

Brand Mobiliser Development Manual

[PRODUCT DOCUMENTATION]



Contents:

1	Introduction	1
1.1	References	1
2	States and Applications	2
2.1	State Machine	2
2.2	Base States	7
2.3	Mobiliser Utility States	7
2.4	Mobiliser Web Service States	8
2.5	Customised 3 rd Party States	8
3	Developing with the Application Composer	9
3.1	State Layout	9
3.2	State Editor	15
4	Example Application	23
4.1	Cash Out Process	23
4.2	Workflow	25
4.3	State Layout	26
4.4	State Editor	27
5	How to Test Applications	28
5.1	Using the Application Simulator	28
6	Importing/Exporting Applications	33
6.1	Export	33
6.2	Import	34
A.	Appendix – Base States	35
B.	Appendix - Mobiliser Utility States	38
C.	Appendix - Mobiliser Web Service States	44
D.	Appendix – Using the Legacy Application Editor View	72
E.	Appendix – Mobiliser Web Service Lookup Values	76

Principal Author

Sybase365 - mCommerce

Revision History

Version 1.1 - March 2011



1 Introduction

Brand Mobiliser makes it very easy for companies to mobilise all aspects of their businesses, including: brand awareness, CRM, mobile banking and financial, mobile payment, commerce, and much more. Brand Mobiliser fulfills the increasing demand of mobile customers for more interactions with their mobile applications.

For partners, Brand Mobiliser provides a plug-in mechanism to develop components to enrich the Mobile Interactive Application experiences even more, allowing integration of any 3rd party systems.

The basis of Brand Mobiliser's web interface is its ability to visually compose, modify and test these conversational services, known as interactive applications. These applications are then fulfilled by the Brand Mobiliser processing engine, which allows for high-throughput session management for customer interaction with these applications.

This document describes the steps and parts of the Brand Mobiliser Web Interface that allow development of interactive applications.

To do this, in the following chapters we introduce and describe;

- States and Applications
- The Application Composer state layout view.
- The Application Composer state editor popup.
- A complete example large application.
- How to test applications.
- How to import and export applications between installations.

1.1 References

1. Brand Mobiliser User Manual; Version 1.1 – March 2011
2. Brand Mobiliser State Developer's Guide; Version 1.1 – March 2011

2 States and Applications

In Brand Mobiliser, states are building blocks that can be combined to model the flow of processes and configured to build interactive applications. This enables you to create customised applications without any knowledge of programming. Brand Mobiliser provides a number of standard and additional Mobiliser states and these can be supplemented with customised states to meet any customer-specific requirements.

Most of the states in Brand Mobiliser Application Composer are accompanied by a short description of their functionality under “Notes”.

2.1 State Machine

A state is an element of a system referred to as a state machine. A state machine defines the flow of processes in an application. An application usually has many states and can include different types of state. A state usually has a previous state and a following state unless it is the initial state or the final state. There can only be one initial state, but many final states are possible depending on the interaction with the user.

The figure below depicts an initial state. This is the first state in an application and has no functionality. The name of this state is “Cash Out Process”. This state is created automatically when an application is created, and cannot be deleted. To add a follow-up state, choose a state from the **Follow-up Selector** (“Select a follow-up state” drop down menu) in the **state editor** as shown in the figure below, and click on the ‘Add Follow-up’ button. This adds a subsequent state. When a new state is added, the state editor will be refreshed to show the configuration items of the newly added state.

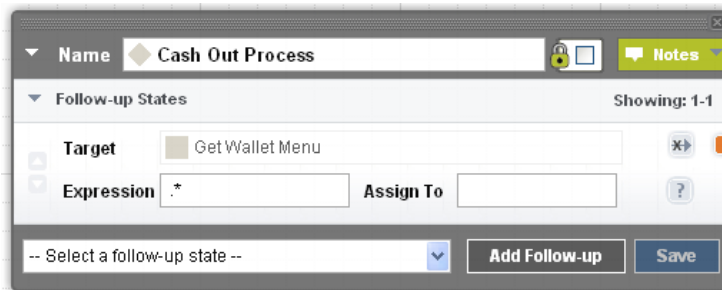


Figure 1. Example Initial State

2.1.1 Keywords and Variables

Some states expect input from the user, for example a keyword, in order to control the flow between follow-up states. The moving between states is known as a **state transition** and can be controlled by matching patterns of text with a supplied keyword.

The keyword may be supplied by the user in response to a ‘Send SMS’ state. Alternatively, keywords may be provided as dynamic outputs from previous state such as Mobiliser or other 3rd party custom states. These dynamic values are the way external systems can communicate specific context information back to the application.

The matching of patterns to the supplied keyword is based on regular expressions¹. Although regular expressions can be sometimes difficult to interpret, they provide for a great amount of flexibility in matching patterns.

Note: Brand Mobiliser version 1.1 introduces a regular expression tester, which offers the ability to test your regular expressions during the development of the application.

A keyword can consist of any combination of characters. Optionally, the keyword input to the state may be saved in a session variable for later use.

Note: In this case, the keyword must be enclosed in brackets ().

For example;

- Using the matching pattern of `'.*'` - a simple match is conducted for the string entered in the keyword field. This pattern will match to any value in the keyword field.
- Using the matching pattern of `'(.*)'` – the same pattern match will be done, but the keyword field can also be assigned to a session variable

In more complex cases, the keyword may be broken into a number regular expression groups, each of which can be assigned to a separate session variable.

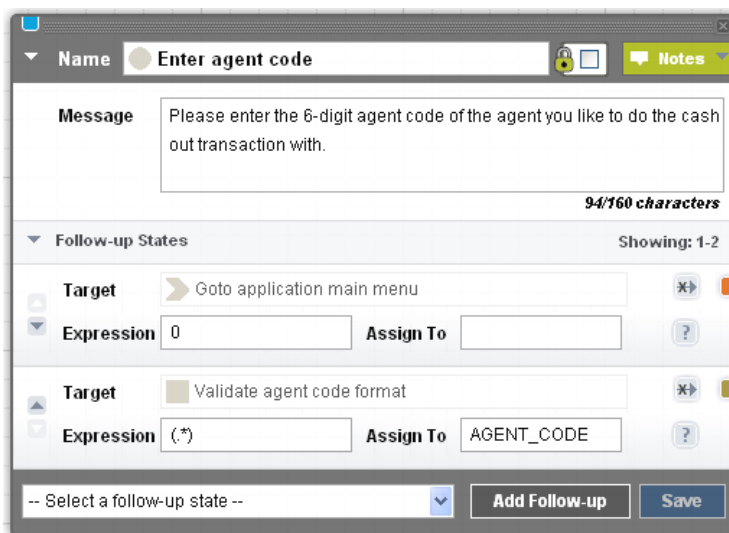


Figure 2. Example State with Keywords, Variables and Transitions

The “Target” identifies the state that follows the current state if the entered keyword is matched.

Note: The sequence of keyword matching is not random.

The first entry in the list is the first matching pattern that is checked, continuing with other checks in the order in which they are shown in the State Editor, but only if there are no previous matching patterns. This means that specific matching patterns should be placed before more generalised patterns. The example in the figure below shows that, a specific response by the user of the single character ‘0’ (a zero) is order before the more generalised pattern of `'.*'` (match any input). In the second list entry, the input received is

¹ See both <http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html> and “Brand Mobiliser User Manual, v.1.0, C. Appendix Regular Expressions”

then also assigned to the session variable named 'AGENT_CODE' since the matching pattern is surrounded with brackets; '(*)'. This allows the user to either go to application main menu, using a specific entry of '0', or to continue with the process by entering an agent code value which will be subsequently validated.

