



# Worksoft Certify Automation

---

## Advanced Topics

# Table of Contents

<b>Overview .....</b>	<b>4</b>
<b>Thought Design .....</b>	<b>5</b>
Process Design .....	5
Comments and Description .....	5
<b>Recordset Continuity .....</b>	<b>6</b>
Hardcoded Recordset Names .....	6
Recordset Variable Method .....	8
Recordset Naming.....	10
<b>Variables .....</b>	<b>12</b>
System Variable Usage .....	13
User Variables .....	13
Runtime Variable Value Changes .....	13
<b>Data Type Format .....</b>	<b>14</b>
<b>Data Type Format Tool .....</b>	<b>16</b>
<b>Import/Export a Recordset .....</b>	<b>18</b>
Exporting a Recordset .....	18
Importing a Recordset.....	19
<b>Recordset Mode .....</b>	<b>20</b>
<b>Creating and Using Attributes .....</b>	<b>21</b>
<b>Requirements .....</b>	<b>21</b>
<b>Execution Flow Rules .....</b>	<b>22</b>
<b>Result Management .....</b>	<b>23</b>
<b>Processing Naming/Storing Convention .....</b>	<b>24</b>
Transaction Name.....	24
Business Process Name .....	25
Layouts.....	25
Recordsets .....	25

Recordset Rows .....	26
Recordset Filter .....	27
<b>Table and Grid Insert .....</b>	<b>28</b>
<b>Selecting from a Tree View .....</b>	<b>30</b>
<b>Tooltips .....</b>	<b>31</b>
<b>Dynamic Objects.....</b>	<b>32</b>

# Overview

---

This guide is an addendum to the Certify Training Manual. It provides information on automation situations that are not covered in the Certify Training Manual. This guide includes the following advanced topics:

- Recordsets
- Variables
- Data Type Format
- Attributes
- Execution Flow Rules
- Result Management
- Process Naming/Storing Convention
- Table/Grid Insert
- Selecting from a Tree View
- Tooltips
- Dynamic Objects

# Thought Design

Processes are used to document and validate the end-to-end execution of your critical business processes. Knowing how to design processes is extremely important and requires some thought.

## Process Design



**Design test cases like your Mother would be running and maintaining it!**

Below are some pointers to remember when designing your processes:

- ▶ Design processes knowing that you will most likely not be the one to run and maintain them.
- ▶ Be as descriptive as possible in the process description field. Make a note of:
  - the transactions involved
  - the processes using recordsets
  - any special situations or needs
- ▶ Make use of comment steps anywhere that unique or complex processes may be done.



Incrementing counters, text compares that change process flow, Pass/Fail changes, etc.

- ▶ Create a Mega-Process (string/integrated test) that will execute the end-to-end scenario.
  - The name will be the same as the end-to-end scenario
- ▶ Create child process within each Mega-Process for every transaction of the end-to-end scenario.
  - The name will start with the transaction code, followed by an underscore (\_), and then the Process Purpose
  - Remove the spaces from the process name to join the words together



VA01\_StandardOrder  
VA01\_StandardOrder\_WithReference

## Comments and Description

Comments and descriptions are used to document or label processes and/or groups of steps within a process. It's important to make comments so others can understand the results they should see in the test.

- ▶ Add comments for all descriptive dynamic dialog steps
- ▶ Add comments before any steps that execute another process

Step#	Application Version	Window	Object	Component Action	Narrative	On True
1	System 1.0	System	Execution	Comment	"Test"	Continue

