HP Service Manager

For the Supported Windows® and UNIX® operating systems

Software Version: 9.32

Document Engine Guide

Document Release Date: August 2013

Software Release Date: August 2013

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notice

Trademark Notices

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 sign in. 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 login 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

Chapter 1

Document Engine overview

The Document Engine is a tailoring tool that includes the possibility to customize the system without the need of RAD changes. It provides a centralized method for setting privileges and behavior for standard actions, such as list, view, and search. It increases consistency across modules. The three main components of the Document Engine are Objects, States, and Processes. Processes can be reused for more modular programming and the integration is seamless, which can reduce development time.

What is the Document Engine?

The Document Engine comprises a set of tools and methodologies for developing and modifying Service Manager workflows. The Document Engine was originally tasked to develop an underlying set of base functionality that would support multiple modules inside Service Manager, improve consistency of the user interface between modules and reduce the amount of code needed for any new module.

The Document Engine extends the display application capabilities with simplified and more extensive actions especially those involving multiple application calls. In addition, the Document Engine supports the use of joined tables and master format control calls. It is designed to meet the needs of most customers out of the box, yet retain flexibility. The relationship between Objects, States, and Processes is hierarchical.

The Document Engine controls behavior with Objects. An Object is referenced whenever a form opens, and the Object determines the behavior for the state of the form (open, list, search, etc.). Objects define overall table behavior. Within the Object, a State describes where a record is in its lifecycle (open, list, search, etc.). Within the State, different Processes are executed depending on the actions initiated by a user on the record. States also define how the system displays a record and what options (actions) are available at specific times or circumstances. For instance, States can determine an action, such as Save, given a user’s access privileges.

Processes are called from States based on the user's action. The Process uses RAD expressions, JavaScript, and calls to existing RAD applications to perform actions against the current record..

Benefits of the Modular Approach

The advantages of the modular design include consistency in development, reduced development time, and flexibility.

Consistency

The fact that the engine allows all applications to run using the same base RAD applications brings consistency to the Service Manager application suite. Core functionality, such as locking, alerts, approvals, and use of record list functionality, will work the same for any module, as they are all using the same code base.

Reduced Development Times

The modularity of the Document Engine allows for reuse of existing code and Processes.

Flexibility

The Document Engine uses Process records as a mechanism to modify the behavior of modules inside of Service Manager applications. You can create a new Process that has a different behavior than the base system, without the need to change or remove the original Process from the system. Additionally, the system’s base Processes can be overridden with your own Processes to give greater flexibility to the system developer when tailoring Service Manager to meet an organization's specific requirements.

Accessing the Document Engine

To access the Document Engine:

1. Start a Service Manager Client and log in as an administrator.
2. In the Service Manager System Navigator, click **Tailoring**.
3. Click **Document Engine**. From here, you can access the main three areas to define Objects, Processes or States. You can also set up Alerts and Approvals to be called from Objects or create Search Configuration Records, which are used by Objects.

Chapter 2

Objects

Objects are a base set of definitions that determine behavior of records and set the definitions and governing rules. Objects correspond one-to-one with database dictionary (dbdict) records in Service Manager. If a table does not have a dedicated Object record, the Document Engine applies the settings in the DEFAULT Object. All Objects should have list and default States defined. If not otherwise supplied, the default States are db.browse, db.list, db.search, and db.view.

NOTE: Do not **modify** or **delete** the DEFAULT Object Record as doing so will cause unpredictable results.

The Object record sets up the definitions and governing rules for the behavior of the table within the Document Engine. For example these may include:

- The application used to create the users profile within this Object that determines what actions this user may take against any record of the table.
- The State records used in specific circumstances (see the States section for more details).
- The category, phase, and paging file names for the Object.
- The name of the number record to be used for this Object.
- How locking is to be used by this Object.
- Setup of revisions for records in the table.
- Which variables should be available to processes that run against this Object.
- Which global lists should always be available when using this Object.
- Use of activity records.
- How alerts are processed against this Object.
- How approvals are processed for this Object.
- Settings for the work queues.
- Ability to set up personal or global views and default templates.
- Notifications on add / update / delete of a record in this Object.
- Ability to configure additional search choices against this Object.

To view a list of the out-of-box Objects, click **Search** from the Object Definition form.

To view a list of the fields and field descriptions for the Object definitions see "Object Definition form and fields" below.

Create and update Objects

To create an Object:

1. Navigate to the Document Engine. See "Accessing the Document Engine" on page 9 for the steps.
2. Double click **Objects**. The Objects form opens.
3. Use the tabs on the Objects form, to fill in the fields required to create an Object that will perform the functions you desire. See the field descriptions for additional information.

To update an Object definition:

1. Access the Document Engine. See "Accessing the Document Engine" on page 9 for the steps.
2. Enter the name of the Object you want to update in the Object field or click **Search** to search for the Object.

Object Definition form and fields

The field descriptions for the Object Definition form are:

Field Name	**Description**
File name *file.name*	Enter the dbdict name for the Object, that is, use the dbdict name that corresponds to this Object's name. (required)
Common name *message*	The system fills in this information from Data Policy. It is a common name for the Object. The common name can be a simple name, such as Work Order.
Unique key *unique.field*	The system fills in this information from the dbdict. This is the unique key for the Object.

This form also includes the following tabs. The fields on these tabs are described in the field descriptions tables for the respective tab.

- "Object Info tab field descriptions" below - specifies the general properties and behavior of the Object.
- "Locking tab field descriptions" on page 14 - determines the locking behavior for the Object.
- "Revisions tab field descriptions" on page 15 - tracks revisions for the Object.
- "Variable and Global Lists tab field descriptions" on page 15 - describes local variables and global variables used by the Object.
- "Activities tab field descriptions" on page 16 - defines logging (activities).
- "Alerts tab field descriptions" on page 16 - defines where to set alerts and the conditions for generating alerts.
- "Approvals tab field descriptions" on page 18 - sets approval options for the Object.
- "Manage Queues tab field descriptions" on page 20 - controls how queues display as well as threading, and who can create views (inboxes).
- "Views and Templates tab field descriptions" on page 21 - defines whether or not a user can create global and personal views and templates on the Object.
- "Notifications tab field descriptions" on page 21 - identifies notifications sent automatically for add, delete, or update activity for any record in the Object.
- "Search Configuration tab field descriptions" on page 22 - controls the available options on the More Choices tab of the search screen.

Object Info tab field descriptions

This tab specifies the general properties and behavior of the Object.

The field descriptions for the Object Info tab are:

Field Name	Description
Description field *desc.field*	Specifies a short description of the Object.
Profile application *profile.appl*	Specifies the RAD application that creates the profile that determines if a user can perform certain functions, such as add and delete. For example, db.environment. (rquired)
Profile variable *profile.variable*	Specifies a variable that can be accessed any time this Object is called without accessing the environment record.For example, $L.env. (required)

Field Name	Description
Number record name *number.record*	Defines a number class for the Object, which can be accessed from a call to the getnumb application either via a Process or FormatControl or RAD call. It can be used to retrieve a sequential number as a unique key for the record. For example, EXWorkOrder.
Category table name *category.file.name*	Specifies the table name that links with the category table associated with this Object's category value. When displaying a record of this type, if a field called category exists, then the Object will search the category table name and for the record with the corresponding name. If found, the system stores the Category File Name as a variable: $L.category.
Phase table name *phase.file.name*	Specifies the table name that links with the phase table associated with this Object, if applicable. When the system displays a record of this type, if a field called phase exists, then the Object goes to the phase table name and selects a record with a corresponding name. If found, the system will store the phase table name as a variable: $L.phase.
Paging table name *paging.file*	Specifies the name of the table used for storing pages. Pages are created as full copies of the current record prior to the latest updates being applied. This is done every time a record is updated, which creates a detailed audit trail.
Master format control *master.fc*	Specifies the name of the Master Format Control record, if one exists for the record Object. Master Format Control allows you to define in one record the Format Control statements that apply to all phases in an area, for example Change Management Request Phases. Typically, the name of the master format control is the name of the dbdict or all categories in Incident Management, Problem Management, or Service Desk.
Joindef *joindef*	Specifies the name of the joindef record used to join multiple tables, for example joincomputer, or an expression that evaluates to a valid joindef name, such as joindef in $L.category
Status field *statusField*	Specifies the field name that contains the record's status information.
Assigned to fields *assignedToFields*	Specifies the field name that contains the assignee name field for this Object. This field will be referred to when Folder Entitlement verifies that the record is assigned to the logged in operator.
Workgroup fields *workgroupFields*	Specifies the field name that contains the assignment group field for this Object. This field will be referred to when Folder Entitlement verifies that the record is assigned to any of the logged in operator's work groups.

Field Name	Description
Open state *open.state*	Specifies the State definition record to use upon opening a new record.
Close state *close.state*	Specifies the State definition record to use for close processing of an existing record.
List state *list.state*	Specifies the State definition record used to display a list of records.
Default state *default.state*	Specifies the name of the State used as default for the Object. To edit a record in this Object, the default State is used.
Search state *search.state*	Specifies the State definition record to use for searching.
Browse state *browse.state*	Defines the State definition record to use when records use locking. Essentially, this field defines a read-only State when a record is currently locked by another user.
Manual states *manual.states*	Specifies the array of States that may be used with this Object other than the life-cycle states of open, close, list, view, search or browse.

Locking tab field descriptions

The Locking tab determines the locking behavior for the Object. This means that when someone is actively updating a record the system does not allow another user to update the same record.

The field descriptions for the Locking tab are:

Field	Description
Use locking *use.locking*	Select this check box to enable locking.
Lock on display *lock.on.display*	If true or a condition that evaluates to true, Service Manager attempts to lock the record as soon as it displays. Use locking must be enabled for this field to work.
Lock parent record *lock.parent*	Locks the current record and the record’s parent. For example, when this field is checked and someone is updating a change task, the change request for the task is also locked.

Field	Description
Parent Id Field *parent.id*	Enter a field name in the current record that contains the ID of the parent to lock.
Parent Filename/Object *parent.object*	The name of the table that contains the parent record.
Watch Variables *watch.variables*	Watch Variables are used when the Document Engine checks to see if a record has been changed or not. Watch Variables must be NULL when the record first displays.

Revisions tab field descriptions

The Revisions tab determines the revision behavior for the Object

The field descriptions for the tab are:

Field Name	Description
Revision table name *revision.file*	Specifies the name of the table in which you want to store revisions to this Object's records.
Max # of revisions *max.revisions*	Specifies the total number of revisions allowed for this Object. If left blank, then there is no upper limit.

Variable and Global Lists tab field descriptions

The Variables/Global List tab describes local variables and global variables used by the Object.

Note: Global variables are built and stored in memory.

The field descriptions for the Variable/Global List tab are:

Field Name	Description
Local variables *local.variables*	Enter a list of local variables that can be used in Processes. Local variables are defined in the application. Local variables are assigned to the Object you are creating and are available to all Processes and States associated with the Object. Local variables are not available to other Objects.
Global lists *global.lists*	Global lists, once created, are available to all Processes. Global lists can be built on login, if they are listed in the Startup list. Global lists are available every time you access the Object.

Activities tab field descriptions

The Activities tab defines update (Activity) logging for the Object.

The field descriptions for the Activities tab are:

Field Name	Description
Activity log table *activitylog.file.name*	The name of the table to hold the activity log entries for the Object.
Selection list variable *activity.selection.var*	The variable to use on an update form to display the types of activities the operator can select when performing an update on a record for a specific Object.
Posting link *activity.post.link*	The name of the link used to post information.
Require update if an activity record is NOT generated? *activity.mandatory*	A check box to indicate that an activity update is required when the selected.
Update field *update.field.var*	The field or variable that contains the activity update on the form. This field appears only when **Require update if an activity record is NOT generated?** is present.
Display message *activity.mandatory.msg*	The message indicating that an activity update is required. This field appears only when **Require update if an activity record is NOT generated?** is present.

