

HP Service Manager

For the Windows and Unix operating systems

Software Version: 9.30

Tailoring Best Practices Guide

Document Release Date: July 2011

Software Release Date: July 2011



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 1994-2011 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

For a complete list of open source and third party acknowledgements, visit the HP Software Support Online web site and search for the product manual called *HP Service Manager Open Source and Third Party License Agreements*.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and log on. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport log on page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at:

<http://www.hp.com/go/hpsoftwaresupport>

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Tailoring Best Practices Guide.....	1
Contents.....	5
Introduction to tailoring best practices.....	8
Assumptions.....	8
What is tailoring?.....	9
Form tailoring.....	10
What is a form?.....	10
Form tailoring use cases.....	10
Selecting values from options in a control.....	11
Combo Box.....	11
Comfill.....	12
Selecting values from options on a form.....	14
Radio Button.....	14
Checkbox.....	15
Changing the options available based on prior user input.....	16
DVD select statements.....	16
Recursive fills.....	18
Displaying related data.....	20
Subform.....	20
Grouping related controls.....	22
Group.....	22
Frame.....	24
Typing text into a control.....	25
Text.....	25
Text Area.....	26
HTML Editor.....	27
Managing global lists.....	28
Removing unused global lists.....	29

Reducing upgrade conflicts.....	29
Queries.....	32
How do I write a good query?.....	32
Indexing.....	32
Query operators.....	33
Function usage.....	33
Variable usage and null values.....	34
Where are queries used?.....	34
Queries in background processes.....	35
Queries in global lists.....	35
Queries in links.....	35
Queries in Format Control.....	35
Database (RDBMS) mapping.....	36
How do I create good RDBMS mappings?.....	36
Database dictionary (DBDICT) utility.....	36
Mapping rights.....	36
Array mapping options.....	37
Field in main table.....	37
Field in alias table (Deprecated).....	39
BLOB in main table.....	39
BLOB in alias table (Deprecated).....	40
Multi-row array table.....	41
Example: Re-mapping an array.....	42
Existing array mapping.....	42
Existing array query performance.....	43
Prepare to re-map the array.....	44
Determine the next free alias table.....	45
Determine the data type of the array sub-element.....	45
Determine the length of character fields.....	46
New array mapping.....	47
New array query performance.....	48
Valid data type changes.....	48

When are RDBMS mappings added or updated?..... 49

Chapter 1

Introduction to tailoring best practices

The purpose of this document is to provide customers with best practices to successfully tailor HP Service Manager. This document describes the most common tailoring activities and how to best accomplish them to meet particular use case goals. These best practices represent the collective experience of both the HP organization and other customers. By following these best practices you should realize the following benefits.

- Less need for support to complete a Service Manager implementation
- Improved system performance and efficiency
- Improved customer experience with your implementation
- Easier upgrades with more opportunity to review new features and functions

The following chapters provide information on how to accomplish common tailoring activities, tips and tricks from experts, and examples of how to meet particular use case goals. These best practices are not the only ways to implement Service Manager, nor are all tailoring scenarios covered.

Assumptions

The information in this document is most useful if all of the following criteria are met.

- You are implementing HP Service Manager version 9.20 or greater
- You are a system implementer with general knowledge of Service Manager tailoring tools and techniques
- You have a design based on a clearly-defined and agreed-upon business plan or system model to direct your tailoring efforts
- You have an understanding of Service Manager's out-of-box data model and ITIL work flows

This document assumes you have access to features available since the HP Service Manager version 9.20 release. Some of these best practices can also apply to earlier versions, but some performance benefits and user experience features require you have access to newer application versions. In general, best practices that rely on a particular tool or control will have the most benefit when implemented with the newest version of the applications.

System implementers will gain the most benefit from this document as it assumes you are already familiar with basic tailoring activities and have access to all the tools referred to within. At a minimum, you will need administrative access to all of Service Manager's tailoring tools. It is also recommended that you have earned an Accredited Integration Specialist certificate.

Having a business plan or system model will help you determine which parts of Service Manager you will need to tailor. Furthermore, having a list of objectives will help you decide between competing goals. For example, if your overall tailoring goal is to provide the best user experience for your Service Manager users, then you may choose options that maximize usability rather than the best practices that are easiest to implement or maintain. Lastly, a business plan allows you to get prior approval for your implementation from your business process owners. The more agreement

there is on the business plan, the more likely your business process owners will accept the tailored implementation.

Many of the suggestions in this document assume you either know or have access to the Service Manager data model and work flows. This document does not describe the out-of-box data model or workflow in detail. If you need to change the out-of-box data model or workflow, see the Service Manager help.

What is tailoring?

Tailoring is any change to the out-of-box system data. Tailoring never changes the actual Service Manager code or program. Service Manager allows you to change almost any record value. The following are examples of tailoring changes you can make by changing the underlying record values.

- Change the look and operation of forms by changing the form definition in the format table.
- Change how fields are mapped in the back-end database by changing the value of the field mapping in the data dictionary.
- Change the workflow of a particular change by changing the values of the underlying phase and task records.
- Change what records a favorite or view displays by changing the value of the underlying query.

Tip: Since tailoring involves changing record values, you should create a back-up of your system data before tailoring your system. Having a backup available gives you the option to revert to a working system should your tailoring changes cause a system outage.

Chapter 2

Form tailoring

You will typically use one or more of the following tools and features to tailor forms to meet your business needs.

- Forms Designer controls
 - Combo Box
 - Comfill
 - Radio Button
 - Checkbox
 - Subform
 - Group
 - Frame
 - Text
 - Text Area
 - HTML Editor
- Dynamic View Dependencies (DVD)
- Link Records
- Global Lists

The following sections describe some of the use cases where you can get the most performance and benefit from these controls and tools.

What is a form?

A form is a graphical input used to access, add, change, and view records in a Service Manager table. Forms serve as the interface between the user and the underlying table. A form can be associated with only one Service Manager table; however, a table may have many associated forms. If you need a form to access data from multiple tables, you can create one or more subordinate forms and link to them using the Subform control. Service Manager stores form records in the format table.

You can create and modify forms with the Forms Designer (FD) utility. You can add or edit rules for processing form data with the Format Control (FC) utility.

Form tailoring use cases

The following use cases will help you decide the best form control to use given a particular situation.

- You want users to select values from a control with a list of values. See ["Selecting values from options in a control" \(on page 11\)](#).

- You want users to select values from a list of values on the form. See "[Selecting values from options on a form](#)" (on page 14).
- You want a user's prior selection to determine what other values are available. See "[Changing the options available based on prior user input](#)" (on page 16).
- You want to display values from another table or record. See "[Displaying related data](#)" (on page 20).
- You want to group logically associated items together. See "[Grouping related controls](#)" (on page 22).
- You want users to type text into a field. See "[Typing text into a control](#)" (on page 25).

The following use cases will help you improve your system performance and maintainability.

- You want to reduce the amount of system resources required during user log on. See "[Managing global lists](#)" (on page 28).
- You want to reduce the effort required to upgrade your tailored system. See "[Reducing upgrade conflicts](#)" (on page 29).

Selecting values from options in a control

Forms Designer offers two controls that allow users to select a value from a list of values.

- Combo Box
- Comfill

You can determine which of these two controls to use by the number of options there are to choose from.

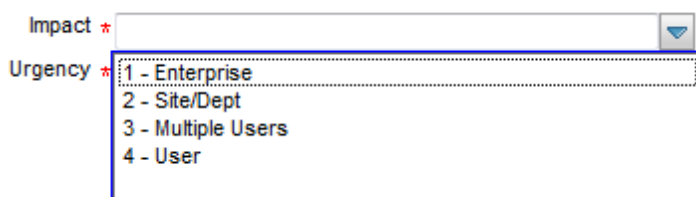
If there are a small number of options to choose from (for example, 32 or less options), then a Combo Box control is typically the most efficient control to use. See "[Combo Box](#)" (on page 11).

If there are large number of options to choose from (for example, 100 or more options), then a Comfill control is typically the most efficient control to use. See "[Comfill](#)" (on page 12).

Combo Box

The Combo Box control lists a set of values for a user to choose from in a drop down box. You determine the list of possible values by either manually entering the values for the control or providing a global list variable for the Value List attribute.

Example Combo Box control



The benefits of a Combo Box control are listed below.

- The user sees a complete list of values to choose from on the same form as the control.
- You can control both the labels users see for each option (the Display List attribute) and the actual value each option has when stored in the database (the Value List attribute).

The costs of a Combo Box control are listed below.

- The more options there are to display, the more system resources are required to display them.
- If the list of values comes from a global list, then the server must send the entire list to the client each time the form is displayed. The more list items there are, the more network bandwidth it takes to transmit the list to the client. Furthermore, the more users there are who can access the global list, the more often the server has to transmit the list to clients.
- If the list of values comes from a global list, then the server must periodically rebuild the list of values. The more items there are in the list, the more system resources it takes to rebuild the list.
- If the list of values comes from a hard-coded list, then each list item increases the size of the form. The larger a form is, the longer it takes for the client to initially display it.

The more options there are to display, the more costly a Combo Box is to your system's performance. There is no set number of values that determines when a Combo Box is less efficient than a Comfill, but in general a Combo Box is best used in the following circumstances.

- There are few enough options to display that the control can display them all without the user needing to scroll down to see more options.
- The list of options either does not change at all or changes infrequently.

If you have a lot of options to display, consider using a Comfill control instead where you can control how many options are displayed per page. Also, if you expect to frequently add or change the options available to the control, you may want to use a Comfill control linked to the data of another field. Using a linked Comfill control allows you to add or update the options as records and guarantees your users always see the most current list of options.

Comfill

The Comfill control lists a set of related records for a user to choose a value from. You determine the list of possible values by creating a link record entry for the Comfill control's Input field and specifying the related database field whose values you want to list in link record line entry. You can also use the link record to associate a Comfill control with a particular search form if you want to allow users to search for related records.

