STANDARD FOR SECONDARY WINDOWS (DIALOGS & WIZARDS)

Secondary windows – boilerplate and possibly a table like this …

Secondary Window	Validation	Action Buttons	Required Pages	Sequenced pages
Single-page dialog	Yes, on exit	OK, Cancel	Yes	NA
Edit… Dialog	Field-level, On exit	OK, Cancel	No	No
Wizard	Field-level, Page-level, On exit	Back, Next Finish, Cancel	At least one	Typical, but not required

Validation in secondary windows

Here we discuss the user experience aspects of validation that occurs on user inputs inside a Settings dialog or a Wizard. Technical aspects of validation can be found in the (see Tech reference). Validation in other vSphere contexts is described in standards sections related to those parts of vSphere. There are three types of validation, corresponding to two variables: 1) at what point in the workflow validation occurs, and 2) what the scope of the validation is.

Field-level validation - occurs when a change has been made to a field, and the cursor is now leaving the field. Field-level validation is used to show a user what fields are required and also what types of inputs a field accepts. It thus prevents a user from making inconvenient mistakes. The scope of a field-level validation is the field itself. Field level validation errors may be "enforced" by disabling the 'Ok' button until they are fixed, or may be ignored if non-destructive. Here is how to visually show field-level validation errors:

Page-level validation – occurs when the user makes an explicit action (button-click) that indicates the entire page is complete. The scope of the validation is the entire page (as when different settings within a page all feed into the validation algorithm, or the contents of the page impact available settings on another page). Another use case for page-level validation is when a lengthy process is required (e.g., reading from or writing to a server over the network).

Validation on 'OK' – occurs when the user clicks the 'OK' button in a dialog or the 'Finish' button in a wizard. The scope of validation on 'OK' is the entire dialog or wizard: all settings. It may entail reading from or writing to a

server over the network. It is also necessary when a user might have made data-destructive mistakes. However, it has some disadvantages and should be used only when necessary. Validation on 'OK' is problematic for users because:

- It is better to see and correct errors as one goes along than to be surprised with them at the point one thinks the task is complete and is ready to exit.
- if multiple errors were committed, providing a whole list of errors is more cumbersome for the programmer and for the user, than indicating errors at field level as they occur.

Types of Secondary Window

Single-page dialog – boilerplate

"Settings" dialog – Overview

The primary purpose of a settings dialog is to make object settings available to a user. In vSphere, all inventory objects and many management-related objects have settings dialogs associated with them. A settings dialog is a collection of property settings presented on multiple pages inside a single window with a single set of exit buttons (one 'Ok' button and one 'Cancel' button).

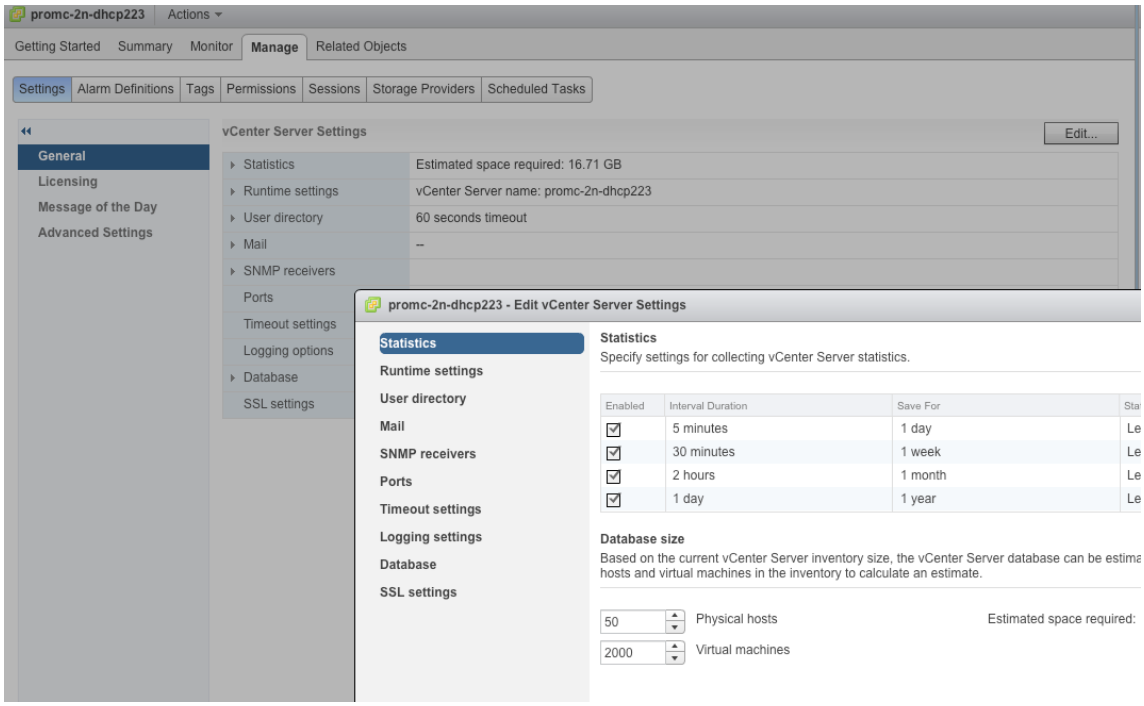
In vSphere, the Settings dialog is usually invoked by entering an object's "Manage" tab, clicking the "Settings" selection-button at the top left, and then clicking the button labeled "Edit…" at the top right.

What settings dialog should look like:

The "Edit…" button opens a dialog with multiple pages representing categories of setting.

A Settings dialog invoked from a main window that shows categories & sub-categories in separate columns.