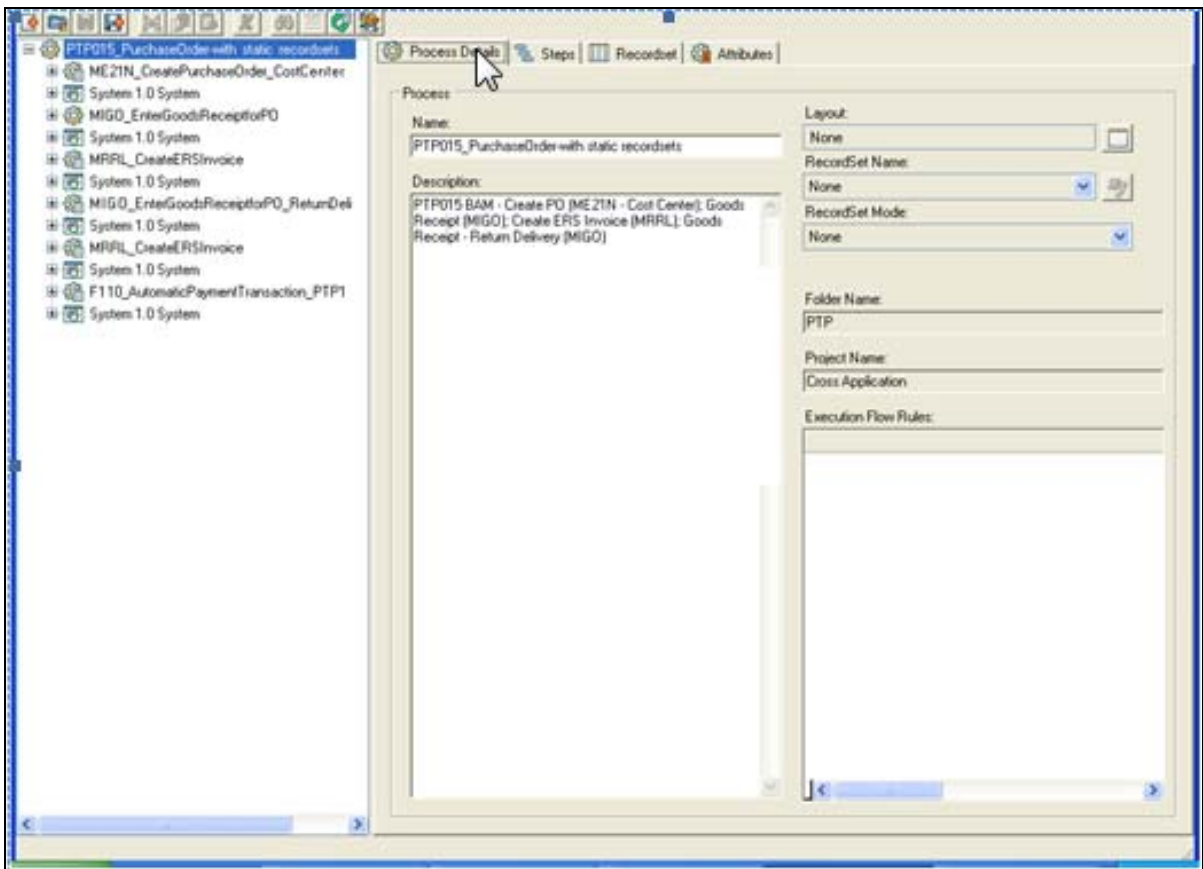
# Recordset Continuity

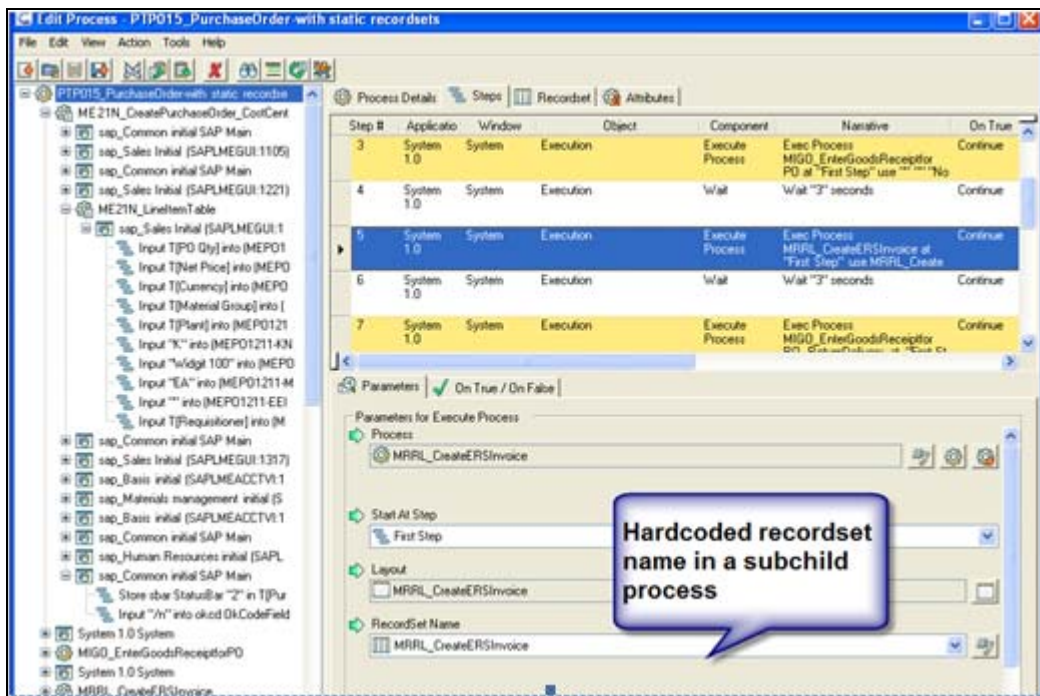
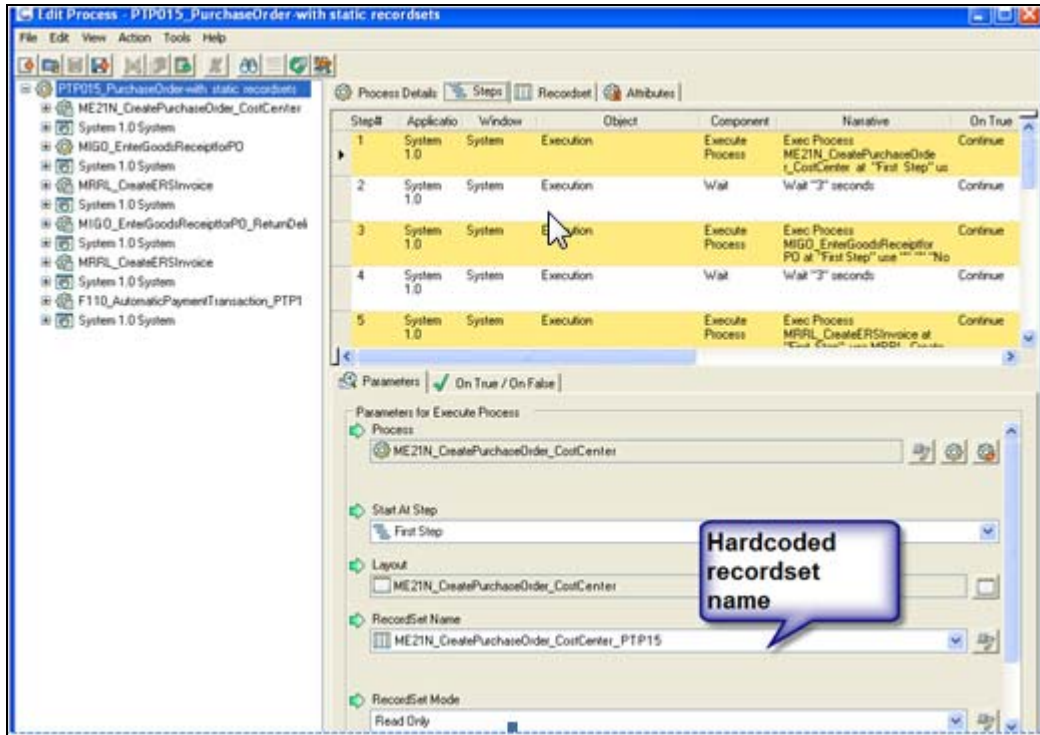
Recordset management is the driving force for Certify automation. It must be well thought out and maintained properly.

One simple way of maintaining continuity is to make use of the System recordset variables. It contains the value of the last recordset opened. If you use this variable as the recordset name for every child process, then it can be maintained and selected at runtime and propagated throughout the child processes – preventing the business owners the need to edit any processes.

## Hardcoded Recordset Names

Below is an example of a test case using hardcoded recordset names and the same test case using the System recordset variable.

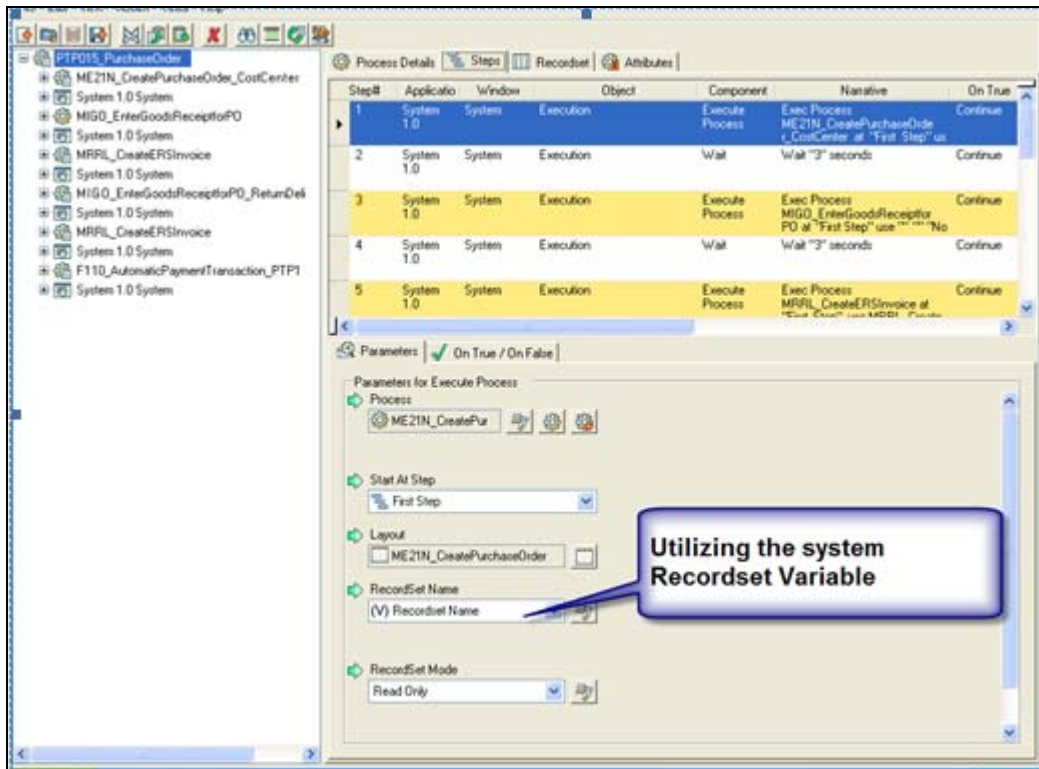
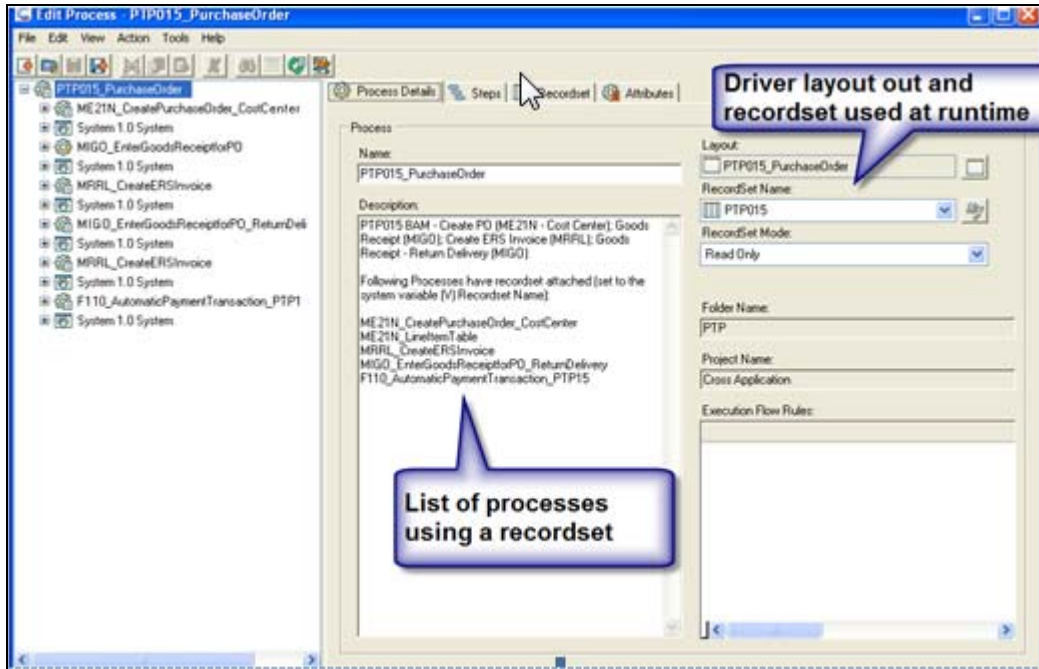