The arrow buttons on the left side of the dialogue box enable you to move a matching pattern up ▲ or down ▼ the list. Use the button with cross over a the transition line ✕ to remove a transition or matching pattern from the list.

Using the Keyword Tester

The Keyword Tester is a pop-up window that can be accessed by clicking the ? icon on the right hand side of each 'Expression' and 'Assign To' entry fields for a follow-up state. The tester will open and populate the 'Expression' and 'Assign To' fields of the tester to the values associated to the follow-up state against which the icon is.

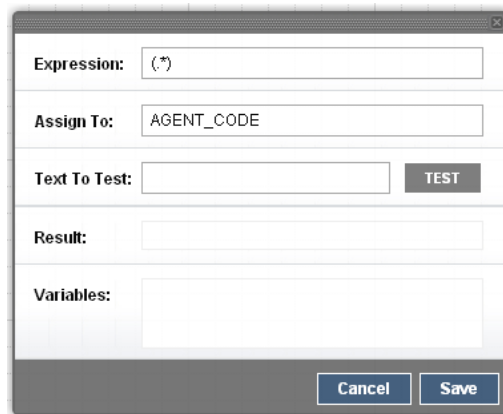


Figure 3. Keyword Tester Pop-up

The tester allows you to assess the suitability of the expression and the assignment of matched parts of a text input to variable. To test an expression you must enter arbitrary text into the 'Text To Test' entry field, then click the 'Test' button.

The test result will be shown in the 'Result' field – either 'Matched' if the expression matches against the entered text, or 'No Match' if there is a no match for the expression against the entered text.

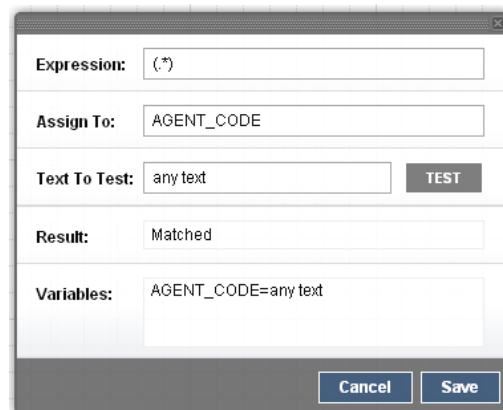


Figure 4. Keyword Tester Pop-up – Basic Expression



The 'Variables' field will show all matching groups and their assigned values. The figure above shows this result for an arbitrary piece of text.

The keyword tester will also indicate if there are;

- matching groups that are not associated with a variable, or
- if there is a variable that is not matched.

The following figure shows example expression values for both of these cases.

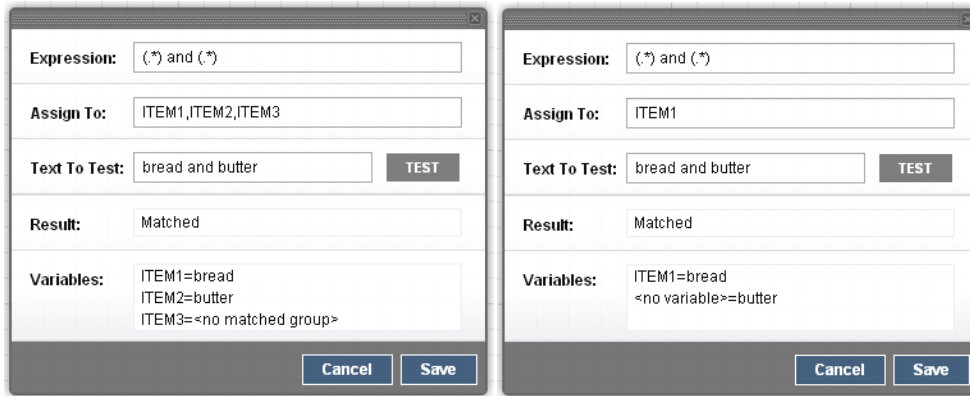


Figure 5. Keyword Tester Popup – Non-Match Groups and Variables

To close the keyword tester, click on the 'Cancel' button or the popup closer icon on the top right of the popup window.

To close the keyword tester and to copy any values in the 'Expression' and 'Assign To' fields back into the state editor popup window, press the 'Save' button'.

2.1.2 Input / Output Parameters:

The Mobiliser and 3rd Party custom states may optionally have input and output parameters.

The input variables allow these states to receive input from the user or from another state or application.

The output variables allow these states to save a parameter in session variables that can be used by another state or to another application.

Input Parameters

Input parameters are the information a state requires to perform its task.

All input variables may be based on constant values or taken from a variable in the current user's session. For each input variable there is as follows;

- **Checkbox:** If the checkbox for an input variable is ticked, the text entry field expects the input of a variable that has to be retrieved. If it is not ticked, the input value will be constant value as written in the field.

For example, if you want to pass the value of a session variable named "CURRENCY", write the variable name "CURRENCY" in the variable entry field and ensure the checkbox is ticked. In this case, whatever value the session variable "CURRENCY" has at the point this state is reached is passed into this state's processing.

- **Entry Field:** Field for the input of a variable name or a constant value (as determined by the checkbox).
- **Description:** A short description of the input type.

The figure below shows an example state that has two mandatory inputs (designated with the red mark next to the description). These mandatory inputs are being provided by two different session variables (i.e., AMOUNT and CURRENCY). The other two inputs are not mandatory, but in this instance are supplied as constant values.

Input Variables		Showing: 1-4
<input checked="" type="checkbox"/>	AMOUNT	Amount to be formatted.
<input checked="" type="checkbox"/>	CURRENCY	The currency of the amount.
<input type="checkbox"/>	10000	Minimum valid amount.
<input type="checkbox"/>	5000000	Maximum valid amount.

Figure 6. Example State with Input Parameters

Output Parameters

Output parameters are a state’s response to an action. All output parameters are available as variables:

- **Checkbox:** The output parameter’s checkbox will always be ticked because the output is always stored in a variable. The checkbox is not editable.
- **Entry field:** This is the session variable into which the output is stored.
- **Description:** A short description of the output type.

The figure below shows an example state that has a single output. In this case, the session variable ‘AMOUNT_FORMATTED’ is assigned this output value of this state.

Output Variables		Showing: 1-1
<input checked="" type="checkbox"/>	AMOUNT_FORMATTED	The formatted amount

Figure 7. Example State with Output Parameter

For all Mobiliser and 3rd party states, there may also be a specific;

- ‘OK’ (or success) transition,
- ‘Fail’ (or failure) transition, or
- ‘Dynamic’ transition – based on a keyword match.

These transitions define which state follows the current state after a programmatic event (that is a choice made directly in the state code regardless of any keyword match).

The ‘OK’ and ‘Fail’ transition do not use keywords or pattern matching to decide which transition to follow. Which transition to use will be based on code in the state and validation provided by or events in, the back-end system being interfaced with.

The configuration of an ‘OK’ or ‘Fail’ transition is entirely based on the individual state being used, and should be described in the state’s description or ‘Notes’ area. Some states will not require the configuration of one or either of these transitions.



Note: If the state does require one or either of these transitions and a follow-up state is not specified, the application will terminate at that point.

For 'Dynamic' transitions, the state code has the option to return a keyword value, which then provides the input to the pattern matching mechanism as described in the section above named 'Keywords and Variables'.

