```
[<database type>]
sqldb:<database name>
sqllogin:<user name/password>
```

An actual example might look something like this:

```
[orahr]
sqldb:HR
sqllogin:servicecenter/scscsc
[oramydb]
sqldb:MYDB
sqllogin:sc/secret
```

After editing the **sc.ini** file, restart ServiceCenter so that the initialization can take place.

Mapping Files to Multiple Databases

To map individual files to a particular database, see *Single File Conversion* on page 176.

Informix Preparation

To ensure ServiceCenter works properly with Informix Dynamic server 7.3 for Windows, you must start the ServiceCenter Service as the Informix database Administrator. The Informix install program should have created an Informix user.

Before converting P4 to Informix Windows NT/2000 example:

1. Access **Services** from Windows Control Panel.
2. Select the ServiceCenter service.
3. Click **Startup**.

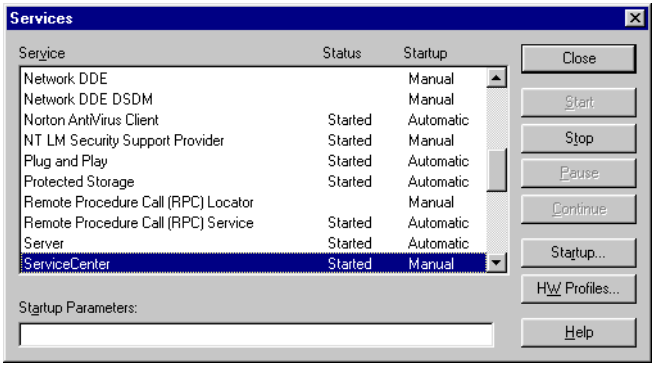


Figure 4-3: Windows Services Dialog Window

4 Ensure that the ServiceCenter Service is configured to start with the Informix user and password as defined in the Windows User Manager, and as defined in your sc.ini.

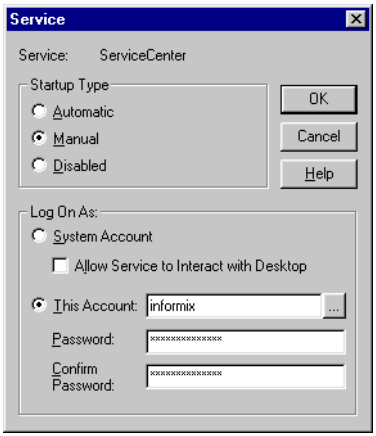


Figure 4-4: Specifying a Service Account Database

Sample sc.ini:

```
path:C:\sc5.1\DATA
shared_memory:48000000
log:C:\sc5.1\sc.log
bitmap_path:C:\sc5.1\BITMAPS
system:12670
auth:NNNNNNNN NNNNNNNN NNNNNNNN NNNNNNNN
scauto:12690
## Informix RDBMS Details  ##
sqldb:sc5.1
sqllogin:informix/password (This is the same as the ServiceCenter
startup parameters)
sqldrop:1
```

The sqldb: parameter is the name of the database, not the Informix server.

Checkpoints

1 Ensure that you have your environment variables set, and are able to connect to the Database for the ServiceCenter conversion.

2 Informix requires environment variables in addition to the sc.ini settings.

- INFORIMIXDIR — the Informix base directory.
- INFORMIXSERVER — the Informix server name.
- INFORMIXSQLHOSTS — the computer name on the network, fully qualified name might be needed. (Windows only)
- INFORMIXSHMBASE — should be set to *163840* if the full P4 file system is going to be converted. (Unix only)

Windows example:

To set the environment variables in Windows, use the Informix Setnet32 utility and ensure the following are set properly:

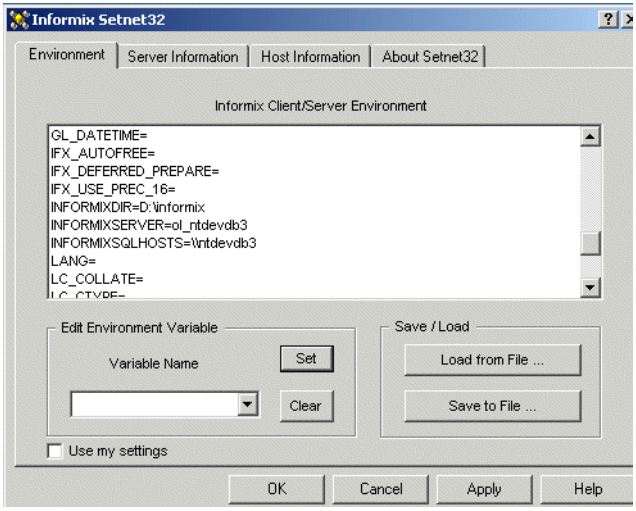


Figure 4-5: Verifying Informix Checkpoints

C shell example:

```
setenv InformixDir /opt/informix
setenv InformixServer db1_online
setenv InformixShmbase 163840
```

3 Perform a standard conversion, as is detailed in *Conversion to an RDBMS* on page 161

Microsoft SQL Server Preparation

By default, ServiceCenter uses an internal file system (P4) in which to store data, such as Incident tickets, changes, and service requests. This chapter provides details on the implementation of the Microsoft SQL server, versions 6.5, 7.0 and 2000, as the database platform instead of P4. It builds from the premise that ServiceCenter and Microsoft SQL server have already been installed. If the SQL server has not yet been installed, specify the correct case sensitivity for sort order during the setup phase.

This section provides details on the implementation of the following versions of Microsoft SQL server:

- *Microsoft SQL Server Version 6.5* on page 122
- *Microsoft SQL Server Version 7.0* on page 133
- *Microsoft SQL Server Version 2000* on page 140

Case Sensitivity

ServiceCenter no longer requires that any third-party RDBMS be setup to use case-sensitivity for searching and sorting. However, the default installation of P4 is case-sensitive and the default installation of Microsoft SQL server is case-insensitive. Therefore, a default Microsoft SQL server installation cannot be successfully used for the ServiceCenter conversion, unless you first set up the P4 file system for case-insensitive searching.

You may either set up the ServiceCenter's P4 file system to be case-insensitive and keep MS SQL case-insensitive, or set up Microsoft SQL server to be case sensitive and keep P4 case sensitive; both must have the same sensitivity.

For more information on case sensitivity in ServiceCenter, see Volume 2 of Database Management and Administration.

Timestamps in ServiceCenter

When Microsoft SQL Server is installed, two datatypes are available, sysname and timestamp.

To insert or update a row in a table, simply insert or update the row. All updating of the timestamp field is automatic, so you don't have to indicate a timestamp value.

To use timestamp in a specified table, add a column to the table, as timestamp, using Microsoft SQL Server Enterprise Manager.

The new column, defined as timestamp, is maintained automatically by Microsoft SQL Server when the table is modified. ServiceCenter does not change the timestamp field when it inserts or updates a row in a table. All updating of the timestamp field is automatically handled by Microsoft SQL Server. Since timestamp is defined as SQLVARBINARY, one of the datatypes that ServiceCenter specifies, it will be retrieved by ServiceCenter correctly.

Microsoft SQL Server Version 6.5

The following procedures convert ServiceCenter from the P4 file system to Microsoft SQL server 6.5.

To set up both to be case-insensitive:

1. Convert P4 so that it will do case-insensitive searching. (For instructions, see Volume 2 of Database Management and Administration.)
2. Confirm that your installation of MS SQL is case insensitive (default installation). See *To ensure that SQL Server 6.5 is set up to use the correct sorting:* on page 123 for instructions.
3. Change MS SQL to case insensitive searching if necessary. See *To change the case sensitivity:* on page 124 for instructions.
4. Create an SQL server database device. See *Creating an SQL Server Database* on page 127 for instructions.
5. Set up a user ID on the SQL server database. See *Setting Up a User Id on the SQL Server Database* on page 132 for instructions.
6. Set up ServiceCenter and perform the conversion. See *On the ServiceCenter Side* on page 132 for instructions.

To set up both to be case sensitive:

1. Convert P4 so that it will do case-sensitive searching. This step will not be necessary unless you have previously converted P4 to be case-insensitive. (For instructions, see Volume 2 of Database Management and Administration.)
2. Confirm that your installation of MS SQL is case-sensitive (default installation). See *To ensure that SQL Server 6.5 is set up to use the correct sorting:* on page 123 for instructions.
3. Change MS SQL to case-sensitive searching if necessary. See *To change the case sensitivity:* on page 124 for instructions.
4. Create an SQL server database device. See *Creating an SQL Server Database* on page 127 for instructions.
5. Set up a user ID on the SQL server database. See *Setting Up a User Id on the SQL Server Database* on page 132 for instructions.
6. Set up ServiceCenter and perform the conversion. See *On the ServiceCenter Side* on page 132 for instructions.

To ensure that SQL Server 6.5 is set up to use the correct sorting:

1. Use the SQL utility and select the `sp_helpsort` query to verify whether or not the SQL server database has been configured to use case-sensitive sorting.

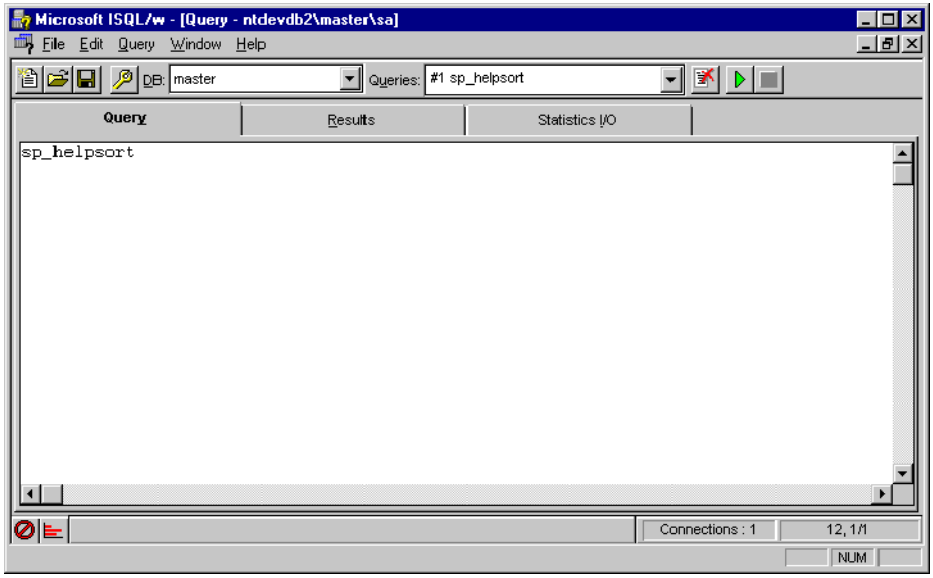


Figure 4-6: Verifying Case-Sensitive Sorting on SQL Server

2 Check the results.

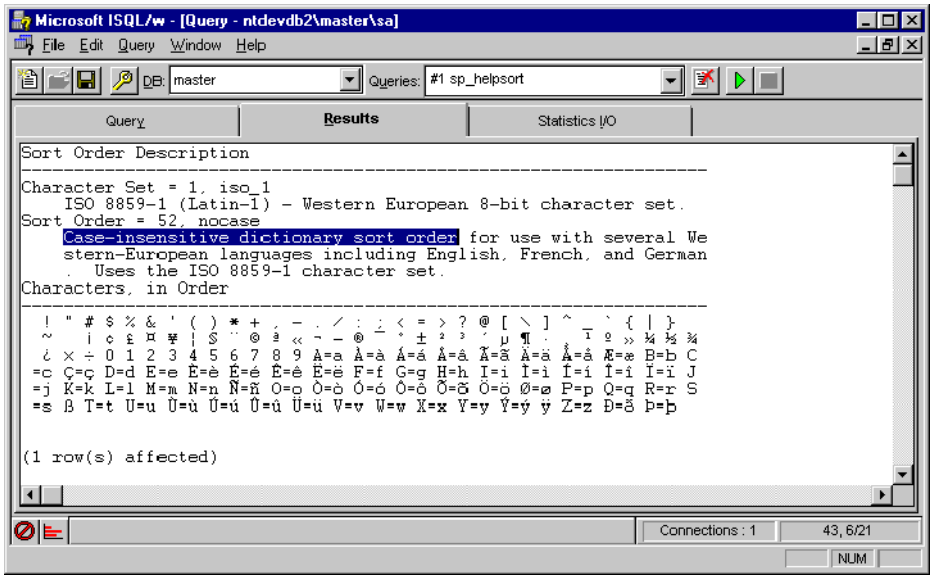


Figure 4-7: Checking Results of Case-Sensitive Sorting on SQL Server

3 If the database is not set up the way you want it to be, run the MS SQL setup and change the master database character set.

Note: The setup program is usually found from the **Start** menu under **Programs > Microsoft SQL server**.

To change the case sensitivity:

1 Select the setup icon.

The following screen is displayed:

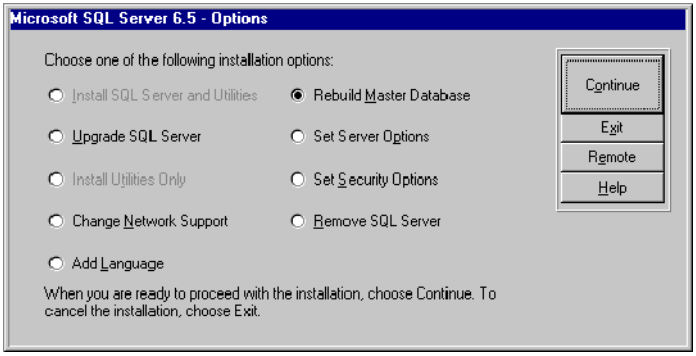


Figure 4-8: Changing Case Sensitivity on SQL Server

2 Select the Rebuild Master Database option.

3 Click **Continue**.

The following confirmation screen is displayed.

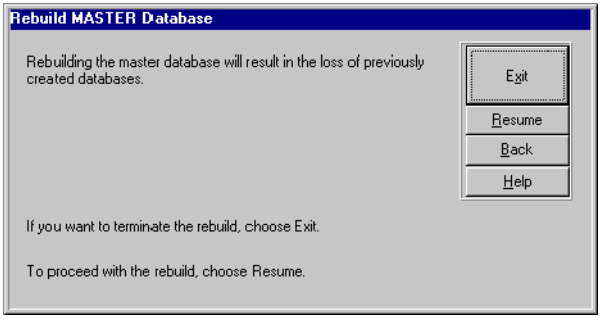


Figure 4-9: Rebuild the Master Database Option on SQL Server

Warning: All previously created databases will be lost if they are not backed up and stored on another site. This option will rewrite the master database character set.

4 Click **Resume** when proper measures have been put into place to preserve any existing databases.

5 Select the **Sort Order** options from the Rebuild Options dialog box.

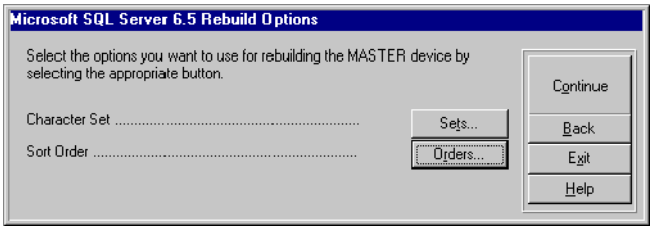


Figure 4-10: Selecting the Sort Order on SQL Server

6 Select the specific sort order desired.

- **Dictionary order, case-sensitive**

  — Or —

- **Dictionary order, case-insensitive**

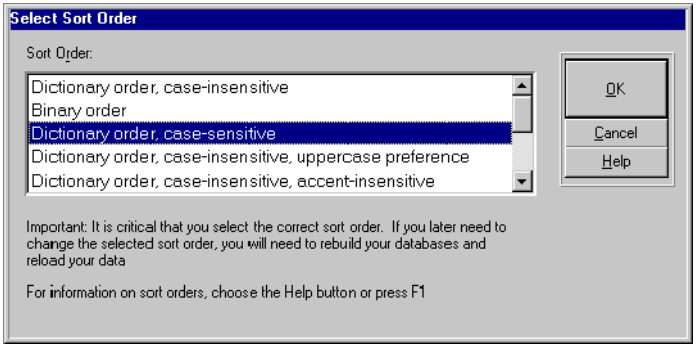


Figure 4-11: Specifying the Sort Order on SQL Server

7 Click **Continue** to rebuild the master database, using the sort methodology.

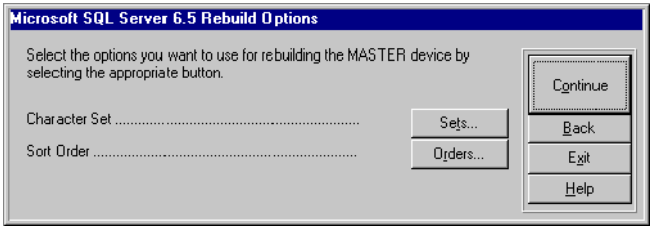


Figure 4-12: Rebuilding the Master Database on SQL Server

A prompt is displayed, asking for the location of the SQL server installation.

8 Provide the appropriate drive specification.

9 Click **Continue**.

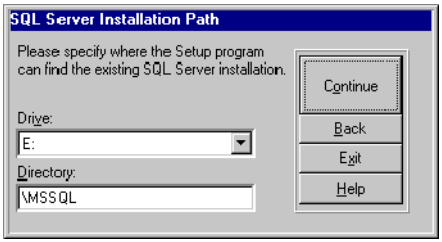


Figure 4-13: Specifying the Drive on SQL Server

A prompt is displayed, asking for the location of the Master device to be built, i.e., filename and drive on which to model the device.

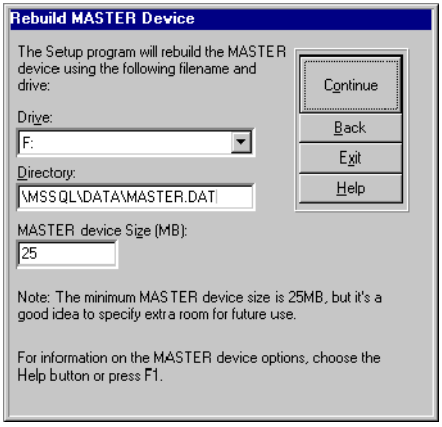


Figure 4-14: Specifying the Filename on SQL Server

10 Specify the device settings and memory allocation and click **Continue**.

The system begins the rebuild of the Master device. It may take some time to complete the construction.

A prompt may appear, displaying the status of the files updated if databases were resident on the drive prior to conversion.

11 Click **OK** to continue.

Creating an SQL Server Database

In order for ServiceCenter to interact with the SQL server, a device needs to be created for ServiceCenter to use.

To create a device:

1 At the SQL Server Enterprise Manager, click the right mouse button to reveal the shortcut menu.

2 Select the **Db Device fields of Selected Server** option.

— Or —

a Pull down the **Manage** menu.

b Select the **db devices** submenu.

c Select the **new device** option.

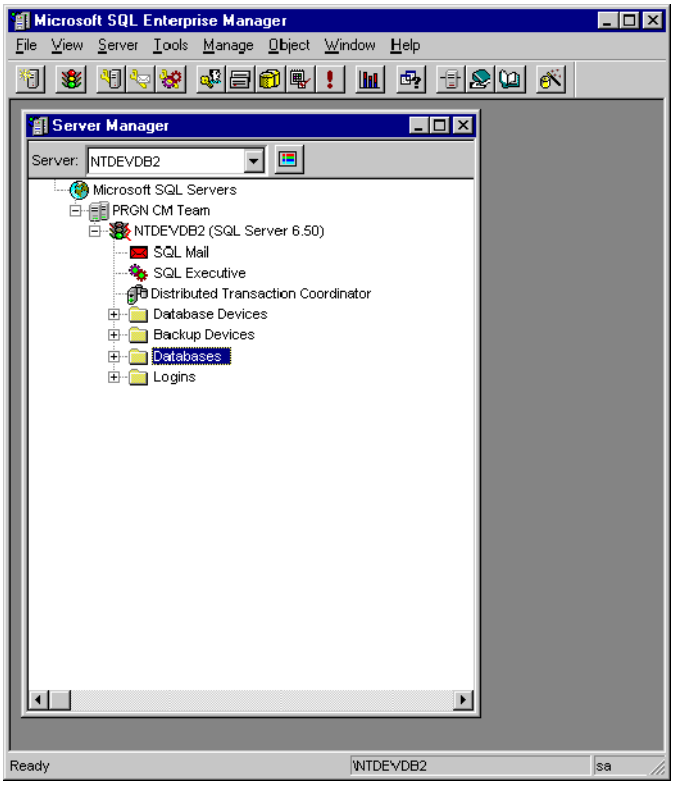


Figure 4-15: Selecting a Device on SQL Server

3 Provide a name for the new device.

4 Specify the location for the device.

When possible separate data on a drive away from other disk drives.

5 Set the size constraints for the device. This value should be 4 to 5 times the size of the P4 database.

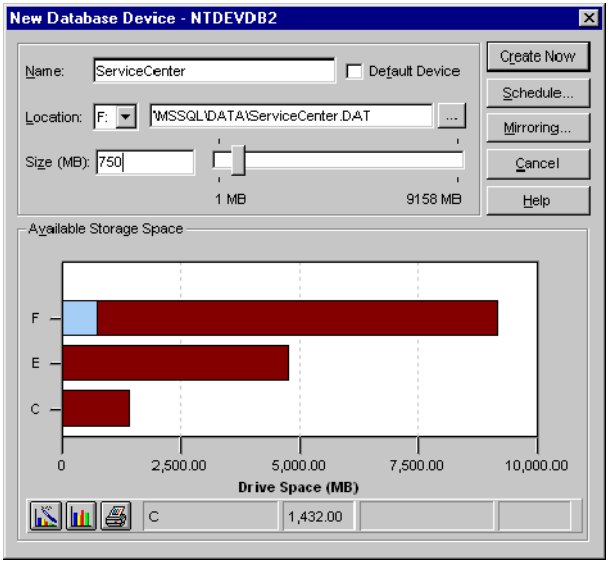


Figure 4-16: Setting the Size of a Device on SQL Server

6 Specify a **Data Device**.

7 Specify a **Log Device** if different from Data Device.

8 Click **Create Now** when ready to build the database record.

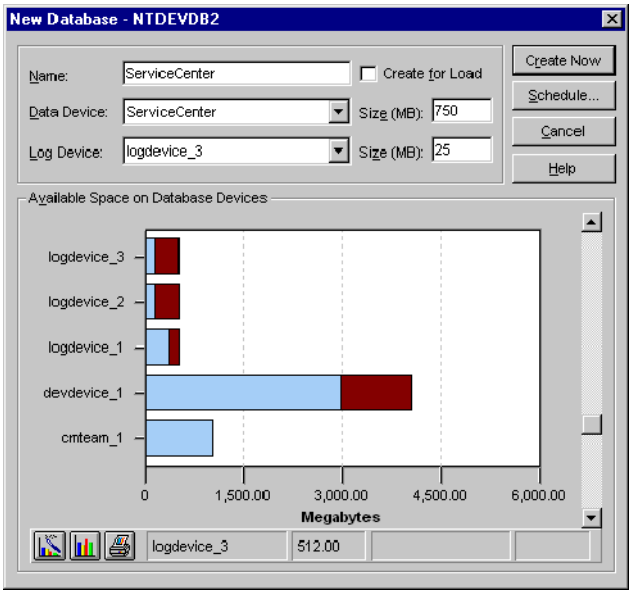


Figure 4-17: Creating a Device on SQL Server

9 Right-click on the database to bring up the shortcut menu.

10 Select **Edit** to display the database options.

11 Select the **Options** tab.

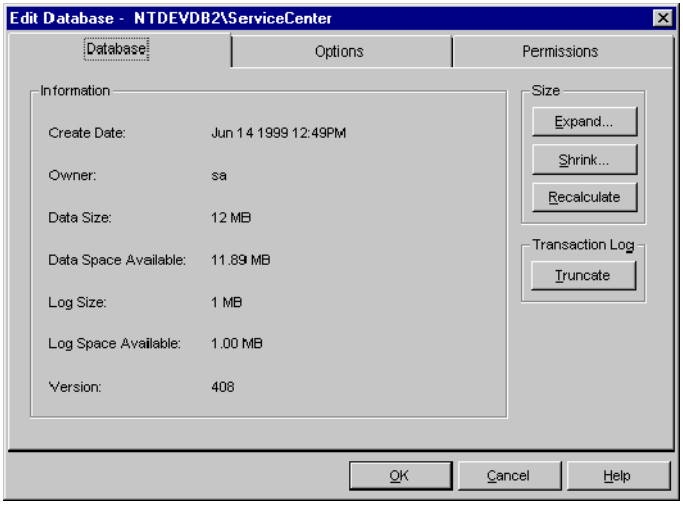


Figure 4-18: Selecting Device Options on SQL Server

12 Set up the database options appropriate for your intentions, including setting **Truncate Log on Checkpoint** to *true*.

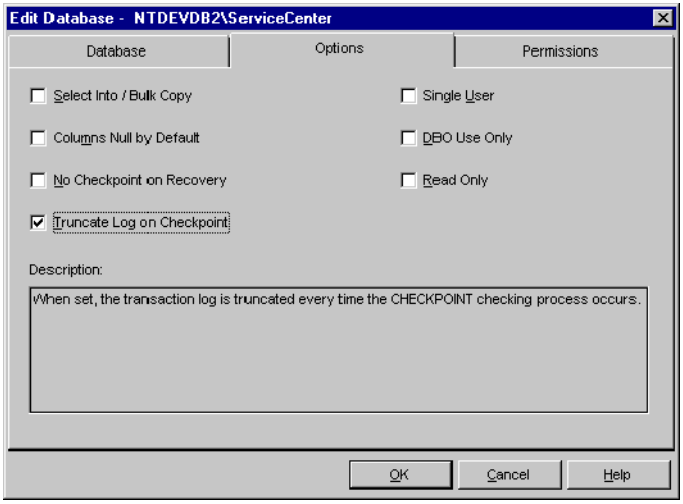


Figure 4-19: Setting Truncate Log on Checkpoint

Setting Up a User Id on the SQL Server Database

To set up a user ID:

1 Right-click on **Logins** in the SQL Server Enterprise Manager to display the shortcut menu.

2 Select **new logins** from the menu.

3 Create an SQL server user ID and login to be used by ServiceCenter.

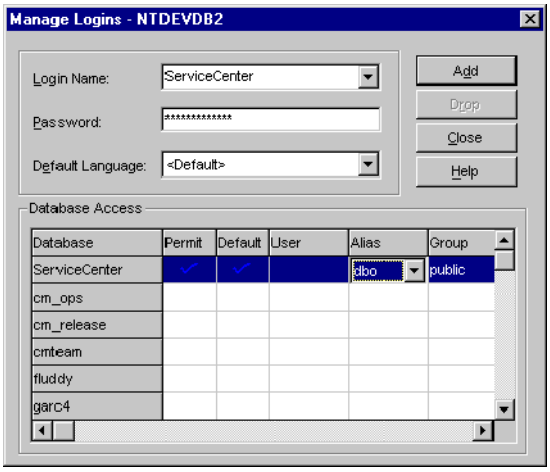


Figure 4-20: Creating a User ID on SQL Server

4 When configuring the login:

- Specify **Permit** and **Default** in the ServiceCenter database.
- Grant DBO to the user for that database.

On the ServiceCenter Side

To perform conversions, the client must connect to ServiceCenter in express mode (thin client). There can only be one user on ServiceCenter at the time of conversion to prevent file corruption. ServiceCenter employs a series of utilities to move data back and forth between the internal P4 system and the external RDBMS.

Actions on the ServiceCenter side:

1 Set the ServiceCenter server **sc.ini** file to connect to the SQL server database.

2 Ensure there is only one user on the ServiceCenter system.

3 Convert the ServiceCenter data to the SQL Server 6.5 database. This conversion is direction and platform dependent. See *Conversion to an RDBMS* on page 161 for further instructions.

Microsoft SQL Server Version 7.0

The following steps convert ServiceCenter from the P4 file system to Microsoft SQL Server 7.0. and 2000

To set up both to be case-insensitive:

1 Convert P4 so that it will do case-insensitive searching. (For instructions, see Volume 2 of Database Management and Administration.)

2 Confirm that your installation of MS SQL is case insensitive (default installation). See *To ensure that SQL Server 7.0 is set up to use the correct sorting:* on page 134 for instructions.

3 Change MS SQL to case insensitive searching if necessary. See *To change the case-sensitivity:* on page 134 for instructions.

4 Create an SQL server database device. See *Creating an SQL Server Database* on page 135 for instructions.

5 Set up a user ID on the SQL server database. See *Setting Up a User on the SQL Server* on page 138 for instructions.

6 Set up ServiceCenter and perform the conversion. See *On the ServiceCenter Side* on page 139 for instructions.

To set up both to be case sensitive:

1 Convert P4 so that it will do case-sensitive searching. This step will not be necessary unless you have previously converted P4 to be case-insensitive. (For instructions, see Volume 2 of Database Management and Administration.)

2 Confirm that your installation of MS SQL is case-sensitive (default installation). See *To ensure that SQL Server 7.0 is set up to use the correct sorting:* on page 134 for instructions.

3 Change MS SQL to case-sensitive searching if necessary. See *To change the case-sensitivity:* on page 134 for instructions.

4 Create an SQL server database device. See *Creating an SQL Server Database* on page 135 for instructions.

5 Set up a user ID on the SQL server database. See *Setting Up a User on the SQL Server* on page 138 for instructions.

6 Set up ServiceCenter and perform the conversion. See *On the ServiceCenter Side* on page 139 for instructions.

To ensure that SQL Server 7.0 is set up to use the correct sorting:

1 Use the Query Analyzer utility and select the **sp_helpsort** query to verify whether or not the SQL server database has been configured using case-sensitive sorting.

2 Check the results.

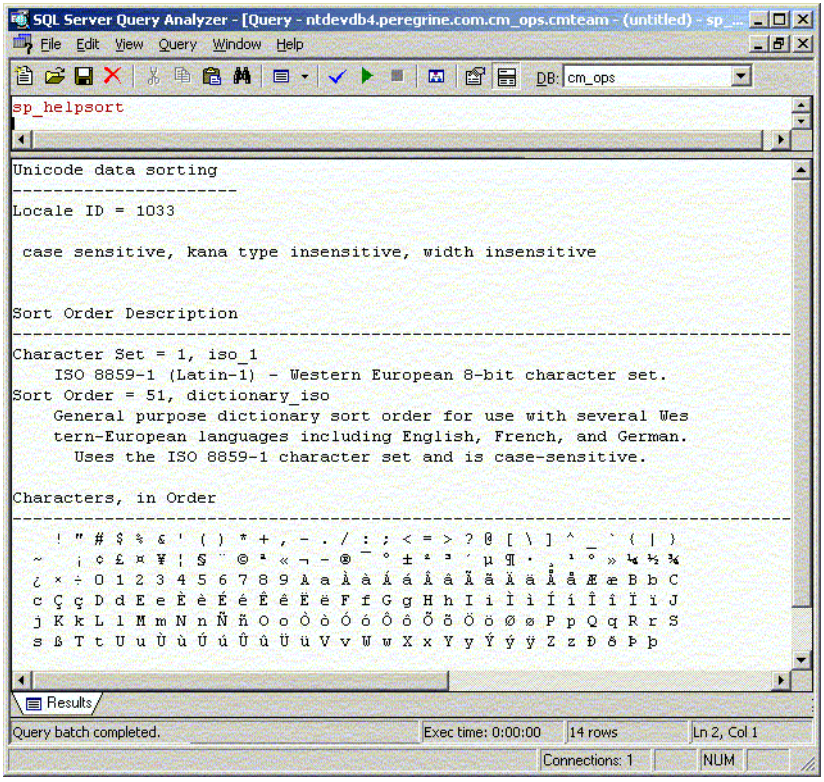


Figure 4-21: Checking Case-Sensitivity

To change the case-sensitivity:

▶ If the database is not set up the way you want it to be, check the MS SQL documentation for instructions on rebuilding the master database with the proper sort order and case-sensitivity.

Creating an SQL Server Database

In order for ServiceCenter to interact with SQL server, a database must be created for ServiceCenter to use.

To create a device:

1 Right-click the SQL Server Enterprise Manager to reveal the shortcut menu.

2 Select **New** > **Database** from the menu bar.

— Or —

Pull down the **Action** menu and select **New** > **Database**.

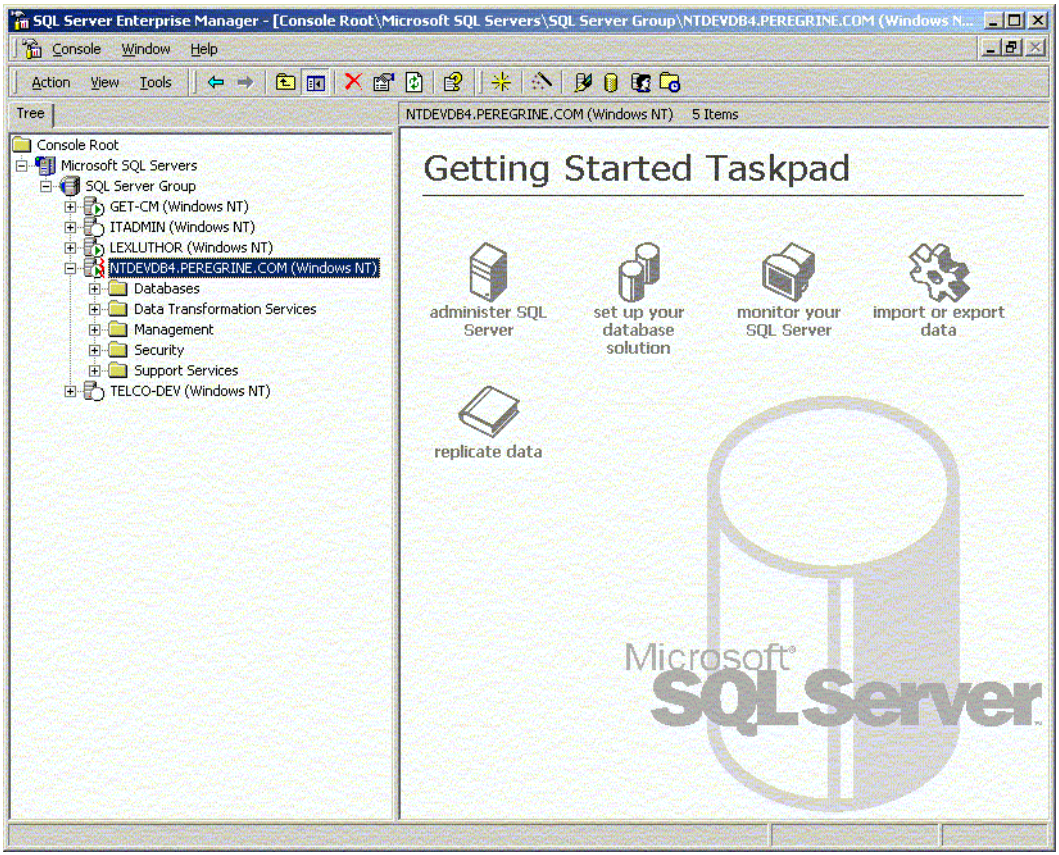


Figure 4-22: SQL Server Enterprise Manager

3 Provide a name for the new database.

4 Specify the location for the database.

When possible, separate data on a drive away from other disk drives.

5 Set the size constraints for the database.

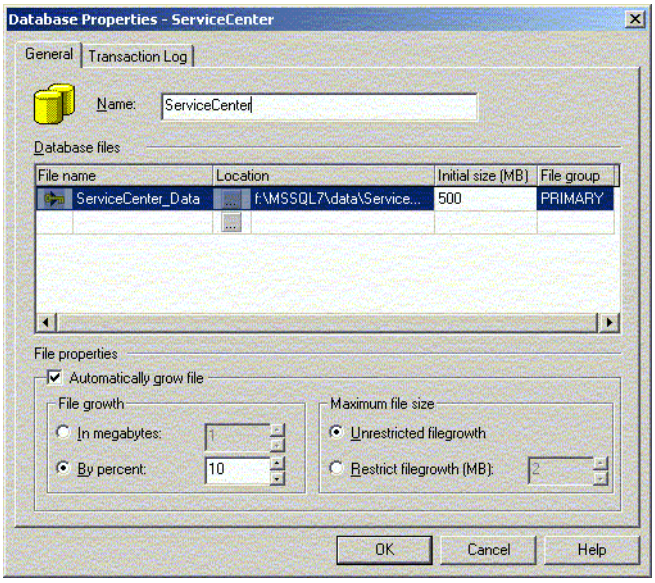


Figure 4-23: Database Properties — Size Constraints

6 Select the **Transaction Log** tab.

7 Specify the location for the transaction log.

8 Set the size constraints for the log.

9 Click **OK**.

To set other properties in the Database:

1 Right-click on the new database.

A popup menu is displayed.

2 Select **Properties** in the pop-up menu.

The ServiceCenter Properties dialog box is displayed. It has several tabs.

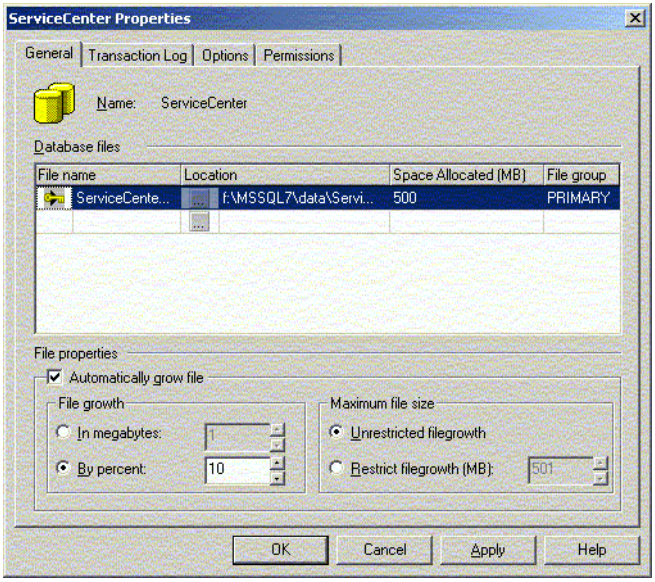


Figure 4-24: Database Properties — ServiceCenter Properties

3 Select the **Options** tab.

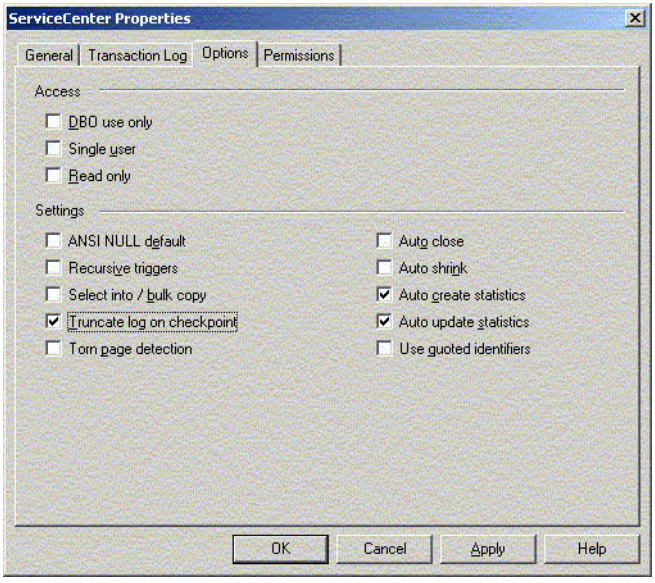


Figure 4-25: Database Properties — Options

4 Set up the database options appropriate for your intentions, including setting **Truncate Log on Checkpoint** to *true*.

Setting Up a User on the SQL Server

To set up a user ID:

1 Expand the security folder.

2 Right-click on **Logins** in the SQL Server Enterprise Manager to display the shortcut menu.

3 Select **new login** from the menu.

4 Create an SQL server user ID and login to be used by ServiceCenter.

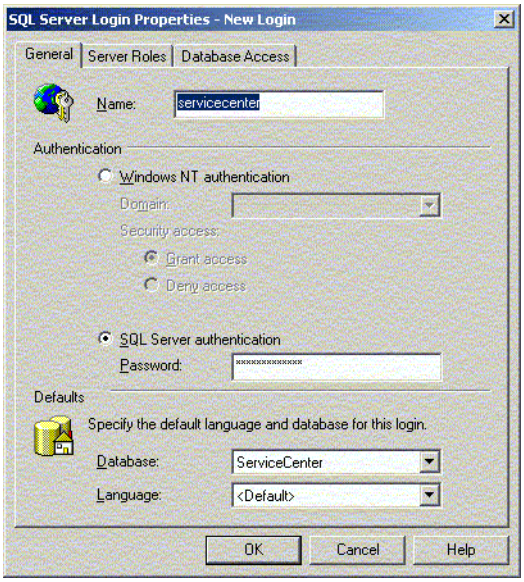


Figure 4-26: Server Login Properties — New Logins

5 When configuring the login, make sure you select **Permit** and **Default** in the ServiceCenter database.

6 Also grant DBO to the user for that database.

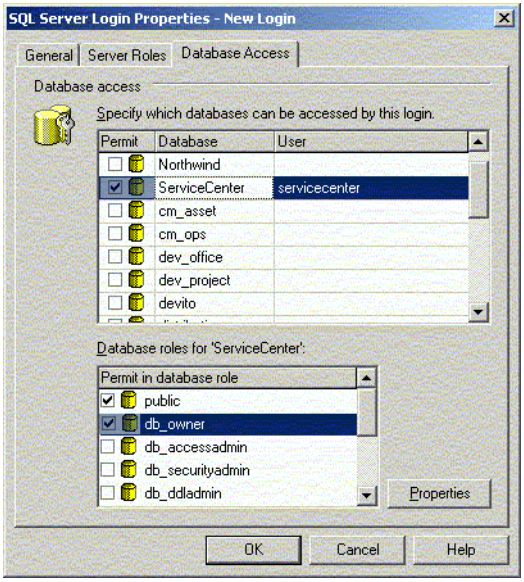


Figure 4-27: Server Login Properties — Database Access

On the ServiceCenter Side

To perform conversions, the client must connect to ServiceCenter in express mode (thin client). There can only be one user on ServiceCenter at the time of conversion to prevent file corruption. ServiceCenter employs a series of utilities to move data back and forth between the internal P4 system and the external RDBMS.

Actions on the ServiceCenter side:

1. Set the ServiceCenter server **sc.ini** file to connect to the SQL server database.
2. Ensure there is only one user on the ServiceCenter system.
3. Convert the ServiceCenter data to the SQL Server 7.0 database. This conversion is direction and platform dependent. See *Conversion to an RDBMS* on page 161 for further instructions.

Microsoft SQL Server Version 2000

The following steps convert ServiceCenter from the P4 file system to Microsoft SQL Server 2000.

To set up both to be case-insensitive:

1. Convert P4 so that it will do case-insensitive searching. (For instructions, see Volume 2 of Database Management and Administration.)
2. Confirm that your installation of MS SQL is case insensitive (default installation). See *To ensure that SQL server is set up to use the correct sorting:* on page 141 for instructions.
3. Change MS SQL to case insensitive searching if necessary. See *To change the case-sensitivity:* on page 134 for instructions.
4. Create an SQL server database device. See *Creating an SQL Server Database* on page 135 for instructions.
5. Set up a user ID on the SQL server database. See *Setting Up a User on the SQL Server* on page 138 for instructions.
6. Set up ServiceCenter and perform the conversion. See *On the ServiceCenter Side* on page 139 for instructions.

To set up both to be case sensitive:

1. Convert P4 so that it will do case-sensitive searching. This step will not be necessary unless you have previously converted P4 to be case-insensitive. (For instructions, see Volume 2 of Database Management and Administration.)
2. Confirm that your installation of MS SQL is case-sensitive (default installation). See *To ensure that SQL server is set up to use the correct sorting:* on page 141 for instructions.
3. Change MS SQL to case-sensitive searching if necessary. See *To ensure that SQL server is set up to use the correct sorting:* on page 141 for instructions.
4. Create an SQL server database device. See *Creating an SQL Server Database* on page 142 for instructions.
5. Set up a user ID on the SQL server database. See *Setting Up a User on the SQL Server Database* on page 148 for instructions.
6. Set up ServiceCenter and perform the conversion. See *Conversion to an RDBMS* on page 161 for instructions.

To ensure that SQL server is set up to use the correct sorting:

1 Use the Query Analyzer utility and select the sp_helpsort query to verify whether or not the SQL server database has been configured using case-sensitive sorting.

