# Smart Docketing

“Docketing” is analogous to logging of actions performed in a system. In ShowCase users can manually create docket lines when they take certain actions. ShowCase also automatically dockets after certain pre-defined actions such as scheduling a court date, or adding a charge to a case. Smart docketing is the exact opposite of this process. Smart Docketing is executed with automatic actions when certain docket entries are entered and/or created.

Workflows play an important role in the smart docketing process because that is where the actions are defined. To define a smart docket, we first need to define a docket code.

## Docket Codes

To enable a docket code for smart docketing, the flag “Smart Docket Active” must be set to true. Furthermore, it must be linked to one or more actions or functions that will need to be executed. These actions are referred to as “Smart Docket Functions” or simply SD Functions. Take a look at the Figure-16 below to see how a docket code is setup as a smart docket.

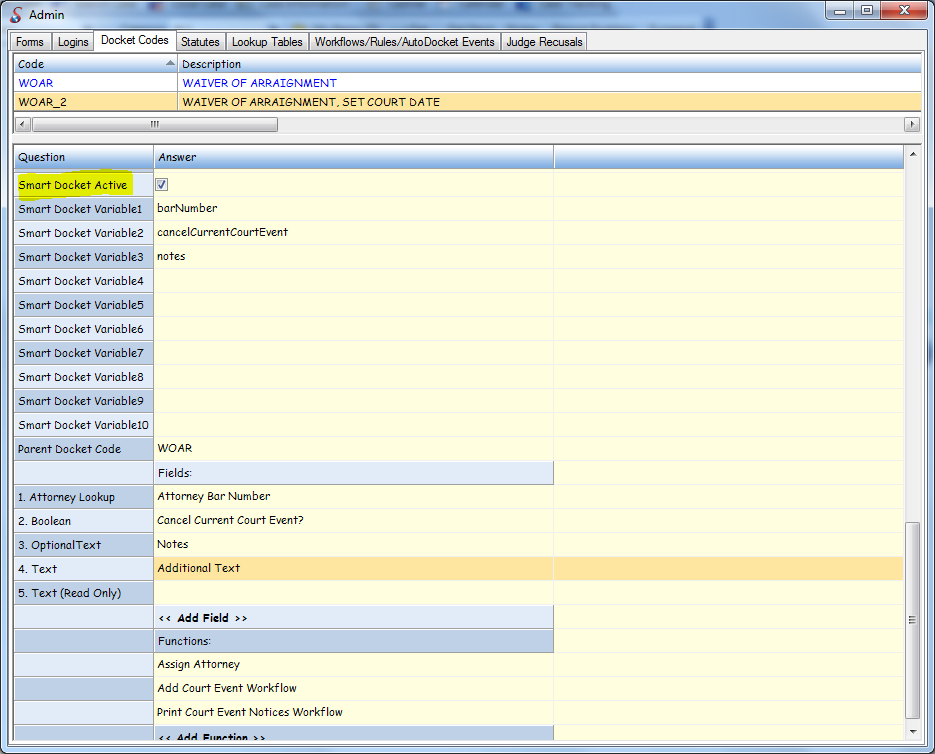


Figure-1: Docket Code Admin Form

Any number of SD Functions can be added to a docket code. All the attached functions are executed sequentially in the order they appear.

A docket code can also be configured with docket fields. A docket field serves as a mechanism to capture input values from the user so they can be supplied to the SD Functions attached to the docket code. For example, in the figure above, the first function “Assign Attorney” requires an attorney’s bar number to be entered by the user. For this purpose, a docket field of type “Attorney Lookup” is configured for this docket code. We will take a detailed look at the SD Functions and Docket Fields in the next few sections.