This dynamic transitioning may be used as an alternative way of transitioning to different successful or a failure outcomes of the state (and so may replace the 'OK' or 'Fail' transition). Or, the dynamic transitioning may communicate additional information back to the application on certain validation problems, for example.

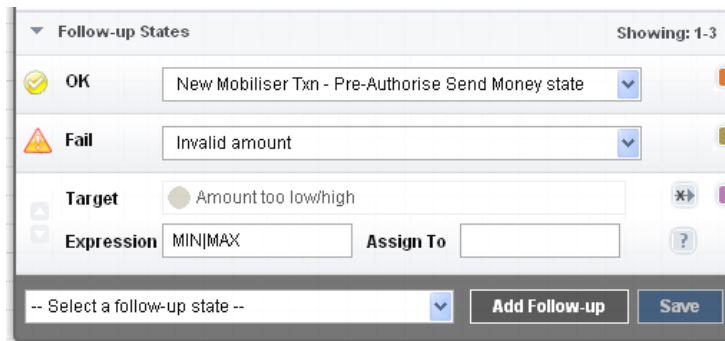


Figure 8. Example Success, Fail and Dynamic Transition

The example in the figure above shows a configuration of a state with an 'OK', 'Fail' and a single dynamic transition. The dynamic transition uses the regular expression 'MIN|MAX' will match against the dynamic values of 'MIN' or 'MAX' that are returned by this state after unsuccessful validation of its input.

2.2 Base States

Brand Mobiliser currently offers a small number of base states. These states provide stand-alone functionality without dependency or interaction with the external services.

Examples of base states include:

- Send SMS – Sends a text message to the customer identified in the current user session.
- Goto Application – Causes the program to flow into another application.
- Set Variable – Sets a session variable for the current user session.

These states are described in detail in the section 'Appendix – Base States'.

2.3 Mobiliser Utility States

Brand Mobiliser also offers a small number of standard utility states. These states possess stand-alone functionality without dependency or interaction with the external services, but are provided with the 'Mobiliser Core States Plugin'.

Examples of Mobiliser utility states include:

- Counter - Increment a named counter.
- Copy Variable – Copy the contents of one session variable into another.
- Method Call/Return – Calls into another application and returns to the called from application.

- Regex – Match a session variable to a regular expression.

These states are described in detail in the section ‘Appendix – Mobiliser Utility States’.

2.4 Mobiliser Web Service States

In contrast to base states and the Mobiliser utility states, which have stand-alone functionality, Mobiliser Web Service States rely on the functionality of the Mobiliser Core which uses open XML standards to trigger integrated and third party applications.

The state concept initially included only stand-alone states and was extended to include states that integrate the Mobiliser Web Services in order to meet the requirements of financial services such as m-Payment and m-TopUp, e.g. money-transfer, prepaid-recharge, view-transactions, view-balance, change-pin, etc. These states include the following functions:

- Get customer information.
- Get a customer’s payment instrument details.
- Get the account balance for a payment instrument.
- Get details of the last transactions for a customer.
- Authorise a P2P transaction.
- Pre-authorise a P2P transaction.
- Verify a customer credential.
- Change a customer credential.

The next section, ‘Example Application’, contains details of some Mobiliser Web Service States and the section ‘Appendix – Mobiliser Web Service States’ describe the complete set of states in more detail.

2.5 Customised 3rd Party States

Additional customised states are developed on an as-needed basis to extend the functionality of Brand Mobiliser to implement client-specific requirements.

Customised components are typically developed by:

- Sybase 365 to implement client specific requirements.
- Third parties as plug-in applications to meet specific client requirements (e.g. storing license plate data for an m-Parking application).

To integrate new custom states requires Java development to a provided API and customisation of the product to include new states package. This customisation is described in a separate document called ‘Brand Mobiliser State Developer’s Guide’.



3 Developing with the Application Composer

The key to effective development of interactive applications is the easy-to-use visualisation of the workflows involved in the business processes that are to be mobilised. This is the primary purpose of the web-based user interface.

3.1 State Layout

The Application Composer state layout view is the primary way of visualising the processing steps of the application workflow, by visualising the states and drawing the transitions between these states. Much as a generalised visual workflow tool would do, the Application Composer allows the application designer the ability to;

- visualise states in the application using an automatic layout,
- dragging and dropping of states to re-arrange layout and to persist the modified layout,
- highlight the context and dependencies, the transitions of states, and
- zooming in and out of the application composer view to see a complete or partial application layout.

The Application Composer also provides complete facilities to traverse through the workflow through a state editor popup window. The state editor popup's purpose is to allow configuration of the state's settings and values. This is described separately in the next section.

3.1.1 Visualising States

The example in the figure below shows the layout of an existing simple application, called 'Mobiliser Counter'. This application uses a counter to increment a session variable, which is shown to the user who responds with an instruction to repeat or to go back to a menu.

Each step in the application, or state, is shown as an image on the canvas, tagged with its name. The canvas has gridlines displayed to show where a state may be located. States are described in more detail in the next section.

The state icon shows;

- the name of this state instance centered on the icon,
- the type of the state in the bottom bar of the icon, and
- a watermark pattern to also help signify the type of the state.

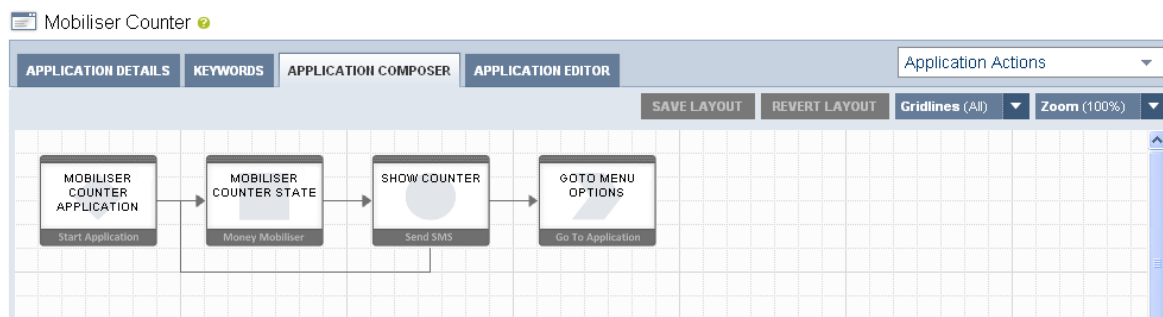


Figure 9. Example Application Composer View

The transition between one state and the next is shown as a directional arrow between the two related states. In more complex applications, transition lines may overlap others.

The buttons specific to the Application Composer view are;

SAVE LAYOUT – stores any changes made to the layout back to the database.

REVERT LAYOUT – re-loads the stored application layout.

Gridlines – Allows the option to display all the grid-lines, a partial grid line or none at all.

Zoom – Allows the option to zoom the Application Composer view panel both out and in so that applications with a large number of states can show the complete workflow on one page (see example in figure below).