2 Check the results.

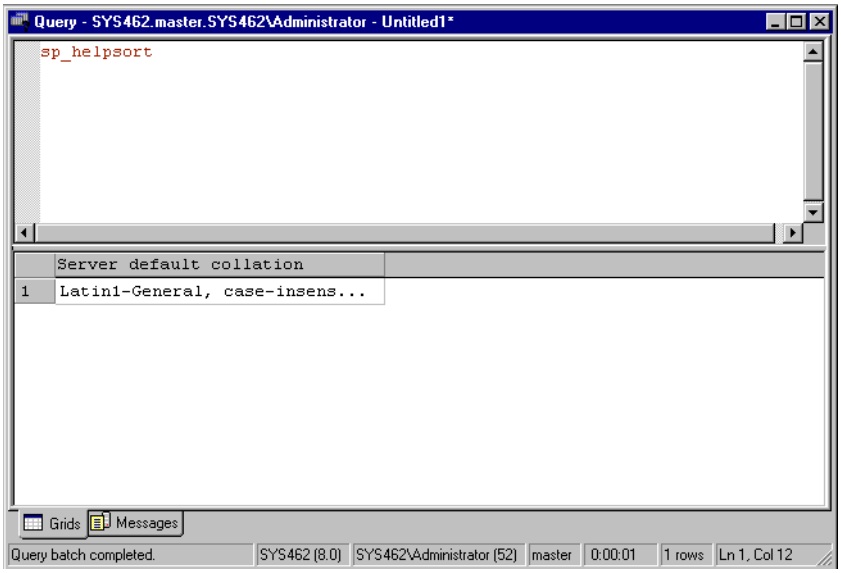


Figure 4-28: Checking Case-Sensitivity

To change the case-sensitivity:

If the database is no set up the way you want it to be, check the MS SQL documentation for instructions on rebuilding the master database with the proper sort order and case-sensitivity.

Creating an SQL Server Database

In order for ServiceCenter to interact with SQL server, a database must be created for ServiceCenter to use.

Creating a New Database

To create a database:

1 Right-click the SQL Server Enterprise Manager to reveal the shortcut menu.

2 Select **New** > **Database** from the menu bar.

— Or —

Open the SQL Server Enterprise Manager **Action** menu and select **New** > **Database** from the menu bar.

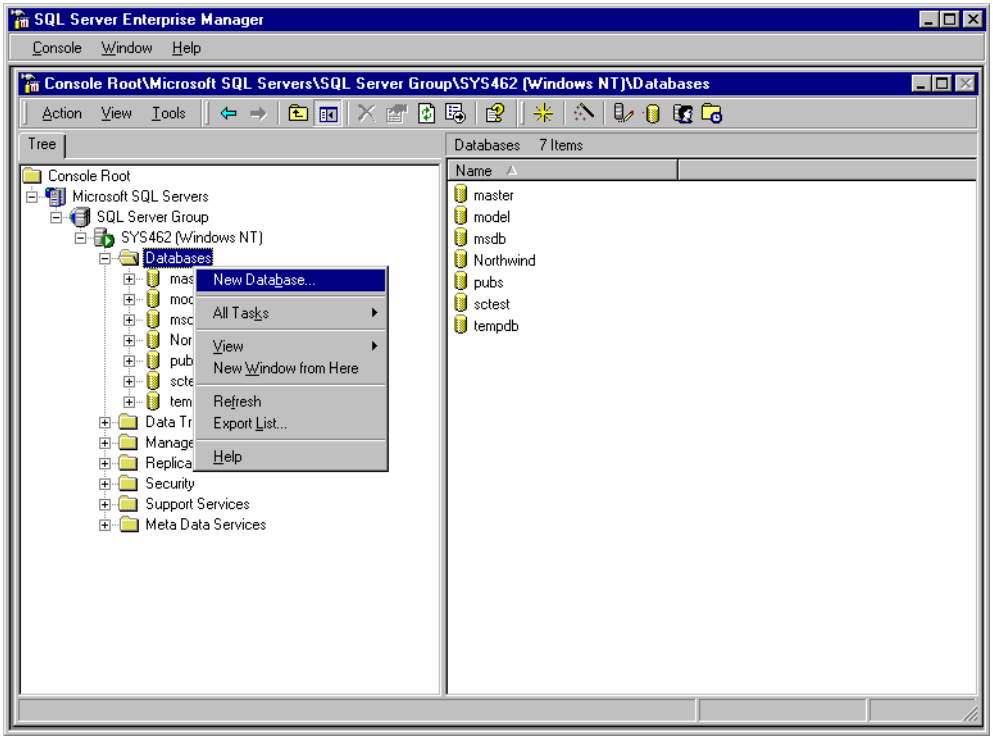


Figure 4-29: SQL Server Enterprise Manager

The Database Properties dialog box will be displayed.

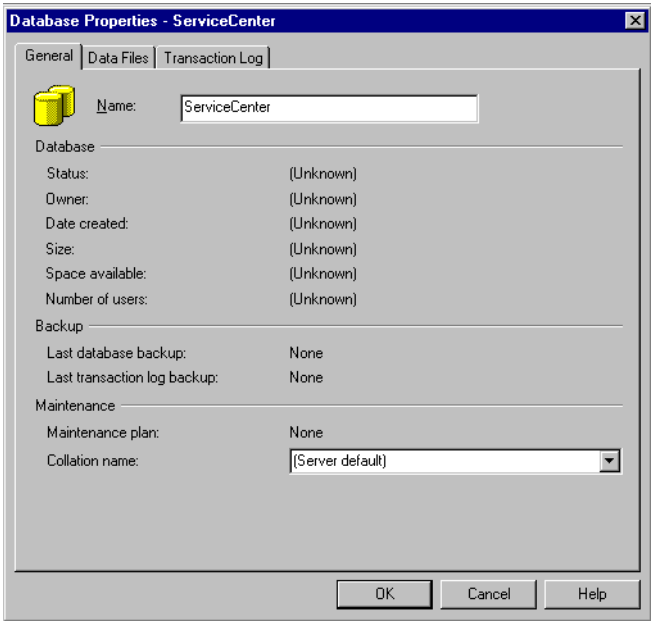


Figure 4-30: Database Properties - General Tab

3 Provide a name for the new database.

4 Specify a Collation name.

5 Select the **Data Files** tab.

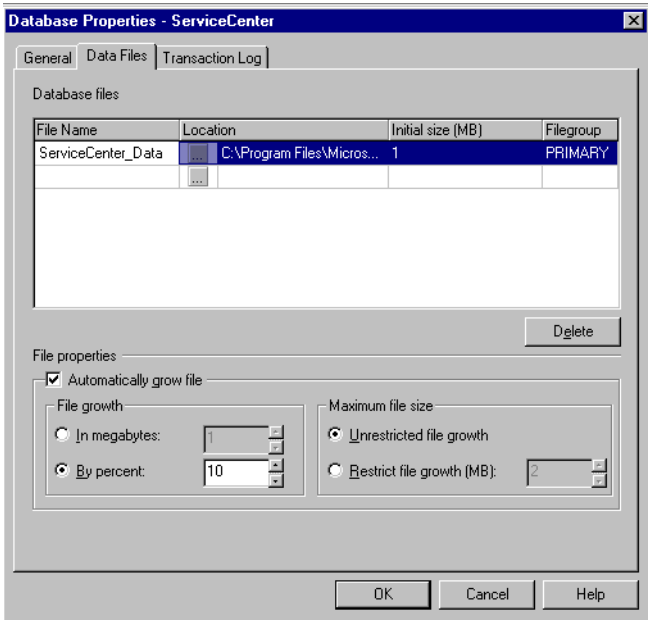


Figure 4-31: Database Properties - Datafiles Tab

6 Specify the location for the database. When possible, separate data on a drive away from other disk drives.

7 Set the size constraints for the database.

8 Select the **Translation Log** tab.

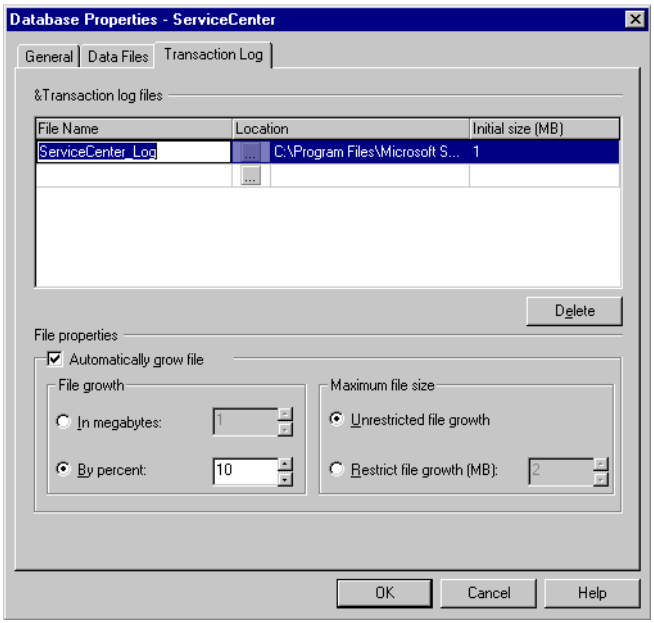


Figure 4-32: Database Properties - Transaction Log Tab

9 Specify the location for the transaction log.

10 Set the size constraints for the log.

11 Click **OK** to save the settings and exit.

To set other properties in the Database:

1 Right-click on the new database.

A popup menu is displayed.

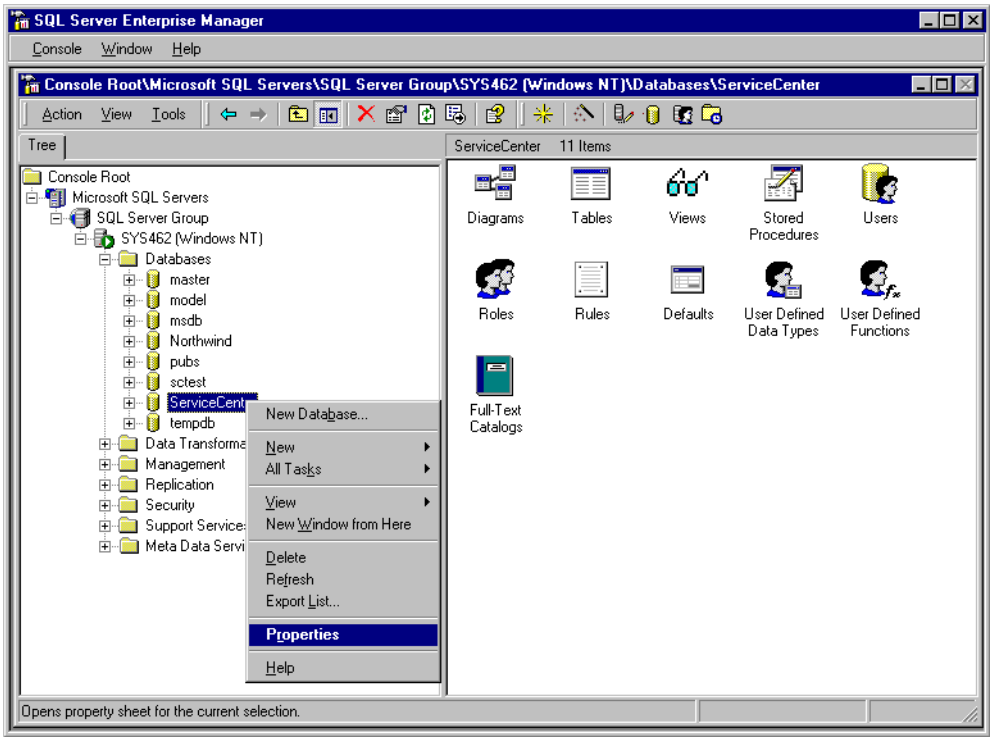


Figure 4-33: Set Properties

2 Select **Properties** in the pop-up menu.

The Database Properties dialog box is displayed. It has several tabs.

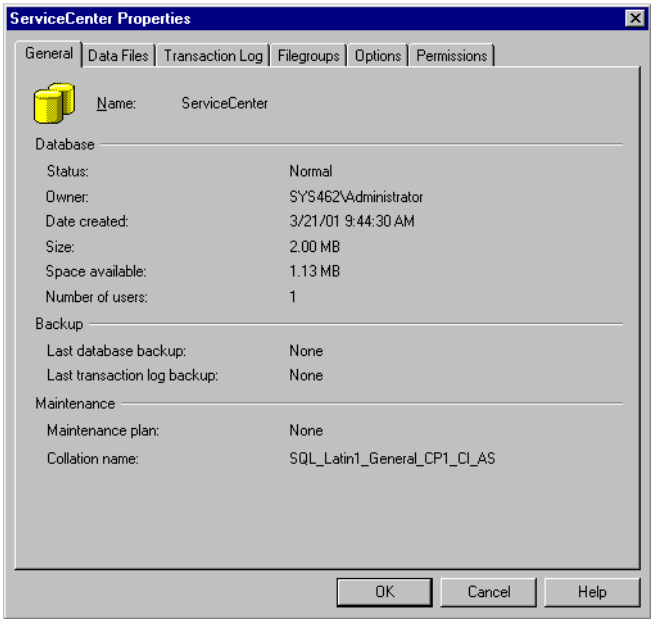


Figure 4-34: Database Properties - General Tab

3 Select the **File Groups** tab and set the appropriate file groups.

4 Select the **Options** tab and set the appropriate options.

5 Select the **Permissions** tab and set the appropriate permissions.

6 Set **TruncateLogonCheckpoint** to *true,* using the instructions in the Microsoft SQL server help.

7 Click **OK** to save changes and exit.

Setting Up a User on the SQL Server Database

To set up a user ID:

1. Expand the security folder.
2. Right-click on **Logins** in the SQL Server Enterprise Manager to display the shortcut menu.
3. Select **new login** from the menu.

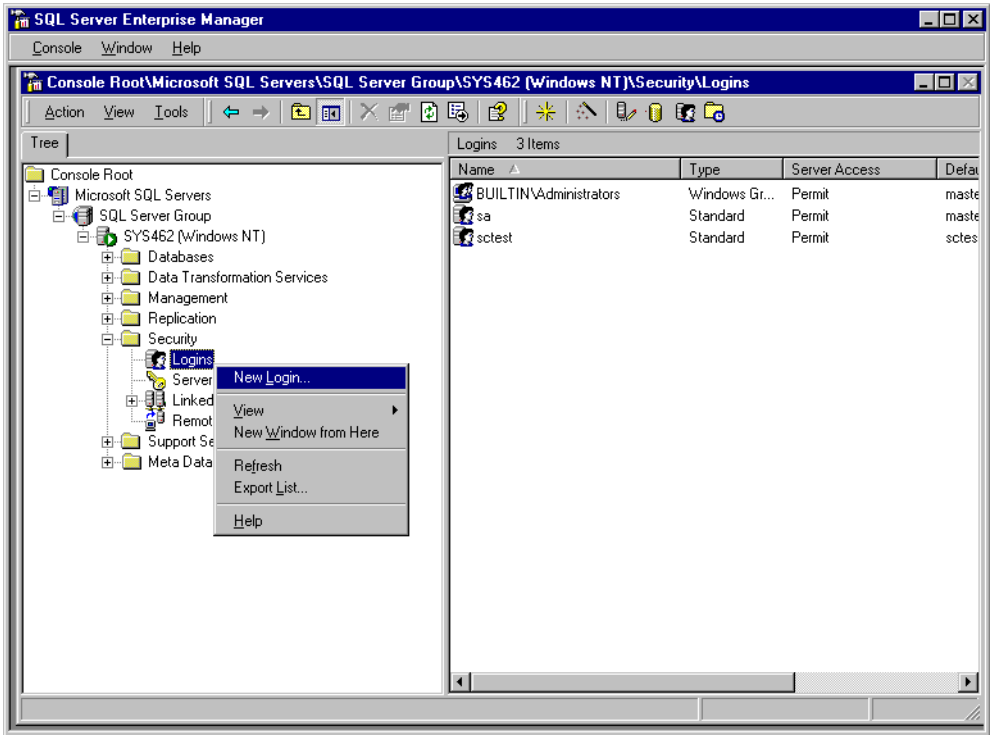


Figure 4-35: Selecting a new login in SQL Server Enterprise Manager

The SQL Server Login Properties - New Login dialog box is displayed. It has several tabs.

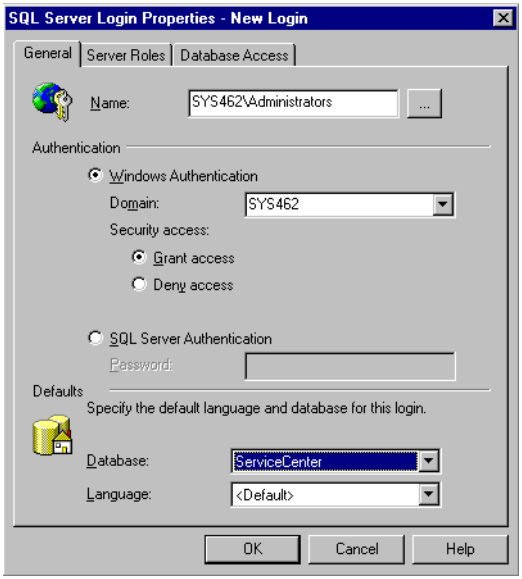


Figure 4-36: Server Login Properties - New Logins

4 Create an SQL server user ID and login to be used by ServiceCenter.

5 Select the **Server Roles** tab.

6 Select the **Database Access** tab.

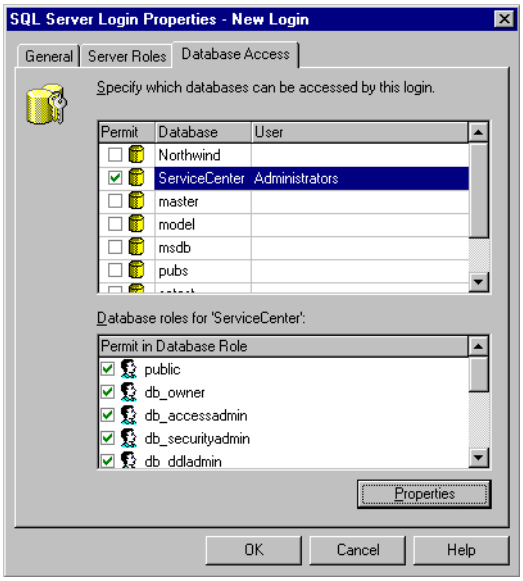


Figure 4-37: Server Login Properties - Database Access Tab

7 Select Permit for the Database User.

8 Select all roles in Permit in Database Role.

On the ServiceCenter Side

To perform conversions, the client must connect to ServiceCenter in express mode (thin client). There can only be one user on ServiceCenter at the time of conversion to prevent file corruption. ServiceCenter employs a series of utilities to move data back and forth between the internal P4 system and the external RDBMS.

Actions on the ServiceCenter side:

1 Set the ServiceCenter server **sc.ini** file to connect to the SQL server database.

2 Ensure there is only one user on the ServiceCenter system.

3 Convert the ServiceCenter data to the SQL server 2000 database. This conversion is direction and platform dependent. See *Conversion to an RDBMS* on page 161 for further instructions.

Sybase 12.5 Preparation

Before starting conversion to Sybase 12.5, the following changes must be made:

Step 1 Change the default Sybase auto identity setting to false.

Step 2 Add output as an RDBMS reserved word in ServiceCenter.

Changing the Sybase Auto Identity Option to False

In Sybase 12.5, when the auto identity option is set to true (on), SYB_IDENTITY_COL is added in each new table created without specifying a primary key, constraint, or IDENTITY. Adding this column causes errors during the ServiceCenter file conversion to Sybase. To prevent those errors, set the auto-identity to false before starting conversion to Sybase 12.5.

In the example below, "sys600" is the service name created for the user's database, and "test" is the database created for conversion.

When text in brackets <> appears in the example below, type the text between the brackets <> at the Command prompt, and then press Enter. Do not type the brackets.

To set auto identity to false:

1. Open a DOS Command prompt.
2. Use the ISQL utility to log on to the database as sa.
3. <isql -Ssys600 -Usa>
4. Press Enter when prompted for a password.
5. <use master>
6. <go>
7. <master .. sp_dboption test, "auto identity", false >
8. <go>
9. <checkpoint>
10. <go>

Adding Output as a Reserved Word in ServiceCenter

The ServiceCenter sqlwords file needs an additional reserved word, output, for Sybase.

When text in brackets <> appears in the example below, type or select the text between the brackets. Do not type the brackets.

To add a reserved word to the file sqlwords:

1. Start ServiceCenter.
2. Start Database Manager.
3. Type <sqlwords> in the **File** textbox.
4. Click **Search**.
5. Select <sybase> from the **RDBMS Type** combo-box.
6. Type <output> in the **Reserved Word** textbox.
7. Click **Add**.

DB2 Universal Database Preparation

This section outlines the tuning and optimization recommendations when using ServiceCenter with IBM's DB2 Universal Database 7.x for Windows and Unix. These recommendations are intended only as a guide and should not be implemented on a production system without extensive testing.

The following recommendations assume the use of an SMS database and the implementation of conventional database tuning and performance measures. Actual results may vary on a customer-by-customer basis based on the tuning expertise available and hardware and software selections.

Set up ServiceCenter's general conversion parameters for DB2:

To set up the conversion parameters:

1. Open the Database Manager.
2. Type in sql.db.info as the form name, and then selecting the Search button.
3. Once the sql.db.info form is displayed, select the db2universal database type.
4. On the initial tab, Data Types, verify that Only One Long per Table is unchecked.

This is very important as DB2 supports multiple longs per table and will dramatically reduce the number of joins needed to complete a query.

5 On the second tab, Data Sizes, modify the Maximum Row Size parameter so that it coincides with the Page size in your DB2 database.

The default page size in DB2 UDB is 4096 bytes (4K). To attain best performance the DB2 tablespace for ServiceCenter should be set to use 32768 byte (32K) pages. For this page size, a value of 32000 for Maximum Row Size is best. This leaves room for DB2 overhead in each row while still optimizing the way ServiceCenter uses available space.

Modify the DDL that ServiceCenter Uses to Create Tables in DB2

Out of the box, ServiceCenter converts long text fields into DB2 as the data type Long Varchar. In DB2, this data type is expensive in regards to performance as it performs a direct I/O operation bypassing all bufferpools and caches and putting other transactions on hold until it is completed.

To address this performance issue requires modifying the ServiceCenter DDL to change these Long Varchar fields into the data type Varchar. The tables that are modified will vary based on implementation.

If a full conversion to DB2 is planned, not every Long Varchar in every table would necessarily need to be changed. If a table is being only occasionally accessed, the amount of work needed to change the Long Varchar would far out weigh the performance gain, if any. The amount of space allocated to each of these varchar fields will vary considerably based on the needs of you implementation.

Here is the DDL that was used to convert the Probsummary table for a recent benchmark (all fields listed as VARCHAR(x) were previously LONG VARCHAR):

```
CREATE TABLE
--P4[probsummary; M1; db2universal; {{""}, {""}, {}, {}}; Tot recs: 307824; Tot bytes:
1620693360]
 PROBSUMMARYM1
-- Tconstraints
  (

--P4[number; 1; M1; 0]                    --P4[sla.contact; 115; M1; 0]
 NUMBER CHAR(60),                        SLA_CONTACT CHAR(60),
   --P4[category; 2; M1; 0]               --P4[sla.vendor; 116; M1; 0]
 CATEGORY CHAR(60),                      SLA_VENDOR CHAR(60),
```

```
  --P4[open.time; 3; M1; 0]
OPEN_TIME TIMESTAMP,
  --P4[opened.by; 4; M1; 0]
OPENED_BY CHAR(50),
  --P4[priority.code; 5; M1; 0]
PRIORITY_CODE CHAR(50),
  --P4[severity.code; 6; M1; 0]
SEVERITY_CODE CHAR(40),
  --P4[update.time; 7; M1; 0]
UPDATE_TIME TIMESTAMP,
  --P4[assignment; 8; M1; 0]
ASSIGNMENT CHAR(60),
  --P4[referral.time; 9; M1; 0]
REFERRAL_TIME TIMESTAMP,
  --P4[referred.to; 10; M1; 0]
REFERRED_TO CHAR(140),
  --P4[alert.time; 11; M1; 0]
ALERT_TIME TIMESTAMP,
  --P4[status; 12; M1; 0]
STATUS CHAR(60),
  --P4[close.time; 13; M1; 0]
CLOSE_TIME TIMESTAMP,
  --P4[closed.by; 14; M1; 0]
CLOSED_BY CHAR(50),
  --P4[elapsed.time; 15; M1; 0]
ELAPSED_TIME TIMESTAMP,
  --P4[vendor; 16; M1; 0]
VENDOR CHAR(140),
  --P4[reference.no; 17; M1; 0]
REFERENCE_NO CHAR(50),
  --P4[contact.time; 18; M1; 0]
CONTACT_TIME TIMESTAMP,
 --P4[referral.to.contact; 19; M1; 0]
REFERRAL_TO_CONTA TIMESTAMP,
  --P4[onsite.time; 20; M1; 0]
ONSITE_TIME TIMESTAMP,
  --P4[contact.to.respond; 21; M1;
0]
CONTACT_TO_RESPON TIMESTAMP,
  --P4[repair.time; 22; M1; 0]
REPAIR_TIME TIMESTAMP,
  --P4[company.sla; 117; M1; 0]
COMPANY_SLA CHAR(30),
  --P4[subcategory; 118; M1; 0]
SUBCATEGORY CHAR(140),
  --P4[hot.tic; 119; M1; 0]
HOT_TIC CHAR(1),
  --P4[application.name; 120; M1; 0]
APPLICATION_NAME CHAR(140),
  --P4[solution.candidate; 121; M1; 0]
SOLUTION_CANDIDAT CHAR(1),
  --P4[agreement.id; 122; M1; 0]
AGREEMENT_ID FLOAT,
  --P4[planned.start; 123; M1; 0]
PLANNED_START TIMESTAMP,
  --P4[planned.end; 124; M1; 0]
PLANNED_END TIMESTAMP,
  --P4[y2k.related; 125; M1; 0]
Y2K_RELATED CHAR(1),
  --P4[operational.device; 126; M1; 0]
OPERATIONAL_DEVIC CHAR(1),
  --P4[junk; 127; M1; 0]
JUNK CHAR(1),
  --P4[contract.id; 128; M1; 0]
CONTRACT_ID FLOAT,
  --P4[sysmodcount; 129; M1; 0]
SYSMODCOUNT FLOAT,
  --P4[sysmoduser; 130; M1; 0]
SYSMODUSER CHAR(30),
  --P4[knownerror; 131; M1; 0]
KNOWNERROR CHAR(1),
  --P4[kpf.id; 132; M1; 0]
KPF_ID CHAR(30),
  --P4[ci.date.time; 133; M1; 0]
CI_DATE_TIME CHAR(30),
  --P4[flow; 134; M1; 0]
FLOW CHAR(30),
  --P4[server.id; 135; M1; 0]

SERVER_ID CHAR(30),
  --P4[units; 136; M1; 0]
UNITS CHAR(30),
```

```
  --P4[open.time; 3; M1; 0]
OPEN_TIME TIMESTAMP,
  --P4[opened.by; 4; M1; 0]
OPENED_BY CHAR(50),
  --P4[priority.code; 5; M1; 0]
PRIORITY_CODE CHAR(50),
  --P4[severity.code; 6; M1; 0]
SEVERITY_CODE CHAR(40),
  --P4[update.time; 7; M1; 0]
UPDATE_TIME TIMESTAMP,
  --P4[assignment; 8; M1; 0]
ASSIGNMENT CHAR(60),
  --P4[referral.time; 9; M1; 0]
REFERRAL_TIME TIMESTAMP,
  --P4[referred.to; 10; M1; 0]
REFERRED_TO CHAR(140),
  --P4[alert.time; 11; M1; 0]
ALERT_TIME TIMESTAMP,
  --P4[status; 12; M1; 0]
STATUS CHAR(60),
  --P4[close.time; 13; M1; 0]
CLOSE_TIME TIMESTAMP,
  --P4[closed.by; 14; M1; 0]
CLOSED_BY CHAR(50),
  --P4[elapsed.time; 15; M1; 0]
ELAPSED_TIME TIMESTAMP,
  --P4[vendor; 16; M1; 0]
VENDOR CHAR(140),
  --P4[reference.no; 17; M1; 0]
REFERENCE_NO CHAR(50),
  --P4[contact.time; 18; M1; 0]
CONTACT_TIME TIMESTAMP,
  --P4[referral.to.contact; 19; M1; 0]
REFERRAL_TO_CONTA TIMESTAMP,
  --P4[onsite.time; 20; M1; 0]
ONSITE_TIME TIMESTAMP,
  --P4[contact.to.respond; 21; M1;
0]
CONTACT_TO_RESPON TIMESTAMP,
  --P4[repair.time; 22; M1; 0]
REPAIR_TIME TIMESTAMP,
  --P4[company.sla; 117; M1; 0]
COMPANY_SLA CHAR(30),
  --P4[subcategory; 118; M1; 0]
SUBCATEGORY CHAR(140),
  --P4[hot.tic; 119; M1; 0]
HOT_TIC CHAR(1),
  --P4[application.name; 120; M1; 0]
APPLICATION_NAME CHAR(140),
  --P4[solution.candidate; 121; M1; 0]
SOLUTION_CANDIDAT CHAR(1),
  --P4[agreement.id; 122; M1; 0]
AGREEMENT_ID FLOAT,
  --P4[planned.start; 123; M1; 0]
PLANNED_START TIMESTAMP,
  --P4[planned.end; 124; M1; 0]
PLANNED_END TIMESTAMP,
  --P4[y2k.related; 125; M1; 0]
Y2K_RELATED CHAR(1),
  --P4[operational.device; 126; M1; 0]
OPERATIONAL_DEVIC CHAR(1),
  --P4[junk; 127; M1; 0]
JUNK CHAR(1),
  --P4[contract.id; 128; M1; 0]
CONTRACT_ID FLOAT,
  --P4[sysmodcount; 129; M1; 0]
SYSMODCOUNT FLOAT,
  --P4[sysmoduser; 130; M1; 0]
SYSMODUSER CHAR(30),
  --P4[knownerror; 131; M1; 0]
KNOWNERROR CHAR(1),
  --P4[kpf.id; 132; M1; 0]
KPF_ID CHAR(30),
  --P4[ci.date.time; 133; M1; 0]
CI_DATE_TIME CHAR(30),
  --P4[flow; 134; M1; 0]
FLOW CHAR(30),
  --P4[server.id; 135; M1; 0]

SERVER_ID CHAR(30),
  --P4[units; 136; M1; 0]
UNITS CHAR(30),
```

```
  --P4[onsite.to.repair; 23; M1; 0]
ONSITE_TO_REPAIR TIMESTAMP,
  --P4[backup.start; 24; M1; 0]
BACKUP_START TIMESTAMP,
  --P4[backup.time; 25; M1; 0]
BACKUP_TIME TIMESTAMP,
  --P4[backup.end; 26; M1; 0]
BACKUP_END TIMESTAMP,
  --P4[downtime; 27; M1; 0]
DOWNTIME TIMESTAMP,
  --P4[cause.code; 28; M1; 0]
CAUSE_CODE CHAR(50),
  --P4[resolution.code; 29; M1; 0]
RESOLUTION_CODE CHAR(50),
  --P4[logical.name; 30; M1; 0]
LOGICAL_NAME CHAR(60),
  --P4[group; 32; M1; 0]
GROUP CHAR(50),
  --P4[job.name; 33; M1; 0]
JOB_NAME CHAR(60),
  --P4[location; 34; M1; 0]
LOCATION CHAR(140),
  --P4[version; 35; M1; 0]
VERSION CHAR(60),
  --P4[type; 36; M1; 0]
TYPE CHAR(60),
  --P4[abend.code; 37; M1; 0]
ABEND_CODE CHAR(60),
  --P4[model; 38; M1; 0]
MODEL CHAR(60),
  --P4[action; 39; M1; 1]
ACTION VARCHAR(8000),
  --P4[resolution; 42; M1; 0]
RESOLUTION VARCHAR(4000),
  --P4[affected; 44; M1; 0]
AFFECTED VARCHAR(1000),
  --P4[xreference; 48; M1; 0]
XREFERENCE VARCHAR(1000),
  --P4[alert1; 49; M1; 0]
ALERT1 CHAR(1),
  --P4[alert2; 50; M1; 0]
  --P4[value; 137; M1; 0]
VALUE CHAR(30),
  --P4[port.index; 138; M1; 0]
PORT_INDEX CHAR(30),
  --P4[system.state; 139; M1; 0]
SYSTEM_STATE CHAR(30),
  --P4[payroll.no; 140; M1; 0]
PAYROLL_NO CHAR(30),
  --P4[critical.user; 141; M1; 0]
CRITICAL_USER CHAR(30),
  --P4[room.floor.ref; 142; M1; 0]
ROOM_FLOOR_REF CHAR(30),
  --P4[user.type; 143; M1; 0]
USER_TYPE CHAR(30),
  --P4[site.category; 144; M1; 0]
SITE_CATEGORY CHAR(30),
  --P4[total.loss; 145; M1; 0]
TOTAL_LOSS CHAR(1),
  --P4[product.type; 146; M1; 0]
PRODUCT_TYPE CHAR(50),
  --P4[problem.type; 147; M1; 0]
PROBLEM_TYPE CHAR(50),
  --P4[fix.type; 148; M1; 0]
FIX_TYPE CHAR(40),
  --P4[no.SDU.fix; 149; M1; 0]
NO_SDU_FIX CHAR(1),
  --P4[resolved.by; 150; M1; 0]
RESOLVED_BY CHAR(50),
  --P4[cost.centre; 151; M1; 0]
COST_CENTRE CHAR(40),
  --P4[customer.no; 152; M1; 0]
CUSTOMER_NO CHAR(30),
  --P4[unsuspend.time; 153; M1; 0]
UNSUSPEND_TIME TIMESTAMP,
  --P4[critical.device; 154; M1; 0]
CRITICAL_DEVICE CHAR(1),
  --P4[serial.no; 155; M1; 0]
SERIAL_NO CHAR(30),
  --P4[failing.serial.no; 156; M1; 0]
FAILING_SERIAL_NO CHAR(30),
  --P4[third.party.name; 158; M1; 0]
```

```
ALERT2 CHAR(1),
  --P4[alert3; 51; M1; 0]
ALERT3 CHAR(1),
  --P4[deadline; 52; M1; 0]
DEADLINE CHAR(1),
  --P4[reassigned; 53; M1; 0]
REASSIGNED CHAR(1),
  --P4[id; 54; M1; 0]
ID CHAR(60),
  --P4[lookup.time; 55; M1; 0]
LOOKUP_TIME TIMESTAMP,
  --P4[total.pages; 56; M1; 0]
TOTAL_PAGES FLOAT,
  --P4[flag; 57; M1; 0]
FLAG CHAR(1),
  --P4[downtime.end; 58; M1; 0]
DOWNTIME_END TIMESTAMP,
  --P4[downtime.start; 59; M1; 0]
DOWNTIME_START TIMESTAMP,
  --P4[assignee.name; 60; M1; 0]
ASSIGNEE_NAME CHAR(50),
  --P4[respond.time; 61; M1; 0]
RESPOND_TIME TIMESTAMP,
  --P4[contact.name; 62; M1; 0]
CONTACT_NAME CHAR(140),
  --P4[seconds; 64; M1; 0]
SECONDS FLOAT,
  --P4[caller.id; 65; M1; 0]
CALLER_ID CHAR(50),
  --P4[contact.phone; 66; M1; 0]
CONTACT_PHONE CHAR(50),
  --P4[actor; 69; M1; 0]
ACTOR CHAR(50),
  --P4[format; 70; M1; 0]
FORMAT CHAR(140),
  --P4[count; 71; M1; 0]
COUNT FLOAT,
  --P4[respond.to.onsite; 72; M1; 0]
RESPOND_TO_ONSITE TIMESTAMP,
  --P4[network.name; 73; M1; 0]
THIRD_PARTY_NAME VARCHAR(500),
  --P4[third.party.reference; 160; M1; 0]
THIRD_PARTY_REFER VARCHAR(500),
  --P4[third.party.referred; 162; M1; 0]
THIRD_PARTY_REFER1 VARCHAR(500),
  --P4[third.party.referred.by; 164; M1; 0]
THIRD_PARTY_REFER2 VARCHAR(500),
  --P4[class; 165; M1; 0]
CLASS CHAR(40),
  --P4[alternate.contact; 166; M1; 0]
ALTERNATE_CONTACT CHAR(40),
  --P4[site.visit.date; 167; M1; 0]
SITE_VISIT_DATE CHAR(40),
  --P4[site.visit.technician; 168; M1; 0]
SITE_VISIT_TECHNI CHAR(40),
  --P4[operating.system; 169; M1; 0]
OPERATING_SYSTEM CHAR(40),
  --P4[os.release.level; 170; M1; 0]
OS_RELEASE_LEVEL CHAR(40),
  --P4[os.maint.level; 171; M1; 0]
OS_MAINT_LEVEL CHAR(40),
  --P4[manufacturer; 172; M1; 0]
MANUFACTURER CHAR(40),
  --P4[failing.component; 173; M1; 0]
FAILING_COMPONENT CHAR(40),
  --P4[country; 174; M1; 0]
COUNTRY CHAR(30),
  --P4[cusomter.reference; 175; M1; 0]
CUSOMTER_REFERENC CHAR(30),
  --P4[expd.response.time; 177; M1; 0]
EXPD_RESPONSE_TIM VARCHAR(100),
  --P4[oti.originator; 178; M1; 0]
OTI_ORIGINATOR CHAR(60),
  --P4[oti.originator.reference; 179; M1; 0]
OTI_ORIGINATOR_RE CHAR(60),
  --P4[oti.originator.version; 180; M1; 0]
OTI_ORIGINATOR_VE CHAR(60),
  --P4[oti.tosc.consumer; 181; M1; 0]
OTI_TOSC_CONSUMER CHAR(30),
  --P4[oti.tosc.consumer.reference; 182; M1;
0]
```

```
NETWORK_NAME CHAR(60),
  --P4[final.close; 74; M1; 0]
FINAL_CLOSE TIMESTAMP,
  --P4[open.group; 75; M1; 0]
OPEN_GROUP CHAR(50),
  --P4[alert.status; 76; M1; 0]
ALERT_STATUS CHAR(50),
  --P4[deadline.group; 77; M1; 0]
DEADLINE_GROUP CHAR(50),
  --P4[deadline.alert; 78; M1; 0]
DEADLINE_ALERT TIMESTAMP,
  --P4[pending.date; 79; M1; 0]
PENDING_DATE TIMESTAMP,
  --P4[referral.count; 80; M1; 0]
REFERRAL_COUNT FLOAT,
  --P4[pending.reason; 81; M1; 0]
PENDING_REASON CHAR(140),
  --P4[network.address; 82; M1; 0]
NETWORK_ADDRESS CHAR(60),
  --P4[outage.type; 83; M1; 0]
OUTAGE_TYPE CHAR(60),
  --P4[parent; 84; M1; 0]
PARENT CHAR(60),
  --P4[domain; 85; M1; 0]
DOMAIN CHAR(60),
  --P4[callback.list; 87; M1; 0]
CALLBACK_LIST VARCHAR(1000),
  --P4[closing.comments; 89; M1; 0]
CLOSING_COMMENTS
VARCHAR(1000),
  --P4[cs.code; 90; M1; 0]
CS_CODE CHAR(30),
  --P4[change.no; 91; M1; 0]
CHANGE_NO FLOAT,
  --P4[last.name; 92; M1; 0]
LAST_NAME CHAR(80),
  --P4[first.name; 93; M1; 0]
FIRST_NAME CHAR(80),
  --P4[company; 94; M1; 0]
COMPANY CHAR(140),
OTI_TOSC_CONSUMER1 CHAR(30),
  --P4[oti.tosc.provider; 183; M1; 0]
OTI_TOSC_PROVIDER CHAR(30),
  --P4[oti.tosc.provider.reference; 184; M1; 0]
OTI_TOSC_PROVIDER1 CHAR(30),
  --P4[oti.message.type; 185; M1; 0]
OTI_MESSAGE_TYPE CHAR(40),
  --P4[oti.fromsc.consumer; 186; M1; 0]
OTI_FROMSC_CONSUM CHAR(30),
  --P4[oti.fromsc.provider; 187; M1; 0]
OTI_FROMSC_PROVID CHAR(30),
  --P4[oti.fromsc.consumer.reference; 188;
M1; 0]
OTI_FROMSC_CONSUM1 CHAR(30),
  --P4[oti.fromsc.provider.reference; 189; M1;
0]
OTI_FROMSC_PROVID1 CHAR(30),
  --P4[pending.change; 190; M1; 0]
PENDING_CHANGE CHAR(1),
  --P4[mandatory.asset; 191; M1; 0]
MANDATORY_ASSET CHAR(1),
  --P4[reg.error; 192; M1; 0]
REG_ERROR CHAR(1),
  --P4[cus.error; 193; M1; 0]
CUS_ERROR CHAR(1),
  --P4[variable1; 194; M1; 0]
VARIABLE1 CHAR(30),
  --P4[variable2; 195; M1; 0]
VARIABLE2 CHAR(30),
  --P4[variable3; 196; M1; 0]
VARIABLE3 CHAR(30),
  --P4[call.origin; 197; M1; 0]
CALL_ORIGIN CHAR(30),
  --P4[source; 198; M1; 0]
SOURCE CHAR(30),
  --P4[first.time.fix; 199; M1; 0]
FIRST_TIME_FIX CHAR(1),
  --P4[resolved.group; 200; M1; 0]
RESOLVED_GROUP CHAR(50),
  --P4[resolved.time; 201; M1; 0]
RESOLVED_TIME TIMESTAMP,
```

```
  --P4[start.time; 95; M1; 0]
 START_TIME TIMESTAMP,
  --P4[title; 96; M1; 0]
 TITLE CHAR(140),
  --P4[brief.description; 97; M1; 1]
 BRIEF_DESCRIPTION VARCHAR(500),
  --P4[document.id; 98; M1; 0]
 DOCUMENT_ID CHAR(50),
  --P4[foreign; 99; M1; 0]
 FOREIGN FLOAT,
  --P4[foreign.id; 100; M1; 0]
 FOREIGN_ID CHAR(50),
  --P4[dept; 101; M1; 0]
 DEPT CHAR(60),
  --P4[serial.no.; 102; M1; 0]
 SERIAL_NO_ CHAR(60),
  --P4[building; 103; M1; 0]
 BUILDING CHAR(60),
  --P4[floor; 104; M1; 0]
 FLOOR CHAR(60),
  --P4[quote.no; 105; M1; 0]
 QUOTE_NO CHAR(60),
  --P4[ticket.owner; 106; M1; 0]
 TICKET_OWNER CHAR(60),
  --P4[incident.id; 107; M1; 0]
 INCIDENT_ID CHAR(30),
  --P4[sysorgsite; 108; M1; 0]
 SYSORGSITE FLOAT,
  --P4[syshomesite; 109; M1; 0]
 SYSHOMESITE FLOAT,
  --P4[sysmodtime; 110; M1; 0]
 SYSMODTIME TIMESTAMP,
  --P4[updated.by; 111; M1; 0]
 UPDATED_BY CHAR(50),
  --P4[problem.status; 112; M1; 0]    );
 PROBLEM_STATUS CHAR(60),
  --P4[closed.group; 202; M1; 0]
 CLOSED_GROUP CHAR(50),
  --P4[sla.alert.time; 203; M1; 0]
 SLA_ALERT_TIME TIMESTAMP,
  --P4[contact.location; 204; M1; 0]
 CONTACT_LOCATION CHAR(40),
  --P4[srvc.manager; 205; M1; 0]
 SRVC_MANAGER CHAR(30),
  --P4[srvc.del.manager; 206; M1; 0]
 SRVC_DEL_MANAGER CHAR(30),
  --P4[different.from.contact; 207; M1; 0]
 DIFFERENT_FROM_CO CHAR(1),
  --P4[alternate.fax; 208; M1; 0]
 ALTERNATE_FAX CHAR(30),
  --P4[alternate.extesnion; 209; M1; 0]
 ALTERNATE_EXTESNI CHAR(30),
  --P4[alternate.phone; 210; M1; 0]
 ALTERNATE_PHONE CHAR(50),
  --P4[user.priority; 211; M1; 0]
 USER_PRIORITY CHAR(40),
  --P4[sla.expire; 212; M1; 0]
 SLA_EXPIRE TIMESTAMP,
  --P4[corp.structure; 213; M1; 0]
 CORP_STRUCTURE CHAR(40),
  --P4[res.anal.code; 214; M1; 0]
 RES_ANAL_CODE CHAR(30),
  --P4[last.activity; 215; M1; 0]
 LAST_ACTIVITY CHAR(30),
  --P4[mobile.checkout; 216; M1; 0]
 MOBILE_CHECKOUT CHAR(1),
  --P4[location.full.name; 217; M1; 0]
 LOCATION_FULL_NAM CHAR(30)

CREATE TABLE
--P4[probsummary; A1; db2universal; {{""}, {""}, {}, {}}; Tot recs: 307824; Tot bytes:
13544256]
 PROBSUMMARYA1
-- Tconstraints
 (
```

```
--P4[number; 1; M1; 0]
 NUMBER CHAR(60),
  record_number INTEGER,
  --P4[key.words; 46; A1; 0]
 KEY_WORDS CHAR(60)

);

CREATE TABLE
--P4[probsummary; A2; db2universal; {{""}, {""}, {}, {}}; Tot recs: 307824; Tot bytes:
13544256]
 PROBSUMMARYA2
-- Tconstraints
  (
--P4[number; 1; M1; 0]
 NUMBER CHAR(60),
  record_number INTEGER,
  --P4[secondary.assignment; 114; A2; 0]
 SECONDARY_ASSIGNM CHAR(50)
);

CREATE TABLE
--P4[probsummary; A3; db2universal; {{""}, {""}, {}, {}}; Tot recs: 307824; Tot bytes:
13544256]
 PROBSUMMARYA3
-- Tconstraints
  (
--P4[number; 1; M1; 0]
 NUMBER CHAR(60),
  record_number INTEGER,
 --P4[update.action; 67; A3; 1]
 UPDATE_ACTION VARCHAR(20000),
);

COMMIT
;
```

At the end of the above DDL, note the addition of the table PROBSUMMARYA3. This table was created to hold the data for the UPDATE_ACTION field. Due to the 32k page size (row length) limitation, there wasn't enough room left in the row to provide adequate space for all the text fields. The decision made in this case was to move the largest one, UPDATE_ACTION (history), to it's own table. It is important to reiterate that the sizes chosen in the above DDL are considered to be a guideline that will work for the majority of implementations. However, it is very important that the needs and proposed use of the ServiceCenter implementation be carefully reviewed and tested before rollout to a production system.

