

tools4ever



---

User Management Resource Administrator

# UMRA Example Projects

Service Management

*Copyright © 2005, Tools4Ever B.V. All rights reserved. No part of the contents of this user guide may be reproduced or transmitted in any form or by any means without the written permission of Tools4Ever.*

*DISCLAIMER - Tools4ever will not be held responsible for the outcome or consequences resulting from your actions or usage of the informational material contained in this user guide. Responsibility for the use of any and all information contained in this user guide is strictly and solely the responsibility of that of the user.*

*All trademarks used are properties of their respective owners.*

# Table of contents

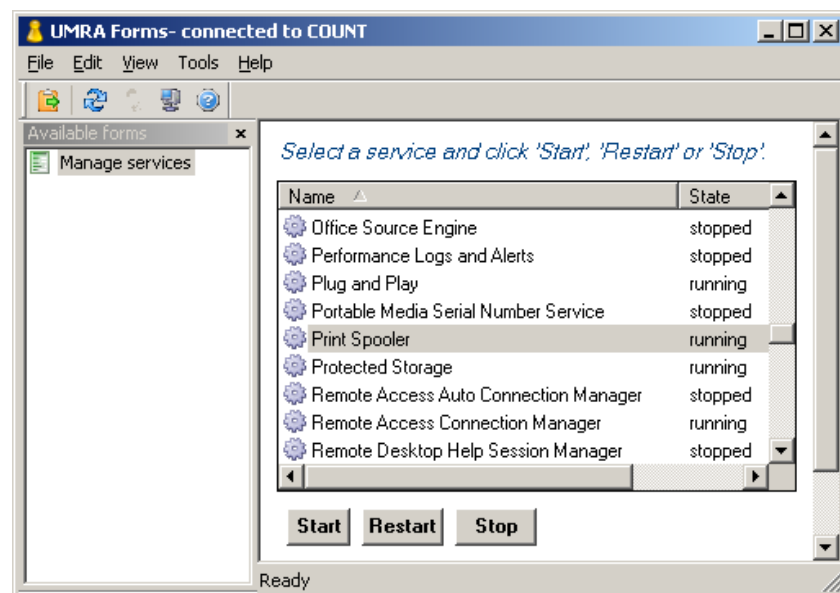
<b>Table of contents</b>	<b>i</b>
Introduction .....	2
Project structure .....	3
Step 1: Environment setup.....	4
Step 2: Form project - Collect Services .....	5
Step 3: Form project - Manage Services .....	9
Manage Services – Form, part 1 .....	9
Manage Services: Form table .....	10
Manage services: Form buttons .....	13
Manage Services: Script .....	16
Manage Services – Link to project Collect Services .....	19
Project execution .....	20
Contacts .....	22

# Introduction

## Goal

Although primarily focusing on user accounts and associated resources, you can also manage services with UMRA. From all computers, including domain controllers and regular workstations, the services can be managed.

In this example project, an implementation is described to (re)start and stop services that can be selected from a list.



In the form shown, the list presents an overview of all or a number of specific services from a specific computer. By selecting a service and clicking one of the buttons, the services can be managed.

The example project can easily be extended to:

1. Support multiple computers. For each computer, a form can be shown or the services of multiple computers are shown in a single form.
2. Only specific services are shown. This is especially useful in a helpdesk environment to allow employees to restart only specific services.
3. The number of commands can be limited. In this case, a user can only restart a not stop a service.

Note that the example project by default supports delegation and logging: Only specific users are granted access to run the form and all service management actions are logged.

---

# Project structure

## Form projects

The example scenario consists of 2 form projects:

1. **Manage Services** project: The main form project that holds the form and the script to manage the selected service. At initialization time, the **Manage Services** project accesses the other project.
2. **Collect Services** project: A very simple project that is used to collect the services information from a specific computer. The **Collect Services** project only contains a script and not a form. The script is used to collect the service information and store the services table in a variable.

Both form projects are available from the Tools4ever web-site. The projects are designed in such a way that only minimal changes are required to make the projects work in your environment.

## Principle of operation

As an initialization project, the script of the **Collect Services** project is executed when the form of the **Manage Services** project is executed. The script of the **Collect Services** projects collects the services information and stores all data in a single variable. The variable is shown as a table in the **Manage Services** project. When a service is selected from the list and one of the buttons is clicked, the script of the **Manage Services** project is executed. Next, the complete cycle starts over again, and again...

---

---

# Step 1: Environment setup

## Prerequisites

To run the project successfully you need to meet the following requirements:

1. You need to be logged on to the network with administrative privileges. During the implementation of the project, the UMRA Service is installed. The service can be installed on any computer but needs to have access to the computer with the services you need to manage.
2. The computer on which UMRA is installed must run one of the following operating systems: Windows XP, Windows 2000 (all versions) or Windows 2003 (all versions).

## UMRA installation

To run the project, you need to install several UMRA modules. Once installed, you can run the product for 30 demo days. After this period, a valid license code is required. To start, download the UMRA software from [www.tools4ever.com](http://www.tools4ever.com) and install at least the modules **UMRA Console** and **UMRA Forms**.

## UMRA setup

Once installed, start the **UMRA Console** application: Select menu option **All programs, User Management Resource Administrator, UMRA Console**. Upon startup, the **User Management Resource Administrator Wizard** is started automatically. You can either run the wizard to become familiar with the product or move forward and start with the installation of the UMRA Service. To do so, **Cancel** out of the wizard and select **UMRA Service, Install or upgrade service**. Follow the instructions of the application.

It is advised to install the UMRA Service on a computer that is a member of the domain containing the user accounts and computer services you wish to manage. For test purposes, you can also install the **UMRA Service** on the same computer that runs the **UMRA Console** application.

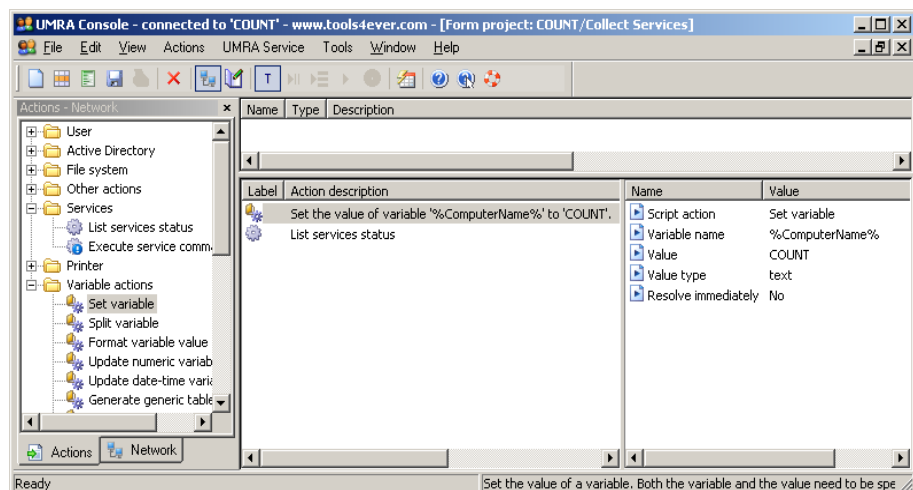
---

## Step 2: Form project - Collect Services

In this step, the help project **Collect Services** is created. As the name describes, the **Collect Services** project is used only to collect the information of a specific computer. The project stores the services information in a table variable that is used by the main project.

