Peregrine

ServiceCenter

Database Conversion and RDBMS Support

Release 6

PART NO: DSC-6-EN12

Peregrine SYSTEMS®

The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document. The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental. 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 of the document applies to version 6.0 of the licensed program.

040507

Peregrine Systems, Inc.
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 Conversion and RDBMS Support

The *Database Conversion and RDBMS Support* aids experienced ServiceCenter® system and database administrators responsible for installing and implementing the ServiceCenter databases or who will be hosting ServiceCenter data and assisting in database conversion.

This guide provides the information necessary for converting ServiceCenter data from its internal format to a storage location on a commercial Relational Database Management System (RDBMS). It also provides technical details on the conversion process and optimization tips.

Topics covered in the preface include:

About this guide

Chapters in this guide include:

- *Introduction to RDBMS Conversion* on page 15 — introduces the conversion process, discusses its pros and cons, and answers some frequently asked questions.
- *Conversion Strategy on page 21* — 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 you can use to help get your data mapped more quickly and efficiently.
- *Data Mapping* on page 63 — 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 85 — explains the configuration necessary before converting ServiceCenter to a Relational Database Management System (RDBMS).
- *Conversion to an RDBMS* on page 119 — explains the process of converting the P4 file system to an RDBMS.
- *Post Conversion* on page 145 — discusses the procedure for editing a database after conversion to an RDBMS format.
- *Tuning for Performance* on page 163 — discusses procedures for enhancing performance for SQL, Oracle, and Sybase RDBMS formats.
- *Troubleshooting* on page 175 — 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 183 — 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 191 — discusses the procedure for converting data in an RDBMS database back into the ServiceCenter P4 format.

Appendices in this guide include:

- *Characteristics of ServiceCenter Files on page 193* — contains information necessary to configure ServiceCenter files for optimal performance.
- *Data Definitions* on page 243 — contains list of data definitions for ServiceCenter Files.

Knowledge requirements

While this guide explains various aspects of ServiceCenter database administration, a certain level of knowledge of ServiceCenter is presumed. You should be thoroughly familiar with your RDBMS and with ServiceCenter administration and tailoring before reading this guide. For more information on ServiceCenter applications, system administration, and system tailoring, refer to the ServiceCenter online help.

Documentation web site

For a complete listing of the current ServiceCenter documentation, refer to 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.

Sample screens and examples

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

Contacting Peregrine Systems

For further information and assistance with this release, you can download documentation or schedule training.

Customer Support

For further information and assistance, contact Peregrine Systems Customer Support at support.peregrine.com.

If the KnowledgeBase does not contain an article that addresses your concerns, you can search for information by product; search discussion forums; and search for product downloads.

Documentation Web site

For a complete listing of current ServiceCenter documentation, see the Documentation pages on the Peregrine Customer Support Web.

You can view PDF files, including release notes using Adobe Reader™, which is available at www.adobe.com.

Education Services Web site

Peregrine Systems offers classroom training anywhere in the world, as well as "at your desk" training via the Internet. For a complete listing of Peregrine's training courses, see www.peregrine.com/education. You can also contact Peregrine Systems Education Services at +1 858.794.5009.

CHAPTER 1

Introduction to RDBMS Conversion

This chapter provides information necessary for making the decision on whether or not to convert ServiceCenter® data from the native P4 system to a Relational Database Management System (RDBMS). It provides an overview of ServiceCenter's Architecture, and a discussion of the conversion process. It also answers some frequently asked questions. Its intended audience consists of managers, system administrators and database administrators.

Topics in this chapter include:

ServiceCenter architecture

ServiceCenter uses the native client libraries of the RDBMS to manage database I/O, with the exception of managing data between ServiceCenter and Microsoft SQL Server, which is handled with 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.

See the compatibility matrix on the ServiceCenter Customer Support website for a list of supported RDBMS servers by operating system.

Points to consider when 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, you can convert data storage in ServiceCenter from its internal P4 format to a storage location on a commercial RDBMS.

The P4 file system takes less space and performs 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 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. Instead, they require that the data be normalized. Therefore, information that can be stored as a single record within the P4 file system may require multiple tables within a relational database. The system must join these tables 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 design minimizes the possibility of corruption to the file system due to a system failure. However, such things as power outages at the wrong time can corrupt a P4 file system. Because there is no back out capability, if corruption occurs you must either patch it, or restore from a backup and re-enter the lost data.

Relational systems have dynamic back out utilities to remove in-flight transactions in such situations and forward recovery capabilities for those cases where you must restore from a backup.

Many organizations have standards 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, you will need a fully trained and experienced database administrator.

The P4 file system is easier to handle and will perform better out-of-box, but still requires a fully trained and experienced person. If you use P4 only, you can use P4 specialists without additional database administrator staff. ServiceCenter will run on most popular relational database systems and it will run without a separate RDBMS. If you have experienced database administrator staff currently in place, you do not need additional P4 specialists.

When making a decision of whether or not to convert to an RDBMS, do not overlook the possibility of shadowing. Shadowing allows you to copy your production system to an RDBMS for creating reports while retaining your working file system in P4. Shadowing updates your file system 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

You can convert your system to an RDBMS at any time after installing ServiceCenter. For discussion of this topic, see *Why does Peregrine recommend initial development in P4 and then conversion to an RDBMS?* on page 19.

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

Conversion requires the following major steps

1. Install ServiceCenter. Refer to the ServiceCenter installation guide for instructions.
2. Configure ServiceCenter. Refer to the ServiceCenter online help for further configuration instructions.
   - Set the ServiceCenter case sensitivity for sort order so that it does not conflict with that of the RDBMS.
3. Install the RDBMS. Refer to the installation and configuration documentation for the RDBMS for information on how to do this.
   a. Set the RDBMS case sensitivity for sort order so that it does not conflict with that of ServiceCenter.

   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.

   b. Set the codepage to Unicode (UTF8). This is optional, but we recommend it. If you do set the codepage to Unicode (UTF8) set `dblanguage:utf8` in the `sc.ini` file.

   Note: Microsoft SQL Server does not support conversion to UTF8. However, you can convert fields individually to UTF16 instead. See the Microsoft SQL Server documentation for information on how to do this.
4. Map the RDBMS Data to ServiceCenter. See *Conversion Strategy on page 21*.
5. Set up ServiceCenter and the RDBMS for the conversion process. See *Preconversion Configuration* on page 85.
6. Convert the data. See *Conversion to an RDBMS* on page 119.
7. Edit the database after conversion. See *Post Conversion* on page 145.

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 how we support Oracle, Sybase, Microsoft SQL server, and DB2Universal from ServiceCenter. This gives our clients maximum independence from any proprietary database technology since it permits them to use the RDBMS of their choice. The mapping for Oracle is much different from the mapping for SQL server but the dbdict shields these differences 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 as desired, then we 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 you have added new fields. The system will also automatically detect fields added to tables in the RDBMS and add those fields to the dbdict. However, 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 scans every record in every file and determines the maximum size of each field. The more data that exists in the file, the more accurate these numbers will be. If you convert a file containing no data, then you 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?

Yes they are. ServiceCenter 6 stores the IR index data within the database in the scirexpert file. It creates this file automatically the first time you run an IR Regen. When you have regenned all IR indexes, then all data will be within scirexpert. No flat files will exist and therefore a backup of the database will also backup the IR index file. If the dbdict file has been converted to an RDBMS then scirexpert will be created in the same RDBMS.

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.

Can I delete the P4 file system after a complete conversion of all ServiceCenter data to a relational database management system (RDBMS)?

After converting all tables, you may delete the P4 files. However, If you delete the P4 files, you must specify the sqldictionary parameter in the sc.ini file, so ServiceCenter knows how to find the dbdict when you start the system.

Important: If you delete the P4 files and do not specify the sqldictionary parameter, ServiceCenter will not start.

Does the P4 file system 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 backups for the target RDBMS are sufficient.

2 Conversion Strategy

CHAPTER

This chapter explains 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 audience consists of system administrators and database administrators.

Topics in this chapter include:

Analyzing ServiceCenter tables for conversion

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, a single ServiceCenter record is stored across multiple rows in multiple tables on a commercial RDBMS. For more information, see *ServiceCenter records* on page 23.

Important: Make sure that your database is well-tuned before beginning the conversion process. A well-tuned database converts most easily and quickly.

Choose the tables to convert

ServiceCenter stores the definitions of tables called database dictionaries or dbdicts. You must evaluate each field in a dbdict to determine if this data is needed in the RDBMS.

You can convert any table, to the RDBMS, but you do not have to convert all tables. The tables chosen for conversion differ from company to company. One possible determination is to choose to convert all tables where you need reporting done.

Determine which files function well together

Using the information given in *Tools to determine file access characteristics* on page 225 to analyze which tables function well together.

- Put tables that function well together within the same the RDBMS tablespace. For example, include the following tables in one tablespace:
  - Contacts
  - Location
  - Category
  - Assignment
  - Clocks

- Put the number and counter tables in their own tablespace since the system accesses these tables for each new record in call, problem, request and change.
- Put tables in the same module, which ServiceCenter updates simultaneously, such as problem and probsummary, into different tablespaces to improve performance.

Calculate the required amount of space

Before converting a table to an RDBMS, calculate the proper space for each table as well as the entire tablespace.

To do this calculation, you need the:

- 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.

ServiceCenter records

A *record* is a discreet unit of related data that ServiceCenter can load, manipulate, and save as a block. ServiceCenter records have defined structures, such as that required for all records containing Incident tickets.

A *file* is the collection of all records with a given structure inside of ServiceCenter. A ServiceCenter file is roughly equivalent to an RDBMS table and contains a large number of similarly structured elements that ServiceCenter can retrieve for manipulation using indexed search methods.

Despite the similarities, ServiceCenter records do differ from rows in RDBMS tables. 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 records

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

As an example of a simple date record, we will create a simple ServiceCenter record for tracking travel and expense information, and call it the **travel** record.

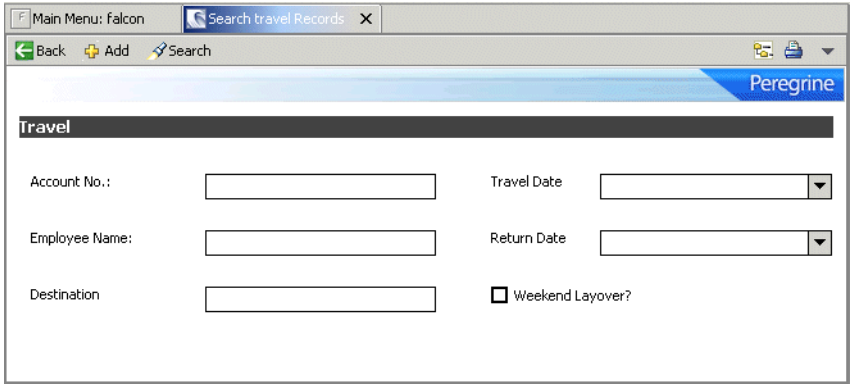


Figure 2-1: A simple ServiceCenter data record

The fields in a travel record have the following data types:

Field Label	Field Name	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 Database Dictionary for this record would look like this:

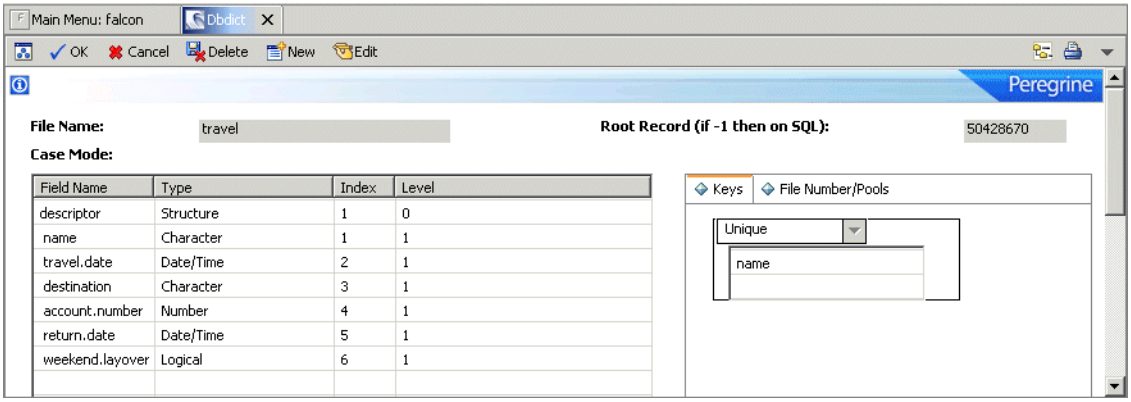
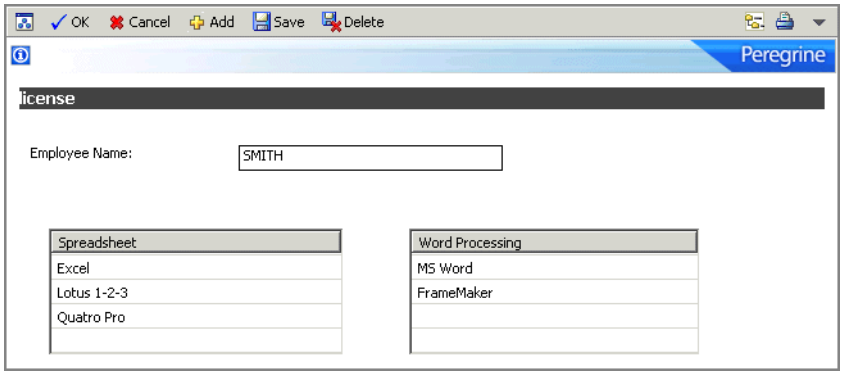


Figure 2-2: Database Dictionary file

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 records with arrays

As an example of data records with arrays, we will create a simple ServiceCenter record for tracking software license information, and call it the **license** record.



The fields in this record 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 for this record would look like this:

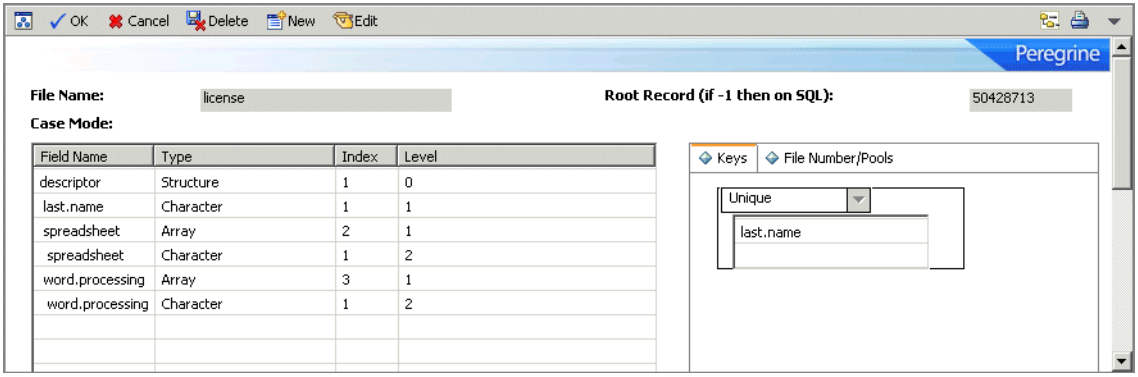


Figure 2-3: Database Dictionary file for arrays of characters

Data records with arrays of structures

Arrays of structures are like sub-tables with multiple rows, similarly formatted. All arrays of structures in ServiceCenter 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.

As an example of data records with arrays of structures, we will create a ServiceCenter record for tracking people checked into the hospital, and call it the hospital record.

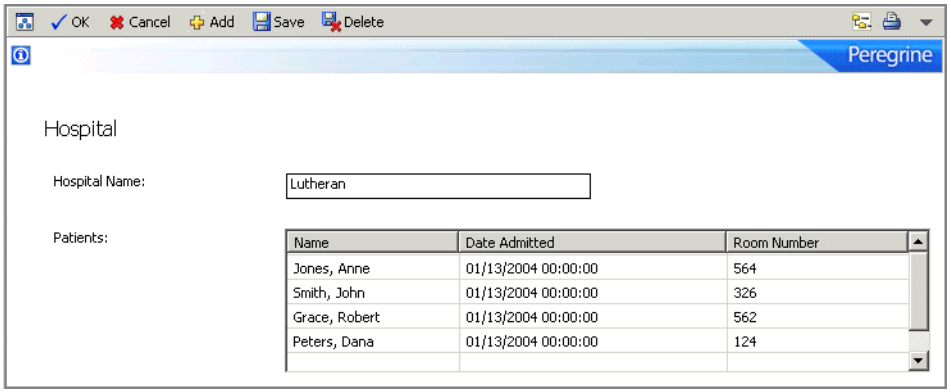


Figure 2-4: Array of structures

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 for this record would look like this:

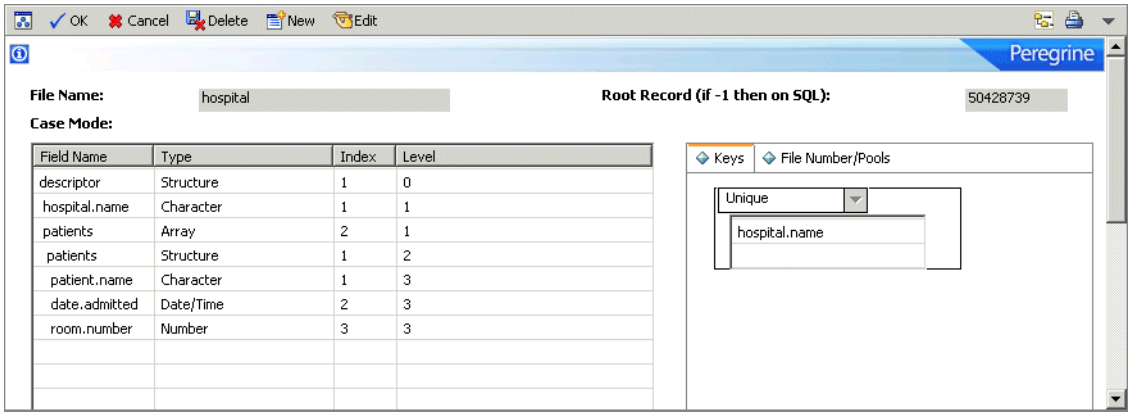


Figure 2-5: Database Dictionary record for *hospital* file

Elemental data types in ServiceCenter

All ServiceCenter records are made up of smaller, elemental data types. ServiceCenter supports 11 different types of data that it can store in records.

ServiceCenter data types are divided into three categories:

- *Simple data types* on page 28
- *Internal data types* on page 29
- *Complex data types* on page 29

Simple data types

The vast majority of operational data in a ServiceCenter system is stored internally in one of the four, *simple* data types. These types are 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.

Data Type	Description
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.

For examples, see *Simple data records* on page 24.

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.

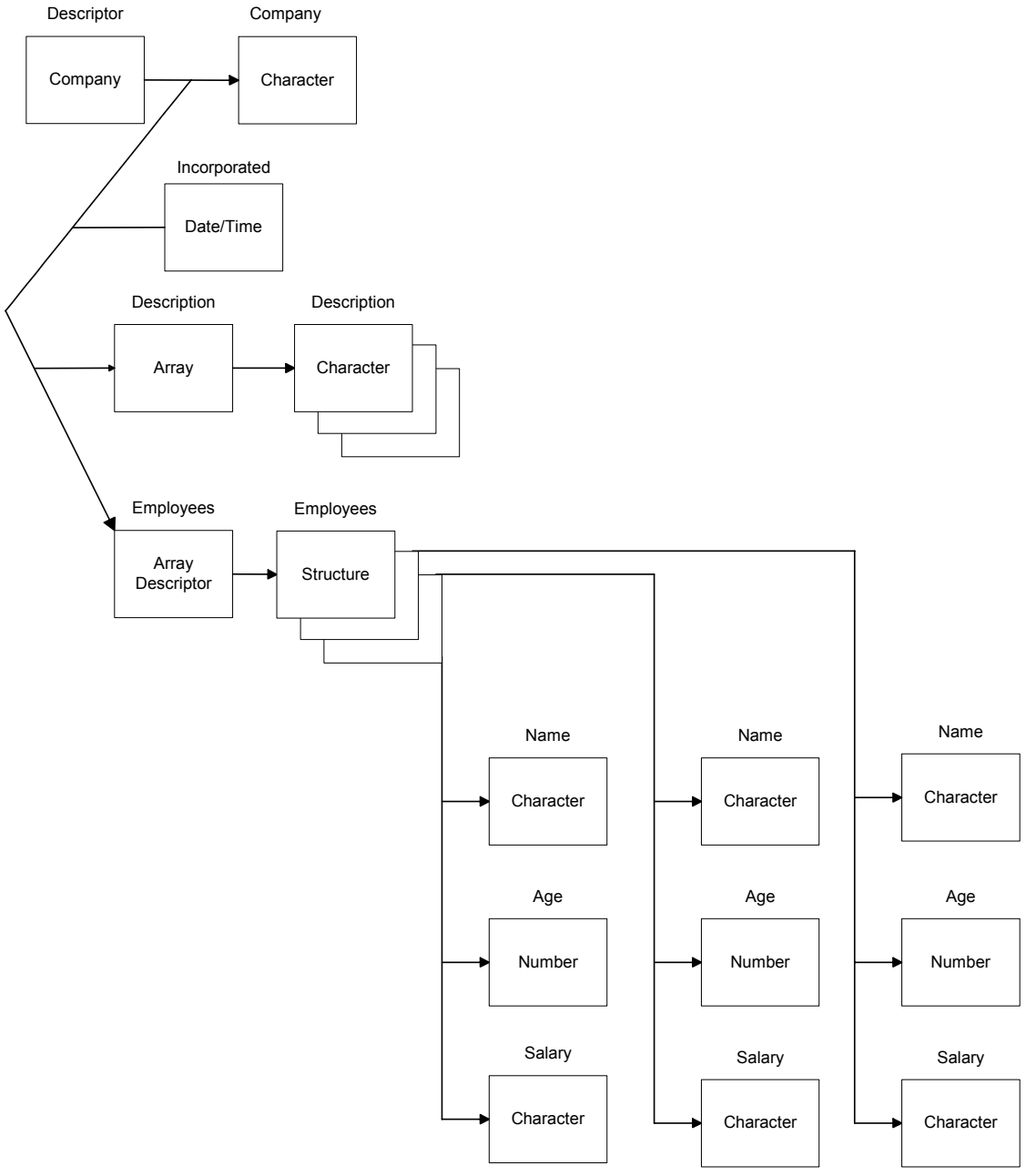
Data Type	Description
Label	Used to store offset information when ServiceCenter is compiling RAD language programs. Labels dictate the flow of RAD programs.
File	Used to pass records between various RAD applications.
Operator	The parsed version of a RAD language expression such as "`$name=operator()`"
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 (for example, an array of character data types or an array of date/time data types.) For examples, see *Data records with arrays* on page 26.
Structure	A data type that contains a collection of other data types (for example, record within a record.) For examples, see*Data records with arrays of structures* on page 27.

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 map to analogous RDBMS data types, and the ServiceCenter internal data types map 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 63.

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. The example table constructions are based on the ServiceCenter record created in *Simple data records* on page 24.

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 creates 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
character	char*	char*	char*
number	float	float	float
logical	char(1)†	char(1)†	char(1)†
date/time	datetime	date	timestamp

* 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 data records* on page 24, 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)
travel.date	date/time	travelm1	travel_date	datetime*
return.date	date/time	travelm1	travel_date	datetime*
weekend.layover	logical	travelm1	weekend_layover	char(1)

* Assumes 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 33), 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
Label	char*	char*	char*
File	image	long raw	long varchar
Operator	binary*	raw*	unknown
Expression	binary*	raw*	unknown

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

Mapping complex data types

We discuss the mapping of two major complex data types.

- *Arrays of characters* on page 34
- *Data records with arrays of structures* on page 27

Arrays of characters

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

When you type some text into a multi-line edit field, ServiceCenter chops text into 60 character sections. 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 in a line may varies with the font and font size chosen.