- **Clicking "Edit…" with a category selected invokes a settings dialog for that category.**
- **Inside the dialog, each sub-category gets a separate page of settings.**



Categories **Sub-Categories** **Dialog with sub-categories mapped to pages (shown as list on left)**

Settings dialog – Behavior

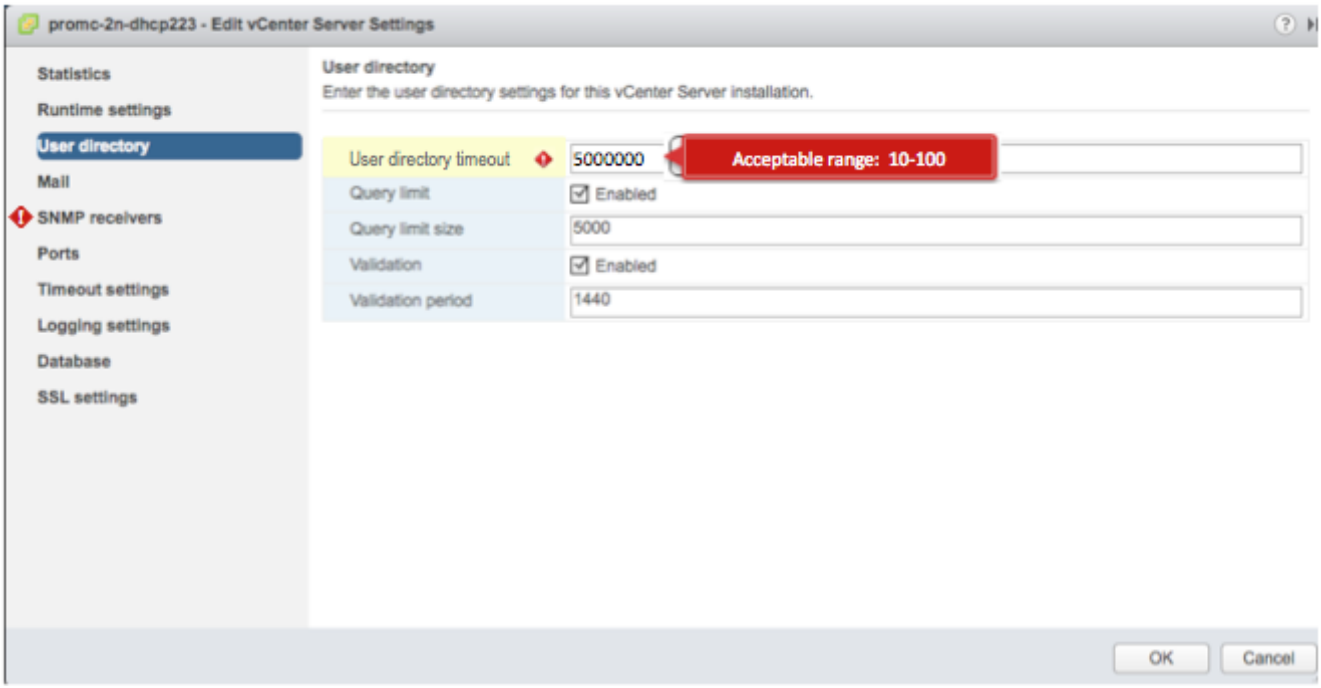
Similar to the "Preferences" dialog of desktop environments and applications, the vSphere Settings dialog allows free navigation between pages. There are no dependencies between pages, so a user can visit any page in any order, and settings made on one page do not affect settings made on another.

The user should also be free to enter and leave the dialog without having to visit any particular pages. That is, the Settings dialog should not contain any pages that MUST be visited in order to exit the dialog and commit changes.

There can be field-level validation, but no page-level validation that would inhibit the ability of a user to navigate freely between pages. This means that a user should be allowed to leave fields incomplete or incorrectly completed on one page, and navigate to other pages inside the dialog.

Required fields, fields with format requirements, and fields with required numeric ranges MUST show errors immediately. Here is how to show field-level errors:

- Turn the field's border red (actually it should have a wide border so that color-deficient users can see)
- Show an error flag with a description of correct inputs. This flag is transient, appearing only on mouse-over.
- Mark the category with yellow fill and a red warning icon (diamond with "!"). This marking is permannt.
- The OK button must remain enabled at all times. This is so that validation-on-OK can be performed.
- Show errors on pages other than the selected page using the "! Diamond" error icon in the table of contents.



Settings dialog – Special Topics

Table of contents – Should have a distinct look from the wizard TOC. Redesign is under way.

Validation - It is at the designer/developer's discretion whether any validation of user inputs will be performed. Many desktop and application Preferences panels do no validation at all. If validation is done, then the following recommendations apply (for more details on how and when to perform validation, see Validation).

Recommendation -> Field-level validation is strongly recommended in all dialogs & wizards.

Recommendation -> Validation on 'OK' is sometimes necessary to prevent a user from making data-destructive mistakes or when inputs must be validated at the server level. However, it should only be used only when necessary.

Recommendation -> Do NOT use page-level validation in a Settings dialog. Page-level validation implies interpage dependencies, which should not exist in the Settings dialog. It blocks navigation to other pages and thus breaks the basic interaction model of a settings dialog. Finally, because the Settings dialog has only a single Ok and Cancel button, the user may become confused about when changes made in a page are committed: on navigating to another page, or on exit? If you must do page-level validation, then use a wizard.

