# Add Recipient, Reassign Recipient, or Recall Message

Last Modified on 07/13/2017 3:00 am EDT

```
using System.Collections.Generic;
using System.Text;
using PNMsoft.Sequence;
using PNMsoft.Sequence.Expressions;
using PNMsoft.Sequence.Forms.Activities;
using PNMsoft.Sequence.Messaging;
```

```
namespace SequenceEx.Tasks.Samples
{
 public class TaskActivityExecutorExample
 {
 public static void SendTask(ActivityInstance activityInstance, string subject, string body, IEnu
merable recipients)
 {
 TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
 Task task = executor.CreateTask();

 if (!string.IsNullOrEmpty(subject))
 {
 task.Subject = SqExpression.Object(EscapeExpressionString(subject));
 }
 if (!string.IsNullOrEmpty(body))
 {
 task.Body = SqExpression.Object(EscapeExpressionString(body));
 }


 task.Recipients.Clear();

 RecipientSet set = new RecipientSet();
 foreach (Recipient recipient in recipients)
 {
 set.Recipients.Add(recipient);
 }

 task.Recipients.Add(set);


 executor.SendTask(task);
 }

 public static void SendTask(ActivityInstance activityInstance, IEnumerable recipients)
 {
 TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
 executor.SendTask(recipients);
 }


 public static void RecallTask(ActivityInstance activityInstance, IEnumerable recipients)
 {
 TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
 executor.RecallTask(recipients);
 }


 public static void ReassignTask(ActivityInstance activityInstance, IEnumerable fromRecipients, I
Enumerable toRecipients)
```

```
IEnumerable toRecipients)
{
TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
executor.ReassignTask(fromRecipients, toRecipients);
}

public static void ReassignTask(ActivityInstance activityInstance, IEnumerable fromRecipients, s
tring subject, string body, IEnumerable toRecipients)
{
TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
Task task = executor.CreateTask();

if (!string.IsNullOrEmpty(subject))
{
task.Subject = SqExpression.Object(EscapeExpressionString(subject));
}
if (!string.IsNullOrEmpty(body))
{
task.Body = SqExpression.Object(EscapeExpressionString(body));
}


task.Recipients.Clear();

RecipientSet set = new RecipientSet();
foreach (Recipient recipient in toRecipients)
{
set.Recipients.Add(recipient);
}

task.Recipients.Add(set);

executor.ReassignTask(fromRecipients, task);
}

public static Task CreateTask(ActivityInstance activityInstance)
{
TaskActivityExecutor executor = new TaskActivityExecutor(activityInstance);
return executor.CreateTask();
}

private static string EscapeExpressionString(string text)
{
if (string.IsNullOrEmpty(text))
{
return string.Empty;
}

StringBuilder sb = new StringBuilder();

sb.Append('"');

for (int i = 0; i < text.Length; i++)
{
char ch = text[i];

if (ch == '"')
{
sb.Append('"');
}

sb.Append(ch);
}

sb.Append('"');
```

```
  sb.Append(' ');

  return sb.ToString();
  }
  }
}
```