### Start UMRA Console

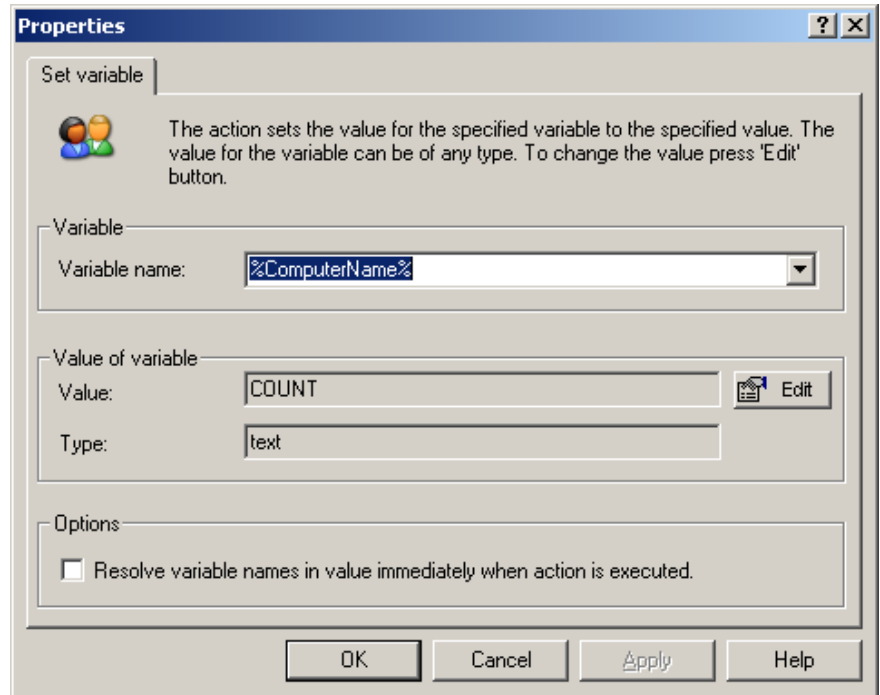
1. Start the **UMRA Console** application and connect to the **UMRA Service**: Select **UMRA Service, Connect...** and connect to the computer on which **the UMRA Service** is installed.
2. To start creating the new form project, select **File, New....** Check button **Form project** and press **OK**. Enter the name of the project, **Collect Services** and press **OK**.
3. The form project window is initially empty. Only the **Window Help** sections are shown. To hide the **Window Help** sections, right click in one of the three window areas and deselect menu option **Show Window Help**.
4. This project is meant only to collect services information using a small script. Therefore, the project contains no form and the upper half of the window is not used to design a form.
5. Next, we will setup the action that creates and initializes the variable that holds the name of the computer from which we want to manage the services. In the **Actions – Network** bar, activate the **Actions** window and expand the tree **Variable actions, Variable Operations**. Add the action **Set variable** to the script of project: drag- and drop the action to the lower left area of the window.



6. Select the new action **Set the value of variable ...** in the lower left window of the project window (not in the **Actions** bar). In the lower right window, the properties of this action are now shown. Double click one of the properties (example: **Variable name** or select main menu action

Setup **Set variable** action

7. Now setup the **Set** variable action as shown below:

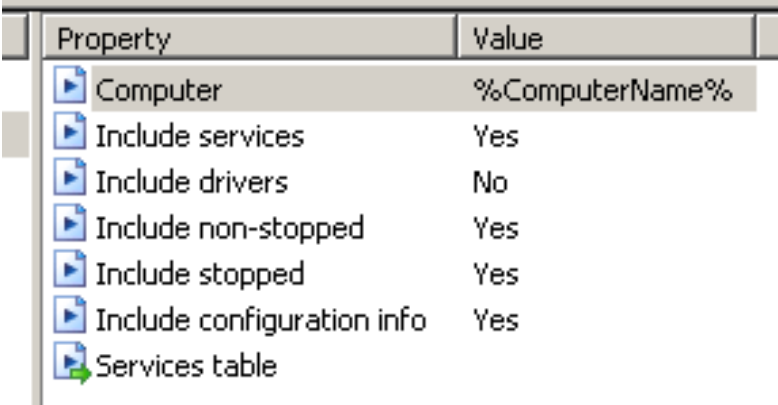


The action will create a variable with the name **%ComputerName%**. The variable will hold the name of the computer from which the services are managed. Press **Edit** to specify the name of the computer. In the example shown, computer COUNT is specified. Note: the variable **%ComputerName%** is used both in this project, and in the next project of the wizard: **Manage Services**. The variable is initialized only once.

Setup **List services status** action

8. Setup the action to collect the service status information. From the **actions** bar, select action **Services, List services status** and drag and drop the action to the script section (lower left) area of the project window. The script action is automatically selected and the properties section (lower right of project window) shows all the properties of this action. You need to setup the properties of this action one-by-one. By double-clicking each of the properties, you can specify the value of the selected property.

9. The most important properties of the **List services status** action are **Computer** and **Services table**. For **Computer**, specify the name of the variable created with previous action **Set variable: %ComputerName%**. The property **Services table** is by default configured as output variable **%ServicesTable%**. The green arrow shown with property **Service table** indicates that the value of the property is stored in an **output** variable. By double-clicking the property, you can setup the property.

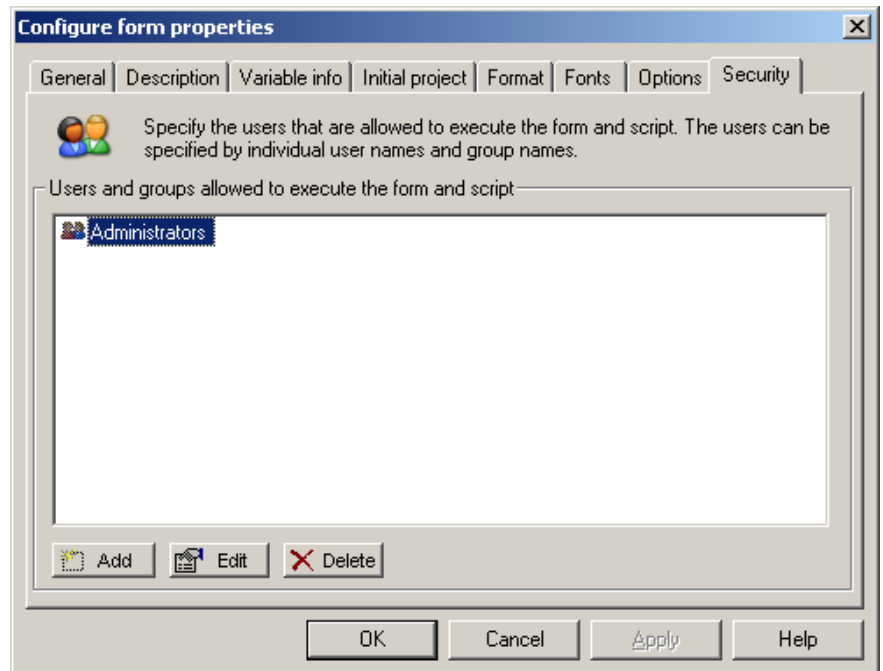


Property	Value
Computer	%ComputerName%
Include services	Yes
Include drivers	No
Include non-stopped	Yes
Include stopped	Yes
Include configuration info	Yes
Services table	

What happens when this action is executed: The UMRA software connects to the computer specified by **%ComputerName%** and collects the status of all of the services. The status information includes, the name of the service, the operational state of each service (running, stopped), type of service (automatic, manual, disabled) and so on. This information is stored as a table in variable **%ServicesTable%**. Note that the single variable will hold a table with multiple rows and columns. The variable is used in the other project.