Wizard – Overview

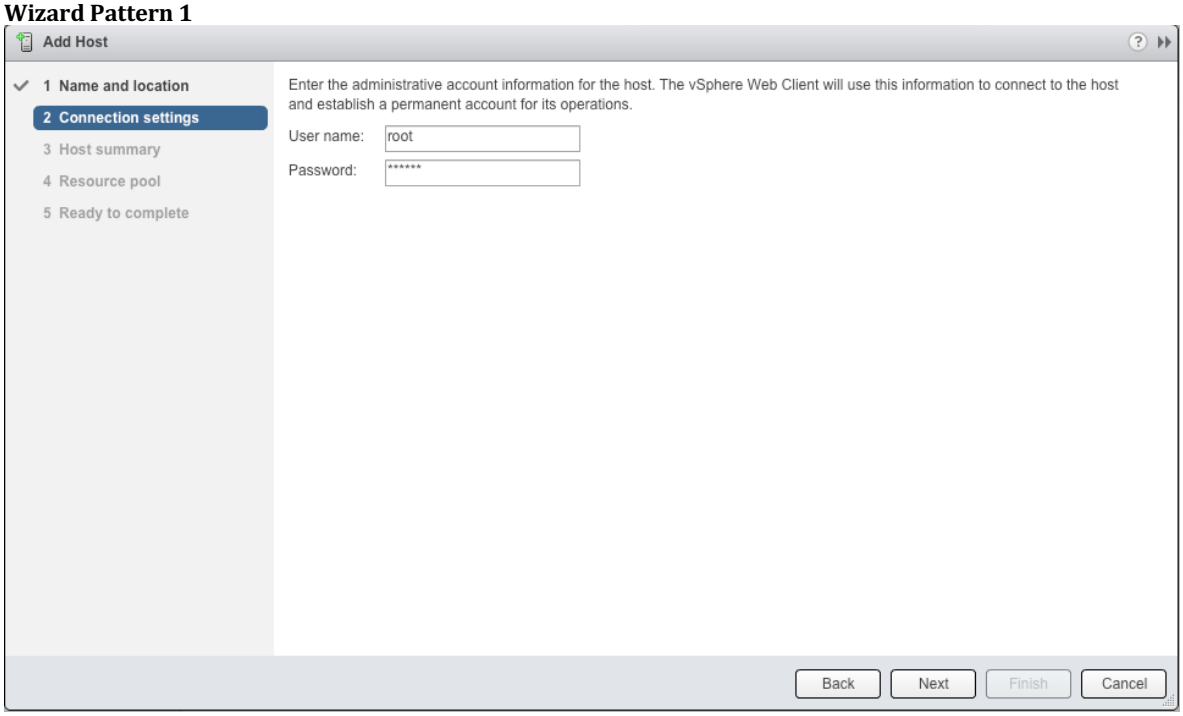
A wizard implements a task or procedure requiring multiple steps. Like the settings in a Settings dialog, similar steps are arranged together in a group of pages all sharing the same window. But UNLIKE the settings dialog, a Wizard restricts the user's behavior to a limited set of choices and procedures.

What a wizard looks like

Wizard implementations in the MN.next release of vSphere offer a wide variety of wizard styles, layouts, behaviors and table-of-contents designs. The first example ("Wizard Pattern 1") shows the main features of a simple wizard:

- Table of contents on left (dimensions discussed at VisualDesignSite, guidelines given BELOW)
- Content area on right
- Buttons at bottom: Back, Next, Finish, Cancel
- Help button at top right
- Minimize to TIWO arrows at top farright

Wizard Pattern 1



Wizard – Behavior:

Action buttons - A wizard has four action buttons: Previous, Next, Finish and Cancel. Together they allow a user to proceed through steps of a procedure without having to make too many decisions. The 'Next' button validates all the fields on the page together, and throws an error if a field is incorrectly populated, if different fields are not coherent, or if information required to move on to the next page is not present.

Controlling User Inputs - Using a wizard, the designer/developer can control:

- which pages *can* be visited, and when (enforcing inter-page dependencies)
- which pages *must* be visited (enforcing required steps)
- page-level validation of inputs

These characteristics make the wizard a good choice in a variety of situations where the designer/developer wishes to protect users from making dangerous (data-destructive) changes to the infrastructure. The Settings dialog lacks these characteristics and so is a poor choice when the designer/developer wants to exercise strong control over user inputs.

Wizard – Special Topics

Table of Contents

Numbering – The reason we number steps is to give users an approximate idea of how long the work flow will take and roughly where they stand in the work flow (e.g., "I've completed 3 out of 6 pages so I'm about halfway through the task"). To ensure that these goals are met, a few simple guidelines should be followed:

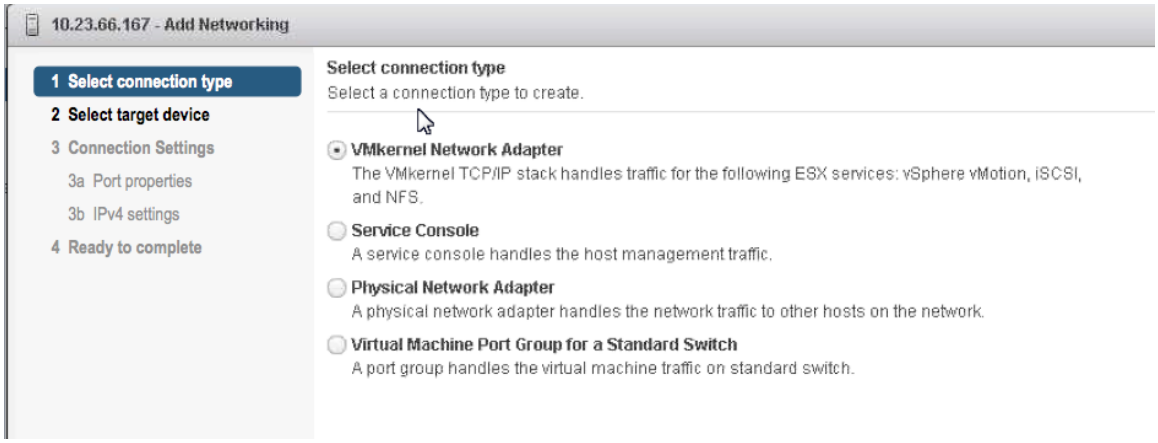
- There should be a 1:1 correspondence between numbered steps and actual wizard pages.
- Avoid sub-steps. Most sub-steps work just as well as main steps.
- Do not use alphabetic extensions except in special cases (described below)