For example, if the font size is such that 80 characters appear on a line, ServiceCenter stores the first 60 characters in the first array element followed by a Tab character signalling that there is more to come in the next array element. ServiceCenter generates the second element from the remaining 20 characters of the first plus 40 out of the second line followed by another Tab and so on. If you press Enter, or otherwise enter a carriage return,

ServiceCenter does not fill that array element with whatever is in the next line, but move to the next array element. When reading the data, 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 35
- *Field in the alias table* on page 36
- *BLOB in the main table* on page 38
- *BLOB in the alias table* on page 39
- *Multi-row array tables* on page 40

The example table constructions (except for multi-row array tables) are based on the ServiceCenter record created in *Data records with arrays* on page 26.

See also *LOB support* on page 42.

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
long text	text	long	long varchar
for bit data	image	long raw	long varchar for bit

Using this strategy, the sample table for the **license** file created in *Data records with arrays* on page 26, would 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, you can only map one array of characters 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 maps to the main table. Additional arrays of characters map as fields in an alias table (*Field in the alias table* on page 36). In our example, the spreadsheet array is in an Oracle main table, while the word.processing array is in the alias table.

Note: By default, each array is mapped as a long data type. This is because ServiceCenter does not know how large the array will grow to be. If you know that an array will have a maximum size, then change the default mapping from *long* to *varchar(xxx)*, and the system will perform better.

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, created in *Data records with arrays* on page 26, would 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, and so on.) It does this by constructing an alias table containing the primary key of the main table and the long text field.