## Setup project security

10. The project **Collect Services** is now almost complete: You still need to setup the security settings of the project: *Who is allowed to execute this form project*. Select menu option **Actions, Form properties...** and click on tab **Security**.



Press the **Add** button and enter the name of the user or group to setup the form project access rights. Press **OK** to finish this step.

11. Finally, save the project and close the project window.

## Summary

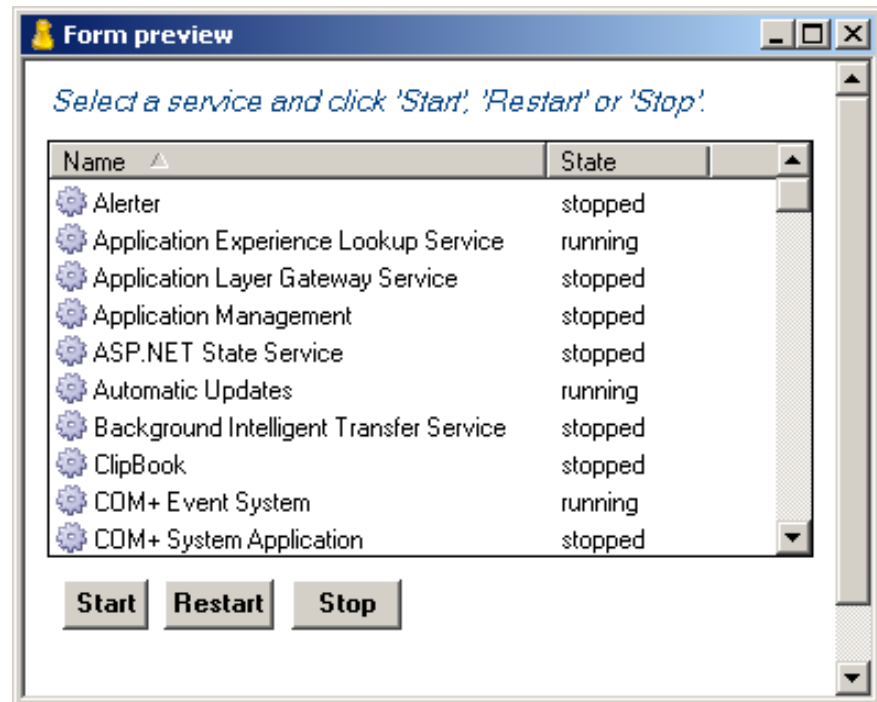
The project **Collect Services** is now ready for use and has the following characteristics:

- The project has no form, only a script
- The script sets a variable **%ComputerName%** to a specific value and collects the services information from the specified computer. The results are stored as a table in variable **%ServicesTable%**.
- The form project access rights are setup

---

## Step 3: Form project - Manage Services

The **Manage Services** project is the main project of this example scenario. The project contains both a form and script. The form shows the services of the computer and buttons to manage the selected service:

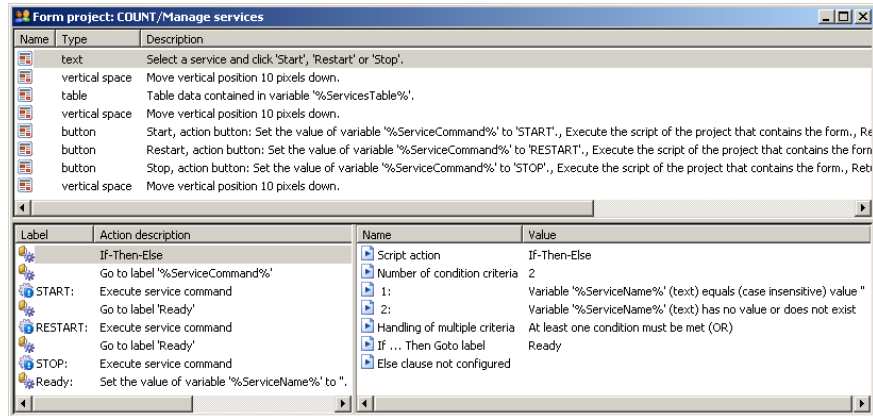


The script of the project executes the **Start**, **Restart** or **Stop** action. When the script action is completed, the list with service status information is refreshed to reflect the new service status.

### Manage Services – Form, part 1

Start a new form project with the name **Manage Services**. See the previous section for more information on how to do this. To setup the form project, the form and the script must be designed. To setup the form, a number of form fields must be added to the form. For each form field, parameters must be specified.

---



To add a form field, right click in the upper form area of the project window and select menu option **Add form field...**

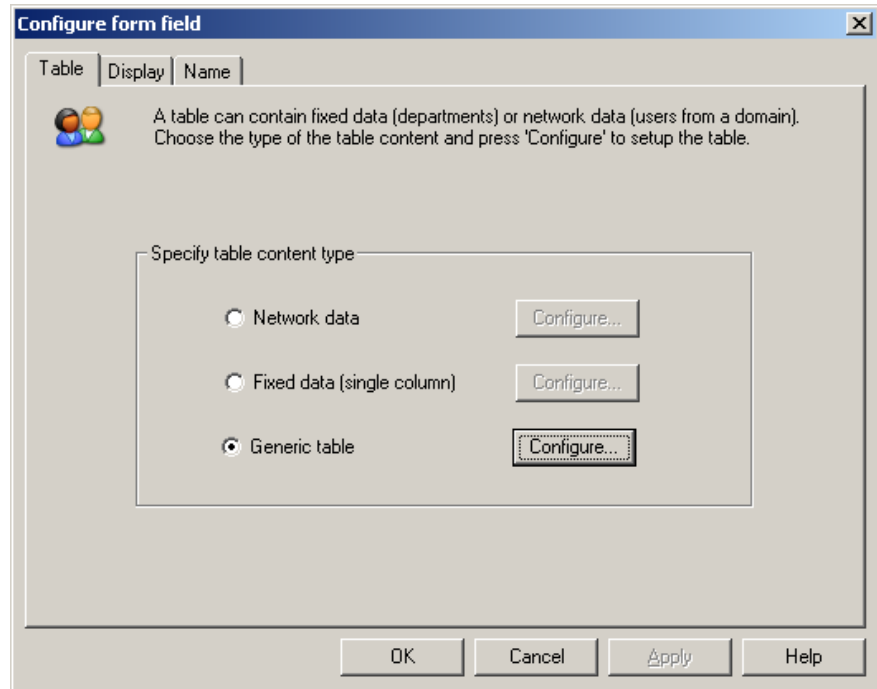
### Adding form fields

Add the following form fields:

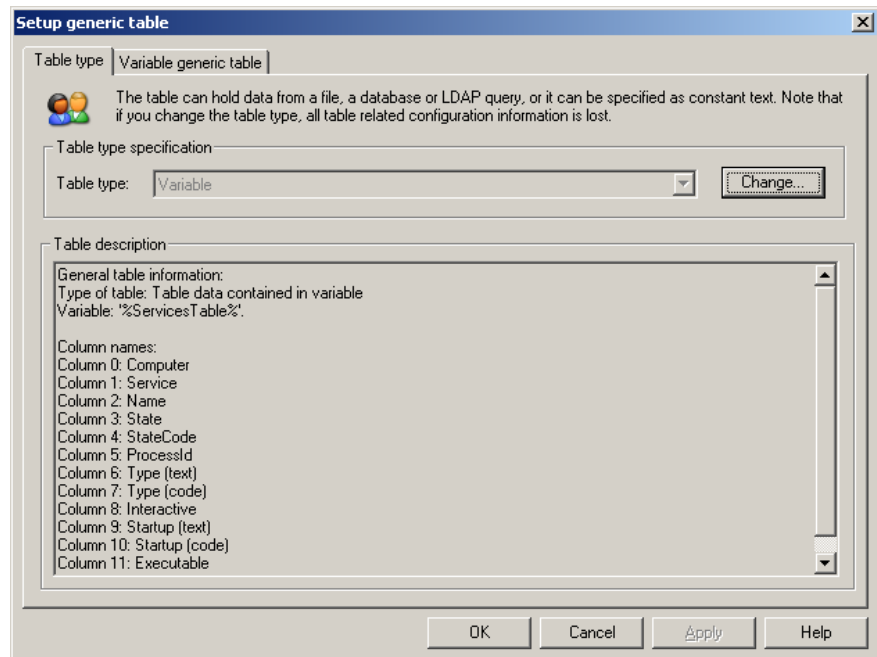
1. Static text field: The field shows the introduction text: *Select a service and click 'Start', 'Restart', or 'Stop'.*
2. Vertical space: Some open space (10 pixels) to outline the form.
3. Table: This is the table form field that lists the services. The table form field configuration is described in the next chapter.

## Manage Services: Form table

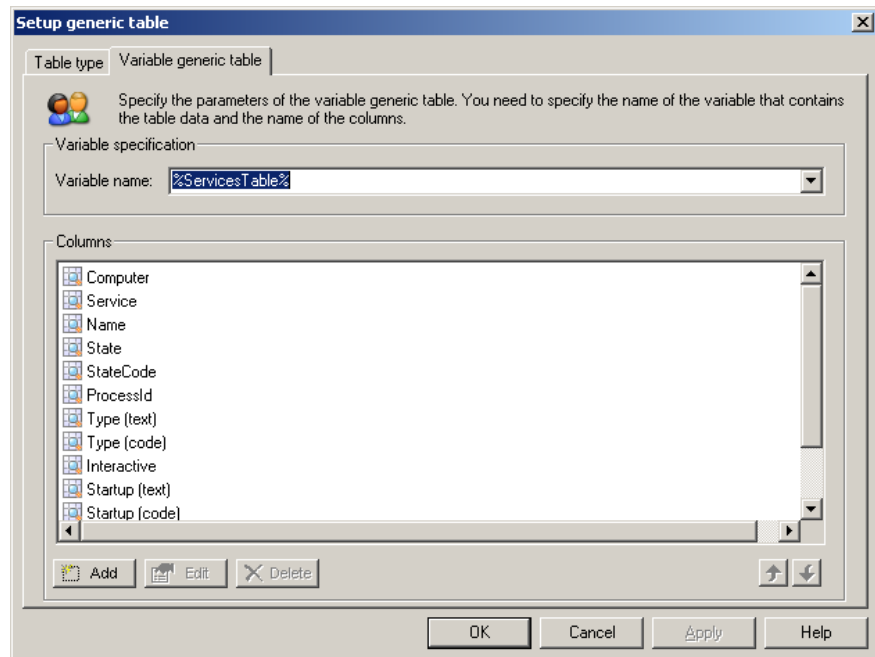
The form table lists the services and services status information. The configuration of the table is described in this chapter. The services information is obtained in project **Collect Services**. In that project, the services information is stored in a table variable. The variable is passed to the **Manage Services** project. The **generic table** type is able to show the table data of a variable. So select **Generic table** as shown in the following figure:



Press **Configure** to continue. Next, you need to select the type (source) of the generic table. Select **Variable** since the table data is obtained from a variable.



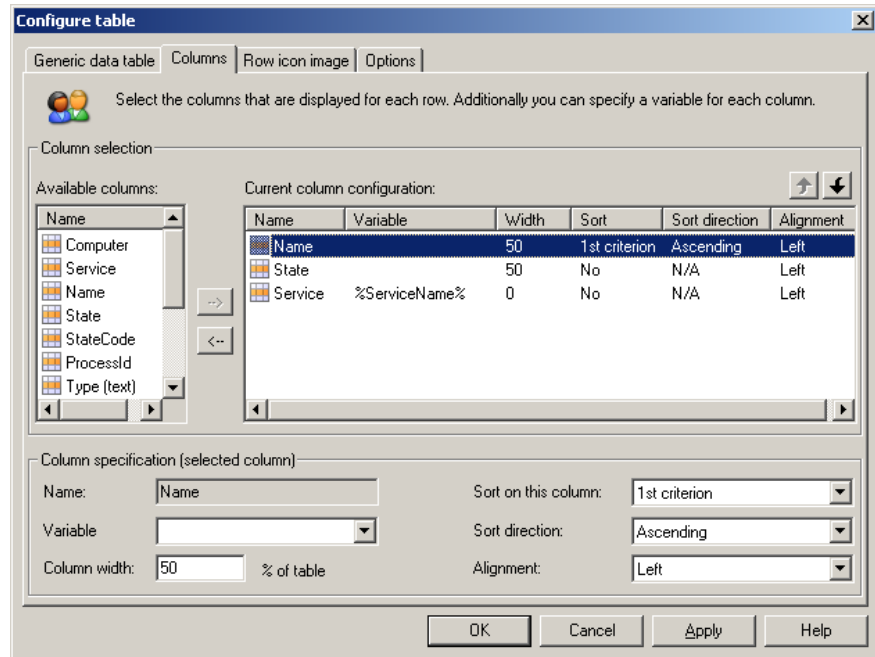
When selected, the configuration window **Variable generic table** can be selected. The window is used to specify the name of the variable and to define the columns of the table.