Example Comfill controls



The image shows two Comfill controls in a form. The first control is labeled 'Contact' with a red asterisk to its right. The second control is labeled 'Service Recipient' with a red asterisk to its right. Each control consists of a text input field followed by two small icons: a magnifying glass (search) and a circular arrow (refresh).

Example Comfill control associated with a search form

The screenshot shows a web form titled "SEARCH CONTACT INFORMATION". At the top, there is a navigation bar with "To Do Queue: My To Do List" and a button "Please select a search criteria." Below this is a toolbar with "Back", "Search", and "Fill" icons. The form contains the following fields:

- Contact Name: *
- Last Name:
- First Name:
- Employee ID:
- Title:
- Department:
- Company:
- Manager:

Example partial query in a Comfill control

The screenshot shows a Comfill control with two input fields. The first field is labeled "Contact *" and contains the letter "S". The second field is labeled "Service Recipient *" and is empty. Both fields have a small icon to their right.

Example query results from a partial query in a Comfill control

The screenshot shows the HP Service Manager interface. The main content area displays a table of contact records. The table has the following columns: Contact Name, Phone, Extension, Department, Company, Contact Id, and Full Name. The records are filtered by a partial query, showing only those contacts whose names start with "S".

Contact Name	Phone	Extension	Department	Company	Contact Id	Full Name
SANCHEZ, JOHN			Australia - Finance	advantage	John.Sanchez	John Sanchez
SANDERS, MATTHEW			Australia - HR & Adminis...	advantage	Matthew.Sanders	Matthew Sanders
SAYLE, RITA			Europe - Finance	advantage	Rita.Sayle	Rita Sayle
SCARFE, KRISTA			Asia - Research & Devel...	advantage	Krista.Scarfe	Krista Scarfe
SCHWARY, TRAVIS			South America - Finance	advantage	Travis.Schwary	Travis Schwary
SCOTT, REBECCA			Europe - Sales	advantage	Rebecca.Scott	Rebecca Scott
SENNA, ARORA			South America - Sales	advantage	Arora.Senna	Arora Senna
SERASINGHE, IRANGANIE			Europe - IT	advantage	Iranganie.Serasinghe	Iranganie Serasinghe
SERVICE DESK APPROVER		201	North America - Finance	advantage	sdapprover	SD Approver
SETH, JAN			South America - IT	advantage	Jan.Seth	Jan Seth
SEYMOUR, JACOB			North America - Wareho...	advantage	Jacob.Seymour	Jacob Seymour
SHELBY, RED			Australia - Finance	advantage	Red.Shelby	Red Shelby
SHERIN, LEILA			South America - HR & A...	advantage	Leila.Sherin	Leila Sherin
SIMMONS, KATIE			Australia - Sales	advantage	Katie.Simmons	Katie Simmons
SIMPSON, NANCY			Australia - Finance	advantage	Nancy.Simpson	Nancy Simpson
SLINGER, CHAT			Africa - Warehouse	advantage	Chat.Slinger	Chat Slinger
SMALLEY, JILL			South America - HR & A...	advantage	Jill.Smalley	Jill Smalley
SMITH, MARY			Africa - Finance	advantage	Mary.Smith	Mary Smith
SNAKE, PATRICIA			North America - HR & Ad...	advantage	Patricia.Snake	Patricia Snake
SORESI, LUCY			Asia - Warehouse	advantage	Lucy.Soresi	Lucy Soresi
SPEARS, CYNTHIA			Europe - HR & Administr...	advantage	Cynthia.Spears	Cynthia Spears
SPELBERG, LENE			Europe - Warehouse	advantage	Lene.Spielberg	Lene Spielberg
SPINX, AMBER			Europe - HR & Administr...	advantage	Amber.Spinx	Amber Spinx
SPRINTALL, CLARE			Europe - Warehouse	advantage	Clare.Sprintall	Clare Sprintall
STAFOLA, ZACK			Africa - Warehouse	advantage	Zack.Stafola	Zack Stafola

At the bottom of the table, there is a "Count Records" section showing "1 to 25 of 36" and a "Pages: 1 2" navigation. On the right, there is a "Show 25 records per page" dropdown menu.

The benefits of a Comfill control are listed below.

- The system only has to display one page's worth of options at a time. It takes less system resources to display a partial list than the full list of options. The longer the list of options is, the more efficient it is to display the list in pages.

- Users can type in partial option values to search for options that match the partial value. For example, typing an "S" in the Contact field and clicking the Fill button shows a list of contacts that begin with the letter S.
- You manage the list of possible values by updating the related records linked to by the fill. For example, as you add, update, or remove contact records any fill linking to contact records will automatically pick up these changes.

The costs of a Comfill control are listed below.

- You must create a link line entry and a query for each Comfill control.
- You must create a search form if you want the fill functionality to use it.

Since Comfills depend upon link queries, any inefficiencies in the underlying query will negatively affect your system performance. Following the best practices in ["How do I write a good query?" \(on page 32\)](#) will reduce this performance impact. In addition, keep in mind that it takes more effort to design and maintain a search form for each fill. You may want to limit search forms to circumstances where you want to provide users with more than one way to search for related records. If one field is sufficient to find the related records, you do not need to add a search form to the Comfill control.

In general, a Comfill is best used in the following circumstances.

- There are enough options to display that the control can display them on multiple pages.
- The list of options changes frequently and you want the control to always have access to the most recent changes.

If you have a limited number of options to display, consider using a Combo Box control instead where you can display all the options at once in a drop-down box. Also, if you do not expect the data to ever change or to change infrequently, consider using a global list to make the data available to any application that needs it.

Selecting values from options on a form

Forms Designer offers two controls that allow users to select a value from a list of options on the form.

- Radio Button
- Checkbox

You can determine which of these two controls to use by determining if the options on the form are mutually exclusive.

If you want users to select one value from a list of mutually exclusive values, then list each option as a Radio Button control. See ["Radio Button" \(on page 14\)](#).

If you want users to select a true or false value for one particular option, then use a Checkbox control for the option. See ["Checkbox" \(on page 15\)](#).

Radio Button

The Radio Button control lists one possible value for a set of mutually exclusive values that a user can choose from. You determine the list of possible values by creating multiple Radio button controls and giving them all the same Input attribute.

Example Radio Button control



Select Audience type for favorite

Selected Operator Selected Groups Selected Roles Everyone

The benefits of a Radio Button control are listed below.

- The user sees a complete list of values to choose from on the same form as the control.
- The user has a visual indicator for the currently selected value.
- You can control both the label users see for the option (the Caption attribute) and the actual value the option has when stored in the database (the Value attribute).

The costs of a Radio Button control are listed below.

- The more options there are to display, the more of the form's screen space is required to display them.
- You have to use a Frame or Group control to indicate that a list of Radio Button options are related.
- The list of values is only stored on the form and not in a global list or related record. This means you have to keep track of which forms use a particular list if you want to re-use the list in another form.
- The only way to change the list of values is to change the value of each individual Radio Button control.
- Each Radio Button option increases the size of the form. The larger a form is, the longer it takes for the client to initially display it.

A list of Radio Button options is very easy for your system to display since there are no link queries run or global lists to update at start-up. However, a list of Radio Button options requires more form space than using a single Combo box or Comfill control.

In general, a Radio Button control is best used in the following circumstances.

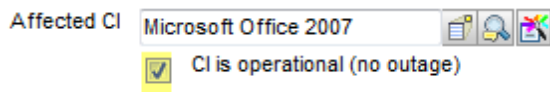
- There are few enough options to display that the user can see them all without needing to scroll to see more options.
- The list of options either does not change at all or changes infrequently.
- You want a visual indicator to highlight the currently selected value.
- You cannot select more than one value at a time with this control.

If you have more than a few options to display, consider using a Combo Box control instead to reduce the amount of screen space required to display the options. Also, if you expect to re-use the list of options in another form or in other circumstances, consider creating a global list so the options will be available for multiple forms. If you want users to be able to select more than one value from a list of values, consider creating a list of checkboxes.

Checkbox

The Checkbox control lists a true or false value for a single option. The list of possible values is always true or false for a Checkbox control.

Example Checkbox control



The benefits of a Checkbox control are listed below.

- The value of each Checkbox control is independent of other Checkbox controls.
- The user has a visual indicator for the currently selected value.
- The actual value of the control is always either true or false.
- You can control the label users see for the control (the Caption attribute)

The costs of a Checkbox control are listed below.

- You cannot use this control to provide a list of mutually exclusive options.
- You may have to use a Frame or Group control to indicate that a list of Checkbox options are related.

A Checkbox control is very easy for your system to display since there are no link queries run or global lists to update at start-up. The longer the Checkbox label is the more form space the control requires.

In general, a Checkbox control is best used in the following circumstances.

- The only options the user needs to choose between are true and false values.
- The values of one control are independent of the values in other controls.
- You want a visual indicator to highlight the currently selected value.

If you want users to select one value from a list of mutually exclusive values, use a set of Radio Button controls instead to ensure that only one value can be selected at a time.

Changing the options available based on prior user input

Forms Designer offers two methods for you to change what options are available based on prior user input.

- Use DVD select statements to change what options appear on a form
- Use recursive fills to change what records the user can select as options

You can determine which of these two methods to use by how you want to change the list of available options.

If you want to change what options are visible on form, then use one or more DVD select statements. See ["DVD select statements" \(on page 16\)](#).

If you want to change the list of records that a user can choose from, then use one or more recursive fills. See ["Recursive fills" \(on page 18\)](#).

DVD select statements

A DVD select statement changes what controls or values appear on a form based a triggering condition. You determine what controls or values the DVD select statement displays by assigning a

condition to each control or value. When the condition is true, the system displays the controls and values. When the condition is false, the system hides the controls and values.