Alerts tab field descriptions

The Alerts tab defines where to set alerts and the conditions for generating alerts. Any Object that has a unique key can use these alerts.

The field descriptions for the Alerts tab are:

Field Name	Description
Alert location *alert.location*	The Alert location defines where the name of the Alert Definition to execute is found or stored. Specify where the Alert Definition is stored: Record: Store alerts in the record itself. Category: Store alerts in the category file defined on the Object Info tab. Phase: Store alerts in the Phase record defined on the Object Info tab. Object: Store alerts in the Object record. When selected an Alerts array table will be displayed to enable you to fill in alert(s) to be used. Note: The Object's table needs to have a unique key for the Alerts to relate to. A table that contains a no nulls key rather than a unique key cannot use Alerts.
Alert condition *alert.condition*	Specify a condition to determine whether or not to process the alert. For example, open in $L.file~=false.
Alert field name *alert.field.name*	Specify the field name that contains the actual alert name, as defined by Alert location.
Alert status field *alert.status.field*	Specifies the field in the current record in which to put the alert status, after the alert is processed.
Alert update process *alert.update.process*	Specifies the name of the Process record for additional functions that the system performs after the alert runs.
Log alerts? *log.alerts*	If selected, the alerts are moved after processing to the Alertlog file to keep an Alert history.
Process alerts on parent? *alerts.against.parent*	If you have selected the Locking Parent Record field on the Locking/Revisions tab and you select this check box, then the alert will register against the parent record as well, when it is activated.
Recalculate alerts if *alert.recalc*	Specifies conditions that determine whether to recalculate conditions on existing alerts.
Reset alerts if *alert.reset*	Determines when to delete existing alerts and recalculate all conditions.

Approvals tab field descriptions

The Approvals tab sets approval options and their associated notifications for the Object. Approvals are defined in the ApprovalDef file.

The field descriptions for the Approvals tab are:

Field Name	**Description**
Approval condition *approval.condition*	If the approval condition evaluates to true, approvals are used on the Object's records.
Approval location *approval.location*	Indicates where the approval information is stored: record, phase, object, or category.
Approval field name *approval.field.name*	The field name that contains the actual Approval name within the table that is defined in the Approval location.
Approval status field *approval.status.field*	The field in the current record in which to store the approval status.
Approval groups *approval.groups*	Stores a variable to contain the groups the current user must belong to in order to issue approvals for this Object.
Approval type *appr.cond.type*	There are four pre-defined approval types: All must approve: All groups/operators defined in the Approval Definition must issue an approval before the system sets the status of the record to approved. If only one or some but not all of the groups/operators issue an approval, then the status is set to pending. One must approve: The record is approved with one approval from any member of the approving group/operator. Quorum: The record is approved as soon as a majority of the approving group indicate approval. All must approve – immediate denial: All groups/operators must approve the record. The first denial causes the status to change to deny. All other approvers do not need to take any action.
Approval notification *single.notify.approva*l	Select the notification to run if the request is approved.
Denial notification *single.notify.denial*	Select the notification to run if one approver denies the request.

Field Name	Description
Retraction notification *single.notify.retraction*	Select the notification to run when retracting a previous action.
Final approval notification *final.notify.approval*	Select the notification to send once the final approval is granted.
Final denial notification *final.notify.denial*	Select the notification to send when the request is denied.
Approval FC *appr.fc*	Specifies the name of the Format Control record to run upon approval.
Approval process *approval.process*	Select the Process to run when the record is approved.
Denial process *denial.process*	Select the Process that runs when the record is denied.
Preapprove on open *preapprove.cond*	Determines whether the record should be automatically approved. If the condition is true and the user belongs to one of the pending approval groups, the approval is processed automatically. If the user does not belong to one of the pending approval groups, the approval does not occur automatically and must then go through the regular approval process. Defaults to true.
Log approvals? *log.approvals*	Select this check box to log the history of approvals in the ApprovalLog table.
Require appr. comments *approval.comments*	If checked, approval comments are requested from the approver.
Aggregate approvals? *aggregate.approvals*	If checked, approvals are cumulative.
Recalculate approvals if *approval.recalc*	Specifies the conditions that determine whether or not to recalculate the conditions on the existing approvals.
Reset approvals if *approval.reset*	Determines when to delete existing approvals and recalculate all conditions.

Manage Queues tab field descriptions

The Manage Queues tab controls how queues and views display as well as threading. These same fields are available in Data Policy for files that do not have an associated Object record, or they are virtual joined into the datadict record from the Object record if it exists.

The field descriptions for the Manage Queues tab are:

Field Name	**Description**
Manage condition *scm.condition*	Specifies a condition that allows only certain users to view the queues that display records of this Object. For example, browse in $G.pm.environment.
Manage display format *scm.manage.screen*	Select the format to use to display the view. The out-of-box Service Manager has a default display format: sc.manage.generic that is used if no other form is chosen. HP recommends that you do not change the sc.manage.generic format.
Manage default view *scm.inbox*	Select the default view for this queue. By specifying a user view for a particular user, a specific list of views can be set up for the HP Service Manager Manage Queues. If a user does not have a specific view defined , the default user view is used.
Manage default query *scm.query*	Specifies a default query to run if no default view is selected.
Default query description *scm.query.name*	Specifies a name for the above field. You can associate a message with this field. For example, scmsg(491, "us").
Thread view -> search? *scm.thread.list.edit*	Specifies true or an expression that evaluates to true to open a new thread when conducting a search.
Search format (if necessary) *scm.search.format*	Select a default search format.
Thread search -> list? *scm.thread.search.list*	Specifies true or an expression that evaluates to true for a new thread when a user finds a list of records to view.
Thread list -> edit? *scm.thread.list.edit*	Specifies true or an expression that evaluates to true for a new thread when a user selects a record to view out of a list of records.
Thread view -> edit? *scm.thread.inbox.edit*	Specifies true or an expression that evaluates to true to open a new thread when the user views an existing record out of the queue.

Field Name	Description
Allow add condition *scm.add.condition*	Specifies an expression that evaluates an operator’s ability to add a record.
Add/open application *scm.add.appl*	Specifies the name of the application to call when a record is added or opened.
Parameter Names *scm.add.names*	Specifies the parameter names to pass to the application specified in the Add/open application field.
Parameter Values *scm.add.values*	Specifies the parameter values to pass to the application specified in the Add/open application field

Views and Templates tab field descriptions

The Views/Templates tab defines whether or not a user can create global and personal views as well as template support.

The field descriptions for the tab are:

Field Name	Description
Can create personal views *personal.inbox*	Specifies a condition that evaluates to true or false. True allows the user to create personal views.
Can create system views *global.inbox*	Specifies a condition that evaluates to true or false to determine if the user can create global views.
Default Template *default.template*	Specifies the name of the default template to use for records in this table.
Supports Templates? *supportTemplates*	Select this check box to enable support for templates for the Object.

Notifications tab field descriptions

The Notification tab identifies notifications sent automatically for add, update, or delete activity for the Object.

The field descriptions for the Notifications tab are:

Field Name	Description
Add *notification.add*	Select the notification that is sent automatically when a record is added to the table.
Update *notification.update*	Select the notification that is sent automatically when a record is updated in the table.
Delete *notification.delete*	Select the notification that is sent automatically when a record is deleted from the table.

Search Configuration tab field descriptions

The Search Configuration tab controls the available choices on the More Choices tab of the search screen.

The field descriptions for the Search Configuration tab are:

Field Name	Description
Table Name *tablename*	The name of the table that will be queried.
Search Format *searchFormat*	The name of the sub-format used for the More Choices tab.
Initialization Process *init.process*	The name of a Process that will be run before displaying the search form.
Allow Advanced Find *allowAdvAccess*	Defines the conditions that determine whether or not to allow Advanced Find.

Defined Queries tab

The Defined Queries tab defines the query statements and labels to use on the More Choices tab of the search screens (for example, More Choices on the Search Incidents form). The fields are defined in the SearchConfig table.

The field descriptions for the Defined Queries tab are:

Field Name	Description
Id *id*	Provides the unique ID for the query and cannot include special characters including spaces.
Query *query*	A query expression using system language syntax. For example, `assignee.name=operator().`
Description *description*	Provides the label for the check box on the More Choices tab.

Ranges tab

The Ranges tab allows you to easily set up a search for begin and end date ranges. To do so click the **Modify Configuration** link, to define a variable each for begin and end to use as input in the form. Enter this variable in the Variable 1 column and use the Field and Operator 1 columns to define the query that will be executed. Modify Configuration is also how modifications are made on the Defined Queries tab as well.

The field descriptions for the Ranges tab are:

Field Name	Description
Field *fieldName*	Specifies the field in the table to use in the query.
Operator 1 *operator1*	The comparison operator in the query. (for example: >=)
Variable 1 *variable1*	Specifies the variable used as input on the form.
Special Type *specialType*	Not used at this time.

Chapter 3

States

States are called by Objects and are defined by Processes. The State record contains information about how a record looks and acts at a specific period in time.

A State record definition can include the following:

- The State name.
- The display screen to use in order to display the record or records.
- The initialization process used when a list is first displayed.
- The format to use to display the data.
- The input condition to indicate whether or not a user can modify the record.
- The Process to run when a user triggers a specific display option (Display Action).

The State record used depends on how many records the user views. There are out-of-box State records for searching for records, viewing a list, viewing a single record, and browsing a record.

Searching

When there are no records in the current file variable, it is assumed that the user is in “search” mode. The State that is used is the “Search State” defined in the table’s Object record. The default search state is “db.search”.

Record Lists

When viewing a list of records, the “List State” defined in the table’s Object record is used. The default list state is “db.list”.

Note: In some earlier versions of Service Manager, record lists are referred to as QBE lists.

Viewing a single record

When viewing a single record, the record is first checked to see if it includes a “State” field in the dbdict. If the field exists and is populated, the contents of that field will be used as the current State of the record. If this field does not exist or is NULL, the “Default State” defined in the table’s Object record is used. The default value for the default state is “db.view”.

Browsing a record

When browsing a record in read-only mode, the “Browse State” defined in the table’s Object record is used. Note that only files that use locking need a browse State. Although viewing a record without having update rights looks similar to the browse screen, it is using the default State, not the browse State. There is no default browse state.

Integration tips for display application

If a display screen does not have any display events associated with it, the system automatically uses the se.default display event that performs se.lock.object in case of “OnFormModified”. If a display option record has the “Modifies Record” check box activated, the record is locked when the button is pressed.

Create and update States

To create a new State:

1. Access the Document Engine. See "Accessing the Document Engine" on page 9.
2. On the State Definition form, fill in the fields required to create a State to perform the functions you desire. See the field descriptions in "State Definition field descriptions" below.

To modify an existing State:

1. Access the Document Engine. See "Accessing the Document Engine" on page 9.
2. Enter the name of the State you want to modify in the State field or click **Search** to search for the State.

State Definition field descriptions

The State definition record defines the behavior and display specifications for a record at a specific period in time.

In the State record non-base methods can be defined or the behavior of base methods can be modified. Base methods are described in the sections Base functions in se.search.engine, Base functions in se.view.engine, and Base functions in se.list.engine. They include functions such as save, find, fill, OK, Cancel, and Search.