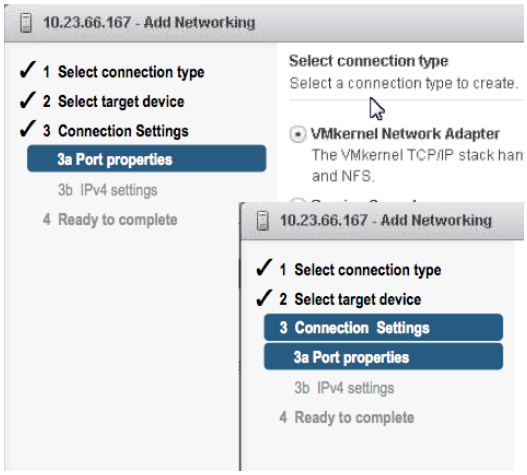
Conceptual headers or grouping labels – Although it might seem helpful to group steps by similarity or sharing "meaning", extra labels in the TOC can distract the user from the work flow. Also, coming up with descriptive & distinctive headers that work in many languages is very difficult.

- Avoid conceptual labels or "headers" that do not correspond to a wizard page.
- If you must use a conceptual header that does not correspond to a wizard page, DO NOT number it.
- If you must use a conceptual header for part of the work flow, you must use headers for ALL the workflow. An example is shown below:

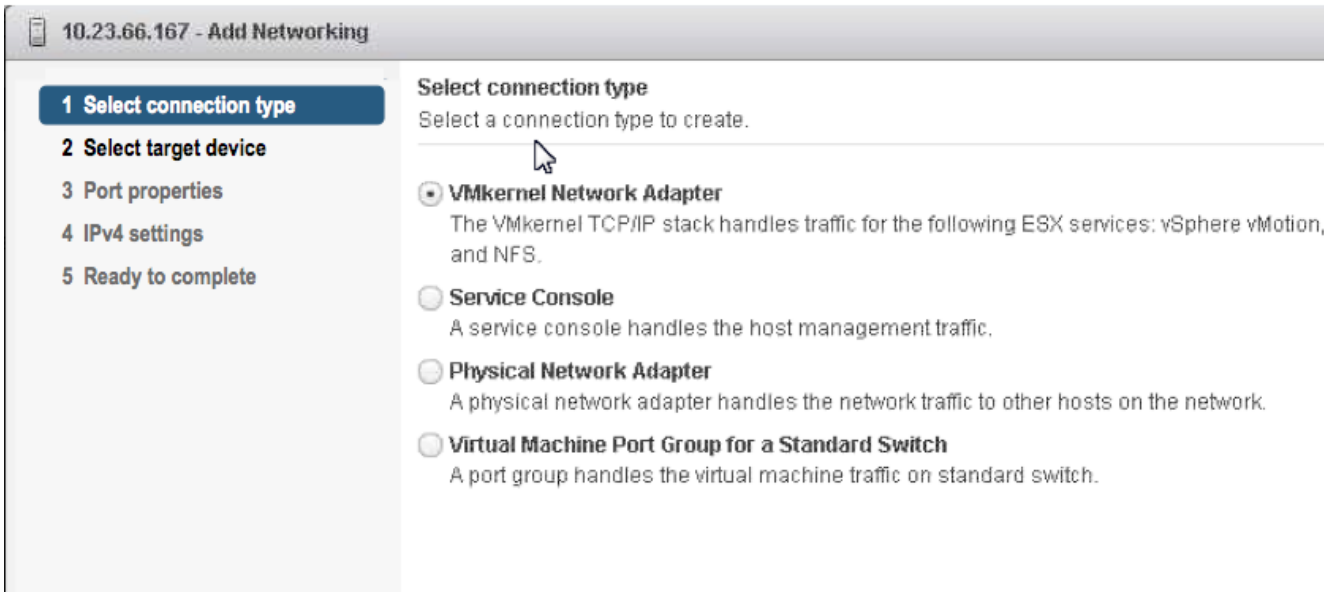
The following incorrect designs illustrate the problems with Numbering & Conceptual headers:

- There are six entries in the TOC, but only 5 actual steps.
- "Connection Settings" is not a step but it gets a number.
- The number "3" appears multiple times, making it difficult to count steps.
- Whether to show the label as part of the current step (or not) is not well defined.
- Whether to show a checkmark by a 'complete' label is not well defined.





Here is the same wizard with a flat list for the TOC. Note that numbering is straightforward and it is easy to see how many steps one has.



Font weight –

- All wizard steps should be shown in **bold** font. Do not use font weight to indicate future steps, optional steps, completed steps .
- Conceptual headers, when used, should be shown in plain font.

Font color – font color is black when a step is available or completed, disabled gray when not available, and white when shown against green ("current step") rectangle

Green rectangle – green rectangle tracks users current page

Check marks – check mark appears next to a step when all the data required for the step is entered.

Tracking & back-tracking – When a user backtracks to formerly completed steps and *makes a change on an already completed page,* the state of the TOC must return to a state reflecting the change. That is:

- All subsequent steps must have their checkmark removed (including any that are optional)
- All subsequent steps must have their font return to plain.
- Any subsequent steps not navigable from the current step must return to disabled gray.
- The green rectangle must return to the chosen backtrack page.