Example DVD select statement where the display condition is false

Search Catalog Item Definitions

Name:

Display Name:

Type: ▼

Available to: ▼

Example DVD select statement where the display condition is true

Search Catalog Item Definitions

Name:

Display Name:

Type: ▼

Available to: ▼

Contains:

- Categories
- Items/bundles

The key benefits of a DVD select statement are listed below.

- The user only sees the controls and values when the appropriate condition applies.
- The user has a visual indicator that new options or controls are available.

The costs of a DVD select statement are listed below.

- The more conditions there are to evaluate, the more system resources it takes to display the form.
- You have to reserve part of the form's screen space to display hidden items.
- You must create a query for the DVD select statement condition.

- The only way to change the list of values is to change the value of each individual control of the DVD select statement.

Since DVD select statements depend upon queries for the condition, any inefficiencies in the underlying query will negatively affect your system performance. Following the best practices in "[How do I write a good query?](#)" (on page 32) will reduce this performance impact. In addition, the more conditions you have, the more of the form's screen space you will have to reserve for the controls hidden by the DVD statement.

In general, a DVD select statement is best used in the following circumstances.

- There are only a few conditions that change the options available on the form.
- The list of options either does not change at all or changes infrequently.

If you have a lot of conditional data to manage, consider using multiple Comfill controls with recursive fills instead. This allows you to restrict the options displayed in one Comfill control based on the selections in a previous Comfill control. Also, if you expect to frequently add or change the options available to the DVD select statement, you may want to use a Comfill control so that you can add or update the options as records.


Recursive fills

A recursive fill uses the value of one Comfill control to determine the available values of another Comfill control. You determine what values the recursive fills display by adding a link record line entry for each Comfill control on the form. When a link record includes the expression `$fill.recurse=true`, then the system evaluates each line of associated link record. You may also want to associate a search form or QBE format for each Comfill control in the recursive fill to make easier for users to select multiple values.

Example Comfill controls linked by recursive fills

Category * 
Area * 
Subarea * 

Example QBE formats allowing users to select a Category, Area, and Sub Area.

← Back  Skip | More ▾

Please Select a Category:

Category
complaint
incident
request for change
request for information


Please Select an Area:


Area	Category
access	incident
data	incident
facilities	incident
failure	incident
hardware	incident
performance	incident
security	incident


Please Select a Subarea:

Subarea	Area	Assignment Group	Company
authorization error	access		DEFAULT
data or file missing	access		DEFAULT
login failure	access		DEFAULT

Example completed recursive fill

Category * incident 

Area * access 

Subarea * authorization error 

The benefits of recursive fills are listed below.

- You can use a query to ensure that users only select valid combinations of values.
- Users can select values for related data in one process.
- The system only has to display one page's worth of options at a time. It takes less system resources to display a partial list than the full list of options. The longer the list of options is, the more efficient it is to display the list in pages.
- You manage the list of possible values by updating the related records linked to by the fill. For example, as you add, update, or remove contact records any fill linking to contact records will automatically pick up these changes.

The costs of recursive fills are listed below.

- You must create a link record entry and a query for each Comfill control.
- You must create a search form if you want the fill functionality to use it.
- Users can only select values that match the query for each Comfill control

Since recursive fills depend upon link queries, any inefficiencies in the underlying query will negatively affect your system performance. Following the best practices in ["How do I write a good](#)

[query?" \(on page 32\)](#) will reduce this performance impact. In addition, you must be very familiar with the Service Manager data model to create the necessary queries.

In general, a recursive fill is best used in the following circumstances.

- The value of one field determines the value another field can have. Typically, all of the fields come from a set of related records.
- The list of options changes frequently, and you want your Comfill controls to always have access to the most recent changes.

If you want to hide or display data on a form, use Combo Box controls and DVD select statements instead.

Displaying related data

Forms Designer offers two controls that allow users to display data from related tables and records.

- Combo Box
- Subform

You can determine which of these two controls to use by how you want users to interact with the related data.

If you want to use a control that allows users to select and interact with related data, then use a Combo Box control. See "[Combo Box" \(on page 11\)](#).

If you want use a control that protects any virtual-joined data by making it read-only, then use a Subform control. See "[Subform" \(on page 20\)](#).

Subform

The Subform control displays a separate form containing data from a different table than the table associated with the main form. A Subform control allows any controls on the subordinate form to use the link record data to perform a virtual join. For example, a text control or table control on the subordinate form can display related data from another table. You determine what related data to display by creating a link record entry for the Subform control's Input field and specifying the related database field whose values you want to display in link record line entry.

Example Subform control on the open interaction form

The screenshot shows a form with three fields. The first field is labeled 'Contact' and contains the text 'AARON, JIM'. The second field is labeled 'Service Recipient' and also contains 'AARON, JIM'. The third field is labeled 'Location' and contains 'North America'. Each of the first two fields has a small icon to its right, and the 'Location' field has a yellow highlight behind the text.

Example Subform control Input value

Subformat	
Property	Value
Name	
X	0
Y	12
Width	105
Height	2
Visible	<input checked="" type="checkbox"/>
Visible Condition	
Tab Stop	0
Format	SD.contacts.location.vj
Virtual Join	<input checked="" type="checkbox"/>
Display Blank	<input checked="" type="checkbox"/>
Display Using Table	<input checked="" type="checkbox"/>
Input	callback.contact

Example link record line entry for the callback.contact field

Field (From/Source):	File (To/Target):	Format (To/Target):	Field (To/Target):
callback.contact	contacts		contact.name
Comment:			
Query:	\$query		
QBE Format:		Structured Array Name:	

◆ Expressions ◆ Javascript

```
$fill.search.format="contacts.search"
if (not null(callback.contact in $File)) then ($query="contact.name#callback.contact in $File") else ($query="true")
if ($G.multi and not null(company in $File)) then ($query=$query+" and company=\""+evaluate(company in $File)+"\"")
```

The benefits of a Subform control are listed below.

- You can display related record data from any other source in the system.
- You can display related records that have one to many relationships. For example, from one input, the contact name, you can use a Subform control to display many related fields such as the contact's location, address, or telephone number.
- You can provide link record data to controls that do not normally have link records such as Text or Table controls.
- Any information on a Subform control is automatically read-only when it is part of a virtual join.

The costs of a Subform control are listed below.

- You must create a separate link line entry and a query for each Subform control.
- You must create and maintain a subordinate form for the linked controls.
- You cannot use the virtual join option if you want users to interact with the controls on the subordinate form. This typically requires using either global lists or hard-coded values for any Combo Box controls on the subordinate form.

Since Subform controls depend upon link queries, any inefficiencies in the underlying query will negatively affect your system performance. Following the best practices in ["How do I write a good query?" \(on page 32\)](#) will reduce this performance impact. In addition, you must be very familiar with the Service Manager data model to create the necessary queries.

In general, a Subform control is best used in the following circumstances.

- You need to display data from a different table than the one associated with the main form
- You want the related information to be displayed in read only controls

If you want users to be able to select or interact with the related data, either use a Comfill control or do not enable the Virtual Join option for the Subform control. See ["Comfill" \(on page 12\)](#) for a description of Comfill costs and benefits.

Grouping related controls

Forms Designer offers two controls that allow users to group related sets of controls in a container.

- Group
- Frame

You can determine which of these two controls to use by which client you want the container to appear in.

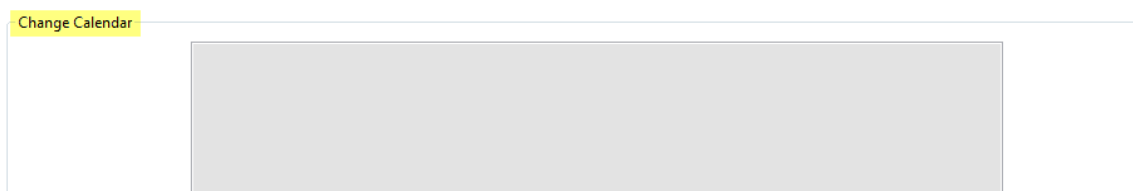
If you want Web client users to be able to collapse and expand the container, then use a Group control. See ["Group" \(on page 22\)](#).

If you only want Windows client users to see a container with line borders, then use a Frame control. See ["Frame" \(on page 24\)](#).

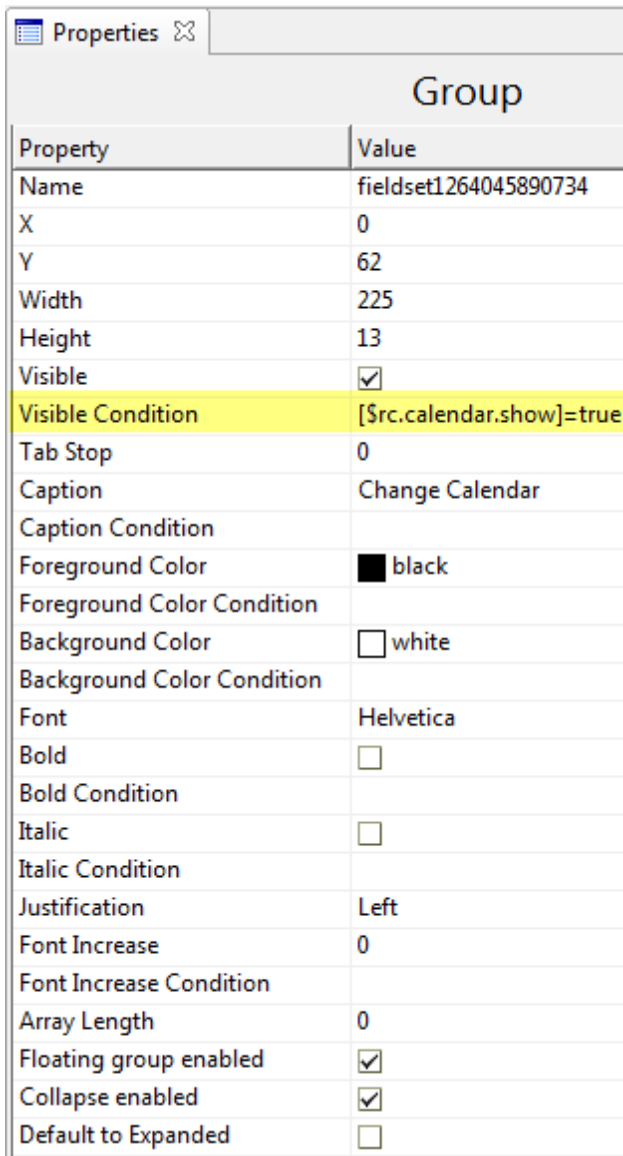
Group

The Group control provides a container with a text label to logically group items and controls. Any control you place within a Group control inherits the visibility condition of the Group.