Specify the name of the variable: **%ServicesTable%**. The name must equal the name of the variable generated by the action **List services status** of project **Collect Services**.

Important: A table variable only holds the data of the table, not the names of columns. You therefore need to add the names of the columns that can be shown with the generic table. As described in the online help, the **List services status** generates a table with the following columns: Computer, Service, Name, State, StateCode, ProcessId, Type (text), Type (code), Interactive, Startup (text), Startup (code), Executable, Log on as. Add these columns one-by-one. The first part of the generic table configuration is now complete. Press **OK**.

You now need to setup the table columns that must be shown in the form. Click on the **Columns** tab.



This window is used to configure the columns that must shown in the form and to specify the variables that are passed to the UMRA Service when the end-user selects a service and presses a submit button.

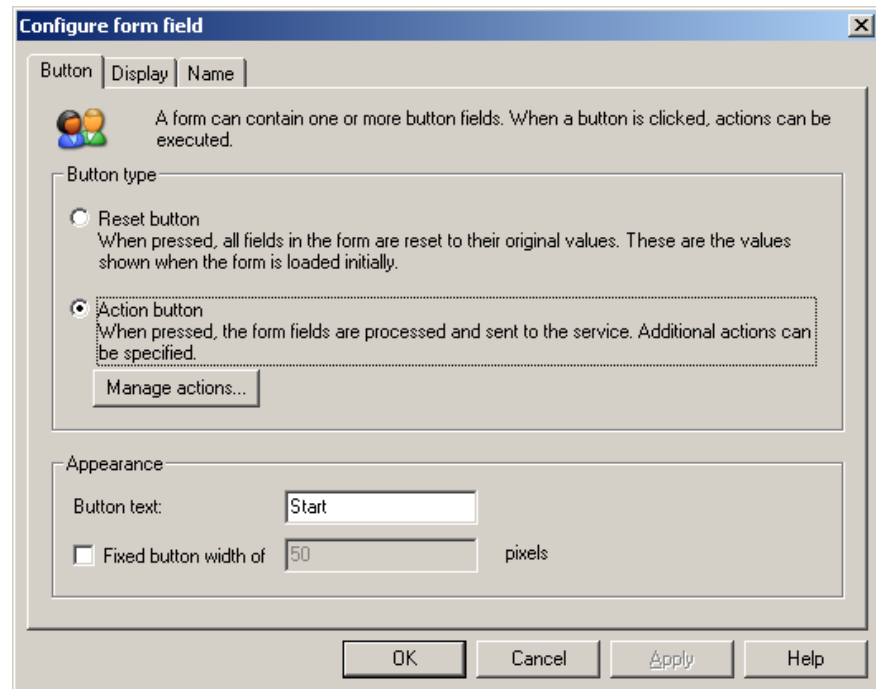
On the left side, the **Available columns** are shown. These columns correspond with the columns configured in the previous step. By using the add (->) and remove (<-) buttons you can setup a column configuration. In the example, the form will show a table with 3 columns. The 3<sup>rd</sup> column is not visible since it has a width of 0%. This column is included since it uniquely specifies the name of the service. When the user selects a service and presses a button, the value of this column is stored in variable **%ServiceName%**. This variable is passed to the UMRA Service and used for further processing.

Use the **Row icon image** and **Options** windows to specify additional table configuration settings. The configuration of the table is now complete.

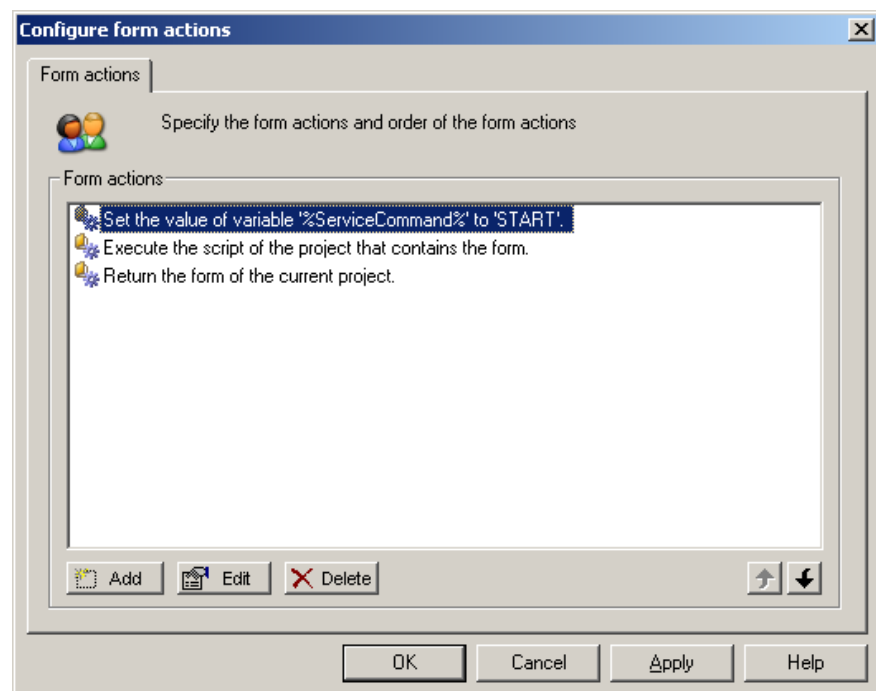
To specify additional table display settings such as font and alignment, select the **Display** window of the **Configure form field** window.

## Manage services: Form buttons

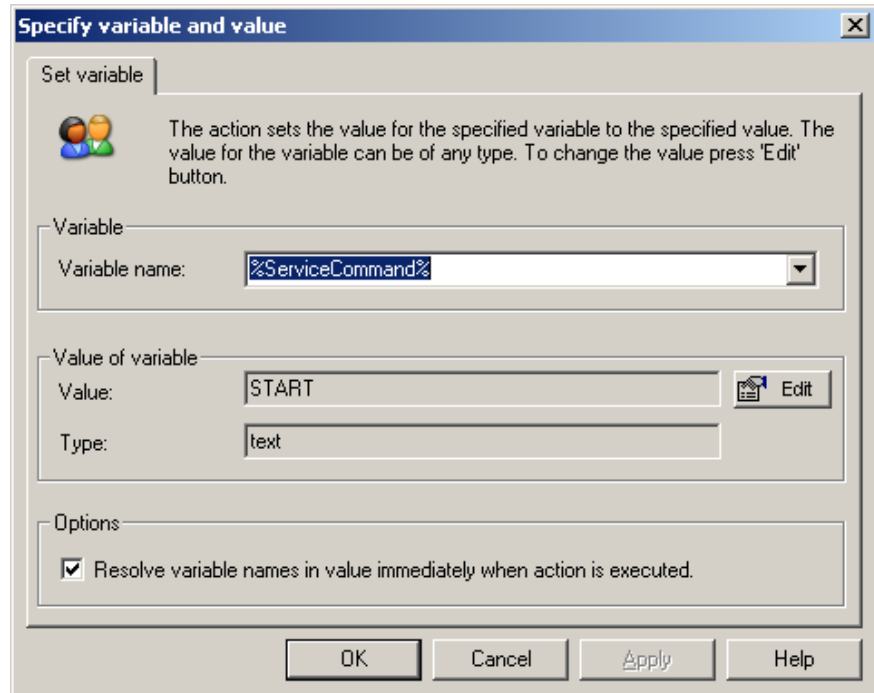
In the example project, after the table form field, some vertical space is added. Next, the 3 submit buttons are configured. To setup the **Start** button, specify the **button type** as **Action button** and enter the text **Start** as **button text**.