The field descriptions for the State Definition form are:

Field Name	Description
State *state*	Specifies the name of the State. (required)
Display Screen *display.screen*	The display screen to associate with the State.
Initialization Process *init.process*	The name of a Process to run prior to entering the State.
Format *format.name*	Specifies the format in which the State record opens. This form is stored in a record, and it can be a variable or hard-coded. The name of the format can be hard-coded or be contained in a variable or retrieved from a record via a system language expression.
Input Condition *input.condition*	The input condition is evaluated for view States and determines whether or not the record is read-only. Specify false for read-only or true for editable. You may also enter an expression that evaluates to true or false.otherwise, enter true or enter an expression that evaluates to true or false.
Non-base methods	
Display Action *process.label*	Specifies the action parameter that comes from a display action after input from a user. The display action comes from the action field in the display option of a button or menu item clicked by a user.
Process Name *valid.process*	Specifies the name of the Process run as a result of the action.
Condition *process.condition*	Specifies an expression that evaluates to true for the Process specified in the Process Name to be called.
Save First *run.save.before*	Choose to run the save Process before you run this Process. Type true to save first; otherwise, enter false. The default is false.

Chapter 4

Processes

Processes are the smallest discrete units of work available to the Document Engine and are the level where the data is manipulated. Users can create their own Process or use one of the over 700 Processes that ship with Service Manager. The Process definition consists of initial expressions, RAD, and final expressions, entered on the Initial Expressions, Initial Javascript, RAD, Final Expressions, Final Javascript, and Next Process tabs, as well as an optional call to a next Process. Expressions are written using standard RAD expressions and/or JavaScript.

Create and modify a Process

To create a Process:

1. Access the Document Engine. See "Accessing the Document Engine" on page 9.
2. Using the tabs on the Process panel, fill in the fields required to create a Process that will perform the functions you desire. See "Process Definition form and field descriptions" below for more information.

To modify a Process:

1. Access the Document Engine. See "Accessing the Document Engine" on page 9.
2. Enter the name of the Process you want to modify in the Process Name field or click **Search** to search for the Process.

Process Definition form and field descriptions

On the Process Definition form, you define new Processes or edit existing Processes. Processes run code or expressions to perform the user selected actions

The field descriptions for the Process Definition form are:

Field Name	**Description**
Process Name *process*	Specifies the name of the process. (required)

Field Name	Description
Save Cursor Position *save.cursor.position*	Select this box if you want to return to the same cursor position after the action (for example on a fill).
Run Standard Process when complete? *run.standard*	When selected, then the system runs a standard process after completing the current action or Process. A standard Process such as save, for example. If you have created a save Process and want to run the save Process that comes with the Document Engine after completing the save process you defined, check this box. A standard process is defined in the base functions sections of this document.
Run in Window? *run.in.window*	When selected, the process runs in a separate window.
Window Title *window.name*	If the **Run in Window?** check box is selected, specify a title for the window. An scmsg expression can be used for localized window titles, such as scmsg(1980, "us").

In addition to the fields described above, there are also tabs on this form that allow you to further define the Process. These tabs are:

- "Initial Expressions tab" below
- "Initial Javascript tab" on the facing page
- "RAD tab" on the facing page
- "Final Expressions tab" on the facing page
- "Final Javascript tab" on page 31
- "Next Process tab" on page 31

Initial Expressions tab

The Initial Expressions tab defines the initial expressions that run prior to the initial Javascript and prior to the RAD calls defined on the RAD tab. The initial expressions are written using standard Service Manager expressions.

Initial Javascript tab

The Initial Javascript tab defines the initial javaScript expressions that run before the RAD defined on the RAD tab.

RAD tab

The RAD tab defines the pre-RAD expressions, RAD calls, and post-RAD expressions that run as part of the Process.

The field descriptions for the RAD tab are:

Note: These fields are repeated for each RAD application defined on this tab.

Field Name	**Description**
Expressions evaluated before RAD call *pre.rad.expressions*	Specify the expression to run prior to the RAD application defined in the RAD Application field. All parameter values passed to the RAD application must be in the form of variables or expressions. The variables must be assigned their values in the pre-RAD expressions; for example, $L.value.name="Test".
RAD Application *application*	Specifies the name of the RAD application to run.
Condition *rad.condition*	Specify under which circumstances the RAD application should be executed (condition evaluates to true) or skipped (condition evaluates to false).
Parameter Names *names*	Specifies the parameter names passed to the RAD application
Parameter Values *values*	Specifies the parameter values passed to the RAD application. These values must be in the form of variables or expressions. String values can be passed when enclosed in double quotes ("Wizard Name").
Post RAD Expressions *post.rad.expressions*	Specifies any RAD expression that will run after the RAD application completes.

Final Expressions tab

The Final Expressions tab defines the final expressions that run after the RAD tab processing completes. The final expressions are written using standard Service Manager expressions.

Final Javascript tab

The Javascript defined on this tab runs after the final expressions and after the RAD applications on the RAD tab.

Next Process tab

This tab specifies the next Process or Processes to run when the current Process completes.

The field descriptions for the Next Process tab are:

Field Name	**Description**
Next Process *next.process*	Specifies the name of the next Process to run.
Condition *process.condition*	Specifies a condition associated with the Process in the Next Process field that evaluates to true or false. For example, true for the sm.close Process.

Chapter 5

Document Engine resources

The Document Engine includes local variables and base functions that can be used by any Object. The base functions are defined out-of-box to execute standard actions that are available to the user. For example, when a user clicks the save button after updating a record, the Document Engine processes this request and writes the changes to the database. A base function can be overwritten in the State record to run a different Process.

DEFAULT Object

Any file accessed by the Database Manager automatically uses the rules and processes defined for the corresponding Object. If no Object has been defined for the file, it uses the DEFAULT Object. The DEFAULT Object duplicates the functionality of the earlier versions of the Database Manager.

A System Administrator may also access a file that has a corresponding Object via the DEFAULT Object by accessing the file through Database Manager with the “Administration Mode” check box marked. This check box is available for System Administrators only. A user with the capability word of "AlwaysAdmin" will always use the DEFAULT Object when accessing information within Service Manager. Assigning the capability word to a user is not recommended.

The Document Engine uses environment profiles. In previous versions of the system, a System Administrator was granted all rights (add, update, delete.), regardless of the format control settings. With the Document Engine, the rights granted match those defined in the format control record. Administrators that prefer the old method of granting rights can simply modify the “Profile Application” setting in the DEFAULT Object record from “db.environment” to “db.environment.sysadmin”.

RAD applications

With a database driven application, a user may be working with any of the following three basic record sets:

- Zero records – when searching for information.
- Many records – when looking at a list of information.
- One record – when making changes to a single record.

From a user perspective when working with data:

- you either view a blank form, where you can enter search information (or create a new record),
- or you are working on updating / viewing a single record,
- or you did a search and now have a set of multiple records displayed as a list.

The Document Engine uses three main RAD applications to mirror this idea. The RAD routine used is determined by the number of records displayed at any time.

se.search.engine

The search engine is used when there are no records being viewed. The main purpose of this routine is to formulate a query and select records from the correct table. This routine may also be used for the initial entry of information into a blank record for the purpose of adding a new record to the database. The Default State is db.search.

se.list.engine

The list engine displays multiple records. Using the list engine a user may select a specific record from the list, or perform actions on the entire list of records. Default State is db.list.

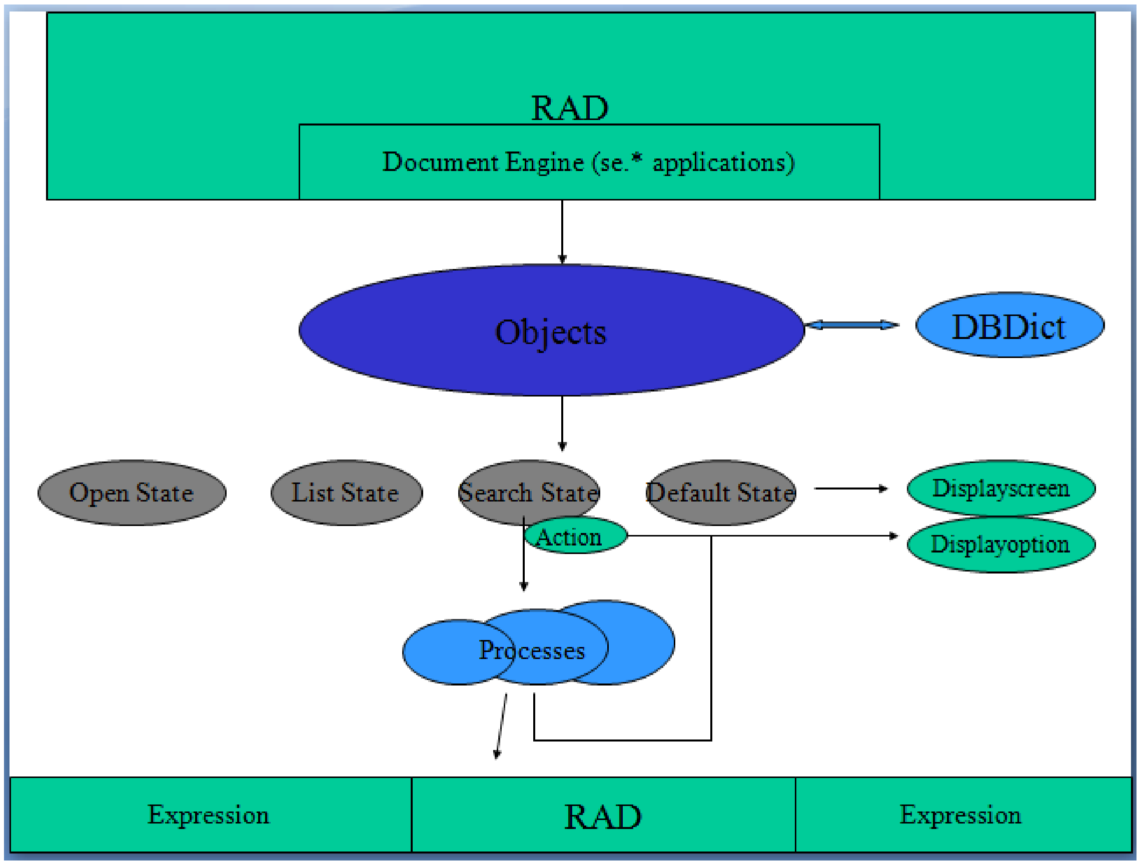
se.view.engine

The view engine displays a single record. This application is used to perform actions against a specific record, such as updates or deletes. Default State is db.view.

Note: When using Service Manager’s record list functionality, the view engine is used for both the list and single record information.

RAD applications flow

The following figure provides an overview of how the other tailoring tools interact with the Document Engine.



Note: All RAD applications related to the Document Engine start with "se", such as se.view.engine. The document engine treats every record as a document made available to the user.

When a file is displayed using the Document Engine it will always be using one of the three applications specified earlier in this document. The display screen, display options, and format are determined by the current State of the record. When a display option is triggered, after the standard display functionality is performed, the engine checks the “action” of that option against the available processes defined in the current State record. If there is a process defined that has a condition that evaluates to true, the engine then performs that process against the current record (or record set). If the action is not defined in the State record, the engine will then check to see if that action is defined as a “Base Process” in the current application. If it is, then the system will perform that base process against the record. If not, no action is taken.

Integration tips for display application

If a display screen does not have any display events associated to it, the system automatically uses the se.default display event that performs se.lock.object in case of “OnFormModified”.

If a display option record has the “Modifies Record” check box activated, the record is locked when the button is pressed.