Optional Steps - TBD

Branching & Sub-steps

Branching - Branching occurs when the user provides input that causes a particular workflow to be shown (among others that are shown in response to different user choices). Branching may require the addition of new steps or replacement of some steps with new ones. This behavior can be disorienting to the user and must be handled carefully in the TOC.

- Branching should happen only in response to clear user choice (e.g., Radio buttons).
- If branching occurs due to a server/database query, explanatory text MUST be given in the content area to explain the change in steps.
- Branching should occur as early as possible in the work flow.

S**ub-steps –** Sub-steps should be limited to a small number of cases where a branch has occurred and the new steps created by the branch are tightly bound to each other and loosely bound to the rest of the steps in the work flow. One example would be the case where sub-steps all share a single validation action. Some guidelines for the use of sub-steps and how to show them in the TOC.

Use sub-steps ONLY when:

- A branch has occurred that requires steps not originally shown in the TOC.
  AND
- The steps that were added to the work flow share a single validation action.

Show sub-steps like this:
Begin the sub-steps section with a conceptual header.
End the sub-steps section with a horizontal rule line vertically aligned with step names and extending to [TBD]
Header to be in plain font and out-dented relative to the main steps of the wizard.
Sub-steps to be numbered with to be in line with main steps of the wizard
Add a conceptual header at the top of the TOC to say “Main work flow” (or similar)

Layout & Paging

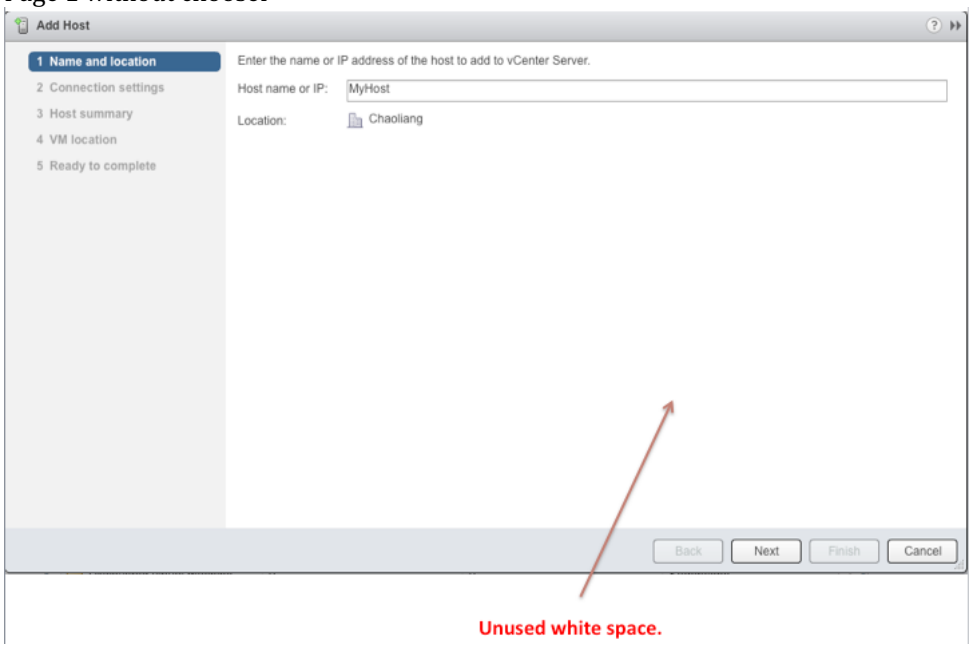
Expert users like VI admins prefer to move through tasks quickly. They become annoyed if they have to click the "Next" button too many times. In order to minimize this source of annoyance, wizards should be designed with as few pages as possible. On the other hand, a page with too many settings can appear cluttered. The goal of the designer/developer should be to find a trade-off between the number of pages and the amount of information per page. The following are rules of thumb, not strict requirements.

Recommendation -> Have at least 4 settings per page, and try to fill at least 75% of the available space with settings. Users are particularly annoyed with a lot of pages when they see that each page is sparsely populated.

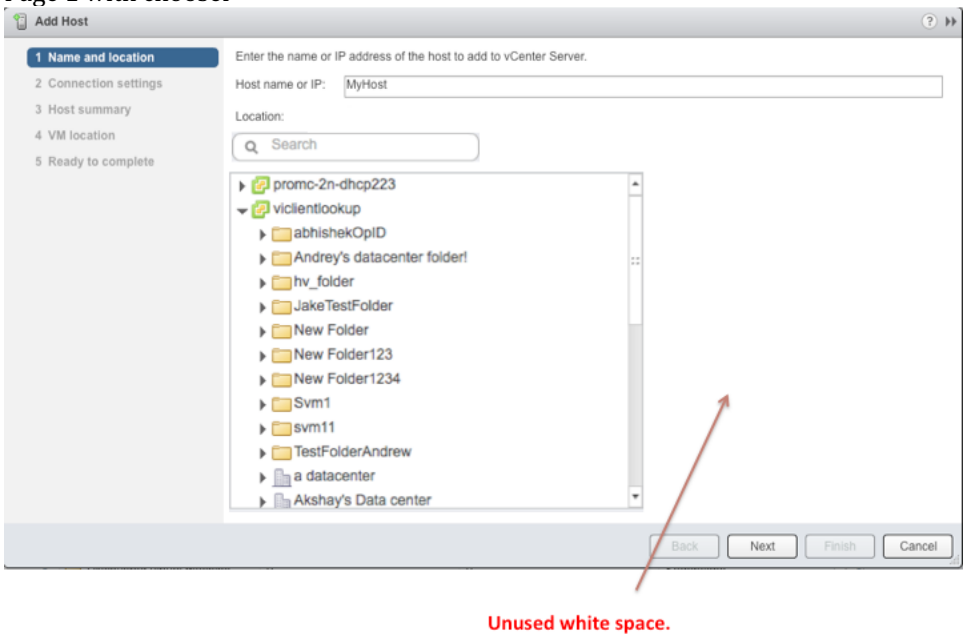
Recommendation -> Group settings together when they can be validated or sent to the server together, and/or when they fit logically together in the work flow of the wizard.