Press the **Manage** actions button to configure the actions that must be executed when the button is pressed in the form.

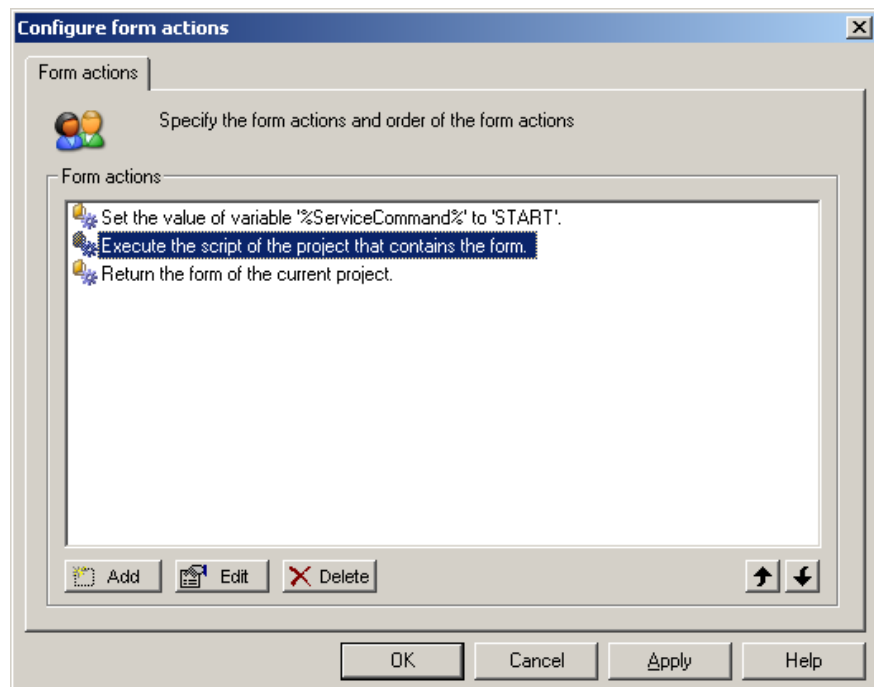


The script of the **Manage Services** project uses the variable **%ServiceCommand%** to jump to the appropriate label of the script. This variable is set as the first action of the script. For the start button, the variable is set to **START**.



So when the user presses the **Start** button, the value of the variable **%ServiceCommand%** is set to the text **START**. Similarly, the value of the variable **%ServiceCommand%** is set to **RESTART** and **STOP** for the other buttons.

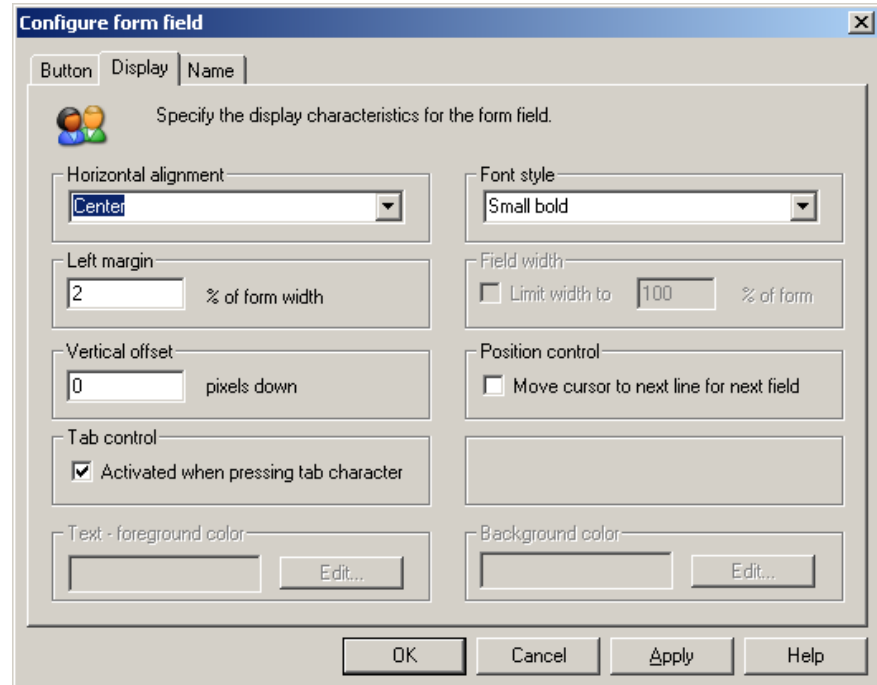
The next action instructs the UMRA Service to execute the script of the **Manage Services** project. This action has no parameters.



Finally, the last action executed when the button is pressed, returns the form of the project. This action needs no additional configuration settings.

The **Restart** and **Stop** buttons are identical to the **Start** button, except for the variable **%ServiceCommand%** as described earlier.

To position the buttons next to each other, configure the display settings for the **Start** and **Restart** buttons as follows:



Note the **left margin** of 2% to separate the buttons. The **Position control** setting **Move cursor to next line for next field** must be unchecked to position the buttons next to each other. For the last button, **Stop**, this option must be checked.

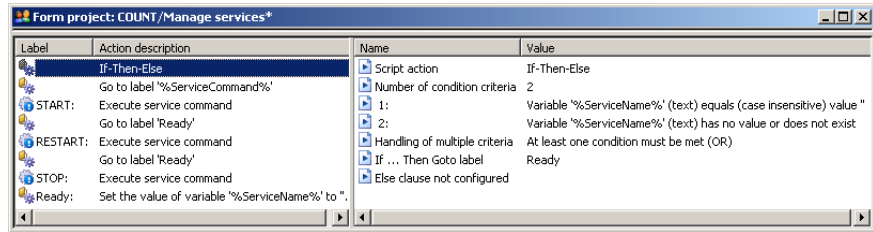
## Manage Services: Script

When one of the submit buttons, **Start**, **Restart** or **Stop** is pressed, the script of the **Manage Services** project is executed as part of the submit button script action execution sequence.