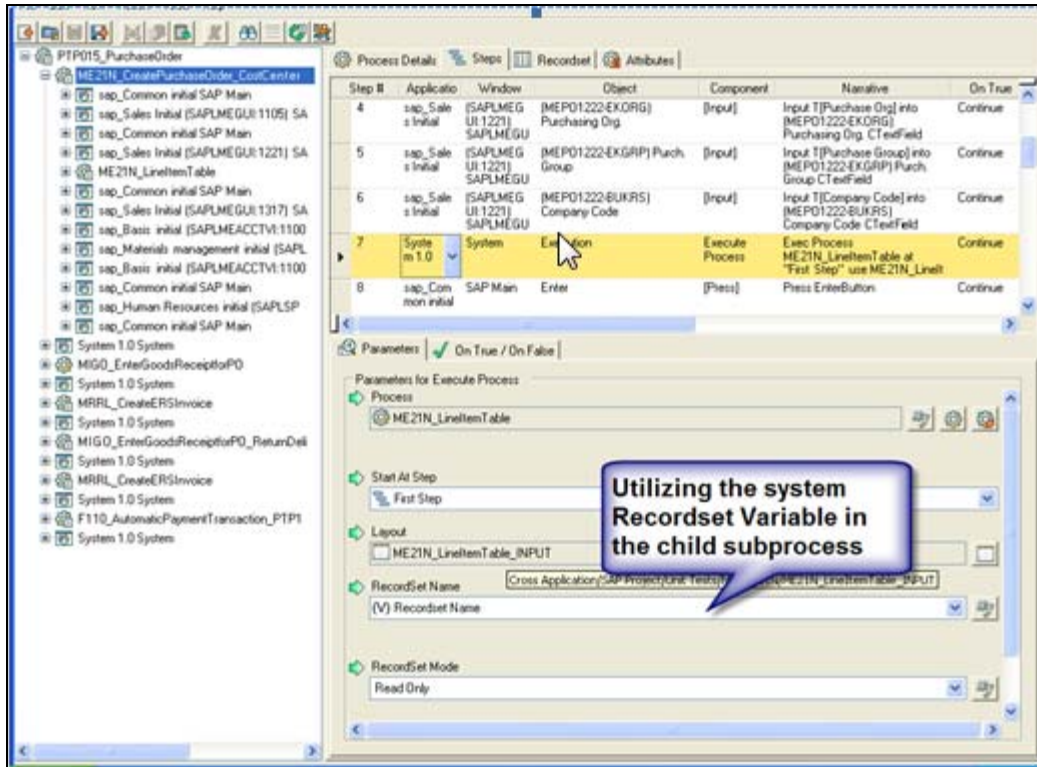


While the hardcoded test works fine, a user would have to edit the process and change the name of one or more recordset at the step levels if a different set of data was needed to run the process.

## Recordset Variable Method

The recordset named PTP015 would be the same named recordset at each child/sub-child process that utilizes a recordset and its name is passed via the System (Recordset Name Variable).





In the variable assigned recordset, the user only needs to maintain the actual recordset and select the correct one at runtime.

## Recordset Naming

The key to recordset naming being successful is naming all the associated records the same name. For example, if you have a Megatest named OTC-009, then all the recordsets should also be named OTC-009 or some variant of it (i.e., OTC-009US, OTC-009nonUS, etc.).



The top level process (the process to be executed) may or may not contain any actual recordset values. It is solely used to select the named recordset at runtime. If no real data is needed at this level, simply create a layout named the same as the process with a single dummy variable assigned to it (like TEMP or DUMMY). Then, create a list of empty recordsets to choose from at runtime (like OTC-009US, OTC-009nonUS, etc.).

**New Layout**

Details | Attributes

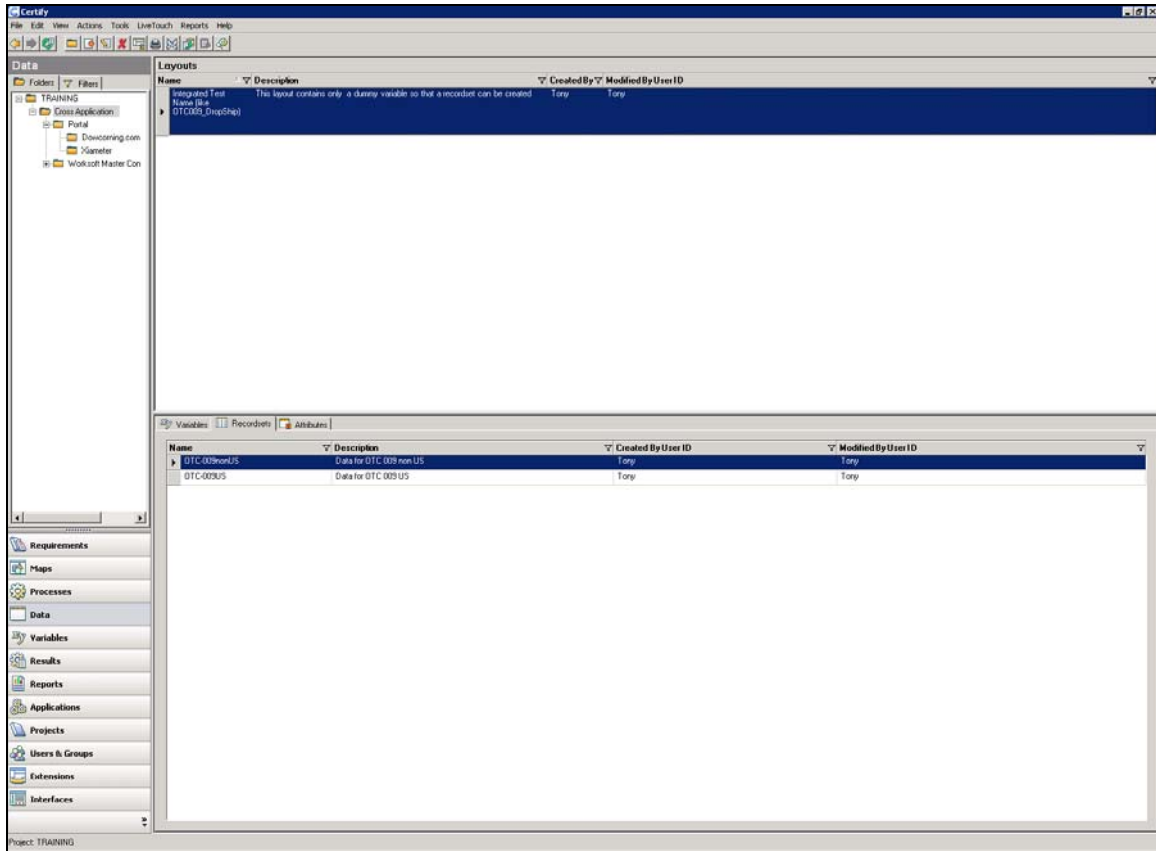
Name: \*  
Integrated Test Name (like OTC009\_DropShip)

Description:  
This layout contains only a dummy variable so that a recordset can be created

Create:  
[Empty dropdown] [Select]

Variables*			
Number	Name	Default Value	Descriptor
1	Dummy Variable for Recordset		

[OK] [Cancel]



# Variables

A variable is a placeholder or name that represents a value. In Certify, variables are an important component of the Business Process Certification because they provide a means of representing the data that you want to use in your processes. Variables contain a specific type of value that can be inserted into a field, acted on, or stored for future comparison against another value.