Base functions in se.search.engine

Display Action	Description
back	Exits the record.
find	Displays detailed information from the record linked to the field.
fill	Fills information from a linked record into a target record. Puts data from a source record field into a target field.
advanced	Starts the Advanced Search process that gives the user more search options. Typically, this is only available for system administrators, but this option can be made available to other users.
clear	Clears the current screen.
openinbox / inbox	Prompts the user to open a view associated with the current file. This is a predefined query that displays results as a view.
search	Performs a standard search by creating a search query using the information provided and then displays the results in a record list.
add	Performs “add” format control and attempts to add a record to the database.
restore	Restores the contents of the screen (after a “clear” is performed).
irquery	Runs an IR Query text search.
validitylookup	Performs Service Manager validity lookup. Validity lookup verifies via an entry in the validity table that a specific value is valid for the defined field. Using validity lookup, the system verifies that a user-entered value complies with the validation rules.
expandarray	Expand array provides additional functionality for editing arrays, such as insert line and delete line.
reset	Resets the current file. This action deletes **ALL** records in the database for the table.
regen	Regens the indexes of the current file. This action applies only to tables that have an IR Key, and it will perform only an IR regen.

Display Action	Description
export/unload	Starts the “no records” export/unload process. Used for unloading empty DBDICT definitions without data only.
views	When selected, the system presents to the user a list of alternate forms in which to display the current record.
findrevision	Displays revisions for records in this object.
initrevision	Creates a revision of a record in this object.
newsite	
newview	
newTable	
addFilter	Triggered by customer performing an Advanced Filter search in the standard search screen.
editFilter	Triggered by customer performing an Advanced Filter search in the standard search screen.
addCompound	Triggered by customer performing an Advanced Filter search in the standard search screen.
removeSelection	Triggered by customer performing an Advanced Filter search in the standard search screen.

Base functions in se.view.engine

Display Action	Description
save	Performs update in Format Control and updates the record in the database if the update is valid.
add	Performs add in Format Control and adds a new record to the database if the new record is valid.
ok	If the record has changed, perform a save, otherwise exit.
reselect	Re-selects the current record from the database (in case it was changed).

Display Action	**Description**
fill	Fills information from a linked record into a target record. Puts information from a source field into a target field.
find	Displays detailed information from the record linked to the field.
next	Move to next record in the list (after checking for changes).
previous	Move to previous record in the list (after checking for changes).
back	Exit to list or search form.
menu	Exits to the calling menu.
delete	Performs delete Format Control and attempts to delete record from the database after validating the request.
views	When selected, the system presents to the user a list of alternate forms in which to display the current record.
print	Prints the current record.
printlist*	Prints the current list of records.
validitylookup	Performs Service Manager validity lookup. Validity lookup verifies via an entry in the validity table that a specific value is valid for the defined field. Using validity lookup, the system verifies that a user entered value complies with the validation rules.
export/unload	Standard Service Manager export/unload functionality (multiple record).
massunload*	Provides unload functionality for a list of records.
massadd*	Adds a new set of records that exactly duplicates an existing set of records except that new set of records will have a unique key value updated with new data. At the end of the processing there will be twice as many records in the table.
massupdate*	Provides the same updates to a set of specified fields in a list of records.
massdelete*	Deletes a specified set of records from a table.
irquery	Runs an IR Query text search.
expandarray	Expand array provides additional functionality for editing arrays, such as insert line and delete line.
count*	Counts records in the record list and displays the total count.

Display Action	Description
audithistory	Calls the standard audit history routine. The audit history is determined by the fields to audit as defined in the auditdef table.
inbox.save / inbox*	Saves the current query as a view.
approval.log	Views the approval history for the current record.
alert.log	Views the alert history for the current record.
alerts	Views the current and scheduled alerts for the current record.
pagelist / listpages	The page list shows a complete history of a record. When paging is enabled, every time an update is performed, a copy of the record is written to the page file. The page list shows the complete history of a record based on the paging file.
clocks	Displays clock records associated with this record.
xmlfill	Handles XML fields, such as the user options in Service Catalog.
createTemplate	Creates a Template record out of the current record.
applyTemplate	Applies an existing Template to the current record.

Note: * These commands only apply if the record list functionality is being used.

Base functions in se.list.engine

Display Action	Description
exit (or back)	Returns to the search (or calling) form.
inbox.save / inbox	Prompts the user to save the current query as a new view.
count	Performs standard count functionality and displays the total record count.
refresh	Refreshes the list using the current query.
big.green	Completely exits the current module. “Big Green Arrow”
print	Prints the list of records displayed.
views	When selected, the system presents to the user a list of alternate forms in which to display the current record.

Display Action	Description
export/unload	Standard Service Manager export/unload functionality (multiple records).
massadd	Adds a new set of records that exactly duplicates an existing set of records except that new set of records will have a unique key value updated with new data. At the end of the processing there will be twice as many records in the table.
massupdate	Provides the same updates to a set of specified fields in a list of records.
massdelete	Deletes a specified set of records from a table.

Local variables

Local variables begin with $L. and persist only within the currently executing RAD application. The server cleans up local variables automatically when it exits a RAD application.

The following is a list of Standard Variables used with the Document Engine:

$L.action - the display action value from the display option

$L.bg - Background flag

$L.category - The category record (if available)

$L.env - The current environment record

$L.exit - internal exit parameter

$L.file - The current file variable

$L.file.save - A copy of the record in its original state

$L.format – name of the format used to display the record

$irspread – determines the IR discovery options: 0=shallow search, 2=deep search, 4=complete match

$L.mode – the mode the viewed record is in, typically add to create a new record, update to modify an existing record or close to finish processing of an existing record

$L.mult - Flag that is true if there are multiple records in the $L.file variable

$L.object - The object record

$L.phase - The phase record (if available)

$L.sql or $L.query- the current query

$L.sort - the current sort order

$L.state – The state record the system is using (=the state the record is in).

Variables that are available in View mode (when viewing a single record)

$L.fc – copy of the detail FormatControl record

$L.fc.master – copy of the master FormatControl record

Chapter 6

Troubleshooting overview

To successfully troubleshoot the Document Engine you will need to gather the following information:

- What dbdict and Object is being used
- What State is the record in
- What is the Process being called
- Steps followed to reproduce (STR) the problem

Research application path through the Document Engine

In troubleshooting the Document Engine, as with troubleshooting any Service Manager application, enter **RTM:3** and **debugdbquery:999** in the Service Manager `sm.ini` file and then start a new client connection. Unless this user process is the very first to invoke the Document Engine processes to debug, it may not show the selection of the State or Process records in the sm.log file when doing this trace, but it will give helpful hints as to which Process was being invoked.

Find dbdict or Object used

To determine which dbdicts or Object is being used, search the log file.

Log sample:

```
1320 07/18/2006 11:00:36 RADTRACE 20 [ 1] se.get.object get.object select CPU( 0 1411 )
1320 07/18/2006 11:00:36 (0x0129AC08) DBACCESS - Cache Find against file Object found 1
record, query: file.name="pcsoftware"
1320 07/18/2006 11:00:36 RADTRACE 20 [ 1] se.get.object set.access process CPU( 0 1411 )
```

Find State for the records

The next question to look at is which State the record is in. To find that information, search for the following in the trace sm.log:

Log sample:

```
1320 07/18/2006 11:00:48 RADTRACE 10 [ 1] se.get.state select.state select CPU( 0 1491 )
```

`1320 07/18/2006 11:00:48 (0x01292FB0) DBACCESS - Cache Find against file States found 1 record, query: state="pcs.list"`

```
1320 07/18/2006 11:00:48 RADTRACE 10 [ 1] se.get.state exit.normal process CPU( 0 1491 )
```

Find the name of the Process

The name of the Process can be found as well, by searching in the `sm.log` containing the trace shown in the following example.

Log sample:

```
1320 07/18/2006 11:00:50 RADTRACE 20 [ 1] se.call.process select.process select CPU( 0
1542 )
```

`1320 07/18/2006 11:00:50 (0x00B56810) DBACCESS - Cache Find against file Process found 1 record, query: process="upgrade.pcs"`

```
1320 07/18/2006 11:00:50 RADTRACE 20 [ 1] se.call.process run.pre.exp process CPU( 0 1542
)
```

Research application errors

Processes call a number of RAD applications and execute a number of expressions, with a possibility of invoking more Processes afterwards. If any of the applications or expressions caused an error exit due to wrong syntax or wrong logic, this information can be found in the sm.log file.

Log sample:

```
Process panel run.pre.exp in RAD se.call.process encountered error in line 1
(se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Bad arg(2) oper = (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Bad arg(2) oper = (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Bad arg(2) oper nullsub (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Bad arg(2) oper in (se.call.process,run.pre.exp)

Cannot evaluate expression (se.call.process,run.pre.exp)

Unrecoverable error in application: se.search.objects on panel call.list.engine

Unrecoverable error in application: se.list.engine on panel call.process.1

Unrecoverable error in application: se.call.process on panel run.pre.exp
```

In this example (not out-of-box), the error occurred in se.call.process, run.pre.exp, or in other words, while evaluating the initial expressions of the Process. To find out which Process was causing the issue, go through the steps outlined above and note the process from the line:

```
DBACCESS - Cache Find against file Process found 1 record, query: process="upgrade.pcs"
```

Go to the Process record by the name of **upgrade.pcs** and check for any statements on the initial expressions tab. In this specific case, the expression will include the word nullsub. For example, the expression in question for this test may be

```
$L.icount=nullsub($L.icount, anynumberIwant)
```

The variable, anynumberIwant, is not a valid field, literal or variable, so it will have to be changed to prevent this issue.

Print values of variables or results of expressions

In the Document Engine, the path through a work flow is often determined by the value assign to a field or variable. To determine the value assigned to a field or variable that influences the work flow, use the `JavaScript print()` function or use the `log rtecall ($L.void=rtecall (“log”, $L.rc, “message”)` in the RAD expressions. The message can be a concatenated string such as

```
$L.message="The value of $L.test is " + $L.test
```

where `$L.test` is a variable that was assigned a character value.

Chapter 7

Work order example overview

What is a work order? A work order is a specific task assigned to a single engineer that identifies some activity necessary to resolve an incident or possibly any other module if the work order system were extended to include changes, problems, or known errors. The following example is based onService Manager version 7.11 and was written to enable users to create, update, and close work orders for Incident Management. It can be modified easily for another Service Manager application.

This example creates a work order system to demonstrate how to use the Document Engine. This work order system will enable users to create work orders for an incident and view the status of these work orders. The work order system also enables users to view and update a work order from Incident Management. All work orders are associated with a particular incident in the system, and the incident cannot be closed until all work orders for the incident are closed.

This example is intended for users who tailor or customize the system. You should have a good understanding of the following tailoring functions:

- Database Dictionary to create a new table
- Forms Designer to modify out-of-box forms and create new forms.
- Wizard creation tool

The work order example walks you through the following steps:.

- Create a new database dictionary (dbdict) using the dbdict utility.
- Specify the key fields for the table.
- Create a EXWorkOrder form for the EXWorkOrder table using Forms Designer.
- Create a Sequential Number File.
- Modify the form to include drop-down lists.
- Create a link for the EXWorkOrder form.
- Create a form for the wizard to use to collect information for the work order.
- Modify the close and update incident forms
- Create aliases to link work orders to incidents
- Create State definitions.
- Create Display Application Screen Definitions for open, close, and view.

- Create Display Application Option Definition of Add, Cancel, Fill and Find for open, close and view.
- Modify the Process definition record im.set.close
- Test the work order example

Create a table

Use the dbdict utility to create a new table. In this example you will create a table named EXWorkOrder. Before you can create this table, you need to know what fields are needed and the attributes of the fields.In this example, the table stores the data for work orders associated with incidents.