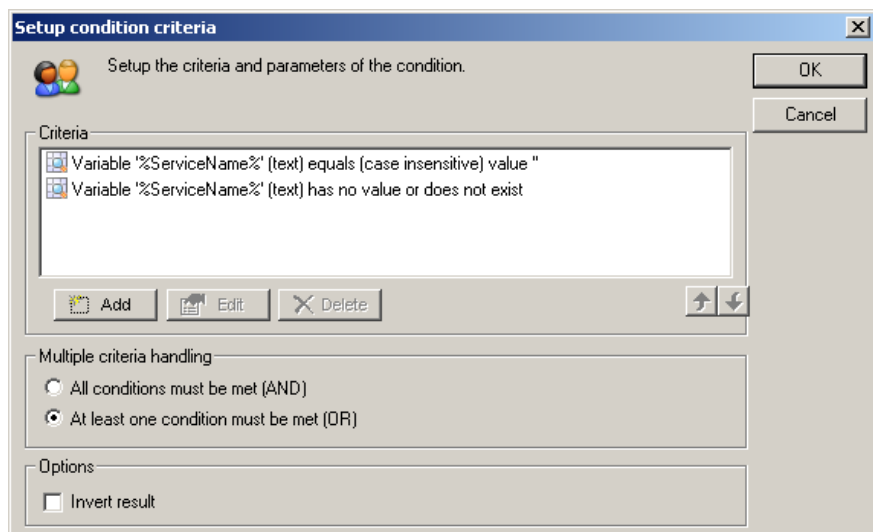
The script uses the following variables:

Variable	Description
%ServiceName%	The name of the service that is selected by the end-user in the form. When the user selects no service, the variable is empty. Otherwise, the value of the variable corresponds with the 3 <sup>rd</sup> column with zero width of the table shown in the form.
%ServiceCommand%	The value of the variable corresponds with the button pressed by the end-user. The value is set as an action when the button is pressed. For each button, the value is different. This button is executed, before the script of the project is executed.
%ComputerName%	The name of the computer that maintains the service. The variable is passed from the other project <b>Collect Services</b> . In a more realistic environment, the variable could be generated or selected in a window of a sequence of wizard windows.

The script first performs some input checking control and then jumps to the correct label and executes the requested action.

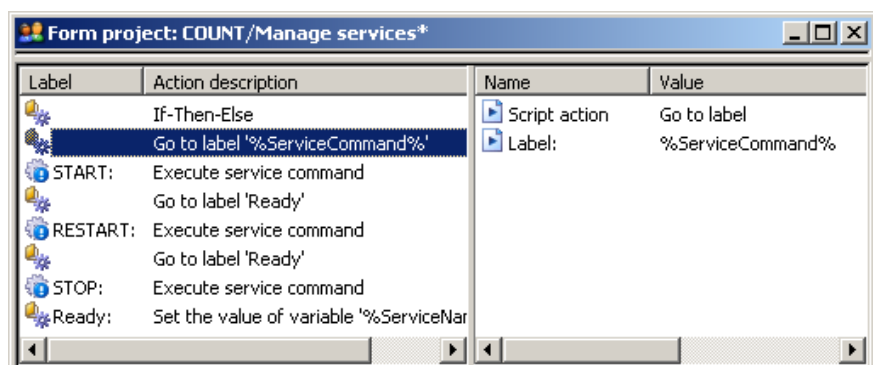


The action checks if the script input variable **%ServiceName%** is empty or does not exist. If this is the case, the end-user did not select a service from the list with services in the form, when one of the buttons was pressed.



When the variable **%ServiceName%** does not hold a valid value, the script execution proceeds with the action with label **Ready**.

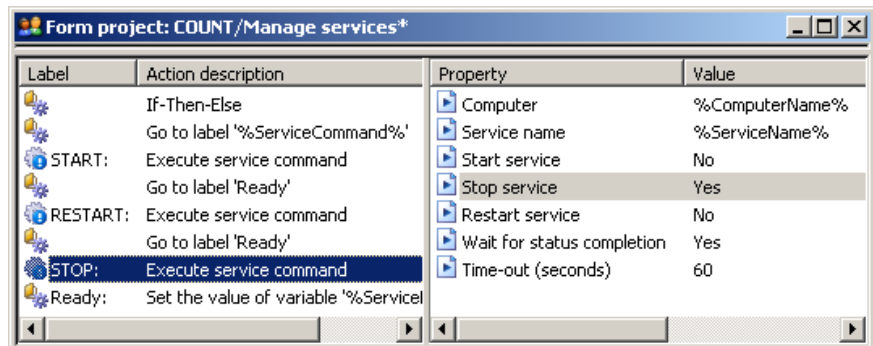
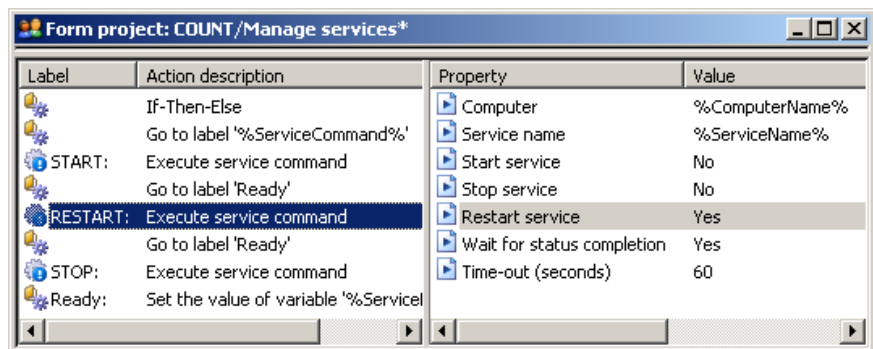
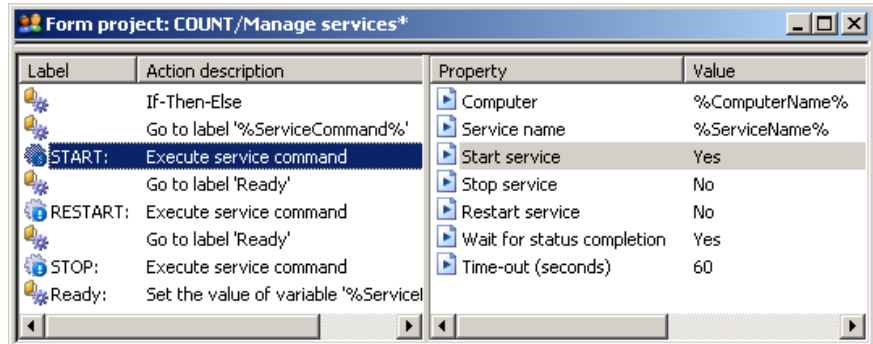
Next, the script jumps to the section that corresponds with the button pressed.



Note that the variable **%ServiceCommand%** is set as an action when one of the submit buttons is pressed. The value of the variable corresponds with the button: **START**, **RESTART** or **STOP**. In the script, exactly these values are used as labels. The **Go to label %ServiceCommand%** simply continues script action execution at one

of the labels.

The **Execute service command** actions are simple to configure. For each of the actions, the **Computer** property is set to the variable **%ComputerName%** that is passed from project **Collect Services**.



The **%ServiceName%** variable is the result from the form. The **Start service**, **Stop service** and **Restart service** properties are configured according to the desired service action.

The **Execute service command** action is followed by a **Go to label 'Ready'** action to continue script execution at the right location.