The most common use of variables is in data-driven testing, where process execution loops through a series of data values. Variables can also be used when you need to store or verify system data, such as the System Date or Machine Name, or when specific data for a user is required, such as a user ID or password to log into an external system or application.

When preparing to use variables, go through your existing processes and identify all of the places where variables can be used in place of static data. As a best practice, create a table or spreadsheet showing the processes and all of the variables that you will use for each process.

Name variables the same name as the field name on the screen. If an exact match is not available, make use of something similar before creating a new one. For example: if you need ORDER NO but ORDER NUMBER is currently in the list, best practice suggest you use ORDER NUMBER instead of creating a new variable, ORDER NO. This method will help you when you create the recordset to be used with your processes.

You will need to determine the type of value each variable will have. Each type of variable contains the following data types:

Data Type	Description
<b>Text</b>	Consists of alphanumeric and special characters and has a system maximum length of 65,535 (64K) characters
<b>Number</b>	Consists of integers, floating point values, and exponential notations
<b>Date</b>	Consists of any combination of month, day, and year in any Certify-supported format  Date variables default to today's date. You can also apply a specific date mask to a date variable to display the date in a certain format. If no mask is chosen, then the default system date format is used.



**NOTE**

If you do not want the value of a variable to display in the results file, select the Mask checkbox. Any values will be replaced with an asterisk (\*).

## System Variable Usage

System variables are pre-defined and used as **read-only** variables to data values during test execution. System variables contain system values that can be added to process steps, but cannot be part of a recordset.

Certify provides the following system variables:

- ▶ **Current date** – returns the data of the local workstation
- ▶ **Current process name** – returns the value of the current process name being executed
- ▶ **Current recordset name** – returns the value of the current recordset opened
- ▶ **Current recordset row number** – returns the row number from the current record being read
- ▶ **Last step status** – returns the status information from the last step executed
- ▶ **Current user name** – returns the user ID signed into Certify
- ▶ **Current layout name** – returns the current layout name opened
- ▶ **Current machine name** – returns the local workstation name

## User Variables

User variables are user-defined variables that are uniquely set by the user logged into Certify. They are created and managed from the Extension window by a Certify Administrator, and they are shared across all projects and all users. The values for user variables are specified by each user by selecting **Set User Variables** from the Tools menu. Example usage would be User Name and User Password.

Every Certify user will be able to set that variable to their own user account and password to access the application under test.



The password variable may have the **Mask** checkbox selected, resulting in the result viewer displaying an asterisk (\*) for the value.

## Runtime Variable Value Changes

During execution, **Watch** variables provide you a way to see a value of a variable each time the variable value changes. You can select variables to watch during execution, and then view the values placed in the variables during execution. You also have the ability to change the value at run time to assist in debugging a test process.



Change a default date, change a field value, etc. This is done by highlighting the current variable in the **Watch** tab and typing in the new value. **Note**; however, that the value may change if any action modifies that variable, such as record read, variable value set action, etc.

## Data Type Format

Data type formats are used in process steps to format/mask date or numerical variables based on the requirements of the field on the screen. These formats are interpreted by Certify during the execution of your tests. This allows the user the flexibility to set a value in the recordset (such as a date) and use that date with different formatting/masking applied at the step level. They are based on the type of variable that you are using in the step and not the step parameter.



(N) Amount may need to be inputted with a mask of S###,###.00 in one step and the same value may need to be inputted into another step with a mask of 000,###.

Certify supports the following format for data:

- ▶ **Number** – Certify accepts only valid numbers, and it will verify these numbers during process step creation. The available number formats are listed in the following table:

Format	Format Pattern
Currency	C
Decimal Integer	D
Scientific	E
Fixed Point	F
Hexadecimal	X
Zero Pad (10)	0000000000 #####
Fraction (5-digit precision)	# #####
Standard	

- **Date** – Certify provides a Date Picker for selecting a date, and no literal text values for the date are allowed. The available date formats are listed in the following table:

Format	Format Pattern
m/d/yy	M/d/yy
m/d/yyyy	M/d/yyyy
mm/dd/yy	MM/dd/yy
mm/dd/yyyy	MM/dd/yyyy
yy/mm/dd	yy/MM/dd
mm-dd-yy	MM-dd-yy
Jan dd,yy	MMM dd,yy
January dd,yy	MMMM dd,yy
Jan dd,yyyy	MMM dd,yyyy
January dd,yyyy	MMMM dd,yyyy
mmdyyy	MMdyyy
mmdyy	MMdyy
dd Jan yy	dd MMM yy
dd January yyyy	dd MMMM yyyy

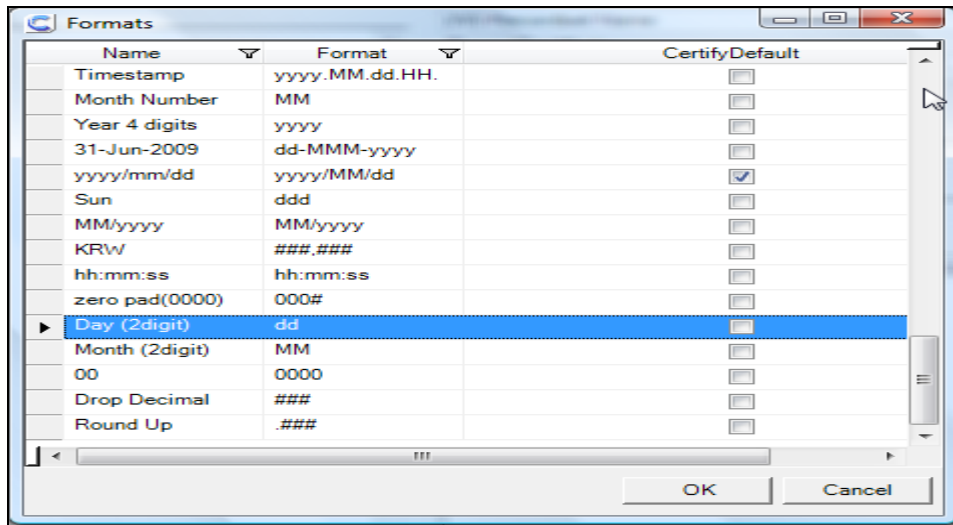


**Must use CAPITAL MM for Month. Lowercase mm is used for minutes.**

# Data Type Format Tool

If the formats (mask) for a particular number or date does not exist, Certify allows you to edit, create, and use new formats. To create a new format:

1. In the Process Editor, click **Tools → Data Type Formats**.  
The Formats dialog box appears.



2. Right-click in the Formats popup window and select **New Format**.  
The New Format dialog box appears.
3. From the **Type** drop-down list, select the format. Options include:
  - Date
  - Number
4. In the **Name** field, type a name. Names can have a maximum of 100 characters. Special characters are allowed, and you can include numbers in the name. Best practices suggest you name the format the same as the actual mask.
5. In the **Format** field, type the new format.
6. Enter a test value and click **Apply Format** to see what your formatted value will look like prior to using the mask.

If you are creating a number format, use one of the following format characters to create your own format:

<b>Currency</b>	C
<b>Decimal Integer</b>	D
<b>Scientific</b>	E
<b>Fixed Point</b>	F
<b>Hexadecimal</b>	X
<b>Zero Pad (10)</b>	0000000000 #####
<b>Fraction (5-digit precision)</b>	# #####
<b>Standard</b>	

If you are creating a date format, use any of the following format patterns to create your own format:

<b>d</b>	The day of the month. Single-digit days do not have a leading zero. Use a lowercase "d" for day.
<b>dd</b>	The day of the month. Single-digit days do not have a leading zero. Use a lowercase "d" for day.
<b>ddd</b>	The abbreviated name of the day of the week. Use a lowercase "d" for day.
<b>dddd</b>	The full name of the day of the week. Use a lowercase "d" for day.
<b>M</b>	The numeric month. Single-digit months do not have a leading zero. You must use a capital "M" to indicate month.
<b>MM</b>	The numeric month. Single-digit months do not have a leading zero. You must use a capital "M" to indicate month.
<b>MMM</b>	The abbreviated name of the month. You must use a capital "M" to indicate month.