To create a table:

1. From the System Navigator, click **Tailoring** > **Database Dictionary**.
   The Database Dictionary form opens.
2. Type `EXWorkOrder` in File Name.
3. Click **New**.
4. On the Fields tab, enter the following information after clicking **NewField/Key**.

Field	**Description**
ID	Will be filled from the number file. Type: character
RelatedID	Will be filled by the unique ID of the related record, example: incident number) Type: character
status	Used to store the status of the work order. Type: character
initiator	The operator who opened the work order. Type: character
assignee.name	The operator assigned to the work order. Type: character

description	Used to hold the description of the work to do for the work order. Type:array of character
category	Filled from the category of the related record. Type: character
RelatedCIs	Used for the list of Configuration Items (CIs) associated with the work order. Type: array of character
impact	Type: character
urgency	Type: character
priority	Type: character
closure.code	Type: character
deadline	Date by which the work order must be completed and filled on open. Type: date/time
est.finish	The assignee's estimate for completion. Type: date/time
update.action	. Type: array of character
closure.comments	. Type: array of character

5. After you have added the fields to the table you need to add keys to the table. You should not exit Database Dictionary utility until you have done this for the **EXWorkOrder** table. See Add key fields to table for the detailed information.

Add key fields to a table

Use the dbdict utility to add keys to a new table. After you create a table named EXWorkOrder, you need to add keys to the table. Keys are used to enable indexed searches as well as ensure data consistency.

To add keys to a table:

1. From the The Database Dictionary click the **Keys** tab.
2. Select the first available entry for a new key field.
3. Click the **New Field/Key**.

4. Enter the following information for each key you create or edit.

Field	**Description**
ID	Type: unique
RelatedID	Type: no nulls
RelatedCIs	Type: nulls & duplicates
assignee.name	Type: nulls & duplicates

5. Click **OK**.

Create a form

Go to Forms Designer to create a form for the EXWorkOrder table. Utilizing the wizard to create the form select Detail of a single record and proceed. This creates a basic form with all of the fields which can be modified as needed. Below is an example for the EXWorkOrder form.

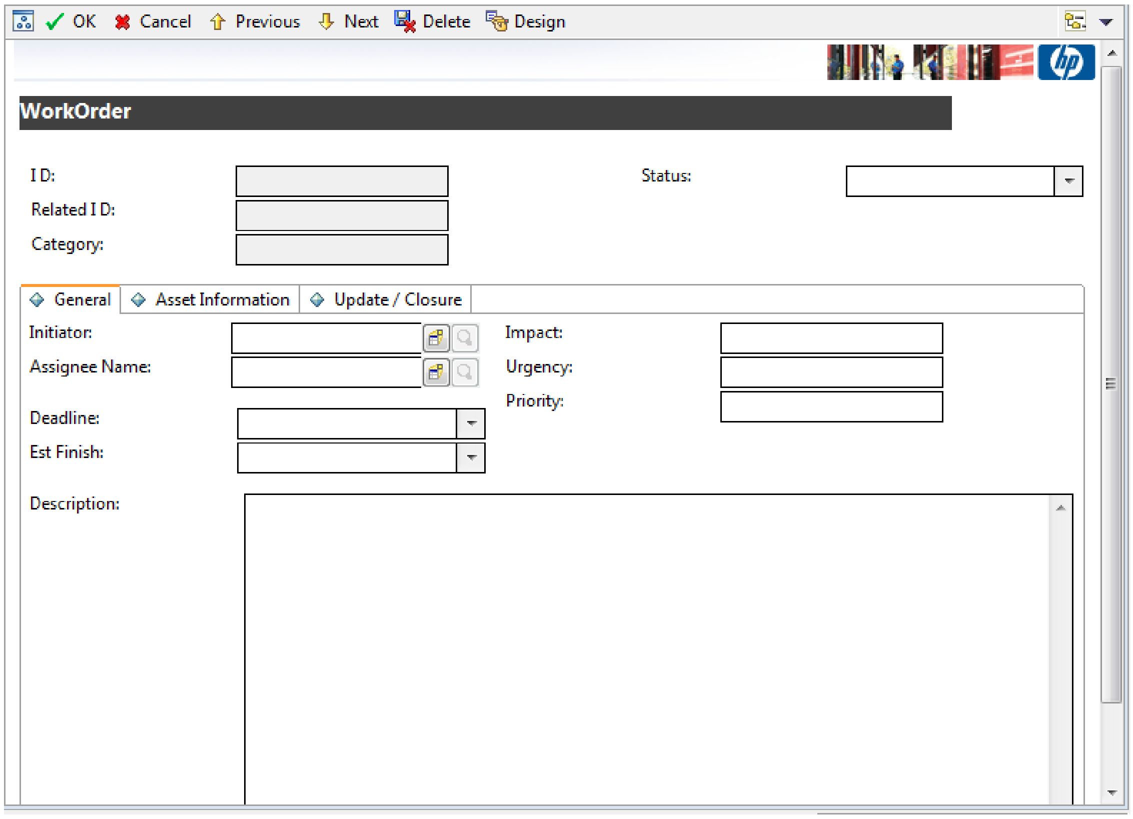
To create a form:

1. From the System Navigator, click **Tailoring** > **Forms Designer**.
   The Forms Designer form opens.
2. Type `EXWorkOrder` in Form Name.
3. Click **New**.
4. Click **Yes** to use the Form Wizard.
5. Type `EXWorkOrder` for the name of the table for which to create a form.
6. Select **Detail of a Single Record** for the type of form you want to create.
7. Click **OK**.
8. Click **Proceed** to accept the defaults for the fields to include on the form.
9. Use the Forms Designer tool to modify the form layout. Below is an example of the **EXWorkOrder** form.
10. The values for Status are:
    - New (default status)

- Open
- Ready
- Closed

11. The Values for the Closure Code are:
    - Implemented
    - Canceled
    - Rollback

The following is a sample EXWorkOrder form.



WorkOrder

I D:		Status:	
Related I D:			
Category:			

Initiator:		Impact:	
Assignee Name:		Urgency:	
		Priority:	
Deadline:			
Est Finish:			
Description:			

Create a copy of a form

In Forms Designer create the sc.manage.WorkOrder form by making a copy of sc.manage.problem. Modify the table column inputs to use the fields from the EXWorkOrder table.

To create a new form from a copy of a form:

1. From the System Navigator, click **Tailoring** > **Forms Designer**.
   The Forms Designer form opens.
2. Type `sc.manage.problem` in Form.
3. Click **Search**.
4. Select sc.manage.problem.g.
5. Click **Copy/Rename** in the detail Options menu.
6. Type `sc.manage.WorkOrder` in New Name
7. Click **OK**.
8. In Forms Designer, update the input fields for the columns using the EXWorkOrder table.
9. - Incident ID - ID
   - Category - category
   - Related Id - RelatedID
   - Status - status
   - Assignee - assignee.name
   - Description - description,1
   - Priority - priority
   - Impact - impact
   - Urgency - urgency
10. Click **OK**.

Create a link for the work order form

One of the advantages of a relational database is the elimination of redundant information. You accomplish this by storing information about a particular subject in one place or table that has links to other subjects. Links are a combination of data and link definitions with sets of conditions containing the relationships for linked information.

To create a link for the work order form:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Links**.
   The Link File form opens.

2. Type `EXWorkOrder` in Name.
3. Add a description in Description.
4. Click **New**.
5. Enter the following information.

Source Field Name	Target File Name	Target Field Name
initiator	operator	name
assignee.name	operator	name
RelatedCIs	device	logical.name

6. Select the initiator row (highlight) then right click **initiator** and choose **Select Line**.
7. In the link.structure.g form enter the following information for initiator. Then repeat Step 6 for **assignee.name** and **RelatedCIs** and use the following information for those fields, respectively.

Source Field (Fill To/Post From)	Target Field Name (Fill To/Post From)
initiator	name
assignee.name	name
RelatedCIs	logical.name

8. Click **Save**.
9. Click **OK**.

Create a sequential number file

Create sequential number file to generate sequence numbers for the records in the EXWorkOrder table.

To create a sequential number file:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Sequential Numbers** The Sequential Number File form opens.
2. Type `EXWorkOrder` in Class.

3. Type `1` in Last Number.
4. Type `Number for WorkOrder` in Description.
5. Type `5` in Length.
6. Type `WO` in Prefix.
7. Click **Add**.

Create an Object definition

The purpose of this Object is to define the characteristics and behavior of the EXWorkOrder Object that determines what data needs to be included in a work order record and how the system will process work orders.

Note: See "Object Definition form and fields" on page 11 for the Object definition field descriptions.

To create an Object definition:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **Objects**.
   The Object Definition form opens.
2. Type *`EXWorkOrder`* in File Name.
3. Click **Add** to create the Object record.
4. On the Object Info tab, enter the following information.

Field	**Value**
Unique key	Automatically filled (ID)
Common name	Automatically filled (EXWorkOrder)
Description field	Table to hold work orders that can relate to any of the modules
Profile application	db.environment
Profile variable	$L.env
Number record name	EXWorkOrder
Category table name	category
Master format control	EXWorkOrder
Status field	status

Assigned to fields	assignee.name
Open state	EXWorkOrder.open
Close state	EXWorkOrder.close
Default state	EXWorkOrder.view
Search state	EXWorkOrder.search

5. Click **Save**.
6. Select the **Manage Queues** tab.
7. On the Manage Queues tab, enter the following information.

Field	**Value**
Manage condition	true
Manage display format	sc.manage.WorkOrder
Manage default query	assignee.name=operator()
Allow add condition	false
Default query description	My WorkOrders

8. Click **Save**.
9. Click **OK**.

Create an Initialization Process definition

These Process definitions for the work order example specifies what initial expressions and RAD applications to use when a user opens a work order record.

To create a Process definition for initialization:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **Processes**.
   The Process Definition form opens.
2. Type `EXWorkOrder.open.initial` in Process Name.
3. Click **Add**.

4. Type `$L.format="EXWorkOrder"` the Initial Expressions tab.
5. Click **Save**.
6. Enter the following information in the RAD tab.

Field	**Value**
Expressions evaluated before RAD call	$L.number.record="EXWorkOrder";$L.number.type="string"
RAD Application	getnumb
Parameter Names	**Parameter Values**
name	$L.number.record
index	ID in $L.file
text	$L.number.type

7. Click **Save**.

Create display application screen definition

You create the display application screen definitions for open, close, and view screens for the work order forms that allow users to open, close, and view work order records.

To create a screen application definition for open, close, and view:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Display Screens**.
   The Display Application Screen Definition form opens.
2. Type `EXWorkOrder.open` in Screen ID.
3. Enter the following information:

Field	**Value**
Screen ID	EXWorkOrder.open
Title	Open New Work Order
Format	$L.format
I/O (If RIO)	true
On option 0:	redraw screen

Language	ENG

4. Click **Add** and then click **OK**.

For this example you also need to create display application screen definitions for a close screen.

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Display Screens**.
   The Display Application Screen Definition form opens.

2. Use the following information:

Field	**Value**
Screen ID	EXWorkOrder.close
Title	Close Work Order
Format	$L.format
I/O (If RIO)	true
On option 0:	redraw screen
Language	ENG

3. Click **ADD** and then click **OK**.

For this example you need to create display application screen definitions for view screen also.

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Display Screens**.
   The Display Application Screen Definition form opens.

2. Use the following information:

Field	**Value**
Screen ID	EXWorkOrder.view
Title	View New Work Order
Format	$L.format
I/O (If RIO)	true
On option 0:	redraw screen
Language	ENG

3. Click **Add** and then click**OK**.

Create display application option definitions

This procedure provides step-by-step instructions for creating add, cancel, fill, and find display application option definitions for the following WorkOrder definitions :

- open
- close
- view

In doing this, you will repeat the same steps four times for each of the WorkOrder screen definitions (open, close, view). However, each time you enter different values in the fields on the Display Application Option Definition form. The tables below the steps provide the values you need to configure the display application options.