CHAPTER 5 Conversion to an RDBMS

This chapter was designed to explain the process of converting the P4 file system to a Relational Database Management System (RDBMS) to ServiceCenter system and database administrators.

Topics in this chapter include:

Pre-Conversion Server Preparation

During file conversion, data reference errors may occur if more than one user is on the system. To avoid this situation, the system administrator performing the file conversion must be the only user on the system, and no background processes can be running.

Three things must be done to prepare the server for conversion:

- *Ensure that No Background Processes are Running* on page 162
- *Establish a Connection with Express Listener on a Windows Client* on page 165
- *Log In with a Windows Client* on page 166

Ensure that No Background Processes are Running

There are two steps to ensuring that no background processes are running:

1. Shut down the Server. See *To shut down the server:* on page 162.
2. Restart it with no background processes running. See one of the following:
   - *To restart the server in express mode in Windows:* on page 163
   - *To restart the server in express mode in Unix:* on page 163
   - *To restart the server in express mode in OS/390:* on page 164

To shut down the server:

1. Choose an off-hour or non-peak time to perform the file conversion.
2. Click **System Status** on the System Administrator's main menu.
3. Use the **Broadcast** option from the Status window to send a message to all users, alerting them of the time when the system will be shut down.
4. Shut down the server at the scheduled time.

Do not restart the server from the ServiceCenter console, as the console starts all background processes. No other processes can be running on the system during the file conversion other than the Express Listener. All other processes will need to be manually killed if they are started. (*To kill any processes that should not be running:* on page 169.)

To restart the server in express mode in Windows:

1 Open a Windows Command prompt on the ServiceCenter application server.

Note: Do not close the Windows Command prompt window until you are finished with the conversion. Closing the window will shut down the server and client process.

2 Change to the ServiceCenter RUN directory.
The default is *C:\Program Files\ServiceCenter\Run*.

3 Restart ServiceCenter with only an *express* connection, using the following command:

start scenter -express:<*xxxx*>

— Or —

start /bg scenter -express:<*xxxx*> (Windows NT only)

Where xxxx is the port number used by the Express Listener. Specify a port number that is not otherwise used by another process on the computer or by regular ServiceCenter users. *12681* is the recommended port. (e.g., start scenter -express:12681)

— Or —

1 Comment out all lines except "scenter -listener" from the SC.CFG file.

2 Restart the ServiceCenter server.

3 Uncomment the lines when completed.

The server starts in *express* mode only, suppressing all background processes.

To restart the server in express mode in Unix:

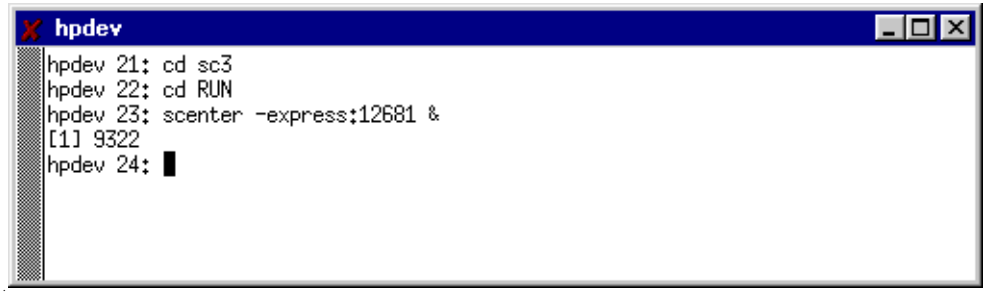
1 Change to the ServiceCenter/RUN directory.

2 Restart ServiceCenter with only an *express* connection, using the following command:

scenter -express:<*xxxx*> &

Where xxxx is the port number used by the Express Listener. Specify a port number that is not otherwise used by another process on the computer or by regular ServiceCenter users.

12681 is the recommended port. (e.g., scenter -express:12681 &).



The server starts in express mode only, suppressing all background processes.

To restart the server in express mode in OS/390:

1. Modify your PARMS member contained in the SAMPLIB PDS (created during installation).
2. Comment out the mvstcp line (APPC/CPI-C users must also comment out the mvscpic line).

   To comment out a line, place a # in column 1.
3. Modify the mvstcpexpress line to appear as follows:

   mvstcpexpress:*<xxxx>*

   Where xxxx is the port number used by the Express Listener. Specify a port number that is not otherwise used by another process on the computer or by regular ServiceCenter users. *12681* is the recommended port. (e.g., mvstcpexpress:12681).

   APPC/CPI-C users must modify the mvscpicexpress line.
4. If SCAutomate is being used, comment out that line as well.

   For standard TCP systems, the PARMS modifications appear as follows:

```
#mvstcp:12670

mvstcpexpress:12681
#scauto:12682
```

For systems that also implement APPC/CPI-C, the PARMS modifications appear as follows:

```
#mvscpic:sccpic14

mvscpicexpress:sccpic12
```

5 Add the following line to the PARM member:

nosystemstartup

6 Restart the ServiceCenter server.

The server starts an *Express Listener* at the specified port.

Note: If you do not use *nosystemstartup*, you will need to enter the Status window in the following section and manually kill all extraneous processes, leaving only the Express Listener, falcon and IO active.

The server starts in *express* mode only, suppressing all background processes.

Establish a Connection with Express Listener on a Windows Client

To establish a connection:

1 Right-click on the Express client desktop or the shortcut in the ServiceCenter folder.

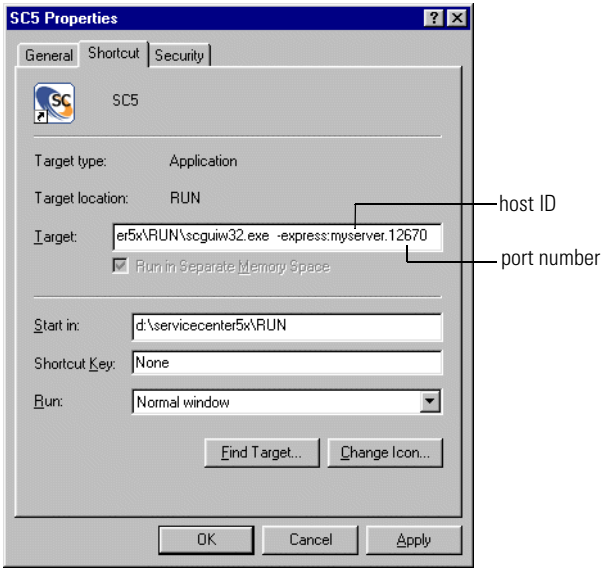
2 Select the **Properties** option from the drop down menu.

3 Modify the **Target** field on the **Shortcut** tab to establish connection with the Express Listener on the newly specified port.

For example:

C:\Program Files\ServiceCenter\RUN\Scguiw32.exe -express:<*hostID*>.12681

Where ***hostID*** is the name of the computer on which the server is running, for example, *express:mvs.12681*



4 Click **Apply** to save the new properties specifications.

Log In with a Windows Client

To log in:

1 Start an Express client with the modified shortcut.

The login prompt is displayed.

2 Log into ServiceCenter as a user with *system administrator* rights.

Warning: You will not be able to enter the SQL conversion utilities unless you are the only user, are using an express client, and there are no background processes running.

3 Click **System Status** in the system administrator's main menu.

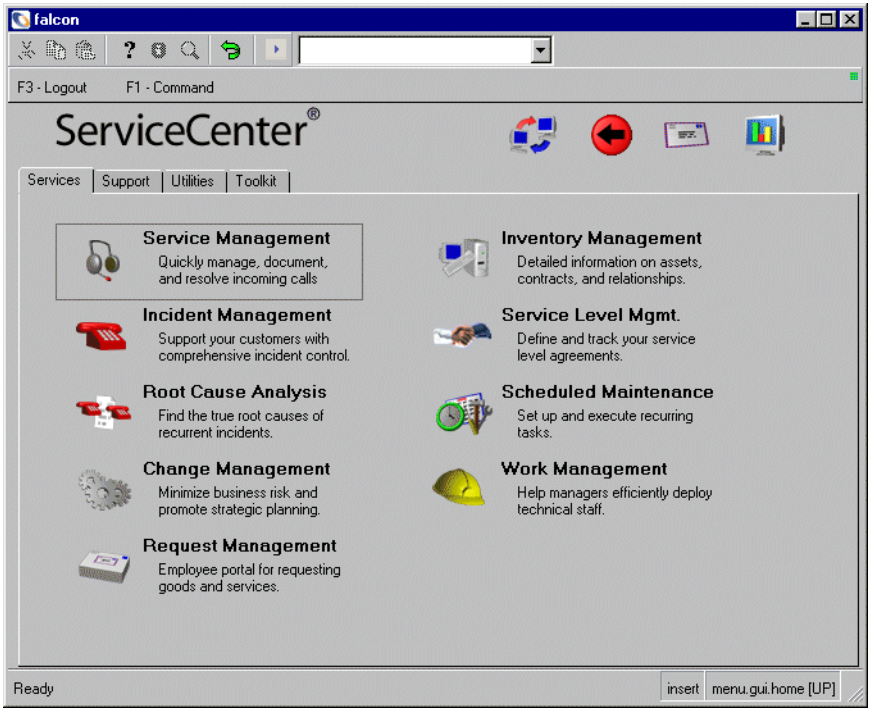


Figure 5-1: System Administrator's Main Menu

A list of processed records is displayed.

4 Windows and Unix users should verify no background processes are running and that the **login** used is the only process listed, as displayed below.

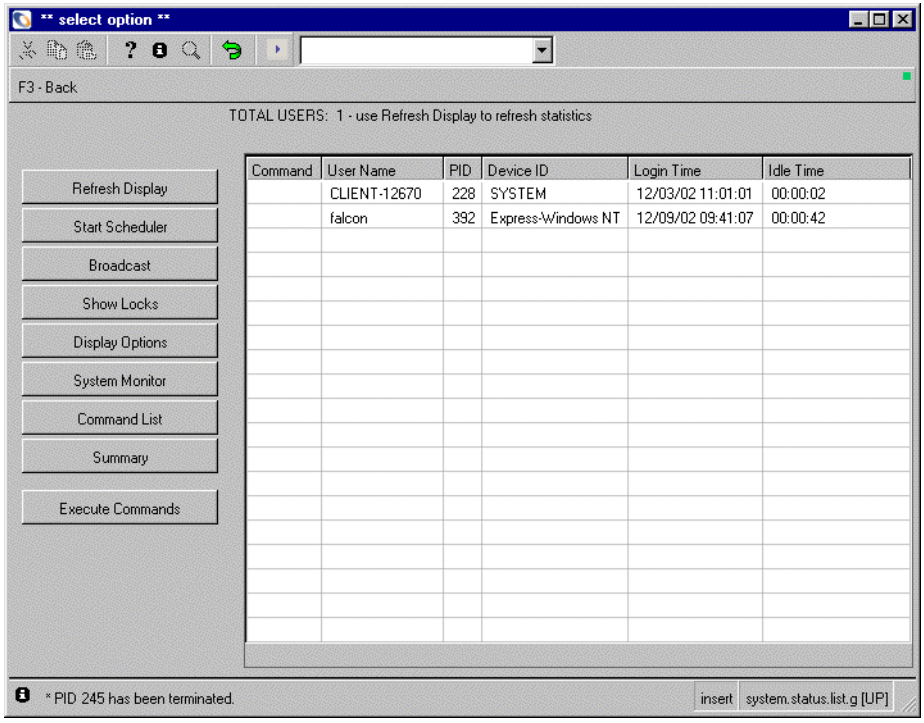


Figure 5-2: ServiceCenter Status Window

To kill any processes that should not be running:

1 Type a k next to each of the processes to be killed.

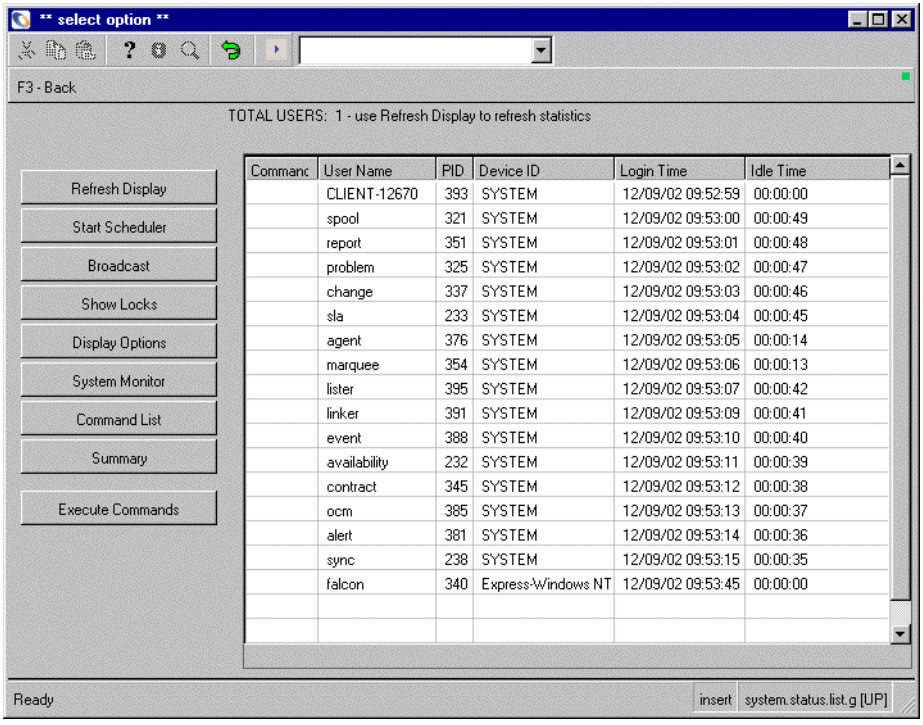


Figure 5-3: Killing Background Processes

2 Click **Execute Commands** to kill the processes.

3 To determine if all processes have been killed, click **Refresh Display**. You should see only the processes listed in Figure 5-2 on page 168.

Conversion Process

The conversion procedure is done from the ServiceCenter client, and is roughly the same for all RDBMS, with minor variations dependent on the server platform.

When the necessary preparations (See *Pre-Conversion Server Preparation* on page 162.) have been completed, you are ready to begin the actual RDBMS conversion. Conversion is typically run from a client workstation. The process will take a considerable amount of time. It will create 2500 tables and indexes and is very disk intensive.

To open the SQL Menu:

1 Select the **Utilities** tab in the System Administrator's main menu.

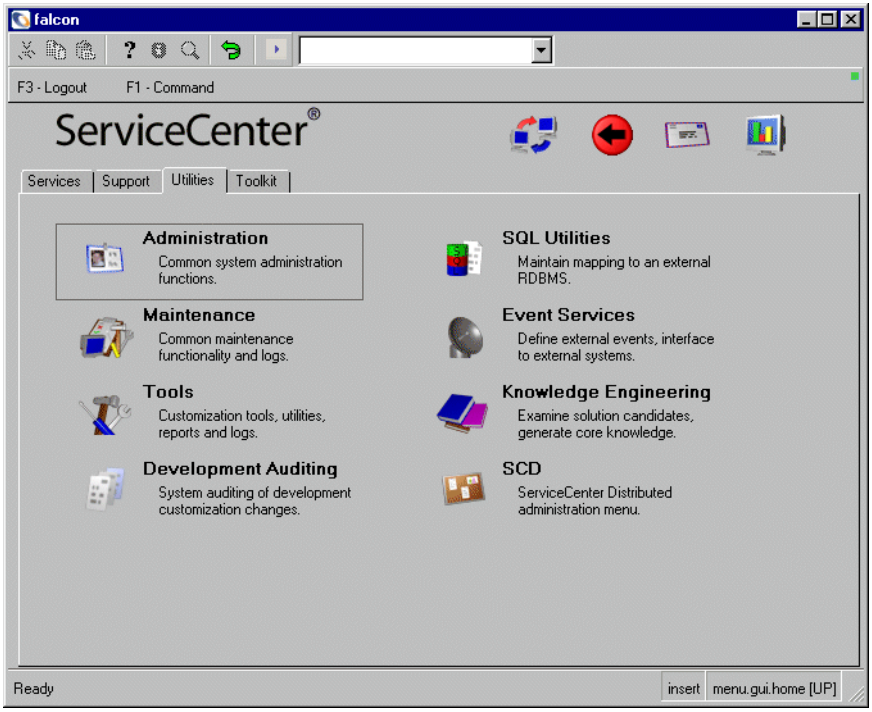


Figure 5-4: Home Menu — Utilities Tab

2 Click **SQL Utilities**.

The **SQL Menu** is displayed.

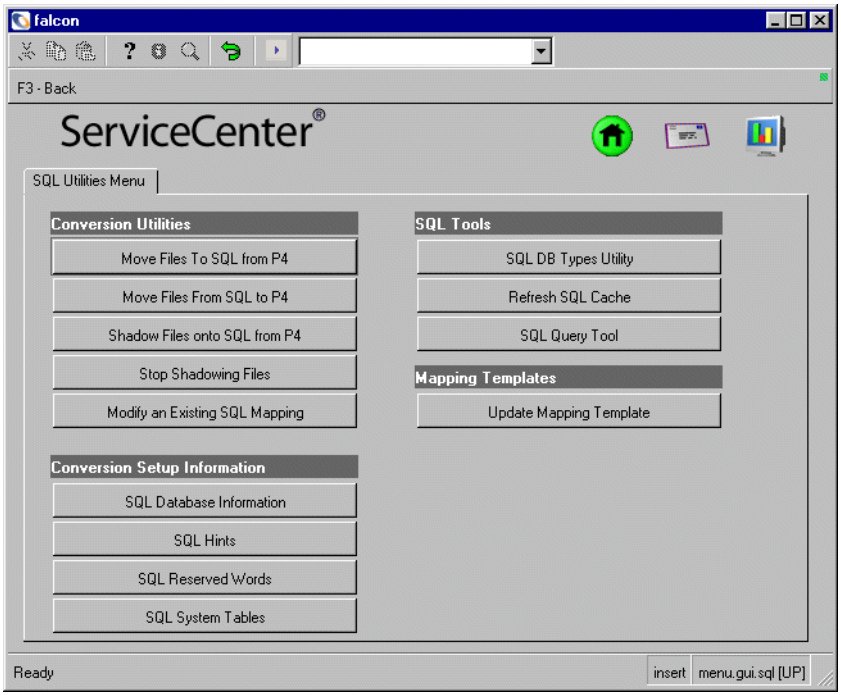


Figure 5-5: The SQL Utilities Menu

To begin conversion:

1 Open the **SQL Menu**. (See *To open the SQL Menu:* on page 170.) Click **Move Files to SQL from P4**. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

The P4 to SQL Conversion Console window is displayed. (Figure 5-6 on page 172)

2 Perform one of two types of conversion. You can choose to convert multiple files, or express additional control over the conversion of each individual file to the RDBMS by using the single file conversion option.

To perform a multiple file conversion, follow the steps in *Multi-file Conversion* on page 172.

To perform a single file conversion or DDL manipulation, follow the steps in *Single File Conversion* on page 176.

Multi-file Conversion

By default the system will begin the process for multiple file conversion. To work with all files, performing operations such as importing or exporting DDL, reviewing and changing mapping tables, and creating tables without exporting data, see *Single File Conversion* on page 176.

To perform a multiple file conversion:

1 Select a Target SQL Database Type (e.g., DB2MVS, Sybase, Oracle) on the **Basic Options** tab. If you are using an out-of-box mapping, select one of the prepared mappings, which are indicated by the letter "x" at the end of the database type name. For more information, see *Out-of-Box Mapping* on page 72.

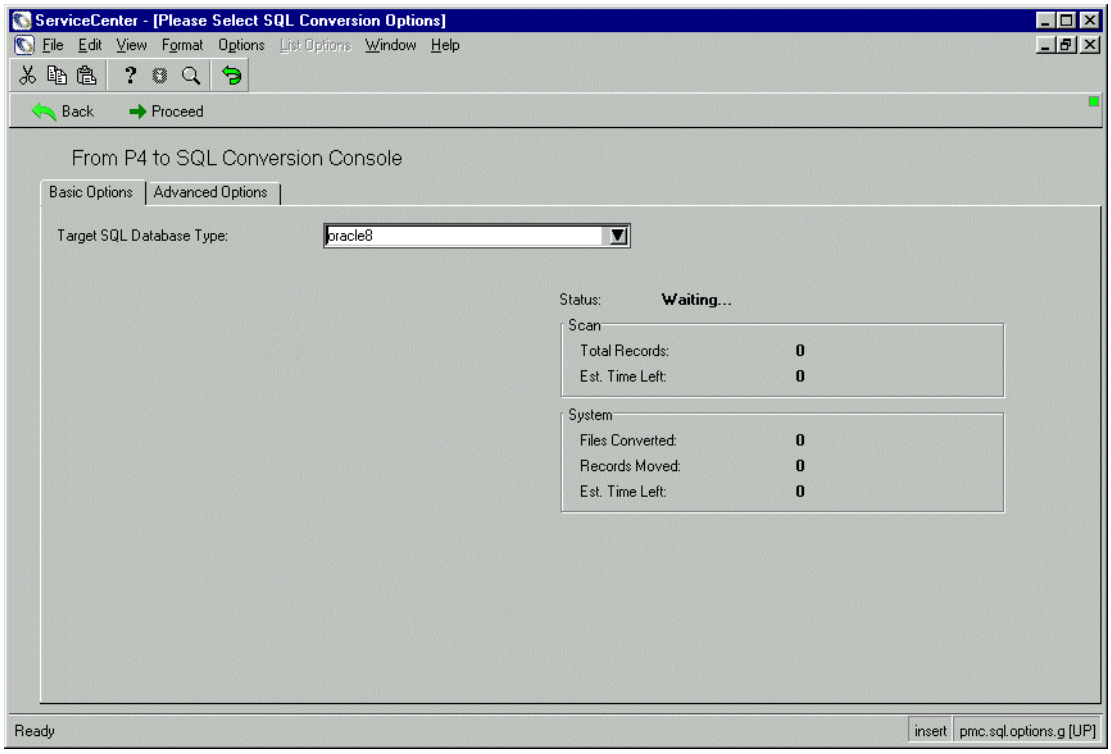


Figure 5-6: Selecting a Destination Database

2 Select the **Advanced Options** tab for additional control options.

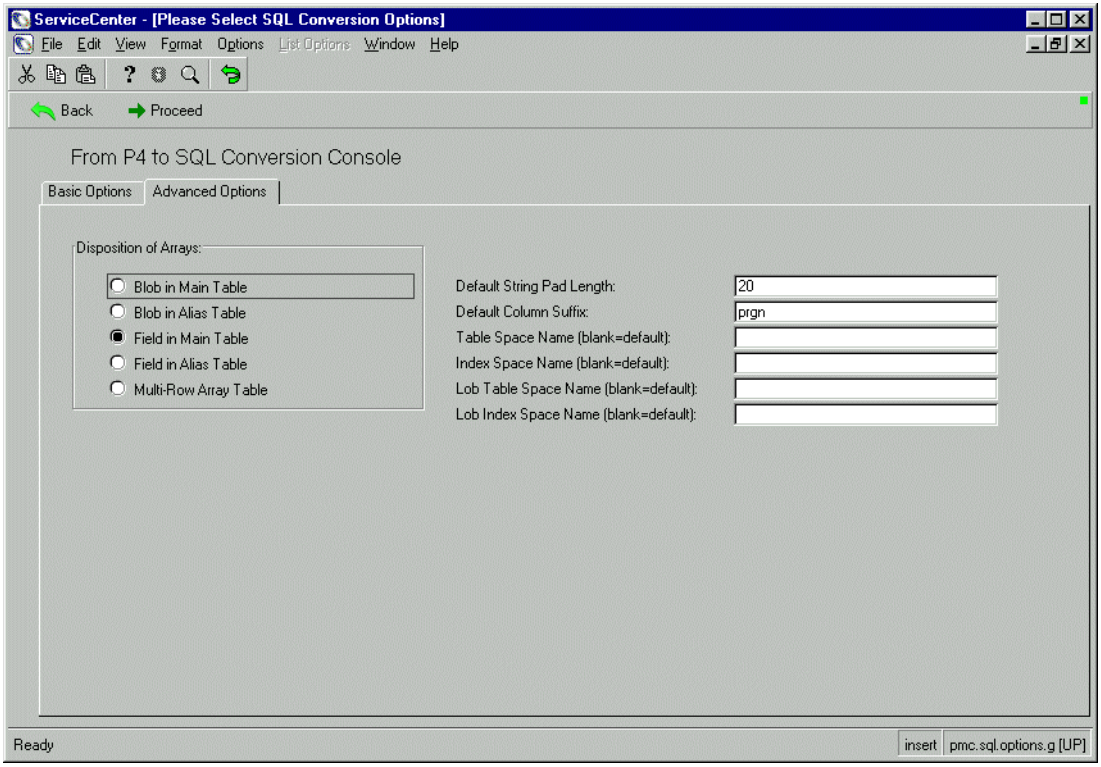


Figure 5-7: Advanced Conversion Options

Important: Peregrine recommends using the default options. Contact your DBA before choosing any options other than the default.

3 Select one of the following options from the Disposition of Arrays structure. The default is **Field in Main Table**. (See Table 5-1 on page 175 for a listing of the fields and their definitions.)

4 Select a **Default String Pad Length**. This value is added, as a safety measure, to the formula determining the length of a **char** or **varchar** field on the SQL server.

5 Define a **Default Column Suffix**.The system attaches this unique suffix to field names in your P4 file system, avoiding conflict with reserved words used by the RDBMS you have selected.

6 In most cases, leave the **Table Space Name** field blank. Enter a name in the **Table Space Name** field if you want to change the name of the table space you have defined for your RDBMS. If you leave the field blank, the system will default to the previously defined name. (DB2 uses the default database name, PRGNDB, unless you enter the value for another tablespace in this field.)

Note: It is possible to point indexes and data to different tablespaces during the conversion process. If the **tablespace** option in the Advance SQL option is not available, one way is to set the default tablespace for the RDBMS user who would be responsible for the conversion, to the desired tablespace. This should be done on the RDBMS side before the conversion takes place.

7 In most cases, leave the **Index Space Name** field blank. The RDBMS uses the default index space associated with the login user ID on that RDBMS unless you enter the value for another index space in this field. (DB2 uses the default database name, PRGNDB, unless you enter the value for another index space in this field.)

Note: All messages generated during the conversion are written to the **sc.log**.

8 Click **Proceed** or press Enter.

The system will warn you that it is about to move your data to the RDBMS database and that the conversion is a lengthy process.

9 Click **Yes** to proceed with the conversion.

If converting to DB2, you will receive the message *Peregrine strongly recommends... you click "yes."*

The conversion application begins scanning the local file base, preparing the files for conversion.

Note: ServiceCenter uses a DLL to start the conversion to the appropriate RDBMS. The DLL used is based on the RDBMS you have selected. See *Windows Dynamic RDBMS Conversion DLLs* on page 101 for more information.

10 Refer to the **Basic Options** tab to see the approximate duration and dynamically updated status of the conversion, as seen in Figure 5-8 on page 175.

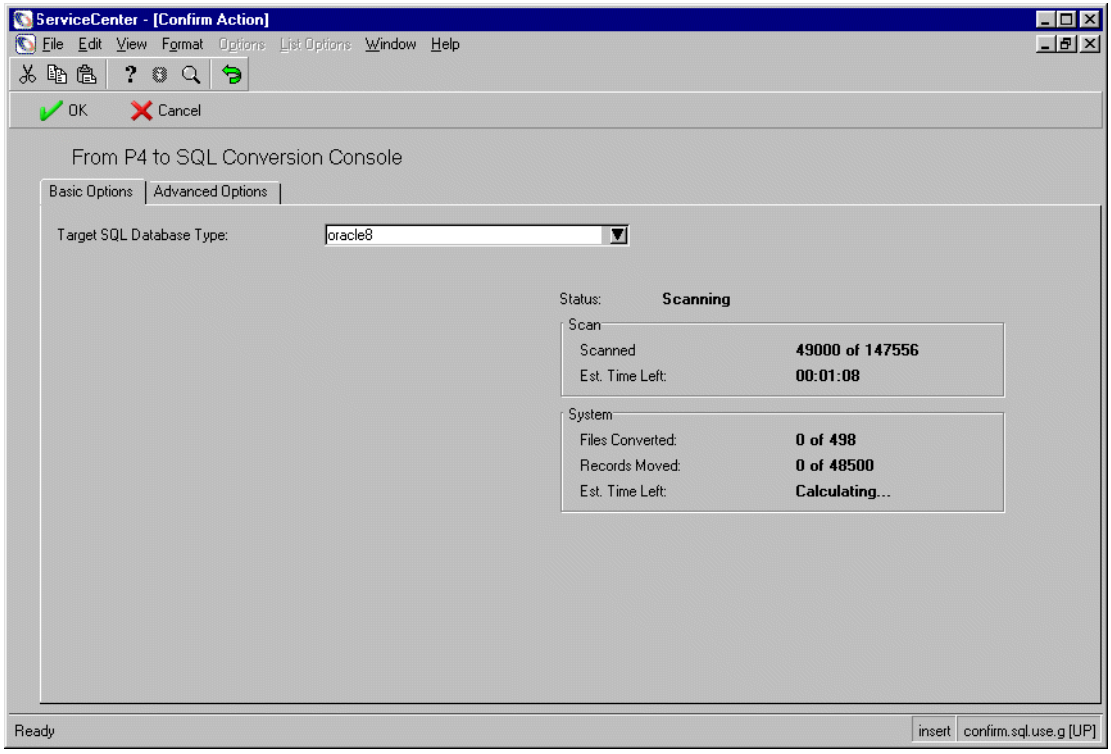


Figure 5-8: Conversion Status Posted to Basic Options Tab

Note: The times calculated here are approximate. Conversion speed will vary based upon your particular server hardware, software, and network configuration.

Table 5-1: Disposition of Arrays Panel

Field	Definition
Blob in Main Table	Translates the ServiceCenter array of characters into a single stream of binary data in an internal ServiceCenter format. This *binary large object* is then stored in the ServiceCenter record's main SQL table as if it were a simple data type.
Blob in Alias Table	Translates arrays of characters into a single stream of long, binary data, but stores each binary stream in its own alias table on the server. Each array in the schedule file will be stored in a new alias table in SQL.

Table 5-1: Disposition of Arrays Panel

Field	Definition
Field in Main Table	Takes the ServiceCenter array and translates it into a single long text string, separating each line with the new line character. For example, an array of {“a”, “b”, “c”} become a single string reading “a\nb\nc”.
Field in Alias Table	Translates arrays of characters into single strings of long text. The mapping strategy stores each string of long text in its own alias table on the server. Each array in the schedule will be stored in a new alias table in SQL.
Multi–Row Array Table	Creates a separate alias table for each array in the ServiceCenter record in which an element of the array is given its own row. Each array in the schedule will be stored in a new alias table in SQL. Each field in this array or structured array is stored as a separate column in this new SQL alias table.

Single File Conversion

For those with more than one database, the single file conversion option can be used to map individual files to a particular database. It also allows you to express more control over the mapping of each file.

To map individual files to a particular database:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click **Move Files to SQL from P4**. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

The pmc.sqloptions.g form is displayed.

3 From the ServiceCenter Options menu, select **Single File**. Control options appear on the **Basic Options** tab.

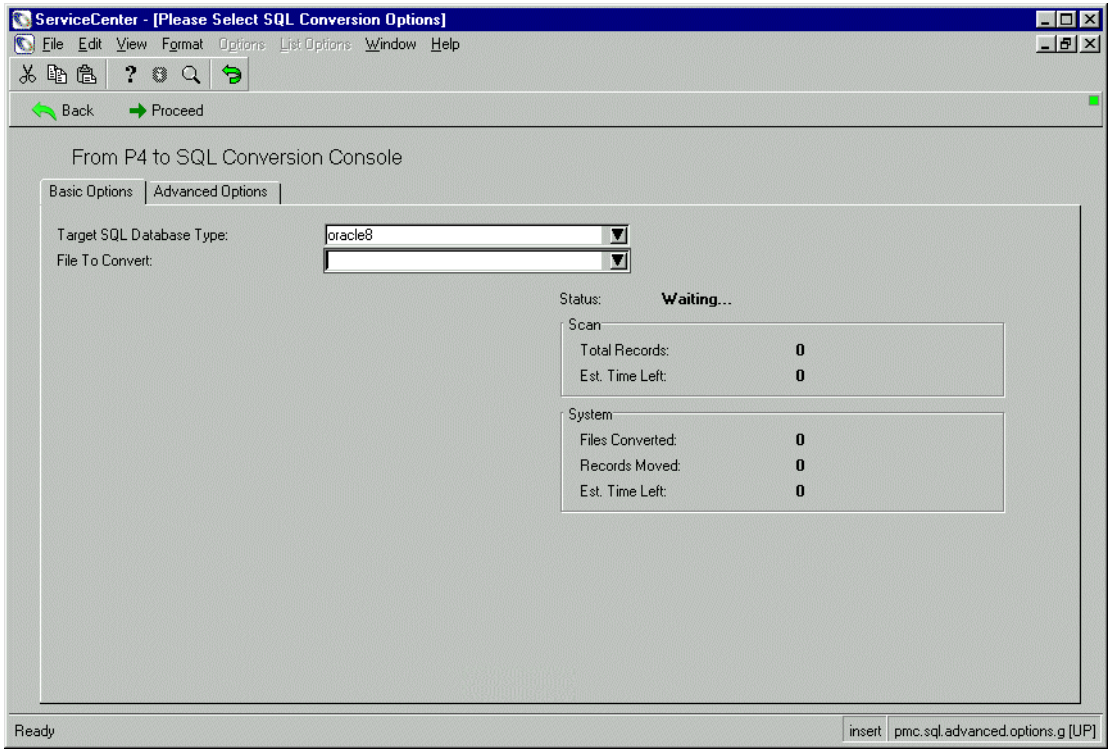


Figure 5-9: Selecting a Destination Database and Local File for Conversion

4 Select a database type using the drop-down list in the **Target SQL Database Type** combo box. If you are using an out-of-box mapping, select one of the prepared mappings, which are indicated by the letter "x" at the end of the database type name. For more information, see *Out-of-Box Mapping* on page 72.

5 Select a file to convert using the drop-down list in the **File to Convert** combo box.

6 Select the **Advanced Options** tab for additional control options.

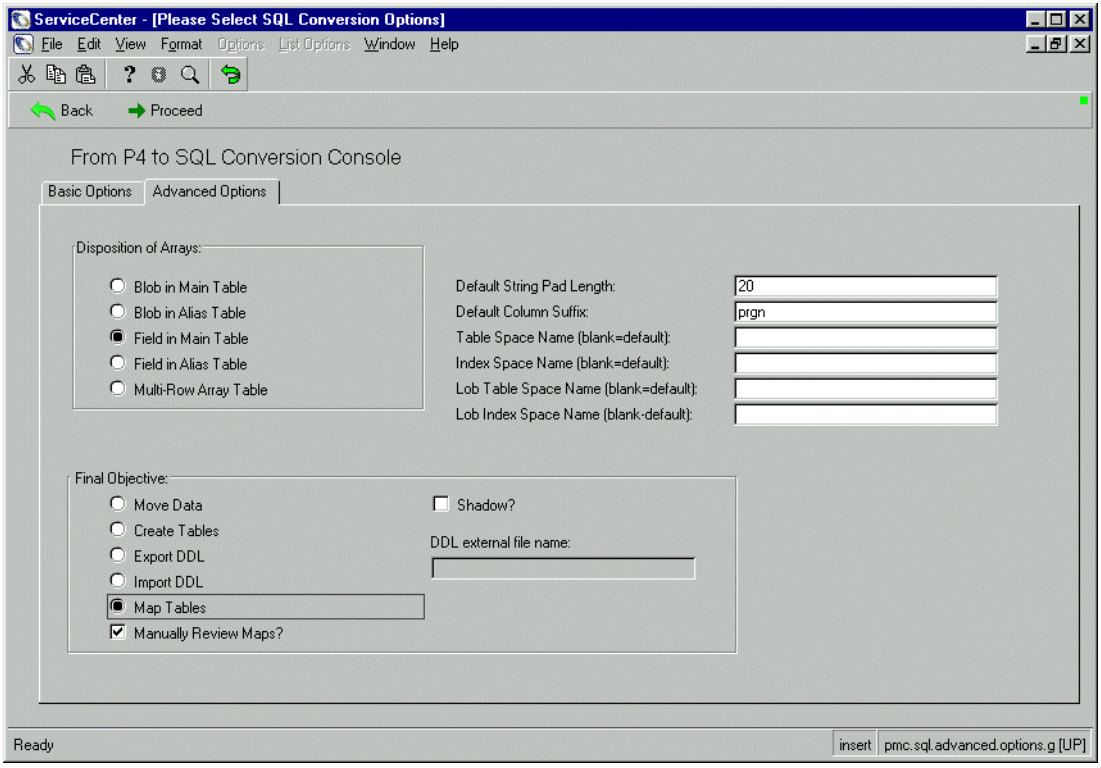


Figure 5-10: Advanced Conversion Options

For details on the **Disposition of Arrays** structure and for the **Default String Pad Length, Default Column Suffix, Table Space Name** and **Index Space Name,** see Table 5-1 on page 175.

7 Make the appropriate selection from the Final Objective structure.

- Move Data — The default, used if you wish to continue with a single file conversion. If a P4 table is to be mapped for the first time to a table that already exists in another RDBMS, Move Data must be selected. Press Proceed or press Enter to continue.
- Shadow — Allows you to select whether you want the conversion to be shadowed, or populated by fully moving the data into the RDBMS.
- You cannot convert a file that is shadowed. You must set the Shadow field to false (deselect it), then convert. See *Shadowing P4* on page 235 for more information on activating and deactivating database shadowing.

- Create Tables — Allows the design of data tables, but does not move data. This option works with the original tables in P4. If a P4 table is to be mapped for the first time to a table that already exists in another RDBMS, Move Data must be selected.
- Export DDL — Allows the customization of table controls and appearance by opening the DDL to the user in a specified file before the conversion. See *Exporting DDL* on page 180 for more information.
- Import DDL — Allows a local DDL file to be read back into ServiceCenter internal data. See *Importing DDL* on page 180 for more information.

8 Set the **Manually Review Maps?** Set the flag to *true* to view the data map. Altering the data map is not usually necessary and is not recommended.

9 Click Proceed or press Enter. To process the conversion for the selected file.

Note: If you selected *All Files*, the system warns you that it is about to move your data to the SQL Database and that the conversion is a lengthy process.

10 Click **Yes** to proceed with the conversion.

A prompt is displayed advising you to place long text or binary fields in alias tables prior to the conversion.

11 Click **Yes** to proceed with the conversion.

The conversion application begins scanning the local file base, preparing the file(s) for conversion.

12 Refer to the **Basic Options** tab to see the approximate duration and dynamically updated status of the conversion, and the status of files as converted. (Figure 5-8 on page 175)

Note: The times calculated here are approximate. Conversion speed will vary based upon the server's hardware, software, and network configuration.

13 After converting a file, either select another file to convert, or change the Target SQL Database Type and begin converting files on another database.

14 Repeat this process until all files are placed on the appropriate database.

Note: It is not necessary to use **Back** when switching Target types. Change the value in the **Target SQL Database Type** field to use a different database type.

Exporting DDL

If a file name is not provided while exporting a DDL, the system specifies P4TOSQL.DDL as a non-qualified name.

Note: Peregrine recommends that you rename the file with appropriate first level qualifiers.

A table and an index file are generated with a *T* or *I* respectively appended to the file name, e.g., P4TOSQL.DDL. Providing two files allows the user to save time by first importing or creating the tables, then moving the data, and finally creating the indexes.

Note: Within the index file, the DBA may change the */*Iconstraint*/* phrase (for DB2, to an appropriate index constraint e.g., SUBPAGES, BUFFERPOOL, etc.) or to perform special work during the conversion. You can search/replace this section with in the file. The same is true for the */*Tconstraints*/* section of the file (for DB2, to an appropriate table constraint, e.g. EDITPROC, VALIDPROC, etc.)

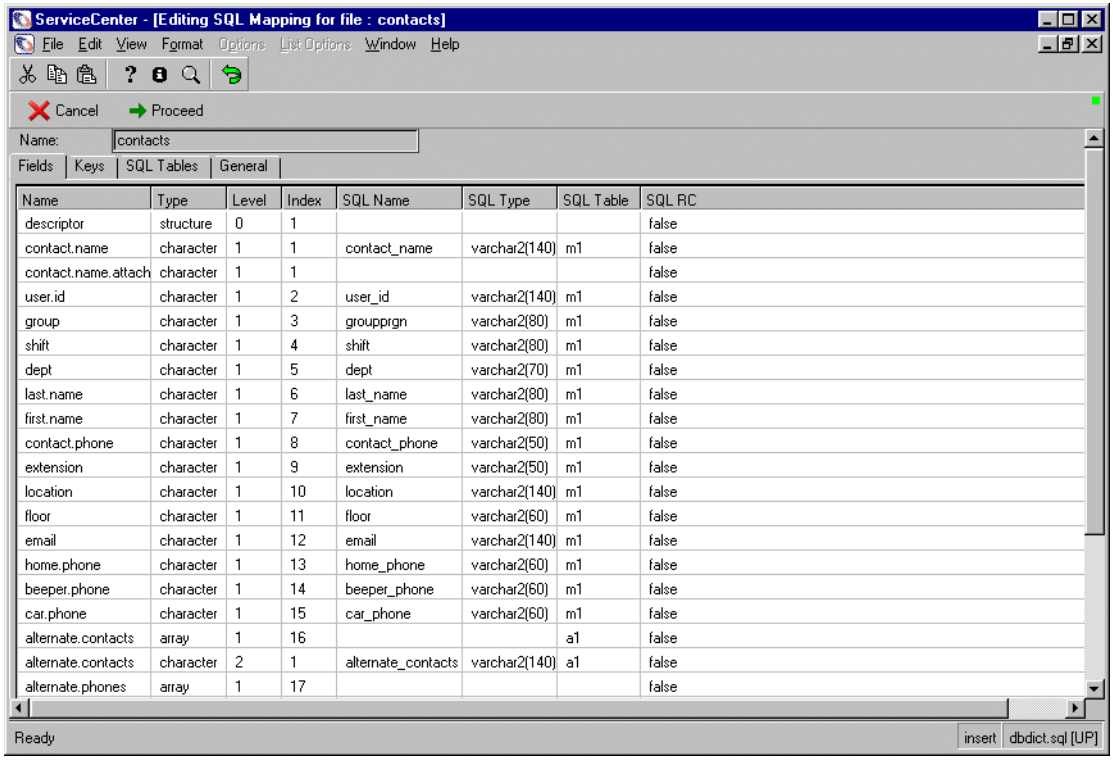
Importing DDL

When importing the DDL, use the base name without the *T* or *I*. It is important when entering the base name not to change any value in the name.