Using this strategy, our example produces the following SQL statements to create the RDBMS tables:

```
CREATE TABLE licensem1 (
last_name char (60) NULL
)|
CREATE TABLE licensea1 (
last_name char(60) NULL,
spreadsheet text NULL
)
CREATE TABLE licensea2 (
last_name char(60) 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
long binary	image	long raw	long varchar for bit

Using this strategy, the sample table for the **license** file created in *Data records with arrays* on page 26, would 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 maps 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 maps to the main table. Additional arrays of characters map as fields in an alias table (*BLOB in the alias table* on page 39). In our example, the word.processing array is 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 created in *Data records with arrays* on page 26, would 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. ServiceCenter needs a way to associate the row in the main table (m1) with the matching row in the alias table (a1, a2, and so on.) It does this by constructing an alias table containing the primary key of the main table and the long text field.

With this scheme, our example produces the following the following SQL statements to create the RDBMS tables:

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

Multi-row array tables

If you have a spread sheet field, ARRAY, which contains a record over 32K, you can use Multi-Row Array Table to avoid data truncation. *LOB support* on page 42 as a possible alternative for Multi-Row Array Tables.

Note: Do not use this option as the default. It is very expensive in terms of performance.

Using 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 created in *Data records with arrays* on page 26, would 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	varchar(255)*
word.processing	array of characters	licensea2	word_processing	varchar(255)*

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

This 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”,“FrameMaker”}

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

Arrays of structures

Arrays of structures are like sub-tables with multiple rows, similarly formatted. 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. See *BLOB in the main table* on page 38.

LOB support

This section outlines how to use LOB (Large Object) and datatypes in supported external databases. ServiceCenter supports the LOB data type for the external databases Oracle 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 40 are a possible alternative for LOB Support.

CLOB or BLOB datatypes

You can use CLOB and BLOB (Character LOB and Binary LOB) 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.
- ServiceCenter can process a CLOB or BLOB field 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.

Setting up LOB datatypes for Oracle

To use CLOB or BLOB datatype, you must make changes in the sqldbinfo file. See the following sections for instructions:

- *To set up LOB datatype in the sqldbinfo file for Oracle* on page 43
- *To set up LOB datatype in the sqldbinfo file for DB2Universal* on page 46

To set up LOB datatype in the sqldbinfo file for Oracle

1 Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2 Click **SQL Database Information** to open the sqldbinfo file.

The system displays the SQL DB Type search form.

3 Select oracle8 from the **SQL DB Type** list box and click **Search** or press Enter.
If the form display more than one option, double click your choice to open it.

The system displays the sql.db.info form:

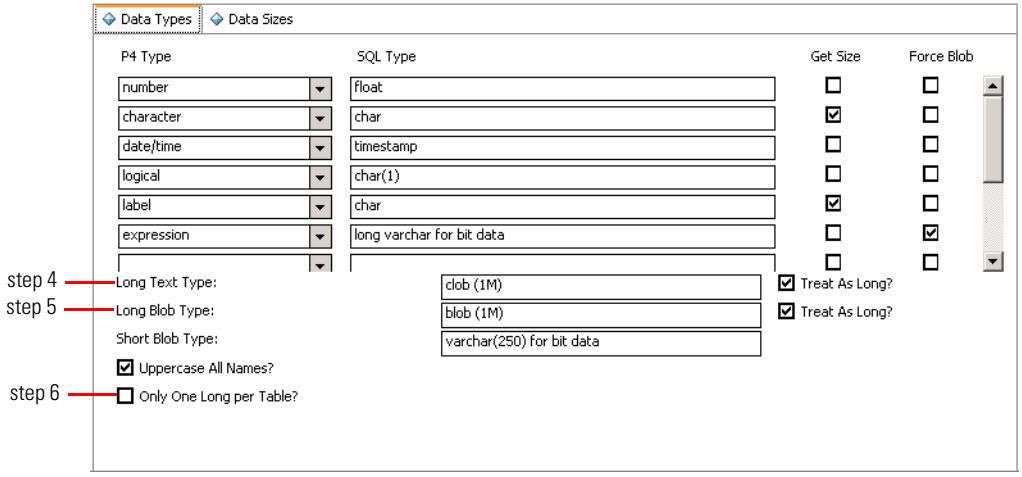


Figure 2-6: The sql.db.info form - Data Types tab

4 In the **Long text type:** text box, replace **long** with **CLOB**; uncheck **Treat as Long?** in the corresponding checkbox, if it is checked.

5 In the **Long Blob type:** text box, replace **long raw** with **BLOB**; uncheck **Treat as Long?** in the corresponding checkbox, if it is checked.

6 Uncheck the **Only One Long per Table?** check box, if it is checked.

7 Select the **Data Sizes** tab.

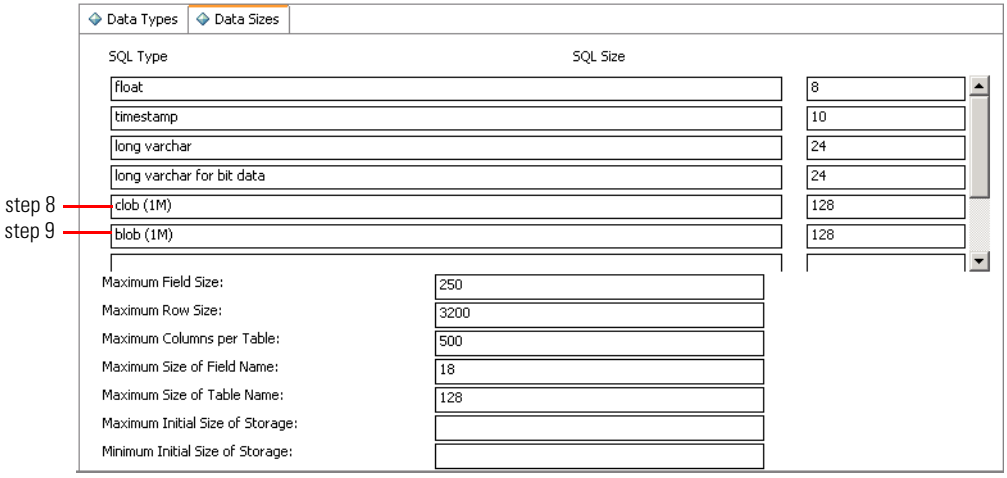


Figure 2-7: The sql.db.info form - Data Sizes tab

8 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.

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

10 Click **Save** to the settings and click **OK** to exit the form.

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 you use only BLOB/CLOB(nK/ nM/ nG), then DB2 defaults to NOT COMPACT and LOGGED.

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) is 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 you 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 you 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 you can reconstruct it. The default is LOGGED.

If you specify LOGGED, 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 you specify NOT LOGGED, 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 LOB datatype in the sqldbinfo file for DB2Universal

1 Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2 Click **SQL Database Information** to open the sqldbinfo file.

The system displays the SQL DB Type search form.

3 Select **db2universal** from the **SQL DB Type** list box and click **Search** or press Enter.

The system displays the sql.db.info form:

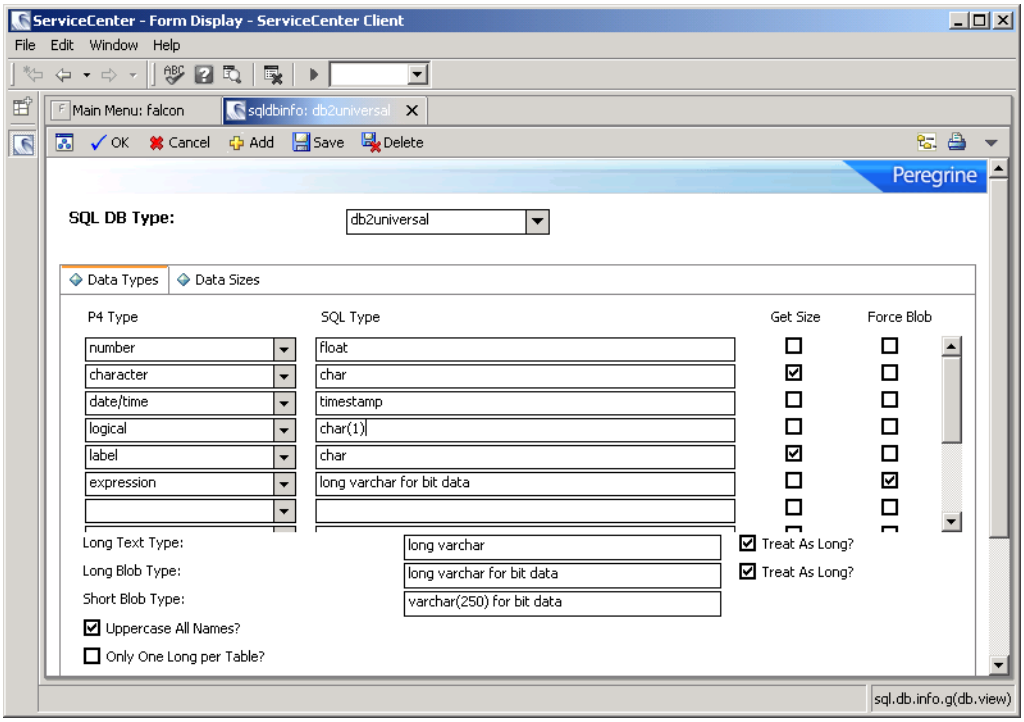


Figure 2-8: The sql.db.info form for DB2Universal

4 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.

5 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.

6 Uncheck **Only One Long per Table** if it is checked as default.

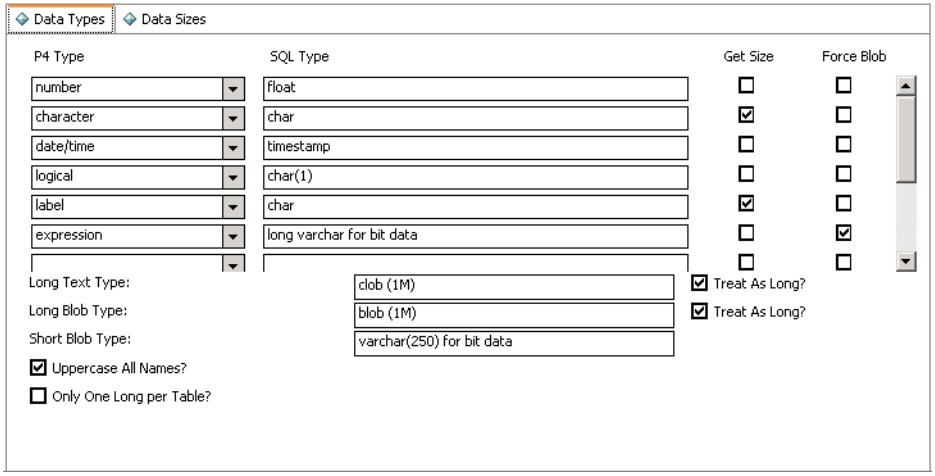


Figure 2-9: Setting data types for DB2Universal

7 Select the **Data Sizes** tab.

8 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 49.

9 Copy what you entered in step 4, and paste it here. The value added here must exactly match what you entered in step 4.

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

10 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 what you entered in step 5, and paste it here. The value here must exactly match what was entered in step 5.

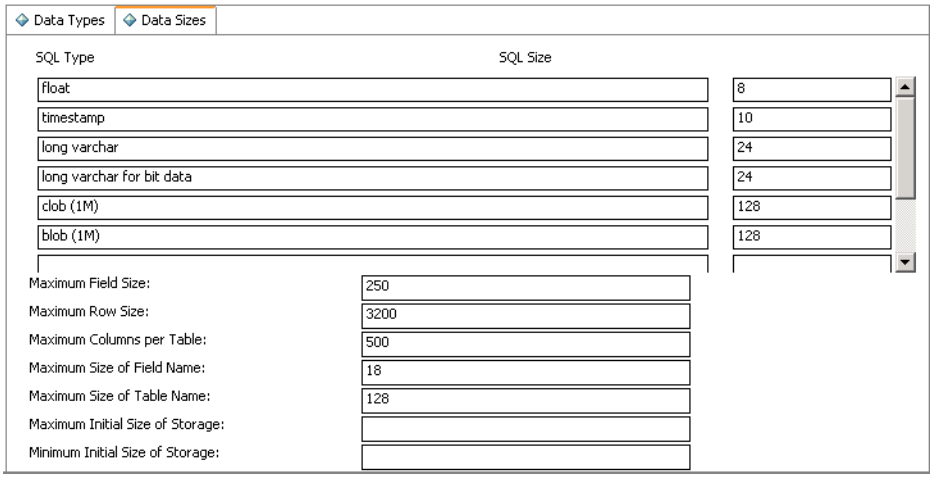


Figure 2-10: Setting data sizes for DB2Universal

11 Click **Save** to the settings and click **OK** to exit the form.

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, calculate N as:

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

Change the maximum row size and maximum columns per table, 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. You have the option to choose LOB over LONG and choose a LOB tablespace that is dedicated to store large objects. You have the 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 specifies 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, you can move the text/image columns in Sybase to a separate LOB tablespace after a table is created. Check the Sybase manual for details.

Specifying LOB tablespaces for Oracle

In Oracle 8.0.x you can store the LOB index separately from the LOB segment. If you specify a tablespace for the LOB segment, then the conversion process will place the LOB index in the same tablespace unless you explicitly specify different tablespace. You may use system-generated names or specify names for the LOB segments.

In Oracle 8I, even though you can specify a tablespace for the LOB index, Oracle ignores it and placing the index in the same tablespace as the LOB data. The LOB index tablespace is ignored if it is different from the LOB tablespace.

When using LOB data, you may notice during the conversion of some files from P4 to Oracle. This is because Oracle inserts a LOB datum in two steps. First it selects the LOB locator(s). Then it inserts the datum into the selected locator(s). One way to speed up the conversion you can 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 database administrator for a solution you detect a severe slowness during file conversion.

Specifying LOB Tablespaces 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 database used to store LOB data is called 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. Key fields from the ServiceCenter records are 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, using that index, only those records whose arrays contain a specific element. In order to allow this same capability in RDBMS, ServiceCenter does not allow you to map an indexed array as a blob or a long text field. ServiceCenter always maps an indexed array to an external array table.

ServiceCenter's record retrieval strategy

ServiceCenter performs record retrieval in three phases:

- Primary key fetch
- Initial record selection
- Block selection

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, created in *Data records with arrays* on page 26, 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)*

To fetch all users whose last name begins with the letter "J"

▶ Enter the J in the Last name field, and initiate the search. Entering J in the last.name files causes ServiceCenter to issue this SQL query:

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

ServiceCenter fetches up to 500 records from the result set and closes the query.

For the purpose of this example, assume the query returned the following primary keys:

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

Retrieval phase 2: initial record selection

After ServiceCenter initiates a query, it always fully populates the first record in the result set. It does this by selecting data from the main SQL table and any associated alias tables. ServiceCenter does 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 that it needs 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

ServiceCenter handles all subsequent requests records from the query as block fetches as described in *Retrieval phase 3: block selection* on page 55. Once ServiceCenter has exhausted its initial request for 500 primary key values, it repeats *Retrieval phase 1: primary key fetch* on page 53 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 pure performance and 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 arrays of characters either as long text fields or long binary fields in the main RDBMS table. However, 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.

To do this, locate the offset within the row itself where the appropriate column is located. If the table contains fixed length columns, you can calculate this offset based upon the table schema. If, however, the table contains variable length columns, then you must scan the row 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. You can improve the speed of these selects by indexing your RDBMS tables efficiently. Avoid the temptation to under-index; this may speed updates or inserts, but slows retrieval.

Defining system files

ServiceCenter stores all control files as *system* files in the RDBMS. When you convert the P4 file system, 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, you must define these files as system files in the **systables** record in the **globallists** file prior to RDBMS conversion. The conversion builds the **systables** global list from records in the **sqlsystemtables** file.

To define a file as a system file

1 Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2 Click **SQL System Tables** to open the **sqlsystemtables** file.

The system displays a blank SQL system tables record.

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

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

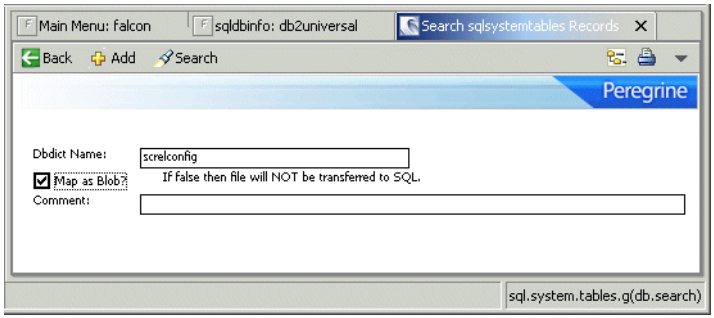


Figure 2-11: SQL system tables file

5 Click Add.

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 using 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, store arrays upon which you want to report as either long text fields or as separate array tables. If you choose to use long text fields, check to ensure that your reporting package can handle the data in question.

Know how arrays are used

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 improves 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 used to remember previous queries are filled and cause performance degradation.

If you have enabled reusable SQL, the query issued changes to the following:

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

The question mark (?) in this query is a *placeholder*. Before issuing the query, we provide the system with the value that it should substitute 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, you must analyze all tables and indexes. 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.

Use commands like these to create a script to 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, for example, information on users within an organization, including email addresses, phone, fax, user IDs, passwords, and privilege levels. Multiple applications can access this central point, there is no need to duplicate 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. Perform actual deletion of data from LDAP from LDAP directly.

Refer to the ServiceCenter online help 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 explains mapping in ServiceCenter®, and assists in mapping data quickly and efficiently. Its intended audience consists of system administrators and database administrators planning to convert a ServiceCenter system to an RDBMS.

Topics in this chapter include:

Out-of-Box mapping

Since the SQL mapping creates the SQL attributes for P4 files, the performance of ServiceCenter in a RDBMS relates to how you map the files. To simplify the mapping process and assist you in tuning ServiceCenter performance in RDBMS, ServiceCenter includes a standard out-of-box mapping. You can use the standard mapping as is, or modify it to suit the needs of your company.

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.

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 you want to use during the conversion process. The compatibility matrix on the customer support website for a list of supported database types.

Step 2 Create the table spaces. See *Tablespaces* on page 64.

Step 3 Modify the SQL mappings. See *SQL mapping* on page 67.

Step 4 Select the SQL mapping options. See *SQL mapping options* on page 69.

Step 5 Use the out-of-box mapping when doing the conversion process. See *Conversion to an RDBMS* on page 119.

Tablespaces

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

DB2 Universal mappings

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

The db2universal and db2ux mappings are identical except that db2ux uses CLOB and BLOB, and therefore, db2ux mapping requires a 32K tablespace. Contact your DBA to ensure that the tablespaces are available and configured with a page size of 32 K.

The Db2ux mapping 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. You can change them back to long/long raw or long varchar/long varchar for bit by resetting the parameters in the file sqldbinfo. If you reset the parameters in sqldbinfo to not use clob/blob, then do not create SCLOB, because ServiceCenter will not use it.

Oracle mappings

The Oracle mapping has a Max Field size 255 and the Oracle8 mapping has a Max Field size of 4000.

The Oracle 8 mapping does not use LOBs, and the Oracle8ix/oracle9 does use LOBS.

Important: If you are on UNIX and use the scenter.oracle executable, use the Oracle 8 group because the scenter.oracle does not support LOBs.

- The Oracle8i mapping 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.*
Tablespace	SCTBCOLD	Hosts tables created for files of low usage
Tablespace	SCINCOLD	Hosts indices created for files of low usage

*By default, Oracle8i maps ServiceCenter binary data or Array of characters as clob/blob. You can change them back to long/long raw or long varchar/long varchar for bit by resetting the parameters in the file sqldbinfo. If you reset the parameters in sqldbinfo to not use clob/blob, then do not create SCLOB, because ServiceCenter will not use it.

SQL Server 7 and SQL Server 2k

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

Sybase

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, ServiceCenter saves a file's DDL options in a file called **sqlmapping**. ServiceCenter uses the mappings in **sqlmapping** if you choose the option to use existing mappings during conversion.

The **sqlmapping** file contains the RDBMS attributes that ServiceCenter uses during table creation. These attributes include table names, data type, table sql options, and others. Use Database Manager to view the **sqlmapping** file and edit the saved mappings. The table **sqloptions** contain the table space, index space and object attributes used in the CREATE TABLE and CREATE INDEX statements.

The advantage of saving SQL mappings in the **sqlmapping** file is that it allows you to choose whether the P4 files share the same default settings or each individual table has its own table options.

To view or modify a mapping

1. Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

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.
5. Click **Search** or press Enter.

The system displays the requested data map.

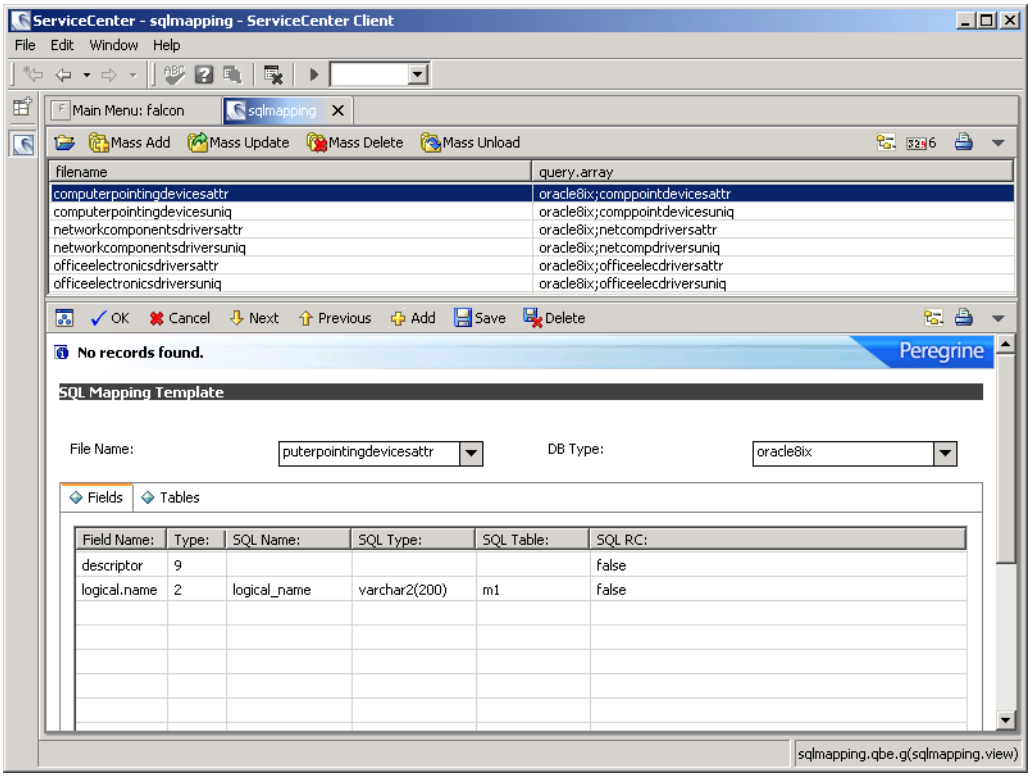


Figure 3-1: Data map

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

Note: Add new fields through the Database Dictionary application. (For instructions, refer to *Database Dictionary* in the ServiceCenter online help.)

7 Click **Save** when you finish modifying the mapping.

The system displays a prompt asking if you want to save the new mapping.

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

The system displays another prompt 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, you can open the sqlmapoptions form for the selected record, from this location, by selecting **Get Mapping Options** from the ServiceCenter pull-down options menu, or by clicking the **DB Type** or **Table Name** buttons. For more information, see *SQL mapping options* on page 69.

SQL mapping options

The system uses the sqlmapoptions form as an interface to allow you to assign the specified DDL options to an individual table for use 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 you do not assign a table's DDL options in the sqlmapoptions file, the default options will be used and shared by all tables.

To edit the sqlmapoptions file

1. Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2. Click **Update Mapping Template.**
3. Select the **DB Type** and click **Search** or press Enter.
4. Open an sql mapping record:
   a. Select the **Tables** tab.
   b. Select any Table Name listed on the Tables tab.
5. Select **Edit Map Options** from the pull-down options menu.

6 The system displays the **sqlmapoptions**record for the table you selected.

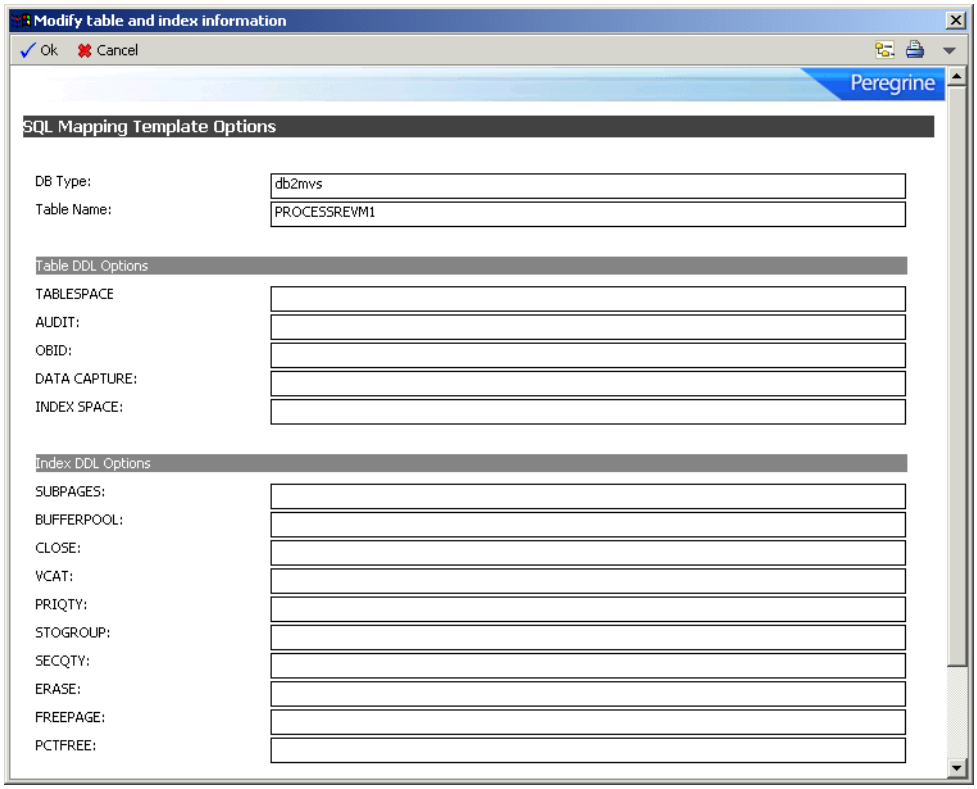


Figure 3-2: SQL Mapping Template

7 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.

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

- *DB2Universal* on page 80
- *Microsoft SQL Server* on page 81
- *Oracle* on page 82
- *Sybase* on page 83

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. Refer to the RDBMS's administrative guide for details.

Custom mappings

ServiceCenter's default mapping options are discussed in *RDBMS mapping strategy* on page 31. The mappings we provide are discussed in *Out-of-Box mapping* on page 64. 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 Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2 Click **SQL Database Information** to open the sql.db.info file.

3 Select the RDBMS from the **SQL DB Type** drop-down list, and click **Search**.

The system displays the following **sql.db.info** form for the RDBMS server:

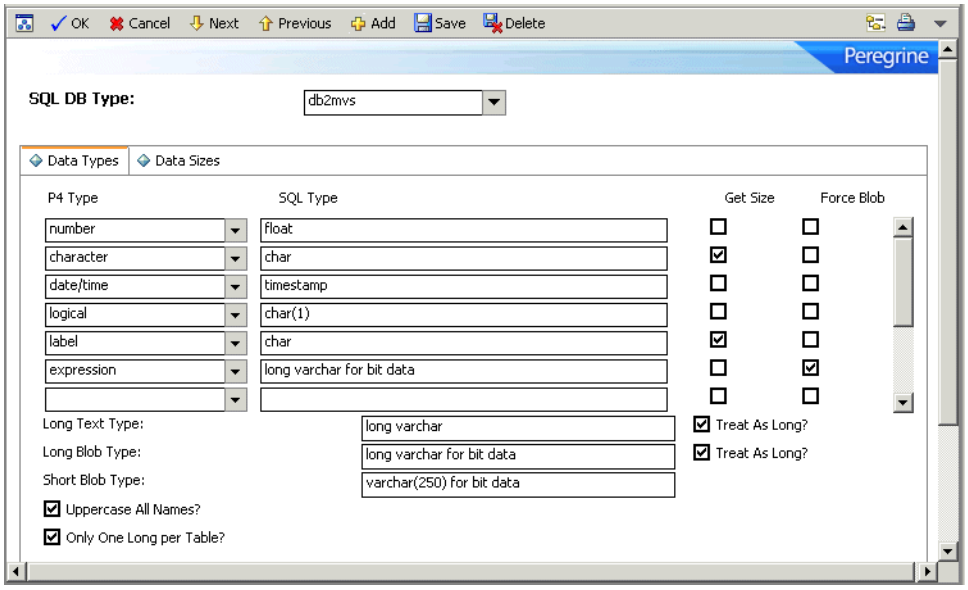


Figure 3-3: SQL DB Type

Note: The information on the SQL DB Type form varies by database type.

4 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.

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

The flags you select tell 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.

Mapping fields to a null table

Since users can customize ServiceCenter applications and files, some fields may be unused. To save storage space and reduce processing overhead, map unnecessary fields 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: If a field is not mapped at all, ServiceCenter assumes that it is mapped to a null table.

To map fields to nulltable

1. Do the preconversion server setup. See *Conversion process* on page 124 for details.
2. Open the **SQL Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

   The system displays the **SQL Utilities** menu (Figure 5-4 on page 126).

3. Click **Move Files to SQL from P4**. This option will not work if you have not done the preconversion server setup.

The system displays the P4 to SQL Conversion Console.

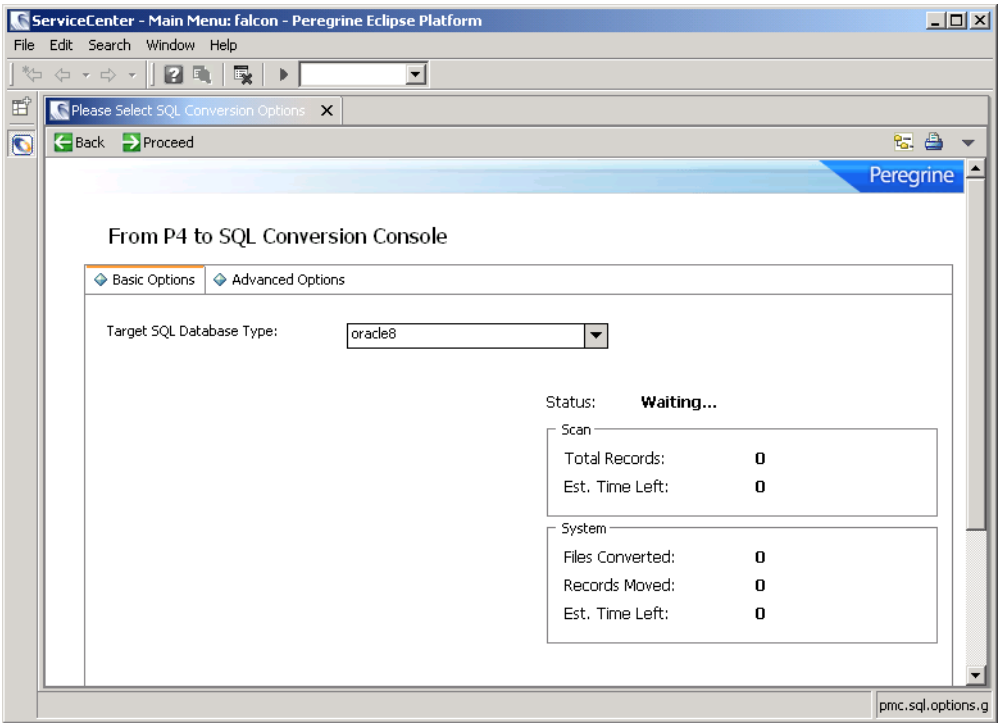


Figure 3-4: P4 to SQL Conversion Console

4 Select **Single File** from the pull-down options menu.

The form displays the **File to Convert** field, allowing you to name a file.

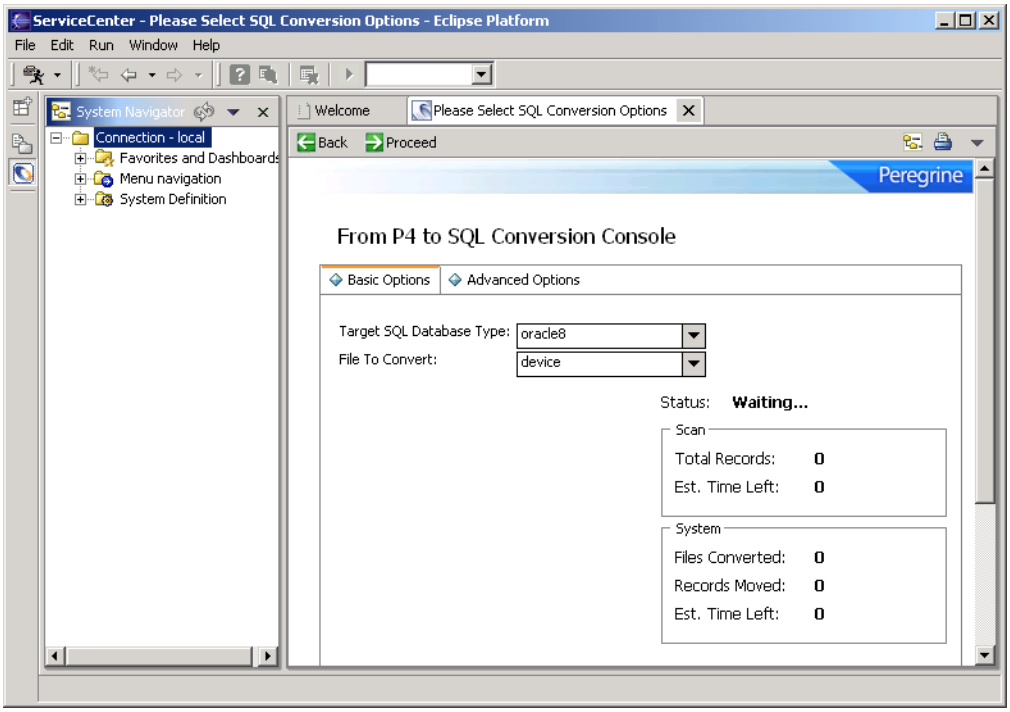


Figure 3-5: SQL Conversion Console for mapping a single file

5 Select the file you want to map from the drop-down list. For this example, we have chosen the **device** file.

6 Select your target RDBMS from the drop-down list in the **Target SQL Database Type** field. For this example, we have chosen **oracle8**.

7 Select the **Advanced Options** tab.

8 Select the following Final Objective options:

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

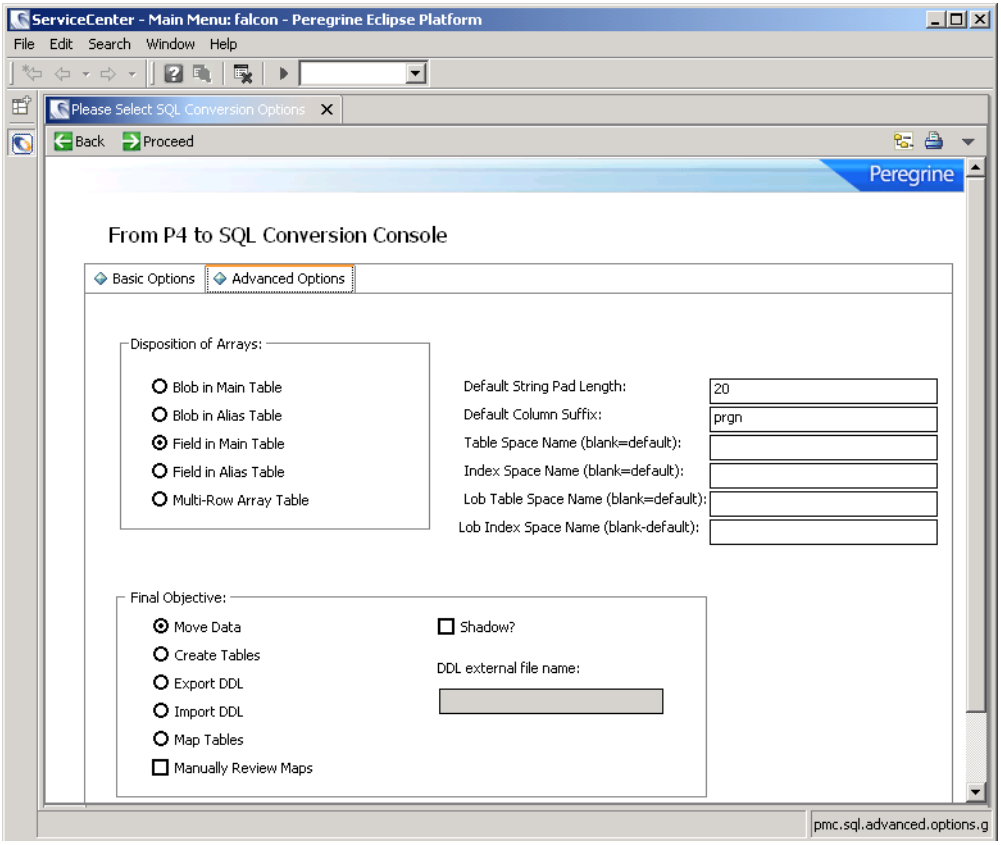


Figure 3-6: SQL mapping options

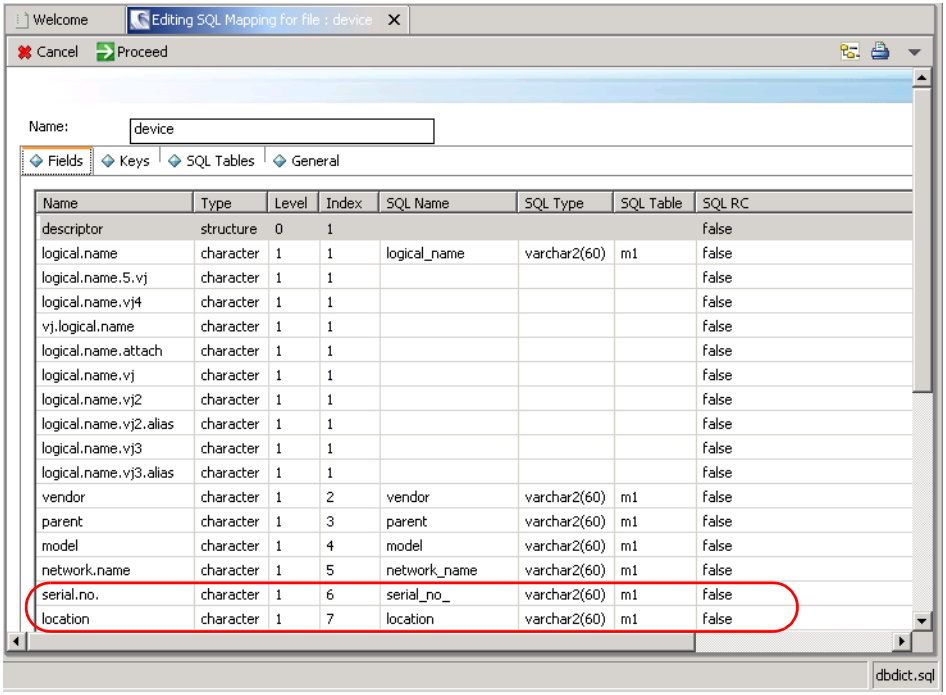
9 Click **Proceed**.

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

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

The system displays the desired map.

For this example, suppose that the **serial.no**, and **location** fields were unnecessary.



Name	Type	Level	Index	SQL Name	SQL Type	SQL Table	SQL RC
descriptor	structure	0	1				false
logical.name	character	1	1	logical_name	varchar2(60)	m1	false
logical.name.5.vj	character	1	1				false
logical.name.vj4	character	1	1				false
vj.logical.name	character	1	1				false
logical.name.attach	character	1	1				false
logical.name.vj	character	1	1				false
logical.name.vj2	character	1	1				false
logical.name.vj2.alias	character	1	1				false
logical.name.vj3	character	1	1				false
logical.name.vj3.alias	character	1	1				false
vendor	character	1	2	vendor	varchar2(60)	m1	false
parent	character	1	3	parent	varchar2(60)	m1	false
model	character	1	4	model	varchar2(60)	m1	false
network.name	character	1	5	network_name	varchar2(60)	m1	false
serial.no.	character	1	6	serial_no_	varchar2(60)	m1	false
location	character	1	7	location	varchar2(60)	m1	false

Figure 3-7: Fields mapped to an Oracle database

11 Select the **SQL Tables** tab.

12 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. Select any letter to define your nulltable.

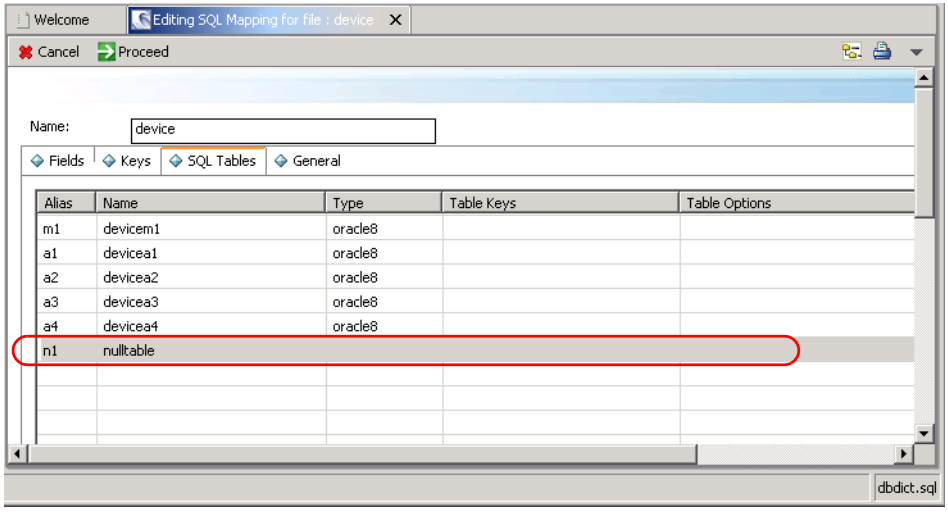
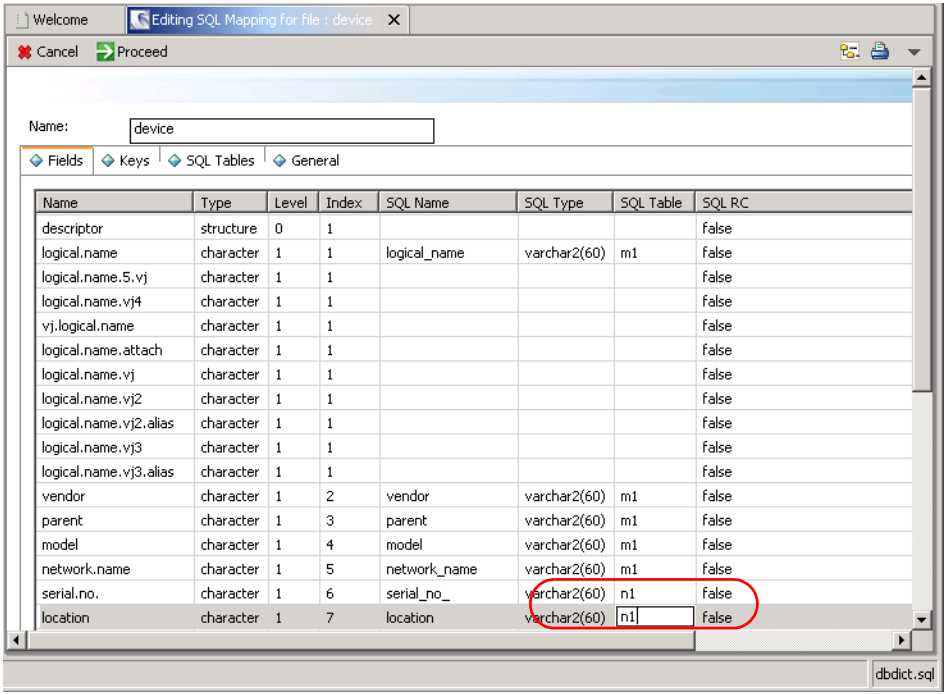


Figure 3-8: SQL Table definitions

13 Select the **Fields** tab.

14 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
descriptor	structure	0	1				false
logical.name	character	1	1	logical_name	varchar2(60)	m1	false
logical.name.5.vj	character	1	1				false
logical.name.vj4	character	1	1				false
vj.logical.name	character	1	1				false
logical.name.attach	character	1	1				false
logical.name.vj	character	1	1				false
logical.name.vj2	character	1	1				false
logical.name.vj2.alias	character	1	1				false
logical.name.vj3	character	1	1				false
logical.name.vj3.alias	character	1	1				false
vendor	character	1	2	vendor	varchar2(60)	m1	false
parent	character	1	3	parent	varchar2(60)	m1	false
model	character	1	4	model	varchar2(60)	m1	false
network.name	character	1	5	network_name	varchar2(60)	m1	false
serial.no.	character	1	6	serial_no_	varchar2(60)	n1	false
location	character	1	7	location	varchar2(60)	n1	false

Figure 3-9: Fields mapped to nulltable

15 Click **Proceed**.

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

DDL options

DB2Universal

Table DDL options

Use the following parameters 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 you use the Capture program to capture changes for this table from the log. If you do not specify a value, the RDBMS defaults to NONE.
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

Use the following parameters 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.

Microsoft SQL Server

Table DDL options

Use the following parameters 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 file group must exist within the database. f you do not specify a value, the RDBMS stores the table in the default file group. This parameter is available for sqlserver7 and higher.

Index DDL options

Use the following parameters 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

Use the following parameters 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. Allocate storage 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

Use the following parameters 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

Use the following parameters 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

Use the following parameters 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.

CHAPTER 4

Preconversion Configuration

This chapter explains the required preconversion configuration of Relational Database Management Systems (RDBMS). Its intended audience consists of system administrators and database administrators preparing to convert a ServiceCenter® system to an RDBMS.

Topics in this chapter include:

General space requirements

Unlike most commercial RDBMS vendors, the ServiceCenter P4 file system stores data in a compressed format. Consequently, the storage space required for data stored in a commercial RDBMS is between four and five times greater than that required in the P4 format.

If you are establishing a new ServiceCenter system, 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: Place all ServiceCenter data in a dedicated table space within a single instance of your RDBMS. This table space 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.

Login ID

Create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. This login must have database administrator authority in the target database.

Note: Database Administrator authority is only required during conversions, and then only if you allow ServiceCenter to issue the DDL to create tables and indices.

The login you provide must use the tablespace created earlier (see *Tablespaces* on page 64) as its default. When ServiceCenter logs in, it always creates its table in the default tablespace defined for that login ID.

Important: When configuring the login, select Permit and Default in the ServiceCenter database, and grant DBO to the user for that database.

Accessing unconverted files

If you use the Direct SQL Interface before accessing an SQL file, add the parameter sqldirect to the end of the sc.ini file to tell the server where to direct the SQL request. Use the value selected from the **Target SQL Database Type** drop-down list on the P4 to SQL Conversion Console for thesqldirect parameter. (See Figure 3-4 on page 74.) For example, sqldirect:oracle indicates that the system should use the Oracle interface.

Setting up time zones for RDBMS reporting

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

For information about using the sqltz parameter, see System Parameters in the ServiceCenter online help.

To change this parameter after conversion, convert all SQL converted tables back to P4 (see *Converting RDBMS Files Back to the P4 Format* on page 191), change the settings, and then reconvert them to your RDBMS. (See *Conversion to an RDBMS* on page 119.)

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.

Limiting the transaction log size

During the conversion, ServiceCenter places 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.

Enabling connectivity

Only ServiceCenter servers connect to the 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 in the ServiceCenter client's initialization files.

To set up the connection ServiceCenter's application server to an RDBMS, you will need:

- The name of the database.
- The login and password required to connect to the database server. Use to login and password created in *Login ID* on page 86.

The ServiceCenter configuration file is called sc.ini. It must be present in the ServiceCenter server RUN directory. Set ServiceCenter server parameters in the sc.ini file.

To enable connectivity

1. Add the following two lines to the sc.ini file in your ServiceCenter server's RUN directory:
   sqldb:<*dbname*>
   sqllogin:<login>/<*password*>
2. Cycle your ServiceCenter server.

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
```

Information for UNIX users

ServiceCenter supplies a different set of server applications for each commercial RDBMS. In order to connect to a specific RDBMS, configure the ServiceCenter server to use the appropriate the binaries. You do not need to change your client installations.