Example Group control



Example Visible Condition for a Group control



Property	Value
Name	fieldset1264045890734
X	0
Y	62
Width	225
Height	13
Visible	<input checked="" type="checkbox"/>
Visible Condition	[\$src.calendar.show]=true
Tab Stop	0
Caption	Change Calendar
Caption Condition	
Foreground Color	<input checked="" type="checkbox"/> black
Foreground Color Condition	
Background Color	<input type="checkbox"/> white
Background Color Condition	
Font	Helvetica
Bold	<input type="checkbox"/>
Bold Condition	
Italic	<input type="checkbox"/>
Italic Condition	
Justification	Left
Font Increase	0
Font Increase Condition	
Array Length	0
Floating group enabled	<input checked="" type="checkbox"/>
Collapse enabled	<input checked="" type="checkbox"/>
Default to Expanded	<input type="checkbox"/>

The benefits of a Group control are listed below.

- You can make the container a section that collapses and expands in the Web client.
- The container has a text label to clearly identify the purpose of the controls within the container.
- The controls within the container inherit the Visible Condition property of the Group. The controls within the group are only visible when the group itself is visible.
- If you move or delete a Group control, all the controls within the container are also moved or deleted.

The costs of a Group control are listed below.

- You have to update the text label should you ever change the controls within the container.
- You have to localize the label if you want to make the form available in other languages.

In general, a Group control is best used in the following circumstances.

- You want a visual indicator to highlight that a group of controls are related.
- You want to allow users to efficiently manage the available form space in the Web client.
- You want to manage the visibility settings of a group of controls from one place.






If you want to change the appearance of the line around the container, then use a Frame control instead. However a Frame control does not allow users to collapse and expand the section in the Web client.

Frame

The Frame control provides a container with a beveled line surrounding the items and controls in the container. Any control you place within a Frame control inherits the visibility condition of the Group. Since the Web client does not display the line surrounding the Frame control, it almost always better to use a Group control rather than a Frame control.

Example Frame control in Windows client

Conversion Rate Information

Root Currency Code:	<input type="text" value="BRL"/>	 
Date:	<input type="text" value="04/13/06 18:00:00"/>	
Exchange Rate:	<input type="text" value="0.38714"/>	
Currency Code:	<input type="text" value="AUD"/>	 






1 Root Currency will buy rate other currency e.g.

Root Currency Code:	BRL
Exchange Rate:	0.38714
Currency Code:	AUD

Means that 1 BRL will buy 0.38714 AUD.

Example Frame control in Web client

Conversion Rate Information

Root Currency Code:	<input type="text" value="BRL"/>	 
Date:	<input type="text" value="04/13/06 18:00:00"/>	
Exchange Rate:	<input type="text" value="0.38714"/>	
Currency Code:	<input type="text" value="AUD"/>	 

1 Root Currency will buy rate other currency e.g.

Root Currency Code:	BRL
Exchange Rate:	0.38714
Currency Code:	AUD

Means that 1 BRL will buy 0.38714 AUD.

The benefits of a Frame control are listed below.

- The container has a border to indicate that the controls are within a container.
- The controls within the container inherit the Visible Condition property of the frame. The controls within the frame are only visible when the frame itself is visible.

The costs of a Frame control are listed below.

- The line surrounding the Frame control is not visible in the Web client
- Users cannot collapse and expand the container in the Web client

The Frame control has less functionality than the Group control. In most cases, you should use a Group control rather than a Frame control.

Typing text into a control

Forms Designer offers three controls that allow users to type text into a control.

- Text
- Text Area
- HTML Editor

You can determine which of these controls to use by the amount and formatting of the text you want users to type.

If you want users to type a single line of unformatted text, then use a Text control. See ["Combo Box" \(on page 11\)](#).

If you want users to type in multiple lines of unformatted text, then use a Text Area control. See ["Comfill" \(on page 12\)](#).

If you want users to type in any amount of HTML-formatted text, then use an HTML Editor control. See ["HTML Editor" \(on page 27\)](#).

Text

The Text control allows users to type in or view a single line of unformatted text. You can also make a Text control read-only to display the value of a particular field or if the control is part of a Subform control virtual join.

Example Text control

Description:

* Use this form to request accounts and access to the Accounts Payable application.

The benefits of a Text control are listed below.

- You can make the control read-only to display data you do not want users to change, such as a record ID number.
- RDBMS tools can typically query Text control fields as long as you map the field to a simple character data type in the back-end RDBMS.
- Third-party reporting tools can typically read Text control fields as long as you map the field to a simple character data type in the back-end RDBMS.

- The user does not have to apply any text formatting to the value typed.
- If the control is editable, users can update the text value at any time.

The costs of a Text control are listed below.

- The content of the control is displayed on one line of text. The database definition mapping of the fields determines the width of the line. For example, if the input field is mapped to a VARCHAR(255) column, then a user can only type 255 characters of text.
- The control uses the character encoding of the back-end RDBMS to determine how to store character data. If users type in text using a character encoding that your system does not support, the data will likely become corrupted.
- It is difficult to validate that users type in meaningful values into the control.
- There is no way for users to format the text they type into the control.
- The longer the width of the Text control the more form space the control requires.

In general, a Text control is best used in the following circumstances.

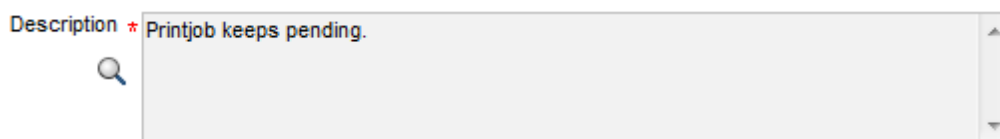
- The user only needs to type in or view a single line of text.
- You want the option to make the text read-only. For example, the text is a unique ID or part of a virtual join in a Subform control.
- You do not need to format the text.

If you want users to type multiple lines of text, use a Text Area controls instead. If you want users to be able to format the text they type in, use an HTML Editor control instead.

Text Area

The Text Area control allows users to type in or view multiple lines of unformatted text. You can also make a Text Area control read-only to display the value of a particular field or if the control is part of a Subform control virtual join.

Example Text Area control



The benefits of a Text Area control are listed below.

- Users have multiple lines in which to type or view text.
- You can make the control read-only to display data you do not want users to change, such as the description of an open interaction.
- RDBMS tools can typically query Text Area control fields as long as you map the Text Area's input field to a multi-row array table. See ["Multi-row array table" \(on page 41\)](#).
- Third-party reporting tools can typically read Text Area control fields as long as you map the Text Area's input field to a multi-row array table. See ["Multi-row array table" \(on page 41\)](#).

- The user does not have to apply any text formatting to the value typed.
- If the control is editable, users can update the text value at any time.

The costs of a Text control are listed below.

- You must choose an array mapping strategy in which to store the Text Area control data. See ["Array mapping options" \(on page 37\)](#).
- The control uses the character encoding of the back-end RDBMS to determine how to store character data. If users type in text using a character encoding that your system does not support, the data will likely become corrupted.
- It is difficult to validate that users type in meaningful values into the control.
- There is no way for users to format the text they type into the control.
- The longer the width of the Text Area control the more form space the control requires.

In general, a Text Area control is best used in the following circumstances.

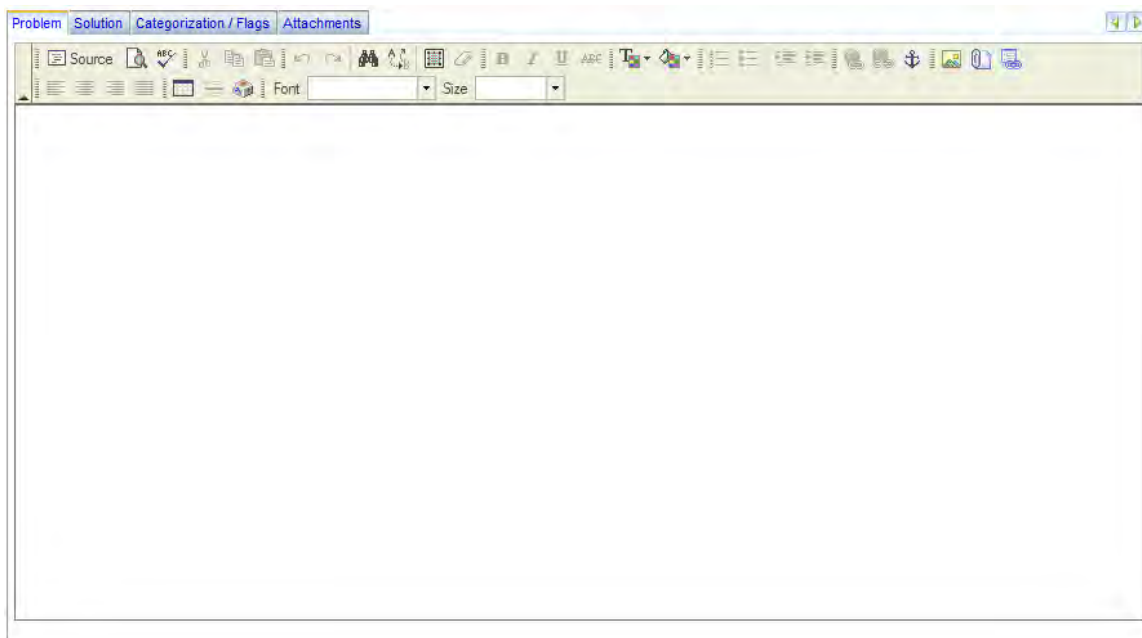
- The user needs to type in or view multiple lines of text.
- You want the option to make the text read-only. For example, the text is a part of a closed record or part of a virtual join in a Subform control.
- You do not need to format the text.