When changing a DDL, the table must be imported into ServiceCenter or the mapping manually edited to match the data structures on both systems.

Editing SQL Mapping for a File

In Figure 5-11 on page 181, each field is displayed along with the table name, SQL Name and SQL Type.



Name	Type	Level	Index	SQL Name	SQL Type	SQL Table	SQL RC
descriptor	structure	0	1				false
contact.name	character	1	1	contact_name	varchar2(140)	m1	false
contact.name.attach	character	1	1				false
user.id	character	1	2	user_id	varchar2(140)	m1	false
group	character	1	3	groupprgn	varchar2(80)	m1	false
shift	character	1	4	shift	varchar2(80)	m1	false
dept	character	1	5	dept	varchar2(70)	m1	false
last.name	character	1	6	last_name	varchar2(80)	m1	false
first.name	character	1	7	first_name	varchar2(80)	m1	false
contact.phone	character	1	8	contact_phone	varchar2(50)	m1	false
extension	character	1	9	extension	varchar2(50)	m1	false
location	character	1	10	location	varchar2(140)	m1	false
floor	character	1	11	floor	varchar2(60)	m1	false
email	character	1	12	email	varchar2(140)	m1	false
home.phone	character	1	13	home_phone	varchar2(60)	m1	false
beeper.phone	character	1	14	beeper_phone	varchar2(60)	m1	false
car.phone	character	1	15	car_phone	varchar2(60)	m1	false
alternate.contacts	array	1	16			a1	false
alternate.contacts	character	2	1	alternate_contacts	varchar2(140)	a1	false
alternate.phones	array	1	17				false

Figure 5-11: Editing SQL Mapping for File

Table 5-2: Field Descriptions for the SQL Mapping File

Field	Descriptions
contact.name	name of field in dbdict resources within ServiceCenter.
M1	the name of the table this field will reside in the RDBMS. The m1 is appended onto the name of resources. This becomes important when using array fields in multi-tables. Each array field will be mapped to a different A table.
contact_name	name of the field in this table in the RDBMS.
Varchar2(140)	type of the field in the RDBMS will be varchar2 and the length will be 140.

Supply the DBA with counts of total fields from each type/length.

For this example, the following information will be supplied:

5 fields with varchar4(140)
4 fields with varchar2(80)
1 field with varchar2(70)
2 fields with varchar2(50)
4 fields with varchar2(60)

By supplying this information, average row length can be determined. For each field that is used in each index, make note of the length/type in the RDBMS. This information should be given to the DBA to answer length of fields in indexes.

Table Creation Displays

Clicking on the **SQL Tables** tab displays the tables that will be created.

This example shows that one main and four array tables will be created. Total up each table that has an alias starting with "a". You will then have a count of the RDBMS tables created from arrays. Since the conversion converts all tables associated with a table (array/main) at the same time, consideration should be made when calculating initial/next extents for the tables depending on how many array tables are present. Sacrificing for optimum sizes on main table should be done so less space is wasted on array tables. The information also shows what tablespace in the RDBMS that this table resides in.

This process needs to be done for all tables that are going into a single tablespace. By combining all the totals, the DBA can create the tablespace for the tables and indexes. Also, the DBA can provide the initial/next information for the table to get converted.

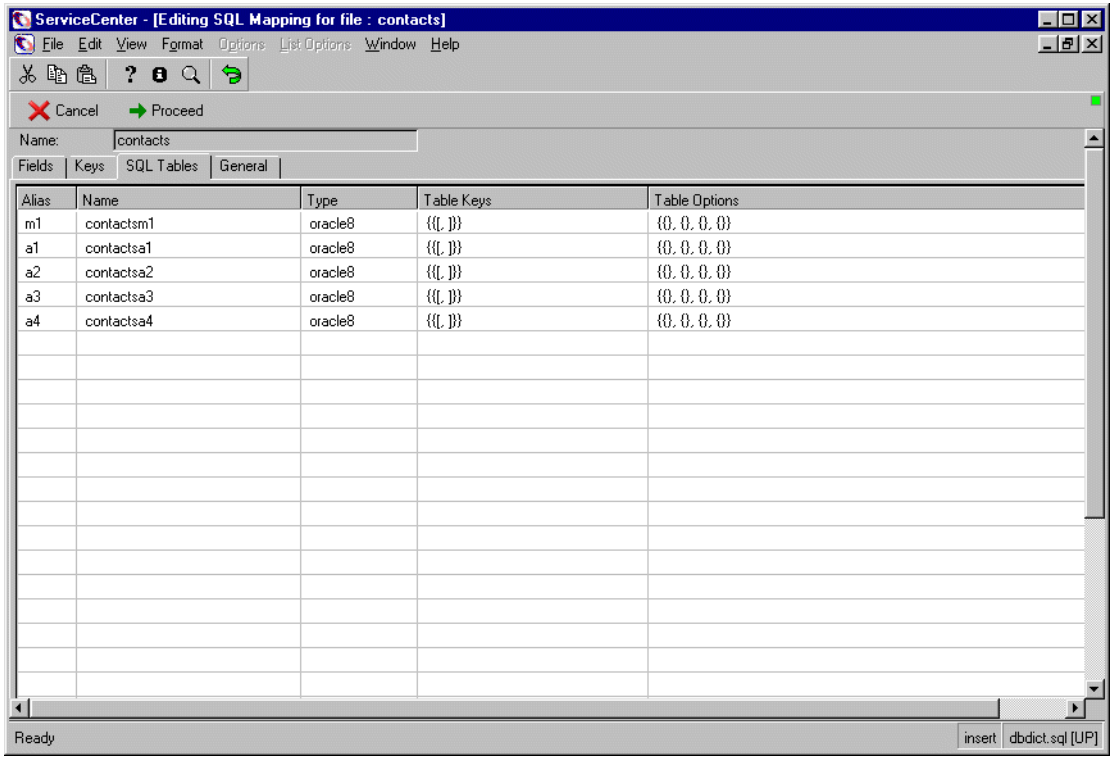


Figure 5-12: Table Creation Displays

Associating a ServiceCenter file with a pre-existing table in an RDBMS

To associate a ServiceCenter files with a pre-existing table:

1. Set up ServiceCenter to have appropriate access to the desired database, as if you were going to map ServiceCenter tables out to that database.
2. In the **sc.ini** file, set **sqldetect:1** and **sqldrop:0**
3. Start up only the express listener on ServiceCenter.
4. Log in as an express client, using an operator with SysAdmin capability.
5. Go to the Database Dictionary utility, and create the **dbdict** that will be used to attach ServiceCenter to the table. You can use any name; it should not contain special characters, etc. EXAMPLE: amAsset. See *System Tailoring* for more information on the Database Dictionary Utility.

a Add one field to this dbdict; you should try to pick a unique key from the RDBMS. EXAMPLE: AssetTag.

b Create a unique key on the dbdict using the field you added.

c Save the dbdict.

6 Go to the **SQL Utilities** tab and click **Move Files to SQL from P4**.

7 Open the **Options** menu, and choose **Single File mode**.

8 On the **Basic Options** tab, select the file you just created. EXAMPLE: amAsset.

9 On the **Advanced Options** tab, under **Final Objective**, choose the Map Tables radio button, and check the Manually Review Maps? checkbox.

10 Click **Proceed** in the tooltray.

11 When the map is displayed, select the **SQL Tables** tab, and modify the table name to point to the existing table in the SQL database.

EXAMPLE: amAssetm1 should be changed to amAsset

12 Click the **Proceed** button in the tooltray; a pop-up will tell you the mapping is completed.

13 From Database Manager, select the dbdict file.

14 Select the dbdict format from the QBE list.

15 Type the name of your dbdict into the Name field, and click **Search**.

16 Change the value of the Root Record to -1, and click **Save**.

17 It takes awhile, but the dbdict will be updated.

18 Click **Cancel** and then redisplay the dbdict.

All the fields have been added to the dbdict, and you can now access this existing table via ServiceCenter.

Monitoring the Conversion Process

As the system converts your files, the status is dynamically updated on the **Basic Options** tab. The current step in the conversion is also noted on this tab. The status of each file as it enters the conversion process is displayed. Overall system status is also shown.

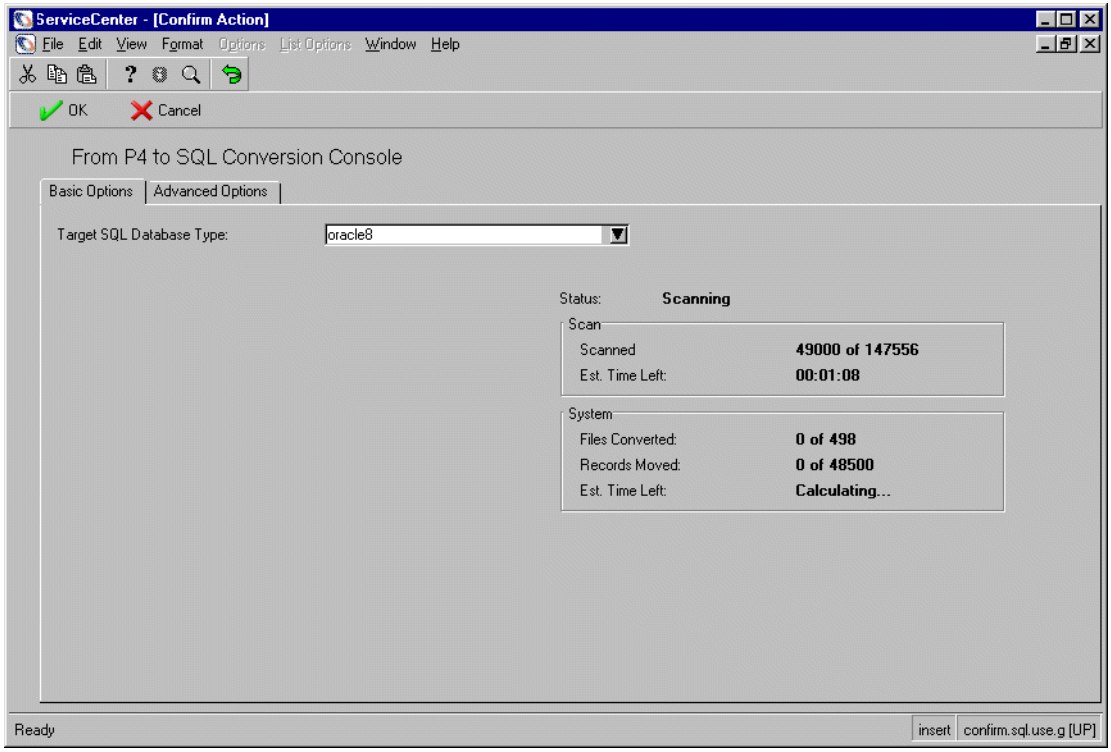


Figure 5-13: Conversion Status Posted to Basic Options Tab.

Conversion status messages are also output into the standard system log file, sc.log as defined in the sc.ini file.

Log File

The standard ServiceCenter log file, sc.log, is used during the conversion to post any messages, including progress, relevant to the process. Entries are added dynamically during the conversion process. Use any text editor to periodically check the log.

Conversion Log Entries

The conversion process creates entries in the ServiceCenter sc.log that display each step of the conversion. If you encounter any problems converting your files, refer to this log to determine where the process broke down.

To access the log:

1 Open a file management utility.

2 Open the directory in which you have installed ServiceCenter.

3 Open the sc.log file on the server.

4 The log entries are displayed, showing each step of the process.

```
142  02/19/98 11:09:29: Initiating a Conversion of resolution
142  02/19/98 11:09:29: Mapping tables resolution
142  02/19/98 11:09:29: Creating SQL Tables for resolution
142  02/19/98 11:09:30: Completed Move of 11 records
142  02/19/98 11:09:31: Completed Conversion of resolution
142  02/19/98 11:09:36: Initiating a Conversion of router
142  02/19/98 11:09:36: Mapping tables router
142  02/19/98 11:09:37: Creating SQL Tables for router
142  02/19/98 11:09:38: Completed Conversion of router
142  02/19/98 11:09:43: Initiating a Conversion of sc
142  02/19/98 11:09:43: Mapping tables sc
142  02/19/98 11:09:44: Creating SQL Tables for sc
142  02/19/98 11:09:45: Completed Conversion of sc
142  02/19/98 11:09:51: Initiating a Conversion of scautotest
142  02/19/98 11:09:51: Mapping tables scautotest
142  02/19/98 11:09:52: Creating SQL Tables for scautotest
142  02/19/98 11:09:52: Completed Move of 1 records
142  02/19/98 11:09:53: Completed Conversion of scautotest
142  02/19/98 11:09:58: Initiating a Conversion of scdsites
142  02/19/98 11:09:58: Mapping tables scdsites
142  02/19/98 11:09:59: Creating SQL Tables for scdsites
142  02/19/98 11:10:00: Completed Conversion of scdsites
142  02/19/98 11:10:05: Initiating a Conversion of schedule
142  02/19/98 11:10:05: Mapping tables schedule
142  02/19/98 11:10:07: Creating SQL Tables for schedule
142  02/19/98 11:10:11: schedule: Moved 100 records
142  02/19/98 11:10:13: Completed Move of 156 records
142  02/19/98 11:10:13: Completed Conversion of schedule
142  02/19/98 11:10:19: Initiating a Conversion of scparms
```

5 Track the progress of the conversion in the sc.log file, execute the following command on the log file:

```
tail –f ../logs/sc.log
```

Testing for Completion

When the conversion is completed, the system returns control of the client. Use these methods to test for completion:

- Open the sc.log file and check for a completion message. See *Log File* on page 185, for more information.
- Display the list of records that were not converted. See *Root Record* on page 187 for more information.
- View the status data on the SQL conversion console. See *Monitoring the Conversion Process* on page 185 for more information.

Root Record

To view files that have not been converted:

1 Open the dbdict form in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

An empty dbdict form is displayed.

2 Select **Advanced Search** from the Options menu.

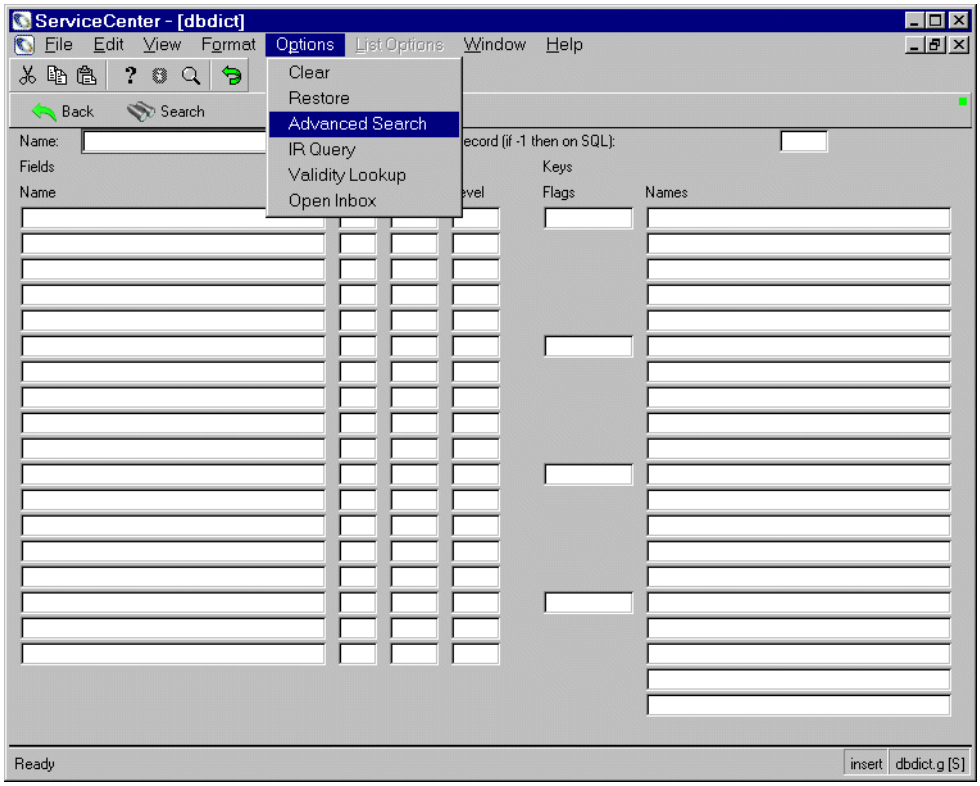


Figure 5-14: Selecting Advanced Search

3 Enter root.record~=-1 in the **Query** field of the Search Window.

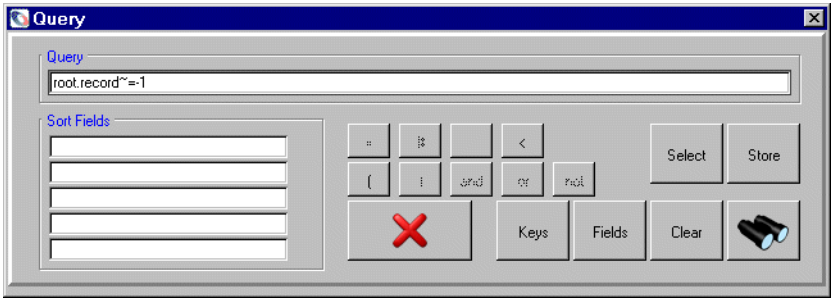


Figure 5-15: Query Window

4 Click **Search** or press Enter.

5 If a search screen is displayed, enter any desired criteria, and click **Search**.

A screen is displayed with a record list of files that have not been converted to the RDBMS.

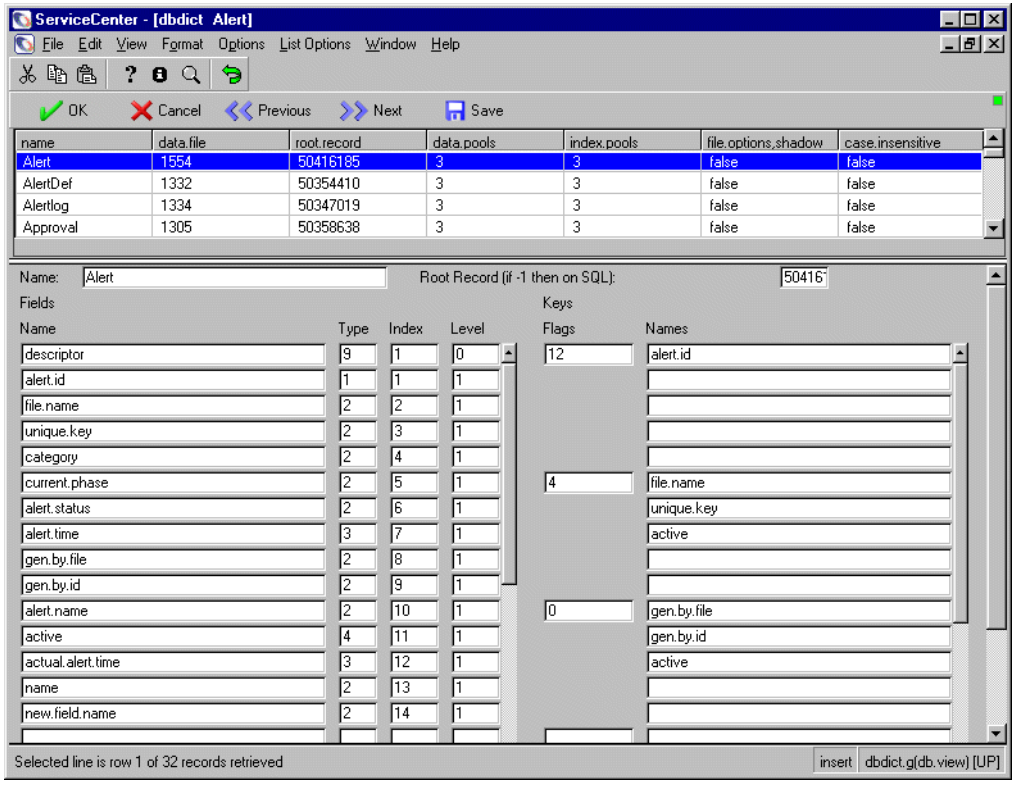


Figure 5-16: Files Not Converted

Automated Conversion Cleanup

IR files are automatically regenerated in the Database Dictionary (dbdict) following the conversion process. As shipped, these include cm3r, core, probcause, probsummary, knowledge, incidents, helptext, KnowledgePak and SLA.

Note: Regeneration of these files occurs for all conversions on all platforms.

Oracle Schema Manager Verification

Within Oracle DBA Studio, a check can be made on the new table as well as indexes. See your oracle documentation for more details.

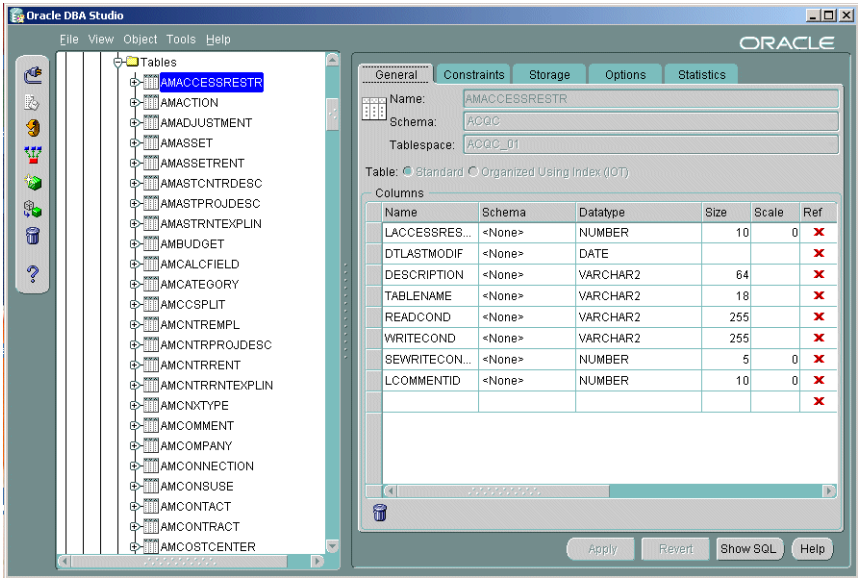


Figure 5-17: Oracle Schema Manager Table Verification

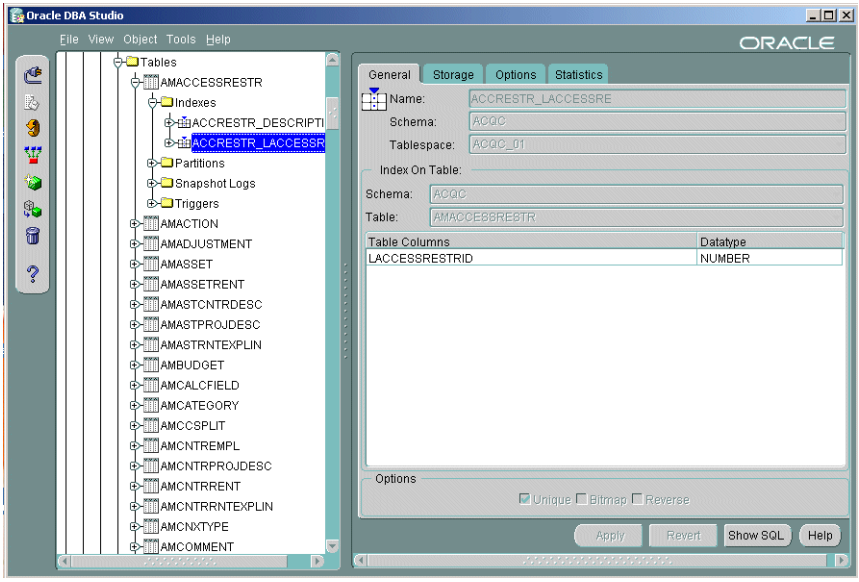


Figure 5-18: Oracle Schema Manager Index Verification

6 Post Conversion

CHAPTER

This chapter was designed to explain the procedures for editing a database after conversion to a Relational Database Management System (RDBMS) format to ServiceCenter system and database administrators.

Topics in this chapter include:

Overview

Adding columns to RDBMS tables and editing data maps are not recommended in most cases, but may be necessary when tailoring a system previously converted to an RDBMS format. Controls have been added to ServiceCenter to facilitate mapping P4 fields to RDBMS columns.

Warning: This procedure is considered an *expert option* and should be undertaken by a qualified database administrator (DBA) only.

For detailed information comparing the database structures of compatible RDBMS formats with that of ServiceCenter's P4 file system, please refer to the *RDBMS Support Overview* on page 15.

Note: After all tables are converted to the RDBMS, the DBA should watch the RDBMS to see if any waits are occurring. If waits are happening, additional rollbacks can be added slowly as checks are being done to ensure the waits are decreasing.

Operator Capability

To grant a user the right to add fields to SQL mapped Database Dictionary files, add the capability word **SQLAdmin** to the user's operator record.

Editing Tables in ServiceCenter

ServiceCenter provides button access to the mapping utility, allowing users to display and edit *single* data maps of files while they reside in the RDBMS format. Updating maps in ServiceCenter does not involve single file conversions.

There are two methods of adding fields to a converted system:

- *Manual Editing* on page 195
- *Process Editing* on page 205

Manual Editing

Overview steps for adding fields manually:

Step 1 Add a column to the RDBMS table.

Step 2 Add the new field to the ServiceCenter Database Dictionary.

Step 3 Map the ServiceCenter field name to the RDBMS column name.

Adding a Column to the RDBMS Table

The first step in the process of editing a converted file is to add a column to your RDBMS table. Refer to the documentation provided with your database format for specific procedures for adding new columns.

Warning: This procedure is considered an *expert option* and should be undertaken by a qualified DBA only.

Adding Fields to the Database Dictionary

In order for ServiceCenter to recognize modified RDBMS tables, the Database Dictionary must be edited also. Be sure the data types (e.g., number, character, date/time, etc.) you are adding to the dbdict match those you have added to the RDBMS table.

To edit a Database Dictionary:

1 Open the Database Dictionary Utility, and enter the name of the existing dbdict. For the example, enter alert. (For instructions, see Volume 2 of Database Management and Administration.)

The dbdict record for your file is displayed.

2 Place the cursor in the descriptor field and click the **New** button.

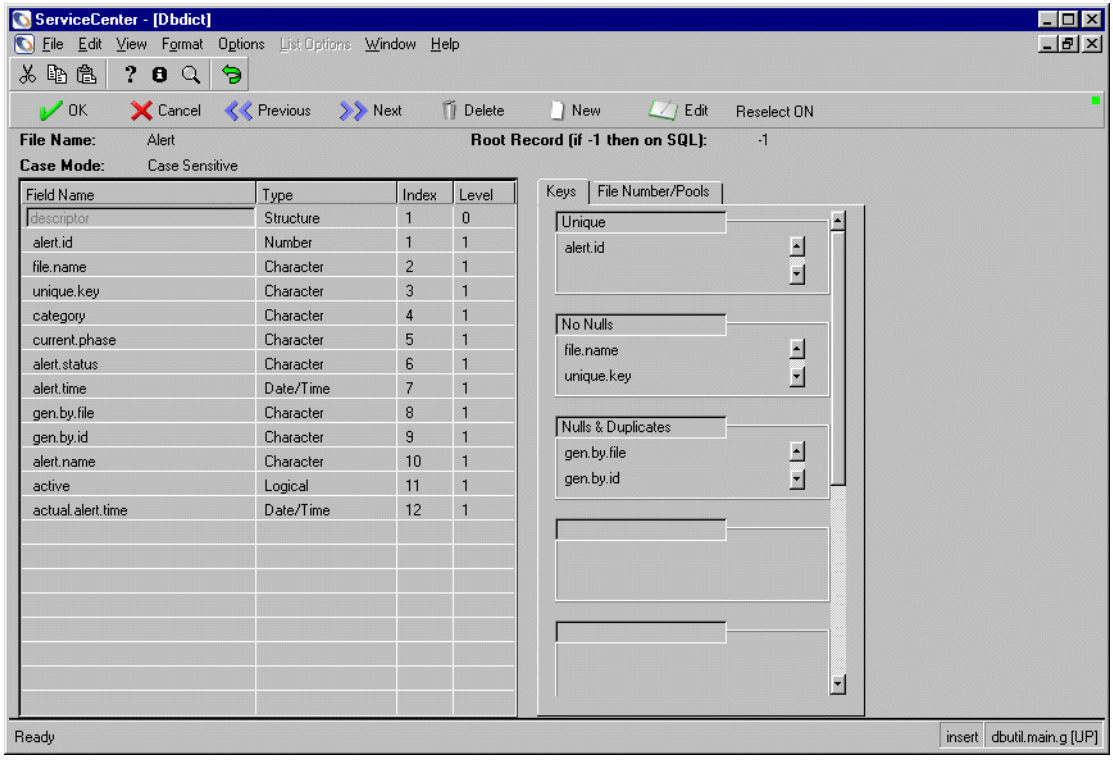


Figure 6-1: Database Dictionary of a Converted File

3 Enter the name of the new field and the data type in the Database Dictionary editor.

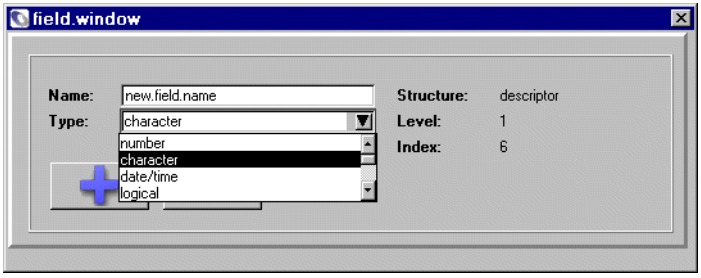


Figure 6-2: Adding a Field with the Database Dictionary Editor

4 Click the **Add** button (+) to add the field to the Database Dictionary record.

5 If the Root Record is **-1**, then a **Reselect ON/OFF** toggle button is displayed at the top of the Database Dictionary file. Choose a reselect option by clicking the **Reselect ON/OFF** button. (The root record will only be -1 if the file has been converted already.) If the Root Record is a number other than **-1**, then the **Reselect ON/OFF** button does not appear on the record.

Note: The **Reselect ON/OFF** button does NOT reflect the current state, but the action you will take when you click the button. For example, if the button says **ON**, the current state is **OFF**

If you have TRIGGERS that modify the data outside of ServiceCenter control, then **Reselect** should be **ON**. This will force ServiceCenter to reselect the record after an update or insert so that the system picks up the changes made by the TRIGGER. If you do not have TRIGGERS, then **Reselect** should be **OFF** to avoid unnecessary re-fetching of data from the RDBMS.

When the **Reselect ON/OFF** button is toggled **OFF**, a prompt is displayed, asking you to refresh your cache. The cache is refreshed by deleting the data currently in the CACHE. The next time the file is accessed, ServiceCenter will go back to the RDBMS to get the current SQL mapping for the file.

Note: Each time you update a record in ServiceCenter, the system checks to see if its copy of that record is current. If the record is not current, then the system will retrieve the latest copy, and you will have to make your changes again. Normally this does not occur. However, if you have established RDBMS trigger functions that change the record during updates or inserts, then ServiceCenter's copy will no longer be current.

6 Click **OK** to close out of the Database Dictionary record.

The **dbdict.alter.g** form is displayed after adding a field to the Database Dictionary record of a file that has been converted to an RDBMS format.

7 Follow the instructions on the screen and select the appropriate method to apply the changes to the RDBMS table.

8 Select **SC Alters** to allow ServiceCenter to update the SQL tables, or select **User Alters** to make the modifications yourself outside of the system and reapply them to the RDBMS. The ALTER statement tells the RDBMS about the alteration.

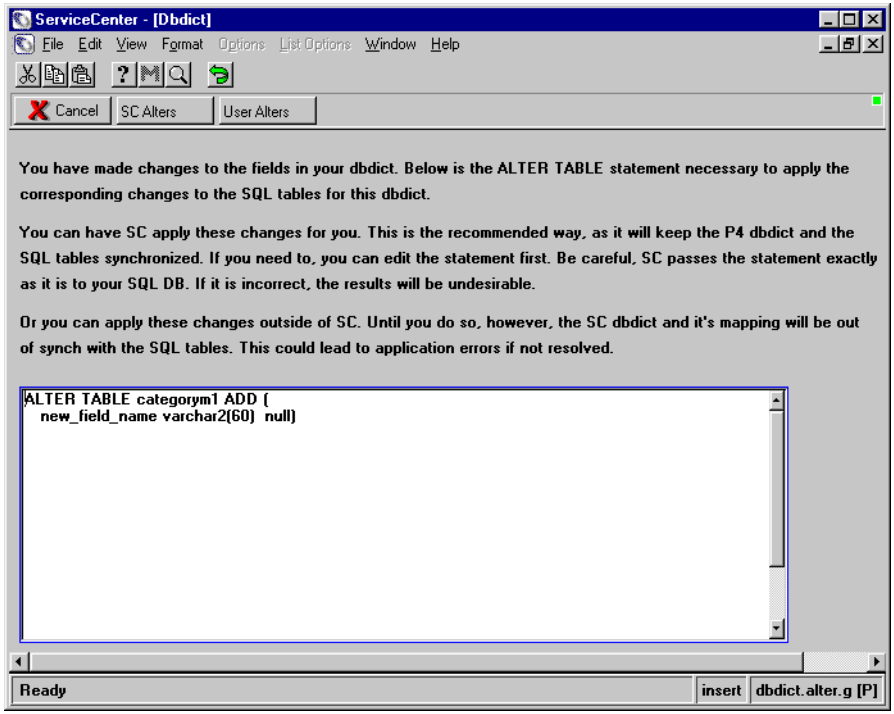


Figure 6-3: The Alter table

Mapping Field Names

The last step in the process of editing an RDBMS table is to map the ServiceCenter field to the RDBMS column name as it appears in the table.

Field name mappings can be modified in two ways:

- *Modify an Existing SQL Mapping* on page 199 — An Expert option, allowing more details to be specified.
- *SQL Hints* on page 203 — the fast and easy option.

Modify an Existing SQL Mapping

Be sure the data type (e.g., number, character, date/time, etc.) of the field you are mapping matches that of the field you have added to the dbdict and to the RDBMS table.

Warning: This procedure is considered an *expert option* and should be undertaken by a qualified DBA only.

To map field names:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click **Modify an Existing SQL Mapping**.

3 Enter the name of the file you want to map in the **File Name** field.

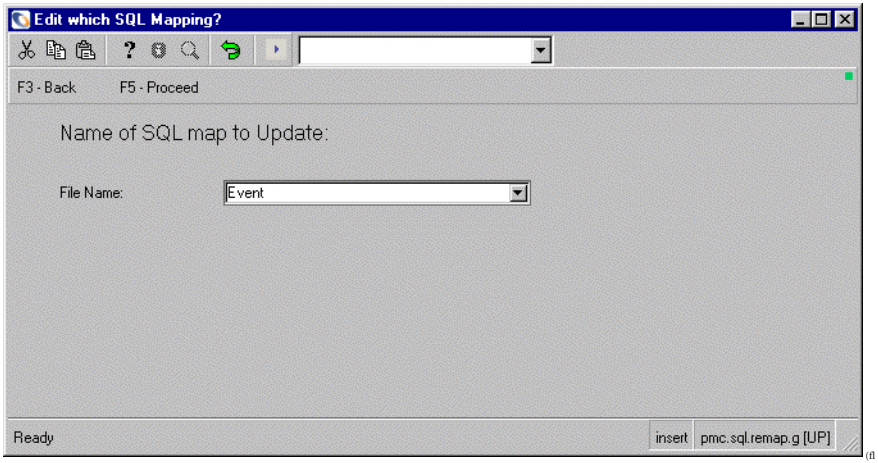
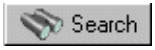


Figure 6-4: Specifying a Map to Edit



4 Click **Search** or press Enter.

The requested data map is displayed.

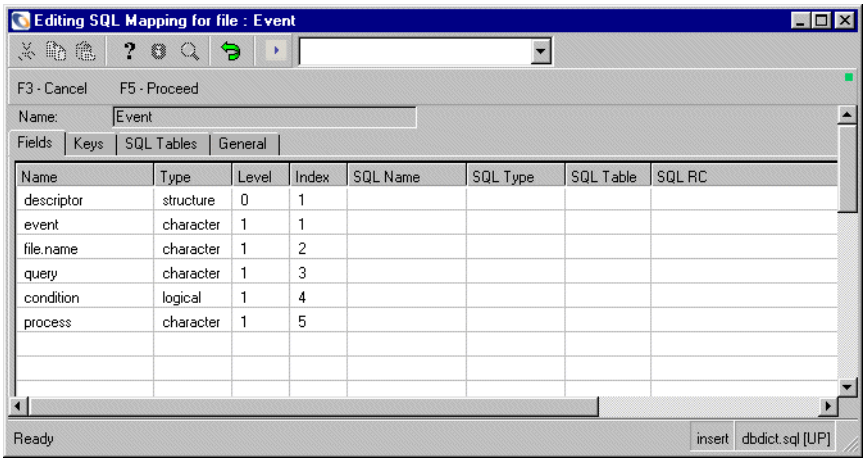


Figure 6-5: Data Map

5 Modify the mapping information for existing fields by clicking in the field and entering the updated values.

Note: New fields may be added through the Database Dictionary application. See *Database Dictionary* in ***System Tailoring*** for more information.
If this file has been converted, all new fields added in the Database Dictionary are automatically added to the field map. If the file has not yet been converted, you can add a field in the Database Dictionary, then remap the file, and the new field will be added without modifying the existing mapping setup for that file.

6 Click **Proceed** when finished modifying the mapping.

A prompt is displayed asking if you want to save the new mapping.

7 Click **Yes** to save your changes.

Another prompt is displayed asking if you want to update the mapping template for the record.

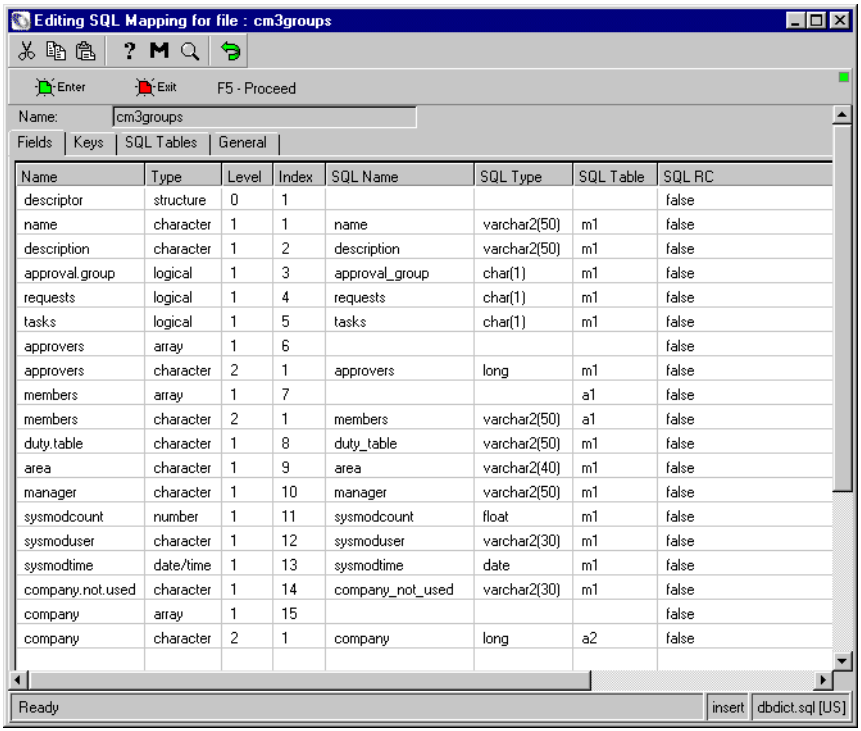
8 Click **Yes** to update the template.

Warning: If you do not make the same changes in the sqlmapping files as you did in the dbdicts, the mappings will be lost when the data is moved back to P4. The sqlmapping files will not contain the those changes, and they will not be available if you then remap to SQL at some point in the future.

Another prompt is displayed informing you that your changes were made.

9 Click **OK** and exit the utility.

The dbdict.sql Form



Name	Type	Level	Index	SQL Name	SQL Type	SQL Table	SQL RC
descriptor	structure	0	1				false
name	character	1	1	name	varchar2(50)	m1	false
description	character	1	2	description	varchar2(50)	m1	false
approval.group	logical	1	3	approval_group	char(1)	m1	false
requests	logical	1	4	requests	char(1)	m1	false
tasks	logical	1	5	tasks	char(1)	m1	false
approvers	array	1	6				false
approvers	character	2	1	approvers	long	m1	false
members	array	1	7			a1	false
members	character	2	1	members	varchar2(50)	a1	false
duty.table	character	1	8	duty_table	varchar2(50)	m1	false
area	character	1	9	area	varchar2(40)	m1	false
manager	character	1	10	manager	varchar2(50)	m1	false
sysmodcount	number	1	11	sysmodcount	float	m1	false
sysmoduser	character	1	12	sysmoduser	varchar2(30)	m1	false
sysmodtime	date/time	1	13	sysmodtime	date	m1	false
company.not.used	character	1	14	company_not_used	varchar2(30)	m1	false
company	array	1	15				false
company	character	2	1	company	long	a2	false

Figure 6-6: Editing an SQL map file

Name – specifies the name of field in dbdict resources within ServiceCenter

Fields on the Fields tab

On this tab, indicate the fields and tablenames for the mapped file.

Field	Definition
Name	specifies the table name.
Type	specifies the RDBMS type.
Level	indicates the field level in the file's dbdict.
Index	indicates the field index in the file's dbdict.
SQL Name	specifies the name of the field in this table in the RDBMS.
SQL Type	specifies the type of the field in the RDBMS. varchar2(60) means that the type of field will be varchar2 and the length will be 60.
SQL Table	indicates the name of the table this field will reside in Oracle. The m1 is appended onto the name of resources. This becomes important when using array fields in multi-tables. Each array field will be mapped to a different A table.
SQL RC	indicates the flag for RC format.

Fields on the Keys tab

On this tab, indicate the key type for the selected field.

Field	Definition
Nulls & Duplicates	allows null and duplicate values.
No Nulls	disallows null values.
No Duplicates	disallows duplicate values.
Unique	indicates the primary key.
IR Key	indicates that the key will be used as an IR Key.

Note: 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, you can 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.

Fields on the SQL Tables tab

On this tab, enter the SQL table information for the selected field.

Field	Definition
Alias	indicates the table's alias name, i.e. m1, m2, a1,...
Name	indicates the table name.
Type	indicates the RDBMS type that was used for the conversion, i.e., sqlserver, db2universal, or oracle8.
Table Keys	
Table Options	indicates tablespace, index space or other spaces used for ServiceCenter table or index.

Fields on the General tab

On this tab, indicate other options for the mapped file.

Field	Definition
Root Record	indicates the file's root record number in P4.
Shadow	if selected, flags the file for Shadowing.
Reselect	

SQL Hints

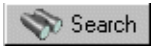
Modifying the sqlhints file is another method which allows you to override the default mapping. Each sqlhints record defines one field in a ServiceCenter file.

To access SQL hints:

1 Open the sql.hints file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

A search form is displayed.

2 Fill in information to restrict your search and click **OK** or press Enter.



3 Click **Search** or press Enter to find the records.

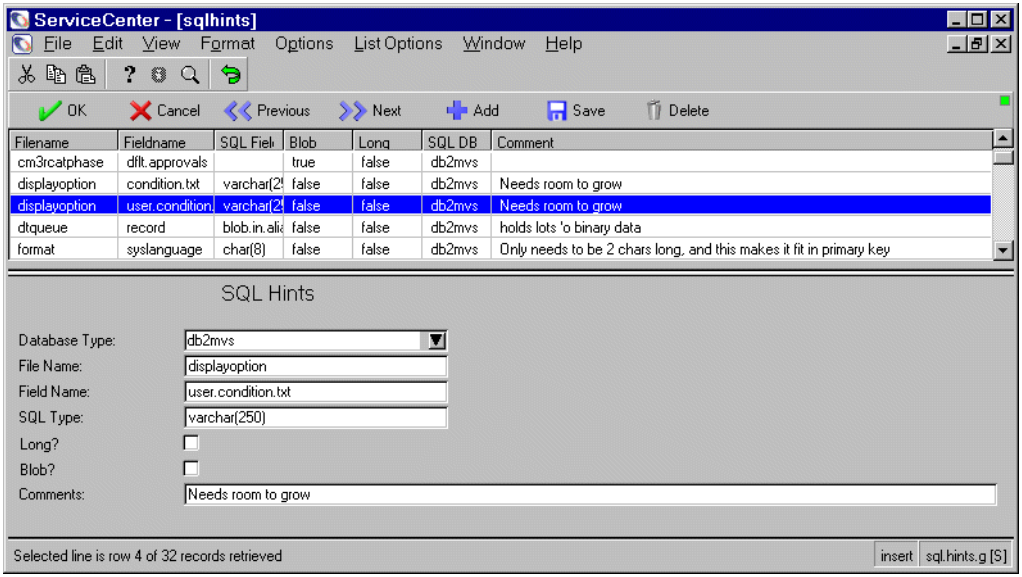


Figure 6-7: SQL Hints Record List

The fields allow you to identify a specific field in a file and customize the manner in which its data is mapped during a conversion.