The ServiceCenter RUN directory contains multiple copies of the scenter file. These copies have names base on the database type, for example:

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

To enable a particular RDBMS, 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
```

Append the ServiceCenter RUN directory to the previous value of the library path statement. If no library path statement exists, create one pointing to the ServiceCenter RUN directory.

- In AIX the environment variable is: LIBPATH
- In Linux and Solaris the environment variable is: LD_LIBRARY_PATH
- In HPUX the environment variable is: SH_LIB_PATH

Information for Windows users

ServiceCenter supplies one of each of the following executable files to support all third party relational databases and P4:

- scautod
- scenter

A DLL file, included with the ServiceCenter installation, provides the ability to connect with specific RDBMS vendors. Database DLLs are dynamically loaded at the time of conversion, based on the RDBMS specified. You do not need to change your client installations.

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

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.

Complete the following procedures prior to converting your database to Sybase.

Step 1 Make sure that ServiceCenter is fully installed. Refer to the ServiceCenter installation guide for more information.

Step 2 Review information specific to your operating system and perform and necessary procedures. See one of the following:

- *Information for UNIX users* on page 89*Information for Windows users* on page 89

Step 3 Allocate data space large enough to hold your data. See *General space requirements* on page 86.

Step 4 Allocate enough additional server connections for all your users. See *Server connections* on page 86.

Step 5 Create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. See *Login ID* on page 86.

Step 6 If you plan to report on ServiceCenter data using RDBMS tools, set up time zones. See *Setting up time zones for RDBMS reporting* on page 87.

Step 7 If you use the Direct SQL Interface before accessing an SQL file, tell the server where to direct the SQL request. See *Accessing unconverted files* on page 87.

Step 8 Set up the general conversion parameters. See *Setting up ServiceCenter's general conversion parameters* on page 91.

Step 9 Modify the DDL that ServiceCenter uses to create tables in DB2. See *Modifying the DDL that ServiceCenter uses to create tables in DB2* on page 92

Step 10 Tune server data for conversion. See *Tuning DB2 for Data Conversion* on page 98.

Step 11 Limit the transaction log size. See *Limiting the transaction log size* on page 88.

Step 12 Change the default Sybase auto identity setting to false. See *Setting the Sybase auto identity option* on page 116.

Step 13 Add output as an SQL reserved word in ServiceCenter. See *Adding output as a reserved word in ServiceCenter* on page 117.

Step 14 Log in to ServiceCenter with just the listener. See *Conversion process* on page 124 for more information.

Note: A fully qualified Sybase administrator should assist with these preparations.

Setting up ServiceCenter's general conversion parameters

To set up the conversion parameters

1 Open the **SQL Utilities Menu** (Figure 5-4 on page 126). For information on how to open the **SQL Utilities Menu**, see *To open the SQL Utilities menu and begin conversion* on page 125.

2 Click **SQL Database Information** to open the sql.db.infoform name.

3 Select the db2universal database type.

4 On the Data Types tab, 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 Data Sizes tab, 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, set the DB2 tablespace for ServiceCenter 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.

6 Change the Maximum Columns per Row to 1012.

Modifying the DDL that ServiceCenter uses to create tables in DB2

Modifying the DDL that ServiceCenter uses to create the tables in DB2 is considerably more involved and varies from implementation to implementation. Out-of-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, because it performs a direct I/O operation to write and retrieve values, rather than using buffer pools and caches. To address this performance issue, modify the ServiceCenter DDL to change these LONG VARCHAR fields into the data type VARCHAR. The tables you must modify vary based on the implementation.

Even if you plan a full conversion to DB2, you may not need to changes every LONG VARCHAR in every table. If the system accesses a table only occasionally, the amount of work needed to change the LONG VARCHAR may far out weigh the performance gain. Also, the amount of space allocated to each of these VARCHAR fields varies considerably based on the needs of the 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]                --P4[company.sla; 117; M1; 0]
 OPEN_TIME TIMESTAMP,                    COMPANY_SLA CHAR(30),
  --P4[opened.by; 4; M1; 0]                --P4[subcategory; 118; M1; 0]
 OPENED_BY CHAR(50),                     SUBCATEGORY CHAR(140),
  --P4[priority.code; 5; M1; 0]            --P4[hot.tic; 119; M1; 0]
 PRIORITY_CODE CHAR(50),                 HOT_TIC CHAR(1),
```

```
  --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[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[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[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[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]
ALERT2 CHAR(1),
  --P4[alert3; 51; M1; 0]
ALERT3 CHAR(1),
  --P4[deadline; 52; M1; 0]
DEADLINE CHAR(1),
  --P4[reassigned; 53; M1; 0]
  --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]
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]
```

```
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]
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]
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]
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]
```

```
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),
  --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),
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[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[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[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.

Tuning DB2 for Data Conversion

Before beginning moving data to DB2, tune DB2 for faster data conversion performance.

For this discussion, we will assume the following:

- The database server machine has 500MB of free memory available
- A 32K SMS table space is being used to contain the ServiceCenter tables.

Enable the database for multi-page file allocation

Enabling multi-page file allocation causes DB2 to allocate new data pages in a table space one extent at a time rather than one page at a time, which reduces overhead of large insert operations.

Note: Enable multi-page file allocation on SMS table spaces only.

To enable multi-page file allocation

▶ Disconnect all applications from the database and, as the instance owner (for example, db2inst1) and execute the following:

```
db2empfa <dbname>
```

Tune the buffer pools

The buffer pool that requires most adjustment is the one that will host ServiceCenter tables.

To understand/verify how your tables are distributed among your table spaces

▶ Execute the following query:

```
SELECT t.tbspace, count(*)
  FROM syscat.tablespaces ts, syscat.tables t
  WHERE ts.tbspaceid=t.tbspaceid and t.type='T'
  GROUP BY t.tbspace
```

The output may look like this (numbers here are fictional)

```
TBSPACE            2
------------------ -----------
SYSCATSPACE                 71
TS32K                     1034
USERSPACE1                   0
```

Size the buffer pools are sized to reflect the way the tables are distributed among table spaces. In this case, most of the tables are in the TS32K table space (as recommended and expected), and so we should increase the corresponding 32K buffer pool.

To view current buffer pool settings

- Execute the following query:
  SELECT bpname, npages, pagesize FROM syscat.bufferpools
  The output should look something like this:

```
BPNAME              NPAGES      PAGESIZE
------------------- ----------- -----------
IBMDEFAULTBP               1000        4096
BP32K                        -1       32768
```

From this output, the NPAGES column shows that the size of the buffer pool is -1, the default value, which is retrieved from the database configuration file and is typically too small for production use. Change this to be an explicit value. Take into account the amount of available free physical memory on your machine. In this example, 500MB of free physical memory is available. A safe starting point is to allocate 60% of this memory to the 32K Buffer pool. Here is an example of how to do the calculation:

500MB free physical memory * 60% = 300MB to be allocated to buffer pools

300MB = 300,000K

300,000K / 32K = 9375 pages for 32K buffer pool

To change the buffer pool size

Use
CONNECT TO <dbname> USER <userid> USING <password>
ALTER BUFFERPOOL BP32K SIZE 9375
TERMINATE

All applications must disconnect from the database for the new buffer pool size to take effect.

Warning: If you are a 32-bit DB2 and a server with lots of memory (> 2G), do not allocate more than 1700MB of real memory to buffer pools. That is, the total amount of memory used by all buffer pools combined in all databases within the same instance should not exceed 1.7G. Otherwise, the system may begin to swap excessively when DB2 is running (real limits depend on the OS being used, but 1.7G is a safe number for all platforms).

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 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.

Complete the following procedures prior to converting your database to Microsoft SQL server.

Step 1 Make sure that ServiceCenter is fully installed. Refer to the ServiceCenter installation guide for more information.

Step 2 Review information specific to your operating system and perform and necessary procedures. See *Information for Windows users* on page 89.

Step 3 Allocate data space large enough to hold your data. See *General space requirements* on page 86.

Step 4 Allocate enough additional server connections for all your users. See *Server connections* on page 86.

Step 5 Create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. See *Login ID* on page 86.

Step 6 If you plan to report on ServiceCenter data using RDBMS tools, set up time zones. See *Setting up time zones for RDBMS reporting* on page 87.

Step 7 If you use the Direct SQL Interface before accessing an SQL file, tell the server where to direct the SQL request. See *Accessing unconverted files* on page 87.

Step 8 Set up the ServiceCenter connectivity to the Microsoft SQL server instance in the ServiceCenter sc.ini file. See *Microsoft SQL Server connectivity* on page 102.

Step 9 Limit the transaction log size. See *Limiting the transaction log size* on page 88.

Step 10 Set up Timestamps See *Timestamps in ServiceCenter* on page 102.

Step 11 Set up the case sensitivity. See *Case sensitivity* on page 103.

Step 12 Set up the general conversion parameters.

Step 13 Modify the DDL that ServiceCenter uses to create tables in Microsoft SQL server.

Step 14 Tune server data for conversion.

Step 15 Log in to ServiceCenter with just the listener. See *Conversion process* on page 124 for more information.

Note: A fully qualified Microsoft SQL server administrator should assist with these preparations.

Microsoft SQL Server connectivity

The general connectivity rules are:

- The database name entered in your *sc.ini* file (above) must correspond to an ODBC data source.
- Configure the ODBC data source as a System DSN. Set it up to use:
  - SQL server authentication
  - ANSI quoted identifiers
  - ANSI nulls, paddings, and warnings

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.

Case sensitivity

ServiceCenter does not require that 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, you cannot use a default installation of Microsoft SQL server 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 rebuild the rebuilding the Microsoft SQL server master database 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.

Setting up case sensitivity

This section describes the general steps necessary for converting ServiceCenter from the P4 file system to a Microsoft SQL Server with either case-sensitive or case-insensitive searching.

To set up Microsoft SQL server and P4 for case-insensitive searching

1. Convert P4 so that it will do case-insensitive searching. (For instructions, refer to *Case mode* in the ServiceCenter online help.)
2. Confirm that your installation of MS SQL is case insensitive (default installation). Use the SQL server 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.
3. Change MS SQL to case insensitive searching if necessary. Check the MS SQL documentation for instructions on rebuilding the master database with the proper sort order and case-sensitivity.

To set up Microsoft SQL server and P4 for case-sensitive searching

1. Convert P4 so that it will do case-sensitive searching. This step is not necessary unless you have previously converted P4 to be case-insensitive. (For instructions, refer to *Case mode* in the ServiceCenter online help.)
2. Confirm that your installation of MS SQL is case-sensitive (set at installation). Use the SQL server 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.
3. Change MS SQL to case-sensitive searching if necessary. Check the MS SQL documentation for instructions on rebuilding the master database with the proper sort order and case-sensitivity.

Oracle server preparation

Complete the following procedures prior to converting your database to the RDBMS.

Step 1 Make sure that ServiceCenter is fully installed. Refer to the ServiceCenter installation guide for more information.

Step 2 Review information specific to your operating system and perform and necessary procedures. See one of the following: See one of the following:

- *Information for UNIX users* on page 89*Information for Windows users* on page 89

Step 3 Allocate data space large enough to hold your data. See *General space requirements* on page 86.

Step 4 Allocate enough additional server connections for all your users. See *Server connections* on page 86.

Step 5 Create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. See *Login ID* on page 86.

Step 6 If you plan to report on ServiceCenter data using RDBMS tools, set up time zones. See *Setting up time zones for RDBMS reporting* on page 87.

Step 7 If you use the Direct SQL Interface before accessing an SQL file, tell the server where to direct the SQL request. See *Accessing unconverted files* on page 87.

Step 8 Make sure that your tables are set up properly. See *Setting up Oracle tables* on page 105.

Step 9 Set up ServiceCenter connectivity to the RDBMS instance in the ServiceCenter sc.ini file. See *Enabling connectivity* on page 88 and *Set up Oracle connectivity* on page 106.

Step 10 Limit the transaction log size. See *Limiting the transaction log size* on page 88.

Step 11 Define a table space. See *Define a default tablespace* on page 107.

Step 12 Set the RDBMS environmental variables. See *Set the Oracle environmental variables* on page 108.

Step 13 Set up the Oracle Call Interface. See *Employing multiple databases with Oracle Call Interface* on page 109.

Step 14 Set up the general conversion parameters.

Step 15 Modify the DDL that ServiceCenter uses to create tables in DB2.

Step 16 Tune server data for conversion.

Step 17 Log in to ServiceCenter with just the listener. See *Conversion process* on page 124 for more information.

Note: A fully qualified RDBMS administrator should assist with these preparations.

Setting up Oracle tables

The majority of tables on an Oracle server hold less than 50K of data. Peregrine 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 thereof.
- 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.

Set the Oracle environment variable

The files scenter.oracle and scenter.oraoci are built with Oracle shared lib, which requires an Oracle installation in order to start scenter. After installing an Oracle, reset your Oracle environment variable.

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

Set up Oracle connectivity

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 using SQL*Net.

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. On UNIX platforms, the **tnsnames.ora** file is located using the TNS_ADMIN environmental variable. (See *UNIX platforms* on page 108.) On Windows platforms, the **tnsname.ora** file is located in the Oracle Home directory. (See *Windows platforms* on page 109.)
sqllogin	The **sqllogin** parameter specifies the Oracle user id and password used to connect to the database.

Sample connections

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 (Refer to the *Oracle SQL*Net* documentation).

Note: Specify only **scora** in the ServiceCenter *sc.ini* file to connect to **scora.world**. The **world** domain is defined in the Oracle **sqlnet.ora** configuration file.

A ServiceCenter sc.ini file with sqllogin and sqldb parameters

This entry in the **sc.ini** or **inferfaces** file connects ServiceCenter to the Oracle instance named *scora*.

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

Define a default tablespace

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

A 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
```

Set the Oracle environmental variables

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.

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 ORACLE_HOME in the registry folder "HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE".

Employing multiple databases with Oracle Call Interface

The ServiceCenter Oracle Call Interface (OCI) features allow you to split data and manage it between multiple databases. This configuration enables groups that own and maintain data on different databases to share and access the data as if it were on the same database. Store one group of files on a database1, and other groups of files on database2, database3, and so on.

You can map a single instance of ServiceCenter to different RDBMS tables. For example you may want to have your Human Resource 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.

You can have:

- 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.)
- 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.

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 you can accomplish this, create multiple new database types in ServiceCenter (one for each additional database), and associate these types 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 User ID *prgn1* password *prgn1*) and datab2 (with connection parameters of User ID *prgn2* password *prgn2*).

Two database types are created using the ServiceCenter SQL utility for datab1 and datab2, for example, 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
```

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.

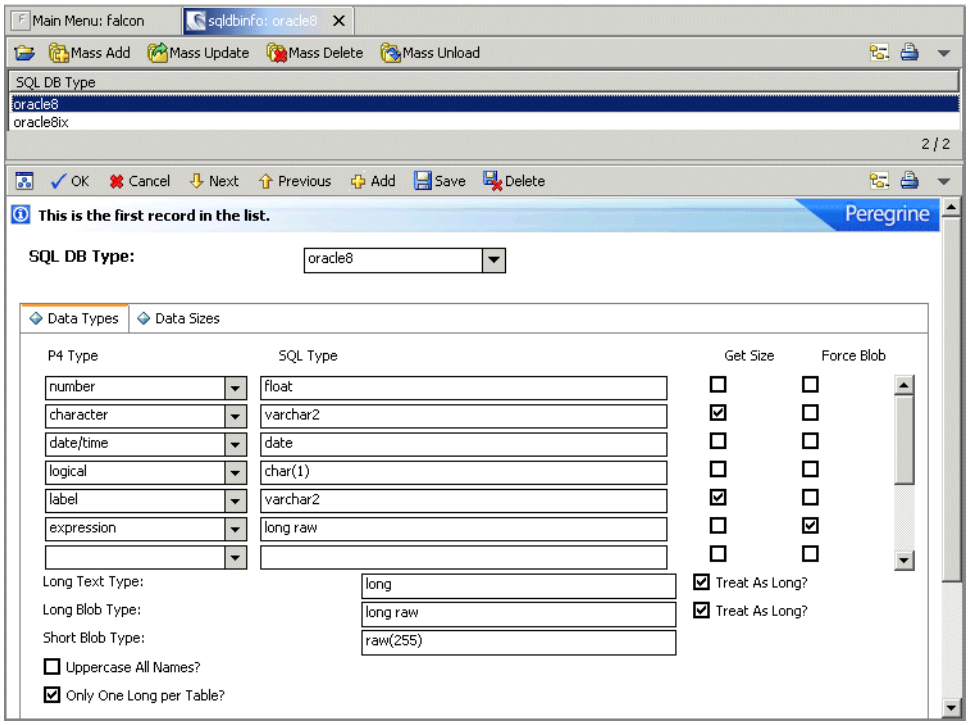


Figure 4-1: SQL database information form

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 Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The system displays the **SQL Utilities** menu (Figure 5-4 on page 126).

2 Click **SQL DB Types Utility**.

3 At the prompt screen, type 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.

This example uses *oracle8*.

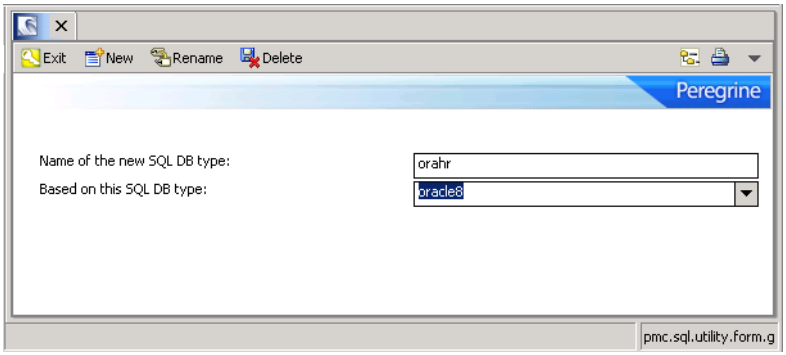


Figure 4-2: Creating a new database type

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

The system displays a prompt, 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 130.

Sybase server preparation

Complete the following procedures prior to converting your database to Sybase.

Step 1 Make sure that ServiceCenter is fully installed. Refer to the ServiceCenter installation guide for more information.

Step 2 Review information specific to your operating system and perform and necessary procedures. See one of the following:

- *Information for UNIX users* on page 89*Information for Windows users* on page 89

Step 3 Allocate data space large enough to hold your data. See *General space requirements* on page 86.

Step 4 Allocate enough additional server connections for all your users. See *Server connections* on page 86.

Step 5 Create a login ID and password for ServiceCenter to use when it connects to your RDBMS server. See *Login ID* on page 86.

Step 6 If you plan to report on ServiceCenter data using RDBMS tools, set up time zones. See *Setting up time zones for RDBMS reporting* on page 87.

Step 7 If you use the Direct SQL Interface before accessing an SQL file, tell the server where to direct the SQL request. See *Accessing unconverted files* on page 87.

Step 8 Set up the ServiceCenter connectivity to the Sybase instance in the ServiceCenter sc.ini file. See *Sybase connectivity* on page 115.

Step 9 Limit the transaction log size. See *Limiting the transaction log size* on page 88.

Step 10 Change the default Sybase auto identity setting to false. See *Setting the Sybase auto identity option* on page 116.

Step 11 Add output as an SQL reserved word in ServiceCenter. See *Adding output as a reserved word in ServiceCenter* on page 117.

Step 12 Set up the general conversion parameters.

Step 13 Modify the DDL that ServiceCenter uses to create tables in Sybase.

Step 14 Tune server data for conversion.

Step 15 Log in to ServiceCenter with just the listener. See *Conversion process* on page 124 for more information.

Note: A fully qualified Sybase administrator should assist with these preparations.

Sybase connectivity

ServiceCenter uses native Sybase libraries to connect to Sybase. For Windows, these libraries require the presence of a file to establish a relationship between a logical database server name (for example, **scsyb**) and a physical IP address and port number. This file is named **sql.ini** in Windows and **interfaces in UNIX**. The **sql.ini** or **Interfaces** file must be present on the application server and must be in the path of the UID that will launch the ServiceCenter application server.

Parameter	Definition
sqldb	The **sqldb** parameter in the ServiceCenter **sc.ini** file specifies the name of a Sybase SQL server connection. The connection name is defined in the **sql.ini** file on Windows Platforms, and in the **interfaces** file on UNIX platforms. This file is not an element of ServiceCenter. The Windows **sql.ini** file must be in the path of the UID that will launch the ServiceCenter application server. (See *A Sybase interfaces or sc.ini file entry specifying the scsyb server description* on page 116.) The UNIX **interfaces** file is located using the SYBASE environmental variable. (See *A Sybase interfaces or sc.ini file entry specifying the scsyb server description* on page 116.) For detailed information about the **sql.ini** or **interfaces** file refer to 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.

Sample connections

An sc.ini file with sqllogin and sqldb parameters

This entry in the **sc.ini** or **inferfaces** file connects ServiceCenter to the Sybase instance named *scsyb*.

```
# database connects to scsyb
sqldb:scsyb
sqllogin:ed/ed
system:7111
auth:xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
```

A Sybase interfaces or sc.ini 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 is:

```
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.

Setting the Sybase auto identity option

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 -sys600 -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.

CHAPTER 5

Conversion to an RDBMS

This chapter explains the process of converting the P4 file system to a Relational Database Management System (RDBMS). Its intended audience consists of system administrators and database administrators converting a ServiceCenter® system to an RDBMS.

Topics in this chapter include:

Pre-Conversion server preparation

During file conversion, the administrator performing the conversion must be the only user on the system, and no background processes can be running.

To prepare the ServiceCenter server for conversion:

- *Ensure that no background processes are running* on page 120
- *Establish a connection with the server and log in* on page 122

Ensure that no background processes are running

To ensure that no background processes are running

1. Shut down the Server. See *To shut down the server* on page 120.
2. Restart the Server with no background processes running. See one of the following:
   - *To restart the server in at the Windows command prompt* on page 121
   - *To restart the server in UNIX* on page 121

To shut down the server

1. Choose an off-hour or non-peak time to perform the file conversion.
2. Click **Status** on the **Top** of the Main Menu.