# Import/Export a Recordset

---

Certify allows you to export and import data between applications such as Excel using the universal CSV format. One common use is to export a blank recordset containing all the variables as column names in a spreadsheet. The business user can then fill in the rows of data needed to execute the test in a spreadsheet fashion, save the spreadsheet as a CSV file, and import that data into the recordset of Certify.

## Exporting a Recordset

To export a recordset to a .csv file, do the following:

1. Open your project (if not already opened) by selecting **File** → **Open Project** and then selecting your project.
2. In the Navigation taskbar, select **Data**.  
The Data window appears.
3. Expand the existing layouts in the Summary pane. This allows you to view all existing layouts.
4. Locate the layout that has the recordset you want to export.
5. In the Layouts Summary pane, select the layout you want to export so you can edit the data.
6. In the Detail pane, select the **Recordset** tab.
7. Select the recordset, then right-click and select **Edit**.  
The Recordset Editor dialog box appears.
8. In the **Records** field, select a recordset.
9. Click **Export**.  
The Export Recordset File dialog box appears.
10. In the **File Name** field, browse to the path where you want to export the recordset. You can only export recordsets in .csv format.



It's a good idea to select the **Include Header** checkbox, so the column headers are displayed.

11. Click **OK**.  
The information is exported to the .csv file you specified. The **Export Recordset** dialog box closes.
12. In the **Recordset Editor** window, click **OK** to close.

## Importing a Recordset

To import a recordset, do the following:

1. Open your project (if not already opened) by selecting **File** → **Open Project** and then selecting your project.
2. In the Navigation taskbar, select **Data**.  
The Data window appears.
3. Expand the existing layouts in the Summary pane. This allows you to view all existing layouts.
4. Locate the layout that has the recordset you want to import.
5. In the Layouts Summary pane, select the layout you want to add a recordset.
6. In the Detail pane, select the **Recordset** tab.
7. Right-click and select **New Recordset**.  
The Recordset Editor dialog box appears.
8. In the **Name** field, type a name for the recordset.
9. In the **Description** field, type a description to identify the recordset.
10. In the **Records** field, click **Import**.  
The Import Recordset dialog box appears.
11. In the **File Name** field, browse to the patch for the recordset you want to import. You can only import recordsets in .csv format.
12. Click **Open**.



Select the **Ignore First Line** checkbox if you do not want the header row imported.

13. Click **OK**.  
The information in the .csv file is imported.
14. In the **Recordset Editor** window, click **OK** to close.

## Recordset Mode

---

After you add a recordset or recordset variable to your process, you will need to set a recordset mode. From the Recordset Mode drop-down list, select one of the following options:

Mode	When Executed	How Executed
<b>Read Only</b>	Reads recordset at the beginning of execution.	Loops process once for each row until End of File.
<b>Append</b>	Writes recordset at the end of execution.	Appends to existing recordset and loops process until Abort or Exit.
<b>Clear and Append</b>	Writes recordset at the end of execution.	Creates new recordset for each execution session and loops process until Exit.
<b>Read and Update</b>	Reads recordset at the beginning of execution.	Updates the recordset at the end of the process.

- ▶ **Read Only (Default Option)** – reads each row of data in the recordset and executes the test for each row read.
- ▶ **Append** – this option will add rows of data to the end of recordsets with the value of the variables set in the process and/or from prior records read.
- ▶ **Clear and Append** – this option will remove the current contents of a recordset and writes a new row of data for every iteration of the process.
- ▶ **Read and Update** – this option allows the row of data to be written back out and updating any values that may be changed as a result of the process execution.

## Creating and Using Attributes

---

Attributes are created to extend the definitions of Certify components and categorize, collect, sort, and extract data. Attributes apply to all projects in the database. For example, to specify a build identification number for every process in the system, create a build identification number attribute for processes. While executing processes, you are able to specify the build identification number.

You must have permission to create, edit, or delete attributes; however, any Certify user can view attributes. You can add attributes to:

- ▶ Requirements
- ▶ Layouts
- ▶ Processes
- ▶ Results

You can create an unlimited number of attributes and set an attribute as “required”. Required attributes are only enforced when creating or editing an existing variable, process, layout, recordset, requirement, or test result that has attributes associated with it.

## Requirements

---

Requirements are business features or functions that describe how your application should work. In Certify, identifying requirements is the first step in defining the Critical Business Process Certification and in managing the associated risks.

Certify supports requirements validation by allowing you to create hierarchical requirements that can be linked to your business processes. When the processes are executed against the application under test, the pass/fail status of the execution determines whether or not the requirement is satisfied. You can review the status of your requirements in the Requirements window.

Certify also provides a requirement coverage report in both detail and summary modes, as well as customer query and report support for reviewing all or part of your requirements information. All of these features work together to enable you to monitor requirements, manage risks, and make important decisions during your testing process.



No two requirements can have the same name at the same level. Requirements cannot be linked to steps, only to processes.

Child requirements can be created at any level in the hierarchy and are attached to the parent requirement.

A key feature of requirements in Certify is the ability to link them to processes. Both the requirement and process must be created separately before they can be linked together. Requirements can be linked to multiple processes, and processes can be linked to multiple requirements.

After executing a process with linked requirements, the status of the requirements is shown in the Requirements window. Visual indicators are provided to help you quickly review the status.

## Execution Flow Rules

---

Each Certify process contains steps that perform actions against objects of the application that is being tested. In the process step execution, the results are either logged as passed or failed. You can set specific actions to manage the results of the step execution. Some actions that you can set include continuing the execution, executing another process, or jumping to another step. You can also implement an execution flow rule.



Execution flow rules are specific to a project and managed within the Process and Data Editor.

Once an execution flow rule is created, you can link it to any process in that project. Rules linked to a process can direct the execution flow in that process when the rule is selected as an **On True or On False Action** option in steps of child processes.

During process execution, Certify checks the list of execution flow rules associated with the current process. If Certify cannot find an execution flow rule in the current process, it looks for a rule inside each process in the execution stack. The rule is then executed, and the process execution runs to the point specified by the rule action. The process execution continues from that point until execution is complete. If no execution flow rule is found, the status of the step is set to abort. After the execution completes, the Result Viewer displays the results of the steps executed, including the result of the execution flow rule step.

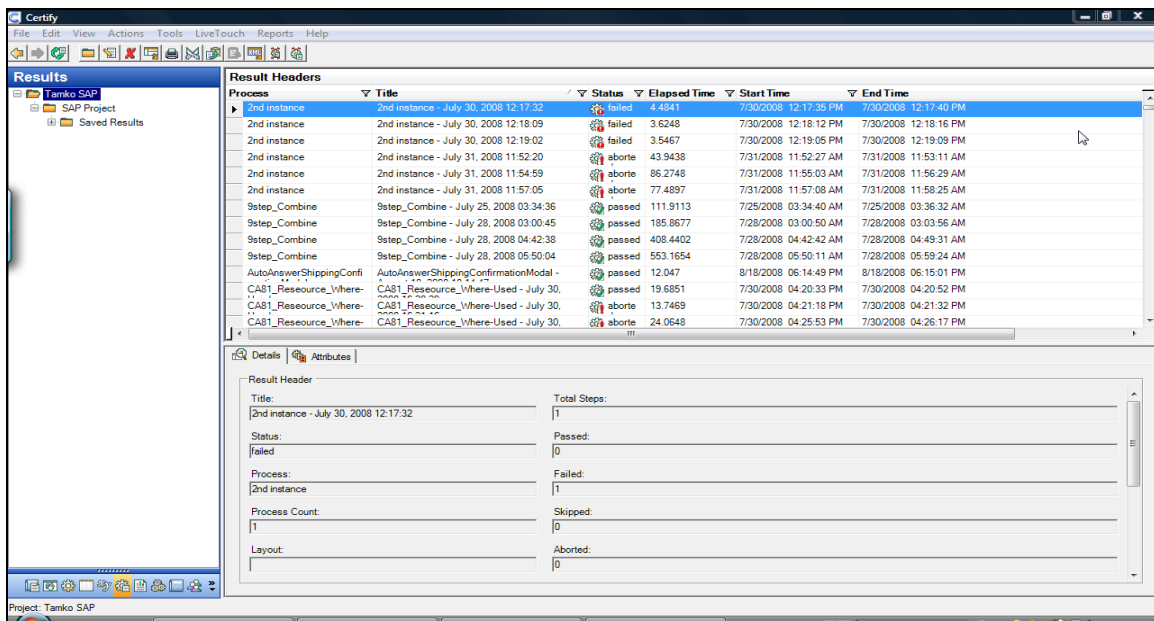


Before removing an execution flow rule from a process, verify that no steps in the current process are using the rule. Removing a rule in a process does not remove the rule from the available rules in the Execution Flow Rules List.