In the wizard shown below, the pages are sparsely populated. Page 1 without a Location chooser is extremely sparse. Even with the chooser added, there is extra white space available.

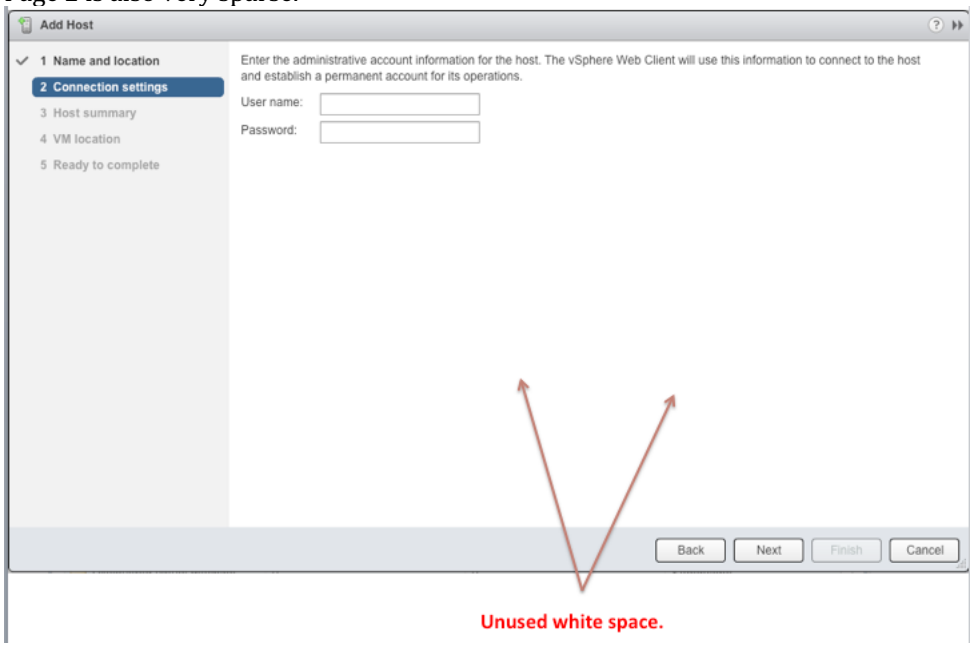
Page 1 without chooser



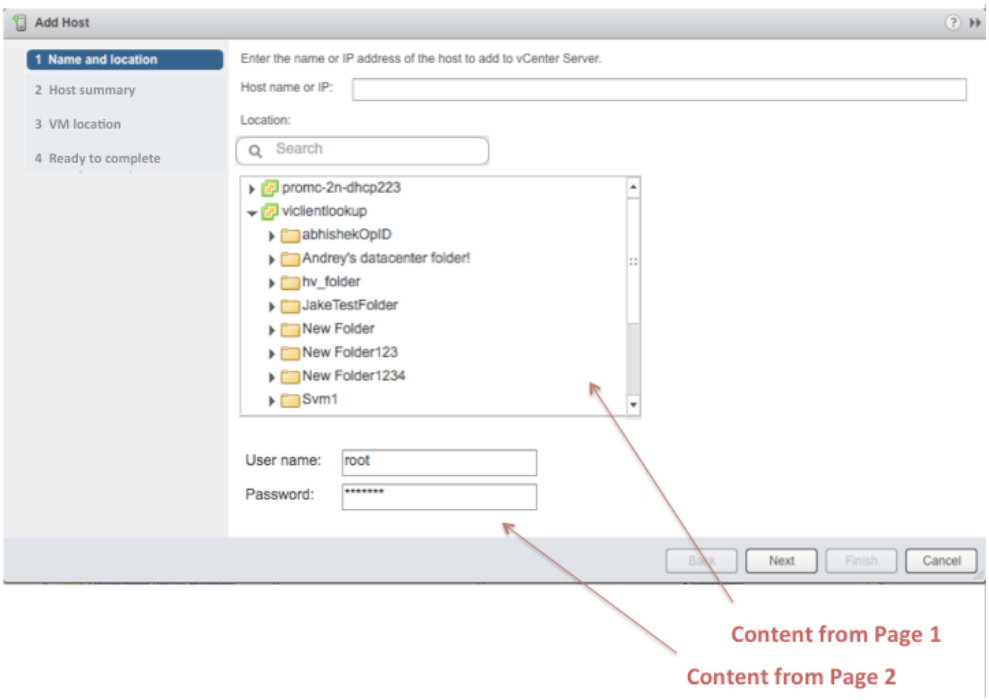
Page 1 with chooser



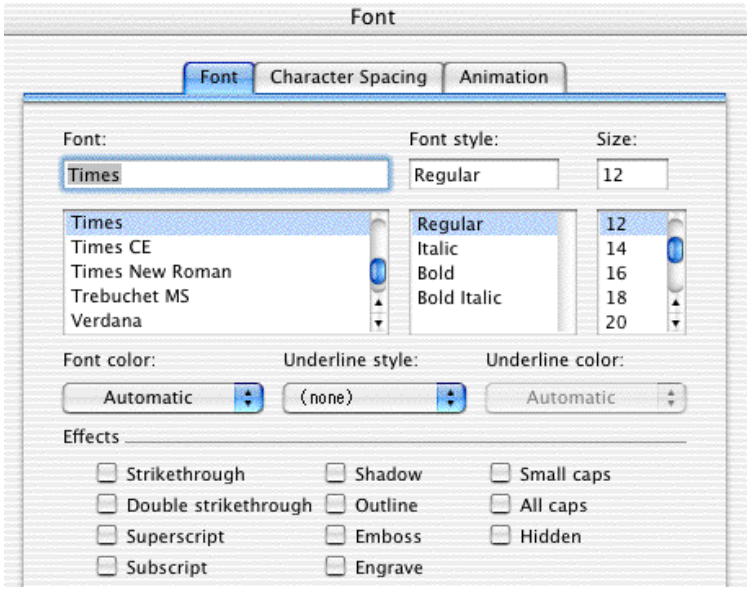
Page 2 is also very sparse.



