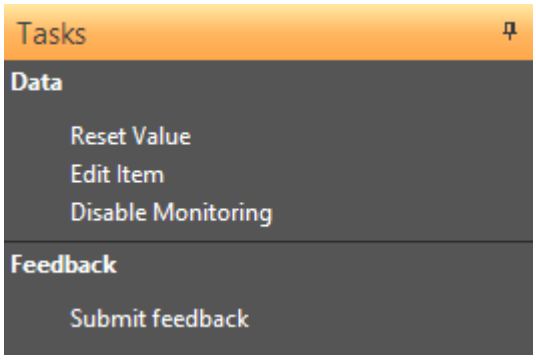
Task Items

Tasks are grouped clickable text items that sit in the Tasks panel in the WMC. Your component should use tasks to provide a uniform method of configuration adjustment.



Adding tasks

Tasks stay visible until they are removed with NavDeleteItem so you can create permanent task items (Like the Submit feedback task above) by not deleting the task and its group. Typically however, you want to add them when your panel becomes active and delete them when the panel is deactivated. See Tree items and panels for information on handling (de)activation events.

To add a task, first insert a task group under the TASKLIST domain with NavInsertItem.

```
// Error handling omitted for brevity
NAVSETITEM taskRootData;
NavGetItem(L"TASKLIST", &taskRootData);     // Get the task root to put our group
under
NavHandle taskRoot = taskRootData.hItem;

NAVSETITEM taskGroupData;
memset(&taskGroupData, 0, sizeof(taskGroupData));
taskGroupData.flags = NI_FLAG_CONTAINER;    // Signals it's a group that will hold
tasks
taskGroupData.strLabel = L"Ad Blocker";
taskGroupData.strDescription = L"Tasks for the Ad Blocker component";
NavInsertItem(taskRoot, L"AdBlockerTaskGroup", &taskGroupData);   // Error handling
removed for brevity
NavHandle taskGroup = taskGroupData.hItem;
```

As with all navigation items, the label passed to NavInsertItem is a free form text field that has to be unique amongst its siblings.

Once the group is inserted, add tasks beneath it in a similar manner.

```
NAVSETITEM settingsTaskData;
memset(&settingsTaskData, 0, sizeof(settingsTaskData));

settingsTaskData.flags = 0;
settingsTaskData.EventMask = NI_EVENT_MOUSE_LCLICK;      // We want to know when
it's clicked
settingsTaskData.strLabel = L"Settings";
settingsTaskData.strDescription = "Configure the Ad Blocker component";
settingsTaskData.strAlias = L"{MyCompanyAdBlockerSettingsTask}";  // Our custom
alias. Optional. Aliases are global so need to be unique
settingsTaskData.lParam = 0;           // Any context data you wish to pass to the
notification interface when clicked
settingsTaskData.pInterface = &notificationInterface;
NavInsertItem(taskGroup, L"SettingsTask", &settingsTaskData);
NavHandle settingsTask = settingsTaskData.hItem;
```

> The NavItemHelpers code in sdk\helpers has more convenient ways to manage tasks

Removing tasks

To remove a task simply call NavDeleteItem with the FQN or alias of the task you wish to delete.

Changing tasks

If you wish to change a task item's properties (label, description etc) you can do so with NavSetItem or NavSetItemEx

```
NAVSETITEM newItemProperties;
newItemProperties.mask = NI_MASK_LABEL | NI_MASK_DESCRIPTION;
newItemProperties.strLabel = L"New Label";
newItemProperties.strDescription = L"New Description";
NavSetItem(aliasOrFQNOfTaskToChange, &newItemProperties);
```

Responding to task activation

In order to respond to user task selection you need to create the task with the NI_EVENT_MOUSE_LCLICK mask bit set in the EventMask field and you need to provide a NavigationItemUIInterface pointer that has a NavItemNotify function. When a user clicks a task then the framework calls your NavItemNotify function with the message value NI_EVENT_MOUSE_LCLICK.

If you have used the same NavigationItemUIInterface for all of your task items, you will need to distinguish them from each other. You can do this by the context data that was supplied when the item was inserted, the Navigation Path that the item was added with, or its alias. For example:

```
LPCWSTR settingsTaskFQN = L"TASKLIST\\AdBlockerTaskGroup\\SettingsTask";
LPCWSTR deleteTaskAlias = L"{MyComponentDeleteTask}";
static const unsigned int addItemTaskId = 1;

// The registered handler for the notify messages for your navigation items. This
example assumes that only task items use this function
UINT NavItemNotify(NAVITEMNOTIFY* notifyMessage)
{
 if ( notifyMessage->nMsg != NI_EVENT_MOUSE_LCLICK)     // Not a click event so no
need to continue
 {
  return 0;
 }

 if ( (unsigned int)notifyMessage->pContext == addItemTaskId )  // addItemTaskId
was set as the lParam field when registering that task
 {
  DoAddItem();
  return 0;
 }

 if ( _wcsicmp(notifyMessage->strAlias, deleteTaskAlias) )   // We set strAlias
when we registered the task
 {
  DoDelete();
  return 0;
 }

 if ( _wcsicmp(notifyMessage->strFQN, settingsTaskFQN) )    // Compare the raw FQN
 {
  DoSettings();
  return 0;
 }

 return 0;
}
```