If you want users to type a single line of text, use a Text controls instead. If you want users to be able to format the text they type in, use an HTML Editor control instead.

HTML Editor

The HTML Editor control allows users to type in and format multiple lines of text for display in an HTML Viewer control. Unlike other text input controls, if you do not want users to change the HTML text, you must display it in an HTML Viewer control.

Example HTML Editor control for contributing a problem/solution knowledge article



The benefits of an HTML Editor control are listed below.

- Users can format multiple lines of text for display with the built-in editing tools or with HTML tags.
- Users can spell check the text.
- RDBMS tools can typically query HTML Editor control fields as long as you map the HTML Editor's input field to a multi-row array table. See ["Multi-row array table" \(on page 41\)](#).
- Third-party reporting tools can typically read HTML Editor control fields as long as you map the HTML Editor's input field to a multi-row array table. See ["Multi-row array table" \(on page 41\)](#).
- Users can update the text value at any time.

The costs of an HTML Editor are listed below.

- You must use a separate HTML Viewer control to display a read-only version of data you do not want users to change, such as an approved knowledge article.
- You must choose RDBMS mapping that has sufficient space for potentially large HTML input. Typically, an array mapping strategy is best for storing HTML Editor control data as there is no risk of data truncation when users type large amounts of text. See ["Array mapping options" \(on page 37\)](#).
- The control uses the character encoding of the back-end RDBMS to determine how to store character data. If users type in text using a character encoding that your system does not support, the data will likely become corrupted.
- It is difficult to validate that users type in meaningful values into the control.
- You must use a separate HTML Viewer control to provide a read-only version of the text.
- The longer the width of the HTML Viewer control the more form space the control requires.

In general, an HTML Editor control is best used in the following circumstances.

- The user needs to type in and format multiple lines of text. For example, the text is a knowledge document.
- You want the option to spell check the text.

If you do not want users to format the text, use either a Text or Text Area control instead. Likewise, if you need to display the value of another text field in the system, use either a Text or Text Area control.

Managing global lists

Global lists provide an efficient way to display frequently-used list values by storing the list in system memory as a global variable. All applications can access the global list variable as needed without needing to query the database each time a control needs to display the list. However, the more items there are in a global list, the more costly the list is to system performance.

A global list has the following performance benefits.

- Since a global list is stored in system memory, there is no need to query the back-end database for a list of values. Using a global list therefore reduces the amount of input and output traffic to the RDBMS.

- Since a global list is available to all users logged into the system, there is no need to create conditional queries to grant access to the information. Using a global list therefore potentially reduces the number of queries needed to access a particular set of information.

When managing global list, keep in mind the following performance costs.

- The server must send the entire list to the client each time the form is displayed. The more list items there are, the more network bandwidth it takes to transmit the list to the client.
- The more users there are who can access the global list, the more often the server has to transmit the list to clients.
- The server must periodically rebuild the list of values. The more items there are in the list, the more system resources it takes to rebuild the list

If you want to maximize your system performance, use the following guidelines to determine when to use global lists.

If there are a small number of options to choose from (for example, 32 or less options), then displaying a global list in a Combo Box control is typically the most efficient strategy. Since the list of options is small, you gain a performance improvement by skipping a query to the back-end database that a Comfill control would require. See "[Combo Box](#)" (on page 11).

If there are large number of options to choose from (for example, 100 or more options), then displaying a list of values from a Comfill control linked to other records in the database is typically more efficient than a global list. Since a global list must always send the entire list of options to the client, the Comfill control has better performance because it can send one page worth of options to the client at a time. In addition, it is more efficient to add, update, or remove the related records a Comfill displays than it is to rebuild an entire global list each time you want to modify the list of values. See "[Comfill](#)" (on page 12).

Removing unused global lists

You should periodically review your global lists to determine if any of them are unused and can be deleted. Since the server periodically rebuilds each global list, you will see a performance improvement if you remove any unused global lists from your system. Removing unused lists frees up system resources and memory that your system can use for other purposes.

Tip: You can archive and purge any unused global lists for later use. Should you find you require the global list you can simply import the global list back into the system from the archive unload.

Reducing upgrade conflicts

The Service Manager application upgrade process allows you to add new features and functions to your existing system. An upgrade conflict occurs when the upgrade utility determines there are multiple versions of the same out-of-box component. For every conflict the upgrade utility identifies, you must manually reconcile between the HP-version of an out-of-box component and the customer-modified version of the same component. This reconciliation process is time-consuming and often delays the adoption of new features and functions because you cannot use the new features and functions until after you have reconciled the conflicts.

You can use the following guidelines to reduce the number of conflicts you have to resolve during an application upgrade.

- Do not directly modify the out-of-box forms. Instead, create your own copies of the forms, give them unique names (such as adding a company abbreviation to the front of each form name), and make any modifications to the copied forms.
- Do not directly modify the out-of-box scripts in the script library. Instead, create your own script library with copies of any out-of-box functions, give them unique names (such as adding a company abbreviation to the front of each script name), and make any modifications to the copied scripts.
- Keep an audit log of any changes you make to the out-of-box Document Engine components such as processes, states, and objects. You can then use your audit log to determine how to merge any changes to Document Engine components.

Making your own copies of forms and scripts prevents you from having any conflicts on these items and allows you to review any new functionality the upgrade adds without changing your existing system. For example, if you create a copy of the out-of-box form `SD.open.interaction` called `HP.SD.open.interaction`, then any upgrade to the out-of-box form does not create a conflict with your custom form. Likewise, you can compare the functionality of the new out-of-box form to your custom form and determine if it has any features or functions you want to include in your custom form.

While keeping an audit log will not prevent upgrade conflicts, it will reduce their impact if you can quickly determine what functionality your changes support and whether the HP-version of the Document Engine components also support this functionality. At a minimum, your audit log should list what Document Engine components you changed and what the out-of-box state was. You may also want to list why you changed the component or what applications, forms, workflow your change supports.

Chapter 3

Queries

A query is a request to the database (RDBMS) that generates a reply. You can author queries by using RAD (the HP Service Manager internal system language) or JavaScript. In either case, the query must be translated to the native language of the RDBMS and the reply translated back into the original requesting language. See the HP Service Manager Programming Guide for information on the HP Service Manager system language or JavaScript.

In order to write an effective query you must be familiar with the logical structure of the Service Manager tables and fields included in the query as they are defined in the system's database dictionary. It is sometimes also helpful to know the architecture of the tables and fields as they are defined in the database; however this information is often restricted to database administrators (DBAs) in enterprise organizations.

How do I write a good query?

An ideal Service Manager query has the following characteristics.

- Runs against indexed columns to avoid full table scans. See ["Indexing" \(on page 32\)](#).
- Uses operators like equal to (=) that map well to SQL to avoid full table scans. See ["Query operators" \(on page 33\)](#).
- Avoids RAD functions like index() that are likely to cause full table scans. See ["Function usage" \(on page 33\)](#).
- Ensures that variables always have a non-null value. See ["Variable usage and null values" \(on page 34\)](#).

Indexing

An ideal query runs against indexed columns. Querying against indexed columns increases the speed at which Service Manager can receive data. However, adding too many indexes adds a performance cost to inserting (adding) new records as well as updating or deleting existing ones. Indexes also require additional storage space.

Your database administrator will weigh the potential benefit of indexing your queries against the cost of increased write times (insertions and updates). Typically, your database administrator may want to index any query that many people access or that gets run frequently. You may want to index tables that are part of the following Service Manager frequently-used queries:

- Public favorites
- Commonly-used forms containing Format controls
- Fills
- Drop-downs (combo boxes)
- Global lists
- Background processes

Query operators

Since Service Manager must translate queries into native RDBMS language, you should try to use operators that map well to native RDBMS languages. In general queries that use the EQUAL TO operator (=) are the most efficient and easiest to translate.

Operator	Example query	Sample SQL statement
EQUAL TO (=)	<code>contact.name="AARON, JIM"</code>	<code>SELECT "CONTACT_NAME" FROM CONTACTSM1 WHERE "CONTACT_NAME"='AARON, JIM';</code>

As the example illustrates, an EQUAL TO query often directly translates into a SQL query as part of a WHERE clause. In addition, queries with EQUAL TO operations are also the most likely to use and benefit from table indexes.

If you cannot use an EQUAL TO operator in a query, the following operators will also in many cases facilitate direct translation into SQL queries.

Operator	Example query	Sample SQL statement
Starts with (#)	<code>contact.name#"AA"</code>	<code>SELECT "CONTACT_NAME" FROM CONTACTSM1 WHERE "CONTACT_NAME" LIKE 'AA%';</code>
LIKE	<code>contact.name like "A*RON*"</code>	<code>SELECT "CONTACT_NAME" FROM CONTACTSM1 WHERE "CONTACT_NAME" LIKE 'A%RON%';</code>
ISIN	<code>contact.name isin {"AARON, JIM", "ARMSTRONG, TRACY"}</code>	<code>SELECT "CONTACT_NAME" FROM CONTACTSM1 WHERE "CONTACT_NAME" IN ('AARON, JIM', 'ARMSTRONG, TRACY');</code>

In these examples, the LIKE operator directly translates into a SQL query as part of a WHERE clause. The ISIN operator translates into an IN predicate. The STARTS WITH operator (#) translates into a LIKE predicate using a wildcard character.