# Result Management

Results are project related and only show after a project has been created and opened. For results of your executions to be shown in Results, you must have created and run a process.

- ▶ You can organize results into folders that you create.
- ▶ You can add as many folders as you want to hold results and you can nest folders beneath other folders.
- ▶ You can copy and paste folders holding results to create new folders with existing results.
- ▶ You can delete folders holding results.



- ▶ Save valid results into a separate folder and remove the debugging results from the root folder.
- ▶ Compress execution results.

As your database grows with execution step results for passed steps, you may want to remove the “passed” steps to reduce your database size and improve performance.



**When you compress your execution results, this information is deleted from the database and cannot be recovered.**

# Processing Naming/Storing Convention

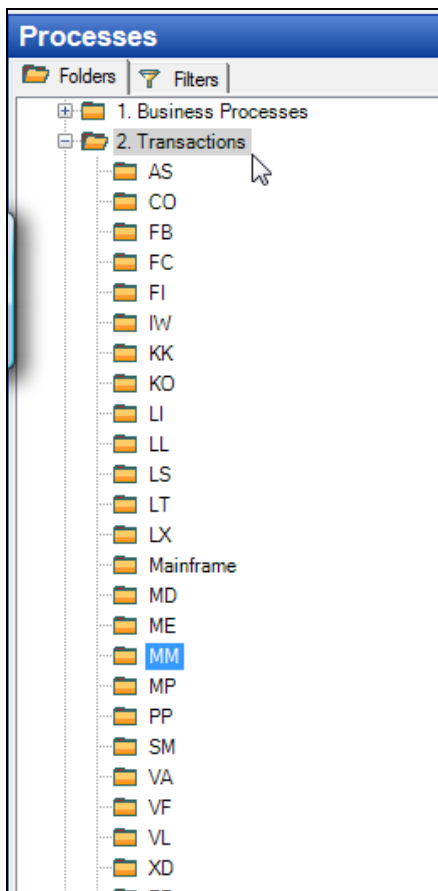
---

## Transaction Name

Create the Transaction name using the transaction code, underscore, and the screen title – all one word. Make sure the first letter of each word is capitalized.


	VA01_CreateSalesOrder
---	-----------------------

Save this new transaction in a folder under "Transactions" called "VA". If the transaction folder does not exist, create it.



## Business Process Name

Create the Business Process name to match the scripted scenarios document.


	<p><b>PM_PM001_Repair_ZM01_Z1</b> will be saved in the PM folder under Business Processes as <b>PM001_Repair_ZM01_Z1</b>.</p> <p>In this case, the functional area is removed from the name.</p>
---	--

## Layouts

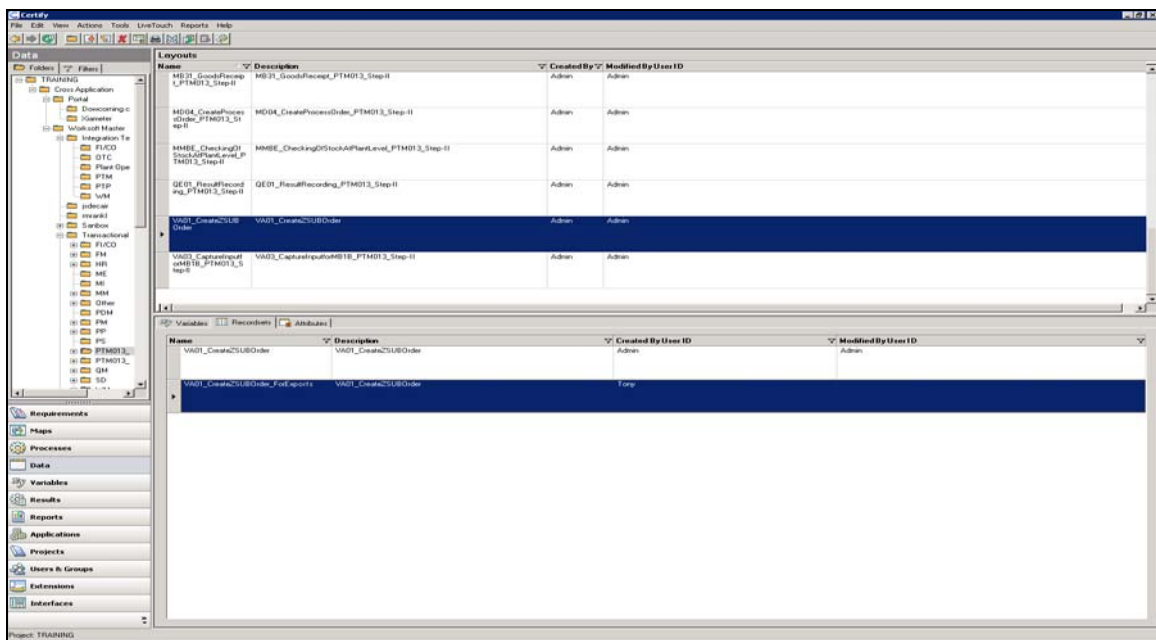
Layouts will be saved in folders the same way transactions are saved. Create a “VA” folder for the layout that will be used to test VA01\_CreateSalesOrder. The layout should also be named the same as the process it’s assigned to.

## Recordsets

Recordsets will be stored in the layout folder. For storing purposes, there is no need to do anything once the layout has been created. The name will be the same as the layout. If more than one recordset will be created, simply add a unique identifier to the end.

	<p><b>OTC009_US, OTC009_nonUS, etc.</b></p>
--	---


You may have different recordset as needed to test a transaction.



## Recordset Rows

When creating recordset rows, consider the following:

- ▶ One row of data, minimally.
- ▶ The process will run from the beginning for each row of data in your recordset.
- ▶ To specify a specific row, insert a step changing the row in the recordset.

	Select layout, recordset, mode = Index and Index = (row)
---	--

Step#	Application Versio	Window	Object	Component Action	Narrative	On True	On False
9	System 1.0	System	Record Set	Read Record	"Index" record in recordset VA01_CreateSalesOrder / VA01_CreateSalesOrder_RS	Continue	Continue