To modify a current setting or create a new sql.hints record:

1. Select the **Database Type** value using the drop down menu button.
2. Identify the **File Name** whose field is being defined.
3. Identify the **Field Name**.
4. Enter a **SQL Type** value unless you are going to set either the **Long?** or **Blob?** fields, in which case leave this field blank.

   If either the Long? or **Blob?** fields are checked, it will ignore the **SQL Type** value. It is recommended that you leave the **SQL Type** field blank if either of the checkboxes contain values. Otherwise, the system uses exactly what appears in the **SQL Type** field. Therefore, when using this field, take into consideration the valid character type requirements for the database type selected, and include the length of the field if necessary. E.g., for Oracle you would use `varchar2(30)`, and for DB2 you would use `char(30)`.
5. Set the **Long?** and **Blob?** fields to either *true* or *false*. If left unchecked (equalling *unknown*), the value defaults to *false*.
6. Add any additional **Comments** or description for this mapping configuration.

7 Click **Add** to add a new sqlhints record, or click **Save** to modify an existing record.

Process Editing

When the sqldetect:1 parameter is added to the sc.ini file, the system updates the datamap and Database Dictionary of any file whose RDBMS tables have been modified. The file is refreshed automatically when touched by any user process. (Please refer to *Appendix A* of this guide for a description of this parameter.)

Note: The sqldetect parameter will only work if the new fields are added to the main table.

To modify your sc.ini file:

1 Log out of ServiceCenter.

2 Open the sc.ini file located in the RUN directory of your ServiceCenter directory.

3 Add the sqldetect:1 parameter to the bottom of the sc.ini file.

4 Select **Save** from the **File** menu.

5 Close the sc.ini file.

6 Restart the ServiceCenter express client.

7 Follow the instructions for adding a column to an RDBMS table. (See *Adding a Column to the RDBMS Table* on page 195.)

Important: Fields added to ServiceCenter's Database Dictionary in this fashion must be scalar in nature, since they are added to the bottom of the dbdict. To add fields to a structure within the dbdict (as in the probsummary file), you must edit the Database Dictionary and the map manually as described in the sections above.

Adding a Database Type

All SQL database types which are to be used in the RDBMS conversion must have ServiceCenter records. Records for many of the most commonly used databases are already in ServiceCenter.

There are three files for each database at work during a conversion:

File	Purpose
sqldbinfo	Defines database type, regulations and constraints. This is used in the mapping step (important that this be set up correctly for each database).
sqlhints	Allows you to override the default mapping for a specific field held in a file. See *SQL Hints* on page 203 for more information.
sqlwords	Allows changing reserved words. If a P4 field uses one of these names, the mapping step adds the suffix field to the resulting SQL field name.

If you find the need to add a new database type, use the following procedure to enter it into the system so the appropriate supporting files are also generated.

To add a database type:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click on the **SQL DB Types Utility** button to create a record for a new RDBMS database type.

3 Enter the name of the new SQL DB type you want to create in field provided on the pmc.sql.utility.form.g form.

When creating a new SQL server type, use an existing type, so the sqldbinfo, sqlhints, and sqlwords files are copied for the new type.

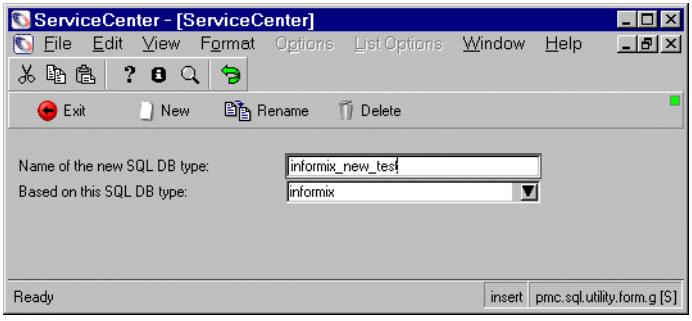


Figure 6-8: Adding an SQL Database

4 Click **New** to create the new record

5 You can also rename an existing record by selecting **Rename**.

When the new record is generated for the SQL server type, a confirmation is displayed.

6 Click **OK** to return to the record.

7 Click **Exit** to return to the **SQL Menu**.

Refreshing Files

All SQL mapping information is cached in memory. If this mapping information is altered for a file that has been converted to an external database, the memory must be flushed before those changes take effect.

ServiceCenter automatically refreshes these files after the conversion process. No other refreshing action is required for these files at that time.

Adding New Array Table to a Converted Table

Before adding a new array table to an existing RDBMS table, the DBA should be notified for space considerations.

To add new array fields to a table that has *already* been converted to the RDBMS, the new tables must first be created in the RDBMS. Within the new RDBMS table, the name must be the same as the main table replacing "m1" with "a" and the next sequential number. All array tables for the same M1 table carry unique index fields as well as a record_number field so each occurrence of the array is unique.

After this is accomplished in the RDBMS, the table/field must be added to the dbdict within ServiceCenter.

In Figure 6-9 on page 208, a new field called ***array.field*** is added to the incidents dbdict as an array with a second entry of character.

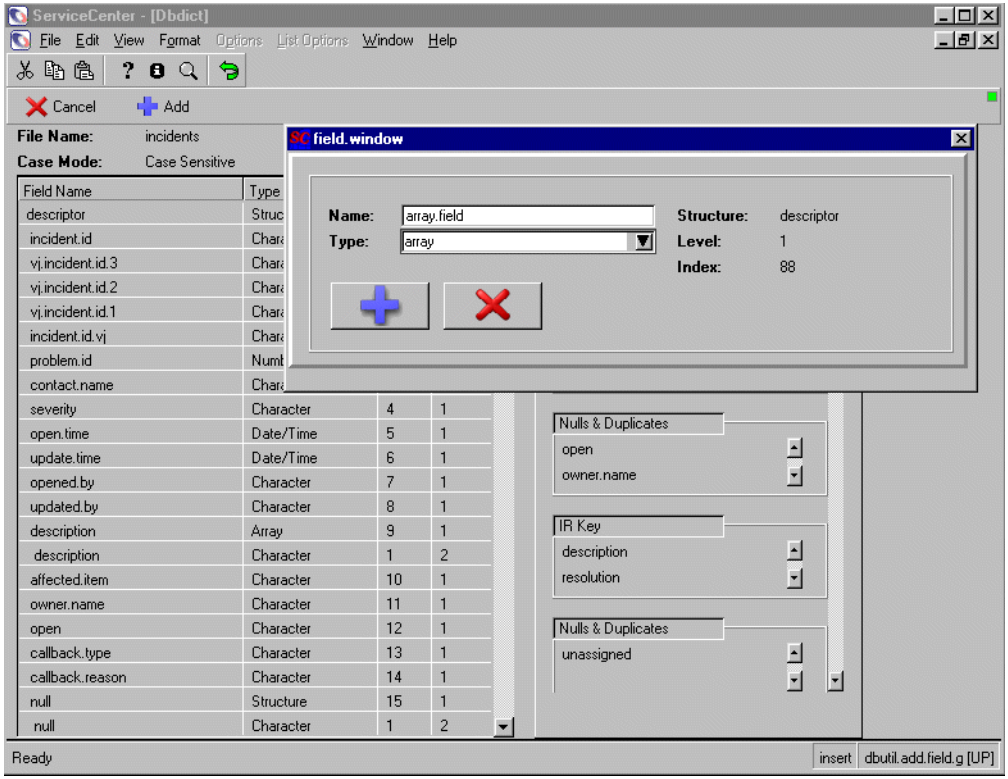


Figure 6-9: Adding New Array Fields to Table

Querying the Database Directly

This feature allows you to directly query the RDBMS database to return a list of requested rows from a table. Use this feature to retrieve information from tables that do not exist within ServiceCenter.

To directly query the RDBMS database:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click **SQL Query Tool** in the **SQL Menu**.

3 Enter your query in the **SQL Query** field, using proper syntax.

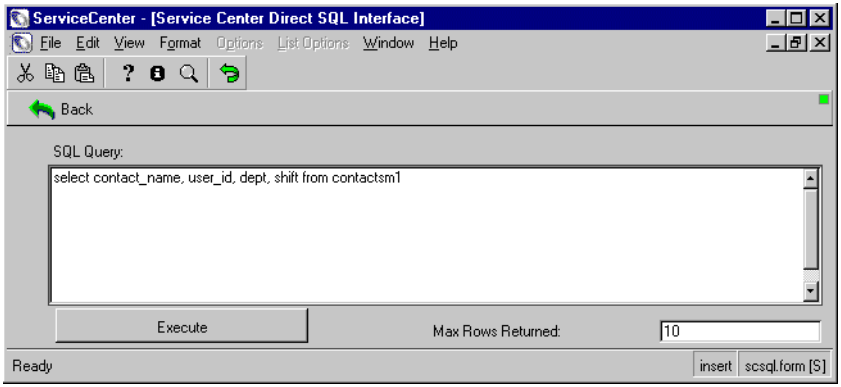


Figure 6-10: Querying the SQL Database

4 Select the number of rows you want to display in the **Max Rows Returned** field.

5 Click **Execute**.

The requested data table is displayed.

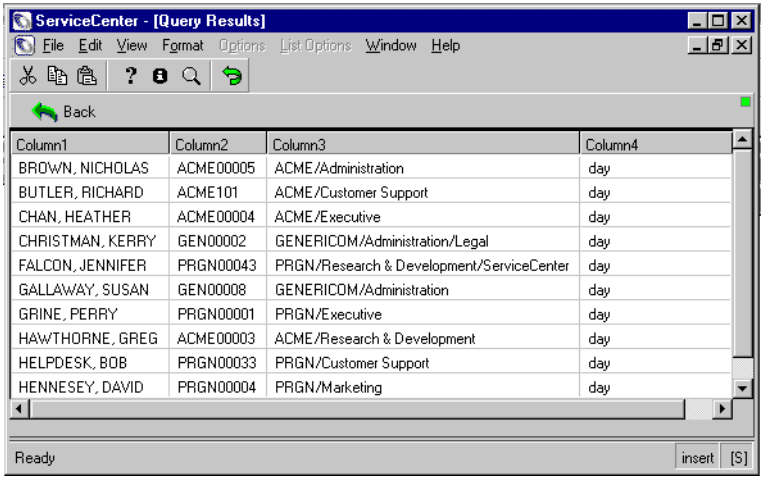


Figure 6-11: Data Table Returned from an SQL Query

Backup after Conversion

At the conclusion of a complete SQL conversion there is no production data that is left in the P4 file system. Therefore, the P4 file system does not have to be involved in any regularly scheduled backups. It is sufficient to use the backup procedures of the target RDBMS.

Warning: Do no delete the P4 file system (scdb* files).

The P4 file system (scdb* files) can not be deleted and there are 4 reasons for this:

1. Company wide time zone record is written into the P4 file system.

   The capability exists to create a company wide time zone record which is the time zone that all processes use when originally hooking into the system. This record is written into the P4 file system but does not belong to any file and is therefore not converted to SQL. This timezone record is usually established during the initial implementation phase and does not change. Therefore a single backup of the nearly empty P4 file system right after a complete conversion can be used to restore this record if their is ever a disaster and the P4 files have to be restored.

2. The bootstrap dbdict is in the P4 file system.

   The dbdict is a file with ServiceCenter that describes all other files. In order for the system to come up initially we have to be able to read the dbdict itself to get file descriptions and the code file to load the applications. The mapping of the file is contained in the dbdict, and the dbdict may itself be mapped. There is a bootstrap dbdict description within the P4 file system that contains the mapping for the dbdict itself. This is a static record and a single backup of the P4 file system after the complete conversion is all that is necessary.

3. New files are originally created in the P4 file system.

   When the Database Dictionary utility is used to create a new file, that file is created initially as a P4 file. Once that file has been defined then the SQL Utilities can be used to map the file to the target RDBMS. The file only exists in P4 until the conversion is done. The file does not contain any data and the definition of the file itself is an entry in the dbdict file which has already been converted to SQL. Therefore no backup of the P4 file system is required.

4. Temporary files for Reports are created in the P4 file system.

Reports created by ServiceCenter sometimes need temporary files to manage and sort the data required for the report. These temporary files are created in the P4 file system. They are truly just temporary files, that are created, used, and destroyed during the reporting process. Therefore no backup of the P4 file system is required.

7 Tuning for Performance

CHAPTER

This chapter was designed to explain the procedures for enhancing performance for SQL, Oracle, and Sybase Relational Database Management System (RDBMS) formats to ServiceCenter system and database administrators.

Topics in this chapter include:

Enhancing SQL Server Performance

After completing the SQL Conversion, the SQL server database administrator (DBA) should tune the data in the SQL server database to improve the overall performance. This procedure is described in the specific documentation for your chosen database, e.g., *Microsoft SQL Server Administrator's Companion*.

To update information about the distribution of key values in specific indexes, the SQL server DBA should run the UPDATE STATISTICS command on Microsoft SQL Servers. This information is useful for tables that have an indexed column and for those to which a great deal of data has been added, changed or removed.

Enhancing Sybase Performance

See the *Sybase System Administrator's Guide* for details on tuning the data in the Sybase database to improve the overall performance.

To update information about the distribution of key values in specific indexes, the Sybase DBA should run the UPDATE STATISTICS command. This information is useful for tables that have an indexed column and for those to which a great deal of data has been added, changed, or removed.

Enhancing Oracle Performance

ServiceCenter creates DDL and allows modifications to it before executing; therefore, some of the changes recommended below can be made by the DBA to the DDL before the tables and indices are created. In earlier versions of ServiceCenter, the DBA was required to allow the default conversion to complete before altering the tables and tablespaces.

Use the following criteria for calculating BLOB (binary large object) sizes for arrays within ServiceCenter.

See the *Oracle Server Administrator's Guide* for details on tuning the data in the Oracle database to improve the overall performance.

Important: Oracle supports only one long field per table. When converting to Oracle, if there is more than one array in a given table, all but the first array must go as single rows in an Oracle table. The default mapping will put the first array into the main table (m1). Later arrays will be placed into array tables (a1,a2, etc.).

To enable Oracle's cost-based optimizer, it is necessary to run the ANALYZE command on all ServiceCenter tables and indexes. These statistics are useful for tables that are accessed by SELECT, INSERT, UPDATE and DELETE statements. For good performance, this needs to be done weekly. The recommended parameters are COMPUTE STATISTICS for indexes, and ESTIMATE STATISTICS SAMPLE 20 PERCENT for tables.

Tablespaces

This section assumes that you are using Oracle Financial Analyzer (OFA), and that you have followed Oracle's recommendations of locating your *system* tablespace, *temp* tablespace, data tablespaces, index tablespaces, rollback segments, redo logs, and archive logs on separate disks as much as possible.

It is possible to point indexes and data to different tablespaces during the conversion process. If the **tablespace** option in the Advance SQL option is not available, one way is to set the default tablespace for the Oracle user who would be responsible for the conversion, to the desired tablespace. This should be done on the Oracle side before the conversion takes place.

With Oracle 7.3 or later, Oracle indexes can be rebuilt using the REBUILD clause. This allow the use of tablespace other than the default or ServiceCenter provided.

Note: Rebuild the indexes only if they were not separated into a different tablespace during conversion.

The syntax for this action is as follows:

```
ALTER INDEX [index name] REBUILD TABLESLocalProductShortPACE
[tablespace name]
```

Important: If you only have a few disks, separate the ServiceCenter data tablespaces from the index tablespaces.

ServiceCenter allows you to specify a data tablespace and a separate index tablespace for each file.

When setting up tables and tablespaces:

1. Drop (or don't create) indexes for tables which are never used, or which contain less than 20 - 100 records.
2. For most tables it is safe to use a 50K default, but for the largest tables, it is important to size the tables and indexes appropriately. ServiceCenter generates a **storage** clause for each table with enough room to store the current records in one extent. You may want to modify these to use standard sizes, or to allow for future growth.
3. To improve performance, create separate tablespaces (on different disks) for each of the most highly updated tables.
4. *Create* separate tablespaces for the indexes for these tables, and locate each of these on a different disk from the associated table.
5. Create a separate tablespace for the spool data tables. These grow rapidly during reports, and then shrink down considerably.

   Periodically export the table definitions, drop and recreate the tablespace and re-import the table definitions to clean up this tablespace.
6. For ease of maintenance, you can move all of the tables that are never used to a separate tablespace, and make it read-only. Another alternative is to leave these in P4.

Table Extents

It is possible to achieve the end result of each table occupying a single extent during the conversion process with left over room for growth in that initial extent. This may not be the best course to follow in all implementations, however.

To set the initial extent when creating a table, alter the default storage of the default tablespace in which the tables will be created using a command similar to the following:

```
ALTER TABLESPACE [default tablespace name] DEFAULT STORAGE (INITIAL
3M)
```

This will alter the tablespace to use an initial extent of 3 MB whenever a table is created in that tablespace.

Note: An issue to consider is that tables in ServiceCenter can be very small or very big. If the default settings are set for the maximum size file, all the files are likely to have a lot of wasted space. Since 1000 or more tables are created, having all of them with an initial extent of the largest table might not be possible on all systems. Available space quickly becomes an issue when choosing this course.
In ServiceCenter the initial size is set automatically by the system.

Tuning Memory

To tune the memory:

1 Verify that your SGA is sized appropriately.

For a database server with no other applications running, the rule of thumb is about 1/3 of the computer's memory for Oracle, and never more than 1/2.

2 Use your OS utilities to monitor memory use and watch for excessive swapping and paging.

Note: The following suggestions may increase the size of your SGA. Check it again at the end to make sure that it is not too large.

3 Oracle ships with the init.ora parameters set small, to ensure that the database will start up.

- Increase these settings after startup to achieve the desired level of system performance.

4 Modify the following Oracle parameters that affect how memory is allocated within the SGA.

Detailed information on tuning is available in the *Oracle 8 Administrator's Guide* and *Oracle 8 Tuning manuals*, and other books from Oracle Press and O'Reilly.

Database Buffer Cache

- Calculate the buffer cache hit ratio with this formula:

Hit Ratio = 1 - (physical reads / (db block gets + consistent gets))

The three numbers you need can be selected from the V$SYSSTAT table:

```
select name, value from V$SYSSTAT;
```

If your hit ratio is less than 90%, you need to increase the DB_BLOCK_BUFFERS initialization parameter.

Shared Pool

```
select (sum(pins - reloads)) / sum(pins) "Lib Cache" from v$librarycache;
```

This ratio should be at least 99%. If it is lower than that, you need to increase the SHARED_POOL_SIZE initialization parameter.

Sorts

- Look at sorts(memory) and sorts(disk) in V$SYSSTAT to see how many sorts are being done in memory and how many on disk.
- If there are many sorts happening on disk, you need to increase the SORT_AREA_SIZE initialization parameter.

Other Parameters

- Set the DML_LOCKS initialization parameter to at least three times the maximum number of simultaneous users.

ServiceCenter Modifications

Multiple Tables

ServiceCenter supports vertical field splitting within a file. In other words, one ServiceCenter file, e.g., the example file, can map to multiple tables: fields 1, 3, and 7 in file examplem1, fields 2, 4, and 5 in file examplem2, and field 6 in file examplea1.

Array Field Mapping

ServiceCenter maps array fields to a *long* field type in Oracle. The first *long* field type is placed in the main table; additional arrays are mapped to tables ending with **a1**, **a2**, etc. This happens because Oracle only allows one *long* field type per table. If you have several arrays in a table, the additional queries, and joins required by Oracle to fetch your data can substantially impact performance.

- Look at the data that you are actually putting into these arrays, and for the arrays that you know will be less than 2000-4000 characters, customize the mapping in ServiceCenter to `VARCHAR2(2000)` or `VARCHAR2(4000)`.

The limits on `VARCHAR2` are *2000* in Oracle 7, and *4000* in Oracle 8.

LOB Support

See *LOB Support* on page 50.

Tuning Indexes

The indexes defined in the ServiceCenter Database Dictionary are optimized for P4. When converting to Oracle, all P4 indexes are converted along with your data. Some of these may not be useful in Oracle.

To tune indices:

1. Determine which indexes are not being used, and drop them.
2. Look at long-running queries, and add indexes where needed.

   You can do this using Oracle's Enterprise Manager Performance Pack, or by using Oracle's SQL Trace facility, TKPROF, and EXPLAIN PLAN.

   Note: You can also locate long-running queries by looking for `sqllimit exceeds` in ServiceCenter's `sc.log` file.

Enhancing DB2 Performance

Steps to enhance DB2 Performance:

1. Run the DB2 utility db2advis. See *The db2advis utility* on page 220.
2. Set the `DSMAX` parameter for DB2 sufficiently high. See *The DSMAX parameter* on page 220
3. Ensure that the EDM buffer pool is large enough. See *The DB2 EDM buffer pool* on page 220
4. Beware of losing space by dropping and creating tables. See *Dropping and Creating tables* on page 221
5. Be sure to set up the appropriate tablespace. See *Tablespace* on page 221.
6. Set up high activity files for greater efficiency. See *High activity files* on page 223.
7. Use the debugging tools. See *Database debugging parameters* on page 224.

The db2advis utility

Run the DB2 utility db2advis against a text file containing captured SQL statements. For more information on how to use the db2advis utility, consult the IBM DB2 Universal Database Command Reference guide.

To run the DB2 utility db2advis:

1. Use the built-in monitoring functionality of DB2 to save a text file containing SQL statements sent from ServiceCenter to DB2.
2. Use the db2advis utility to examine the SQL statement(s) and compare them to the indexes existing in the database.

   Note: If the utility finds a statement that it decides requires an index, it outputs that information to a table.

Here is an index that was proposed by db2advis during a recent benchmark test:

```
CREATE INDEX WIZ41 on db2inst1.probsummarym1 (flag desc, priority_code desc);
```

ServiceCenter creates a DDL and allows modifications to it before executing; therefore, some of the changes recommended below can be made by the DBA to the DDL before the actual data is converted. In earlier versions of ServiceCenter, the DBA was required to allow the default conversion to complete, and then alter the system as appropriate.

The DSMAX parameter

A fully converted system introduces a large number of tables and indexes.

- Ensure that the DSMAX parameter for DB2 is set sufficiently high to handle the new files. A value of *5000* is a good starting point.

The DB2 EDM buffer pool

A fully converted system creates all the tables in a single database (PRGNDB). This creates a large DBD control block. DBD control blocks are loaded in the DB2 EDM buffer pool.

- Monitor the DB2 EDM buffer pool to make sure it is large enough. A value of 1MB is a recommended starting point.

Dropping and Creating tables

Important: Dropping tables and recreating them in the same database can increase the size of the DBD. Dropping a table does not automatically remove the entry from the DBD.

To make sure the DBD has a minimal size:

1. REORG the tablespaces for tables that have been dropped and recreated.
2. Run MODIFY RECOVERY to remove the old image copy information for the dropped table.
3. Make new image copies.

Tablespace

To setup the appropriate tablespace:

1. Set the default close rule if necessary.
2. Set your installation default to *TYPE 2* indexes.
3. Verify that the 32K buffer pool is large enough to handle the ServiceCenter volume.

   A number of ServiceCenter files must be in the 32K tablespace. By default, the conversion will place all tables into the 32K tablespace. ServiceCenter system files (see the table below) generally contain records that exceed 4K and these must remain in the 32K tablespace. The tables defined for these files generally contain a single **LONGVARCHAR** field, to hold the binary contents of the data and a few other fields that are used as KEYS to retrieve the records. No other files are required to go into the 32K tablespace. If the number of fields and the mapping done is small enough, these tables can be modified to go into the 4K tablespace. In the future, ServiceCenter will recognize this, and will use the 4K tablespace where indicated. Currently the DBA must look at the DDL for the tables created and then move those files that will fit into the 4K tablespace.

System Files

application	dbdict	displaycache
displayevent	displaymaster	displayoption
displayscreen	dtqueue	dtshad
enclapplication	eventin	eventout

System Files

formatctrl	info	link
macro	macrodef	macroheader
menu	sc	scparms
screlconfig	scripts	sqlqueue
status	termtype	tzfile
upgdbdict	upgrade	Upgradeapplication
upgradeddbdict		

4 Determine if any of the array tables can be moved into the 4K tablespace. A ServiceCenter file (Database Dictionary entry) may consist of a number of array fields.

The operator file contains arrays to hold such items as:

- The names and values for parameters for the first menu displayed for this user
- The names of the security groups to which the user belongs
- The capabilities that the user has
- The abbreviations used to display months for this user

Each array has a variable length and each array could hold as much as 32K worth of data. Therefore, the default action is to turn each array into a table that contains a single **LONGVARCHAR** field. Thus the operator ServiceCenter file will generate numerous DB2 tables (operatorm1, operatora1, operatora2, operatora3, etc.). Each of the array tables (a1,a2,a3, etc.) is placed into the 32K tablespace. If you know that the data going into the array will not exceed 4K, the table can be put into a 4K tablespace.

5 Combine the tables for files that are never used into a single segmented tablespace and remove all the indexes for the files. Combining these files into a single segmented tablespace eases the maintenance load. Also put these files into a separate database so that these tables do not take up space in the primary DBD and EDM pool. The other alternative is to leave them in P4.

6 Consider combining the following files into a single tablespace. The reason they can be combined is that the first user of the system normally reads them into shared memory. Putting them into a single tablespace reduces the number of datasets that are opened. The files that are read once and cached are: joindefs, erddef, sctypecheck, scmandant, scaccess, and scdsites.

7 Use segmented tablespaces. This will be done automatically in future releases of ServiceCenter.

A segmented tablespace has more efficient space utilization, which results in less overhead for inserts and for variable-length row updates.

8 ServiceCenter supports vertical field splitting within a file. In other words, one ServiceCenter file, e.g., the example file, can map to multiple tables: fields 1, 3, and 7 in file examplem1, fields 2, 4, and 5 in file examplem2, and field 6 in file examplea1.

Important: Use caution when you use views and joins under the covers to create data needed by ServiceCenter. ServiceCenter always uses an order by clause to force the data to be returned in primary key order. Whenever you specify columns from multiple tables in the order by clause of a join statement, DB2 invokes a sort. ServiceCenter does not expect that the primary key has been split into more than one table.

High activity files

- Consider setting MAXROWS=1 for the high usage files so that there is no locking contention on the records. These files include: number, counter, clocks, work, schedule, dtqueue, and sqlqueue. Also consider giving the above files their own tablespace and bufferpool so queries that bring in data do not flush data from these files out of the pool.
- Make sure the high activity files are not over indexed from a DB2 point of view.
  - The indexes defined in the ServiceCenter Database Dictionary are designed for quick access by P4 in the standard product. P4 needs composite indexes to work efficiently. It could be that the indexes needed for a standard P4 system are not appropriate for a customized DB2 system. Verify that the indexes are actually necessary and remove those that are never used, as they cause extra overhead during *inserts* and *updates*.
  - Fewer indexes may be needed in DB2, as DB2 is able to do *index OR* processing, where P4 requires a composite index for each set of fields being used in a query.

Database debugging parameters

- Use the ServiceCenter **debugdbquery** and **sqldebug:1** parameters to determine the queries that are being used in your system, then use EXPLAIN in DB2 to determine the access path for these queries. Drop any indexes that are not being used.
- Use the ServiceCenter **sqldebug:1** parameter to determine the columns that are generally updated in a ticket. Reorganize the table so that the updated columns are defined last. This will reduce the amount of logging that DB2 has to do when the record changes.

CHAPTER 8

Troubleshooting

This chapter was designed to aid ServiceCenter® system and database administrators in trouble shooting their systems. It discusses common Relational Database Management System (RDBMS) related errors for each database format, and contains information that can help improve CPU performance, and network performance.

Topics in this chapter include:

General Issues

If Conversion Fails

If the conversion process fails, contact Peregrine Customer Support and be prepared to supply the contents of the following files:

- sc.ini file
- sc.log file

Tracking Errors on OS/390

Most errors occur when one of the preceding steps has not been completed. Some of the most common ones follow. If you or the database administrator (DBA) cannot determine the cause of the problem, do not hesitate to contact Peregrine Systems Support.

Error messages can be viewed in one of two places:

- The Message pop-up window.

  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.

- The joblog of the ServiceCenter job. This joblog is accessible via SDSF or some other equivalent.

Common RDBMS Related Errors

DB2/OS/390

Problem:

User cannot connect to the DB2 database.

Solution:

Timestamp mismatch exists between the BIND timestamp of DBRM and the LOADLIB, message id -818.

- The DBRM file and the LOADLIB from the install tape must match. The timestamp matches by taking the DBRM file off the tape from which the LOADLIB was unloaded.

- Before using the DB2/OS/390 interface you must BIND the DB2 Application Plan as stated in above.
- The sqldb initialization parameter in the ServiceCenter sc.ini file should be set to the DB2 subsystem name. See *Initialization Parameters for RDBMS* on page 297 for more details.

Informix

Problem:

Cannot connect to Informix database.

Solution:

- The user environment variable INFORMIXSERVER should be set and should match the name of the Informix server to which you are connecting.
- If the Informix server is located on a remote site, the user environment variable INFORMIXDIR should be set and the file $INFORMIXDIR/etc/sqlhosts must contain the correct connection information.
- Make sure the user can connect to the Informix server by running the Informix **dbaccess** utility and connecting to the database name and the server name, such as:

  my_database_name@server_name

- The **sqldb** initialization parameter in the ServiceCenter sc.ini file should be set to the Informix server name. See *Initialization Parameters for RDBMS* on page 297 for more details.

Oracle

Problem:

Cannot connect to Oracle database.

Solution:

- The user environment variable **ORACLE_SID** should be set and should match the name of the Oracle server to which you are connecting.
- Make sure that the user can connect to the Oracle server by running the Oracle sqlplus utility with the following parameters:

- sqlplus <userid>/<password>
- The sqldb initialization parameter in the ServiceCenter sc.ini file should be set to the connection string for Oracle. See *Initialization Parameters for RDBMS* on page 297 for more details.

Problem:

Error connecting to Oracle using SQL*Net V2.

Solution:

- Make sure that the user can connect to the Oracle remote server by running the Oracle **sqlplus** utility with the following parameters:

  sqlplus <userid>/<password>@<TNS_NAME>

- The sqldb initialization parameter in the ServiceCenter sc.ini file should be set to the name of the service found in the file tnsnames.ora. See *Initialization Parameters for RDBMS* on page 297 for more details.

SQL Server

Problem:

Cannot connect to SQL server database.

Solution:

- Make sure that the user can connect to the SQL server database by checking the login using the Microsoft SQL Enterprise Manager or by running the SQL server **isql** utility with the following parameters:

  isql -U <userid> -P <password> -S <servername>

- The sqldb initialization parameter in the ServiceCenter sc.ini file should be set to the name of the SQL server. See *Initialization Parameters for RDBMS* on page 297 for more details.

Sybase

Problem:

Cannot connect to Sybase database.

Error Message:

Sybase error 20012 - Server name not found in interface file.

Solution:

- The user environment variable **DSQUERY** should be set and should match the name of the SQL server to which you are connecting. The SQL server names can be found in the Sybase interfaces file.

- Make sure that the user can connect to the Sybase database by running the Sybase isql utility with the following parameters:

  isql -U <userid> -P <password> -S <servername>

  The sqldb initialization parameter in the ServiceCenter sc.ini file should be set to the name of the SQL server. See *Initialization Parameters for RDBMS* on page 297 for more details.

Error messages

ORA-0001: unique constraint (tablename) violated.

An UPDATE or INSERT statement attempted to insert a duplicate key. Run scserver or scenter with the sqldebug initialization parameter to view the contents of the UPDATE or INSERT statement. Either remove the data from the Oracle database or do not insert the key.

ORA-00955: name is already used by an existing object.

An attempt was made to create a database table or index that already exists. Run scserver or scenter with the sqldebug initialization parameter to view the contents of the CREATE statement. Enter a unique name for the database table or drop the existing table.

ORA-12150 thru 1212699: network related errors

Subset ORA-12196 through 12285 are TNS (Transparent Network Sustrate) errors generated specifically by NR (or routing) components.

ORA-12203: unable to connect to destination

- This scenario could have been brought about by any of the following conditions:
- Invalid TNS address was supplied.
- Destination is not listening.
- Underlying network transport problems.
- Make sure to set the following parameters in the sc.ini file:
  sqldb: DBNAME
  sqllogin:IDSTR <userid/password>

Note: DBNAME above is the actual Oracle database name.

- Make sure to copy **scserver** to **scserver.old** and copy **scserver.dbname** to **scserver**, where **dbname** is either Oracle, Sybase, or another database name.
- Make sure the owner of ServiceCenter can connect to the remote server by running the **sqlplus** utility. For example, to connect to the Oracle database, use the following parameters:

  sqlplus <userid>/<password>@<TNS_NAME>.
- The sql initialization parameter should be set to the name of the service found in the file tnsnames.ora. For example, from the **ServiceCenter ../RUN directory**, type the following to test connectivity:

  sqlplus sqllogin values <userid/password> @ sqldb value <TNSNAME>
- It is recommended to run normal TCP/IP checks, (Ping, Telnet, etc.), verify servicename, verify listener is running at remote node and verify that the address parameters specified in **TNSNAMES.ORA** are correct.
- If interchanges are used, check that all needed to make the connection are up and running. Ensure **ORACLE_SID** is correct (matches to the server to which it is connecting), and ensure **TNS_ADMIN** is set correctly to point at the location of the configuration (.ORA) files.

Network Performance Troubleshooting

Ping Utility

Use **ping** to determine how long it takes to get data from one computer to another.

Example:

```
>ping hpgen
Pinging hpgen.peregrine.com [204.33.92.13] with 32 bytes of data:
Reply from 204.33.92.13: bytes=32 time<10ms TTL=254
Reply from 204.33.92.13: bytes=32 time<10ms TTL=254
Reply from 204.33.92.13: bytes=32 time<10ms TTL=254
Reply from 204.33.92.13: bytes=32 time<10ms TTL=254
```

Tracert Utility

Use **tracert** to determine the number of "hops" used to get data from one computer to another.

Example:

```
>tracert hpgen
Tracing route to hpgen.peregrine.com [204.33.92.13]over a maximum of
30 hops:
 1 <10 ms <10 ms <10 ms 7500fe-6-1-0.peregrine.com [172.17.1.1]
 2 <10 ms 10 ms <10 ms hpgen.peregrine.com [204.33.92.13]
```

Debugtransport Sc.ini Parameter

Enter the **debugtransport** parameter in the **sc.ini** file. This parameter should be put in both the client and server **sc.ini** file.

Warning: An extremely large amount of Text will be written to **sc.log**.

Example client **sc.log**:

```
------------Logfile opened------------
-3769399 11/04/99 09:57:10 Character translation initialized for
american language
-3769399 11/04/99 09:57:10 SC3270 character translation initialized
for american language
-3769399 11/04/99 09:57:10 Multibyte system initialized successfully
-3769399 11/04/99 09:57:11 bitmap cache enabled for 48 bitmaps
-3769399 11/04/99 09:57:13 xprt_init: entered with type -1 and
service 172.20.1.1.4001
-3769399 11/04/99 09:57:13 xprt_open: entered
-3769399 11/04/99 09:57:13 xprt_open: succeeded
-3769399 11/04/99 09:57:13 clntxprt connecting
-3769399 11/04/99 09:57:13 xprt_connect: entered
-3769399 11/04/99 09:57:13 xprt_connect: using port 4001
-3769399 11/04/99 09:57:22 xprt_connect: connected to 172.20.1.1:4001
-3769399 11/04/99 09:57:22 xprt_connect: succeeded
-3769399 11/04/99 09:57:22 clntxprt connected
-3769399 11/04/99 09:57:22 xprt_send: Sending 869 bytes
-3769399 11/04/99 09:57:22 xprt_send: send() returned 869
-3769399 11/04/99 09:57:22 xprt_status: entered
```

Example server **sc.log**:

```
254 11/04/99 09:58:44 Character translation initialized for default
language
254 11/04/99 09:58:44 info from ENV vars: Domain = <2>, Socket# <7>
Name = <SC  >, Task = <SASCL04f>

254 11/04/99 09:58:44 xprt_init: entered with type 1 and service 4001
1 254 11/04/99 09:58:44 xprt_status: entered
254 11/04/99 09:58:44 xprt_status: nStatus(1)
254 11/04/99 09:58:44 nStatus is 1 in xprt_status
```

```
254 11/04/99 09:58:44 Socket 49 is read ready
254 11/04/99 09:58:44 recvTCP(): bytes to read = 8
254 11/04/99 09:58:44 recvTCP(): requesting 8 bytes
254 11/04/99 09:58:44 recvTCP(): recv returned 8
254 11/04/99 09:58:44 xprt_receive: initial receive obtained 8 bytes
254 11/04/99 09:58:44 recvTCP(): bytes to read = 861
254 11/04/99 09:58:44 recvTCP(): requesting 861 bytes
254 11/04/99 09:58:44 recvTCP(): recv returned 861
254 11/04/99 09:58:44 xprt_receive: second receive() obtained 861
bytes
254 11/04/99 09:58:44 GetMyIP entered with unknown and 12670
254 11/04/99 09:58:44 ServiceCenter scserver pid (254)
254 11/04/99 09:58:44 Release 2.1 SP3d (9903121336) System: 12670
(0x03137f00)
254 11/04/99 09:58:44 on 9672 running MVS/SP (6.0.5) from
(172.20.1.1)
254 11/04/99 09:58:44 connected to client at 128.1.72.229:1243
254 11/04/99 09:58:44 Attaching to resources for system
'172.20.1.1.4001' with key 3137f00
254 11/04/99 09:58:44 xprt_send: Sending 680 bytes
254 11/04/99 09:58:44 xprt_send: send() returned 680
254 11/04/99 09:58:44 xprt_status: entered
254 11/04/99 09:58:44 xprt_status: nStatus(1)
254 11/04/99 09:58:44 nStatus is 1 in xprt_status
254 11/04/99 09:58:44 Socket 49 is read ready
254 11/04/99 09:58:44 recvTCP(): bytes to read = 8
254 11/04/99 09:58:44 recvTCP(): requesting 8 bytes
254 11/04/99 09:58:44 recvTCP(): recv returned 8
254 11/04/99 09:58:44 xprt_receive: initial receive obtained 8 bytes
```

CPU Performance Troubleshooting

ServiceCenter Benchmark Parameter

Use the ServiceCenter **benchmark** parameter into determine how long it takes to accomplish certain tasks.

Example:

```
>scenter -benchmark
output in the sc.log:
311 11/11/99 07:58:11 >>>> Benchmark Start >>>>>
311 11/11/99 07:58:11 Elapsed 0.231 CPU 0.200 Test: 250,000 strcpy
311 11/11/99 07:58:11 Elapsed 0.060 CPU 0.060 Test: 250,000 overlappd
moves
311 11/11/99 07:58:11 Elapsed 0.020 CPU 0.020 Test: 250,000 MEMCPY
311 11/11/99 07:58:11 Elapsed 0.010 CPU 0.010 Test: 2,500 MALLOC /
FREE
311 11/11/99 07:58:11 Elapsed 0.010 CPU 0.010 Test: 2,500 CALLOC /
FREE
```

```
311 11/11/99 07:58:11 Elapsed 0.000 CPU 0.000 Test: 100,000 stfrees
311 11/11/99 07:58:11 Elapsed 0.030 CPU 0.030 Test: 1,000 parses of
$XNUMBERRESULT=1+3
311 11/11/99 07:58:11 Elapsed 0.030 CPU 0.030 Test: 1,000 eval of
$XNUMBERRESULT=1+3
311 11/11/99 07:58:11 Elapsed 0.040 CPU 0.040 Test: 1,000 eval of
$Y=10*2+3+2+1
311 11/11/99 07:58:11 Elapsed 0.000 CPU 0.000 Test: 2,000 rcget on
the format dbdict
311 11/11/99 07:58:11 Elapsed 0.070 CPU 0.050 Test: 100 Init's and
Term's of probsummary
311 11/11/99 07:58:12 Elapsed 0.160 CPU 0.140 Test: 100 true queries
against probsummary
311 11/11/99 07:58:12 Elapsed 0.040 CPU 0.040 Test: 500 rcget's to
initialize a DATUM
311 11/11/99 07:58:12 Elapsed 0.070 CPU 0.071 Test: 500 rcinit's to
initialize a DATUM
311 11/11/99 07:58:12 >>>> Benchmark Stop >>>>>
```

Elapsed time is the total real time. CPU time is how it took the CPU to process the request.

Status Command for OS/390

Use the modify **status** command on OS/390

Example:

```
/f <jobname>,status
Output from the above:
12.33.56  STC09386    SC040     STATUS COMMAND BEING PROCESSED

12.33.56  STC09386  SC010   PID USERID   CPU   MEMORY   IN USE   I/O's SEM
12.33.56  STC09386  -----   --- ------   ----  ------   ------  ----  ---
12.33.56  STC09386  SC011   14  inactive 0.88  953192   902040  13    0
12.33.56  STC09386  SC011   13  linker   0.89  953192   897272  11    0
12.33.56  STC09386  SC011   2   lister   0.99  1046001  972881  66    0
12.33.56  STC09386  SC011   11  marquee  0.92  1014036  938736  36    0
12.33.56  STC09386  SC011   10  agent    1.23  1200775  1126275 109   0
12.33.56  STC09386  SC011   9   S_C_AUTO 0.06  815153   766597  0     0
12.33.56  STC09386  SC011   8   change   0.98  957288   902220  137   0
12.33.56  STC09386  SC011   7   problem  2.40  1091798  994758
```

The key is to do the ‘status’ command repeatedly over a period of time and notice the difference in CPU accumulation (delta).

9 Shadowing P4

CHAPTER

This chapter was designed to aid ServiceCenter® system and database administrators in shadowing their P4 systems. It discusses the concept and strategy of shadowing, including procedures for shadowing individual files and for stopping shadowing.

Topics in this chapter include:

Overview

Shadowing allows you to copy your production system to a Relational Database Management System (RDBMS) for creating reports while retaining your working file system in P4. When shadowing is in effect, your file system is updated in both places.

There are two ways in which shadowing occurs within ServiceCenter:

- Synchronous
- Asynchronous

Note: You cannot convert a file that is shadowed. You must set the **Shadow** field to *false* (deselected), then convert.

Shadowing your entire P4 file system is not necessary. The best strategy is to shadow individual files against which you intend to run reports and leave your application files in the P4 format.

Synchronous Shadowing

When shadowing is *synchronous,* as defined with the **immediateshadow** parameter in the sc.ini file, the RDBMS update is performed immediately after the update to P4. In this mode, control is not returned to the user until both the P4 and the RDBMS databases have been updated. If a problem occurs with the RDBMS update, the update to P4 proceeds normally; however, the RDBMS database will be out of sync with the P4 file system.

Asynchronous Shadowing

Asynchronous shadowing is the ServiceCenter default. When this occurs, an update is made to P4, and a corresponding queue record is written to the sqlqueue file. Control is then returned to the user.

The sqlqueue file is processed by a background task, which removes entries from the queue and updates the RDBMS database. The background task is started with a scenter -que:sql command which must be added to the *scstart* procedure if you intend to use RDBMS shadowing, but does not have to be continuously running. If you want RDBMS shadowing to occur at specific times during the day, you need only start the background process at those times. The task will process the sqlqueue file until it is empty, then wait for more entries. It will continue to process entries as they appear until the process is terminated.

Shadowing

You must be the only user accessing the server during shadowing procedure. Please see the instructions for starting the server with only one user in *Pre-Conversion Server Preparation* on page 162.

To shadow files:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

Note: You cannot perform an RDBMS conversion of a file that is shadowed. You must set the **Shadow** field to *false* (deselected), then convert.

2 Click **Shadow Files onto SQL from P4** to convert your RDBMS database files to P4 files. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

The P4 to SQL Shadow Console is displayed.