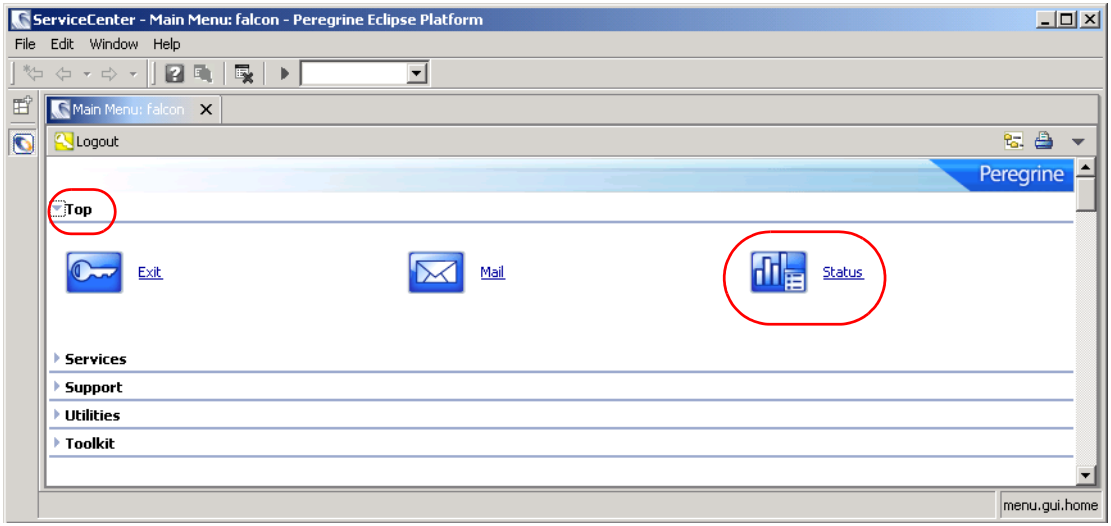


Figure 5-1: Main Menu — Top

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.

To restart the server in at the Windows command prompt

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, using the following command:

```
start /B scenter -listener:<xxxx>
```

Where *<xxxx>* is the port number used by the Listener. Specify a port number that is not otherwise used by another process on the computer or by regular ServiceCenter users. The recommended port is 12681. (for example, start scenter -listener:12681)

To restart the server in Windows after editing the configuration file

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

2 Restart the ServiceCenter server from the Start menu or desktop icon.

3 Uncomment the lines and restart the server again when finished.

The server starts, suppressing all background processes.

To restart the server in UNIX

1 Change to the ServiceCenter/RUN directory.

2 Restart ServiceCenter using the following command:

```
scenter -listener:<xxxx> &
```

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

The recommended port is 12681. (for example, scenter -listener:12681

The server starts a Listener at the specified port.

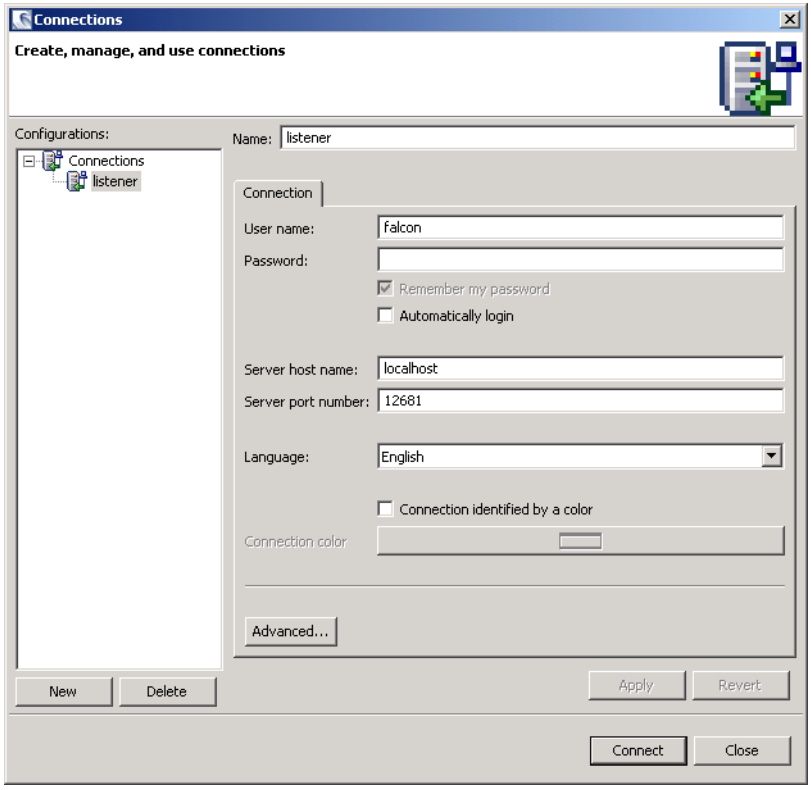
Note: If you do not use nosystemstartup, you will need to open the Status window and manually kill all extraneous processes, leaving only the Listener, and falcon active.

The server starts, suppressing all background processes.

Establish a connection with the server and log in

To establish a connection and log in

1. Start the ServiceCenter Client.
2. Select **Connect** > **Connections** from the File menu.
3. In the **Name:** text box, type a name.
   In this example, we use <*listener*>
4. In the **User Name:** text box, type the login of a user with *system administrator* rights.
   In this example, we use <*falcon*>.
5. In the **Password:** text box, type the password of that user.
6. In the **Server Name:** text box, type the name of the server.
7. In the **Server Port Number:** text box, type the server port number you specified when setting up your server to start with the Listener only.
   In this example, we use the recommended port, <*12681*>.



8 Click **Apply** to save the new connection specifications.

9 Click **Connect** to start the connection.

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

10 Click **Status** on the **Top** of the Main Menu.

The system displays a list of processes.

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

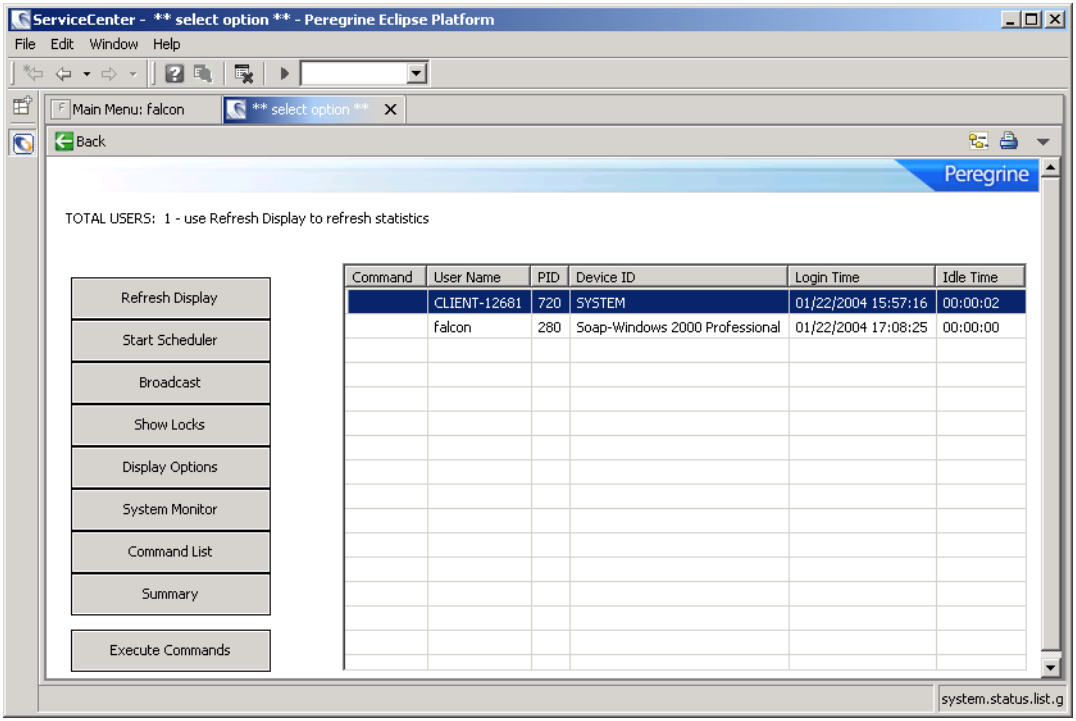


Figure 5-2: ServiceCenter Status window

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 *Conversion process* on page 124.) 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.

Note: In DB2, the actual dataset creations and insertions will use large quantities of logs. This amount will vary depending upon your DB2 configuration.

Important: Run the DB2 conversion when the DB2 subsystem is not busy processing BINDs. Determine the most convenient time to run a process that will create 2500 objects.

To open the SQL Utilities menu and begin conversion

1 Select **Utilities** in the Main Menu.

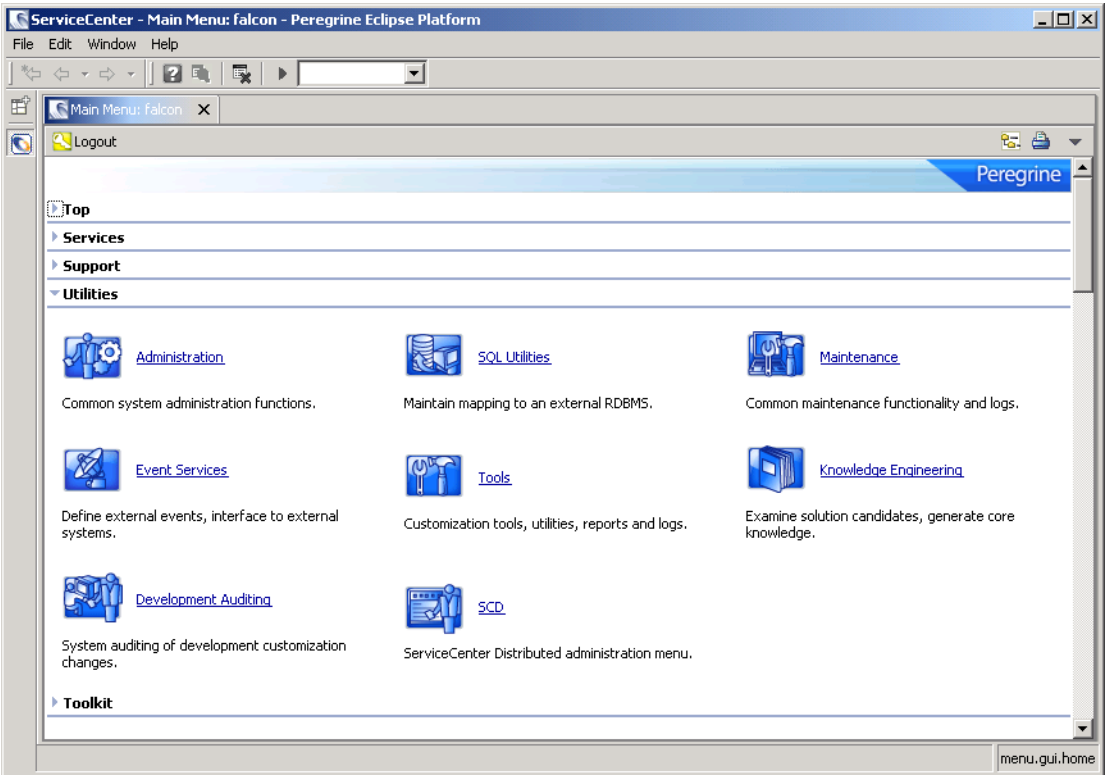


Figure 5-3: Main Menu — Utilities

2 Click **SQL Utilities**.

The system displays the **SQL Utilities** menu.

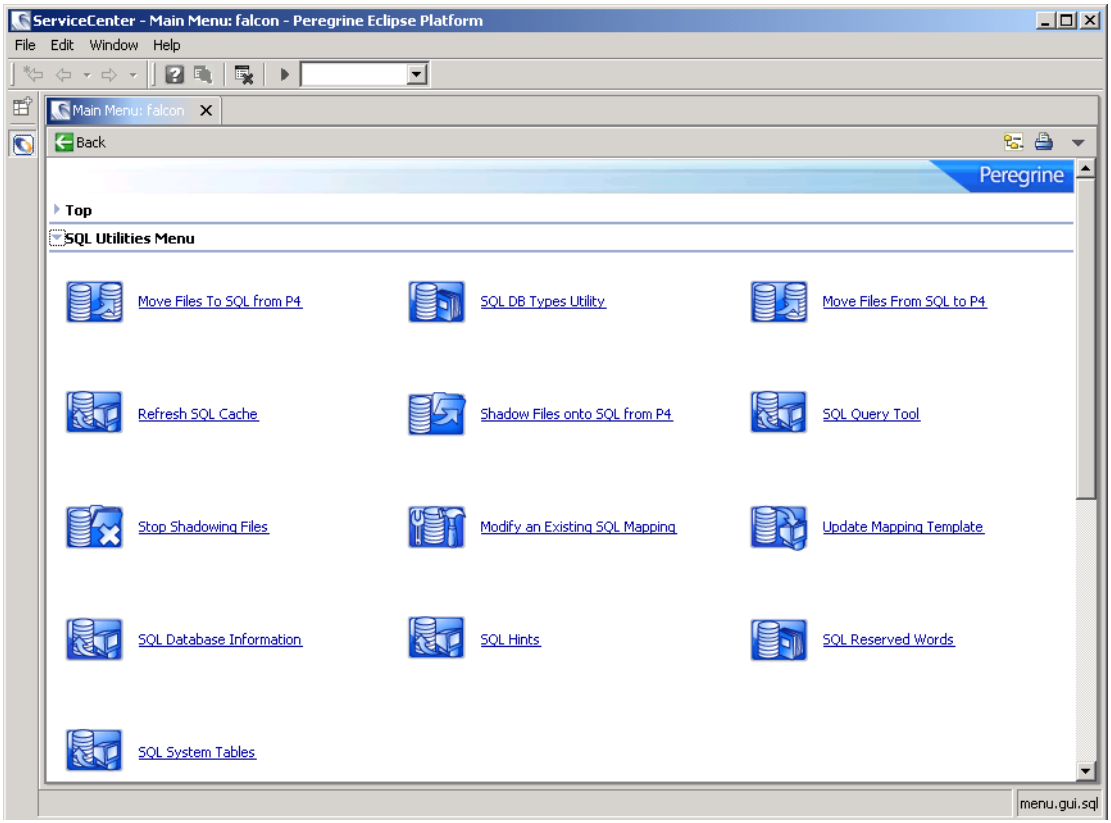


Figure 5-4: The SQL Utilities menu

3 Click **Move Files to SQL from P4**. This option will not work if you have not done the preconversion server setup. See *Conversion process* on page 124 for instructions.

The system displays the P4 to SQL Conversion Console window. (Figure 5-5 on page 127)

4 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 127.
- To perform a single file conversion or DDL manipulation, follow the steps in *Single file conversion* on page 130.

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 130.

To perform a multiple file conversion

1. Open the SQL Utilities Menu and begin conversion. See *To open the SQL Utilities menu and begin conversion* on page 125.
2. Select a Target SQL Database Type (for example, Sybase or 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 64.

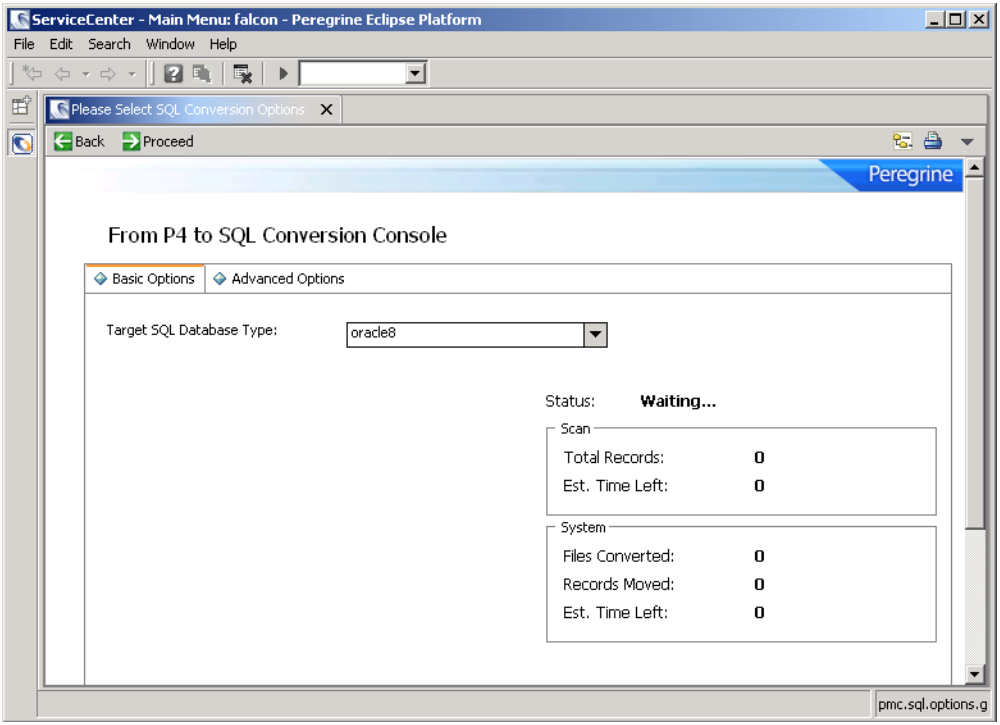


Figure 5-5: Selecting a destination database

The value you select for the **Target SQL Database Type** must match the following parameters in the sc.ini file:

[<target SQL database>]
sqldirect:<target SQL database>
sqldictionary:<target SQL database>

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

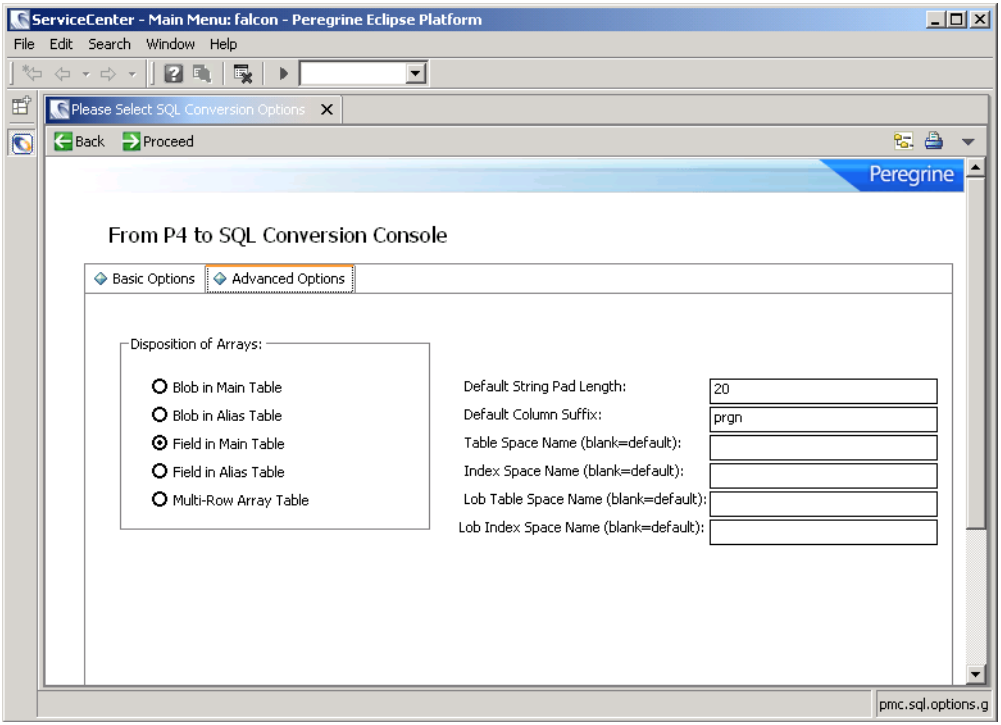


Figure 5-6: Advanced conversion options

Important: Contact your database administrator before choosing any options *other* than the default options.

4 Select one of the options from the **Disposition of Arrays** structure. The default is **Field in Main Table**. (See Table 5-1 on page 129 for a listing of the fields and their definitions.)

5 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.

6 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.

7 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 responsible for the conversion, to the desired tablespace. Do this on the RDBMS side before the conversion takes place.

8 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.

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 file will be stored in a new alias table in SQL.
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".

Table 5-1: Disposition of Arrays panel

Field	Definition
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 file 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 file 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. **Warning:** This option should never be used as default when converting more than one file.

9 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.

10 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 *Information for Windows users* on page 89 for more information.

11 Monitor the **Basic Options** tab for a dynamically updated status of the conversion. Other information is included in the **sc.log** file. See *Monitoring the conversion process* on page 139.

Single file conversion

If you have more than one database, you can use the single file conversion to map individual files to a particular database. Single file conversion 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 Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Click **Move Files to SQL from P4**. This option will not work if you have not done the preconversion server setup. See *Conversion process* on page 124 for instructions.

The pmc.sqloptions.g form opens.

3 Select **Single File** from the ServiceCenter pull-down options menu. Control options appear on the **Basic Options** tab.

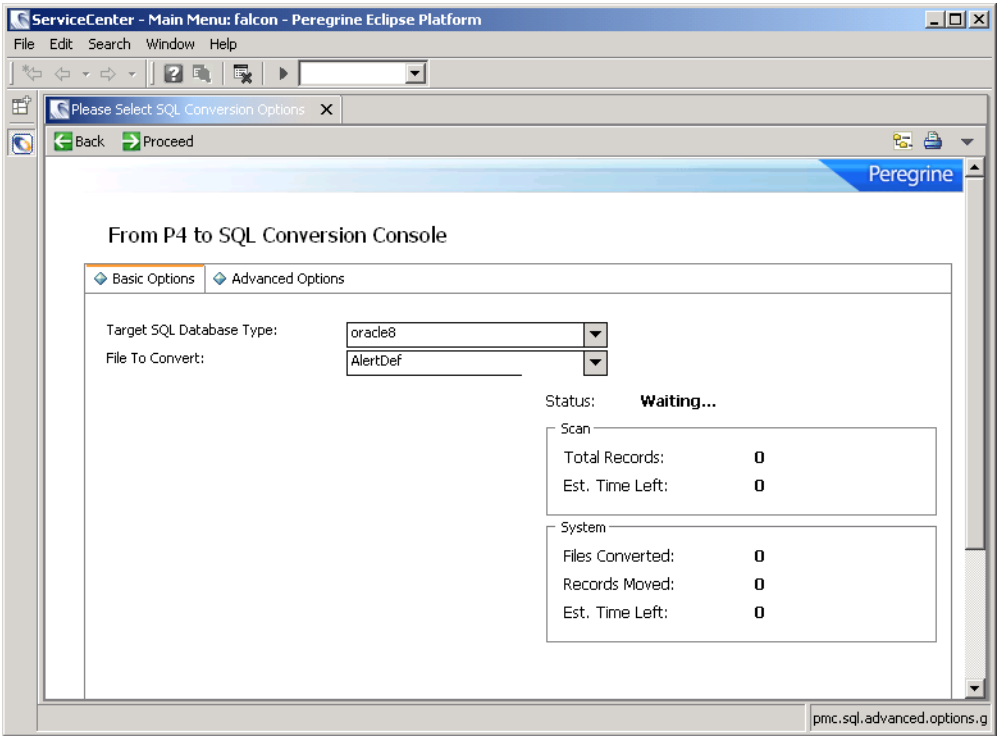


Figure 5-7: Selecting a destination database and local file for conversion

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 64.

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.