Function usage

Any query that contains a RAD system language function such as `index()` can potentially cause a full table scan because it may not translate to SQL. Generally, you want to limit the use of functions in queries to avoid degrading your system's performance every time the query is run. If you anticipate a query will run frequently, consider rewriting the query to use operators instead of functions. The most typical high-use queries are those in the following areas:

- Favorites
- Views
- Links
- Stored queries.

For example, suppose you want to create a view that displays closed high priority incidents with activity from a particular operator named Manager. You could create this query by using the `index()` function.

Inefficient query example: index() function

View definition field	View definition value
Table	probsummary
Query	(flag=false) and (priority.code<="2") and index("Manager", assignee.name)>0

As written this is an inefficient query. However, you can rewrite this query to use simple operators and prevent unnecessary table scans. For example, the following query accomplishes the same objective.

Efficient query example: using simple operators

View definition field	View definition value
Table	probsummary
Query	(flag=false) and (priority.code<="2") and (assignee.name like "*Manager*")

If you cannot rewrite the query (for example, if you are using an exclusive feature of the function), you can minimize its performance impact by limiting access to the query.

Variable usage and null values

Anytime you use a custom variable in query you should ensure that it does not have a null value. Null values in query variables can cause full table scans which lower your system's performance. You can avoid null values in custom variables by using the following methods:

- Use the nullsub() function to replace the null value with a default value.
- Replace a custom variable with system variable. System variables obtain a value when a user logs in to the system or accesses a particular application.

For example, suppose you create a custom variable \$name to store a contact's name. You could provide a default value for the variable with the following expression.

```
$name=nullsub($name, contact.name in $L.file)
```

In this case, the default value will become the contact name listed in any file that uses this contact.name field. If query does not have access to the contact.name field value, you may consider hard-coding the default value, but the utility of such a hard-coded value may be limited.

Alternatively, you may want to use an existing system variable such as \$lo.user.name or \$lo.ufname to provide the name of current operator.

Where are queries used?

There are several places within Service Manager where you can tailor queries. You should review any custom queries in these areas to ensure they are as efficient as possible:

- Background processes, See ["Queries in background processes" \(on page 35\)](#).
- Global lists. See ["Queries in global lists" \(on page 35\)](#).

- Links. See ["Queries in links" \(on page 35\)](#).
- Format controls. See ["Queries in Format Control" \(on page 35\)](#).

Service Manager runs these queries frequently because they are either part of a scheduled task or common user actions trigger them.

Queries in background processes

Most background processes issue queries to your RDBMS at regularly scheduled intervals. While these queries already have proper indexes in an out-of-box system, you may consider limiting how often these queries run or eliminating them entirely if your system does not use them. You should review the following background processes to determine how often your system needs to run them:

- Agent (not started in an out-of-box system)
- Marquee (not started in an out-of-box system)
- Lister

You can control how often these background processes run from the Service Manager schedule record. You can set these processes to run infrequently or never run at all if they are not needed. See the Service Manager help for more information.

Queries in global lists

Because Service Manager often builds global lists at startup, the total number of global lists you have determines how long your system spends generating these lists. If you have custom global lists, you should verify that they are still in use and if not clear the Build List at Startup option. Reducing the number of global lists you build reduces the number of queries your system has to run and will result in improved login performance.

In addition, you should verify that your custom global list query returns a useful number of results. The more items there are in a global list the more memory a Service Manager client requires to display them. Consider tuning your global list queries to return a manageable number of results to improve your system's performance.

Queries in links

Because link queries are run frequently, it is essential that you index them correctly and write the queries with the most restrictive fields first in the search order. You can also reduce the amount of computation needed to create links results by minimizing the use of pre- and post-link expressions.

Queries in Format Control

You can use Format Control to specify the conditions that trigger a query to run. It is most efficient if you only run queries when they are needed. For example, if you plan to use query results during the addition or update of a record, then set the Format Control conditions so that the query only during add or update operations. The same suggestion applies to any custom Format Control calculations and subroutines you use. Verify that you need the custom query and that it runs only when it is needed. By restricting the conditions under which a Format Control query runs you minimize unnecessary calls to the RDBMS and keep system resources free for other operations.

Chapter 4

Database (RDBMS) mapping

The Service Manager database dictionary (DBDICT) contains a logical definition of the tables and columns that make up your relational database management system (RDBMS). Each DBDICT record maps a Service Manager logical file to one or more RDBMS table and a Service Manager logical field to an RDBMS column. In addition, database dictionary records specify what data type the system uses to store data in the RDBMS.

How do I create good RDBMS mappings?

The following tips will help you create good mappings between your Service Manager DBDICT records and your RDBMS.

- Use the DBDICT utility to create and update RDBMS mappings. See "[Database dictionary \(DBDICT\) utility](#)" (on page 36).
- Provide Service Manager with create, alter, and drop table rights. See "[Mapping rights](#)" (on page 36).
- Choose an array mapping that meets your data access needs. See "[Array mapping options](#)" (on page 37).
- Only change data type mappings to increase the column width and between character data types. See "[Valid data type changes](#)" (on page 48).
- Know when your database dictionary changes will cause Service Manager to add, update, or delete RDBMS mappings. See "[When are RDBMS mappings added or updated?](#)" (on page 49).

Database dictionary (DBDICT) utility

The database dictionary (DBDICT) utility allows you to view, create, and update the Service Manager logical files, fields, and keys that your system maps to tables, columns, and indexes in your relational database management system (RDBMS). The key benefits of the DBDICT utility over other database management utilities are:

- You can define or change how Service Manager maps array data
- You can create database definition language (DDL) for any additions and changes in environments where Service Manager does not have access rights to your RDBMS
- You can view and create mappings to null tables in environments where Service Manager does not have access rights to your RDBMS
- You can create mappings by importing columns from your RDBMS

See the Service Manager help for more information about using the DBDICT utility to create database dictionary records.

Mapping rights

As of version 7.11, your Service Manager system automatically determines if it has the rights to create, alter, and drop tables. If it has such rights, then Service Manager will automatically update

your RDBMS anytime you make changes in a DBDICT record.

If it does not have these rights, Service Manager creates a null table mapping to act as a placeholder until you can update the DBDICT record with the actual table and column mappings. In addition, Service Manager offers the option to export DBDICT record changes as database definition language (DDL) so that a database administrator (DBA) can review and manually make the desired changes to the RDBMS. After the DBA has added the needed tables and columns, you can import the actual table and column mappings into the DBDICT record.

Array mapping options

In a database dictionary record, an arrays contain one or more elements of a single data type or a structure field. If an array contains a structure element it is referred to as an array of structure. Arrays of structure contain one or more fields that can be of any data type.

You can map your array data in the following five formats.

- ["Field in main table" \(on page 37\)](#)
- ["Field in alias table \(Deprecated\)" \(on page 39\)](#)
- ["BLOB in main table" \(on page 39\)](#)
- ["BLOB in alias table \(Deprecated\)" \(on page 40\)](#)
- ["Multi-row array table" \(on page 41\)](#)

The array mapping format you choose determines the following factors about your array data.

- Can your RDBMS directly query your array data?
- Can third-party tools report against your array data?
- Is your array data stored with other fields in one or more main tables?
- Is your array data stored separately in one or more dedicated alias tables?
- Can your array support complex arrays of structure?
- How efficiently does your system retrieve and write data to your array?

See the following sections for a description of each array mapping option.

Field in main table

The field in main table format maps a Service Manager array field as a character large object (CLOB) within a main table. This array format is good for situations where the following conditions apply:

- You want to be able to use third-party tools to run reports against your array data
- You want the option to store very large array elements
- You do not want to re-map an existing array

The following table summarize the advantages of using this array mapping format.

Advantages

Advantage	Description
Third-party tools can report against the array	Service Manager stores array data as a character string that third-party reporting tools can read. The system separates each element of the array with a \n character delimiter.
The array can store very large array elements	Since the array is mapped to a character large object column, the column can occupy a large amount of space. Most RDBMS vendors support at least 2 gigabytes of large object data in such a column.
The array has access to other fields in the main table	There is a greater probability that other fields reside in the same main table as the array. Having all or most of your fields in a main table reduces the number of SELECT and JOIN statements your system has to make to fetch a record. The less RDBMS traffic required, the better performance your queries and reports will have.
There is no need to re-map if you use the default array mapping format	The field in main table format is Service Manager's out-of-box array mapping format. Unless someone has changed the default array mapping format, all your array data will automatically be in this format.

The following table summarize the disadvantages of using this array mapping format.

Disadvantages

Disadvantage	Description
The RDBMS cannot directly query the array	Most RDBMS vendors cannot query a column containing a reference to character large object data.
Any query against the array is inefficient	The RDBMS cannot directly query this array format because the column is mapped to a large object data type. Instead, the RDBMS must scan the table to return all rows of the large object column, and then forward the results to Service Manager for evaluation.
The array shares table space with other fields in the table	The array does not have a dedicated table in which to store data. Each RDBMS vendor has its own rules to determine the maximum number of columns allowed in a table, the maximum size of any row, and the maximum table size. Any array in a main table will consume some of these RDBMS resources. If an array requires more resources than any one main table can provide, Service Manager will move the array to another main table.

To set up an array in a field in main table mapping, you must create the following mappings in a database dictionary record.

Database dictionary element type	RDBMS mapping required
Array data type	Not mapped
Sub-field of the array	Mapped as a CLOB column in a main table
RC flag	False

Field in alias table (Deprecated)

The field in alias table format maps a Service Manager array field as a character large object (CLOB) within an alias table. **This array format is deprecated.** In earlier versions, this format was intended to support the following features:

- Your RDBMS vendor could only support one "long" data type per table (This is still a restriction on Oracle databases but can be worked around by using CLOB and BLOB data types instead.)
- You wanted the option to store very large array elements
- You wanted to be able to use third-party tools to run reports against your array data

You should use one of the following array formats instead of the field in alias table format.