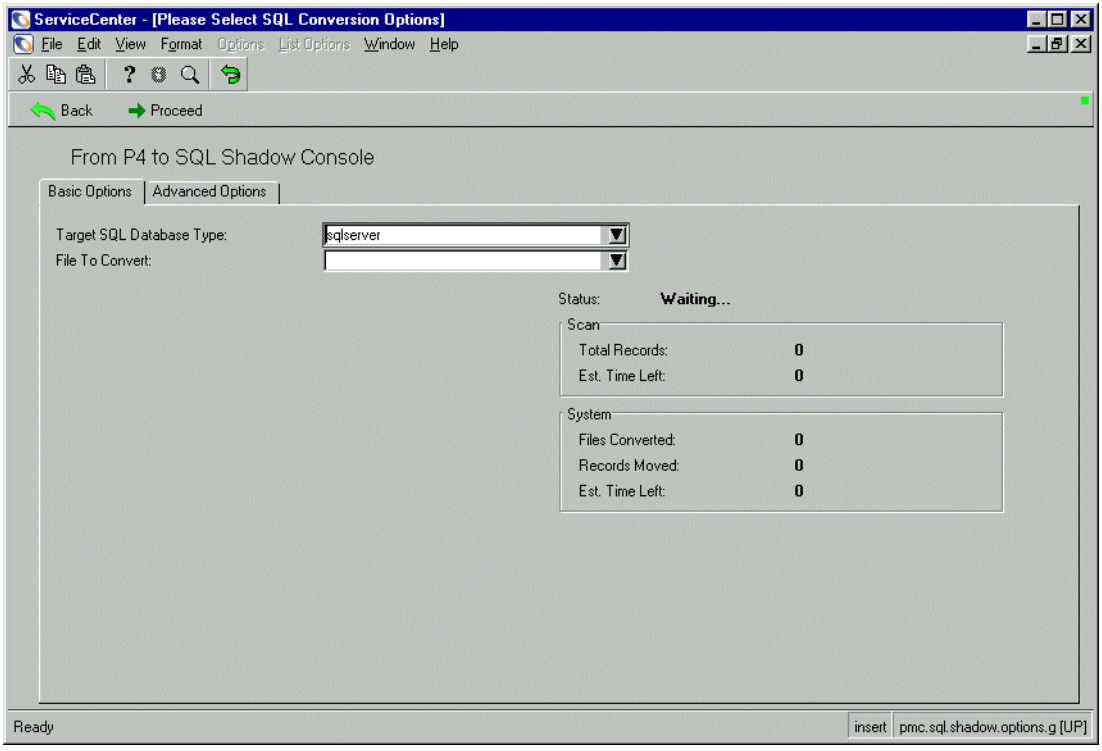


Figure 9-1: P4 to SQL Shadow Console

3 Select a file from the **File To Convert** list.

Note: It is recommended that you select specific files, and perform this process individually for each file. Do not select the *__All Files__* option from the drop-down list, as this may compromise system performance speed and can lead to data errors on certain RDBMS databases.

4 Select a Target SQL Database Type for shadowing.

5 Select the **Advanced Options** tab for special programming considerations.

Important: Peregrine recommends using the default options. Contact your database administrator (DBA) to select any option *other* than the default.

6 Click **Proceed** to shadow the file selected to your RDBMS format.

A message is displayed stating that the file has been moved to your RDBMS, indicating the file has been shadowed.

Stop Shadowing

The shadowing status of a ServiceCenter file that has already been converted can be disabled, in order to reconvert or update the file on an RDBMS.

Select one of the following two methods to stop shadowing files:

- Use the *SQL Menu Option* on page 239, to turn off shadowing and unconvert the files.
- Use the *Database Manager Method* on page 240 to turn off the shadowing feature without unconverting the file.

SQL Menu Option

Use the **Stop Shadowing Files** option to stop shadowing one or more files. You must be the only user accessing the server during stop shadowing procedure. Please see the instructions for starting the server with only one user in *Pre-Conversion Server Preparation* on page 162.

To turn off the shadowing feature:

1 Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

The **SQL Menu** (Figure 5-5 on page 171) is displayed.

2 Click **Stop Shadowing Files** to stop shadowing. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

3 Select a file from the **File to Convert to P4** drop-down list.

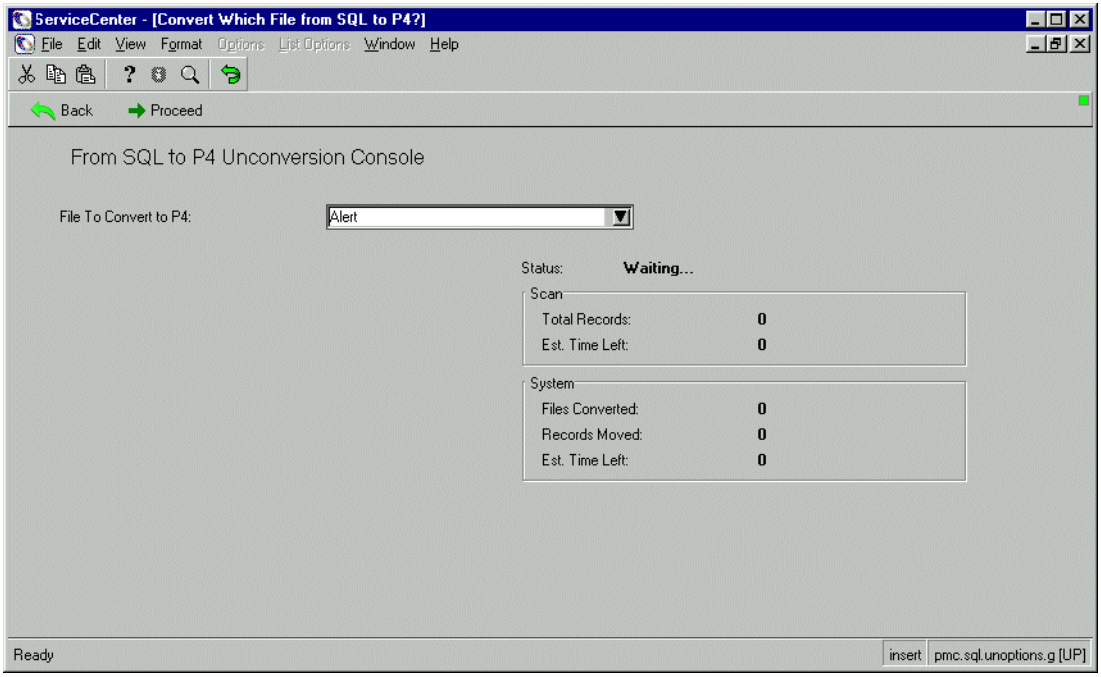


Figure 9-2: Converting Shadowed Files to P4

Note: If you select the *__All Files__* option, all files will be unshadowed and unconverted from all RDBMS databases.

4 Click **Proceed** to stop shadowing.

Database Manager Method

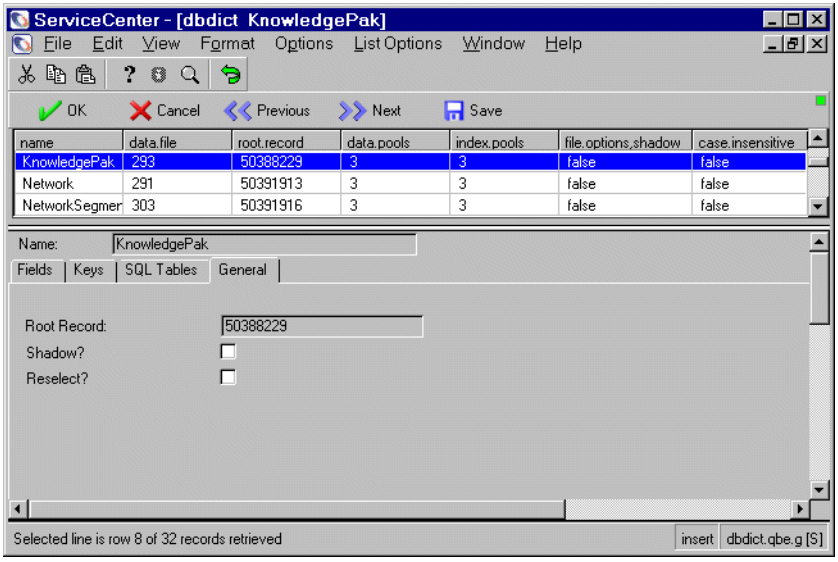
To turn off the shadowing feature without unconverting the file:

1 Open the dbdict.sql file in Database Manager. (For instructions, see Volume 2 of Database Management and Administration.)

A blank dbdict.sql form is displayed.

2 Click the **Search** button from the system tray to retrieve a list of all records accessed by this form.

3 Locate the desired record in the record list and click on the name to open the record.



4 Select the **General** tab.

5 Deselect the **Shadow?** checkbox.

6 Click **OK** to save the change, then **Back** to exit Database Manager.

The selected file is no longer shadowed.

10 CHAPTER Converting RDBMS Files Back to the P4 Format

This chapter was designed to aid ServiceCenter® system and database administrators in reconversion to P4. It discusses the procedure for converting an Relational Database Management System (RDBMS) database back into the ServiceCenter P4 format.

Conversion Procedure

Note: ServiceCenter version 2.1/A9802 and earlier required a manual IR key regeneration routine after the files were loaded back into P4. Later versions of ServiceCenter have automatic regeneration of these keys.

To convert all or part of a database back to P4:

1. Start your system as a sole user, as described in *Pre-Conversion Server Preparation* on page 162; shut down the server, restart the server in express mode, and connect with an express client.
2. Open the **SQL Menu**. For information on how to open the **SQL Menu**, see *To begin conversion:* on page 171.

   The **SQL Menu** (Figure 5-5 on page 171) is displayed.
3. Click **Move Files From SQL to P4** in the **SQL Menu** to convert your RDBMS database files to P4. This option will not work if you have not done the preconversion server setup. See *Pre-Conversion Server Preparation* on page 162 for instructions.

4 Select *All Files* or a single file name from the drop down list in the **File to Convert to P4** list in the Conversion Console.

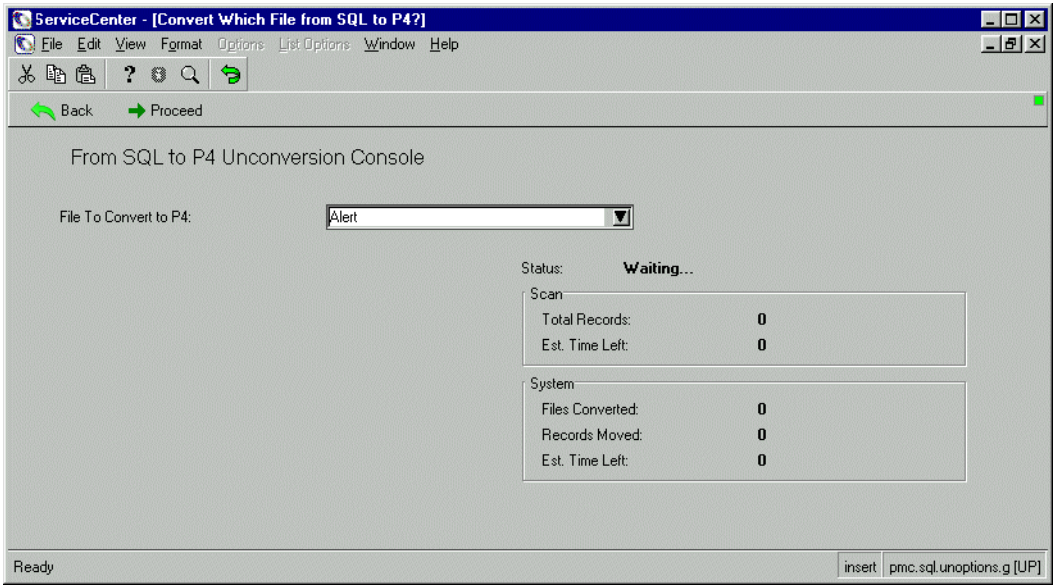


Figure 10-1: Selecting Files to Convert to P4

5 Click **Proceed** or press Enter.

After you convert using *All Files*, the system initiates an automatic shut down, and your session is logged off. If you convert single files, your session is logged out after exiting the SQL Conversion Utility. The system needs to cycle after the files have been converted back into P4 in order to re-establish connectivity to the updated data.

6 Restart the ServiceCenter server.

This restarts all the processes killed at the beginning of the session and establishes links to the converted data.

7 Start a new client session.

APPENDIX A

Characteristics of ServiceCenter Files

This appendix was designed provide a ServiceCenter System or database administrator the information necessary to configure ServiceCenter files for optimal performance. It gives a brief description and the access characteristics of each file, followed by a recommendation for how to organize these files to achieve the best performance.

Topics in this appendix include:

Introduction

ServiceCenter is a system that is easily tailored to meet individual organization requirements, and contains a rich set of application functionality. As not every organization uses every application, and each organization tailors the applications used to meet its requirements, only general recommendations are given.

Extremely Low Usage Files

Normally Empty Files

The following files have no physical database activity in a non-customized system. These are virtual files used to store data internally, and do not have to be converted to the RDBMS format. It causes no harm to convert them but since they will not hold any data it is not necessary. The menucmdlist file never contains data, and is therefore never converted to the RDBMS format. The other files do not contain data unless the system has been customized to use them.

cdbkey	comparedbdict	dbfield
field	icmdecide	keys
linkline	m3eventack	mapappl
menucmdlist	ocmcowork	ocmlwork
status		

Files from Applications that are Not Currently Supported

The following file would be used by the ServiceCenter financial applications, but is not currently supported.

taxcodes

Special Purpose Files

Status of Background Processes

ServiceCenter includes a process that can track the status of other processes in the system, as well as keep itself running. This process uses the following files to define its purpose and to maintain its own status.

File	Description	Activity
anubiscontrol	Reference file to define which processes should be tracked	
anubiseventlog	File used to record events	Insert activity only
anubisstats	File used to track status of other processes	Reference and insert activity
anubissystemstate	File used to track the current state	Update activity

Very Low Activity System Reference Files

The following system reference files generally have low activity.

File	Description
datadump	Contains the specifications for unloading data to be used by the Insight product.
datamap	Contains specifications so that during an unload of data of a primary file, related data from other files is also unloaded. For example, when applications are unloaded, the related formats, formatctrl and link records are also unloaded.

Files Used to Control and Track Changes in the System

The following development files are active if development revision tracking has been turned on.

File	Description
devaudit	Records the changes to items that are being audited. The list of files that are audited is predefined and involves files that deal with application development.
devauditcontrol	Controls what type of revision tracking is taking place. This file will be read once by each process in the system to see if revision tracking is active.

Files Used to Control and Track Changes in the System

The following files are ServiceCenter data files

File	Description
audit	Records changes in field values.
auditspecs	Contains the specifications for ServiceCenter data files and fields within a file that should be audited.

File Used to Monitor Database Activity

The following files is used to monitor database activity.

File	Description	Activity
transactioncount	This file is used to track the type of access being done on all files within the system. As the system is running, it keeps track of these statistics in shared memory. A scenter -reportdbstats command prints the statistics, and causes them to be written to the transactioncount file.	This file has high insert and update activity, but only when a scenter -reportdbdstats command is issued.

Files Used to Gather System Runtime Statistics

The following files are used to gather system runtime statistics.

File	Description
stathistory	Records the number of processes active in the system, as captured by the agent scheduler.
systemmonitor	Captures system monitor information. Used only when the system monitor is used to capture statistics on a scheduled basis. Inserts are then performed each time the job is scheduled to run.

File Used to Save Benchmark Results

File	Description
benchmark	Records the results of the benchmark application.

RDBMS Conversion Files

The following files are only used during RDBMS conversion. They are primarily read-only files with the exception of **sqloptions**, which stores the options chosen during RDBMS conversion. They change only during an upgrade and would only be referenced during RDBMS conversions. Files that are flagged as *obsolete* may safely be deleted from systems 4.0 and later.

sqldbdict - obsolete	sqldbfield - obsolete	sqldbinfo
sqlhints	sqlmapfields - obsolete	sqlmaptables - obsolete
sqloptions	sqlsystemtables	sqltransactionlog
sqlupgrade	sqlwords	

File Used to Display Version Information

The following file is used to display version information.

File	Description
scversion	Used to report RAD and RTE version and licensing options.

Files Associated with Upgrades

The following files are only used when an application upgrade is performed. They should be separated from the main application files. During an upgrade, these files have extremely high activity.

errorlog	patches	signaturemake
signatures	upgdbdict	upgenclapplication
upgrade	upgradehistory	upgradeobjects
upgradepseudolog	upgraderesults	upgradestatus
upgradesystables		

Application Development Files

The following files are reference-only files in a production system. They are RAD programs source files and change only if you have a RAD license or during an upgrades.

analysisalias	applanalysis	application
applicationfields	enclapplication	enclapplrev

Files with Low Activity (Read-only with Caching Involved)

The following files have very low activity during normal daily ServiceCenter operations. They are accessed during system startup operations and for each user during login. These files can be placed in an area away from the primary application files. They are generally read-only and are cached in memory after the initial lead.

System Related Files

The following system related files have low activity.

File	Description	Read
config	Contains the configuration parameters for the system.	Read during process initialization.
datadict	Contains the data policy definitions for each file.	Read from the Database Dictionary only when required, and cached for the process. A given data policy record (for one file) is only read once per user.
erddef	Companion file to joindefs that contains information about how to join files (which fields should be used to join the files). This is a small file that does not frequently change.	Read once at system initialization and cached in shared memory.
info	Global system initialization parameters for the company.	Read during process initialization.
joindefs	Contains information about logical joins defined in ServiceCenter. This is a small file that does not frequently change. Its records are read once at initialization and cached in shared memory.	Read during system initialization and cached in shared memory.
scaccess	Defines which files in the system have mandant protection via a static query that is appended to all other queries against the file. This is a small file that does not frequently change.	Read once at system initialization and cached in shared memory.
scdsites	Contains the definition of the ServiceCenter Distributed Network.	Read once by the first process that needs it and is maintained in shared memory.
scmandant	Defines which files in the system should have mandant protection. This is a small file that does not frequently change.	Read once at system initialization and cached in shared memory.

File	Description	Read
scsecuritygroup	Contains the mandant values to which this user has access.	Read during process initialization.
sctypecheck	Contains specifications on which files should have data type checking done when a new record is inserted or updated. This is a small file that does not frequently change.	Read once at system initialization and cached in shared memory.
SystemEvents	Defines events that can be sent between applications.	Read during process initialization.
techterms	Contains synonyms used in IR Expert parsing.	This file is read into shared memory during initialization.
termtype	Contains the terminal definition for the user.	Read during process initialization.
triggers	Defines the database triggers (RAD applications) that will get invoked during adds/changes/deletes to records in the system. This is a small file that does not frequently change. Generally, this file is referenced with a query that returns all triggers associated with a file (table.name field).	Queried during add/changes/deletes against other files but only accessed once for each file, as the data is cached in shared memory.
tzfile	Contains the timezone definitions. Read during process initialization. This file does not frequently change.	Accessed with a unique query against the name field and saved in shared memory.

Application Related Administrative Files

The following files application related administrative files have low activity.

File	Use	Application
assignment	Assignment group definitions.	Incident Management
availabilitymap	Defines the hours of availability that should be used to track downtime for devices.	Incident Management
category	Defines incident categories	Incident Management

File	Use	Application
ccdef	Single record in file for Service Management to find related Incident Management, Change Management, Request Management, and other Service Management entries when the user clicks show related.	Service Management
cm3groups	Defines message groups.	Change Management
cm3messages	Defines Change Management work flow. This file has a high reference rate by background processes	Change Management
cm3profile	Change Management configuration file that defines options that are valid for the current user	Change Management
cm3ralerts	Defines request alerts.	Change Management
cm3rcategory	Defines request categories.	Change Management
cm3rcatphase	Defines request phases.	Change Management
cm3talerts	Defines task alerts.	Change Management
cm3tcategory	Defines task categories.	Change Management
cm3tcatphase	Defines task phases.	Change Management
cmcontrol	Contains a single record that is read during process initialization.	Contract Management
contractcategory	Defines contract categories	Contract Managment
contracttemplate	Contains copies of existing contracts that can be used as templates when creating new contracts.	Contract Managment
contractterms	Contains standard terms and conditions used by contracts.	Contract Management
ctenv	Contract Management configuration file, which defines the options that are valid for the current user.	Contract Management
ddescript	Configuration file for interfacing with DDE.	system-wide

File	Use	Application
environment	Stores the application (Service Management, Incident Management, Change Management, Request Management, etc.) environmental settings. The file contains a small number of records that are cached in shared memory the first time they are read.	system-wide
group	Defines groups of users who share an application profile.	Incident Management Service Management Inventory Management
icmenv	Inventory configuration profile records.	Inventory Management Configuration Management
inbox	Contains the definition of queries that create queues for Incident Management, Service Management, Change Management, Request Management, Service Level Agreements. These records are read during process initialization and cached. During process initialization numerous inbox records are read to populate combo boxes	system-wide
model	Component and model definitions.	Request Management
modelconfig	Model / inventory configuration. Single record	Request Management
modelvendor	Vendor information for each model.	Request Management
ocmcatselect	Catalog selection categories.	Request Management
ocmevents	Event definition.	Request Management
ocmgroups	Defines approval groups.	Request Management
ocmlcat	Line item categories.	Request Management
ocmocat	Order categories.	Request Management

File	Use	Application
ocmoptions	Phase definition. This file is shared by Requests, Line Items, and Orders.	Request Management
ocmprofile	Defines what options are valid for the current user.	Request Management
ocmqcat	Quote category definition.	Request Management
pmenv	Contains the Incident Management security profile for users or groups. Usually a small number of records that are cached (in shared memory) the first time they are read.	Incident Management
pmstatus	Defines the statuses for incident tickets and what clocks should be started/stopped on entry to the status.	Incident Management
pmtapi	Configuration file used to gather information from a received call TAPI event and fill information in Service Management and Incident Management records.	Service Management Incident Management
querystored	Contains stored (predefined) queries that can be used to restrict queries that users are able to issue.	system-wide
remotecontrol	Single record used to configure Remote Management.	Remote Management
resolution	Standard resolution codes and text.	Service Management Incident Management
screlconfig	Specifications used by applications to find related information in other applications. For example, to find related Incident Management records while in Service Management. Reference file only.	Service Management Incident Management
scripts	Used to create flows that direct user to enter various inputs, piece by piece, rather than by providing a form to be filled in.	system-wide

File	Use	Application
shutdown	Single record used to schedule automatic shutdown of ServiceCenter.	system-wide
sla	Defines the Service Level Agreements within the system. Generally, this is a small number of records.	Service Level Agreements
slaassign	Defines the Service Level Agreement used by the departments for specific categories.	Service Level Agreements
slacontrol	Contains a single record that is read during process initialization.	Service Level Agreements
smenv	Contains the Service Management security profile for a user or group. Usually a small number of records that are cached in the process storage the first time they are read.	Service Management
subcategory	Defines subcategories within each category.	Incident Management
subtotals	Contains specifications for how to total up the cost for a quote and other subtotals.	system wide

Files with Very Low Activity (Generally Read-only)

The following files have very low activity during normal day to day ServiceCenter operations. They are accessed only as needed. These files can be placed in an area away from the primary application files.

File	Description of Activity
alphabet	Used to print banner pages in reports
displayevent	Light reference by each user
displaymaster	Light reference by each user

File	Description of Activity
displayoption	Rarely used, as display information comes from displaycache
displayscreen	Rarely used, as display information comes from displaycache
errmsg	Stores the ServiceCenter Distributed error and informational messages

Files Cached in Shared Memory as Needed

The following files are primarily read-only files that are accessed throughout the session, but the data from these files is cached on both the client and the server. This means, the first time an entry is needed from these files, it is read from the ServiceCenter P4 file system (or an RDBMS). Then it is placed in cache for subsequent reads from the current user or any other user in the system.

File	Description
displaycache	The compilation of displayoptions and formats. This file contains system requirements for displaying data, and the options that should be available to a user.
environment	Stores the application (Service Management, Incident Management, Change Management, Request Management, etc.) environmental settings. The file contains a small number of records that are cached the first time they are read.
formatctrl	Contains settings for system processing or custom form operations.
link	Contains the definitions for how to relate data from more than one physical file.
menu	Contains the definitions and options for menus displayed to users.
pmenv	Contains the Incident Management security profile for a user or group. Usually a small number of records that are cached the first time they are read.
smenv	Contains the Service Management security profile for a user or group. Usually a small number of records that are cached in the process storage the first time they are read.
usergrid	Defines the fields that a user wants displayed in record lists.

High Reference Files

Attention should be given to the following files as needed, even though each record in these files is cached in shared memory when first referenced. Make sure the access to these files is not a full file scan (non-keyed query). ServiceCenter always accesses these files via the file's unique key. (For key definitions, see Volume 2 of Database Management and Administration.)

File	Description
code	This is the executable for a RAD application. Code records are read into shared memory by the first user that requires the code record. Subsequent users will execute the shared memory version of the code. The records in this file are usually large.
dbdict	This is the file that contains the definitions (schema) for other files. It is a small file that does not frequently change. Its records are cached in shared memory as they are accessed. This file is generally accessed via a unique query against the **name** field. The Database Dictionary record is read the first time any file is processed in the system. The results are cached in shared memory.
dtshad	This file contains information about records that have been distributed and files that have been replicated. The replication records are cached but the distributed records are not. Reference is always made via the unique key value.
format	This file contains the layout for the presentation of data and forms to the user. Access is generally via the unique key value for the file. The data is cached, but the number of records and size of those records warrants some attention to make sure file scans are not performed by the RDBMS.
scmessage	This file contains the strings used by ServiceCenter that need to be translated so users are presented with forms and messages in the language of their choice. Records from this file are cached, but the activity is high and consideration should be given to this file to optimize access. Access is generally via the unique key value for the file.

Generally High Activity / High Update files

The following files are always high activity files both in query activity and in update activity. These files and indexes should be separated from each other if possible.

File	Description
clocks	Clocks records are used to track the amount of time required to complete various tasks. Updates to calls, tickets, and change requests involve clocks. This file has a high update rate and grows in size, so its file size needs to be managed.
counters	The counters file is similar to numbers in that both are used to generate unique key values for records. However, counters is managed by the database layer rather than by the application. The value assigned to a counter is unknown to an application program until after the record has been inserted. Counters generate unique key values for schedule records. This file generally has a high update rate, but remains the same size.
dtqueue	Contains transactions that are being distributed in a ServiceCenter Distributed environment. If ServiceCenter Distributed is not used, this file is not used.
number	Used to generate unique key values for various files in the system. Generally this file will be accessed via the unique key on the file and the record is only referenced so that it can be updated. For example, the number file is used to track the next key value used in Incident Management. This file will generally have a high update rate, but remain the same size.
schedule	Schedule records are used by many applications within ServiceCenter. They are used to schedule messages, reports, escalations, etc. All the background processes in ServiceCenter are driven off of schedule records. This file has high reference, update, insert and delete activity.
sqlqueue	Contains transactions that need to be shadowed to an RDBMS database. If RDBMS shadowing is not used this, file is not used.
work	File where applications maintain status for currently active records. This file has a high update rate, and grows in size, so its file size needs to be managed.

Schedule, **dtqueue**, and **sqlqueue** have a high transaction rate. If the system is functioning properly, though, they should not contain large numbers of records. **Dtqueue** and **sqlqueue** records should remain in the files for a maximum time of one minute. The **schedule** file should have a record count equal to the number of open incident, change and ocm tickets. It will only grow in size if something goes wrong. This can cause slow response time. The basic health of the system can be checked by looking in these three files.

Files with High Insert and Deletion Activity

The following files have high insert and delete activity, and should be separated from the main application files. These files are used in printing reports, so the size of the file is highly volatile. When these files are active, the activity is intense. If these files reside on the same physical device as the main application files, the associated activity may impact system performance.

File	Description
spool	Contains the pages of the reports
spoolheader	Contains report header information

Files with Low Insert and Update Activity

The following files have low insert activity.

File	Description
msglog	Receives messages that indicate an action took place in the system. This file can be purged monthly to control its size.
operator	The operator record is updated each time a user logs in.
syslog	Events within the system are written to the syslog file. When a user logs in or logs off an entry is written. Each time a background task starts or ends a record is written. This file can be purged monthly to control its size.

Files with Moderate Reference Activity (Generally Read-only)

The following files have moderate reference activity.

File	Description
agent	Defines the actions and queries issued by the agent background scheduler
caldaily	Contains records used in scheduling operations on a calendar.
inbox	Defines Incident ticket and call information displayed by default for the user.
msgclass	Defines where messages should be directed when specific actions occur in the system.

Infrequently Accessed Reference Files

The following reference files are accessed infrequently.

File	Use	Activity
bulletin	Contains system bulletins.	
caldutyhours	Used in the generation of caldaily.	Static once established.
calholidays	Used in the generation of caldaily.	Static once established.
currency	Contains information on all of the world's currencies, including display format details.	Static once established.
region	Contains world region codes.	Static once established.
country	Contains country codes.	Static once established.
escommand	Reference file that is only used in emergency login situations.	Static
export	Contains the export descriptions for the database export facility.	Used only during data exports.

File	Use	Activity
help	Used for on-line help.	Inserts an update only when help is added or changed. Read when users request help.
helptext	Used for on-line help.	Inserts an update only when help is added or changed. Read when users request help.
import	Contains the import descriptions for the database import facility.	Used only during data imports.
jcl	Contains prototype jcl for OS/390 systems only.	
marquee	Handles the publishing of the marquee entries in the system. This file is processed by a background job and will read and update the entries to indicate they have been processed.	The background process will periodically check to see if any marquees have been added or updated.
msgjcl	Contains prototype JCL for OS/390 systems only.	
msgtext	Contains error messages that may be issued by the OS/390 VSAM interface.	
pageinfo	Reference file that contains information about paging systems.	Rare
report	Specifications for ServiceCenter Report Writer reports.	Used only when Report Writer reports are in use.
reportquery	Contains the query portion of a ServiceCenter Report Writer report, enabling the user to modify report queries at the time the report is run.	Used only when Report Writer reports are in use.
scsearchstep	Specifications used by the search utility.	
typecheck	Contains screen images that are seldom used	Static

File	Use	Activity
unload	Specifications for unloading data from system.	
vsaminfo	Contains information for the vsam background scheduler.	

Primary Application Files

Files Shared by Multiple Applications

The following files are shared by multiple applications:

File	Description
assignment	Individual assignment records are cached in shared memory. However, lists of assignment values for a given operator are retrieved as needed when call queues are displayed.
availability	Keeps track of the availability of devices. Availabilitymap is an associated reference file that defines the hours a device should be available.
cmlabor	Maintains information on the number of hours on which an item was worked to track labor costs for Contract Management.
company	Contains information about companies.
contacts	Contains information about contacts (people).
curconvert	Contains rate information for the conversion between currencies.
currency	Contains information on all of the world's currencies, including display format details.
dept	Definition of departments and their SLA requirements.
device	Primary inventory file contains a record for each item in the inventory. High reference activity. Joined with numerous attribute files based on the type of inventory item. When looking at inventory items there would be retrieval activity against the attribute, devtype, model, outage, contacts, location, vendor, devparent, and pcsoftware files.
expline	Contains detailed information on each expense line for the associated outage costs.

File	Description
globallists	Contains lists of various relatively static data in the system. For example, it contains a list of valid operator IDs. These lists are used to populate combo boxes to provide pick lists. This file is accessed frequently to retrieve value lists. In general, an update to almost any administrative file spawns some activity against the globallist file.
inbox	Defines the default queries that specify the set of records that a user wishes displayed in the queue. A list of inboxes are read cached in the process for combo boxes. However, the specific inbox for a user for a particular application is not cached but is retrieved each time it is needed.
location	Contains information about places.
macro	Used by macro editor for creating and modifying system macros.
macrodef	Used for Macro definitions where initialization and posting expressions are defined.
macroheader	Names of fields affected by macros.
outage	Contains detailed information on a device's outage history.
outagedetail	System or area outages, including date and cost; used by SLA Management.
screlation	Maintains a list of relationships between applications, for example, the list of Incident Management tickets associated with a Service Management call. Frequent queries are done against this table to check for relationships.
servicecontract	Contains details about a service contract.
slaactive	Contains details of SLA activity and responses.
slamonthly	Contains details of SLA monthly performance.
slamonthlyag	Contains details of average SLA monthly performance.
slaresponse	Contains details of SLA performance to activity checks.
SYSBLOB	Bitmaps and OLE containers. This file is primarily a reference file but it may contain a large amount of data. This file should be separated from the primary application files due to its size.
validity	Used primarily by Change and Request Management to confirm that fields in a record conform to specifications. Referenced only during normal operations. Contains a small number of records.

File	Description
vendor	Contains information about providers.
work	File where applications maintain status of current active records.

Change Management

The following files are used be Change Management:

File	Description
cm3r	Contains primary information about a change request. High reference activity. Updates to this file also involve cm3rpage, work, counters, schedule, cm3rcatphase, cm3rcategory, and vendor.
cm3rpage	Contains information about each change that is made to the change request. Generally only referenced by unique value.
cm3t	Contains primary information about a change task. High reference activity.
cm3tpage	Contains information about each change that is made to the change task. Generally only referenced by unique value.

Contract Management

Primary Application Files

The following files are used by Contract Management:

File	Description
contract	The primary contract file, it contains a record for each asset contract. Joined with several attribute files based on the type of contract. When looking at contract items, there could be a retrieval activity against the attribute, contractitem, company, operator, dept, budgetcenter, budget code, language, vendor, contacts, model , occq, ocmo, costcenter, currency, curcontract, payment, and contract terms files.
contract category	Defines the type of contracts (attribute files) within the system.
contractitem	The bridge between the device and contract files. Contains the relationship information that is needed to bind these two files together as well as cost allocation information.

File	Description
contracttemplate	Contains copies of existing contracts that can be used as templates when creating new contracts.
contractterms	A low usage reference file containing standard contract terms and conditions.
payment	Contains information on contract payments. Payments that have been submitted/paid may generate records in the expline file.
softwarecounter	Used to check for software compliance by counting software installations and software licenses.

Attribute Files

contractlease	contractsoftware	contractsupport
contractmaintenance	contractwarranty	

Event Services

The following files are used by Event Services. Event Services runs in the background and processes events into and out of ServiceCenter. These events update information in the system, such as adding inventory items and creating Incident Management tickets. These files are accessed at the same time and are the primary files for the application receiving or sending the event.

File	Description	Activity
distgroup	Reference file	
eventfilter	Reference file	
eventin	Contains input events	High insert/delete activity.
eventmap	Reference file to control the mapping of data.	
eventout	Contains output events	High insert/delete activity
eventregister	Reference file	
goeconfig	Reference file	

Incident Management

File	Description
cmparts	Keeps track of the parts and labor portion of processing a ticket for contract management.
downtime	Track device downtime. Update activity.
pmcost	Keeps track of the cost associated with a ticket. This file is added or updated by a background task that is processing expense lines (expline) records for contract management. Files updates at the same time as pmcost are expline and servicecontract.
pmnotes	File where users record notes regarding a ticket.
problem	Primary file that keeps track of tickets. High update and insert activity.
probsummary	One entry for each Incident Management Ticket. Summarizes the information from the problem pages. High reference activity in addition to updates and inserts. Opening, updating or closing a ticket will also cause updates to device, outage, screlation, servicecontract, counters, schedule, clocks, work, problem.
activity	Optional. One entry for every update to an incident ticket.

Inventory Management

Primary Application Files

The following files are used by Inventory Management

File	Description
device	Primary inventory file contains a record for each item in the inventory. High reference activity. Joined with numerous attribute files based on the type of inventory item. When looking at inventory items there could be retrieval activity against the assignment, company, contractitem, costcenter, devparent, location, pcsoftware, unitofmeasure, and vendor files.
devtype	Defines the types of devices (attribute files) within the system.
pcsoftware	Defines the software present in an organization (version, license information, vendor, etc.)

Attribute Files		
computer	displaydevice	example
furnishings	handhelds	mainframe
networkcomponents	officeelectronics	softwarelicense
storage	telecom	

IR Expert

The files used for IR Expert searches must be treated carefully, especially if the IR queries being run combine an IR Expert search with a QBE search. With a combined search, the IR Engine selects a set of records, then each of these records is individually retrieved from the RDBMS database via the unique key, and the QBE qualification is checked. This can present a lot of activity against these files. Therefore, they should be isolated so the I/O done against these files does not interfere with the primary application data files.

File	Description
core	IR Expert file created from other knowledge sources. Central file where all IR Expert knowledge (knowledge, KnowledgePak, probsummary, etc.) is held.
keyword	The key words that are relevant to probable cause determination.
knowledge	IR Expert file from KnowledgeBroker.
KnowledgePak	IR Expert file from ServiceWare Inc.
probcause	IR Expert file set up with known probable causes for incidents.
protocore	Temporary holding area until the knowledge engineer reviews the records and either deletes them or moves then into the core file.

Request Management

File	Descriptions
ocml	Line item pertaining to a quote (ocmq)
ocmlpage	Contains details of each change to a line item.
ocmlrec	Request management receiving log.
ocmo	Primary file containing orders. High reference activity. An update of this file also hits schedule and counters.
ocmopage	Contains details about each change to an order.
ocmphaselog	Request Management phase log. Keeps track of the phases of requests, line items, and orders.
ocmq	Primary file containing quotes. High reference activity. An update of this file also hits schedule, counters, and ocmapprlog. Inserts of new quotes hits numbers, scripts, ocmphaselog, and ocmalertlog.
ocmqpage	Contains details of each change to an order.

Service Management

File	Descriptions
incdepends	Contains dependencies that a call must meet before the call can be closed. Dependencies are created and modified by the user.
incidents	Contains the data relating to an incoming call. Generally high reference activity, high insert activity, high update activity. Any query against incidents will also involve contacts and incdepends. Any update against incidents will also involve clocks. Any insert against incidents will also involve SLA, dept, cmlabor, expline, caldaily, counters, number, and screlation, if these files are used in your installation.

Work Management

Work Management exclusively uses the following files. If Work Management is not used, these files can be ignored.

Reference Files

File	Description
wdCategory	A mirror of records from **category**, **cm3tcategory** and **cm3rcategory** files. A reference file during normal operation, but is modified if any of the source files are modified.
wdCodeDetails	A reference file during normal operation.
wdCriteria	Reference file
wdCustomReports	Reference file during normal operations.
wdCustomSections	Reference file during normal operations.
wdPriorityLookup	Keeps track of Work Management priorities. A reference file during normal operations.
wdQueueValues	Reference file during normal operations.
wdShowContact	Reference file during normal operations

Files with Update Activity

File	Description
Customize	A record is inserted into this file each time a new user logs into Work Management. This record is updated if the user changes any Work Management options.
wdOffHours	Stores the operator commitments. This file has update/insert/delete activity.
wdResExp	Stores operator expertise. This file has update/insert/delete activity.
wdResHierarchy	Defines manager-operator information for Work Management. Modified whenever the assignment definition is modified in ServiceCenter, change message group information is modified in ServiceCenter, or sharing is defined for operators
wdSchFilter	A record is inserted into this file each time a new user logs into Work Management. This record is updated if the user changes any Work Management options.

File	Description
wdSchOptions	A record is inserted into this file each time a new user logs into Work Management. This record is updated if the user changes any Work Management options.
wdTauCounter	Keeps track of running counters.

Recommended Placement of ServiceCenter Files

The following section provides general recommendations for file placement to increased performance. It keys off file characteristics detailed in the previous section.

Note: For DB2 customers a *tablespace* equates to a database - a database for files that are empty, so these tables do not take up space in the DBD or the EDM pool.

Tablespace for Files that are Empty or Unused

For those files in ServiceCenter that are virtual files (never contain data) or are just not used, they can either be left in P4, or RDBMS tables can be created for them.

The following files fall into this category:

- *Normally Empty Files* on page 246
- *Files from Applications that are Not Currently Supported* on page 246

Tablespace for Special Purpose Files

The following sets of files should be looked at and, if the feature is being used, these files should be put into their own tablespaces. If the feature is not being used, the files can go into the same tablespaces as is used for the files above.

- *Status of Background Processes* on page 247
- *Files Used to Control and Track Changes in the System* on page 248
- *File Used to Monitor Database Activity* on page 248
- *Files Used to Gather System Runtime Statistics* on page 249
- *File Used to Save Benchmark Results* on page 249
- *File Used to Display Version Information* on page 249

Tablespace for Routinely Read-only and Rarely Used File

Files Associated with Upgrades

applanalysis	analysisalias	application
applicationfields	enclapplication	enclapplrecv
textfields		

Files that aid in RDBMS conversions

sqldbinfo	sqlhints	sqlsystemtables
sqltransactionslog	sqlupgrade	sqlwords

Files with low activity

abdetcodes	abendcodes	alphabet
datadump	datamap	displayevent
displaymaster	displayoption	displayscreen
errmsg		

Tablespace for Administrative and Setup Files

The following files are relatively small. They are setup and administrative files, as such they do not often change. Many of these are read-only once and then cached; therefore, the read activity is low.

assignment	availability	bulletin
caldutyhours	calholidays	category
ccdef	cm3groups	cm3messages
cm3profile	cm3ralerts	cm3rcategory
cm3rcatphase	cm3talerts	cm3tcategory
cm3tcatphase	cmcontrol	config
contractcategory	contracttemplate	contractterms
country	ctenv	currency

datadict	ddescript	distgroup
environment	erddef	escommand
eventfilter	eventmap	eventregister
export	goeconfig	group
help	helptext	icmenv
import	inbox	info
jcl	joindefs	macrodef
marquee	model	modelconfig
modelvendor	msgjcl	msgtext
ocmcatselect	ocmevents	ocmgroups
ocmlcat	ocmocat	ocmooptions
ocmprofile	ocmqcat	pageinfo
pmenv	pmstatus	pmtapi
querystored	region	remotecontrol
report	reportquery	resolution
scaccess	scdsites	scmandant
screlconfig	scripts	scsearchstep
scsecuritygroup	sctypecheck	shutdown
sla	slaassign	slacontrol
smenv	softwarecounter	subcategory
subtotal	systemevents	techterms
termtype	triggers	typecheck
tzfile	unload	vsaminfo

Tablespace of High Usage Reference Files

The following files have a high rate of reference activity. As an entry is read, it is cached in shared memory.

agent	caldaily	code
dbdict	displaycache	format

globallists	inbox	link
macro	macroheader	menu
msgclass	scmessage	usergrid

Tablespace for Upgrade Files

Files associated with the upgrade process can be isolated in a tablespace of their own. When an upgrade is performed, the activity on these files is intense, but at other times the files remain untouched.

errorlog	patches	signaturemake
signatures	upgrade	upgradehistory
upgradeobjects	upgradepseudolog	upgraderesults
upgradestatus	upgdbdict	upgenclapplication
upgradesystables		

Tablespace for Spool Data and Logs

The spooling files are active during reporting sessions, but otherwise inactive. They have high insert and delete activity and, as they are used for reporting, have a tendency to become large, then shrink down periodically.

Spool	spoolheader	Syslog	msglog

Tablespace for Event Data

The event files are guaranteed to be accessed at the same time as other files (target for the event) and should be in separate tablespaces if possible.

Eventin	eventout

Tablespaces for Inventory Data

Inventory data is accessed by all components of the system; therefore, you should consider separating the inventory data from the other files. In addition, the attribute files that are heavily used should be separated from the **device** file, as the **device** file and associated attribute file will always be accessed concurrently.

computer	device	displaydevice
example	furnishings	handhelds
mainframe	networkcomponents	officeelectronics
pcsoftware	softwarelicense	storage
telecom		

Tablespaces for High Activity Files

The following files have high update activity. Consider placing these files in a tablespace of their own. Also, if possible these files should be organized such that there is a single record per page to increase concurrency. When page level locking is active, processes can complete without waiting for other processes, even when dealing with different records.

clocks	counters	dtqueue - (if using ServiceCenter Distributed)
dtshad	number	schedule
sqlqueue - (if using RDBMS shadowing)	work	

Files of Variable Activity

The remaining files are all files that are directly related to a specific feature of ServiceCenter. If you are using the feature, the files have a high usage. If you are not using the feature, the files have little or no activity. The tools available in ServiceCenter (see the appropriate *Tips* section on the following pages) or provided by the target RDBMS should be used to determine the frequency and concurrency of use of the files.