## Docket Fields

A docket field is a system defined field that can be used to capture input from the user for any smart docket. Common fields such as a “Text”, “Boolean”, “DateTime” etc are readily available. There are several other system defined fields available to use in smart docketing. Table below lists the most commonly used fields. For a full list of all the fields, please refer to the Docket Fields lookup table in the Admin section of ShowCase. The behavior of a docket field is dependent on the type definition of the field – specifically the CalcType. The CalcType dictates how a field behaves when used in smart docketing. There are 15 different types of docket fields defined by ShowCase. Table below provides the definition of these types. New docket fields can be added to the system as long as they belong to one of these pre-defined CalcTypes.

|  |  |
| --- | --- |
| **CalcType (Field Type)** | **Description** |
| 0 | Plain text field |
| 1 | Lookup field. User input is validated against a lookup table based on following criteria:  Calc1 – Name of the lookup table to search e.g, *Agencys*  Calc2 – Function (on the lookup table collection class) that does the lookup  Calc3 – Property (on the lookup table class) that serves as return value of the field |
| 2 | Format Date – user input is formatted to a date time. An error is thrown if the input cannot be converted to a date time. |
| 3 | Static List – Provides user with a static list of values to pick from.  Calc1 – A pipe delimited list of static values. Putting a pipe character at the beginning of the string will add an empty item in the dropdown. |
| 4 | Static Default Text – The value typed in Cacl1 field is the return value of the field. |
| 5 | Calculated Default Text – Performs a calculation to identify the default text.  Calc1 – name of the class  Calc2 – method or property name  Calc3 – Specify if Calc2 is a method or property. 1 for method, 2 for property |
| 6 | Same as CalcType 3, but if the selected value is the same as that specified in Calc2, the ImageRequired property of the docket code will be set to true. This means if the user selects a desired value the system will make sure a document is scanned before executing any functions. |
| 7 | Create a word merge document if the user asks for it..  Calc1 – A static list, typically “Yes|No”.  Calc2 – One of the options in Calc1, typically “Yes”. This means if the user chooses “Yes”, then run the word merge specified in Calc3.  Calc3 – Program control name of a word merge document. |
| 8 | Provides the user with a dropdown list of values. The list is populated using a dictionary object obtained from:  Calc1 – name of the class  Calc2 – name of the function that returns a dictionary.  The return value of this field is the Key value used in the Dictionary. |
| 9 | Boolean |
| 10 | Same as calc type 8, with one of the values pre-selected.  Calc1 – name of the class  Calc2 – name of the function that returns a dictionary.  Calc3 – Name of a docket field of CalcType 5. The field defined in Calc3 is evaluated to identify the default value to pre-select. |
| 11 | Provides the user with a dropdown list of values. The list is populated using a dictionary object obtained from:  Calc1 – name of the class  Calc2 – name of the function that returns a dictionary.  Calc3 – Calc10 – input values to the function defined in Calc2. These input values can be captured from the user input or default values can be specified. A default value is anything that is enclosed in double quotes e.g., “1”, “Hello World” etc. To read the user input a variable name must be used. Then this variable name must be plugged into one of the SDVariable1 – SDVariable10 columns of a docket code. |
| 12 | Provides the user with a dropdown list of values used in a pre-defined enumeration.  Calc1 – name of the enumeration e.g., ShowCaseClassLib.eCaseBalanceLedgerTypes |
| 13 | NOT USED |
| 14 | Returns the value of a ShowCase setting.  Cacl1 – name of the showcase setting |
| 15 | Decimal |

## SD Functions

Smart docket functions are used to express actions that can be executed dynamically. These actions could be simple function calls, popping up user interfaces or executing workflows. No matter what the action is, it must be defined in the lookup table SD Functions. Table below looks at the different elements needed to define a smart docket function.

|  |  |
| --- | --- |
| **Name** | **Description** |
| Description | The name and/or brief description of the SD Function. |
| SD Class Map | The name and path to a class |
| SD Class Function | Name of the function to execute |
| SD Variable 1 – SD Variable 10 | Input parameters to the function. There are 3 different kinds of parameters that can be supplied.  ***CCase*** – This is a hard coded parameter which instructs ShowCase to pass the case object as the input parameter  Static parameter – A default or static value to be supplied as a parameter value. This parameter is entered as string value inside double quotes e.g., “11”, “Hello World”.  Variable parameter – The value of the parameter is unknown at design time, but can be captured from the user when they are ready to execute the smart docket. This parameter is entered as a string that represents a variable name, e.g., *barNumber.* Then this variable name must be entered in one of the SDVariable1 – SDVariable10 fields of the Docket Code itself. |

Table below shows how to define an SD function to execute a workflow.