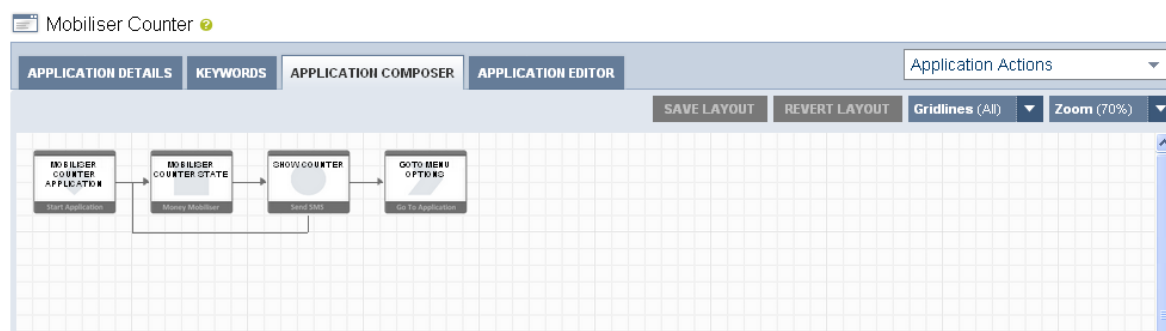


Figure 10. Example Zoomed-Out Application Composer View

3.1.2 Re-arranging State Layout

On first entry to the Application Composer view, any states that have been added to the application are laid out automatically on the canvas. Occasionally, it is worth to re-arrange the layout of the states, to get a better view of the transitions between the states, particularly when transition lines are overlapping.

The state icons on the canvas are sensitive to drag-and-drop into fixed grid positions on the canvas.

To start a drag;

- move the mouse over the state icon you want to move,
- left-click on the mouse and hold the click down, and
- drag to an alternative grid position

The figure below shows that when being dragged, the state icon will become transparent and target grid positions will be highlighted when the mouse enters that grid area.

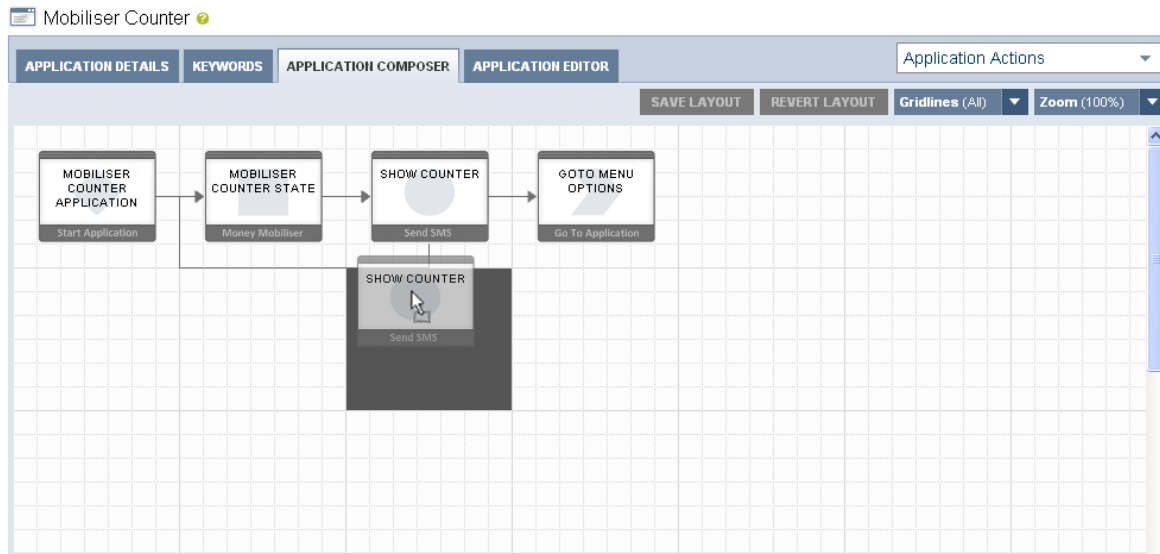


Figure 11. Dragging a State in the Application Composer View

Note: The canvas will not allow absolute free-form positions and each state will be snapped-to a specific grid position as highlighted when being dragged.

Releasing the left mouse button while positioned over a target grid position will cause that state icon to be moved into that grid position and all transition lines into and out of that state will be automatically re-drawn. The figure below shows the outcome of moving a number of the states in the figure above into different grid positions.

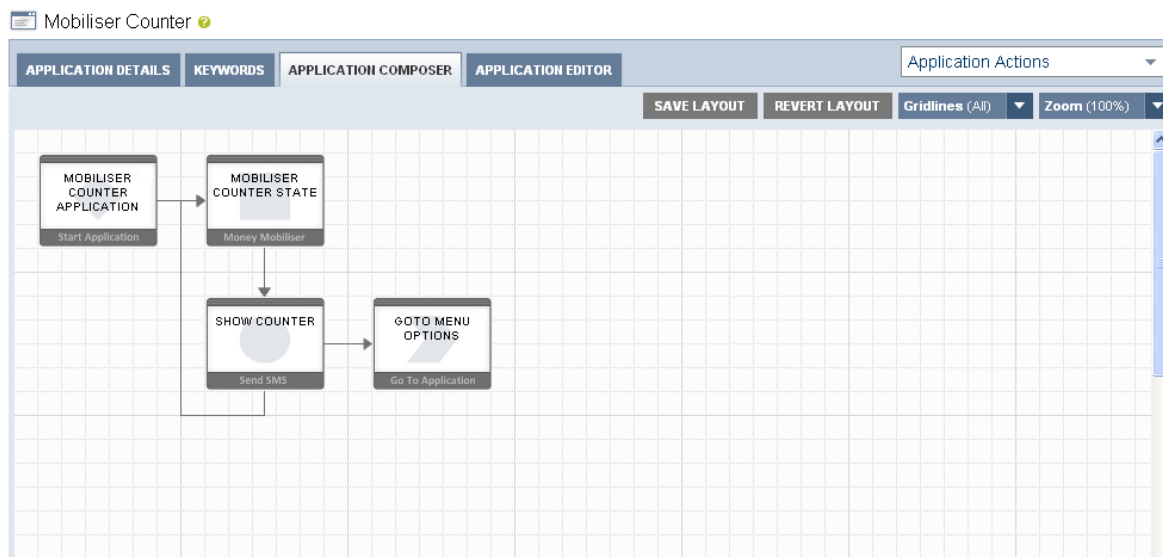


Figure 12. Moved State in the Application Composer View

The 'SAVE LAYOUT' button and the 'REVERT LAYOUT' button will be enabled when state has been moved or there are layout changes. The 'SAVE LAYOUT' button will save the layout to the database. The 'REVERT LAYOUT' button will reload the last saved layout.

Note:

- **Currently, the transition lines are always automatically positioned and there is no way to move a transition line itself.**
- **To ensure that the current layout change is stored, you must click on the ‘SAVE LAYOUT’ button to persist the change to the database. The layout change will be lost if not saved and you move off of the Application Composer view. This includes selecting a different tab for the same application.**
- **If you have zoomed out from the default 100% view, you must reset the zoom level back to 100% before making any layout changes.**

3.1.3 Context and Dependency Highlighting

As described previously, states are linked in the application through directional transition lines. In simple cases, these lines clearly show the linked states; the state from which the transition is from and to. However, in many instances there will be a series of lines coming out of and going into a single state

To get a better sense of the context and dependencies of an individual state, any state can be ‘highlighted’. Highlighting a state allows all of its transition lines, that “go to it” and “go from it”, to also be highlighted along with the states they come from or go to. It also opens the state editor popup for this state.

To highlight any states;

- move the mouse over the state icon you want to move, and
- left-click on the mouse and release.

The dependent states and transition lines are then highlighted in different colours:

- The highlighted state is shown with a dark grey surround.
- Any states that transition TO the highlighted state have a blue surround, with the transition line emboldened in blue.
- Any states that are transitioned FROM the highlighted state have an orange surround, with the transition line emboldened in orange.
- In some cases, a state has both a transition FROM and a transition TO another state. In this case the highlighted state have a part blue/part orange surround, with associated transition lines coloured blue or orange.