ServiceCenter System Files

The following files are of variable activity:

assignment	availability	cmlabor
company	contacts	curconvert
dept	expline	inbox
location	outage	outagedetail
screlation	slaactive	slamonthly
slamonthlyag	slaresponse	SYSBLOB
validity	vendor	

Change Management Files

The following files are of variable activity:

cm3r	cm3rpage	cm3t	cm3tpage

Contract Management Files

The following files are of variable activity:

contract	contractlease	contractmaintenance
contractsoftware	contractsupport	contractwarranty
payment		

Incident Management Files

The following files are of variable activity:

pmnotes	problem	probsummary

Request Management Files

The following files are of variable activity:

ocml	ocmlpage	ocmlrec

ocmo	ocmopage	ocmphaselog
ocmq	ocmqpage	

Service Contract Files

The following files are of variable activity:

cmparts	downtime	pmcost

Service Management Files

The following files are of variable activity:

incidents	incdepends

Work Management Files

The following files are of variable activity:

wdCategory	wdCodeDetails	wdCriteria
wdCustomize	wdCustomReports	wdCustomSections
wdOffHours	wdPriorityLookup	wdQueueValues
wdResExp	wdResHierarchy	wdSchFilter

IR Expert/Knowledge Engineering Files

The following files are of variable activity:

core	keyword	knowledge
KnowledgePak	probcause	protocore

Tools to Determine File Access Characteristics

The easiest way to identify files used and files updated by ServiceCenter application processing is to use the ServiceCenter trace utilities. ServiceCenter keeps statistics on the type of access done against each file.

This data is kept in shared memory and can be accessed via the following command. This command lists all files accessed by the system during processing:

scenter - reportdbstats

Sample trace of an Incident Management transaction

Filename	selects	Inserts	Updates	Deletes	Counts	Sorts	Finds
format	0	0	0	0	0	0	22
triggers	31	0	0	0	0	0	0
SYSBLOB	1	0	0	0	0	0	0
dtshad	0	0	0	0	0	0	46
schedule	2	10	0	0	0	0	0
location	1	0	0	0	0	0	0
sla	4	0	0	0	0	0	0
macroheader	1	0	0	0	0	0	0
dbdict	31	0	0	0	0	0	0
work	2	2	0	0	0	0	0
link	0	0	0	0	0	0	6
servicecontract	2	0	2	0	0	0	0
assignment	4	0	0	0	0	0	2
probsummary	4	2	0	0	0	0	0
subcategory	4	0	0	0	0	0	0
clocks	8	5	1	0	0	0	0
transactioncount	0	0	37	0	0	0	37
datadict	0	0	0	0	0	0	13
expline	0	1	0	0	0	0	0
formatctrl	0	0	0	0	0	0	3
device	5	0	1	0	0	0	0
caldaily	5	0	0	0	0	0	0
globallists	1	0	0	0	0	0	0
menu	0	0	0	0	0	0	1
pmstatus	6	0	0	0	0	0	0
techterms	1	0	0	0	0	0	0
cmparts	1	1	0	0	0	0	0
displaymaster	1	0	0	0	0	0	0
usergrid	0	0	0	0	0	0	4
dept	2	0	0	0	0	0	0
contacts	6	0	0	0	0	0	0

goeconfig	1	0	0	0	0	0	0
number	2	0	2	0	0	0	0
counters	8	0	13	0	0	0	13
displaycache	0	0	0	0	0	0	3
scmessage	0	0	0	0	0	0	120
code	0	0	0	0	0	0	64
outage	1	1	0	0	0	0	0
model	1	0	1	0	0	0	0
problem	1	2	0	0	0	0	0
category	1	0	0	0	0	0	2

Using this report, administrators can identify the files accessed during typical processing.

The strategy to identify the files is quick and should be done prior to crossing over from a test to a production system.

To Identify Files:

1. Start ServiceCenter.
2. Begin a full day of typical processing. Include updates to supporting application areas such as adding new users, adding new categories, locations.
3. Run the reportdbstats report.
4. Inspect the report for all files that have been updated (add, delete, update).
5. Exclude files such as transactioncount that do not contain user data.
6. Consider excluding files such as the event and schedule files.
7. Map the files to the relational database

   Administrators can run the following command to find the files that have been updated since the last report using the following command:

   scenter - reportdbstats:2

   This command should be run frequently to test new transactions as they are identified.

Note: Each ServiceCenter implementation is different and therefore, if the customer site is not going to fully map the file system, it is up to the customer to identify the files used by their implementation using the technique described above.

If it is important to see the sequence of access to files, use the **debugdbquery** parameter in the **sc.ini** file to track all database access performed by any process.

A specification of **debugdbquery:999** causes the system to write an entry in the log for each database access. The log entries all begin with *DBACCESS*.

- DBACCESS identifies the start of a database event.
- DBACCESS* identifies the end of a database event.
- DBACCESS! identifies database events that resulted in a database update.

Some database operations are recursive when TRIGGERS are involved. The > symbols in the sample log below indicate the active levels of recursion.

Sample debugdbquery:999 log:

```
387 03/19/99 07:53:06 DBACCESS - Insert against file schedule
387 03/19/99 07:53:06 >DBACCESS - Select against file dbdict
387 03/19/99 07:53:06 >DBACCESS* - Select completed against file
dbdict in 0.050000 seconds
387 03/19/99 07:53:06 >DBACCESS - Select against file triggers
387 03/19/99 07:53:06 >DBACCESS* - Select completed against file
triggers in 0.010000 seconds
387 03/19/99 07:53:06 >DBACCESS - Initialization against file
counters
387 03/19/99 07:53:06 >DBACCESS - Select against file counters
387 03/19/99 07:53:06 >DBACCESS* - Select completed against file
counters in 0.110000 seconds
387 03/19/99 07:53:06 >DBACCESS - Find against file counters
387 03/19/99 07:53:06 >DBACCESS* - Find completed against file
counters in 0.020000 seconds
387 03/19/99 07:53:06 >DBACCESS - Update against file counters
387 03/19/99 07:53:06 >>DBACCESS - Initialization against file erddef
387 03/19/99 07:53:06 >>DBACCESS - Select against file erddef
387 03/19/99 07:53:06 >>DBACCESS* - Select completed against file
erddef in 0.060000 seconds
387 03/19/99 07:53:06 >>DBACCESS - Select against file dbdict
387 03/19/99 07:53:06 >>>DBACCESS - Termination against file erddef
387 03/19/99 07:53:06 >>DBACCESS* - Select completed against file
dbdict in 0.130000 seconds
387 03/19/99 07:53:06 >>DBACCESS - Select against file triggers
387 03/19/99 07:53:06 >>DBACCESS* - Select completed against file
triggers in 0.020000 seconds
387 03/19/99 07:53:06 >>DBACCESS - Initialization against file dtshad
387 03/19/99 07:53:06 >>DBACCESS - Find against file dtshad
387 03/19/99 07:53:06 >>DBACCESS* - Find completed against file
dtshad in 0.070000 seconds
387 03/19/99 07:53:06 >>DBACCESS - Find against file dtshad
387 03/19/99 07:53:06 >>DBACCESS* - Find completed against file
dtshad in 0.010000 seconds
```

```
387 03/19/99 07:53:06 >DBACCESS! - Update completed against file
counters in 0.431000 seconds
387 03/19/99 07:53:07 >DBACCESS - Find against file dtshad
387 03/19/99 07:53:07 >DBACCESS* - Find completed against file dtshad
in 0.020000 seconds
387 03/19/99 07:53:07 >DBACCESS - Find against file dtshad
387 03/19/99 07:53:07 >DBACCESS* - Find completed against file dtshad
in 0.010000 seconds
```
387 03/19/99 07:53:07 DBACCESS! – Insert completed against file schedule in 1.061000 seconds

Displaying Results of ServiceCenter Caching

ServiceCenter uses shared memory to cache frequently used records from the database. The use of caching means that the first time something is done the system remembers so that all subsequent times the same thing is done it is performed faster. ServiceCenter uses shared memory to cache entries from the database that are likely to have a high usage. This is done both on the server and on the client. The ServiceCenter applications also attempt to reduce the number of times they need to read data by using global variables, allowing data to be read a single time and referenced by multiple application programs.

Effect of Caching on Login

Below is a sample of the database access for the first user to login to ServiceCenter. Entries in ***bold italic*** represent database requests that only get issued for the first login.

First User to Log In

```
Initialization against file dbdict
Initialization against file tzfile
Termination against file tzfile
Initialization against file code
Initialization against file format
Initialization against file link
Select against file dbdict
Initialization against file triggers
Select against file triggers
Initialization against file SystemEvents
Select against file SystemEvents
Termination against file SystemEvents
Find against file code
Select against file dbdict
Select against file triggers
```

```
Initialization against file config
Find against file config
Termination against file config
Select against file dbdict
Select against file triggers
Initialization against file termtype
Find against file termtype
Termination against file termtype
Select against file dbdict
Select against file triggers
Initialization against file info
Select against file info
Select against file dbdict
Select against file triggers
Initialization against file scmessage
Find against file scmessage
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file globallists
Select against file globallists
Termination against file globallists
Find against file format
Find against file scmessage
Initialization against file scmessage
Select against file scmessage
Initialization against file info
Select against file info
Select against file dbdict
Select against file triggers
Initialization against file operator
Select against file operator
Select against file dbdict
Select against file triggers
Initialization against file scsecuritygroup
Find against file scsecuritygroup
Termination against file scsecuritygroup
Termination against file info
Termination against file operator
Find against file code
Initialization against file operator
Select against file operator
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Termination against file scmessage
Find against file code
Find against file code
```

```
Select against file dbdict
Select against file triggers
Initialization against file cmcontrol
Select against file cmcontrol
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file environment
Find against file environment
Select against file dbdict
Select against file triggers
Initialization against file smenv
Select against file smenv
Initialization against file environment
Find against file environment
Select against file dbdict
Select against file triggers
Initialization against file pmenv
Find against file pmenv
Select against file dbdict
Select against file triggers
Initialization against file category
Initialization against file globallists
Select against file globallists
Initialization against file globallists
Select against file globallists
Find against file format
Find against file code
Find against file code
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file inbox
Select against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Initialization against file inbox
Select against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Initialization against file inbox
```

```
Select against file inbox
Termination against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Initialization against file inbox
Select against file inbox
Termination against file inbox
Find against file format
Find against file code
Find against file format
Find against file code
Termination against file globallists
Find against file code
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Initialization against file environment
Initialization against file operator
Select against file dbdict
Select against file triggers
Initialization against file cm3profile
Select against file dbdict
Select against file triggers
Initialization against file cm3profilegrp
Find against file environment
Select against file cm3profile
Select against file cm3profile
Find against file code
Termination against file operator
Termination against file environment
Termination against file cm3profilegrp
Initialization against file environment
Initialization against file operator
Initialization against file cm3profile
Initialization against file cm3profilegrp
Find against file environment
Select against file cm3profile
Select against file cm3profile
Termination against file operator
Termination against file environment
Termination against file cm3profilegrp
Termination against file globallists
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file slacontrol
Select against file slacontrol
```

```
Find against file code
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Initialization against file environment
Select against file dbdict
Select against file triggers
Initialization against file ocmprofile
Initialization against file operator
Select against file operator
Find against file environment
Find against file code
Find against file scmessage
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file ocmgroups
Select against file ocmprofile
Select against file ocmprofile
Termination against file ocmgroups
Find against file code
Termination against file environment
Termination against file operator
Find against file code
Initialization against file environment
Initialization against file ocmprofile
Initialization against file operator
Select against file operator
Find against file environment
Initialization against file ocmgroups
Select against file ocmprofile
Select against file ocmprofile
Termination against file ocmgroups
Find against file code
Termination against file environment
Termination against file operator
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Initialization against file environment
Initialization against file ocmprofile
```

```
Initialization against file operator
Select against file operator
Find against file environment
Initialization against file ocmgroups
Select against file ocmprofile
Select against file ocmprofile
Termination against file ocmgroups
Find against file code
Termination against file environment
Termination against file operator
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file syslog
Select against file dbdict
Select against file triggers
Initialization against file datadict
Find against file datadict
Termination against file datadict
Insert against file syslog
Select against file dbdict
Select against file triggers
Initialization against file counters
Select against file counters
Initialization against file erddef
Select against file erddef
Select against file dbdict
Termination against file erddef
Select against file triggers
Initialization against file dtshad
Find against file dtshad
Find against file dtshad
Find against file dtshad
Termination against file syslog
Find against file code
Find against file scmessage
Find against file scmessage
Initialization against file tzfile
Find against file tzfile
Initialization against file tzfile
Find against file code
Find against file scmessage
Select against file dbdict
Termination against file tzfile
Select against file triggers
```

```
Initialization against file mail
Select against file mail
Count against file mail
Termination against file mail
Find against file code
Find against file code
Initialization against file info
Select against file info
Select against file info
Termination against file info
Initialization against file datadict
Find against file datadict
Termination against file datadict
Update against file operator
Initialization against file operator
Find against file code
Initialization against file operator
Find against file code
Find against file dtshad
Find against file dtshad
Find against file dtshad
Select against file dbdict
Select against file triggers
Initialization against file formatctrl
Find against file formatctrl
Find against file formatctrl
Find against file code
Find against file code
Initialization against file operator
Initialization against file operator
Termination against file operator
Find against file code
Termination against file info
Termination against file formatctrl
Find against file format
Termination against file operator
Termination against file operator
Termination against file operator
Find against file format
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
```

```
Initialization against file info
Select against file info
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file menu
Find against file menu
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file format
Initialization against file datadict
Find against file datadict
Termination against file datadict
Find against file scmessage
Find against file scmessage
```

All finds performed against *code, format, dtshad, datadict, triggers, formatctrl, menu,* and *scmessage* files are not done for subsequent users. They are not done because the original results were cached in shared memory. Each find that is eliminated means that searches against the data using the *unique* key value for that file have been eliminated. Once a file has been initialized the overhead of subsequent initializations is much less because the dbdict file record is cached. This is especially true of files mapped to SQL, as the SQL DESCRIBE of the tables is only done once.

Effect of Caching on Incident Management QuickOpen Function

In another example of caching, the report below shows activity that takes place when a Quick Open is performed in Incident Management. As with the login example, the difference between the first Quick Open and subsequent Quick Opens is shown. Entries in ***bold italic*** represent actions that are only done the first time. Either the data is cached globally in shared memory or locally within the process.

Note: Access to code, scmessage, triggers, displaycache, dtshad, datadict, dbdict, counters, and format has been eliminated.

First QuickOpen in Incident Management

```
Initialization against file menu
Find against file menu
Find against file scmessage
Find against file format
Termination against file menu
Find against file format
Find against file code
Find against file format
Find against file format
Find against file format
Initialization against file menu
Find against file code
Find against file scmessage
Find against file scmessage
Select against file dbdict
Initialization against file triggers
Select against file triggers
Initialization against file problem
Initialization against file problem
Initialization against file formatctrl
Find against file formatctrl
Initialization against file problem
Find against file code
Initialization against file problem
Termination against file problem
Find against file code
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file number
Select against file number
Initialization against file datadict
Find against file datadict
Termination against file datadict
Update against file number
Initialization against file number
Find against file dtshad
Find against file dtshad
Find against file dtshad
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
```

```
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file displaymaster
Select against file displaymaster
Initialization against file problem
Select against file dbdict
Select against file triggers
Initialization against file displayscreen
Select against file dbdict
Select against file triggers
Initialization against file displayoption
Select against file dbdict
Select against file triggers
Initialization against file displaycache
Find against file code
Termination against file number
Termination against file number
Find against file displaycache
Termination against file problem
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file scmessage
Find against file format
Initialization against file datadict
Find against file datadict
Termination against file datadict
Find against file link
Select against file dbdict
Select against file triggers
```

```
Initialization against file contacts
Select against file contacts
Find against file format
Termination against file contacts
Select against file dbdict
Select against file triggers
Initialization against file device
Select against file device
Find against file format
Termination against file device
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file subcategory
Select against file subcategory
Termination against file subcategory
Initialization against file subcategory
Select against file subcategory
Termination against file subcategory
Termination against file displayscreen
Termination against file displaycache
Termination against file displayoption
Initialization against file category
Find against file category
Find against file code
Initialization against file globallists
Select against file globallists
Termination against file globallists
Initialization against file problem
Termination against file problem
Find against file code
Find against file code
Initialization against file category
Select against file dbdict
Select against file triggers
Initialization against file assignment
Find against file assignment
Find against file code
Find against file scmessage
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file cmparts
Termination against file cmparts
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file cmlabor
Termination against file cmlabor
```

```
Insert against file problem
Select against file counters
Find against file dtshad
Initialization against file problem
Find against file code
Find against file scmessage
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file macroheader
Select against file macroheader
Initialization against file category
Initialization against file assignment
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file schedule
Select against file schedule
Termination against file schedule
Find against file code
Find against file code
Termination against file problem
Find against file code
Find against file code
Find against file scmessage
Find against file scmessage
Find against file scmessage
Select against file dbdict
Select against file triggers
Initialization against file caldaily
Select against file caldaily
Find against file code
Initialization against file schedule
Initialization against file datadict
Find against file datadict
Termination against file datadict
Insert against file schedule
Select against file counters
Find against file counters
Update against file counters
Find against file dtshad
Find against file dtshad
Find against file dtshad
Find against file dtshad
Find against file dtshad
Find against file dtshad
Termination against file schedule
Find against file assignment
Termination against file caldaily
```

```
Initialization against file caldaily
Select against file caldaily
Initialization against file schedule
Initialization against file datadict
Termination against file datadict
Insert against file schedule
Find against file counters
Update against file counters
Termination against file schedule
Initialization against file assignment
Select against file assignment
Termination against file assignment
Initialization against file problem
Find against file code
Find against file scmessage
Find against file code
Find against file scmessage
Find against file code
Initialization against file category
Initialization against file assignment
Select against file assignment
Termination against file assignment
Termination against file category
Find against file code
Initialization against file schedule
Initialization against file datadict
Termination against file datadict
Insert against file schedule
Find against file counters
Update against file counters
Find against file code
Initialization against file problem
Select against file dbdict
Select against file triggers
Initialization against file probsummary
Select against file probsummary
Initialization against file link
Find against file link
Initialization against file datadict
Find against file datadict
Termination against file datadict
Insert against file probsummary
Select against file counters
Find against file dtshad
Initialization against file probsummary
Find against file code
Termination against file caldaily
Find against file code
Select against file dbdict
```

```
Select against file triggers
Initialization against file pmstatus
Select against file pmstatus
Find against file code
Select against file pmstatus
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file clocks
Select against file clocks
Initialization against file datadict
Find against file datadict
Termination against file datadict
Insert against file clocks
Select against file counters
Find against file dtshad
Find against file dtshad
Find against file dtshad
Termination against file clocks
Initialization against file clocks
Select against file clocks
Initialization against file datadict
Termination against file datadict
Insert against file clocks
Termination against file clocks
Termination against file pmstatus
Find against file code
Initialization against file schedule
Insert against file schedule
Find against file counters
Update against file counters
Termination against file schedule
Find against file scmessage
Initialization against file probsummary
Find against file code
Find against file code
Find against file code
Select against file dbdict
Select against file triggers
Initialization against file work
Select against file work
Initialization against file link
Find against file link
Initialization against file datadict
Find against file datadict
Termination against file datadict
Insert against file work
Select against file counters
Initialization against file work
```

```
Find against file code
Find against file dtshad
Find against file dtshad
Find against file dtshad
Termination against file link
Find against file dtshad
Find against file dtshad
Termination against file problem
Initialization against file problem
Termination against file problem
Initialization against file probsummary
Select against file probsummary
Find against file dtshad
Find against file dtshad
Find against file dtshad
Find against file dtshad
Initialization against file problem
Initialization against file problem
Termination against file probsummary
Initialization against file problem
Termination against file problem
Initialization against file number
Select against file number
Initialization against file datadict
Termination against file datadict
Update against file number
Initialization against file number
Initialization against file problem
Initialization against file displayscreen
Initialization against file displayoption
Initialization against file displaycache
Termination against file number
Termination against file number
Termination against file problem
Initialization against file contacts
Termination against file contacts
Initialization against file device
Termination against file device
Initialization against file subcategory
Select against file subcategory
Termination against file subcategory
Initialization against file subcategory
Select against file subcategory
Termination against file displayscreen
Termination against file displaycache
Termination against file displayoption
Find against file scmessage
Find against file code
Termination against file problem
```

```
Termination against file category
Termination against file assignment
Termination against file category
Termination against file displaymaster
Termination against file problem
Termination against file schedule
Termination against file problem
Termination against file formatctrl
Termination against file subcategory
Termination against file problem
Termination against file problem
Termination against file problem
Termination against file work
Termination against file work
```

APPENDIX B

Initialization Parameters for RDBMS

This appendix was designed to give ServiceCenter database administrators and explanation of the ServiceCenter Relational Database Management System (RDBMS) startup parameters. RDBMS initialization parameters control how the SQL Interface interacts with the RDBMS. Because some of these settings control security and connectivity issues, you may need to consult the DBA and operations personnel. The SQL Interface initialization parameters are specified in the same file (sc.ini) as the ServiceCenter initialization parameters.

Important: SQL parameters added to the sc.ini file are evaluated for each new user logging into the system. It may be necessary to cycle a client before newly added parameters take affect.

Parameters

This defines the current list of initialization parameters used to configure the ServiceCenter sc.ini file.

For online information about these parameters, enter -helpsql at the Command prompt.

Parameter	Definition
[db2mvs] [oracle] [sqlserver] [sybase]	Represent the name of a database type. They are used to demarcate sections for different database types within the sc.ini file to allow for more than one sqldb parameter, thus enabling connection to more than one RDBMS. The names inside the brackets must match the SQL dbtype used when converting a file to the RDBMS. The parameters for each database type are entered after the database type name. These optional parameters are for Windows only. They are entered in the server sc.ini file. A sample initialization file might include: [oracle8xi] ... [sqlserver2k]
immediateshadow	Indicates that the SQL Shadowing functions should occur at the same time as the P4 update. Asynchronous shadowing is the default value when the SQL function is queued for the sqlqueue process. The sqlqueue process is started with scenter -queLsql. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 0. Any value except 0 turns the parameter on. 0 - Immediateshadow is turned off. (default) 1 - Immediateshadow is turned on. A sample initialization file might include: immediateshadow:1

Parameter	Definition
language:<language name>	Language for localized ServiceCenter interface. Default is English. Add this parameter to the odbc sc.ini file to display odbc data in a foreign language. Specifies the language resource table that will be loaded as part of the ServiceCenter interface. Possible values: language identifiers (for example, en, fr, or gr). Refer to the ServiceCenter *Technical Reference* for a complete list of available language identifiers.
odbccharacterarrays:*0*	Indicates that ODBC should treat arrays of characters as tables.
sqlautosort:*0*	Is used to determine if the dbDict keys should be analyzed against the query when deciding what fields should be used in an SQL sort. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - Autosort is turned off. No automatic sort based on the query and the dbDict keys is to be done. (default) 1 - Autosort is turned on. The system automatically picks a sort key based on the fields used in the query. A sample initialization file might include: sqlautosort:1
sqlbatchcount:*32*	Specifies the number of records ServiceCenter should request from the relational database with a single select call. This parameter can reduce the number of interactions between ServiceCenter and the relational database system. Rather than selecting and fetching each record for a QBE list individually, the specified number of records is selected with a single SELECT statement. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 64. With this sample initialization file entry, ServiceCenter would retrieve the first 32 complete record from the RDBMS with every query. sqlbatchcount:32

Parameter	Definition
sqlcascade:*1*	Indicates that you want cascading deletes. It is only valid if cascading deletes are supported by your RDBMS. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - Cascading deletes are turned off. (default) 1 - Cascading deletes are turned on. A sample initialization file might include: sqlcascade:1
sqldb:*database_specification*	Required. Designates the database to be connected to. For DB2/OS/390, database_specification is the name of the DB2 subsystem. For example, to connect to the subsystem called *db31*, enter: sqldb:db31 For Oracle, database_specification is the connection string. For example, to connect to a local database called *HDSK* using the SQL*Net V1 Oracle Pipe driver, enter: sqldb:P:HDSK or, to connect to a local or remote database using SQL*Net V2, enter: sqldb:HDSK For Sybase, database_specification is the name of the database server. For example, to connect to a Sybase database server called *helpdesk*, enter: sqldb:helpdesk For Informix,database_specification is the name of the database. The database server is set by the INFORMIXSERVER environmental variable. For example, to connect to an Informix database called *helpdesk*, enter: sqldb:helpdesk For SQL Server,database_specification is the name of the database server. For example, to connect to a SQL server database server called *helpdesk*, enter: sqldb:helpdesk This parameter requires a restart of the system before it takes effect.

Parameter	Definition
sqldb:*DBNAME*	Designates the database to be connected to. It is required for database connectivity. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include all valid dbnames, and vary according to which RDBMS is being used. See the examples below for database specific information. **Note:** The sqldb parameter requires a restart of the system before it takes effect. **DB2** DBNAME is the name of the DB2 subsystem. Using this sample ini file entry would connect to the subsystem called *db31*: sqldb:db31 **Informix** DBNAME is the name of the database. The database server is set by the INFORMIXSERVER environmental variable. Using this sample ini file entry would connect to an Informix database called *helpdesk*: sqldb:helpdesk **Oracle** DBNAME is the connection string. Using this sample ini file entry would connect to a local database called *HDSK* using the SQL*Net V1 Oracle Pipe driver: sqldb:P:HDSK Using this sample ini file entry would connect to a local or remote database using SQL*Net V2: sqldb:HDSK **SQL Server** DBNAME is the name of the database server. Using this sample ini file entry would connect to a SQL Server database server called *helpdesk*: sqldb:helpdesk **Sybase** DBNAME is the name of the database server. Using this sample ini file entry would connect to a Sybase database server called *helpdesk*: sqldb:helpdesk

Parameter	Definition
-sqldb2mvstarget:1	Indicates that DB2 client connectivity is used to connect to a DB2 running on an OS/30 system. This optional parameter is for DB2 on OS/390 (MVS) only. It is entered in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 0. 0 - The sqldb2mvstarget parameter is turned off. (default) A value of 1 or more means connect to DB2 on OS/390. A sample initialization file might include: sqldb2mvstarget:1
sqldebug	Causes the SQL statements being executed by the SQL Interface to be printed to the ServiceCenter log file. If this parameter is active, the time it takes to login to SQL (sqllogintime) and the time it takes to do a SQL query request (sqlquerytime) is logged in the log file. This optional parameter is set in the server initialization file (sc.ini or PARMS). This parameter does not take values. If it is found in the initialization file, the statements will be printed to the log file.

Parameter	Definition
sqldetect:*0*	Detects fields in SQL tables that are not defined in the corresponding ServiceCenter dbDict entry. If such a field is detected then the dbDict is updated with a new definition. The new field includes the SQL mapping for the field. With this feature a Database Administrator can make a change to a table and ServiceCenter automatically makes the corresponding update to the dbDict the next time the table is referenced. The recommended method of adding fields (columns) to a ServiceCenter Database Dictionary record for a converted file is to add the columns to the RDBMS table, then refresh the ServiceCenter client. New fields can be added either on a main or an alias table. To change field definitions, for example. from varchar(10) to varchar(12), first change the RDBMS table definition using the alter table statement, then update the mapping and refresh the SQL cache. If sqldetect is active, when a column is added to the RDBMS main table for probsummarym1, the dbDict is refreshed each time a user queries the probsummary file. If a column has been added to an RDBMS (e.g., Oracle) table without using Database Dictionary, the cache must be refreshed for the client to recognize the added column. You can select **Refresh SQL Cache** from the SQL Utilities menu if the file with the new column has not been accessed, otherwise you must log your client out, then log back in, in order to see the added column. **Note:** The **Refresh SQL Cache** option on the SQL Utilities menu must be executed from an *Express* client, as the option has no effect when executed in client/server mode. If a file has been accessed by any process within ServiceCenter, then select the **Refresh SQL Cache** option on the SQL Utilities menu to flush the SQL information from ServiceCenter shared memory. Once the **Refresh SQL Cache** option has been used, ServiceCenter detects the changes made to the SQL table on the next access to the file from any ServiceCenter process that has not already accessed the file. Processes that have already accessed the file must log off and log on in order to pick up the changes. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - Autodetect is turned on. (default) 1 - Autodetect is turned off.

Parameter	Definition
sqldictionary	Specifies that the dbDict entries for ServiceCenter are in an RDBMS database rather than the P4 files. This parameter is required if the dbdict file was converted to RDBMS using ServiceCenter 2.x binaries. Otherwise, it is not necessary. This parameter is set in the server initialization file (sc.ini or PARMS). The *sqldictkey*, *sqldictrecord* and *sqldicttable* parameters can be used with it. This parameter does not take values. If it is found in the server initialization file, ServiceCenter will assume that the database dictionary entries are in an RDBMS.
sqldictkey	Specifies the key column name if it is not *name*. It is only used with the sqldictionary parameter. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include any valid column name used as the primary key and containing the name field of the dbdict record. The default value is name. With this sample initialization file entry, the key column name would be *dbdictkeyfield*. sqldictkey:dbdictkeyfield
sqldictrecord	Specifies the full record column name if it is not *descriptor*. It is only used with the sqldictionary parameter. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include any valid column name containing the full image of one dbdict record. The default value is descriptor. With this sample initialization file entry, the full record column would be *dbdictrecord*. sqldictrecord:dbdictrecord
sqldicttable	Specifies the table name for dbDict if it is not *dbdictm1*. It is only used with the sqldictionary parameter. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include any valid table name containing the dbdict file. The default value is dbdictm1. With this sample initialization file entry, the table name would be *ourdbdicttable*. sqldicttable:ourdbdicttable

Parameter	Definition
sqldirect	Specifies specifies the parameters to be used when directly executing SQL statements using the SQLexecute RAD function. Those parameters determine which RDBMS will be connected to. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are characters. They can be any section name found in the initialization file. See *[db2mvs]*, *[oracle]*, *[sqlserver]*, and *[sybase]* on page 298. A sample initialization file might include: sqldirect:oracle8-remote [oracle8] ... [oracle8-remote]
sqldisconnect:*0*	Specifies when to disconnect from external database. Normally, connections to the external database last from when the user first accesses the RDBMS until the user logs off from ServiceCenter. When this parameter is active, users will be disconnected from the RDBMS when an application reaches a new screen. This will create extra SQL statements, thus creating more overhead, but will limit the number of processes connected to the RDBMS. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - sqldisconnect is turned off. (default) 1 - sqdisconnect is turned on.
sqldrop:*0*	Indicates that tables should be dropped before creation. Issues a DROP TABLE statement before an SQL table is created during the P4 to SQL conversion process. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - sqldrop is turned off. (default) 1 - sqdrop is turned on.

Parameter	Definition
sqlfetchrows:*0*	Indicates the number of rows to fetch into keyset. It allows the user to fetch multiple rows at one time (one hit on server). This speeds up the record retrieving in cases where there is heavy network traffic between a client and a server. sqlfetchrows:n is only recognized by Oracle OCI. It tells the oci module that *n* rows are to be fetched at one time. To secure memory availability, *n* is forced inside the OCI module to be 500 if it is set greater than 500. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only valid for Oracle OCI. Valid values are numeric. The default value is 0. With this sample initialization file entry, the number of rows retrieved would be set to *10*: sqlfetchrows:10
sqlfetchs:*500*	Specifies the number of primary keys fetched into internal record list prior to fetching actual records from RDBMS This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 500. With this sample initialization file entry, the number of keys retrieved would be set to *100*: sqlfetchs:100
sqlidentify	Tells ServiceCenter to use the userid and password of the user logging into ServiceCenter when connecting to the external database. The userid and password are set in the ServiceCenter operator record. If the ServiceCenter user id has not been established but we still need to access data (for example, before login, during system initialization) ServiceCenter will use the sqllogin parameter rather than sqlidentify. This optional parameter is set in the server initialization file (sc.ini or PARMS). This parameter does not take values. If sqlidentify is found in the server initialization file, ServiceCenter will assume that it should identify users by ServiceCenter login name.

Parameter	Definition
sqljoinsok	A comma-delimited list of table names for which outer joins are allowed. This optional parameter is set in the server initialization file (sc.ini or PARMS). This parameter is used only by systems mapped to DB2 on MVS platforms. This parameter will be ignored in other environments. DB2 on MVS allows only outer joins on tables if the primary key is a single field, not a composite key. This list is referenced to determine validity whenever a join file is used. Valid values are strings referencing table names.
sqllimit:*30*	Is used to monitor the performance of SQL calls made through the SQL Interface. When an SQL call exceeds the time limit specified on the sqllimit parameter, a message will be written to the ServiceCenter log file documenting the name of the user making the call, how long the SQL call took, and the SQL statement. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include numeric floating points. The default time limit is 30. With this sample initialization file entry, all SQL statements that take longer than *15* seconds to complete would be written to the log file: sqllimit:15.0
sqllockretry:*5*	Controls processing of locking errors which occur when executing an SQL statement. It specifies how many times the SQL Interface should retry an SQL statement which failed due to a locking error (deadlock, lock time-out, etc.). The default value is 5 times. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 5. With this sample initialization file entry, the statement would be retried *10* times: sqllockretry:10
sqllocktimeout:*0*	Maximum time in seconds to wait for locks.

Parameter	Definition
sqllockwait:*0*	Controls processing of locking errors which occur when executing an SQL statement. It specifies how long in seconds the SQL Interface should wait before retrying an SQL statement which failed due to a locking error (deadlock, lock time-out, etc.). This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 0. 0 - do not wait before retrying. (default) Any other value is the number of seconds to wait before retrying. With this sample initialization file entry, ServiceCenter will wait of *2* seconds before retrying if an SQL statement failed because of a lock: sqllockwait:2
sqllogin:*IDSTR*	Specifies the userid and password to use when connecting to the database. If you omit sqllogin, the SQL Interface connects to the database using the userid of the user who started the ServiceCenter scenter or scserver. **DB2 MVS:** The password is NOT required and should NOT be used. **Informix, DB2 Universal:** The user ID and password must be those of a valid Unix or Windows login. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid entries include valid ServiceCenter userids and passwords. **Note:** Use of this parameter requires a restart of the system before it takes effect. With this sample initialization file entry, the user *SCUSER*, with a password *SCPASSWD* will be connected to the database, rather than the user who started scenter or scserver: sqllogin:SCUSER/SCPASSWD

Parameter	Definition
sqlloginretry:*15*	Specifies the number of login retries the system will make when attempting to connect to the RDBMS. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric: The default value is 15. With this sample initialization file entry, ServiceCenter will try up to *20* times to connect to the RDBMS before terminating: sqlloginretry:20
sqllogintime:*60*	Specifies the amount of time ServiceCenter should wait while attempting to connect to SQL Server or Sybase. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only valid for SQL Server or Sybase. Valid values are numeric. The default value is 60. With this sample initialization file entry, SQL Server/Sybase has *120* seconds to answer to connect request before the connection request times out: sqllogintime:120
sqlloginwait:*15*	Specifies the time, in seconds, to wait between attempts to connect to the RDBMS. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric. The default value is 15. With this sample initialization file entry, ServiceCenter will wait *10* seconds after a connection attempt before retrying: sqlloginwait:10

Parameter	Definition
sqlmodcount:*0*	Specifies the use of the sqlmodcount field to determine if record has changed. It is recognized by Oracle only. When a ServiceCenter system is mapped to Oracle and a record is updated, ServiceCenter must reselect the record to make sure it has not changed since being read and displayed. This re-selection can weigh down system resources, when there is high update activity. A field has been created which eliminates the need for the automatic reselect. If a sysmodcount field is present in the file, rather than re-selecting the record, a qualified update (where sysmodcount = *nnnn*) is performed. The value of the sysmodcount field is added to incrementally with each update. In so doing, if the record has been modified since being read, the value of sysmodcount will have changed and the update fails. The user receives a message indicating the record has changed since being selected. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only valid for Oracle. Valid values are numeric. The default value is 1. 1 - sqlmod count is turned off. Any other value turns sqlmod count on. A sample initialization file might include: sqlmodcount:0
sqloptimizerrows:*0*	Specifies whether or not (and how) Oracle should try to optimize performance during SELECT statements. It is recognized by Oracle only. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only valid for by Oracle. Valid values are numeric (0, 1, >1). The default value is 0. This parameter is valid for Oracle only. 0 - no optimizing goal is specified. 1 - the goal is FIRST_ROWS, meaning Oracle attempts to retrieve the first records as fast as possible. >1 - If this parameter is set to a value larger than 1, the goal is ALL_ROWS, meaning Oracle attempts to retrieve the whole set of records matching a SELECT statement as fast as possible. A sample initialization file might include: sqloptimizerrows:1

Parameter	Definition
sqloracle8x:*0*	Indicates that an Oracle version of 8.1.5 or higher is installed and therefore causes ServiceCenter to load the SQOR8CLI.DLL in order to connect to the Oracle database. However, this parameter is only used when ServiceCenter cannot determine the Oracle version by itself. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only valid for by Oracle. Valid values are numeric. The default value is 1. 0 - sqloracle8x count is turned off. Any other value turns sqloracle8x count on. A sample initialization file might include: sqloracle8x:0
sqlouterjoins:*1*	Disables the outer join process. Outer joins are used when processing *or* statements against multiple array tables, e.g., the standard inbox query which searches primary assignment and secondary assignment fields looking for a match for the current operator's assignment group. If you are not using *or* statements, this parameter can significantly decreases the amount of time a query requires to run. When used, however, query results only include items which fulfill the whole query as entered. No partial query results are returned. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are numeric (0 or 1). The default value is 0. 0 - sqlouterjoins is turned off. (default) 1 - sqlouterjoins is turned on. A sample initialization file might include: sqlouterjoins:1

Parameter	Definition
sqlpacketsize:*0*	Specifies the size (in bytes) of a TDS (Tabular Data Stream). This is the amount of data that is transferred between the RDBMS server and the client. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is recognized only by SQL Server and Sybase. Valid values are numeric, between 0 and 65535. The default value depends on the SQL Server default value. 0 - sqlpacketsize is turned off. 1 - sqlpacketsize is turned on. With this sample initialization file entry, the packet size would be set to *4096*: sqlpacketsize:4096
sqlplan:*scplan1*	Used to set the plan for connecting to DB2/OS/390. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is recognized only by SQL Server and Sybase. Valid values include any valid DB2 plan name. The default value is SCPLAN1. With this sample initialization file entry, the plan name would be set to *DB2PLAN*: sqlplan:DB2PLAN
sqlquerytime:*0*	The time, in seconds, to wait for the RDBMS server to respond to a SQL command. Specifies the amount of time ServiceCenter will wait for RDBMS server to respond to a SQL statement. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is recognized only by SQL Server and Sybase. Valid values are numeric, between 0 and 65535. 0 - Wait until a response is received. The default value depends on the SQL Server default value. Any other value is the number of seconds to wait. With this sample initialization file entry, the SQL statements would time out after 30 seconds: sqlquerytime:20

Parameter	Definition
sqlreferential:*1*	Indicates you want referential integrity constraints if supported by SQL. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is recognized only by Oracle, SQL Server and Sybase. Valid values are numeric (0 or 1). The default value is 1. 0 - turns sqlreferential off. 1 - turns sqlreferential on. Any other value is the number of seconds to wait. A sample initialization file might include: sqlreferential:0
sqlreuseablesql:*1*	Toggles the Reusable SQL feature. Reusable SQL allows the system to cache query results for use later, and decreases the rate at which buffers are filled with recorded queries. For more information, see the "Reusable SQL" section in the *Database Management and Administration Guide*.This optional parameter is set in the server initialization file (sc.ini or PARMS). It is only recognized by Oracle and DB2/Universal. Valid values are numeric. The default value is 1. 0 - turns sqlreuseablesql off. 1 - turns sqlreuseablesql on. Any other value is the number of seconds to wait. A sample initialization file might include: sqlreuseablesql:0
sqltextdateformat:*MM/DD/YY*	Specifies the format for date/time values when stored as text in SQL database. This parameter also defines the format that should be used when a date array is mapped to a single field. Each date in the array is turned into a STRING and separated by new line characters. The sqltextdateformat parameter indicates the format that should be used for all dates when written out as STRINGS in the database. It causes all dates to have the same format regardless of the date format specified for the user. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values are: YYYY/MM/DD, YY/MM/DD, MM/DD/YYYY, MM/DD/YY, DD/MM/YYYY, and DD/MM/YY. The default date format is MM/DD/YY.

Parameter	Definition
sqltz:*Greenwich/Universal*	Specifies the timezone to use as a base for all date/time values. The timezone is specified as the name of the timezone record in the ServiceCenter tzfile. If you are using Distributed Ticketing, ensure that each ServiceCenter instance is set to the same time zone. This optional parameter is set in the server initialization file (sc.ini or PARMS). Valid values include any time zone record included in the time zone table (tzfile). The default time zone is Greenwich/Universal (GMT). **Note:** Use of this parameter requires a restart of the system before it takes effect. With this sample initialization file entry, the timezone is set to *Pacific* time: sqltz:US/Pacific
sqlwildcard:*1*	SQL wildcard searches are allowed. This optional parameter is set in the server initialization file (sc.ini or PARMS). It is recognized only by Oracle, SQL Server and Sybase. Valid values are numeric (0 or 1). The default value is 0. 0 - turns sqlwildcard off. Any other value turns sqlwildcard on. A sample initialization file might include: sqlwildcard:1
validateodbcfieldnames	Causes the ODBC driver to edit field names to conform to SQL naming standards. This parameter replaces the period in ServiceCenter field names with an underscore for external applications (such as Microsoft Access and Excel) that use periods as a reserved symbol. When the field names are brought back into ServiceCenter, the underscore is replaced with a period. This optional parameter may be set in the server initialization file (sc.ini or PARMS), or they may appear in the sc.ini file that appears in the Crystal Reports or MyDocuments directory. This parameter does not take values. If it is found in the initialization file, then the ODBC driver will conform to SQL naming standards.

APPENDIX C

Data Definitions

This appendix was designed to give database and system administrators a better understanding of what the ServiceCenter fields are for. It contains data definitions for ServiceCenter files.

Topics in this appendix include:

Change Management Files

cm3r

Fields in the cm3r table:

Field Names	Types	Definition
header	**Structure**	
number	Character	Record number assigned when submitted to the database.
number.attach	Character	Alias field used for virtually joining attachments.
number.apprlog	Character	Alias field used for virtually joining approval log.
number.vj	Character	Alias field used for virtually joining approvals.
page	Number	When using paging this determines the page number.
total.pages	Number	Total number of pages for this record.
category	Character	Category for the record.
status	Character	Keeps the status of the record.
approval.status	Character	Keeps the approval status of the record.
requested.by	Character	Keeps the requestor's name/ID.
request.dept	Character	Keeps the department to which the requestor belongs.
request.phone	Character	Keeps the requestor's phone number.
request.date	Date	Keeps the date when the request is needed by.
assigned.to	Character	Keeps the assignee's name.
assign.dept	Character	Keeps the assignee's department.
assign.phone	Character	Keeps the assignee's phone number.
assign.date	Date	Keeps the date when the request was assigned.
coordinator	Character	Keeps the coordinator's name.
coord.dept	Character	Keeps the coordinator's department.
coord.phone	Character	Keeps the coordinator's phone number.
coord.date	Date	Keeps the date when the coordinator was assigned.
planned.start	Date	Keeps the date when the request is scheduled to being.

Field Names	Types	Definition
system	Character	General information field.
key.item.affected	Character	General information field.
planned.end	Date	Keeps the date when the request is scheduled to end.
reason	Character	Reason for on outage.
duration	Date	Duration of the outage.
current.phase	Character	The current phase of the record.
risk.assessment	Character	The risk assessment number for risk calculation.
ipl.required	Character	Mainframe device which needs to be restarted.
location.code	Character	Location code value from the location support table.
priority	Character	Priority for the ticket.
ipl.type	Character	Type of mainframe job.
date.entered	Date	Keeps the date when the request was opened.
operator	Character	Name of the operator that opened the request.
last	Logical	Determines if this is the last page (most recent version of the record).
open	Logical	Determines if the request is active or not.
resolved.problems	Array of Characters	General information field.
prereq.changes	Array of Numbers	General information field.
coreq.changes	Array of Numbers	General information field.
alert.stage	Character	Stage which alerts have reached for this request.
orig.date.entered	Date	Keeps the date when the request was opened.
orig.operator	Character	General information field.
business.area	Character	Business area value from the location support table.
backout.duration	Date	Duration of the backout process in case the change is unsuccessful.
close.time	Date	Keeps the date and time when the request is closed.
foreign.id	Character	General information field.
work.manager	Character	Work Management manager name for ticket assignment.