For additional information about the display application options, see the online help topics for the display application on the HP Service Manager online help server.

After you create the display application screen definitions for open, close, and view, you need to create display application options for the open, close, and view screens.

To create a display application option definition:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Display Options**.
   The Display Application Screen Definition form opens.
2. Type `EXWorkOrder.open` in Screen ID.
3. Enter the following information.

Field	**Value**
Unique ID	Automatically generated (EXWorkOrder.open_add)
Action	add
GUI option	4
Text Option	4
Bank	1
Condition	true
Default Label	Add

4. Click **Add**.
5. Click **OK**.

6. Repeat steps 1 - 6 for each of the following display option definitions using the values provided in the table.

Note: In some cases the value required for the Action field is not available in the drop-down list, but you can type the applicable value in the field.

Open - Cancel

Field	**Value**
Screen ID	EXWorkOrder.open
Unique ID	Generated by the system. (EXWorkOrder.open_cancel)
Action	back
GUI option	3
Text Option	3
Bank	1
Condition	true
Default Label	Cancel

Open - Fill

Field	**Value**
Screen ID	EXWorkOrder.open
Unique ID	Generated by the system. (EXWorkOrder.open_fill)
Action	fill
GUI option	9
Text Option	9
Bank	1
Condition	true
Default Label	Fill

Open - Find

Field	**Value**

Screen ID	EXWorkOrder.open
Unique ID	Generated by the system. (EXWorkOrder.open_find)
Action	find
GUI option	8
Text Option	8
Bank	1
Condition	true
Default Label	Find

Close - Cancel

Field	**Value**
Screen ID	EXWorkOrder.close
Unique ID	Generated by the system. (EXWorkOrder.close_cancel)
Action	back
GUI option	3
Text Option	3
Bank	1
Condition	true
Default Label	Cancel

Close - Close

Field	**Value**
Screen ID	EXWorkOrder.close
Unique ID	Generated by the system. (EXWorkOrder.close_close)
Action	close
GUI option	5
Text Option	5

Bank	1
Condition	true
Default Label	Close

Close - Fill

Field	**Value**
Screen ID	EXWorkOrder.close
Unique ID	Generated by the system. (EXWorkOrder.close_fill)
Action	fill
GUI option	9
Text Option	9
Bank	1
Condition	true
Default Label	Fill

Close - Find

Field	**Value**
Screen ID	EXWorkOrder.close
Unique ID	Generated by the system. (EXWorkOrder.close_find)
Action	find
GUI option	8
Text Option	8
Bank	1
Condition	true
Default Label	Find

View - Cancel

Field	**Value**

Screen ID	EXWorkOrder.view
Unique ID	Generated by the system. (EXWorkOrder.view_cancel)
Action	back
GUI option	3
Text Option	3
Bank	1
Condition	true
Default Label	Cancel

View - Fill

Field	**Value**
Screen ID	EXWorkOrder.view
Unique ID	Generated by the system. (EXWorkOrder.view_fill)
Action	fill
GUI option	9
Text Option	9
Bank	1
Condition	true
Default Label	Fill

View - Next

Field	**Value**
Screen ID	EXWorkOrder.view
Unique ID	Generated by the system. (EXWorkOrder.view_next)
Action	next
GUI option	10
Text Option	10

Bank	1
Condition	true
Default Label	Next

View - Find

Field	**Value**
Screen ID	EXWorkOrder.view
Unique ID	Generated by the system. (EXWorkOrder.view_find)
Action	find
GUI option	8
Text Option	8
Bank	1
Condition	true
Default Label	Find

View - Save

Field	**Value**
Screen ID	EXWorkOrder.view
Unique ID	Generated by the system. (EXWorkOrder.view_save)
Action	save
GUI option	4
Text Option	4
Bank	1
Condition	true
Default Label	Save

Create a State definition

These State definitions for the work order example specifies what process to use and what actions are allowed when a user opens, closes, or views a work order record.

Note: See "State Definition field descriptions" on page 25 for the State definition field descriptions.

To create a State definition for open:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **States**.
   The State Definition form opens.

2. Enter the following information:

Field	**Value**
State	EXWorkOrder.open
Display Screen	EXWorkOrder.open
Initialization Process	EXWorkOrder.open.initial
Format	$L.format
Input Condition	true

3. Click **ADD**.

To create a State definition for close:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **States**.
   The State Definition form opens.

2. Enter the following information:

Field	**Value**
State	EXWorkOrder.close
Display Screen	EXWorkOrder.close
Initialization Process	EXWorkOrder.close.initial
Format	$L.format
Input Condition	true

3. Click **ADD**.

To create a State definition for view:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **States**.
   The State Definition form opens.

2. Enter the following information:

Field	Value
State	EXWorkOrder.view
Display Screen	EXWorkOrder.view
Initialization Process	EXWorkOrder.view.initial
Format	$L.format
Input Condition	true

3. Click **ADD**.

To create a State definition for search:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **States**.
   The State Definition form opens.

2. Enter the following information:

Field	Value
State	EXWorkOrder.search
Display Screen	db.search
Initialization Process	EXWorkOrder.view.search
Format	$L.format
Input Condition	true

3. Click **OK**.

Add a close work order button

When a user completes the task or tasks, the work order should be closed. This procedure describes how to put a close button on the EXWorkOrder form and update the status of the work order to close. This procedure adds a display application option definition, adds a State definition for EXWorkOrder.view, and adds a Process definition for EXWorkOrder.close.

Note: This procedure does not explain how to include validation for this activity. You would use Format Control to add validation.

To create a display application option definition for EXWorkOrder.view:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Display Options**.
   The Display Application Screen Definition form opens.

2. Type EX`WorkOrder.view` in Screen ID.

3. Enter the following information.

Field	**Value**
Unique ID	Automatically generated (EXWorkOrder.view_close)
Action	close
GUI option	5
Text Option	5
Bank	1
Condition	true
Default Label	Close

4. Click **Add**.

5. Click **OK**.

To update the EXWorkOrder.view state definition:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **States**.
   The State Definition form opens.

2. In the State field, type **EXWorkOrder.view** and then click **Search**. The EXWorkOrder.view state definition form opens.

3. Add the following to update the state definition form.

Field	**Value**
Display Action	close
Process Name	EXWorkOrder.close
Condition	true

4. Click **Save** and then click **OK**.

To add a process definition record for EXWorkOrder.close:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **Processes**.
   The Process Definition form opens.
2. Type `EXWorkOrder.close` in Process Name.
3. Click **Add**.
4. On the Initial Expressions tab, type the following expression:
   - status in $L.file="Closed"
   - $L.mode="closed"
5. Click **Save**.
6. On the RAD tab, enter the following information:

Field	**Value**
Expressions evaluated before RAD call	$L.EXaction="update"
RAD Application	se.base.method
Condition	true
Parameter Names	**Parameter Values**
file	$L.file
prompt	$L.EXaction
second file	$L.file.save
record	$L.fc
second.record	$L.object
boolean1	false

7. Click **Save** and then **OK**.

Create a wizard for the work order

This examples uses a wizard to create a work order because a wizard simplifies the task of opening a work order for the user and the functionality is already available in the system.

This example uses the wizard creation tool to create a wizard to allow users to create a work order from the Incident Management module. When a user creates a work order from an incident, the wizard prompts the user for information and auto-fills some of the fields in the work order.

To create a wizard for the work order form:

1. From the System Navigator, click **Tailoring** > **Wizards**.
   The Wizard Information form opens.
2. Type `Create Workorder - 1` in Wizard Name.
3. Click **Add** to create the Wizard record.
4. On the Wizard Info tab, enter the following information:

Field	**Value**
Brief Description	Create a new work order.
Window Title	Create Workorder
Title	Create Workorder
Start Node	Select (set to true) to indicate that this is the first wizard in a series of wizards in cases where you have a series of wizard records.

5. Click **Save**.
6. In the File Selection tab of the Wizard Information form, select the **Select $L.file by** tab.
7. On the Select $L.file by tab, enter the following information:

Field	**Value**
Create a record	true
of type	EXWorkOrder

8. Click **Save**.
9. On the Usage tab, enter the following information:

Field	**Value**
Wizard Usage	Click **Request user input** in the Wizard Usage section.
Sub Format to Display	Type createWO.assigneeAndCIs
Display Screen	Type wizard.display
Activate "Finish" option	Select (set to true) to have a Finish button on the wizard form.

10. Click **Save**.

11. On the Actions tab, enter the following information:

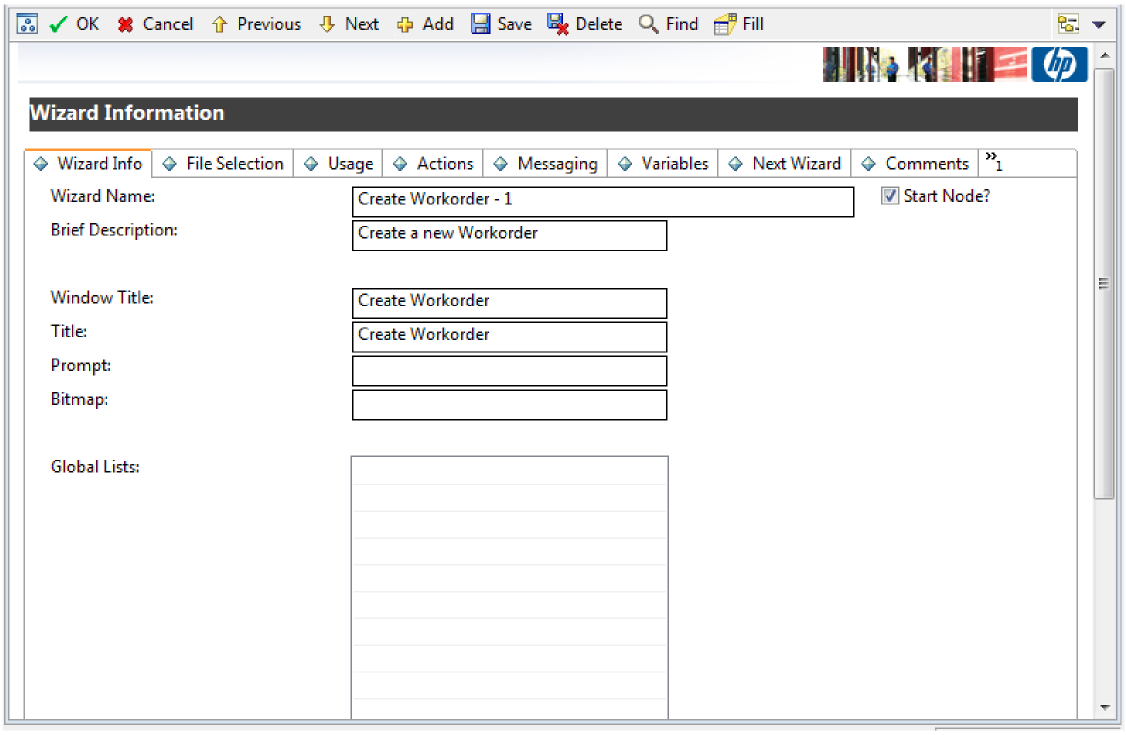
Field	Value
Perform Actions On	Select Current File ($L.file).
Actions to Perform > Expressions	initiator in $L.file=operator() status in $L.file="New" RelatedID in $L.file=number in $relatedRec category in $L.file=category in $relatedRec impact in $L.file=initial.impact in $relatedRec urgency in $Lfile=severity in $relatedRec priority in $L.file=priority.code in $relatedRec
Display Record(s) when complete?	Click **Display Record(s) when complete?**
Mode	Select Add.