The example below shows the simplest case where the ‘Start Application’ state is highlighted, and is shown with a dark grey surround. In this case, there are no states that transition back to the start state. However, the ‘Mobiliser Counter State’ is transitioned TO, and so is displayed with an orange surround and the transition line is emboldened in orange.



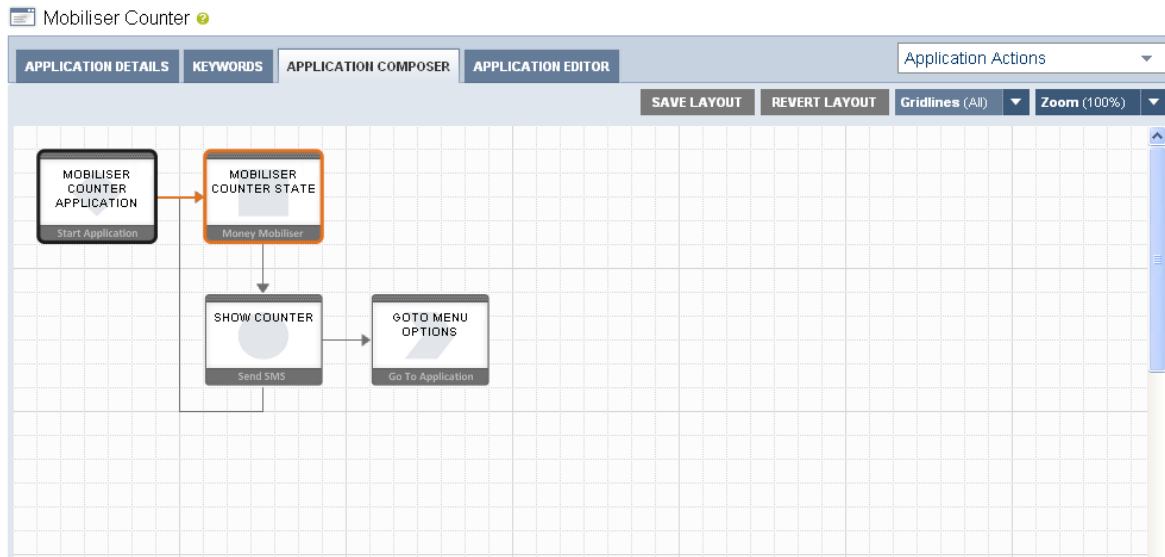


Figure 13. Highlighted Start State

If we instead highlight the 'Mobiliser Counter State' by clicking on it, we will see that state given a dark grey surround, as in the figure below.

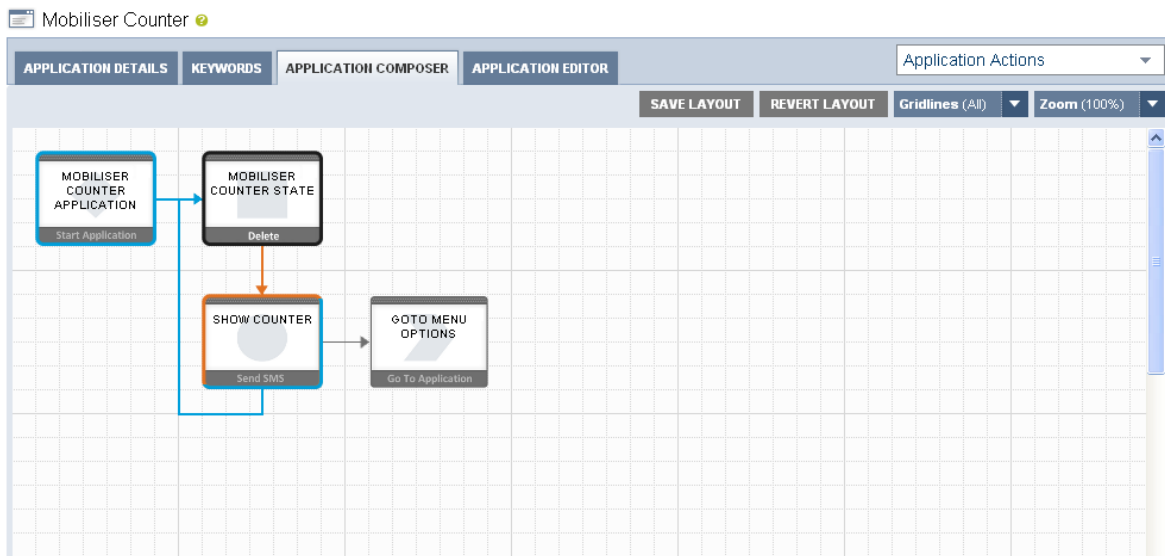


Figure 14. State with Entry and Exit Transitions Highlighted

Since the 'Mobiliser Counter Application' state transitions TO the 'Mobiliser Counter State', it is given a blue surround with that transition emboldened in blue.

The 'Show Counter' state has a transition FROM the 'Mobiliser Counter State' and is therefore has an orange emboldened transition line.

In this configuration, it is more complex because the 'Show Counter' state also has the configuration to transition back to the 'Mobiliser Counter State', hence a second blue emboldened transition line into the 'Mobiliser Counter State'.

So, because the 'Show Counter' state is both a transition TO and a transition FROM state for 'Mobiliser Counter State' we show a part orange/part blue surround. Transition TO lines are always drawn to the left or top of a state, hence the orange part surround in that area. Transition FROM lines are always drawn from the bottom or the right of a state, hence the blue part surround in that area.

Note: To remove all the highlight, surrounds and bold transition lines, click again on the highlighted state.

The figure below, shows the exit point to this application, a 'Go To Application' type state, as highlighted, with the only transitioned FROM state being the 'Show Counter' state, with a blue surround.

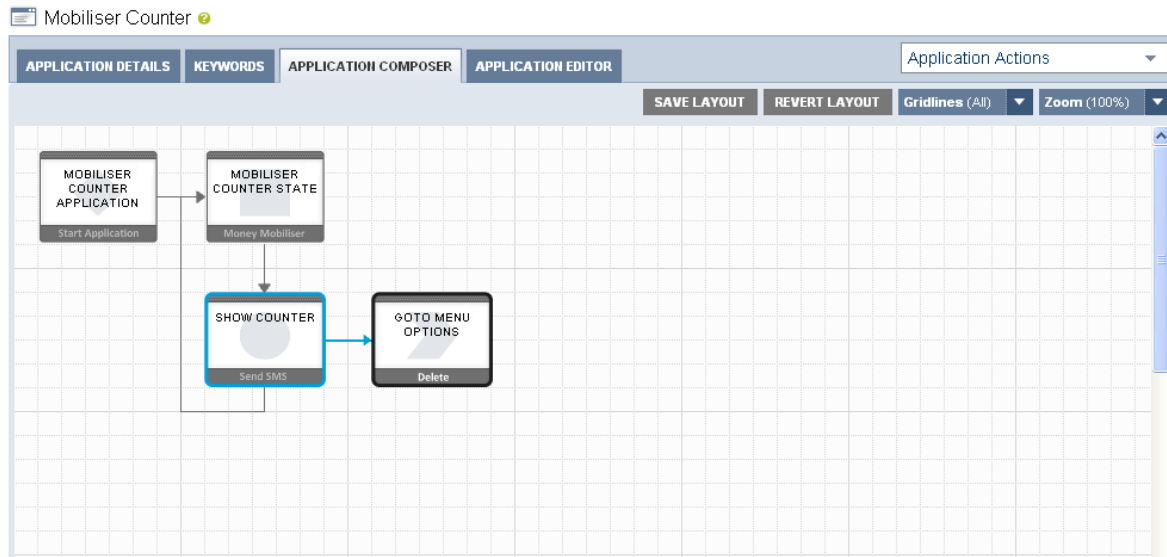


Figure 15. Exit State Highlighted

The final figure below shows how the context and dependency highlight helps in a more complex application, with many states and overlapping transition lines.