Parameters |  On True / On False

Parameters for Read Record

- Read record from
- Record Set
- Mode
- Index

## Recordset Filter

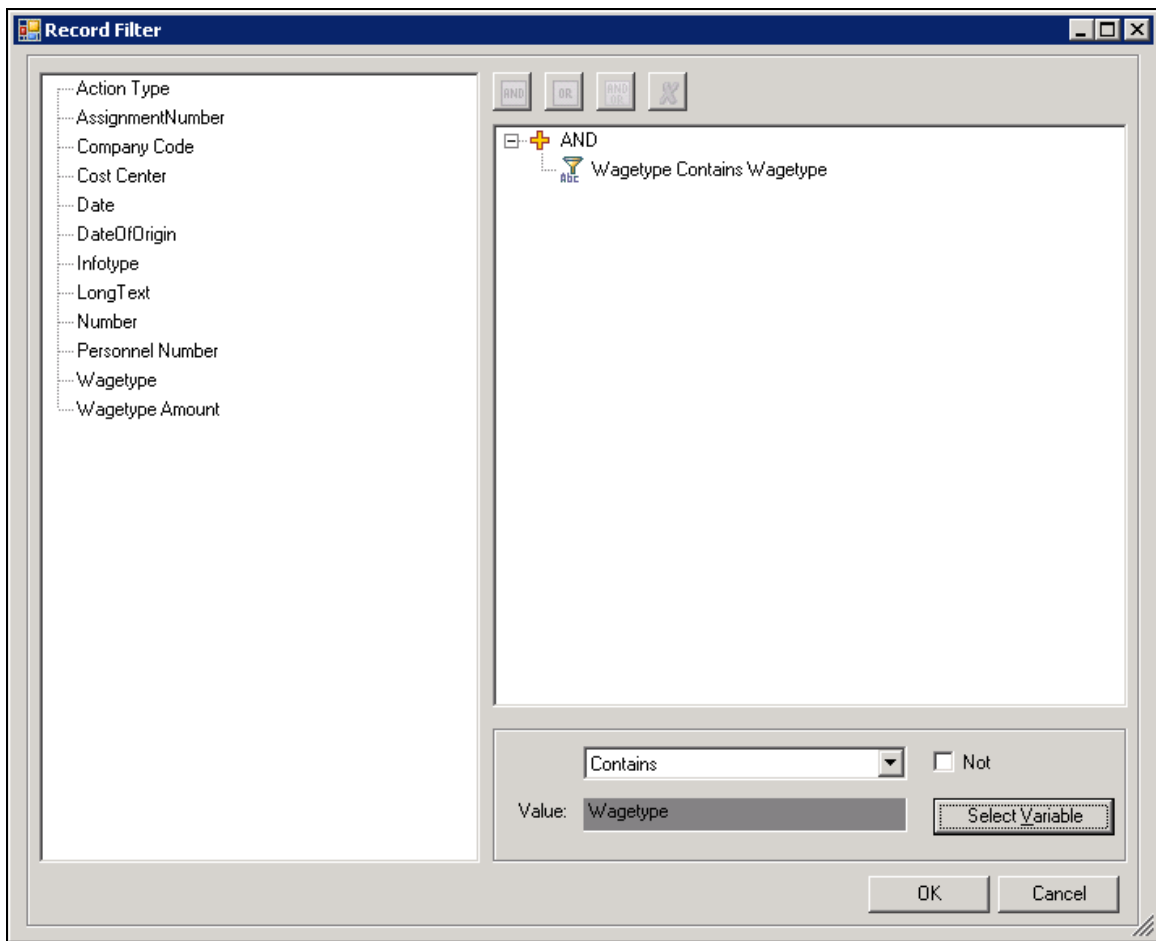
Recordset filter allows you to only use a portion of a recordset that meets certain criteria. This is handled via the recordset filter editor. By selecting certain data elements in a recordset you can specify exactly what you need for a particular test.



**EXAMPLE**

Let's say we have an HR process that utilizes both Salary and Hourly information. Rather than set up a separate test and recordset for each condition, we would simply have a recordset that contained the information for both Salary and Hourly data, and set the filter to use only one or the other based on a variable that may be set from the master record (such as S or H). If the master (parent level) recordset reads a row of data and sets the HR\_Type variable to H, the recordset filter on the child process will then only select the data that matches HR\_Type in that child process recordset.

When using the record filter option, it will look and feel like a SQL statement. Simply grab the data element you will filter on and drop on the + symbol, change the type of condition (contains, starts with, etc.), and finally add the variable to be used to match against. You can do other complex compares as well, such as OR, AND/OR, etc.





1. Find the empty row:
  - ▶ Use LiveTouch to find the table name.
  - ▶ Click **Done** and LiveTouch returns the table name.
  - ▶ In Certify, select the Component Action: **[Find Row]**.
  - ▶ Search for your input columns to make sure they are blank. In this case, we chose column 1 and 2. (see screenshot above)
  
2. Select the value from the combo box:
  - ▶ Use LiveTouch to get the name of the GuiComboBox.
  - ▶ In Certify, select the Component Action: **[Select]**.
  - ▶ In the **Item** field, enter your item. In the case of long phrases, use the first few words. They have to be unique to the list.
  - ▶ Select the Criteria: **Starts with**.
  - ▶ In the **Row** field, type **1**.
  
3. Input data into the Partner field:
  - ▶ Use LiveTouch to get the input field name for Partner.
  - ▶ In Certify, select the Component Action: **[Input]**.
  - ▶ In the **Row** field, type **1**.

This example will work for most table inserts. Remember, for Find Row, look for:

- ▶ the columns that you will be entering data into.
- ▶ a column that has to have a value in it.

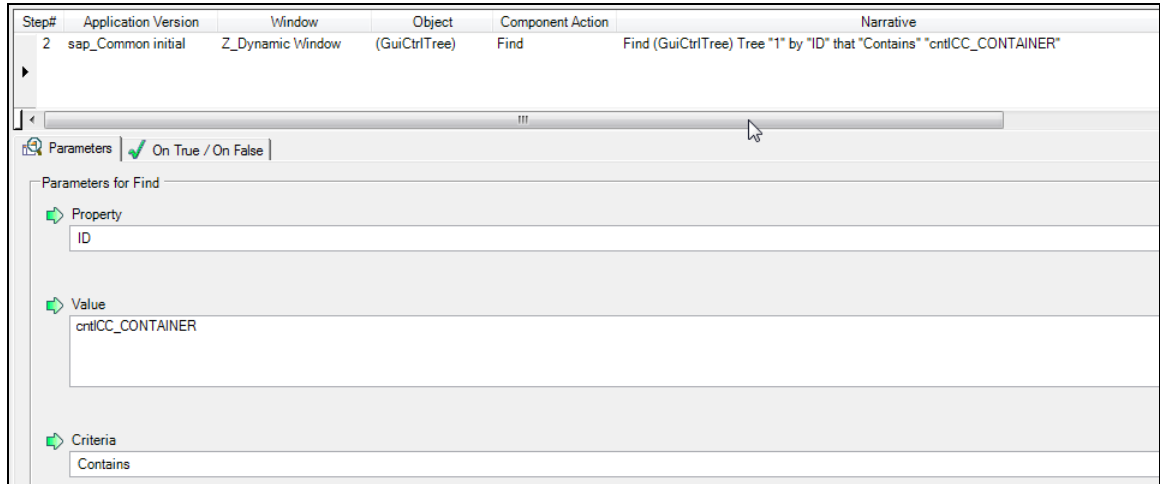


Option fields will be blank on different rows on large tables. This will cause Find Row to choose the wrong row.

## Selecting from a Tree View

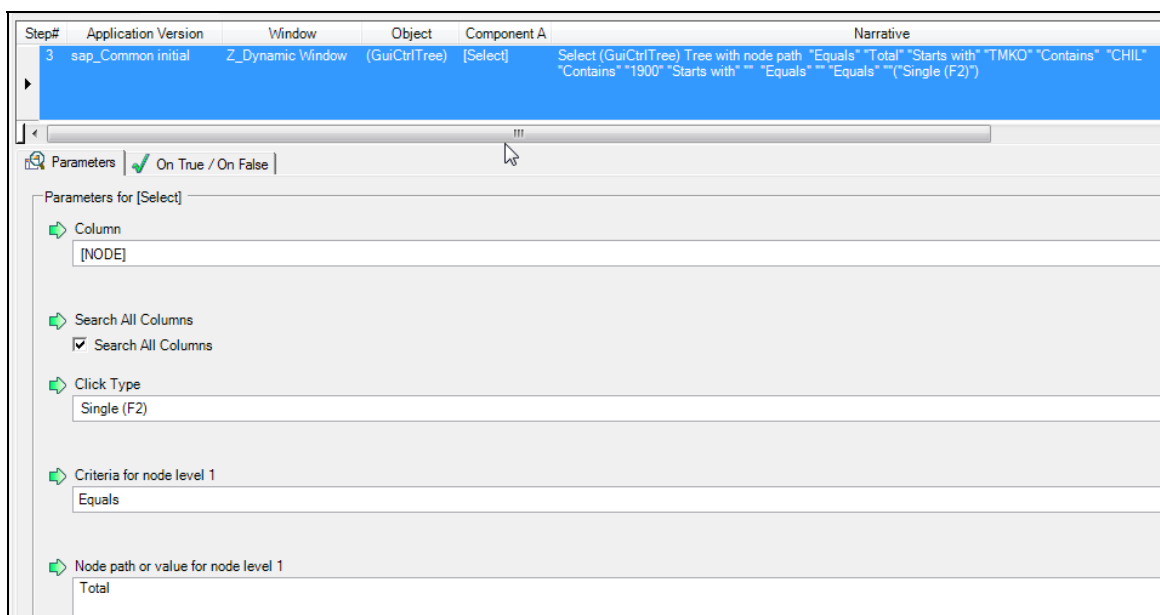
This section will demonstrate the steps for selecting nodes within an SAP tree view using Certify.