12. Click **Save**.

13. On the Cancel Expressions tab, enter the following information:

Field	Value
Expressions Executed on Cancel	cleanup($relatedRec)

14. Click **Save** and then **OK**.



Wizard Name:	Create Workorder - 1
Start Node?	✓
Brief Description:	Create a new Workorder
Window Title:	Create Workorder
Title:	Create Workorder
Prompt:	
Bitmap:	
Global Lists:	

Add a Process definition record

The Process definition record defines how the system responds to a user's action. The Process definition uses RAD expressions, JavaScript, and calls to existing RAD applications to perform actions against the current record, in this case, a work order record.

To add a process definition record:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **Processes**
   The Process Definition form opens.

2. Type **create.WorkOrder** in Process Name.

3. Click **Add**.

4. On the Initial Expressions tab, type the following expression:
   - **$L.void=fduplicate($relatedRec, $L.file)**
   - **$relatedCIs={}**

5. Click **Save**.

6. On the Initial Javascript tab, type the following expression:

```
system.vars.$relatedCIs=system.library.BSGFunctions.getMembers
(system.vars.$L_file.affected_item, false, 3)
```

Note: This expression must be entered on one line.

7. Click **Save**.
8. On the RAD tab, enter the following information:

Field	Value
Expressions evaluated before RAD call	Type the following two expressions: $L.wiz.name="Create Workorder-1" if (not null(logical.name in $L.file)) then ($relatedCIs=insert ($RelatedCIs, 1, 1, logical.name in $L.file)) **Note**: Be sure to enter this expression on one line. Also note that when entering this expression, there is no space after the word insert. For example, the expression above continues at the word 'insert' as follows: insert($RelatedCIs, 1, 1, logical.name in $L.file))
RAD Application	wizard.run
Parameter Names	name
Parameter Values	$L.wiz.name
Parameter Names	text
Parameter Values	$L.exit
Condition	true

9. Click **Save**.
10. On the Final Expressions tab, type `cleanup($relatedRec)`.
11. Click **Save** and then **OK**.

Create an input form for the wizard

This form is the first form to display in the work order wizard. The user enters the information the user needs to provide for the wizard to create the work order record. Use Forms Designer to create this form. For this example, the form name is createWO.assigneeAndCIs.

To create an input form for the wizard:

1. From the System Navigator, click **Tailoring** > **Forms Designer**.
   The Forms Designer search/create new form opens.

2. Type **createWO.assigneeAndCIs** in the Form for the form name.

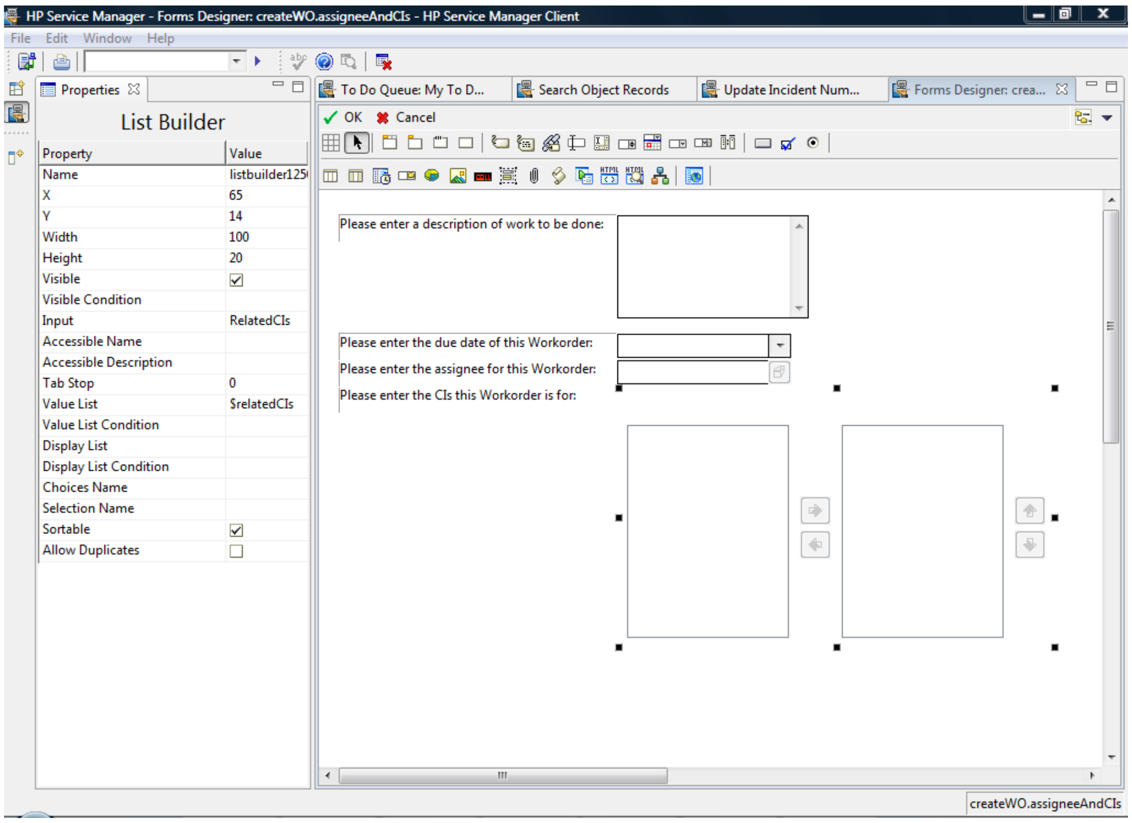
3. Click **New**. You do not need to use the Forms Designer wizard for this form.

4. Create the following input fields on the form:
   - Description (Please enter a description of the work to be done)
   - Due date (Please enter the due date for this work order)
   - Assignee (Please enter the assignee for this work order)
   - CIs (Please enter the CIs this work order is for)

5. The Properties for this form should include:
   - Input: **RelatedCIs**
   - Value List: $**relatedCIs**
   - Sortable: checked

6. Click **Save**.

The following figure shows a sample createWO.assigneeAndCIs form for the EXWorkorder wizard input form.



List Builder

Property	Value
Name	listbuilder125
X	65
Y	14
Width	100
Height	20
Visible	✓
Visible Condition	
Input	RelatedCIs
Accessible Name	
Accessible Description	
Tab Stop	0
Value List	$relatedCIs
Value List Condition	
Display List	
Display List Condition	
Choices Name	
Selection Name	
Sortable	✓
Allow Duplicates	☐

Please enter a description of work to be done:

Please enter the due date of this Workorder:

Please enter the assignee for this Workorder:

Please enter the CIs this Workorder is for:

Modify the close and update incident forms

For the work order example, you need to modify the close incident and update incident form so that you can view the work orders assigned to the incident and then double click and read or edit the work order from the incident. Use Forms Designer to update the *IM.update.incident* and *IM.close.incident* form. To do this you must add a tab and then a form on the tab on both the IM.update.incident and IM.close.incident forms.

To modify update incident form:

Note: Use these same general procedures to update the close incident form.

1. From the System Navigator, click **Tailoring** > **Forms Designer**.
   The Forms Designer search form opens.

2. Type **IM.update.incident** in the Form for the form name.

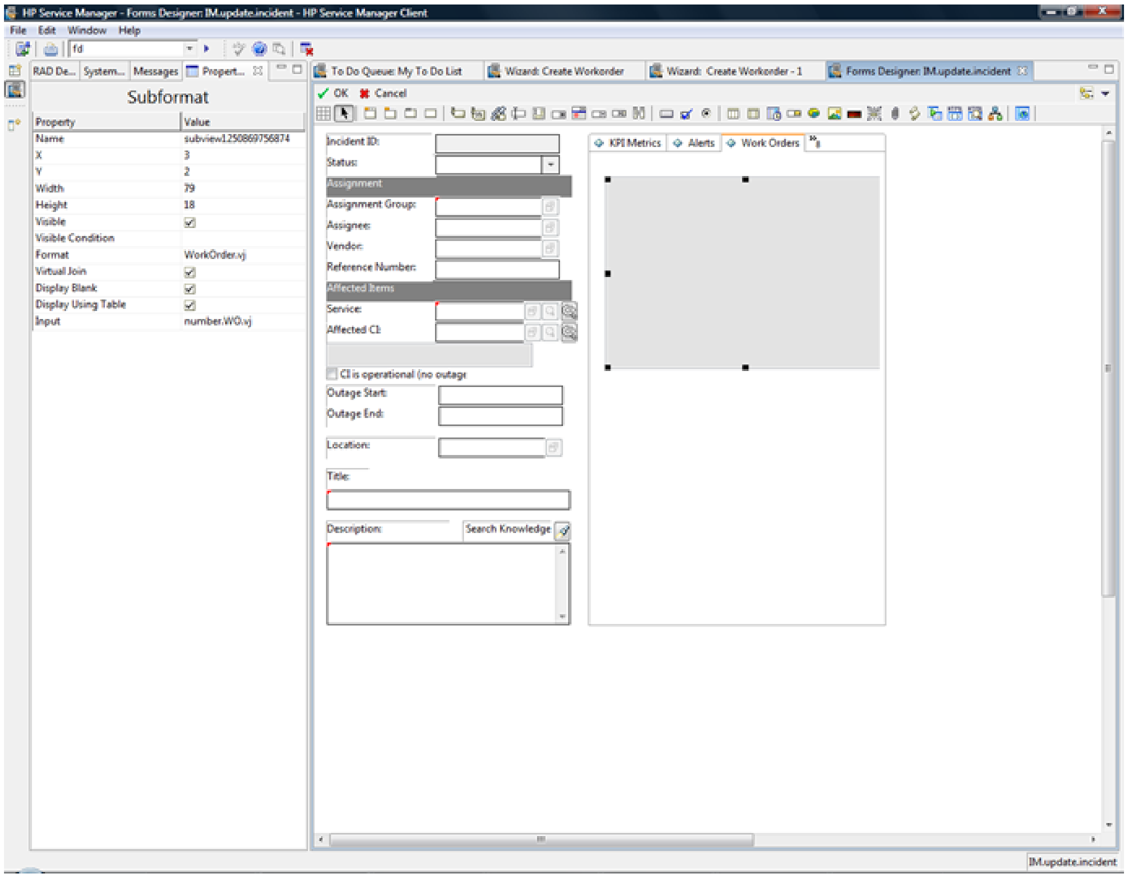
3. Click **Search**.

4. Click **Design**.

5. Add a new tab to the existing notebook.
6. Set the caption of the tab to WorkOrder.
7. Add a subformat with the following properties:
   - Visible: checked
   - Format: WorkOrder.vj
   - Virtual Join: checked
   - Display Blank: checked
   - Display Using Table: checked
   - Input: number.WO.vj
8. Click **Save**.
9. Use the Forms Designer tool to create the WorkOrder.vj form:
   - WorkOrderID: ID
   - WorkOrder Deadline: deadline
   - WorkOrder Status: status
   - WorkOrder Assignee: assign.name
   - WorkOrder Description: description
10. Repeat steps 1 - 8 for the **IM.close.incident**.