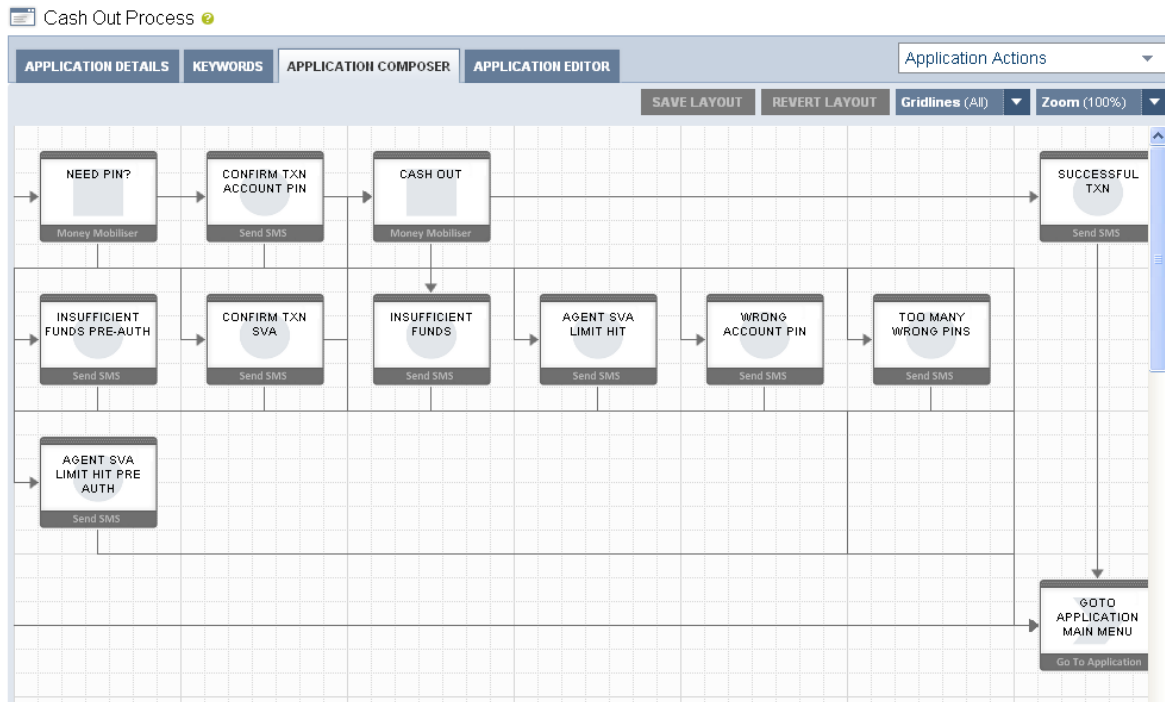


Figure 16. More Complex Cash Out Application

Note: Context and Dependency Highlighting will still work if you have zoomed out from the standard 100% zoom level.

3.2 State Editor

The Application Composer also provides facilities to create new states, delete old states, traverse through the workflow, modify the configuration of a state and test keywords for a follow-up transition.

All this is possible through a state editor popup window.

3.2.1 The State Editor Popup

The popup window will open automatically when a state has been highlighted, as described in the previous section. Depending on the type of the state that has been opened the state editor will show a number of options, context sensitive hyperlinks and entry fields.

The figure below shows what a general Mobiliser or 3rd Party Custom state may look like in the state editor popup. This is a typical state only and different state types have differing configuration capabilities.

Each of the labels are annotated as follows.

1. **Entry Nodes** – An entry node identifies a link to another state that flows TO this state. Entry Nodes are click sensitive, and if you click on it you will open the state editor for the state that this transition relates to. Mouse over on an entry node will show the state name associated with it.
2. **State Type Watermark and Icon** – Alongside the state's name is a watermark icon associated with that state type. The same watermark is shown as the back to a state icon on the layout view, and is also shown in the 'Target' of the 'Follow-up States' section. The watermark allows for quick recognition of different state types in the layout view and the popup window.

3. **Popup Drag Area** – The state window can be moved anywhere inside the application composer page. To move the state click on the dotted header area and drag the state to the desired position.

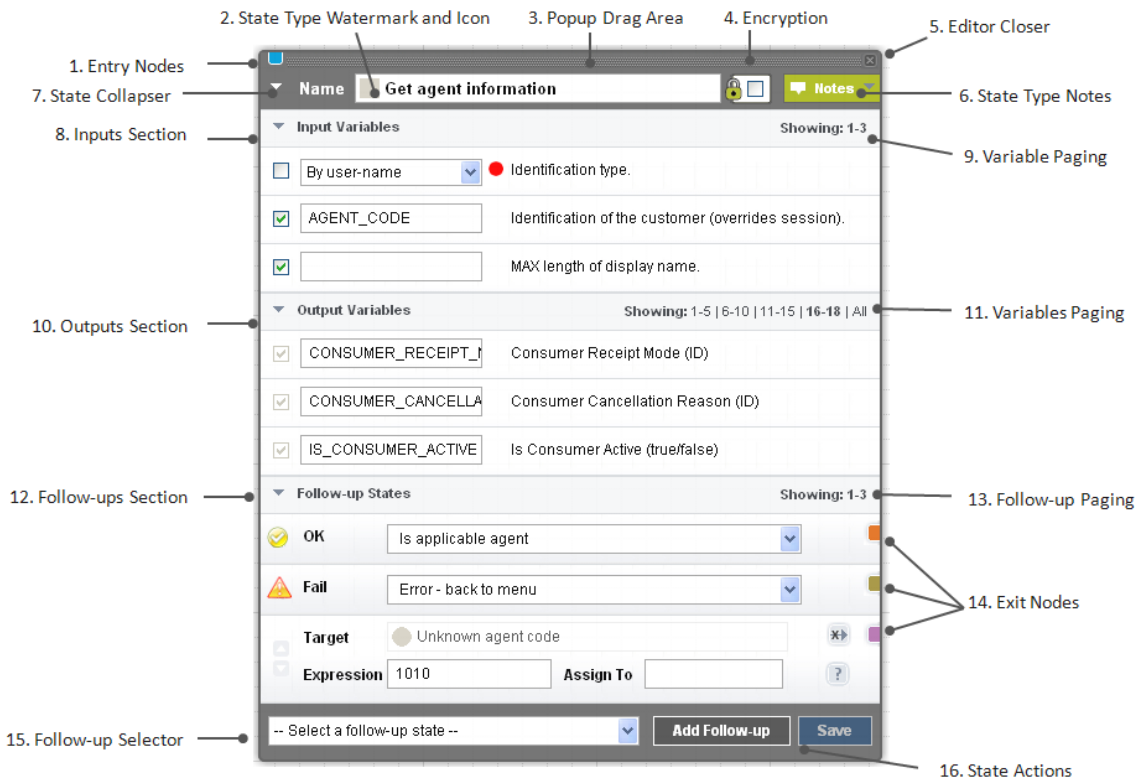


Figure 17. State Editor Popup

4. **Encryption** – The state encryption option allows a state to never show the messages it sends out or inputs it receives back in clear text in the message logs. This is a security feature to allow passwords and PINs to be restricted.
5. **Editor Closer** – Clicking on this icon will close the state editor popup window. If you have pending changes that haven't been saved you will be prompted to either save or discard these changes, as shown in the figure below. If you discard the changes you will continue with the close. If you saved the changes you will have to repeat the close action.