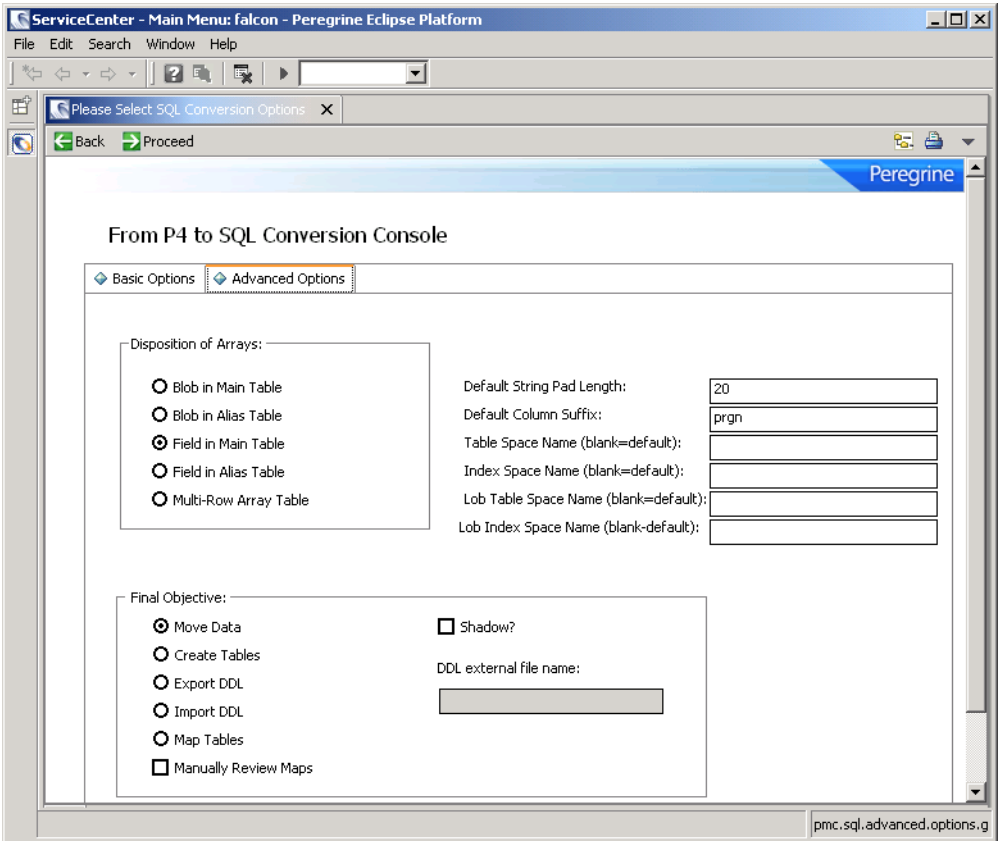


Figure 5-8: Advanced conversion options

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 129.

7 Make the appropriate selection from the Final Objective structure.

- **Move Data** — the default, used if you want 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.

  Click **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 183 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 134 for more information.

- **Import DDL** — allows a local DDL file to be read back into ServiceCenter internal data. See *Importing DDL* on page 134 for more information.

8 Set the **Manually Review Maps?** 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 opens 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 Note: on page 139)

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: 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, for example, `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.

Within the index file, the you may change the **/*Iconstraint*/** phrase to an appropriate index constraint (For example, for DB2, SUBPAGES, BUFFERPOOL, and so on), 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, for example, EDITPROC, VALIDPROC, and so on.)

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 the SQL mapping for a file

To edit the SQL mapping for a file

1 Open the **SQL Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Click **Modify and Existing SQL Mapping**.

3 Select the file you want to edit.

4 Click Proceed.

The screen displays each field, the table name, SQL Name and SQL Type.

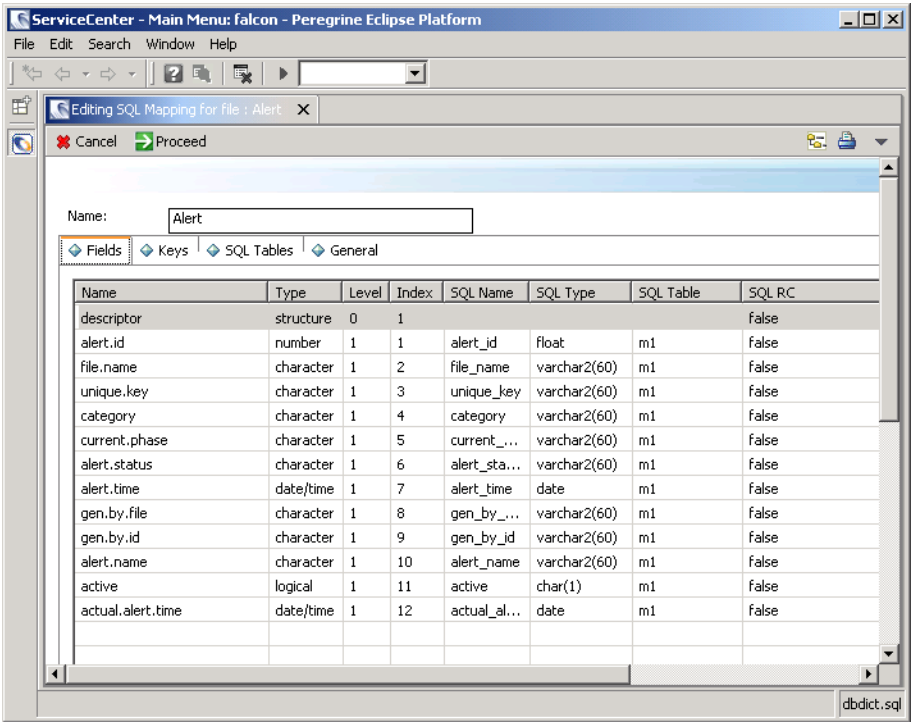


Figure 5-9: Editing the SQL mapping for a file

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 database administrator with counts of total fields from each type/length.

The information might look like this:

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)

With this information, you can determine average row length. For each field in each index, make note of the length/type in the RDBMS. Use it to determine the length of fields in indexes.

Displaying tables

Click the **SQL Tables** tab to display the tables that 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, when calculating initial/next extents for the tables consider how many array tables are present. Sacrificing for optimum sizes on main table so less space is wasted on array tables.

Do this process for all tables that are going into a single tablespace. By combining all the totals, you can create the tablespace for the tables and indexes. Also, you can provide the initial/next information for the table to be converted.

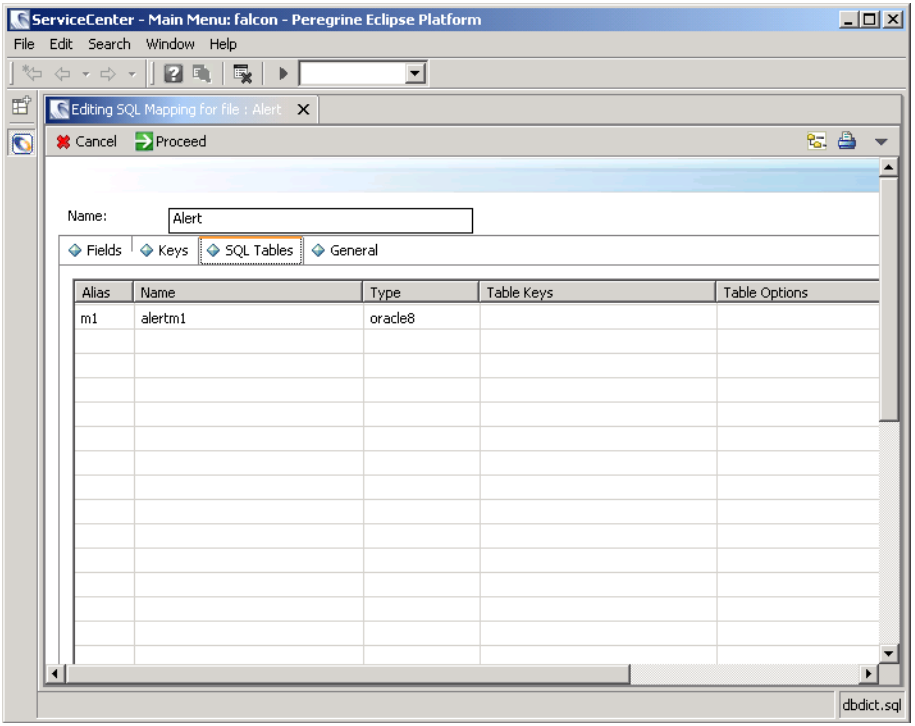


Figure 5-10: Table creation displays

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 Listener on ServiceCenter.
4. Log in 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, for example: amAsset. For instructions, refer to *Database Dictionary* in the ServiceCenter online help.

   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 Select **Single File** from the pull-down options menu.

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**.

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

   EXAMPLE: Change amAssetm1 to amAsset

12 Click **Proceed**. A pop-up will tell you the mapping is completed.

13 From Database Manager, open 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**.

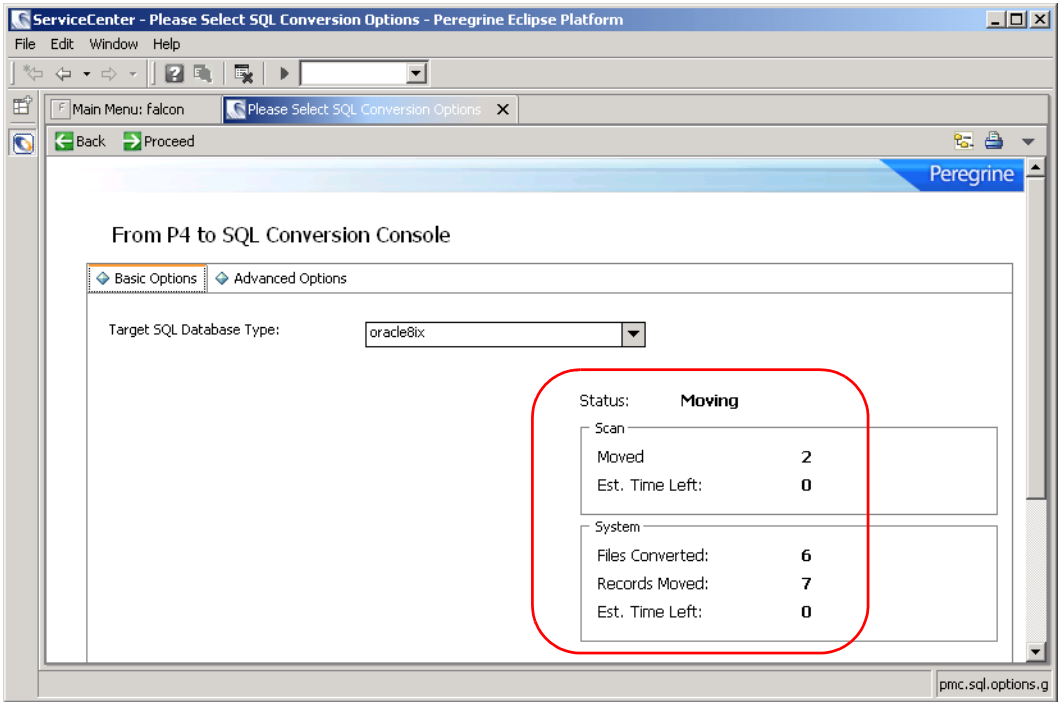
17 It takes awhile, but the dbdict will be updated.

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 using ServiceCenter.

Monitoring the conversion process

As the system converts your files, the status is dynamically updated on the **Basic Options** tab. The tab displays the current step in the conversion, the status of each file as it enters the conversion process, and the overall system status.



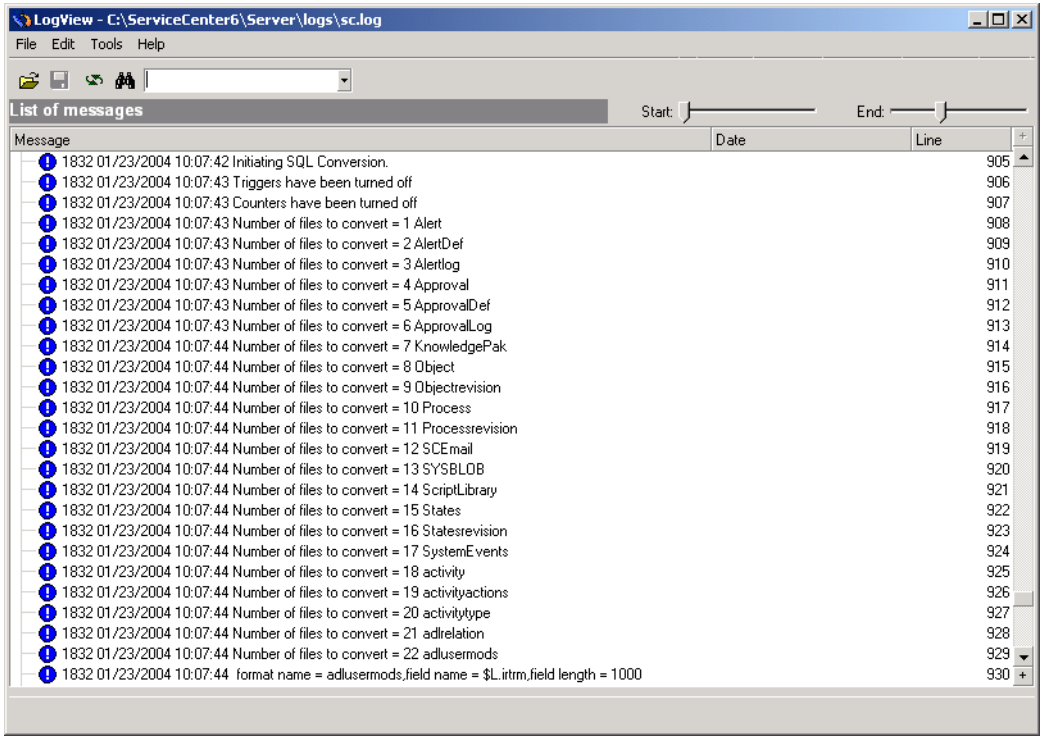
Note: The times calculated here are approximate. Conversion speed will vary based upon your particular server hardware, software, and network configuration.

ServiceCenter Log file

Conversion status messages are output into the standard system log file, sc.log as defined in the sc.ini file. During conversion, ServiceCenter posts any relevant messages to the standard ServiceCenter log file, sc.log. Use any text editor to periodically check the log. The conversion process creates entries in the ServiceCenter sc.log that display each step of the conversion. It adds these entries dynamically during the conversion process. If you encounter any problems converting your files, refer to this log to determine where the process broke down.

To access the log

1. Navigate to the directory in ServiceCenter server resides.
2. Open logs directory.
3. Open the file sc.log.
4. The log entries are displayed, showing each step of the process.



5 To 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 *ServiceCenter Log file* on page 140, for more information.
- Display the list of records that were not converted. See *Root record* on page 141 for more information.
- View the status data on the SQL conversion console. See *Monitoring the conversion process* on page 139 for more information.

Root record

To view files that have not been converted

1 Open the dbdict form in Database Manager. (For instructions, refer to *Database Manager* in the ServiceCenter online help.)

An empty dbdict form opens.

2 Select **Advanced Search** from the pull-down options menu.

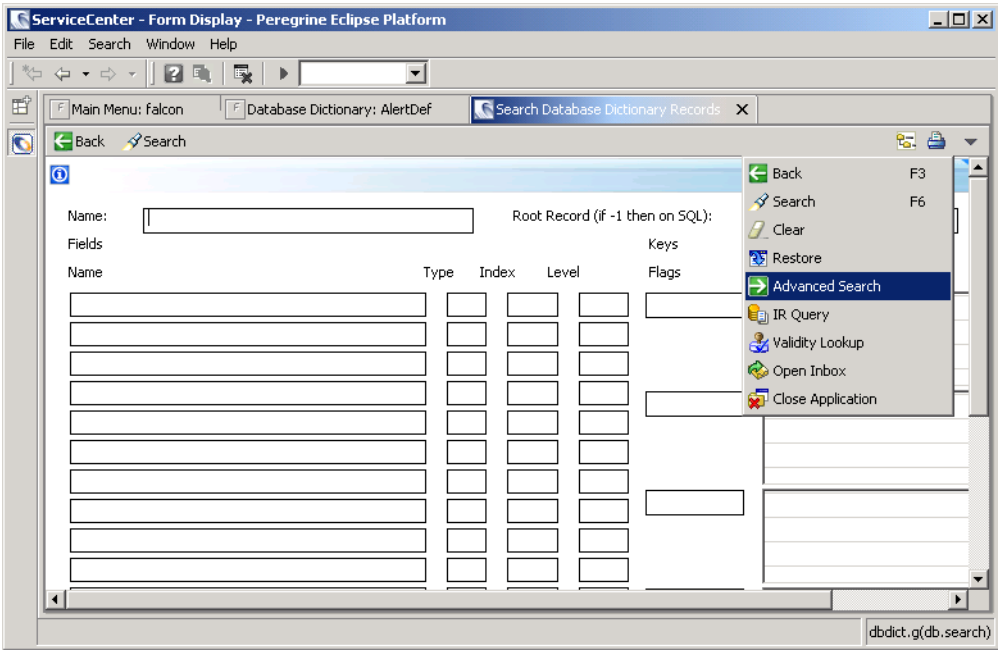


Figure 5-11: Selecting Advanced Search

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

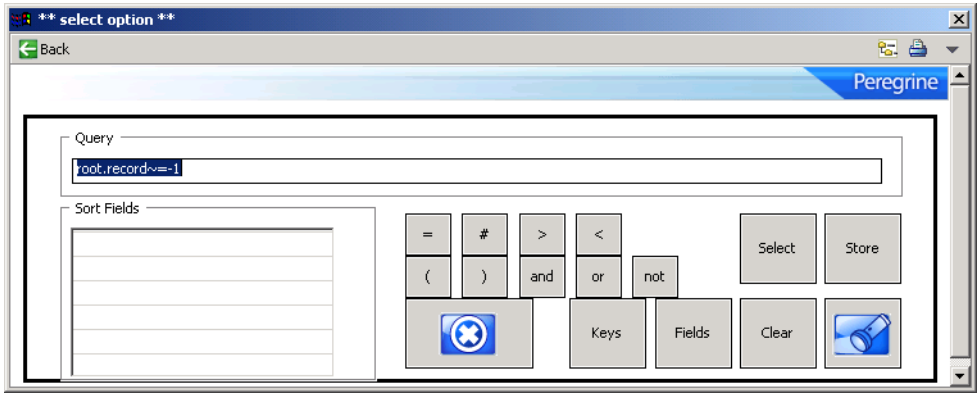


Figure 5-12: Query Window

4 Click **Execute Search** or press Enter.

5 If a search screen opens, enter any desired criteria, and Click **Search**.

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

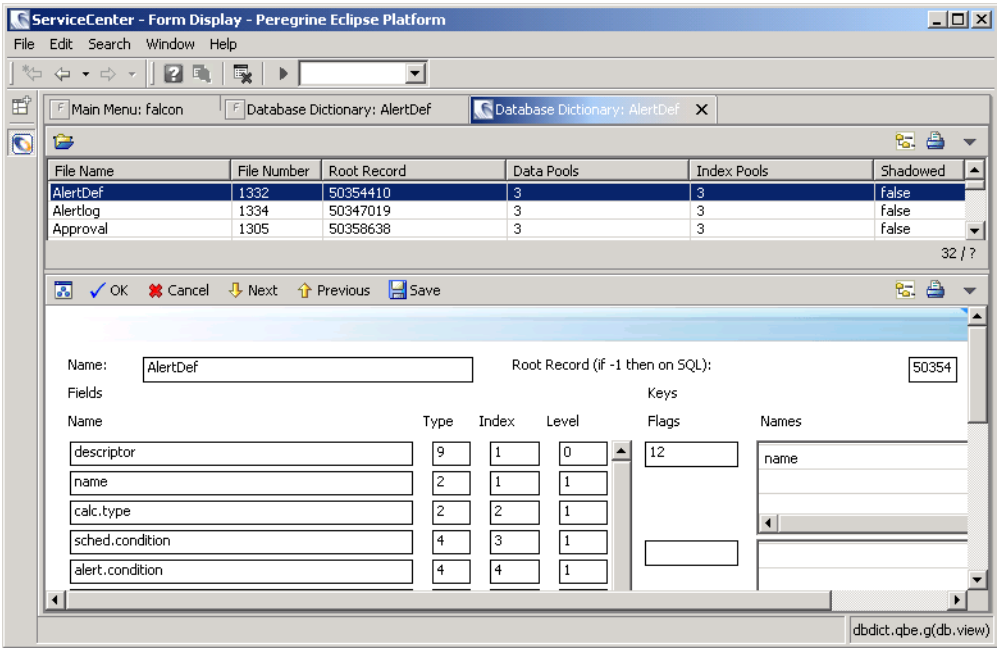


Figure 5-13: Files not converted

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, check the new table as well as indexes. See your oracle documentation for more details.

CHAPTER 6

Post Conversion

This chapter explains the procedures for editing a database after conversion to a Relational Database Management System (RDBMS) format. Its intended readers are system administrators and database administrators with a newly converted ServiceCenter® system.

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*. Perform it only if you are a qualified database administrator.

For detailed information comparing the database structures of compatible RDBMS formats with that of ServiceCenter's P4 file system, refer to the *Points to consider when deciding whether to use P4 or convert to an RDBMS* on page 16.

Note: After all tables are converted to the RDBMS, the database administrator should watch the RDBMS to see if any waits are occurring. If waits occur you can perform add rollbacks slowly as you perform checks 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 you 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 147
- *Process editing* on page 157

Manual editing

Overview steps for adding fields manually

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.* Perform it only if you are a qualified database administrator.

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 (for example, number, character, date/time, and so on) 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, refer to *Database Dictionary* in the ServiceCenter online help.)

   The dbdict record for your file opens.

2. Place the cursor in the **descriptor** field you want the new field to be under and click  **New**. (Alternatively, right-click the descriptor field and select New from the pop-up menu.)

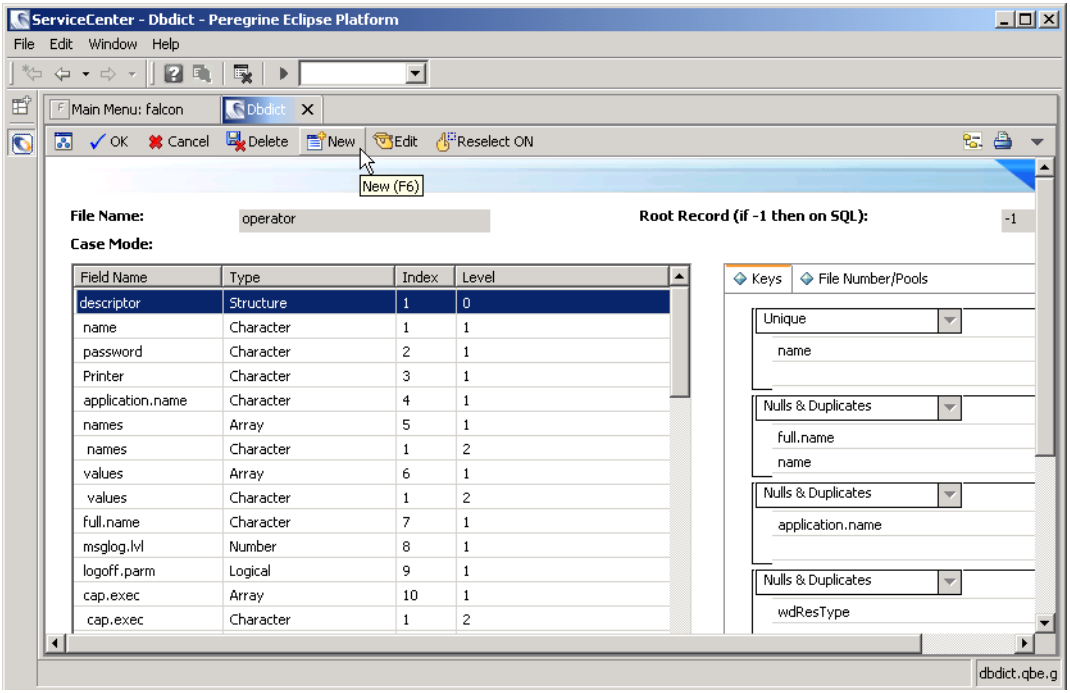


Figure 6-1: Database Dictionary of a converted file

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

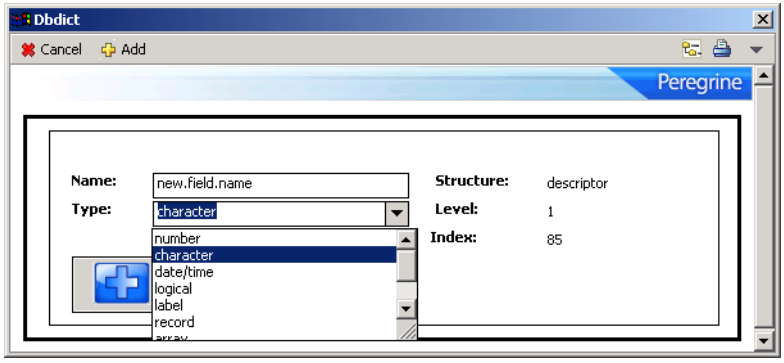
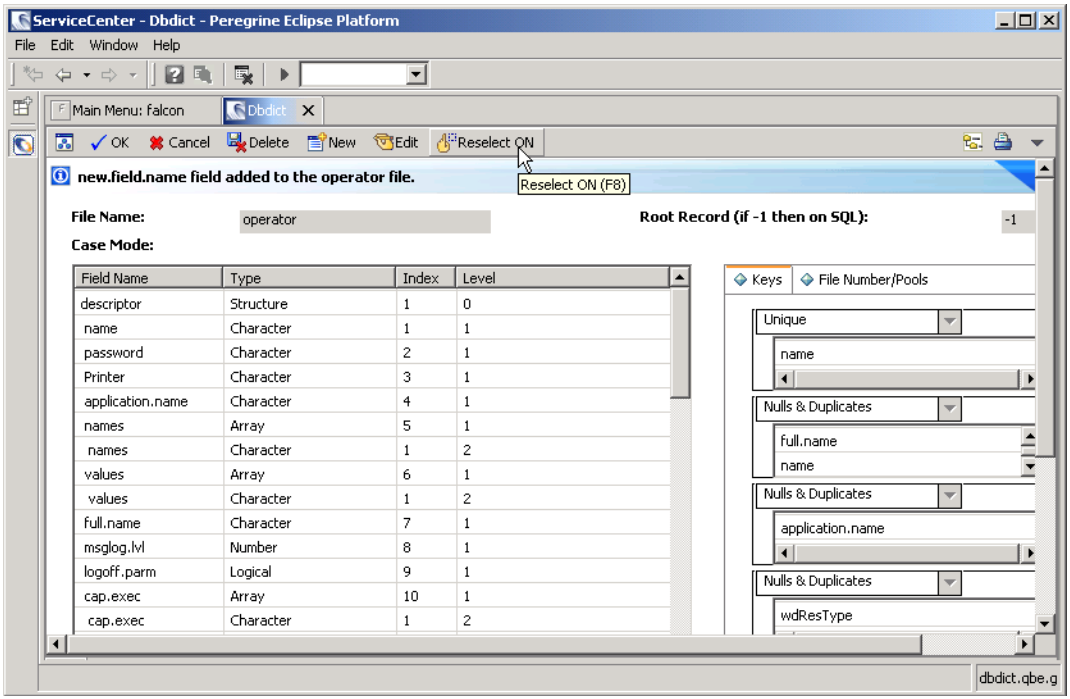


Figure 6-2: Adding a field with the Database Dictionary editor

4 Click **Add** to add the field to the Database Dictionary record.

5 Set the Reselect option.



The dbdict form displays the **Reselect ON/OFF** toggle button only when the Root Record is **-1**. (The root record will only be -1 if the file has already been converted.)