|  |  |
| --- | --- |
| **Name** | **Description** |
| Description | The name and/or brief description of the SD Function. |
| SD Class Map | ***CGlobal.ProcessWorkflows*** |
| SD Class Function | ***ExecuteWorkflowByName*** |
| SD Variable 1 | ***CCase*** |
| SD Variable 2 | Category name of the workflow e.g., ***“SmartDocketing.Criminal”***. This is a static parameter hence it must be inside double quotes. |
| SD Variable 3 | Name of the workflow e.g., ***“ContinuanceWorkflow”***. This is a static parameter hence it must be inside double quotes. |
| SD Variable 4 – SD Variable 10 | Any input parameters to the workflow |

## Putting it all together

Smart docketing is a versatile process that allows the definition and automation of new processes without the need for versioning the product or expensive customization requests. So how does it all look after the entire setup is done? Take a look at the Figure-17 below.

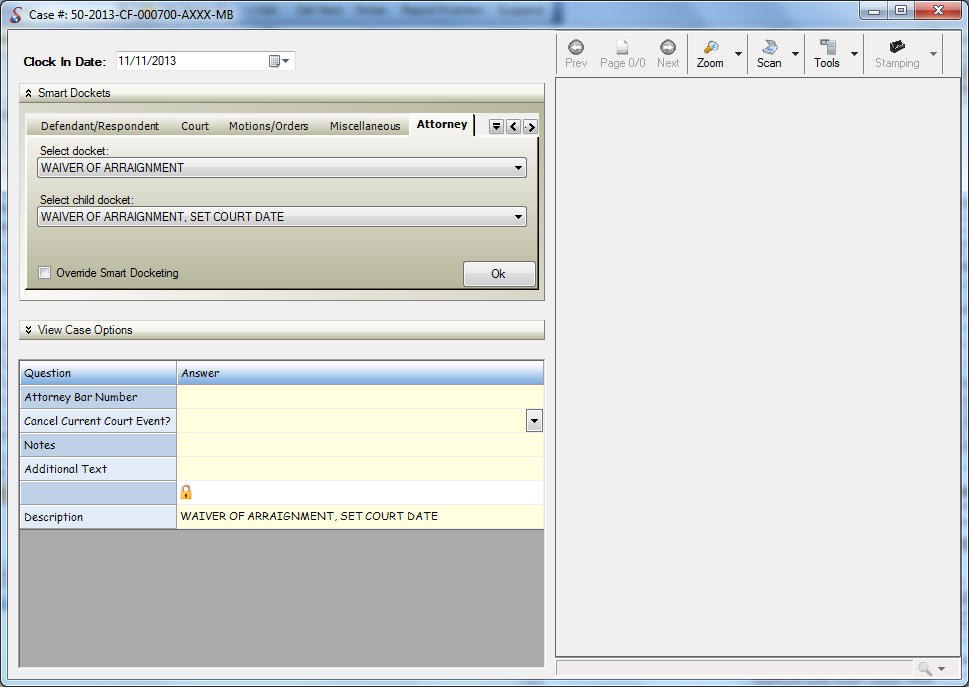


Figure-2: Smart Docketing form

Smart Docketing is a document driven process. To get to this form, Open a case, navigate to the dockets tab and click on the “Smart Docketing” button. The idea is to allow the system to automatically run required processes with minimal data input from the user after scanning a document. A document is identified by a docket Code and the functions attached to the docket code drive the document driven process.

As seen in the figure above, all smart dockets are organized into Categories called Docket Categories. These are user defined and can be assigned to a docket code. Every category is represented in the smart docketing form as a tab. The docket code descriptions appear inside the dropdown list provided under each tab.

A smart docket can also have a parent-child relationship with other smart dockets. The figure above shows such scenario. When a smart docket is selected in the first dropdown, if it has any child smart dockets, they will appear in the second dropdown list. This hierarchy is established by simply setting the ParentDocketCode property of every child docket code.

Once the smart docket is selected, any docket fields configured are displayed in the grid below. These serve as questions to the user to provide answers for. The values entered in this grid are mapped to the input values to the functions and finally by the click of the Ok button, any configured smart docket functions are executed. The checkbox “Override Smart Docketing” can be used to disable the execution of any configured smart docket functions.