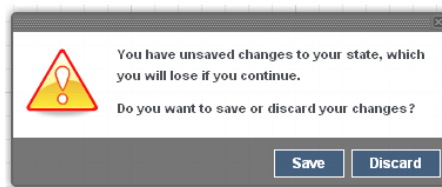


Figure 18. Pending Changes Save/Discard Message

6. **State Type Notes** – Each state type has the ability to describe its function, its required inputs and outputs and any follow-up state transitions that are expected. By clicking on the 'Drop-Down' Notes icon, you will see this text. An example is shown in the figure below. Clicking on the icon again will cause the text to be hidden.

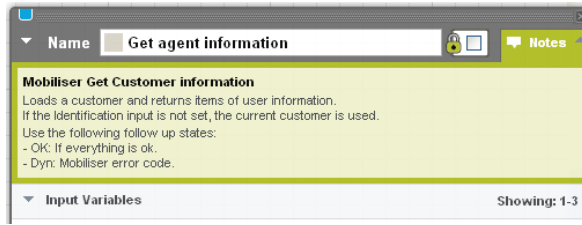


Figure 19. State Type Notes Drop-Down Text

- State Collapser** – The state editor popup is always displayed on top of the layout view and can take up a large amount of space. The state collapser can be used to show and hide the main contents of the state, for example to temporarily minimise the amount of space taken up by the state editor popup, as shown in the figure below.



Figure 20. Hidden State

- Inputs Section** – This is an optional area and is only shown for certain state types. It lists the expected input variables (as described in a previous section). This section can be closed independently from the whole state, or other collapsible areas, by clicking the icon next to the 'Input Variables' text. The figure below shows this section collapsed.



Figure 21. Hidden Inputs

- Variables Paging for Inputs Section** – The inputs section can contain a varying number of entries (depending on the state type). Normally, only five variables are shown at any one time, to minimise the vertical space used by this section. Therefore, if there are more than five variables, you can page through the others by clicking on the relevant page set. Clicking the 'All' link will cause all the variables to be displayed, which should be used with caution, because it may cause the state editor window to take up a large amount of space and go beyond the viewable area on the page.
- Outputs Section** – This is an optional and works the same as the 'Inputs Section' described above.
- Variables Paging for Outputs Section** – There are separate paging options for the output variables, and they work the same as the 'Paging Options' for inputs, described above.
- Follow-ups Section** – This area is displayed for most types of state (but not all) and allows the configuration, change and test of follow-up states. It also allows the collapse of the follow-up states section. When this section is collapsed, the 'Exit Nodes' are shown on the follow-up sections header, as shown in the figure below.

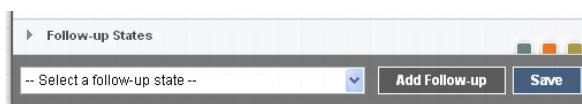


Figure 22. Collapsed Follow-up States and Exit Nodes

13. **Follow-up Paging** – Paging through the follow-up states works the same as the Inputs and Outputs paging, except there are only three follow-up states shown per page, because of the amount of additional vertical space used by Follow-up state ‘Expression’ and ‘Assign To’ fields.
14. **Exit Nodes** – An exit node identifies a link to another state that flows FROM this state. Exit Nodes are click sensitive, and if you click on it you will open the state editor for the state that this transition relates to. Mouse over on an exit node will show the state name associated with it.
15. **Follow-up Selector** – All states, except the ‘Go To Application’ state, allow adding a follow-up state. All possible follow-up states are available from this selector.
16. **State Actions** – The available actions for a state are shown here, normally ‘Add Follow-up’ and ‘Save’. The ‘Save’ button is only enabled when there are changes that have been made to the state configuration.

3.2.2 Adding States

After creation of a new application, there will always be the ‘Start Application’ as the initial state.

New states are always added from the state editor of existing states, therefore the initial state acts as the seed point of the application. The figure below shows the Application Composer view for a new application, after the ‘My First Application’ state has been clicked, and hence highlighted and the state editor has been opened.

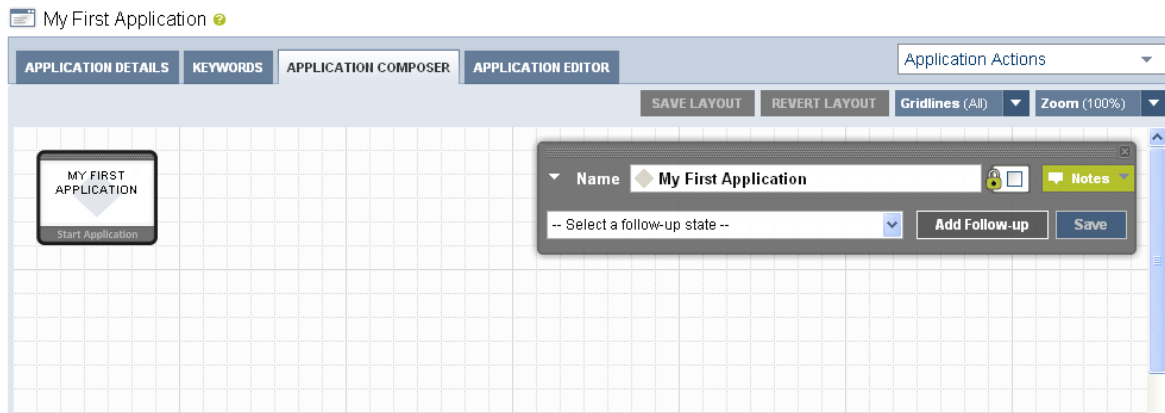




Figure 23. Start Application State

This state allows for two main actions, identified by the icons on the state;

	<p>Remove this state.</p> <p>Note: The delete icon will appear once the state is selected by clicking on it. The delete icon is not shown in the figure above because you cannot remove a ‘Start Application’ state, but it will shown on any other type of state.</p>
	<p>Add a follow-up state, as selected from the follow-up selector to the left of the button.</p>



To add a transition from the “Start Application” state (i.e., My First Application), choose a state from the follow-up selector, and then click the **Add Follow-up** button. The figure below shows all the available follow up states.