To change the reselect option click Reselect ON/OFF.

- If you have TRIGGERS that modify the data outside of ServiceCenter control, set **Reselect ON/OFF** to on. This forces 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, set **Reselect ON/OFF** to avoid unnecessary re-fetching of data from the RDBMS.

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.**

When the **Reselect ON/OFF** button is toggled **OFF**, a prompt opens, 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 exit the Database Dictionary record.

ServiceCenter displays the dbdict.alter.g form 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.

a Select **SC Alters** to allow ServiceCenter to update the SQL tables.

b 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.

Mapping the ServiceCenter field to the RDBMS column name

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.

You can modify the Field name mappings in two ways:

- *Modify an existing SQL mapping* on page 151 — the Expert option, it allows you to specify more details
- *SQL hints* on page 155 — the fast and easy option

Modify an existing SQL mapping

Be sure the data type (for example, number, character, date/time, and so on) 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.* Perform it only if you are a qualified database administrator.

To map field names

1 Open the **SQL Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Click **Modify an Existing SQL Mapping**.

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

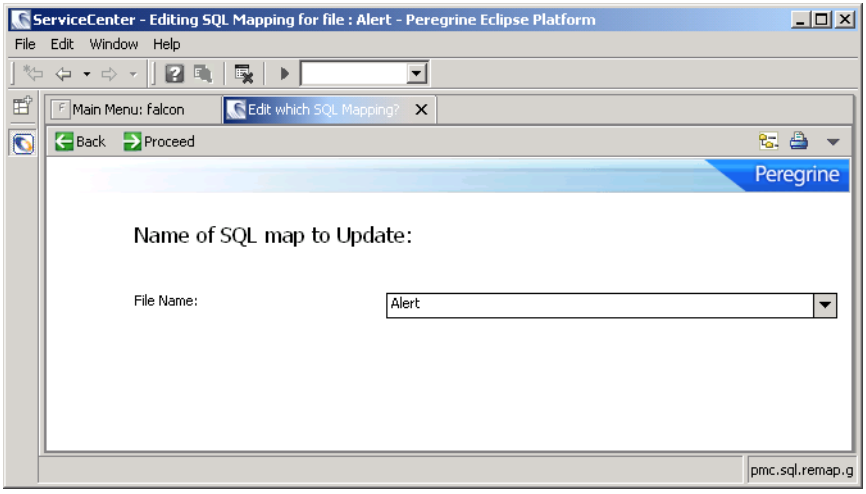


Figure 6-3: Specifying a map to edit

4 Click **Search** or press Enter.

The requested data map opens.

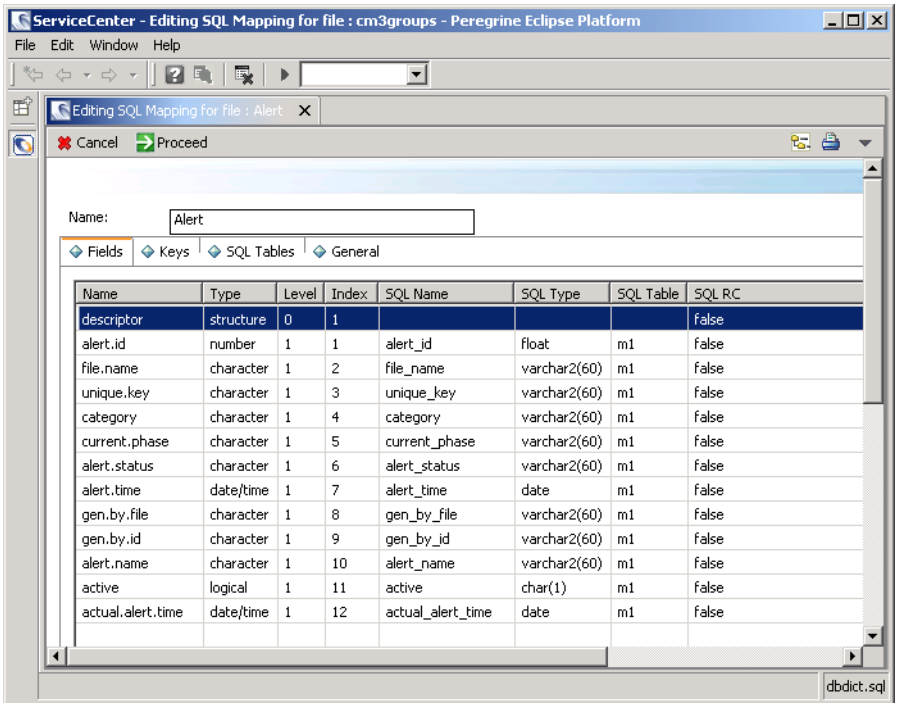


Figure 6-4: Editing an existing mapping using dbdict.sql

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

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.

Field	Definition
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.

Field	Definition
Table Keys	
Table Options	indicates tablespace, index space or other spaces used for ServiceCenter table or index.

Fields on the General tab

1 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	

Note: You can add new fields add using the Database Dictionary application. (For instructions, refer to *Database Dictionary* in the ServiceCenter online help.)
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.

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

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

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

Another prompt opens asking if you want to update the mapping template for the record.

4 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 opens informing you that your changes were made.

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

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 Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Select **SQL hints** to open the sql.hints file.

A search form opens.

3 Fill in information to restrict your search and click ✓ **OK** or press Enter.

4 Click **Search** or press Enter to find the records.

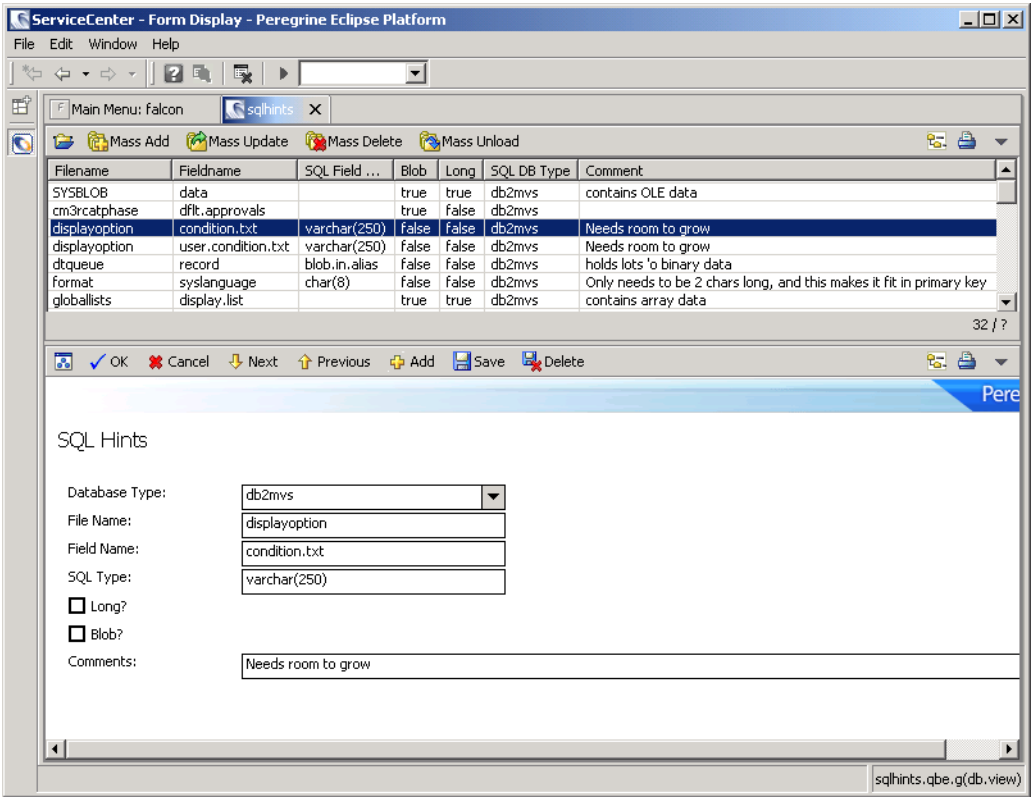


Figure 6-5: SQL Hints record list

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 you are defining.

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. 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. For example, 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. (See System Parameters in the ServiceCenter Online help.)

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 client.

7 Follow the instructions for adding a column to an RDBMS table. (See *Adding a column to the RDBMS table* on page 147.)

Important: Fields added to ServiceCenter's Database Dictionary in this fashion must be scalar in nature, since ServiceCenter adds them 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	Used in the mapping step, sqldbinfo defines database type, regulations and constraints. It is important that sqldbinfo is 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 155 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 Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

   The **SQL Utilities** menu (Figure 5-4 on page 126) opens.
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.

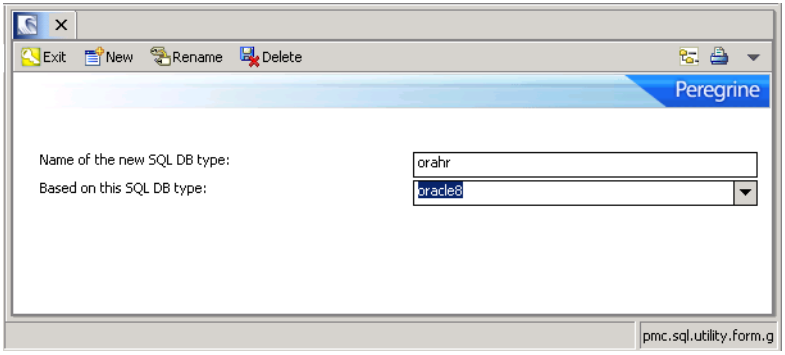


Figure 6-6: Adding an SQL database

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 opens.

6 Click ✓ **OK** to return to the record.

7 Click **Exit** to return to the **SQL Utilities** 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, discuss space considerations with the database administrator.

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 6-7 on page 160, a new field, array.field, is added to the incidents dbdict as an array with a second entry of character.

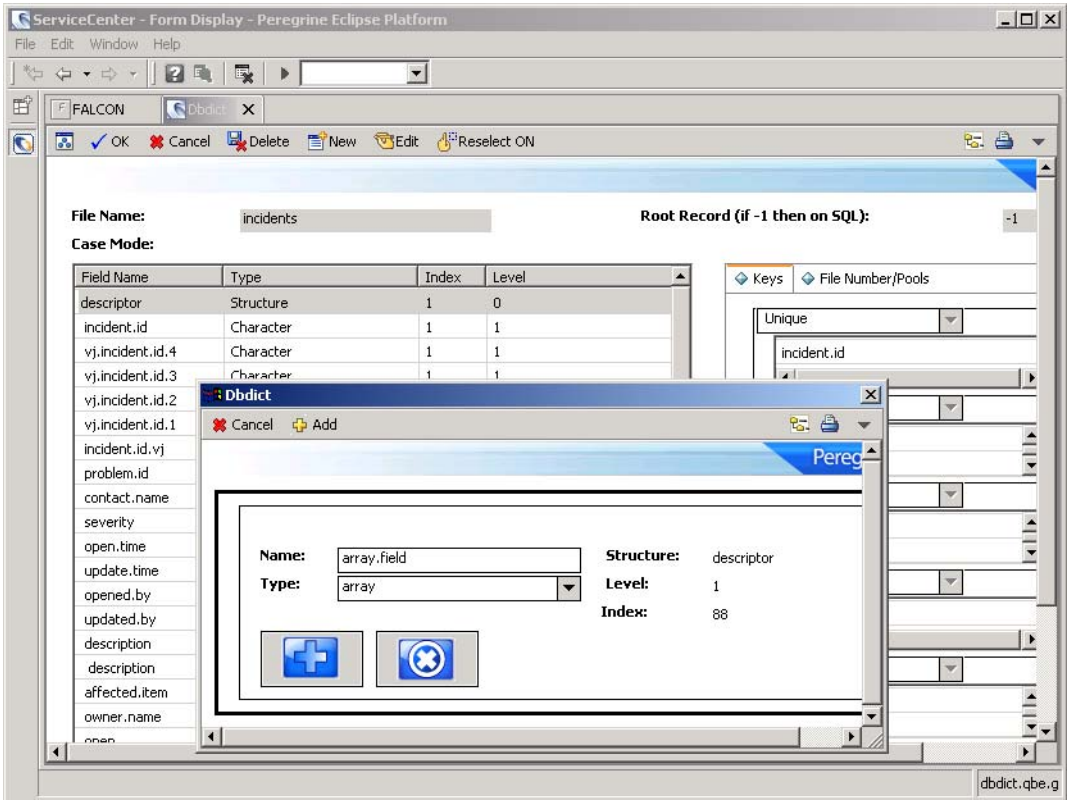


Figure 6-7: Adding new array fields to a table

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 Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Click **SQL Query Tool** in the **SQL Utilities** menu.

3 Enter your query in the **SQL Query** field, using proper syntax.

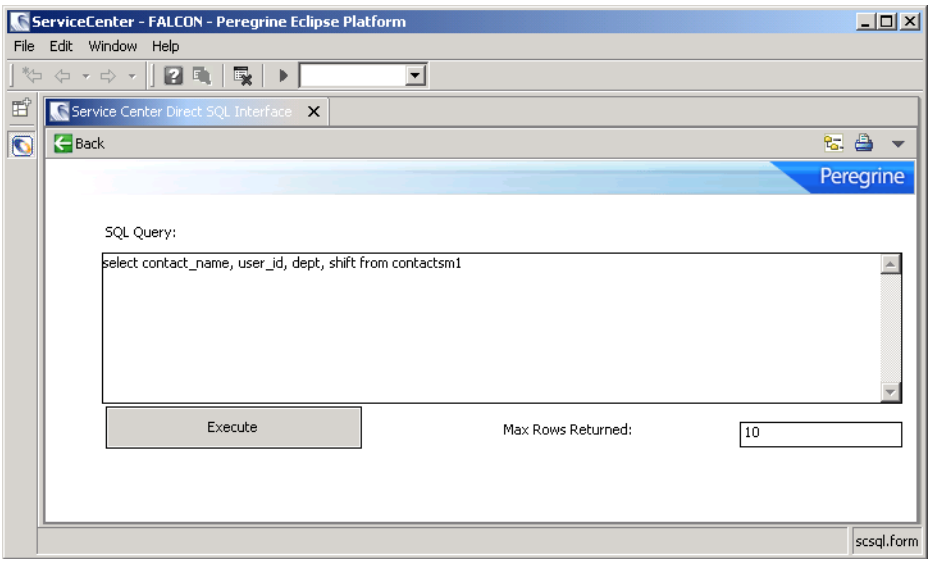


Figure 6-8: Querying the SQL database

4 Select the number of rows you want to display in the **Max Rows Returned** field.

5 Click **Execute**.

The requested data table opens.

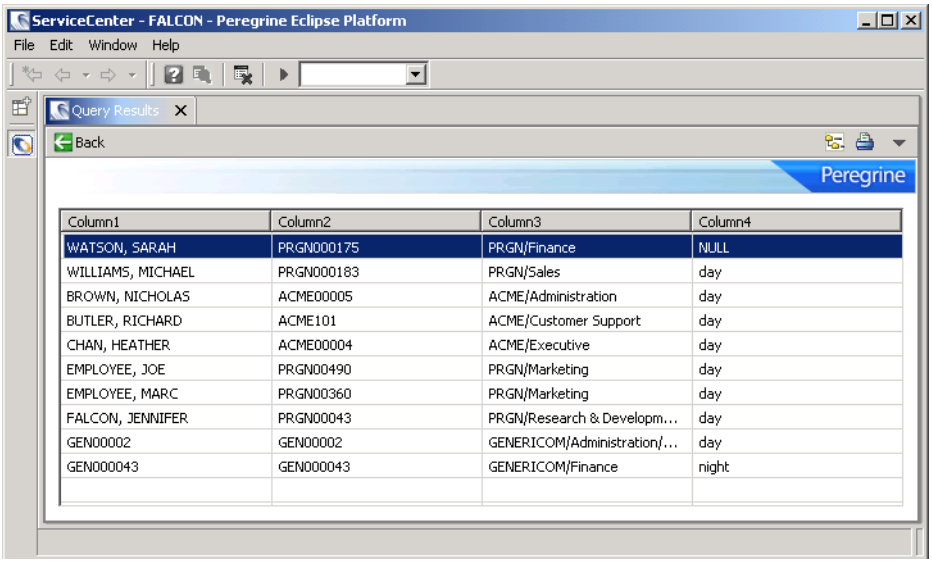


Figure 6-9: Data table returned from an SQL query

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.

Note: The P4 file system can be deleted after all data has been removed.

CHAPTER 7

Tuning for Performance

This chapter explains the procedures for enhancing performance for SQL, Oracle, and Sybase Relational Database Management System (RDBMS) formats. Its intended readers are system administrators and database administrators.

Topics in this chapter include:

Enhancing Microsoft SQL Server performance

After completing the SQL Conversion, 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, for example, *Microsoft SQL Server Administrator's Companion*.

To update information about the distribution of key values in specific indexes, 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

With ServiceCenter 5.1.1.0, Peregrine changed the default interface for Sybase to use the CTLIB/CSLIB interface. Prior to 5.1.1.0, ServiceCenter used the Sybase DBLIB interface only.

- To continue to use the DBLIB interface, specify the sybasedblib parameter in the sc.ini file.

  If the parameter is not specified, ServiceCenter uses the newer CTLIB/CSLIB interface by default. There are no values associated with this parameter.

  For more information on this parameter and other ServiceCenter parameters, refer to the Technical Reference section of the ServiceCenter online help.

  Refer to 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 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 a DDL and allows modifications to it before executing; therefore, you can make some of the changes, recommended below, to the DLL before the tables and indices are created. Refer to 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, and so on).

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, do this weekly. Use the parameter 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 recommendation 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, you could set the default tablespace for the Oracle user responsible for the conversion, to the desired tablespace. Do this on the Oracle side before performing the conversion.

With Oracle 7.3 or later, you can rebuild Oracle indexes 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: Tables in ServiceCenter can be very small or very big. ServiceCenter sets the initial table size is set automatically. If the default setting is that of the maximum size file, all the files are likely to have a lot of wasted space. Since 1000 or more tables are created, it might not be possible, on all systems, to give all tables the same table extent as the largest one. Available space quickly becomes an issue if you choose this course.

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) )
```

Select the three numbers you need from the V$SYSSTAT table:

```
select name, value from V$SYSSTAT;
```

If your hit ratio is less than 90%, 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 99%, 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, for example, 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. It places the first long field type in the main table; and maps additional arrays to tables ending with a1, a2, and so on. 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 42.

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 170.
2. Set the `DSMAX` parameter for DB2 sufficiently high. See *The DSMAX parameter* on page 170
3. Ensure that the EDM buffer pool is large enough. See *The DB2 EDM buffer pool* on page 170

4 Beware of losing space by dropping and creating tables. See *Dropping and creating tables* on page 171

5 Be sure to set up the appropriate tablespace. See *Tablespace* on page 171.

6 Set up high activity files for greater efficiency. See *High activity files* on page 173.

7 Use the debugging tools. See *Database debugging parameters* on page 174.

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, you can make some of the changes to the DDL, recommended below, before the actual data is converted.

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. Start with a value of 1MB.

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 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, you can these modify tables to go into the 4K tablespace. In the future, ServiceCenter will recognize this, and will use the 4K tablespace where indicated. Currently the you 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

System Files

displayscreen	dtqueue	dtshad
enclapplication	eventin	eventout
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, and so on.) It places each of the array tables (a1,a2,a3, and so on) into the 32K tablespace. If you know that the data going into the array will not exceed 4K, you can put the table 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, for example, 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.

8 Troubleshooting

CHAPTER

This chapter was designed to aid in trouble shooting conversion and converted 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. Its intended audience consists of system administrators and database administrators.

Topics in this chapter include:

General issues

Most errors occur when one of the conversion steps was not completed properly. If you or the database administrator cannot determine the cause of the problem, do not hesitate to contact Peregrine Systems Support.

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

You can find error messages in:

- The Message pop-up window.

  Click **View Messages** 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 sc.log file

For more information, see *Monitoring the conversion process* on page 139.

Tokenized SQL for Oracle & DB2 mapped systems

By default, ServiceCenter attempts to use tokenized SQL in the database operations of Oracle & DB2 mapped systems.

When you execute a query, or a series of queries, ServiceCenter prepares SQL statements once, with tokens for the parameters. Then, as you repeat the query on a table with a different argument, ServiceCenter only has to change the value of the token target, and reissue the modified SQL command, rather than preparing a new set of SQL statements.

What should I do first?

If there is some type of problem with your ServiceCenter system related to the RDBMS, typically the first debugging step is to add the parameter sqldebug:1 to your sc.ini file, restart your system and reproduce the problem. If you have done this, you will probably see SQL statements like this in the log:

```
SELECT :nameh FROM probsummarym1 WHERE number=:nameh;
```

:nameh is typically the token you see in the SQL statements in the log. It may not tell you all you need for troubleshooting.

To force ServiceCenter to use dynamic SQL instead of tokens (and force the debugging to display the actual values being used in the database operations), specify this parameter in the sc.ini file (along with sqldebug:1):

```
sqlreuseablesql:0
```

This parameter defaults to the value of 1 (ServiceCenter reuses SQL where possible, as shown with the :nameh tokens, above). Setting sqlreuseablesql to 0 turns it off, causing ServiceCenter to generate and prepare fresh SQL statements for each operation. Using sqlreuseablesql:0 and sqldebug:1, the log would show the earlier SELECT example as something like:

```
SELECT * FROM probsummarym1 WHERE number=IM1001;
```

Note: Occasionally, the SQL problem you are having is caused by something in the reusable SQL process. If you set sqlreuseablesql:0, the problem might no longer occur, but queries will take longer.

Common RDBMS related errors

Oracle

Problem

Cannot connect to Oracle database.

Solution

- Set the user environment variable **ORACLE_SID** to the name of the Oracle server to which you want to connect.

- Make sure that the user can connect to the Oracle server by running the Oracle sqlplus utility with the following parameters:
- sqlplus <userid>/<password>
- Set the sqldb initialization parameter in the ServiceCenter sc.ini file to the connection string for Oracle. See System Parameters in the ServiceCenter Online help 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>

- Set the sqldb initialization parameter in the ServiceCenter sc.ini file to the name of the service in the file tnsnames.ora. See System Parameters in the ServiceCenter Online help 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.
- Ensure 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>.
- Set the sql initialization parameter 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>
- Run normal TCP/IP checks, (Ping, Telnet, and so on), verify the servicename, verify that the listener is running at the remote node and verify that the address parameters specified in TNSNAMES.ORA are correct.

If you use interchanges, check that all needed to make the connection are up and running. Ensure ORACLE_SID is correct (matches the server to which it is connecting), and ensure TNS_ADMIN is set correctly to point at the location of the configuration (.ORA) files.

Microsoft SQL Server

Problem

Cannot connect to SQL server database.

Solution

- Ensure 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>
- Ensure that the ODBC driver is configured correctly.

To edit the ODBC driver configuration, from the Windows Start menu, open **Settings** > **Control Panel** > **Administrative Tools** > **Data Source (ODBC)** and click **Configure**.

- Ensure that the ODBC driver can connect.

  To test the connection, use any ODBC query tool. In Excel, open: **Data** > **Get External Data** > **New Database Query**. Choose the ServiceCenter ODBC driver as your data source. If it connects, you will be able to see the ServiceCenter tables.

Sybase

Problem

Cannot connect to Sybase database.

Error message

Sybase error 20012 - Server name not found in interface file.

Solution

- Set the user environment variable **DSQUERY** to the name of the SQL server to which you want to connect. You can find the SQL server names in the Sybase interfaces file.
- Ensure 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>
  ```

  Set the sqldb initialization parameter in the ServiceCenter sc.ini file to the name of the SQL server. See System Parameters in the ServiceCenter Online help for more details.