Finally, the value of variable **%ServiceName%** is cleared to reset the input for the next form execution cycle.

**Properties** [?] [X]

**Set variable**

The action sets the value for the specified variable to the specified value. The value for the variable can be of any type. To change the value press 'Edit' button.

**Variable**

Variable name: [%ServiceName%]

**Value of variable**

Value: [ ] [Edit]

Type: text

**Options**

Resolve variable names in value immediately when action is executed.

OK Cancel Apply Help

Note that when the script is executed, the next action of the form submit button action sequence is executed: **Return the form of the current project.**

## Manage Services – Link to project Collect Services

As described earlier, the form of the **Manage Services** project shows the services information collected in the script of project **Collect Services**. To achieve this behavior, the **Manage Services** project must be configured to execute the script of the project **Collect Services** *before the form of the Manage Services project is shown.*

Right click in the upper form section of the **Manage Services** project window and select **Form properties....** Select tab **Initial project** and specify project **Collect Services**. The **script** of the **Collect Services** project will now be executed, each time the **form** of the **Manage Services** project is shown. The variables that are generated in the **Collect Services** project can be used in the form fields of the **Manage Services** project.

To finish the project, select the **Security** tab to setup the access rights for the **Manage Services** project.

---

## Project execution

Now what happens when the user select the form **Manage Services** in the **UMRA Forms** application?

1. A request is send to the from the **UMRA Forms** application to the **UMRA Service** to generate and return the **Manage Services** form.
2. The **UMRA Service** checks the access rights of the end-user and the **Manage Services** project is loaded by the **UMRA Service** and the form generation is initialized.
3. As part of the **Manage Services** form generation process, the project **Collect Services** is loaded and the script is executed. Resulting variables (**%ServicesTable%**) are stored and passed to the form generation process.
4. The form of the **Manage Service** project is generated. The table holds the data from the variable generated by the **Collect Services** project.
5. The form is returned to the **UMRA Forms** application and shown.
6. The user selects a service and presses one of the buttons.
7. The selected service is stored in variable **%ServiceName%** and send with information of the pressed button to the **UMRA Service**.
8. The **UMRA Service** checks the access rights of the end-user and the actions configured for the button are executed.
9. The form of the **Manage Services** project is generated and returned to the **UMRA Forms** application. As part of the form generation process, the script of the **Collect Services** project is executed.

The following section shows the **UMRA Service** log file information for a complete session:

---

```
12:19:47 09/21/2005 Form message: '09/21/2005,12:19:47,"SSP\J.
Vriens","Forms list",OK,N/A,"1 projects found for user 'SSP\J.
Vriens'.'"
12:19:49 09/21/2005 Executing form initialization project 'Collect
Services'.
12:19:49 09/21/2005 Variable 1: %UmraFormSubmitAccount%=SSP\J.
Vriens
12:19:49 09/21/2005 Getting services information from computer:
'COUNT'. Options: include services, exclude drivers, include non-
stopped services and/or drivers, include stopped services and/or
drivers, include configuration info.
12:19:49 09/21/2005 Form message: '09/21/2005,12:19:49,"SSP\J.
Vriens","Form load",OK,"Manage services",'
12:19:54 09/21/2005 Variable 1: %ServiceName%=W3SVC
12:19:54 09/21/2005 Variable 2: %UmraFormSubmitAccount%=SSP\J.
Vriens
12:19:54 09/21/2005 Variable 3: %ComputerName%=COUNT
12:19:54 09/21/2005 Variable 4: %ServicesTable%=Table with 96 rows
12:19:54 09/21/2005 Variable 5: %NowDay%=21
12:19:54 09/21/2005 Variable 6: %NowMonth%=09
12:19:54 09/21/2005 Variable 7: %NowYear%=2005
12:19:54 09/21/2005 Variable 8: %NowHour%=12
12:19:54 09/21/2005 Variable 9: %NowMinute%=19
12:19:54 09/21/2005 Variable 10: %NowSecond%=54
12:19:54 09/21/2005 Variable 11: %ServiceCommand%=RESTART
12:19:54 09/21/2005 If-Then-Else condition [Variable
'%ServiceName%' (text) equals (case insensitive) value '' OR
Variable '%ServiceName%' (text) has no value or does not exist]
result is FALSE, continue script execution with next action.
12:19:54 09/21/2005 Jump to script action with label 'RESTART'.
12:19:54 09/21/2005 Executing command for service 'W3SVC' on
computer 'COUNT': Restart service.
12:19:54 09/21/2005 Waiting 60 seconds for status completion.
12:19:56 09/21/2005 Service successfully 'restarted (stopped)'.
12:19:57 09/21/2005 Service successfully 'restarted (started)'.
12:19:57 09/21/2005 Jump to script action with label 'Ready'.
12:19:57 09/21/2005 Executing form initialization project 'Collect
Services'.
12:19:57 09/21/2005 Variable 1: %ServiceName%=
12:19:57 09/21/2005 Variable 2: %UmraFormSubmitAccount%=SSP\J.
Vriens
12:19:57 09/21/2005 Variable 3: %ComputerName%=COUNT
12:19:57 09/21/2005 Variable 4: %ServicesTable%=Table with 96 rows
12:19:57 09/21/2005 Variable 5: %NowDay%=21
12:19:57 09/21/2005 Variable 6: %NowMonth%=09
12:19:57 09/21/2005 Variable 7: %NowYear%=2005
12:19:57 09/21/2005 Variable 8: %NowHour%=12
12:19:57 09/21/2005 Variable 9: %NowMinute%=19
12:19:57 09/21/2005 Variable 10: %NowSecond%=54
12:19:57 09/21/2005 Variable 11: %ServiceCommand%=RESTART
12:19:57 09/21/2005 Getting services information from computer:
'COUNT'. Options: include services, exclude drivers, include non-
stopped services and/or drivers, include stopped services and/or
drivers, include configuration info.
12:19:57 09/21/2005 Form message: '09/21/2005,12:19:54,"SSP\J.
Vriens","Form submit",OK,"Manage services"'
```

First, at 12:19:47, the forms are loaded into the UMRA Forms application. Then, starting at 12:19:49 the form of project **Manage Services** is generated. This includes the execution of project **Collect Services**. At 12:19:54 the web-service is selected from the list and the **Restart** submit button is pressed. The script of the **Manage Services** project is executed. At 12:19:54 the service is requested to stop. 2 seconds later, at 12:19:56, the service is stopped and started at 12:19:57. Next, the script of project **Collect Services** is executed as part of the form generation process of project **Manage Service** project. At 12:19:57 the form is returned to the **UMRA Forms** client application and the process is complete.

---

# Contacts

If you have any comments regarding the content of this guide, please contact [EmielVanWezel@tools4ever.com](mailto:EmielVanWezel@tools4ever.com).

You can also visit the Tools4ever website for more information about our products:

***<http://www.tools4ever.com/>*** (<http://www.tools4ever.com/>)

***<http://forum.tools4ever.com/>*** (<http://forum.tools4ever.com/>)

---