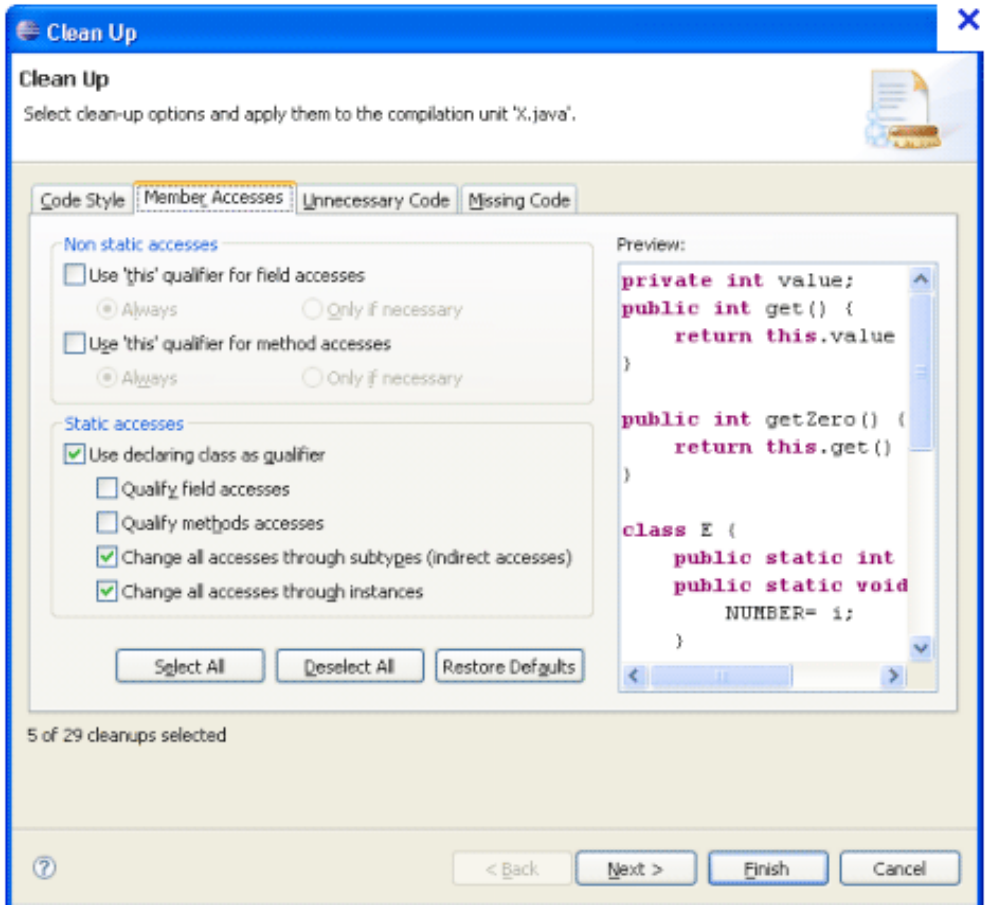
Validation of pages 1 and 2 occurs at the same time (when the "Next" button at the bottom of page 2 is clicked). Therefore, the settings on the two pages could be combined and shown on a single page, reducing the page count by one without adding too much visual clutter.



Here is an example of a non-VMware page with high information density, but not necessarily confusing because most users are familiar with font settings and know how to find the settings they need.



Here is an example of non-VMware wizard page with high information density and unfamiliar settings. It will likely be confusing.



Minimizing a Wizard to the "Things I'm Working On" (TIWO) list

Wizards with the ">>" widget in the top right corner can be minimized without losing work completed. The wizard enters a list on the righthand side of the main console. While a wizard is in this state, the underlying system or inventory can experience changes that might impact the accuracy of information in the wizard. For example, a host that was listed in the wizard may have been taken off line, a VM listed in the wizard might have been moved from the inventory, and so on.
Because of changes like this that can occur, the wizard must go through re-validation when it is reopened from TIWO.

For the user, this can be a confusing experience, learning that the wizard's state has changed, and so it is very important to keep the user informed and to help users to remember where they were in the work flow they were working on, and to regain their orientation with respect to the wizard.

Here are some requirements for the behavior of a wizard on reopening from TIWO

Requirement -> If validation finds that the input on certain wizard pages is no longer valid, the wizard must:

- open to the earliest (lowest numbered) page that is valid
- repaint the table of contents items following that page to show they are incomplete (whether they were completed previously or not).

Requirement -> If validation finds that all input is valid, then it must return to the last page completed by the user before the user minimized it. That is, if a user completed all pages up to and including page 7, then the wizard must return to page 7 on reopen from TIWO.

Requirement -> All fields that failed to validation should be marked as erroneous according to guidelines for field-level validation (described in "Validation" section).

NO PICTURES NEEDED HERE.