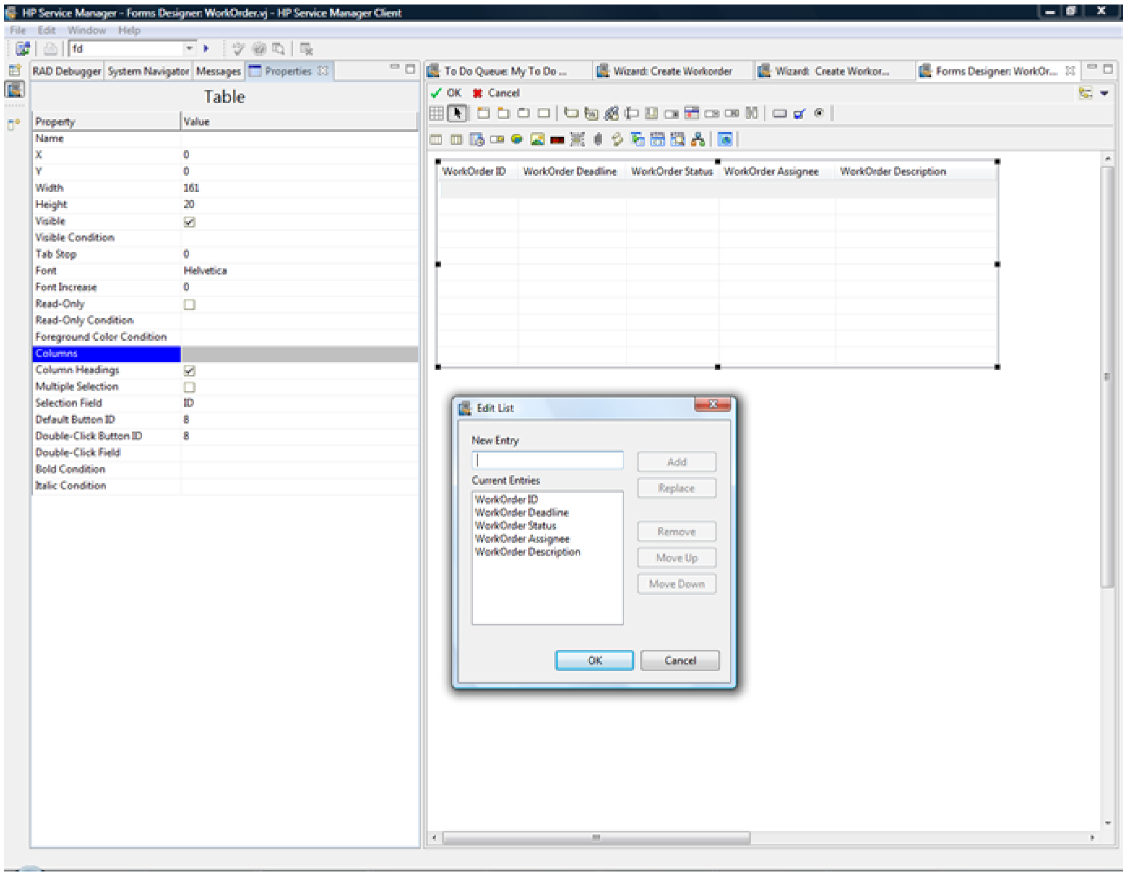
    Note: You can use the WorkOrder.vj form created in step 9 for the virtual join.

The following figures show samples of the tabs to add.



Subformat

Property	Value
Name	subview1250869756874
X	3
Y	2
Width	79
Height	18
Visible	✓
Visible Condition	
Format	WorkOrder.vj
Virtual Join	✓
Display Blank	✓
Display Using Table	✓
Input	number.WO.vj



Table

Property	Value
Name	
X	0
Y	0
Width	161
Height	20
Visible	✓
Visible Condition	
Tab Stop	0
Font	Helvetica
Font Increase	0
Read-Only	☐
Read-Only Condition	
Foreground Color Condition	
Columns	
Column Headings	✓
Multiple Selection	☐
Selection Field	ID
Default Button ID	8
Double-Click Button ID	8
Double-Click Field	
Bold Condition	
Italic Condition	

Edit List

New Entry

Current Entries

WorkOrder ID
WorkOrder Deadline
WorkOrder Status
WorkOrder Assignee
WorkOrder Description

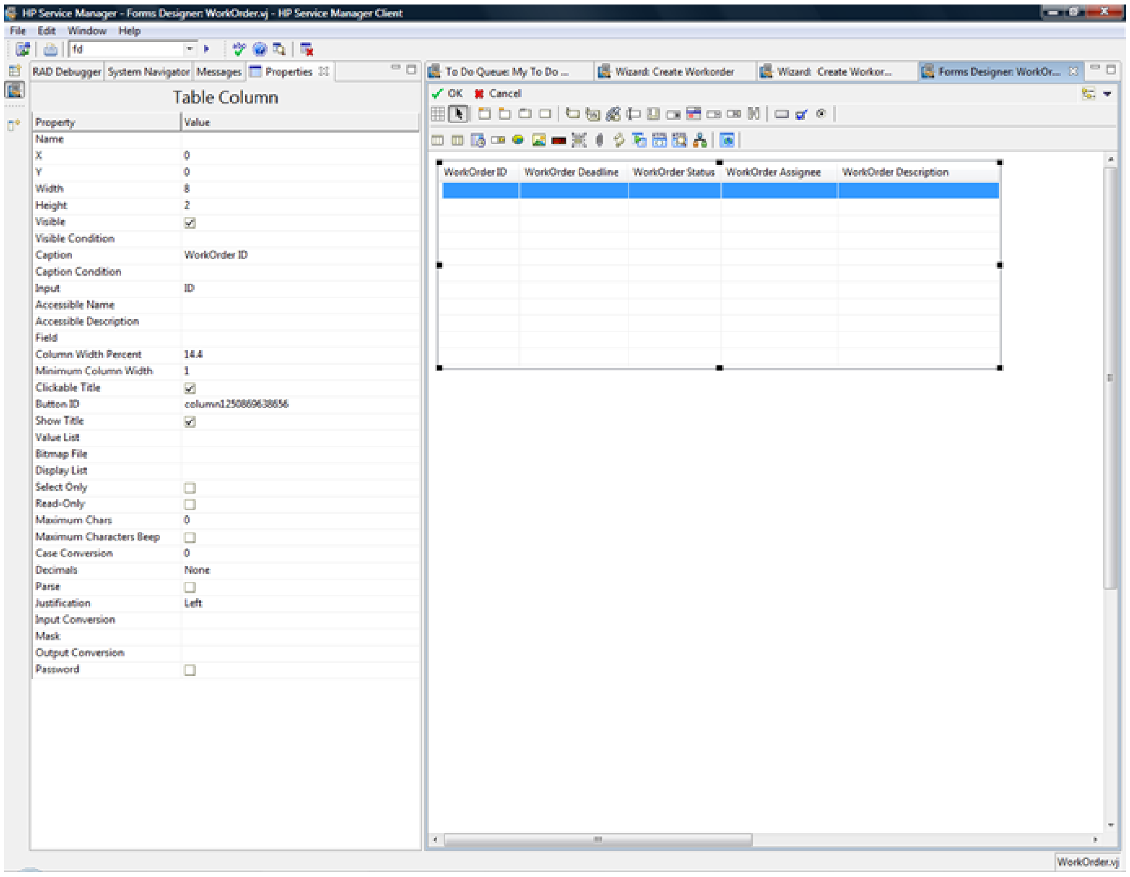


Table Column

Property	Value
Name	
X	0
Y	0
Width	8
Height	2
Visible	✓
Visible Condition	
Caption	WorkOrder ID
Caption Condition	
Input	ID
Accessible Name	
Accessible Description	
Field	
Column Width Percent	14.4
Minimum Column Width	1
Clickable Title	✓
Button ID	column1250869638656
Show Title	✓
Value List	
Bitmap File	
Display List	
Select Only	☐
Read-Only	☐
Maximum Chars	0
Maximum Characters Beep	☐
Case Conversion	0
Decimals	None
Parse	☐
Justification	Left
Input Conversion	
Mask	
Output Conversion	
Password	☐

Create an alias in probsummary table for link

In the work order example, users need to be able to access a work order from an incident record when the incident has a work order associated with it. In order to do this, you must create an alias in the probsummary table and then use that alias to create a link between the probsummary table and the EXWorkOrder table.

To add an alias to the probsummary table:

1. From the System Navigator, click **Tailoring** > **Database Dictionary**.
   The Database Dictionary form opens.
2. Type `probsummary` in File Name.
3. Click **Search**.
4. On the Fields tab, select the number field and click **Edit Field/Key**.
5. Click **Create Alias** and type `number.WO.vj` for name and character for type.
6. Click **OK**.

To link the EXWorkOrder table with the probsummary table:

1. From the System Navigator, click **Tailoring** > **Tailoring Tools** > **Links.** The Link File form opens.
2. Type `probsummary` in Name.
3. Click **Search**.
4. Click after the last entry to create a blank line for a new entry.
5. In the Source field, type `number.WO.vj`.
6. Select the entire new line and then click **Select Line**.
7. Enter the following information:

Field	**Value**
Field (From/Source)	number.WO.vj
File (To/Target)	EXWorkOrder
Field (To/Target)	RelatedID
Query	$query
Expressions	$query="RelatedID=\""+number in $File+"\""

8. Click **Save** and then **Back**.
9. Repeat steps to create a link line for the ID field.
10. Enter the following information.

Field	**Value**
Field (From/Source)	ID
File (To/Target)	EXWorkOrder
Field (To/Target)	ID
Query	$query
Expression	$query="ID=\""+nullsub(cursor.field.contents(), "xxx")+"\""

11. Click **Save** and then **Back**.

Modify im.set.close Process definition

This process needs to be modified so that a user cannot close an incident if there are work orders still open for the incident.

To modify a process definition record:

1. From the System Navigator, click **Tailoring** > **Document Engine** > **Processes**
   The Process Definition form opens.

2. Type **im.set.close** in Process Name.

3. Click **Search**.

4. On the Initial Javascript tab, type the following JavaScript:
   ```
   var WO=new SCFile ("EXWorkOrder")

   var FoundOpenWO=WO.doSelect ("RelatedID=\""+system.vars.$L_
   file.number + "\""+" and status ~=\"" + "Closed" + "\""

   if (FoundOpenWO == RC_SUCCESS)

   {

   system.vars.$openWO=true;

   }

   else

   {

   system .vars.$openWO=false;

   }
   ```

5. Click **Save**.

6. On the RAD tab, enter the following information:

Field	**Value**
Note: Some of the fields on the RAD tab are pre-filled with values that you do not need to modify	

RAD Application - Select the empty section below the section calling the RAD Application us.consume.wrapper and then enter the following information:	
Expressions evaluated before RAD call	$L.text="There are still open work orders. This incident cannot be closed yet."
RAD Application	apm.mb.ok
Condition	$openWO=true
Parameter Names	text
Parameter Values	$L.text

7. Click **Save**.
8. On the Final Expressions tab, enter the following information:
   - if ($openWO=true) then ($L.exit="badval")
   - $L.exit="closestate"
9. Click **Save** and then **OK**.

Test the work order example

After you complete the tasks to create a work order system, you need to verify that it works correctly. To verify the basic functionality of the work order example, do the following:

- In Incident Manager, find an open incident or create one.
- Use the Options menu or button on the tool bar to create a work order. Create Workorder should display on the Options menu.
- In the Workorder wizard, enter the data to create a work order for the incident.
- Add the work order to the incident and save the changes to the incident.
- Open the incident for editing, and from the WorkOrder tab edit the work order.
- Save your changes.
- Open the incident again, and this time edit the work order and close it.
- You should now be able to close the incident.

- Repeat these steps, but this time create two work orders for an incident.
- Close only one work order and then attempt to close the incident. The system should generate an error message indicating that there are still open work orders for the incident, and therefore it cannot be closed.
- Close all work orders for the incident. You should now be able to close the incident.

We appreciate your feedback!

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Service Manager, 9.32 Document Engine Guide

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to oudoc-ITSM@hp.com.