- Field in main table
- BLOB in main table
- Multi-row array table

BLOB in main table

The BLOB in main table format maps a Service Manager array field as a binary large object (BLOB) within a main table. This array format is good for situations where the following conditions apply:

- Your array data is in a binary format (such as attachment data or RAD language expressions)
- You want the option to store very large array elements
- You want to support complex arrays of structure
- You do not need to use third-party tools to run reports against your array data

The following table summarize the advantages of using this array mapping format.

Advantages

Advantage	Description
Preservation of any native binary data	The RDBMS does not have to do any code page translations to store binary data in the array. Since there is no translation from a binary format to a character-based format, there is no chance that the translation will result in data corruption such as high-bit characters becoming question marks.
The array can store very large array elements	Since the array is mapped to a binary large object column, the column can occupy a large amount of space. Most RDBMS vendors support at least 2 gigabytes of large object data in such a column.

Advantage	Description
The array has access to other fields in the main table	There is a greater probability that other fields reside in the same main table as the array. Having all or most of your fields in a main table reduces the number of SELECT and JOIN statements your system has to make to fetch a record. The less RDBMS traffic required, the better performance your queries and reports will have.
Support for complex arrays of structure	This array format supports complex arrays of structure such as an array of structure within an array of structure.

The following table summarize the disadvantages of using this array mapping format.

Disadvantages

Disadvantage	Description
Third-party tools cannot report against the array	Service Manager stores array data in a proprietary binary format that third-party reporting tools cannot read.
The RDBMS cannot directly query the array	Most RDBMS vendors cannot query a column containing a reference to character large object data.
Any query against the array is inefficient	The RDBMS cannot directly query this array format because the column is mapped to a large object data type. Instead, the RDBMS must scan the table to return all rows of the large object column, and then forward the results to Service Manager for evaluation.

To set up an array in a field in main table mapping, you must create the following mappings in a database dictionary record.

Database dictionary element type	RDBMS mapping required
Array data type	Mapped as a BLOB column in a main table
Sub-field of the array	Not mapped
RC flag	True

BLOB in alias table (Deprecated)

The BLOB in alias table format maps a Service Manager array field as a binary large object (BLOB) within an alias table. **This array format is deprecated.** In earlier versions, this format was intended to support the following features:

- Your RDBMS vendor could only support one "long" data type per table (This is still a restriction on Oracle databases but can be worked around by using CLOB and BLOB data types instead.)
- You want the option to store very large array elements
- You want to support complex arrays of structure
- You do not need to use third-party tools to run reports against your array data

You should use one of the following array formats instead of the BLOB in alias table format.

- Field in main table
- BLOB in main table
- Multi-row array table

Multi-row array table

The multi-row array table format maps a Service Manager array field as a character string within one or more rows in an alias table. This array format is good for situations where the following conditions apply:

- You want to be able to run RDBMS queries on your array data
- You want to be able to use third-party tools to run reports against your array data
- You want the option to store very large array elements
- You want to support simple arrays of structure

The following table summarize the advantages of using this array mapping format.

Advantages

Advantage	Description
The RDBMS can directly query the array	In this array format, Service Manager stores each array element as a simple character string in one or more rows. This format allows RDBMS queries to read each individual array element without needing to forward the query to Service Manager for evaluation.
Third-party tools can report against the array	Service Manager stores each element of the array in its own separate row. Third-party reporting tools can read the row and extract the element value.
There is no risk of data truncation for new array elements	Each array element occupies as many rows as necessary to store the full element. This means that the length of the array element is not limited by the length of the column. Any array element that exceeds the column length for one row continues onto the next row. This feature prevents Service Manager from truncating array element data.
Support for simple arrays of structure	This array format supports simple arrays of structure such as an array of structure within an array of structure.

The following table summarize the disadvantages of using this array mapping format.

Disadvantages

Disadvantage	Description
Service Manager must maintain a separate table	In order to fetch a record, the system must make SELECT and JOIN statements between the main and alias tables. These statements require more RDBMS traffic and can lower your RDBMS performance.
The array uses more table rows	Each array element will require one or more table rows. Each

Disadvantage	Description
	time you fetch an array record, the system must select multiple rows from an alias table. The more rows the system has to select, the more costly your record retrieval is in terms of RDBMS performance.

To set up an array in a field in main table mapping, you must create the following mappings in a database dictionary record.

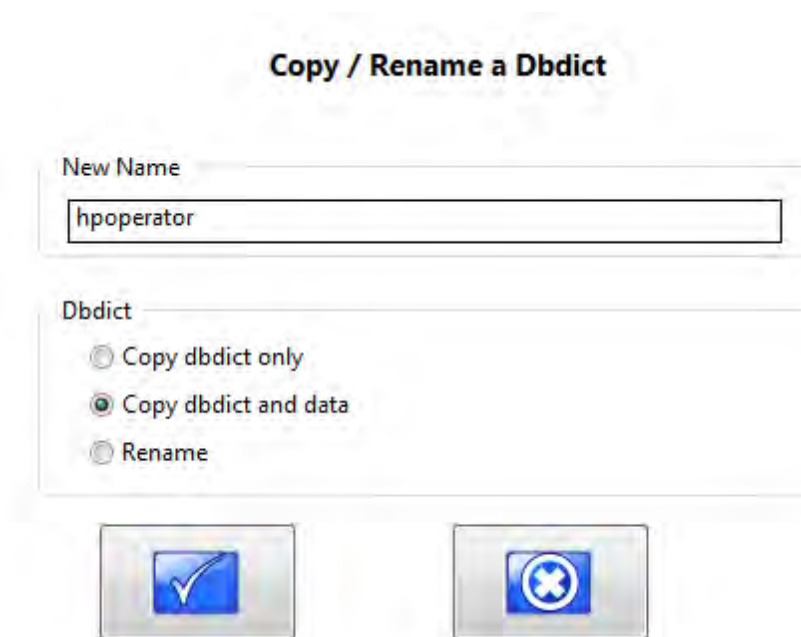
Database dictionary element type	RDBMS mapping required
Array data type	Mapped to an alias table
Sub-field of the array	Mapped as a character column in a main table
RC flag	False

Example: Re-mapping an array

The following example illustrates re-mapping an array from a field in main table format to a multi-row array table format. Typically, you would re-map a field in main table array in order to improve your query performance. Using a multi-row array table array format allows the RDBMS to directly query the array rather than having to forward the query to Service Manager for evaluation.

Existing array mapping

Suppose you want to re-map the `assignment.groups` array field in the `operator` table from a field in a main table array to a multi-row array table array. To ensure that your tailoring efforts do not negatively impact your system, you should first make a copy of the `operator` table. For example, create a new table called `hoperator`.



The existing array might be mapped as follows on a Microsoft SQL Server RDBMS.

Name: Status:

Case Mode: Case Sensitive

Fields Keys SQL Tables

Name	Type	Level	Index	SQL Name	SQL Type	SQL Table
assignment.groups	array	1	88			
assignment.groups	character	2	1	ASSIGNMENT_GROUPS	TEXT	m1

Existing array field mapping: field in main

Database dictionary field	Value
Name	assignment.groups
Type	array
Level	1
Index	88
SQL Name	
SQL Type	
SQL Table	
SQL RC	

Since this is a field in main table mapping, the array itself has no direct mapping on the RDBMS. Only the array's sub-field has a SQL mapping.

Existing sub-field mapping: field in main

Database dictionary field	Value
Name	assignment.groups
Type	character
Level	2
Index	1
SQL Name	ASSIGNMENT_GROUPS
SQL Type	TEXT
SQL Table	m1
SQL RC	False

The sub-field of the array is mapped to a character large object data type. For example, TEXT is a character large object data type in Microsoft SQL Server.

Existing array query performance

Suppose you make the following Service Manager expert search from the Search Interaction Records form.

```
assignment.groups="Application"
```

If you enable the `sqldebug:1` parameter, you can see that Service Manager converts this query in the following SQL statement:

```
SELECT NAME, ASSIGNMENT_GROUPS FROM HPOPERATORM1 WHERE ASSIGNMENT_GROUPS='Application';
```

However since the `ASSIGNMENT_GROUPS` column is in a character large object data type, the RDBMS cannot query the field directly. Instead, the RDBMS must forward each row to Service Manager for it evaluate the contents of the array. Using the out-of-box data, this query returns all 18 sample incident rows.

18 rows forwarded to Service Manager for evaluation

Operator ID	Assignment Groups
CMS.Process	Application, Hardware, Network
Catherine.Campbell	Application, Network, Office Support (North America) SAP Support (North America)
...	Application, ...
Rachel.Boudreau	Application, E-mail / Webmail (North America), Network, SAP Support (North America)

Because of the inefficiency of this query, the system produces performance alert messages.

```
Partial File Scan for query involving fields {assignment.groups}
(se.search.engine,select.records)
```

```
SQL Query incomplete because field (assignment.groups) in file
(hoperator) can not be used in an SQL query
(se.search.engine,select.records)
```

```
Hit Ratio not achieved on file hoperator and query
(assignment.groups="Application"): Of 101 records checked so far, 94
did not match the query (display,show.rio)
```

A partial file scan has two adverse performance impacts on your system:

- The RDBMS uses network traffic to send rows to Service Manager
- Service Manager must search within the array of each row the RDBMS forwards to it
- The hit ratio can be low for such queries

Prepare to re-map the array

Before you re-map your array, you need to determine the answers to following questions.

- What is the next free alias table for the database dictionary record?
- What SQL data type will the array sub-element require?
- What length should your character fields have?

Determine the next free alias table

To re-map the array as a multi-row array table, you must first determine what the next free alias table is for the database dictionary record. You can see what alias tables are already in use from the SQL Tables tab of the database dictionary record.