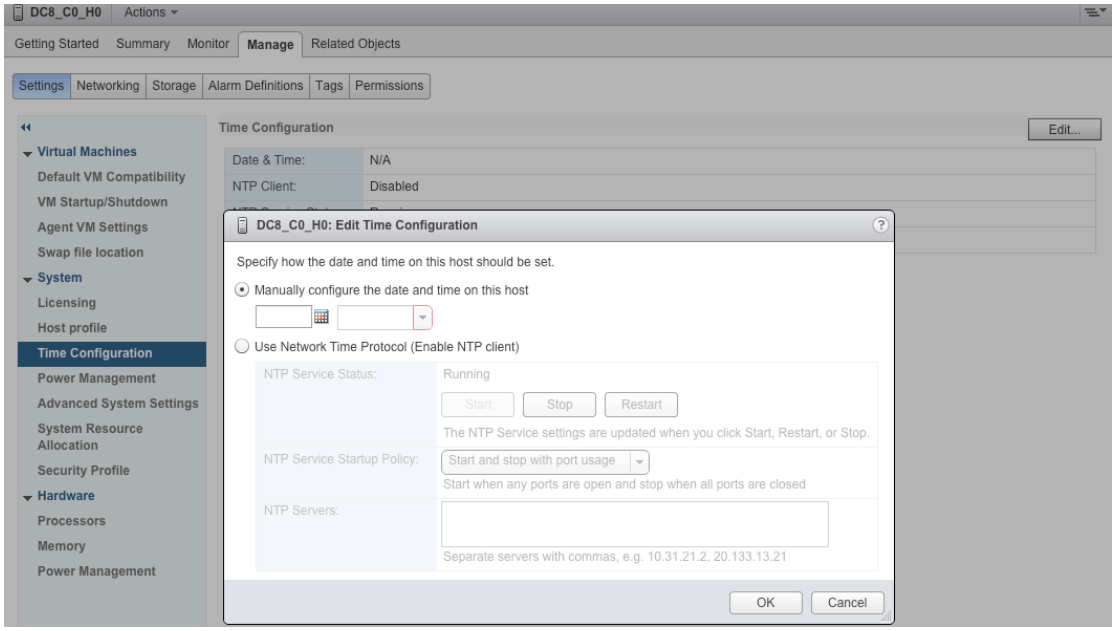
When to use a Wizard versus a Settings Dialog

.

NON-STANDARD IMPLEMENTATIONS
(rejected design patterns)

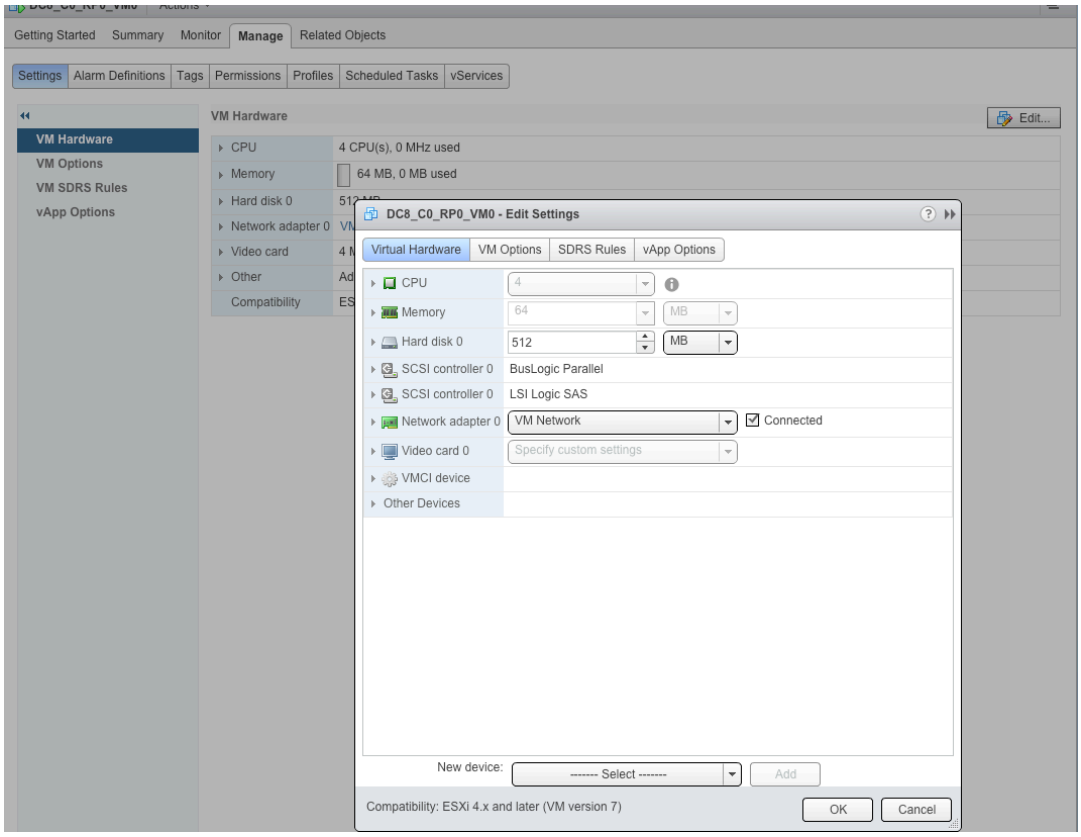
Pattern 1 - some objects have a complete list of settings showing on their "Manage -> Settings" window, and the "Edit…" button opens a single dialog at a time, one for each setting. An example of this is shown as "Design Pattern 1" on the next page.

Design Pattern 1 – A single-page dialog opened from a list of settings
Background page should be redesigned to show categories & sub-categories in separate columns.



Detailed settings list **Single-page dialog**

Design Pattern 2 – A Settings dialog with categories mapped to separate pages inside dialog
NON-standard – Buttons should go away and be replaced with TOC on the left.



Category list **Dialog with categories mapped to pages (shown as buttons at top)**

Wizard Pattern 2