Field Names	Types	Definition
sla.alert1	Date	For SLA Management use. This field keeps track of the next SLA alert time.
sla.target	Date	For SLA Management use. This field keeps track of the calculated targeted completion date.
sla.deadline	Date	For SLA Management use. This field keeps track of the absolute completion date to meet SLA.
skip.phases	Logical	
extension	Character	Extension number from the contacts support table.
still.to.go.saved	Date	For SLA Management use. Stores the calculated interval before the sla.target date.
estimate.approved	Logical	
type.level2	Character	For SLA Management use. This information can come from the cm3rsubcat support table.
company	Character	Company name brought over from either the contacts or company support table.
brief.description	Character	Short description of the nature of the request.
subcategory	Character	For SLA Management use. This information can come from the cm3rsubcat support table.
number.string	Character	
vj.number.string.4	Character	Virtual join alias field.
vj.number.string.3	Character	Virtual join alias field.
vj.number.string.2	Character	Virtual join alias field.
vj.number.string.1	Character	Virtual join alias field.
billto	Character	Department to bill the request to.
billtype	Character	Type of bill to issue.
gl.number	Character	Ledger number for the bill.
description.structure	**Structure**	
description	Array of Characters	Detailed description of the request.
justification	Array of Characters	Justification for opening the request.
backout.method	Array of Characters	Detailed method for backing out the changes in cased of failure.
approval.structure	**Structure**	

Field Names	Types	Definition
reviewer.class	Array of Characters	List of reviewers for this phase.
approved.groups	Array of Characters	Groups that have already approved on this phase.
approved.dates	Array of Dates	Dates when the approval groups have approved on this phase.
approved.oper	Array of Characters	Operator name/ID belonging to the approval group that approved on this phase.
approved.action	Array of Characters	Action taken by the operator when approving on this phase.
approvals.required	Array of Characters	List of future approval groups.
current.pending.groups	Array of Characters	List of approval groups remaining.
approvals.req.seq	Array of Numbers	Sequence of future approvals.
approved.req.seq	Array of Numbers	Sequence of groups that have approved.
current.req.seq	Array of Numbers	Sequence of current approval groups.
approve.desc	Array of Characters	Comments section of the Approvals tab.
middle	**Structure**	
type	Character	General information field.
backup.device	Character	General information field.
model	Character	General information field.
vendor	Character	General information field.
fixed.asset.no.	Character	General information field.
manufacturer	Character	General information field.
cpu.interruption	Character	General information field.
program.name	Character	General information field.
operating.system	Character	General information field.
maint.level	Character	General information field.
library.affected	Character	General information field.
release.level	Character	General information field.
data.set.affected	Character	General information field.
network.affected	Character	General information field.
version	Character	General information field.

Field Names	Types	Definition
install.or.remove	Character	General information field.
upgrade	Character	General information field.
vtam.name	Character	General information field.
vtam.parent	Character	General information field.
parent	Character	General information field.
product.no.	Character	General information field.
logical.name	Character	Name of the device from the Inventory Management module.
location	Character	Name of the location from the location support table.
serial.no.	Character	General information field.
jobname	Character	General information field.
misc1	Character	General information field.
misc2	Character	General information field.
misc3	Character	General information field.
misc4	Character	General information field.
misc5	Character	General information field.
misc6	Character	General information field.
misc7	Character	General information field.
misc8	Character	General information field.
misc9	Character	General information field.
misc10	Character	General information field.
group	Character	General information field.
down.start	Date	General information field.
down.end	Date	General information field.
ram.current	Character	General information field.
hard.disc.current	Character	General information field.
location.code.current	Character	General information field.
ram.new	Character	General information field.
hard.disc.new	Character	General information field.

Field Names	Types	Definition
location.code.new	Character	General information field.
size	Character	General information field.
install.date	Date	General information field.
sched.outage.start	Date	Time when an outage should be expected to start.
sched.outage.end	Date	Time when an outage should be expected to end.
actual.outage.start	Date	Time when an outage actually began.
actual.outage.end	Date	Time when an outage actually ended.
outage.comments	Array of Characters	Comments on the outage.
resched.outages	Array of Dates	General information field.
cancelled.outages	Array of Dates	General information field.
move.flag	Logical	General information field.
add.flag	Logical	General information field.
change.flag	Logical	General information field.
account.type	Character	General information field.
account.id	Character	General information field.
user.name	Character	General information field.
misc.array1	Array of Characters	General information field.
misc.array2	Array of Characters	General information field.
contract.id	Number	General information field.
misc.array3	Array of Characters	General information field.
erp.unique.id	Character	General information field.
erp.description	Character	General information field.
erp.development.sid	Character	General information field.
erp.development.client	Character	General information field.
erp.development.approver	Character	General information field.
erp.development.gateway.id	Character	General information field.
erp.released	Character	General information field.
erp.target.sid	Character	General information field.
erp.target.client	Character	General information field.

Field Names	Types	Definition
erp.id.requested	Character	General information field.
erp.instances	Arrayed Structure	General information field.
erp.type	Character	General information field.
erp.sid	Character	General information field.
erp.client	Character	General information field.
erp.approver	Character	General information field.
erp.gateway.id	Character	General information field.
erp.sequence.no	Number	General information field.
erp.override.reschedule	Logical	General information field.
assets	Array of Characters	General information field.
contact.cost.centre	Character	General information field.
estimate.units	Character	General information field.
estimate.description	Character	General information field.
estimate.price	Character	General information field.
estimate.delivery	Character	General information field.
estimate.budget	Character	General information field.
estimate.effort	Character	General information field.
estimate.grade	Character	General information field.
actual.cost	Character	General information field.
actual.units	Character	General information field.
actual.price	Character	General information field.
actual.grade	Character	General information field.
corp.structure	Character	General information field.
asset.comments	Array of Characters	General information field.
close	**Structure**	
completion.code	Number	Closure code for the request.
hours.worked	Date	Hours worked on the request (manually entered).
closing.comments	Array of Characters	Closing comments for the request.
parts	**Arrayed Structure**	

Field Names	Types	Definition
date	Date	Date a part was used.
part.no	Character	Part number for the part.
quantity	Number	Quantity of the part used.
labor	**Arrayed Structure**	
labor.date	Date	Date worked.
sc.operator	Character	Operator that worked.
sc.hours.worked	Number	Hours operator worked.
li.contract.id	Number	Listed contract through Contract Management.
sysmodcount	Number	Revision tracking field for update counts.
sysmoduser	Character	Revision tracking field for operator who updated the record.
sysmodtime	Date	Revision tracking field for last update date and time.
contact.last.name	Character	Last name for the requestor.
contact.first.name	Character	First name for the requestor.

cm3t

Fields in the cm3t table:

Field Names	Types	Description
header	**Structure**	
number	Character	Record number assigned when submitted to the database.
number.attach	Character	Alias field used for virtually joining attachments.
number.apprlog	Character	Alias field used for virtually joining approval log.
number.vj	Character	Alias field used for virtually joining approvals.
page	Number	When using paging this determines the page number.
total.pages	Number	Total number of pages for this record.
category	Character	Category for the record.
status	Character	Keeps the status of the record.
approval.status	Character	Keeps the approval status of the record.

Field Names	Types	Description
requested.by	Character	Keeps the requestor's name/ID.
request.dept	Character	Keeps the department to which the requestor belongs.
request.phone	Character	Keeps the requestor's phone number.
request.date	Date	Keeps the date when the task is needed by.
assigned.to	Character	Keeps the assignee's name.
assign.dept	Character	Keeps the assignee's department.
assign.phone	Character	Keeps the assignee's phone number.
assign.date	Date	Keeps the date when the task was assigned.
coordinator	Character	Keeps the coordinator's name.
coord.dept	Character	Keeps the coordinator's department.
coord.phone	Character	Keeps the coordinator's phone number.
coord.date	Date	Keeps the date when the coordinator was assigned.
planned.start	Date	Keeps the date when the task is scheduled to being.
system	Character	General information field.
key.item.affected	Character	General information field.
planned.end	Date	Keeps the date when the task is scheduled to end.
reason	Character	Reason for on outage.
duration	Date	Duration of the outage.
current.phase	Character	The current phase of the record.
risk.assessment	Character	The risk assessment number for risk calculation.
ipl.required	Character	Mainframe device which needs to be restarted.
location.code	Character	Location code value from the location support table.
priority	Character	Priority for the ticket.
ipl.type	Character	Type of mainframe job.
date.entered	Date	Keeps the date when the task was opened.
operator	Character	Name of the operator that opened the request.
last	Logical	Determines if this is the last page (most recent version of the record).
open	Logical	Determines if the task is active or not.

Field Names	Types	Description
resolved.problems	Array of Characters	General information field.
prereq.tasks	Array of Numbers	General information field.
coreq.tasks	Array of Numbers	General information field.
alert.stage	Character	Stage which alerts have reached for this task.
orig.date.entered	Date	Keeps the date when the request was opened.
orig.operator	Character	General information field.
parent.change	Character	Number for the parent request.
parent.change.vj	Character	Virtual join to the change request.
business.area	Character	Business area value from the location support table.
backout.duration	Date	Duration of the backout process in case the change is unsuccessful.
parent.phase	Character	Current phase of the parent change request.
qbe.flag	Character	General information field.
close.time	Date	Keeps the date and time when the task is closed.
foreign.id	Character	General information field.
fparent.change	Character	No longer used.
is.parent	Logical	Determines if this task is a parent task for another task.
parent.task	Character	Parent task for this task.
fparent.task	Character	No longer used.
work.manager	Character	Work Management manager name for ticket assignment.
submit	Logical	General information field.
company	Character	Company name brought over from either the contacts or company support table.
billto	Character	Department to bill the task to.
billtype	Character	Type of bill to issue.
gl.number	Character	Ledger number for the bill.
description.structure	**Structure**	
description	Array of Characters	Detailed description of the task.

Field Names	Types	Description
justification	Array of Characters	Justification for opening the task.
backout.method	Array of Characters	Detailed method for backing out the changes in cased of failure.
approval.structure	**Structure**	
reviewer.class	Array of Characters	List of reviewers for this phase.
approved.groups	Array of Characters	Groups that have already approved on this phase.
approved.dates	Array of Dates	Dates when the approval groups have approved on this phase.
approved.oper	Array of Characters	Operator name/ID belonging to the approval group that approved on this phase.
approved.action	Array of Characters	Action taken by the operator when approving on this phase.
approvals.required	Array of Characters	List of future approval groups.
current.pending.groups	Array of Characters	List of approval groups remaining.
approvals.req.seq	Array of Numbers	Sequence of future approvals.
approved.req.seq	Array of Numbers	Sequence of groups that have approved.
current.req.seq	Array of Numbers	Sequence of current approval groups.
approve.desc	Array of Characters	Comments section of the Approvals tab.
middle	**Structure**	
type	Character	General information field.
backup.device	Character	General information field.
model	Character	General information field.
vendor	Character	General information field.
fixed.asset.no.	Character	General information field.
manufacturer	Character	General information field.
cpu.interruption	Character	General information field.
program.name	Character	General information field.
operating.system	Character	General information field.
maint.level	Character	General information field.
library.affected	Character	General information field.

Field Names	Types	Description
release.level	Character	General information field.
data.set.affected	Character	General information field.
network.affected	Character	General information field.
version	Character	General information field.
install.or.remove	Character	General information field.
upgrade	Character	General information field.
vtam.name	Character	General information field.
vtam.parent	Character	General information field.
parent	Character	General information field.
product.no.	Character	General information field.
logical.name	Character	Name of the device from the Inventory Management module.
location	Character	Name of the location from the location support table.
serial.no.	Character	General information field.
jobname	Character	General information field.
misc1	Character	General information field.
misc2	Character	General information field.
misc3	Character	General information field.
misc4	Character	General information field.
misc5	Character	General information field.
misc6	Character	General information field.
misc7	Character	General information field.
misc8	Character	General information field.
misc9	Character	General information field.
misc10	Character	General information field.
group	Character	General information field.
brief.desc	Character	Short description for the task
down.start	Date	General information field.
down.end	Date	General information field.

Field Names	Types	Description
work.notes	Array of Characters	General information field.
work.start	Array of Dates	General information field.
work.end	Array of Dates	General information field.
ram.current	Character	General information field.
hard.disc.current	Character	General information field.
location.code.current	Character	General information field.
ram.new	Character	General information field.
hard.disc.new	Character	General information field.
location.code.new	Character	General information field.
install.date	Date	General information field.
size	Character	General information field.
tape.description	Array of Characters	General information field.
tape.name	Character	General information field.
tape.location	Character	General information field.
tape.date	Date	General information field.
delete.step	Logical	General information field.
create.step	Logical	General information field.
volume.step	Logical	General information field.
restore.step	Logical	General information field.
test.step	Logical	General information field.
application.name	Character	General information field.
application.name.old	Character	General information field.
version.old	Character	General information field.
manufacturer.old	Character	General information field.
license.number	Character	General information field.
contact.name	Character	General information field.
user.id	Character	General information field.
dept	Character	General information field.
dept.old	Character	General information field.

Field Names	Types	Description
contact.phone	Character	General information field.
account.name	Character	General information field.
account.type	Character	General information field.
account.group	Character	General information field.
asset	Array of Characters	General information field.
asset.comments	Array of Characters	General information field.
building	Character	General information field.
floor	Character	General information field.
room	Character	General information field.
misc.array1	Array of Characters	General information field.
misc.array2	Array of Characters	General information field.
misc.array3	Array of Characters	General information field.
misc.array4	Array of Characters	General information field.
misc.array5	Array of Characters	General information field.
misc.array6	Array of Logicals	General information field.
contract.id	Number	General information field.
erp.unique.id	Character	General information field.
erp.parent.unique.id	Character	General information field.
erp.released	Character	General information field.
erp.type	Character	General information field.
erp.sid	Character	General information field.
erp.client	Character	General information field.
erp.gateway.id	Character	General information field.
erp.approver	Character	General information field.
erp.imported	Character	General information field.
erp.development.client	Character	General information field.
erp.development.sid	Character	General information field.
erp.development.gateway.id	Character	General information field.
erp.sequence.no	Number	General information field.

Field Names	Types	Description
erp.active.flag	Logical	General information field.
erp.override.reschedule	Logical	General information field.
estimate.units	Character	General information field.
estimate.price	Character	General information field.
estimate.description	Character	General information field.
estimate.delivery	Character	General information field.
estimate.budget	Character	General information field.
estimate.effort	Character	General information field.
estimate.grade	Character	General information field.
actual.units	Character	General information field.
actual.cost	Character	General information field.
actual.price	Character	General information field.
actual.grade	Character	General information field.
corp.structure	Character	General information field.
close	**Structure**	
completion.code	Number	Closure code for the request.
hours.worked	Date	Hours worked on the request (manually entered).
closing.comments	Array of Characters	Closing comments for the request.
parts	**Arrayed Structure**	
date	Date	Date a part was used.
part.no	Character	Part number for the part.
quantity	Number	Quantity of the part used.
labor	**Arrayed Structure**	
labor.date	Date	Date worked.
operator	Character	Operator that worked.
hours.worked	Number	Hours operator worked.
li.contract.id	Number	Listed contract through Contract Management.
sysmodcount	Number	Revision tracking field for update counts.

Field Names	Types	Description
sysmoduser	Character	Revision tracking field for operator who updated the record.
sysmodtime	Date	Revision tracking field for last update date and time.

Incident Management Files

problem

Fields in the problem table:

Field Name	Types	Description
header	**Structure**	
number	Character	The record number for the ticket submitted to the database.
number.attach	Character	Virtual join alias field.
vj.number.5	Character	Virtual join alias field.
vj.number.4	Character	Virtual join alias field.
vj.number.3	Character	Virtual join alias field.
vj.number.2	Character	Virtual join alias field.
vj.number.1	Character	Virtual join alias field.
number.vj	Character	Virtual join alias field.
page	Number	When using paging this number increments from one for each record that gets added as an individual page.
total.pages	Number	The total number of pages (updates) on the ticket.
open.time	Date	The time when the ticket was opened.
category	Character	Category classifying the issue of the ticket.
alert.time	Date	The time when the next alert will fire off.
assignment	Character	The current group responsible for resolving the issue.
update.time	Date	The time when the ticket was last updated.

Field Name	Types	Description
asgnchg	Number	The number of times the ticket has changed assignment groups.
status	Character	Ticket's alert status field.
close.time	Date	Time when the ticket was closed.
reopen.time	Date	Should the ticket be reopened, determines the time it was reopened at.
last	Logical	Determines if the record is the last page for a ticket when using paging.
deadline.alert	Date	Determined the time when Deadline Alert will fire off for this ticket.
deadline.group	Character	Determines the group to assign the ticket when Deadline Alert is reached based on the category definition.
deadline.alert.flag	Logical	Determines if the ticket is in Deadline Alert.
lookup.time	Date	General information field.
priority.code	Character	The order in which to address this issue in comparison to others.
flag	Logical	Determines if the ticket is active or not.
change.no	Number	General information field.
document.id	Character	General information field.
foreign	Number	General information field.
foreign.id	Character	General information field.
brief.description	Character	Short description for the issue reported.
ticket.owner	Character	Determines who the owner of the ticket is.
updated.by	Character	Determines the last person to update the ticket.
problem.status	Character	Determines the ticket status.
secondary.assignment	Array of Characters	Determines the list of additional assignment groups (other than the primary) involved in the resolution of the ticket.
reopened.by	Character	Should the ticket be reopened, determines the person who reopened the ticket.
sla.contact	Character	General information field.

Field Name	Types	Description
sla.vendor	Character	General information field.
hot.tic	Logical	Determines if the ticket should be identified as a "hot" issue.
agreement.id	Number	SLA agreement code.
sla.started	Logical	General information field.
sla.ended	Logical	General information field.
y2k.related	Logical	General information field.
operational.device	Logical	General information field.
prev.update.time	Date	General information field.
knownerror	Logical	General information field.
unsuspend.time	Date	Determines when the ticket should be unsuspended when using the Suspend state.
oti.originator	Character	General information field.
oti.originator.reference	Character	General information field.
oti.originator.version	Character	General information field.
oti.tosc.consumer	Character	General information field.
oti.tosc.consumer.reference	Character	General information field.
oti.tosc.provider	Character	General information field.
oti.tosc.provider.reference	Character	General information field.
oti.message.type	Character	General information field.
oti.action	Character	General information field.
oti.last.message	Character	General information field.
oti.fromsc.consumer	Character	General information field.
oti.fromsc.consumer.referenc	Character	General information field.
oti.fromsc.provider	Character	General information field.
oti.fromsc.provider.referenc	Character	General information field.
call.origin	Character	General information field.
adj.resolution.time	Date	General information field.
res.anal.code	Character	General information field.

Field Name	Types	Description
last.activity	Character	General information field.
billto	Character	General information field.
billtype	Character	General information field.
gl.number	Character	General information field.
format	Character	The format to use when the ticket comes up within the module based on the category formats identified.
pagelist.format	Character	General information field.
Action	**Structure**	
action	Array of Characters	Field containing a description of the reported issue.
resolution	Array of Characters	Field containing the resolution to the issue reported.
update.action	Array of Characters	Field containing an account of the updates
resolution.code	Character	General information field.
opened.by	Character	The person that opened the ticket.
actor	Character	General information field.
comments	Array of Characters	General information field.
justification	Array of Characters	General information field.
severity.code	Character	The impact the issue has.
cause.code	Character	General information field.
affected	Array of Characters	General information field.
closed.by	Character	The person who closed the ticket.
kpf.id	Character	Knowledge article number (for Knowlix for ServiceCenter integration).
site.visit.date	Date	General information field.
site.visit.technician	Character	General information field.
site.visit.count	Character	General information field.
resolved.group	Character	General information field.
closed.group	Character	General information field.
resolved.time	Date	Time when the ticket was resolved.
resolved.by	Character	Determines the person that resolved the ticket when using the "resolved" alert status.

Field Name	Types	Description
middle	**Structure**	
contact.name	Character	Contact person for the reported issue.
phone	Character	General information field.
customer.no.	Character	General information field.
version	Character	General information field.
release.no.	Character	General information field.
model	Character	General information field.
type	Character	General information field.
vendor	Character	General information field.
serial.no.	Character	General information field.
maint.level	Character	General information field.
executing.device	Character	General information field.
location	Character	General information field.
os.release.level	Character	General information field.
os.maint.level	Character	General information field.
abend.code	Character	General information field.
operating.system	Character	General information field.
os.hang	Character	General information field.
install.fail	Character	General information field.
other.docum	Character	General information field.
severity	Character	No longer used.
related.components	Character	General information field.
product.no.	Character	General information field.
syslog	Character	General information field.
dump	Character	General information field.
joblog	Character	General information field.
printout	Character	General information field.
user.group	Character	General information field.
requested.date	Character	General information field.

Field Name	Types	Description
impact	Date	General information field.
time.estimate	Character	General information field.
cost.estimate	Number	General information field.
scheduled.start	Date	General information field.
scheduled.completion	Date	General information field.
department.affected	Array of Characters	General information field.
logical.name	Character	Device reported with the ticket.
failing.component	Character	General information field.
other.symptom	Character	General information field.
vtam.name	Character	General information field.
explanation	Array of Characters	General information field.
application	Character	General information field.
contact.phone	Character	General information field.
assignee.name	Character	Name of the person the ticket is assigned to.
assignee.phone	Character	General information field.
domain	Character	General information field.
system	Character	General information field.
symptoms	Array of Characters	General information field.
documents	Array of Characters	General information field.
job.number	Character	General information field.
job.name	Character	General information field.
manufacturer	Character	General information field.
dept	Character	General information field.
network.name	Character	General information field.
id	Character	General information field.
key.words	Array of Characters	General information field.
contact.time	Date	General information field.
referral.time	Date	General information field.
backup.start	Date	General information field.

Field Name	Types	Description
backup.end	Date	General information field.
reference.no	Character	General information field.
circuit.no.	Character	General information field.
respond.time	Date	General information field.
onsite.time	Date	General information field.
repair.time	Date	General information field.
group	Character	General information field.
downtime.start	Date	General information field.
downtime.end	Date	General information field.
referred.to	Character	General information field.
caller.id	Character	General information field.
assignee.email	Character	General information field.
network.address	Character	General information field.
open.group	Character	General information field.
objid	Character	General information field.
callback.list	Array of Characters	Determines the list of people to notify when the ticket is closed.
parent	Character	General information field.
building	Character	General information field.
floor	Character	General information field.
quote.no	Character	General information field.
incident.id	Character	General information field.
company	Character	General information field.
subcategory	Character	General information field.
application.name	Character	General information field.
planned.start	Date	General information field.
planned.end	Date	General information field.
junk	Logical	General information field.
contract.id	Number	Contract Management code number.

Field Name	Types	Description
fix.type	Character	General information field.
payroll.no	Character	General information field.
critical.user	Character	General information field.
room	Character	General information field.
user.type	Character	General information field.
site.category	Character	Determines the critical state of the site involved.
total.loss	Logical	Determines if the issue involves a Total Loss of Service.
product.type	Character	Categorization field for issue identification.
problem.type	Character	Categorization field for issue identification.
no.SDU.fix	Logical	Determines if there's no correction to the issue.
first.name	Character	General information field.
last.name	Character	General information field.
extension	Character	General information field.
manager.name	Character	General information field.
manager.phone	Character	General information field.
manager.email	Character	General information field.
cost.centre	Character	General information field.
contact.email	Character	General information field.
critical.device	Logical	General information field.
contact.location	Character	General information field.
serial.no	Character	General information field.
third.party.name	Array of Characters	General information field.
third.party.reference	Array of Characters	General information field.
third.party.referred	Array of Dates	General information field.
third.party.referred.by	Array of Characters	General information field.
time.spent	Date	General information field.
different.from.contact	Logical	General information field.
alternate.contact	Character	General information field.

Field Name	Types	Description
alternate.phone	Character	General information field.
alternate.extension	Character	General information field.
class	Character	General information field.
country	Character	General information field.
customer.reference	Character	General information field.
component.category	Character	General information field.
expd.response.time	Array of Dates	General information field.
site	Character	General information field.
address.1	Character	General information field.
address.2	Character	General information field.
county	Character	General information field.
postcode	Character	General information field.
fax	Character	General information field.
alternate.fax	Character	General information field.
part.number	Array of Characters	General information field.
part.quantity	Array of Characters	General information field.
part.description	Array of Characters	General information field.
manufacture.date	Date	General information field.
pending.change	Logical	General information field.
mandatory.asset	Logical	General information field.
address.3	Character	General information field.
city	Character	General information field.
source	Character	General information field.
first.time.fix	Logical	General information field.
user.id	Character	General information field.
variable1	Character	General information field.
variable2	Character	General information field.
variable3	Character	General information field.
cus.error	Logical	General information field.

Field Name	Types	Description
reg.error	Logical	General information field.
sla.alert.time	Date	General information field.
srvc.manager	Character	General information field.
srvc.del.manager	Character	General information field.
adj.resolution.by	Character	General information field.
user.priority	Character	General information field.
sla.expire	Date	General information field.
corp.structure	Character	General information field.
parent.serial.no	Character	General information field.
number.for.wp	Character	General information field.
warranty	**Structure**	
failing.product.no.	Character	General information field.
failing.serial.no.	Character	General information field.
warranty.checked.flag	Logical	General information field.
warranty.status	Character	General information field.
warranty.notes	Array of Characters	General information field.
location.full.name	Character	General information field.
asset.status	Character	General information field.
supervisor	Character	General information field.
sysorgsite	Number	SCD site information.
syshomesite	Number	SCD site information.
sysmodtime	Date	Determines the last time the record was updated.
solution.candidate	Logical	Determines if the issue is worthy of adding to the internal knowledge base.
parts	**Arrayed Structure**	
date	Date	General information field.
part.no	Character	General information field.
quantity	Number	General information field.
labor	**Arrayed Structure**	

Field Name	Types	Description
date	Date	General information field.
operator	Character	General information field.
hours.worked	Number	General information field.
li.contract.id	Number	General information field.
contract.consumed	Logical	General information field.
sysmodcount	Number	Determines how many times the record has been updated.
sysmoduser	Character	Determines the last person to update the record.
mobile.update.time	Date	General information field.
mobile.checkout	Logical	General information field.

probsummary

Fields in the probsummary table:

Field Name	Types	Description
number	Character	The record number for the ticket submitted to the database.
number.vj	Character	Virtual join alias field.
vj.number.1	Character	Virtual join alias field.
vj.number.2	Character	Virtual join alias field.
vj.number.3	Character	Virtual join alias field.
vj.number.4	Character	Virtual join alias field.
vj.number.5	Character	Virtual join alias field.
number.attach	Character	Virtual join alias field.
category	Character	Category classifying the issue of the ticket.
open.time	Date	The time when the ticket was opened.
opened.by	Character	The person that opened the ticket.
priority.code	Character	The order in which to address this issue in comparison to others.
severity.code	Character	The impact the issue has.

Field Name	Types	Description
update.time	Date	The time when the ticket was last updated.
assignment	Character	The current group responsible for resolving the issue.
referral.time	Date	General information field.
referred.to	Character	General information field.
alert.time	Date	The time when the next alert will fire off.
status	Character	Ticket's alert status field.
close.time	Date	Time when the ticket was closed.
closed.by	Character	The person who closed the ticket.
elapsed.time	Date	The total time spent on resolving this ticket based on the time entered with each update.
vendor	Character	General information field.
reference.no	Character	General information field.
contact.time	Date	General information field.
referral.to.contact	Date	General information field.
onsite.time	Date	General information field.
contact.to.respond	Date	General information field.
repair.time	Date	General information field.
onsite.to.repair	Date	General information field.
backup.start	Date	General information field.
backup.time	Date	General information field.
backup.end	Date	General information field.
downtime	Date	General information field.
cause.code	Character	General information field.
resolution.code	Character	General information field.
logical.name	Character	Device reported with the ticket.
logical.name.vj	Character	Virtual join alias field.
group	Character	General information field.
job.name	Character	General information field.

Field Name	Types	Description
location	Character	General information field.
version	Character	General information field.
type	Character	General information field.
abend.code	Character	General information field.
model	Character	General information field.
action	Array of Characters	Field containing a description of the reported issue.
resolution	Array of Characters	Field containing the resolution to the issue reported.
affected	Array of Characters	General information field.
key.words	Array of Characters	General information field.
xreference	Array of Characters	General information field.
alert1	Logical	Determines if the ticket is in Alert Stage 1.
alert2	Logical	Determines if the ticket is in Alert Stage 2.
alert3	Logical	Determines if the ticket is in Alert Stage 3.
deadline	Logical	Determines if the ticket is in Deadline Alert.
reassigned	Logical	Determines if the ticket is in Reassignment Alert.
id	Character	General information field.
lookup.time	Date	General information field.
total.pages	Number	Total number of pages (updates) if paging is enabled for the category.
flag	Logical	Determines if the ticket is active or not.
downtime.end	Date	General information field.
downtime.start	Date	General information field.
assignee.name	Character	Person assigned to the ticket.
respond.time	Date	General information field.
contact.name	Character	Contact person for the reported issue.
contact.name.vj	Character	Virtual join alias field.
seconds	Number	General information field.
caller.id	Character	General information field.

Field Name	Types	Description
contact.phone	Character	General information field.
update.action	Array of Characters	Field containing an account of the updates
actor	Character	General information field.
format	Character	The format to use when the ticket comes up within the module based on the category formats identified.
count	Number	General information field.
asgnchg	Number	General information field.
respond.to.onsite	Date	General information field.
network.name	Character	General information field.
final.close	Date	General information field.
open.group	Character	General information field.
alert.status	Character	General information field.
deadline.group	Character	Determines the group to assign the ticket when Deadline Alert is reached based on the category definition.
deadline.alert	Date	Determined the time when Deadline Alert will fire off for this ticket.
pending.date	Date	General information field.
referral.count	Number	General information field.
pending.reason	Character	General information field.
network.address	Character	General information field.
outage.type	Character	General information field.
parent	Character	General information field.
domain	Character	General information field.
callback.list	Array of Characters	Determines the list of people to notify when the ticket is closed.
closing.comments	Array of Characters	General information field.
cs.code	Character	General information field.
change.no	Number	General information field.
last.name	Character	General information field.

Field Name	Types	Description
first.name	Character	General information field.
company	Character	General information field.
start.time	Date	General information field.
title	Character	General information field.
brief.description	Character	Short description for the issue reported.
document.id	Character	General information field.
foreign	Number	General information field.
foreign.id	Character	General information field.
dept	Character	General information field.
serial.no.	Character	General information field.
building	Character	General information field.
floor	Character	General information field.
quote.no	Character	General information field.
ticket.owner	Character	Determines who the owner of the ticket is.
incident.id	Character	General information field.
sysorgsite	Number	SCD site information.
syshomesite	Number	SCD site information.
sysmodtime	Date	Determines the last time the record was updated.
updated.by	Character	Determines the last person to update the ticket.
problem.status	Character	Determines the ticket status.
secondary.assignment	Array of Characters	Determines the list of additional assignment groups (other than the primary) involved in the resolution of the ticket.
sla.contact	Character	General information field.
sla.vendor	Character	General information field.
company.sla	Character	General information field.
subcategory	Character	Categorization field for issue identification.
hot.tic	Logical	Determines if the ticket should be identified as a "hot" issue.
application.name	Character	General information field.

Field Name	Types	Description
solution.candidate	Character	Determines if the issue is worthy of adding to the internal knowledge base.
agreement.id	Number	SLA agreement code.
planned.start	Date	General information field.
planned.end	Date	General information field.
y2k.related	Logical	General information field.
operational.device	Logical	General information field.
junk	Logical	General information field.
contract.id	Number	Contract Management code number.
sysmodcount	Number	Determines how many times the record has been updated.
sysmoduser	Character	Determines the last person to update the record.
knownerror	Logical	General information field.
kpf.id	Character	Knowledge article number (for Knowlix for ServiceCenter integration).
ci.date.time	Character	General information field.
flow	Character	General information field.
server.id	Character	General information field.
units	Character	General information field.
value	Character	General information field.
port.index	Character	General information field.
system.state	Character	General information field.
payroll.no	Character	General information field.
critical.user	Character	General information field.
room.floor.ref	Character	General information field.
user.type	Character	General information field.
site.category	Character	Determines the critical state of the site involved.
total.loss	Logical	Determines if the issue involves a Total Loss of Service.
product.type	Character	Categorization field for issue identification.

Field Name	Types	Description
problem.type	Character	Categorization field for issue identification.
fix.type	Character	General information field.
no.SDU.fix	Logical	Determines if there's no correction to the issue.
resolved.by	Character	Determines the person that resolved the ticket when using the "resolved" alert status.
cost.centre	Character	General information field.
customer.no	Character	General information field.
unsuspend.time	Date	Determines when the ticket should be unsuspended when using the Suspend state.
critical.device	Logical	General information field.
serial.no	Character	General information field.
failing.serial.no	Character	General information field.
third.party.name	Array of Characters	General information field.
third.party.reference	Array of Characters	General information field.
third.party.referred	Array of Dates	General information field.
third.party.referred.by	Array of Characters	General information field.
class	Character	General information field.
alternate.contact	Character	General information field.
site.visit.date	Date	General information field.
site.visit.technician	Character	General information field.
operating.system	Character	General information field.
os.release.level	Character	General information field.
os.maint.level	Character	General information field.
manufacturer	Character	General information field.
failing.component	Character	General information field.
country	Character	General information field.
customer.reference	Character	General information field.
expd.response.time	Array of Dates	General information field.
oti.originator	Character	General information field.

Field Name	Types	Description
oti.originator.reference	Character	General information field.
oti.originator.version	Character	General information field.
oti.tosc.consumer	Character	General information field.
oti.tosc.consumer.reference	Character	General information field.
oti.tosc.provider	Character	General information field.
oti.tosc.provider.reference	Character	General information field.
oti.message.type	Character	General information field.
oti.fromsc.consumer	Character	General information field.
oti.fromsc.provider	Character	General information field.
oti.fromsc.consumer.reference	Character	General information field.
oti.fromsc.provider.reference	Character	General information field.
pending.change	Logical	General information field.
mandatory.asset	Logical	General information field.
reg.error	Logical	General information field.
cus.error	Logical	General information field.
variable1	Character	General information field.
variable2	Character	General information field.
variable3	Character	General information field.
call.origin	Character	General information field.
source	Character	General information field.
first.time.fix	Logical	General information field.
resolved.group	Character	General information field.
resolved.time	Date	Time when the ticket was resolved.
closed.group	Character	General information field.
sla.alert.time	Date	General information field.
contact.location	Character	General information field.
srvc.manager	Character	General information field.
srvc.del.manager	Character	General information field.
different.from.contact	Logical	General information field.

Field Name	Types	Description
alternate.fax	Character	General information field.
alternate.extension	Character	General information field.
alternate.phone	Character	General information field.
user.priority	Character	General information field.
sla.expire	Date	General information field.
corp.structure	Character	General information field.
res.anal.code	Character	General information field.
last.activity	Character	General information field.
mobile.checkout	Logical	General information field.
location.full.name	Character	General information field.
prev.update.time	Date	General information field.
mobile.update.time	Date	General information field.
reopen.time	Date	Should the ticket be reopened, determines the time it was reopened at.
reopened.by	Character	Should the ticket be reopened, determines the person who reopened the ticket.
billto	Character	General information field.
billtype	Character	General information field.
gl.number	Character	General information field.
time.spent	Character	General information field.
explanation	Array of Characters	General information field.
address.1	Character	General information field.
address.2	Character	General information field.
address.3	Character	General information field.
asset.status	Character	General information field.
city	Character	General information field.
contact.email	Character	General information field.
county	Character	General information field.
extension	Character	General information field.

Field Name	Types	Description
fax	Character	General information field.
parts	Arrayed Structure	General information field.
date	Date	General information field.
part.no	Character	General information field.
quantity	Number	General information field.
labor	Arrayed Structure	General information field.
labor.date	Date	General information field.
operator	Character	General information field.
hours.worked	Number	General information field.
li.contract.id	Number	General information field.
contract.consumed	Logical	General information field.
site	Character	General information field.
dump	Array of Characters	General information field.

Inventory Management Files

device

Fields in the **device** table:

Field Names	Types	Description
logical.name	Character	Unique name/ID for the device.
logical.name.attach	Character	Virtual join field for attachments.
logical.name.vj	Character	Additional virtual join field.
vendor	Character	General information field.
parent	Character	General information field.
model	Character	General information field.
network.name	Character	General information field.
serial.no.	Character	General information field.
location	Character	General information field.

Field Names	Types	Description
group	Character	General information field.
format.name	Character	Name of the format to use when displaying this record.
type	Character	Type of the selected device.
estatus	Character	General information field.
pstatus	Character	General information field.
id	Character	General information field.
last.update	Date	Time when the record was last updated.
updated.by	Character	Person who updated the record last.
description	Character	Description of the record.
view.name	Character	General information field.
container	Character	General information field.
end.point.1	Character	General information field.
end.point.2	Character	General information field.
pcount	Number	General information field.
nondevice	Logical	General information field.
problem.category	Character	Category to use when opening an Incident Management ticket.
table.name	Character	General information field.
network.address	Character	General information field.
objid	Character	General information field.
domain	Character	General information field.
protocol	Character	General information field.
protocol.addr	Character	General information field.
contact.name	Character	Name of the contact associated with this record.
part.no	Character	General information field.
istatus	Character	Install status of the record.
version	Character	General information field.
icount	Number	General information field.
subtype	Character	General information field.

Field Names	Types	Description
user.id	Character	General information field.
location.code	Character	General information field.
vendor.id	Character	General information field.
comments	Array of Characters	General information field.
building	Character	General information field.
floor	Character	General information field.
room	Character	General information field.
last.name	Character	General information field.
contract.no	Character	General information field.
service.agreement.no	Character	General information field.
is.down	Logical	General information field.
event.updated	Logical	General information field.
sysmodtime	Date	Revision tracking field for last update date and time.
y2k.status	Character	General information field.
asset.tag	Character	General information field.
sysmodcount	Number	Revision tracking field for update counts.
sysmoduser	Character	Revision tracking field for operator who updated the record.
contract.id	Number	General information field.
problem.priority	Character	Default priority when opening an Incident Management ticket.
family.name	Character	General information field.
family.uri	Character	General information field.
model.uri	Character	General information field.
vendor.uri	Character	General information field.
operating.system	Character	General information field.
os.uri	Character	General information field.
mtbf	Character	General information field.
total.downtime	Number	General information field.
install.date	Date	General information field.

Field Names	Types	Description
server.id	Character	General information field.
port.desc	Array of Characters	General information field.
port.index	Array of Numbers	General information field.
dest.mac	Array of Characters	General information field.
dest.port.index	Array of Numbers	General information field.
ind.removed	Logical	General information field.
breaks	Number	General information field.
primary.app.name	Character	General information field.
primary.app.uri	Character	General information field.
device.severity	Logical	General information field.
manufacturer	Character	General information field.
cost.centre	Character	General information field.
site.category	Character	General information field.
company	Character	General information field.
pending.change	Logical	General information field.
corp.structure	Character	General information field.
order.number	Character	Request Management order number that obtained this device.
order.line.item	Character	Request Management line item number that obtained this device.
ac.category	Character	General information field.
feature.id	Array of Characters	General information field.
feature.size	Array of Characters	General information field.
feature.description	Array of Characters	General information field.
nm.id	Number	IND mapping field

workstation

Fields in the workstation table:

Field Name	Types	Description
logical.name	Character	Unique name/ID for the device record.
primary.contact	Character	General information field.
primary.phone	Character	General information field.
building	Character	General information field.
floor	Character	General information field.
install.date	Date	General information field.
manufacturer	Character	General information field.
alternate.id	Character	General information field.
network.address	Character	General information field.
network.parent	Character	General information field.
local.software	Array of Characters	General information field.
remote.software	Array of Characters	General information field.
controlling.software	Array of Characters	General information field.
feature.id	Array of Characters	General information field.
feature.description	Array of Characters	General information field.
feature.vendor	Array of Characters	General information field.
comments	Array of Characters	General information field.
device.type	Character	Type of device for the record.
media	Array of Characters	General information field.
adapter	Array of Characters	General information field.
processor	Character	General information field.
math	Character	General information field.
bios	Character	General information field.
os.version	Character	General information field.
drivers	Array of Characters	General information field.
boot.files	Array of Characters	General information field.

Field Name	Types	Description
memory	Array of Characters	General information field.
operating.system	Character	General information field.
shell.version	Character	General information field.
bridge.address	Character	General information field.
gateway	Character	General information field.
subnet.mask	Character	General information field.
sw.name	Array of Characters	General information field.
sw.version	Array of Characters	General information field.
sw.vendor	Array of Characters	General information field.
sw.install	Array of Characters	General information field.
feature.size	Array of Characters	General information field.
dns.prime	Character	General information field.
dns.second	Character	General information field.
nis	Character	General information field.
nis.master	Character	General information field.
updated.by	Character	Last person that updated the record.
event.updated	Logical	Determines if the record was updated by an Event Services input event.
sysmodtime	Date	Revision tracking field for last update date and time.
sysmodcount	Number	Revision tracking field for update counts.
sysmoduser	Character	Revision tracking field for operator who updated the record.

Service Management Files

incidents

Fields in the incidents table:

Field Names	Types	Description
incident.id	Character	Record number assigned by the system.
vj.incident.id.3	Character	Virtual join alias field.
vj.incident.id.2	Character	Virtual join alias field.
vj.incident.id.1	Character	Virtual join alias field.
incident.id.vj	Character	Virtual join alias field.
problem.id	Number	No longer used.
contact.name	Character	Contact person for the call ticket.
severity	Character	Identifies the impact the issue reported has.
open.time	Date	Time the call ticket was opened.
update.time	Date	Time the call ticket was last updated.
opened.by	Character	Person who opened the call ticket.
updated.by	Character	Person who last updated the call ticket.
description	Array of Characters	A full description of the issue reported.
affected.item	Character	The device the contact person reports the issue with.
owner.name	Character	The person responsible for the call's completion.
open	Character	Status field for the call ticket.
callback.type	Character	The method used to notify the contact upon completion of the call.
callback.reason	Character	General information field.
null	**Structure**	
	Character	General information field.
resolution	Array of Characters	Description of the confirmed solution to the issue reported.
assignment	Array of Characters	Assignment group list for the reported issue on the call ticket.

Field Names	Types	Description
unassigned	Logical	Determines if the issue has yet to be assigned.
category	Character	Category that describes the issue reported.
handle.time	Date	Total time it took from the moment the call screen was brought up to the time the call ticket was committed to the database.
model	Character	General information field.
type	Character	General information field.
dept	Character	General information field.
location	Character	General information field.
close.time	Date	Time that the call ticket was closed.
closed.by	Character	Person who closed the call ticket.
solution.candidate	Logical	Determines if the issue and resolution are worth adding to the internal knowledge base.
agreement.id	Number	Identifies the SLA agreement code.
priority.code	Character	Determines the importance of this issue over others.
first.call	Logical	Determines if this issue was resolved as a First Call Resolution (no escalation required).
contract.id	Number	Contract Management code.
contract.consumed	Logical	Determines if this issue consumes available Contract Management services.
worked.time	Date	Determines the amount of time spent on resolving the issue reported.
sysmodcount	Number	System field that keeps track of the number of times the record was updated.
sysmoduser	Character	Determines the last person to update the record.
sysmodtime	Date	Determines the last time someone updated the record.
kpf.id	Character	The knowledge article code number (for Knowlix for ServiceCenter integration).
payroll.no	Character	General information field.
critical.user	Logical	General information field.
room	Character	General information field.
user.type	Character	General information field.

Field Names	Types	Description
site.category	Character	SLA site identification type.
total.loss	Logical	Determines if the contact person is experiencing a Total Loss of Service.
temp.update	Array of Characters	Update field for the call ticket.
subcategory	Character	Categorization field for issue identification.
product.type	Character	Categorization field for issue identification.
problem.type	Character	Categorization field for issue identification.
failed.entitlement	Logical	Based on Contract Management service dates determines if the contact is entitled to service.
cost.centre	Character	General information field.
contact.location	Character	General information field.
phone	Character	General information field.
extension	Character	General information field.
critical.device	Logical	Determines if the device reported is a critical asset to the enterprise.
cause.code	Character	General information field.
resolution.code	Character	General information field.
company	Character	General information field.
company.id	Character	General information field.
vendor	Character	General information field.
class	Character	General information field.
country	Character	General information field.
alternate.contact	Character	General information field.
engineer	Logical	General information field.
different.from.contact	Logical	Determines if the alternate contact is different from the contact on the call ticket.
alternate.phone	Character	General information field.
alternate.extension	Character	General information field.
customer.reference	Character	General information field.
fax	Character	General information field.

Field Names	Types	Description
alternate.fax	Character	General information field.
pending.change	Logical	General information field.
mandatory.asset	Logical	General information field.
floor	Character	General information field.
building	Character	General information field.
variable1	Character	General information field.
variable2	Character	General information field.
variable3	Character	General information field.
corp.structure	Character	General information field.
corp.level1	Character	General information field.
corp.level2	Character	General information field.
corp.level3	Character	General information field.
contact.email	Character	General information field.
location.full.name	Character	General information field.
contact.first	Character	General information field.
contact.last	Character	General information field.
billto	Character	General information field.
billtype	Character	General information field.
gl.number	Character	General information field.
entitlement.ref	Character	General information field.

Index

E

F

G

H

I

Q

R

S

T

U

V

W


Peregrine
SYSTEMS