Name:

Case Mode: Case Sensitive

Alias	Name	Type
m1	HPOPERATORM1	sqlserver
a1	HPOPERATORA1	sqlserver
a2	HPOPERATORA2	sqlserver
a3	HPOPERATORA3	sqlserver
a4	HPOPERATORA4	sqlserver
a5	HPOPERATORA5	sqlserver

In this case, the incidents database dictionary already has five existing alias tables, so the next free alias table is a6. We shall create a new alias table called a6 that will store our re-mapped array data.

Determine the data type of the array sub-element

Next, determine what data type the array sub-element will require in SQL terms. You can use the Service Manager `sqldbinfo` table as a guideline for mapping a Service Manager data type to a SQL data type. For example, the `assignment.groups` sub-element uses the character data type. According to the `sqldbinfo` table, a character data type maps to a varchar data type on Microsoft SQL server.

SQL DB Type:

SM Type	SQL Type	Get Size	Force Blob
number	float	<input type="checkbox"/>	<input type="checkbox"/>
character	varchar	<input checked="" type="checkbox"/>	<input type="checkbox"/>
date/time	datetime	<input type="checkbox"/>	<input type="checkbox"/>
logical	char(1)	<input type="checkbox"/>	<input type="checkbox"/>
label	varchar	<input checked="" type="checkbox"/>	<input type="checkbox"/>
expression	image	<input type="checkbox"/>	<input checked="" type="checkbox"/>
record	image	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Long Text Type: Treat As Long?

Long Blob Type: Treat As Long?

Short Blob Type:

Uppercase All Names?

Only One Long per Table?

Verify that your SQL data type mapping meets your data needs. Ideally, you want to identify the source of your data and match its SQL mapping. For example, the source of the `assignment.groups` field in the `operator` table is the `name` field in the `assignment` table.

When you look at the database dictionary for the source data, you see that the source data is mapped as a varchar(60) data type.

Name: Status:

Case Mode: Case Sensitive

Name	Type	Level	Index	SQL Name	SQL Type	SQL Table
descriptor	structure	0	1			
name	character	1	1	NAME	VARCHAR(60)	m1

In this example, we shall re-map the array sub-element to use the varchar(60) data type to match the source data type.

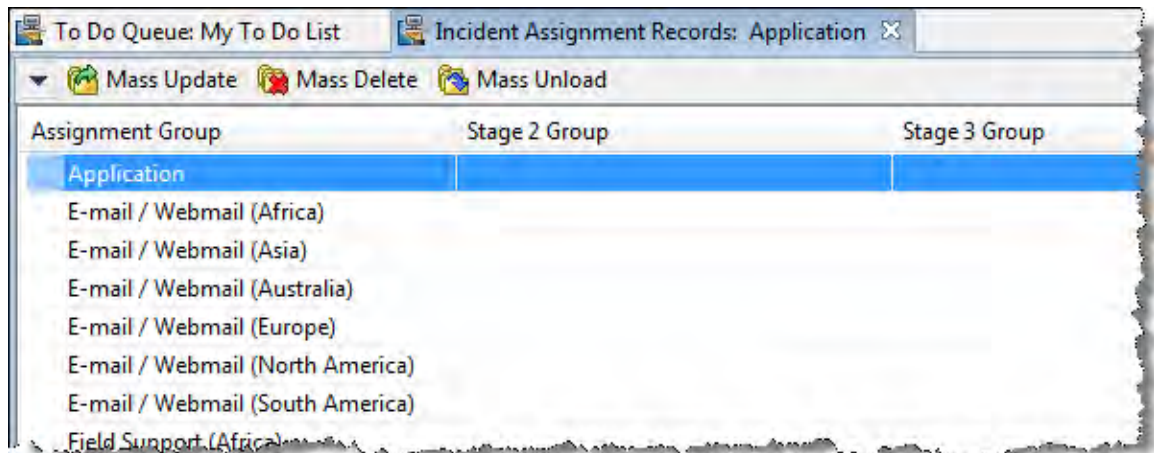
Determine the length of character fields

If you want to re-map your array sub-element with a simple character data type, you must choose a length for the character field. Remember that multi-row array tables cannot use character large object data types. There are two methods to determine an appropriate length for your character data.

- Match the length of the data source (preferred)
- Review your existing data and add a padding

Ideally, you want to identify the source of your data and match its SQL mapping. For example, the source of the `assignment.groups` field is the `name` field of the `assignment` table. Since the `name` field is mapped as a `varchar(60)` data type, we will map our array to the same length of `varchar(60)`.

You can also use your existing data to determine the proper size of your character fields. For example, if you review the source data of the `assignment.groups` field, you will see text strings from 20 to 40 character long.



You could therefore re-map this data as a `varchar(40)` data type and likely not lose any data. However, you may want to provide the option to store assignment groups more than 40 characters long without having to store the array elements in separate rows. In this case, you may want to use a larger value such as `varchar(50)` or `varchar(60)`.

When choosing a length value, you want to avoid selecting a SQL length shorter than the maximum length of your existing data. The system will wrap any existing data that exceeds the SQL length of the character field into a new row. Wrapping data into a new row can result in data corruption and broken queries. For example, if you select a length of `varchar(10)`, then you any existing assignment group longer than ten characters will wrap into two or more rows. Using this mapping, the system would change an existing assignment group of "Application" to have one array element row with the value "Applicatio" and another array element row with the value "n". Any RDBMS query searching for the value "Application" would fail because there is no longer one element with this value.

New array mapping

Suppose we have chosen to re-map our `assignment.groups` array to a multi-row array table with the following values.

Multi-row array table re-mapping options

Mapping option	Value
Next available alias table	a6
Array field re-mapped to alias table	a6
Array sub-element re-mapped to alias table	a6
Array sub-element re-mapped to simple data type	VARCHAR
Array sub-element length re-mapped to appropriate value	VARCHAR(60)

You can use the following steps to re-map `agreement.ids` field from a field in main table array to a multi-row array table.

1. Open the `hoperator` file in the database dictionary utility.
2. Double-click on the `assignment.groups` array field.
3. In the SQL Table field, type `a6`.
4. Click **Next** to change the mapping of the array sub-element.
5. In the SQL Name field, verify that the column name meets the naming conventions for your RDBMS. For example, `AGREEMENT_IDS` is a valid name for a Microsoft SQL Server column.
6. In the SQL Type field, type `VARCHAR(60)`.
7. In the SQL Table field, type `a6`.
8. Click **OK** to update the mapping
9. Click **OK** to save the incident database dictionary record. Service Manager displays re-mapping status messages in the top right of the form.

Name: Status:

Case Mode: Case Sensitive

Fields						
Name	Type	Level	Index	SQL Name	SQL Type	SQL Table
assignment.groups	array	1	88			a6
assignment.groups	character	2	1	ASSIGNMENT_GROUPS	VARCHAR(60)	a6

New array query performance

Suppose we re-use the same Service Manager expert search from the Search Interaction Records form.

```
assignment.groups="Application"
```

Service Manager generates a new SQL statement that joins the main and alias tables:

```
SELECT m1."NAME" FROM HPOPERATORM1 m1 JOIN HPOPERATORA6 a6 ON  
(m1."NAME" = a6."NAME") WHERE ((a6."ASSIGNMENT_  
GROUPS"='Application'));
```

The join statement requires some additional system resources that the old query did not, however since the ASSIGNMENT_GROUPS column is now a simple character data type, the RDBMS can query the field directly. While the query still returns the same 18 sample operator records, this time it does not have to forward any rows to Service Manager for evaluation. Nor does running this query generate performance warning messages. This means that the new query is more efficient despite the added overhead of a join statement.

Valid data type changes

You can only change existing database dictionary mappings under certain conditions. If your database dictionary change does not fall within one of the following conditions, you will have to create a new database dictionary mapping to replace the existing mapping.

Valid data type changes

Starting data type	New data type	Example
Fixed-length character field	Variable-length character field with the same column length	Change a CHAR(10) mapping to a VARCHAR(10) mapping
Variable-length character field	Fixed-length character field with the same column length	Change a VARCHAR(10) mapping to a CHAR(10) mapping
Fixed-length character field	Fixed-length character field with a longer column width	Change a CHAR(10) mapping to a CHAR(15) mapping
Fixed-length character field	Variable-length character field with a longer column width	Change a CHAR(10) mapping to a VARCHAR(15) mapping
Variable-length character field	Variable-length character field with a longer column width	Change a VARCHAR(20) mapping to a VARCHAR(40) mapping
Variable-length character field	Fixed-length character field with a longer column width	Change a VARCHAR(20) mapping to a CHAR(40) mapping

In addition to changing a column data type, you can also rename a column or a table.

Valid name changes

Database dictionary item	Equivalent RDBMS item	Example
File name	Table name	Change a table name from MY_TABLE to SAMPLE_TABLE
Field name	Column name	Change a column name from MY_COLUMN to SAMPLE_COLUMN

When are RDBMS mappings added or updated?

Service Manager must have proper RDBMS rights to directly change RDBMS mappings. If the system does not have these rights, it will not add or update any RDBMS mappings.

Service Manager issues CREATE TABLE statements to create RDBMS mapping in the following circumstances:

- You create a new file (a logical view of a table) in a database dictionary record
- You create new fields (a logical view of a column) in a database dictionary record

Service Manager issues ALTER TABLE statements to update existing RDBMS mappings in the following circumstances.

- You change a fixed-width character field mapping to a variable-width character field mapping
- You change a variable-width character field mapping to a fixed-width character field mapping
- You widen a column
- You rename a column
- You rename a table

Service Manager issues DROP TABLE statements to delete existing RDBMS mappings in the following circumstances.

- You remove a table entry the SQL Table field in a database dictionary record
- You remove a table from the SQL Tables file

Service Manager does a full table copy of an existing table in the following circumstances.

- You change the RC flag for any field in a database dictionary record
- You change a field alias that is not mapped to a NULLTABLE alias
- You associate a new main or alias table to an existing database dictionary record
- You change the RDBMS connection information to use a new type of RDBMS