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]
```

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.

9 Shadowing P4

CHAPTER

This chapter aids in shadowing their P4 systems. It discusses the concept and strategy of shadowing, as well as the procedures for shadowing individual files and for stopping shadowing. Its intended audience consists of ServiceCenter® system administrators and database administrators interested in shadowing a database.

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, ServiceCenter does not return control 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. After that, ServiceCenter returns control 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. See the instructions for starting the server with only one user in *Conversion process* on page 124.

To shadow files

1 Open the **SQL Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

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 *Conversion process* on page 124 for instructions.

The P4 to SQL Shadow Console opens.

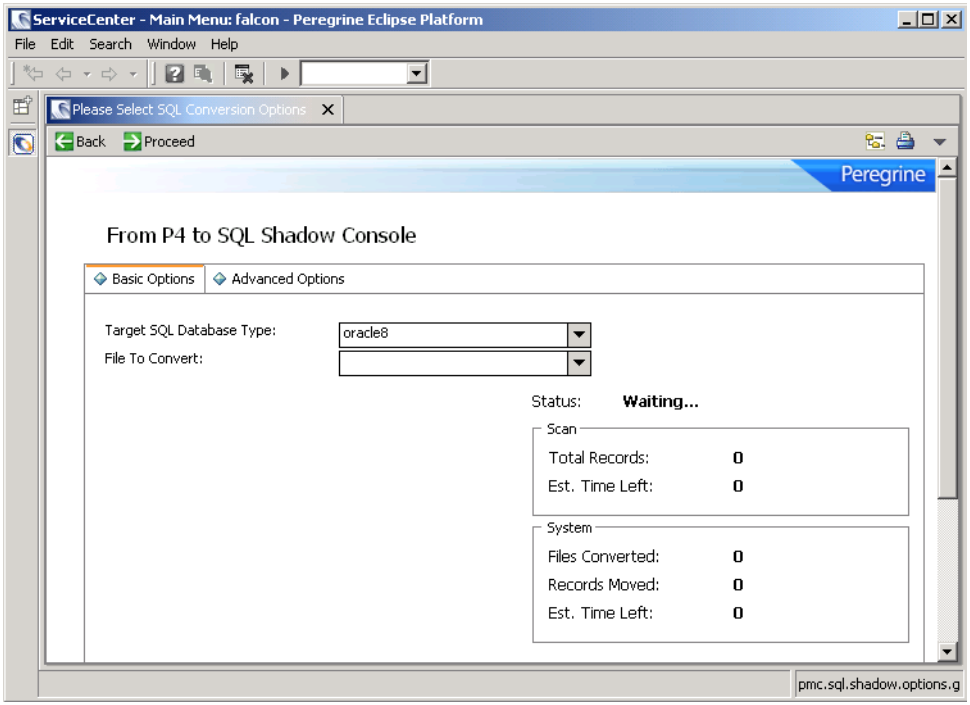


Figure 9-1: P4 to SQL Shadow Console

3 Select a file from the **File To Convert** list.

Note: 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: Contact your database administrator before selecting any option *other* than the default.

6 Click **Proceed** to shadow the file selected to your RDBMS format.

A message opens stating that the file has been moved to your RDBMS, indicating the file has been shadowed.

Stop shadowing

Disable the shadowing status of a ServiceCenter file that has been converted, in order to reconvert or update the file on an RDBMS.

Select one of the following two methods to stop shadowing files:

- Use the *Stop Shadowing Files menu option* on page 187, to turn off shadowing and unconvert the files.
- Use the *Database Manager method* on page 188 to turn off the shadowing feature without unconverting the file.

Stop Shadowing Files 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. See the instructions for starting the server with only one user in *Conversion process* on page 124.

To turn off the shadowing feature

1 Open the **SQL Utilities** menu. See *To open the SQL Utilities menu and begin conversion* on page 125.

The **SQL Utilities** menu (Figure 5-4 on page 126) opens.

2 Click **Stop Shadowing Files** to stop shadowing. This option will not work if you have not done the preconversion server setup. See *Conversion process* on page 124 for instructions.

3 Select a file from the **File to Convert to P4** drop-down list.

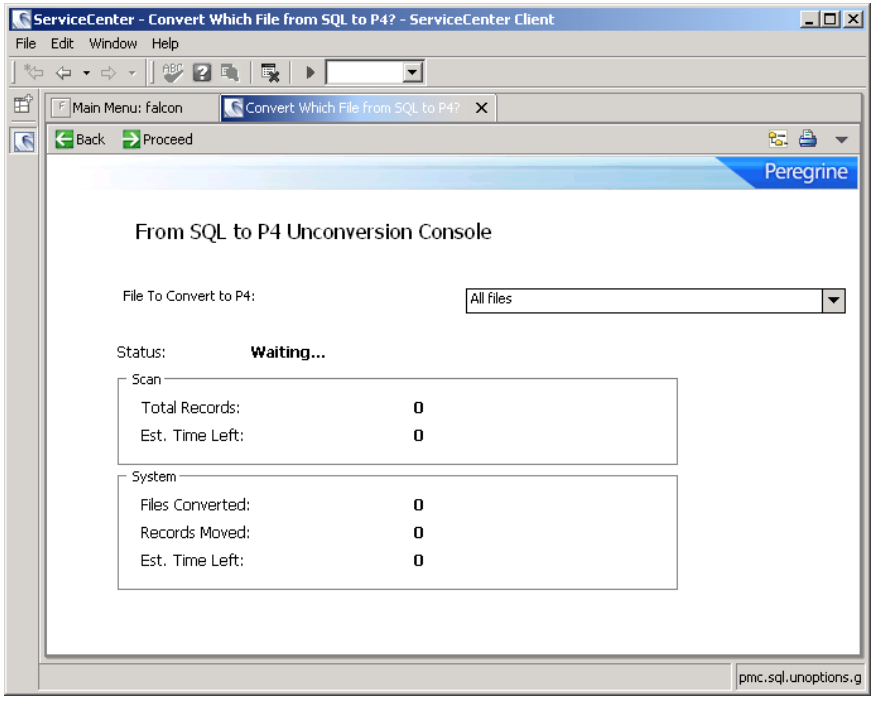


Figure 9-2: Converting shadowed files to P4

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

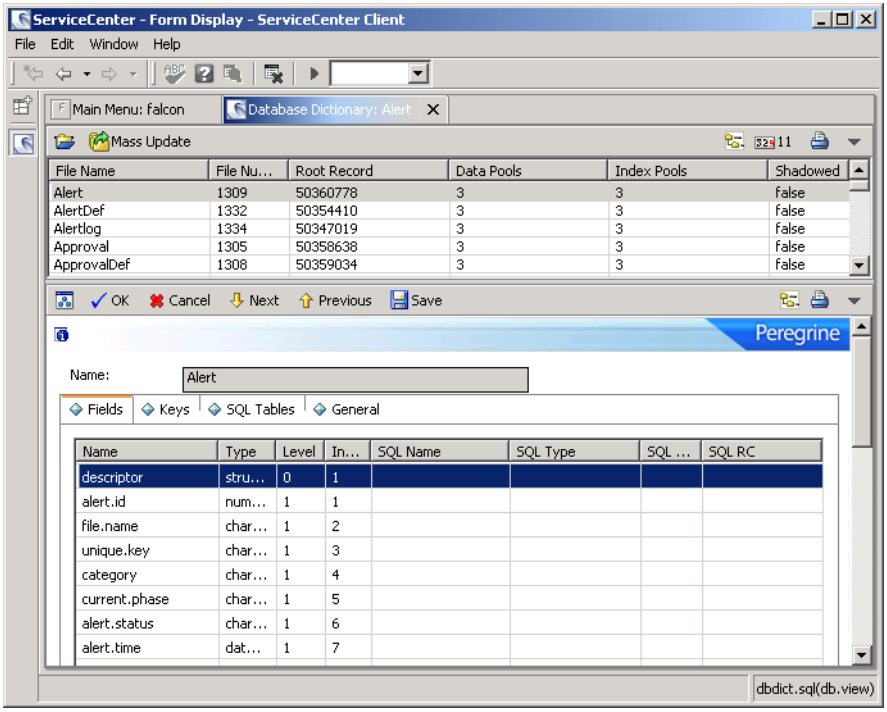
To turn off the shadowing feature without unconverting the file

1 Open the **dbdict.sql** file in Database Manager. (For instructions, see the Database Manager online help.)

A blank **dbdict.sql** form opens.

2 Click **Search** in 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 discusses the procedure for converting an Relational Database Management System (RDBMS) database back into the ServiceCenter® P4 format. Its intended audience consists of system administrators and database administrators.

Conversion procedure

To convert all or part of a database back to P4

1. Start your system as a sole user, as described in *Conversion process* on page 124; shut down the server, restart the server, and connect with a client.
2. Open the **SQL Utilities** menu. For information on how to open the **SQL Utilities** menu, see *To open the SQL Utilities menu and begin conversion* on page 125.

   The **SQL Utilities** menu (Figure 5-4 on page 126) opens.
3. Click **Move Files From SQL to P4** in the **SQL Utilities** menu to convert your RDBMS database files to P4. This option will not work if you have not done the preconversion server setup. See *Conversion process* on page 124 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.

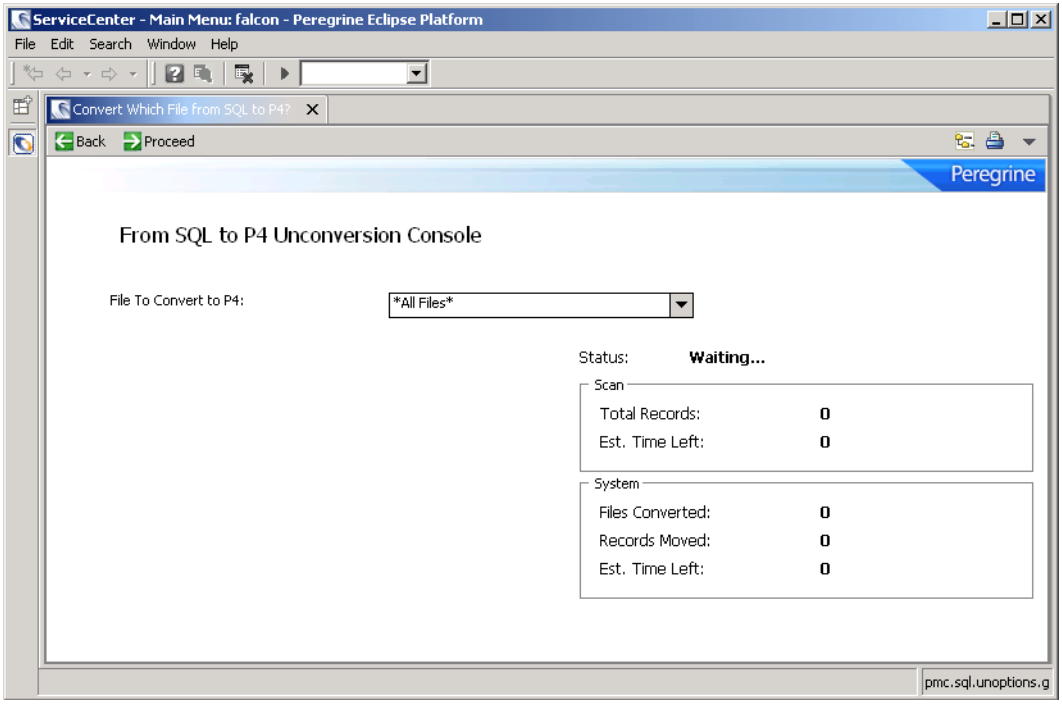


Figure 10-1: Selecting files to convert to P4

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 provides 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. Its intended audience consists of system administrators and database administrators.

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

These files have no physical database activity in a non-customized system. They are virtual files used to store data internally, and therefore 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 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

File	Description
audit	A ServiceCenter data file that records changes in field values.
auditspecs	A ServiceCenter data file that contains the specifications for ServiceCenter data files and fields within a file that should be audited.
devaudit	Active only if development revision tracking has been turned on, this file 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	Active only if development revision tracking has been turned on, this file controls what type of revision tracking ServiceCenter performs. This file will be read once by each process in the system to see if revision tracking is active.

File used to monitor database activity

This file monitors database activity.

File	Description	Activity
transactioncount	This file tracks the type of access being done on all files within the system. ServiceCenter keeps track of these statistics in shared memory as it runs. 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 you issue the command **scenter -reportdbstats**. For more information, see *Tools to determine file access characteristics* on page 225.

Files used to gather system runtime statistics

These files 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

These 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. You can safely delete files that are flagged as obsolete from systems 4.0 and later.

sqldbdict - obsolete	sqldbfield - obsolete	sqldbinfo
sqlhints	sqlmapfields - obsolete	sqlmapoptions
sqlmapping	sqlmaptables - obsolete	sqloptions
sqlsystemtables	sqltransactionlog	sqlupgrade
sqlwords		

File used to display version information

This file displays version information.

File	Description
scversion	Used to report RAD and RTE version and licensing options.

Files associated with upgrades

These files are only used when an application upgrade is performed. Separate them from the main application files. During an upgrade, these files have extremely high activity.

errorlog	patches	signaturemake
signatures	upgdbdict	upgenclapplication
upginfo	upgrade	upgradehistory
upgradeobjects	upgradepseudolog	upgraderesults
upgradestatus	upgradesystables	upgsqlsys

Application development files

These files are reference-only files in a production system. They are RAD program 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)

These files have very low activity during normal daily ServiceCenter operations. ServiceCenter accesses them during system startup operations and for each user during login. You can place these files in an area away from the primary application files. They are generally read-only and are cached in memory after the initial read.

System related files

These 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 using 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 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 you can use 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, and so on) environmental settings. The file contains a small number of records that are cached in shared memory the first time ServiceCenter reads them.	system-wide
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
ocmoptions	Phase definition. This file is shared by Requests, Line Items, and Orders.	Request Management

File	Use	Application
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 ServiceCenter reads them.	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 you can use 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 ServiceCenter reads them.	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)

These files have very low activity during normal day to day ServiceCenter operations. ServiceCenter accesses them only as needed. You can place these files 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

These files are primarily read-only files that ServiceCenter accesses throughout the session, but the data from these files is cached on both the client and the server. The first time an entry is needed from these files, it is read from the ServiceCenter P4 file system (or an RDBMS) and then 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, and so on) environmental settings. The file contains a small number of records that ServiceCenter caches the first time it reads them.
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 displayed menus.
pmenv	Contains the Incident Management security profile for a user or group. Usually a small number of records that ServiceCenter caches the first time it reads them.
smenv	Contains the Service Management security profile for a user or group. Usually a small number of records that ServiceCenter caches in the process storage the first time it reads them.
usergrid	Defines the fields that a user wants displayed in record lists.

High reference files

Pay attention to the following files as needed. 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 using the file's unique key. (For key definitions, refer to *Database Manager* in the ServiceCenter online help.)

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 using 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 using the unique key value.
format	This file contains the layout for the presentation of data and forms to the user. Access is generally using 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 before ServiceCenter can display forms and messages in other language. When optimizing access to this file, consider that records from this file are cached, but the activity is high. This file is generally accessed using the unique key value for the file.

Generally high activity / high update files

These files are always high activity files both in query activity and in update activity. Separate the files and indexes, if possible.

File	Description
clocks	Tracks the amount of time required to complete various tasks. Updates to calls, tickets, and change requests involve the **clocks** file. This file has a high update rate and grows in size. Be sure to manage its file size.
counters	The **counters** file is similar to **numbers** in that both 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 using the unique key on the file and the record is only referenced so that it can be updated. For example, the number file tracks 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, and so on. 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. Be sure to manage its file size.

The schedule, dtqueue, and sqlqueue files have a high transaction rate. If the system is functioning properly, though, they should not contain large numbers of records. Records should remain in the files dtqueue and sqlqueue 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.

Note: Check the basic health of the system by looking in the files schedule, dtqueue, and sqlqueue.

Files with high insert and deletion activity

These files have high insert and delete activity. Separate them from the main application files. These files are used in printing reports, so their 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. Purge it 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. Purge it 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 a user requests help.
helptext	Used for on-line help.	Inserts an update only when you add or change the help. Read when a user requests help.
import	Contains the import descriptions for the database import facility.	Used only during data imports.
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.
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
unload	Specifications for unloading data from system.	
vsaminfo	Contains information for the vsam background scheduler.	

Primary application files

Files shared by multiple applications

Multiple applications share theses files:

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. The availabilitymap file is an associated reference file that defines the hours a device should be available.
budgetcenter	Contains the areas or companies credited for assets specified in a contract.
budgetcode	Contains codes or IDs used to track the different budgets managed by different departments.
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 will 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 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 wants 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.
saphrcostcenter	Contains the areas or companies paying for assets specified in a contract.
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. Because of its size, separate this file from the primary application files.
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

Change Management uses these files:

File	Description
cm3r	Contains primary information about change requests. 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

Contract Management uses these files:

File	Description
contract	The primary contract file, contains a record for each asset contract. Joined with numerous attribute files based on the type of contract. When looking at contract items, there could be retrieval activity against the budgetcenter, budgetcode, company, contacts, contractitem, contractterms, currency, dept, language, model, ocmo, ocmq, operator, payment, sapharcostcenter, and vendor files.
contractcategory	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 you can use 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	Checks for software compliance by counting software installations and software licenses.

Attribute Files

contractlease	contractsoftware	contractsupport
contractmaintenance	contractwarranty	

Event Services

Event Services uses these files. 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

Incident management uses these files:

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 that update 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

Inventory Management uses these files:

File	Description
device	Primary inventory file containing a record for each asset 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, devparent, location, pcsoftware, 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, and so on.)

Attribute Files		
computer	displaydevice	furnishings
handhelds	mainframe	networkcomponents
officeelectronics	softwarelicense	storage
telecom		

IR Expert

Treat the files used for IR Expert searches 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 using the unique key, and the QBE qualification is checked. This can present a lot of activity against these files. Isolate these files so the I/O done against them does not interfere with the primary application data files.

IR Expert uses these files:

File	Description
core	IR Expert file created from other knowledge sources. Central file where all IR Expert knowledge (knowledge, KnowledgePak, probsummary, and so on) 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.
scirexpert	With ServiceCenter 6, this file is used to store IR data, instead of the old external IR Expert files (ir.*).

Request Management

Request Management uses these files:

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

Service Management uses these files:

File	Descriptions
incdepends	Contains dependencies that a call must meet before a user can close the call. 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 also involves SLA, dept, cmlabor, expline, caldaily, counters, number, and screlation, if these files are used in your installation.

Work Management

Work Management is the only module that uses these files. If Work Management is not used by your system, ignore these files.

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

File	Description
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.
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 increase 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

You can either leave virtual files (never contain data) and files that are not used in P4, or create RDBMS tables for them.

The following files fall into this category:

- *Normally empty files* on page 194
- *Files from applications that are not currently supported* on page 194

Tablespace for special purpose files

Look at the following sets of files and, if the feature is being used, put these files into their own tablespaces. If the feature is not being used, the files can go into the same tablespaces as the other unused files.

These files are feature specific:

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

These 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
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

These 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

You can isolate files associated with the upgrade process in a tablespace of their own. When you perform an upgrade, the activity on these files is intense, but at other times the files remain untouched.

errorlog	patches	signaturemake
signatures	upgdbdict	upgenclapplication
upginfo	upgrade	upgradehistory
upgradeobjects	upgradepseudolog	upgraderesults
upgradestatus	upgradesystables	upgsqlsys

Tablespace for spool data and logs

The spooling files are active during reporting sessions, but otherwise inactive. They have high insert and delete activity. Because they are used for reporting, they have a tendency to become large, then shrink down periodically.

Spool	spoolheader	syslog	msglog

Tablespace for event data

The event files are accessed at the same time as the target for the event. Put the event file and its target in separate tablespaces if possible.

Eventin	eventout

Tablespaces for Inventory data

Inventory data is accessed by all components of the system. Consider separating inventory data from the other files. In addition, separate heavily used attribute files from their device files, as the device file and associated attribute file are always accessed concurrently.

computer	device	displaydevice
furnishings	handhelds	mainframe
networkcomponents	officeelectronics	pcsoftware
softwarelicense	storage	telecom

Tablespaces for high activity files

These files have high update activity. Consider placing these files in a tablespace of their own. Also, if possible organize these files so that there is a single record per page to increase concurrency. Then, 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. Use the tools available in ServiceCenter or provided by the target RDBMS to determine the frequency and concurrency of use of the files.

ServiceCenter system files

assignment	availability	cmlabor
company	contacts	curconvert
dept	expline	inbox
location	operator	outage
outagedetail	screlation	slaactive
slamonthly	slamonthlyag	slaresponse
SYSBLOB	validity	vendor

Change Management files

cm3r	cm3rpage	cm3t	cm3tpage

Contract Management files

contract	contractlease	contractmaintenance
contractsoftware	contractsupport	contractwarranty
payment		

Incident Management files

pmnotes	problem	probsummary

Request Management files

ocml	ocmlpage	ocmlrec
ocmo	ocmopage	ocmphaselog
ocmq	ocmqpage	

Service Contract files

cmparts	downtime	pmcost

Service Management files

incidents	incdepends

Work Management files

wdCategory	wdCodeDetails	wdCriteria
wdCustomize	wdCustomReports	wdCustomSections
wdOffHours	wdPriorityLookup	wdQueueValues
wdResExp	wdResHierarchy	wdSchFilter

IR Expert/Knowledge Engineering files

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. It updates the transactioncount file. See *File used to monitor database activity* on page 196.

Running a trace

This data is kept in shared memory. You can access it by issuing the following command from the ServiceCenter RUN directory:

scenter - reportdbstats

This command lists all files accessed by the system during processing.

Important: Using this report, administrators can identify the files accessed during typical processing. The strategy to identify the files is quick. Identify the files prior to changing from a test to a production system.

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

Identifying a file

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

   Run this command 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.

Determine the sequence of access to a file

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 debugdb242query: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 *italic* represent database requests 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
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 *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

Data Definitions

This appendix explains what various ServiceCenter® fields are for. It contains data definitions for ServiceCenter files. Its intended audience consists of system administrators and database administrators.

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 that 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.
unassigned	Logical	Determines if the issue has yet to be assigned.

Field Names	Types	Description
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.
site.category	Character	SLA site identification type.

Field Names	Types	Description
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.
alternate.fax	Character	General information field.

Field Names	Types	Description
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

I

K

L

Q

R

S

T

U

V

W


Peregrine
SYSTEMS