1. Find the name of the GuiCtrlTree container. In this case, we used the 'ID' property.



2. Enter the path for the Node that needs to be selected:

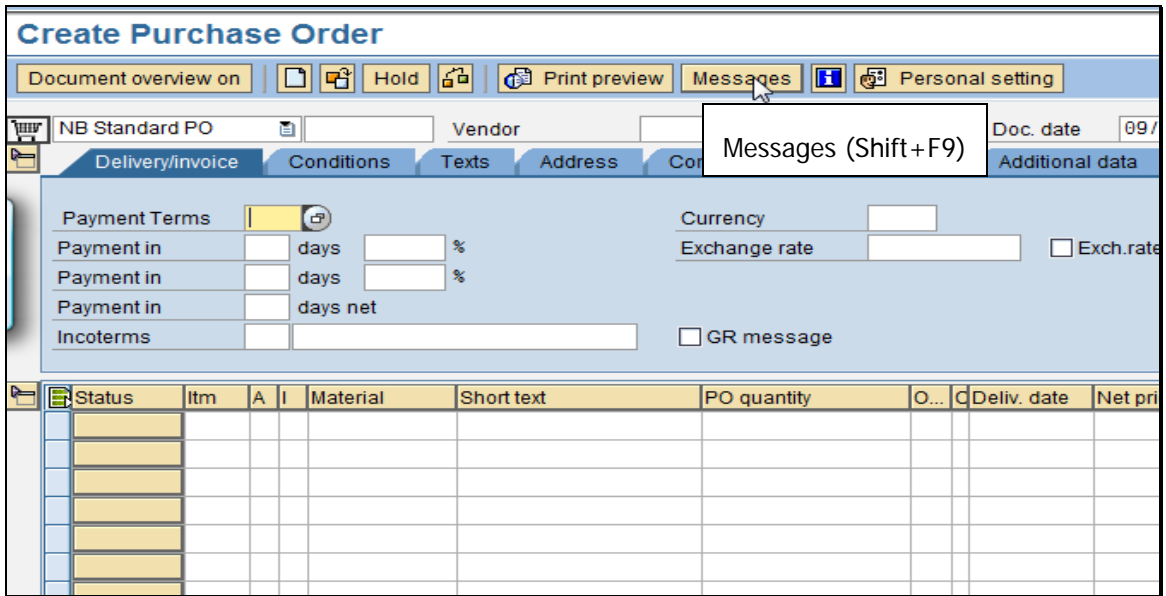
- ▶ In the **Column** field, select **[NODE]**.
- ▶ In the **Click Type** field, select an option from the drop-down list.
- ▶ In the **Criteria for node level 1** field, enter name of first node.
- ▶ In the **Node path or value for node level 1** enter a value.
- ▶ Repeat for each node until you find the one you are searching for.



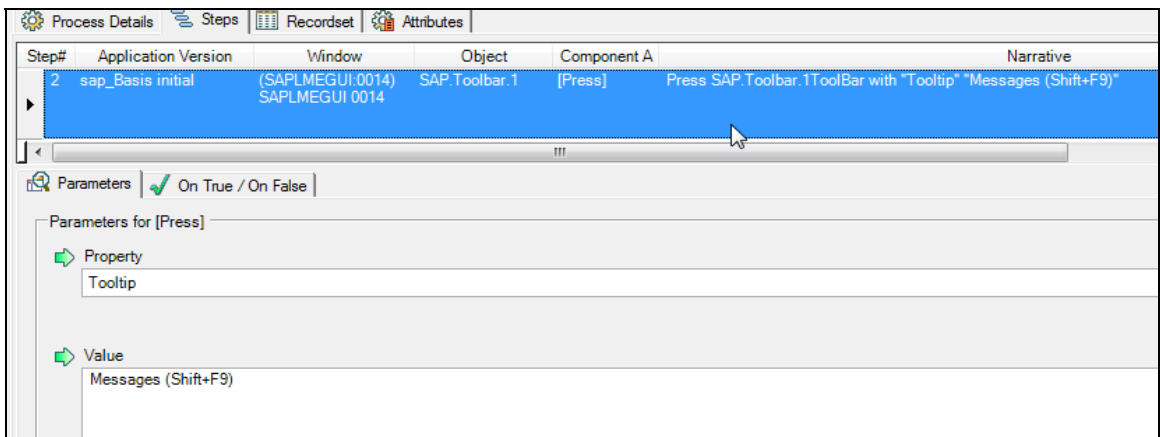
# Tooltips

Tooltips are another way of accessing an object. Below are the steps to click a toolbar button using tooltips:

- ▶ Find the toolbar name using LiveTouch.
- ▶ Mouse over the object to get the tooltip. By mousing over the object you will be able to see the tooltip. (see example below)



Below is the Certify step to click that button:

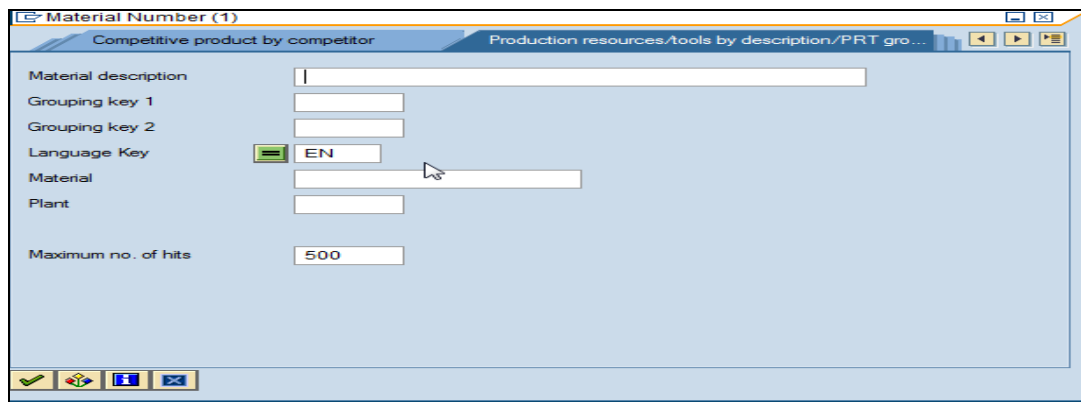


## Dynamic Objects

When using LiveTouch, some objects return with no properties (i.e., name or text). GUI buttons on modal are a good example. Another example is a modal on top of another modal. LiveTouch will not be able to read the top level modal.

The first thing you need to do is find the object, and then use it. Below are the steps needed to automate objects dynamically:

- ▶ Use LiveTouch to find the internal name of the object.
- ▶ Right-click on the name of the object and select **Copy**.
- ▶ In Certify, in the Find Step field, insert the name.
- ▶ Insert another step to use the object you just found.



LiveTouch is not able to read this modal. To enable the green check button, enter these steps into Certify:

### Step One:

- ▶ Select the Application Version: **sap\_Common initial**.
- ▶ Select the Window: **Z\_Generic\_Modal**.
- ▶ Select the Object: **(GuiButton)**.
- ▶ Select the Component Action: **Find**.

### Step Two:

- ▶ The Application Version, Window, and Object are the same as Step One.
- ▶ Select the Component Action: **Press**.



You first want to find the object, then press it. (See screenshot for example)

Process Details | Steps | Recordset | Attributes

Step#	Application Version	Window	Object	Component A	Narrative
3	sap_Common initial	Z_Generic_Modal	(GuiButton)	Find	Find (GuiButton) Button "1" by "Name" that "Equals" "btn[0]"
4	sap_Common initial	Z_Generic_Modal	(GuiButton)	[Press]	Press (GuiButton)Button

Parameters | On True / On False

Parameters for Find

- Property: Name
- Value: btn[0]
- Criteria: Equals
- Instance (number, next, previous, last): 1