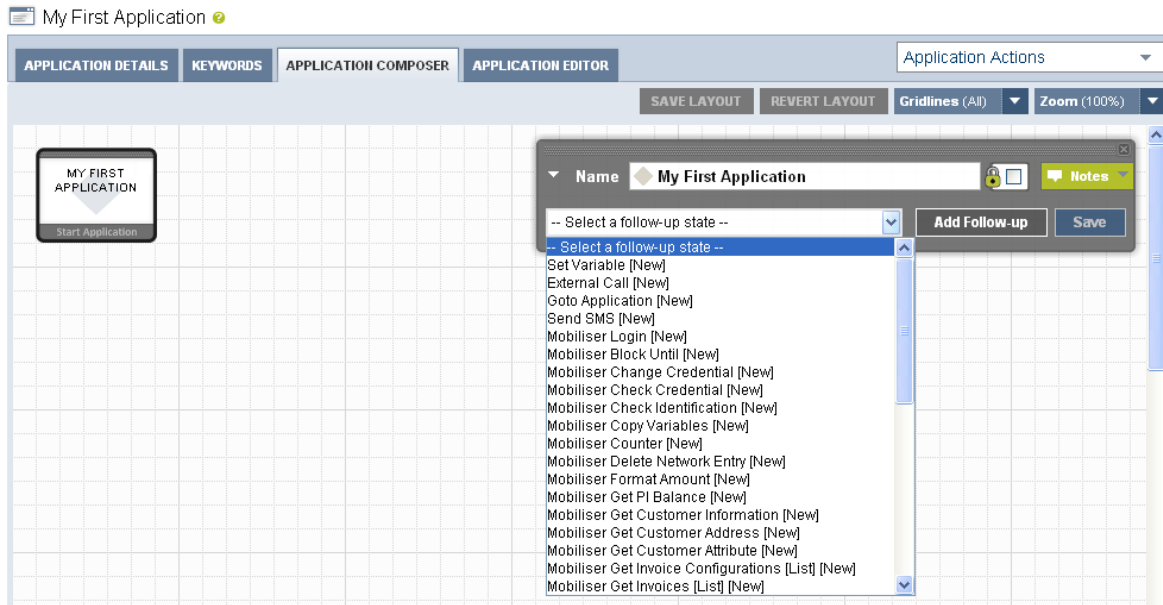


Figure 24. Choose Next State

Note: For a new application, you can add new instances of states. When you have created some states, you can instead add a follow-up to an existing state. This will result in the application’s control-of-flow going to the point of that existing state.

The following figure shows the Application Composer view, after adding a new ‘Send SMS’ state.

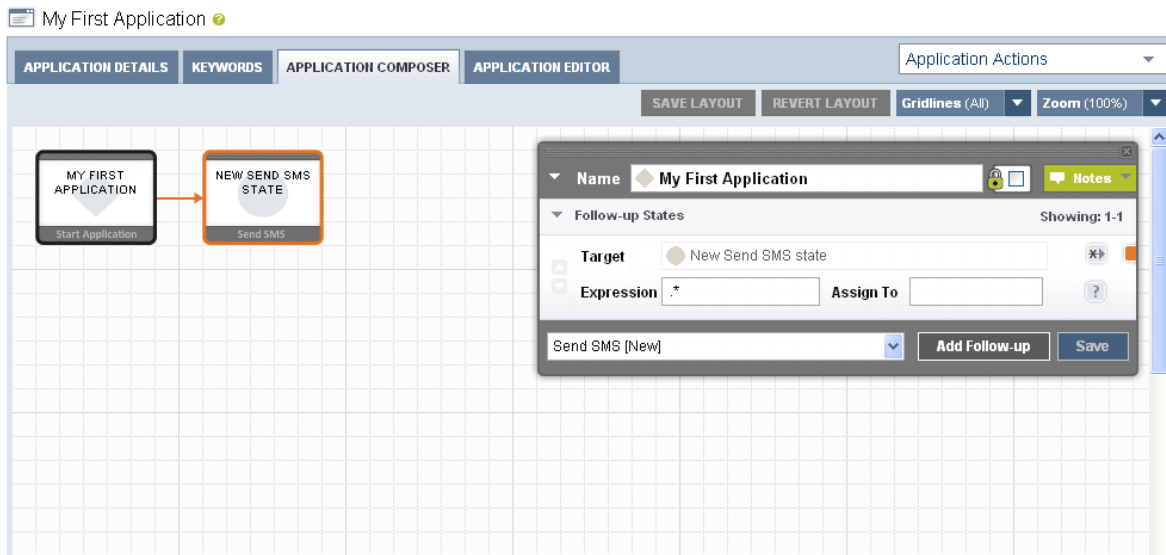


Figure 25. New State Added

Note: After a new state is added, it is automatically assigned the name 'New <type of state> state'. This name is shown in the layout state icon and the follow-up section. It is recommended that you change the name of the state as soon as possible to something relevant, as state names are not unique.

Note: After a new state is added, it is also automatically assigned the keyword pattern of '.*' (match to any and all characters).

To open the 'New Send SMS state', click on that state in the layout view to highlight it and allow adding of a follow-up state from it.

The figure below shows what the application looks like after a follow-up state has been added which is an existing state (the initial state). This application has been modified so that if the user replies to the message with the response 'again', they will see the message again, otherwise the application ends.

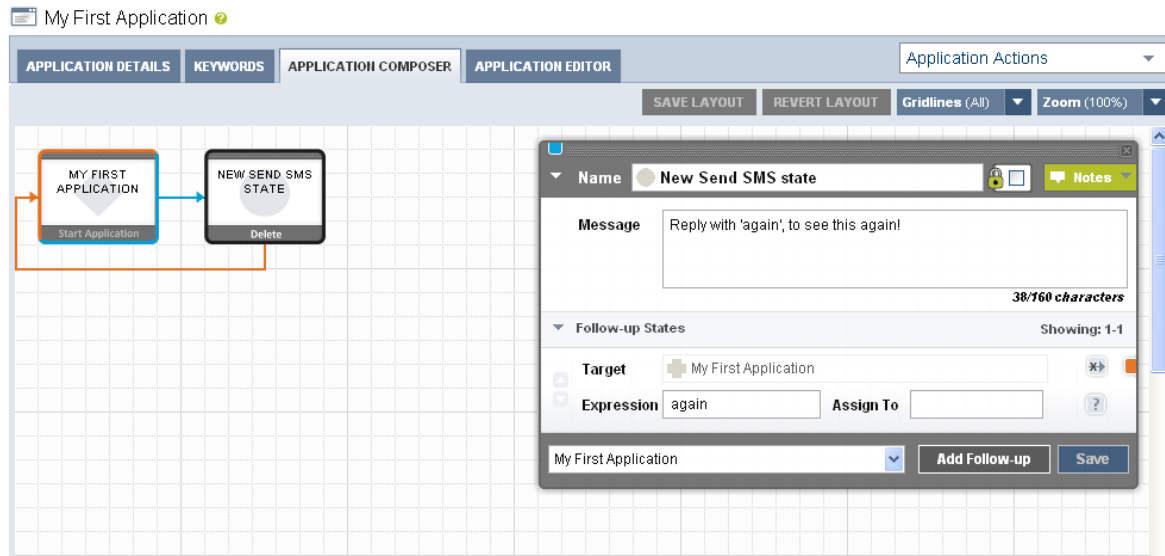


Figure 26. New Link to Existing State Added

3.2.3 Modifying State Configuration

All states can be modified by clicking on the state in the layout view to open its state editor popup. Some changes that are made on the state editor popup are immediately stored into the database. These are;

- Adding a new follow-up state and transition.
- Adding a transition to an existing state.
- Removing a transition to an existing state.
- Moving a transition up or down in the list of follow-ups.

Other changes that are not immediately stored into the database require the user to click on the 'Save' button in order to store the changes are;

- Change to state name.
- Change to state encryption option.
- Change to message text.
- Changes to input variables.
- Changes to output variables.



- Changes to follow-up state 'Expression' and 'Assign To' values.

3.2.4 Removing States and Transitions

To change any state configuration you click on the state in the layout view to highlight that state and open the state editor popup.

Applications may have states and transitions removed from them. It is important to understand the different implications of removing either a state or a transition, and the resulting changes made to application.

Removing a State

Removing a state will permanently delete that state and any transitions that were associated with it. To remove a state click on the area of the state icon in the layout view that is marked as 'Delete'. You are prompted to ensure you know the implications of deleting the state, and can continue or cancel the delete action.

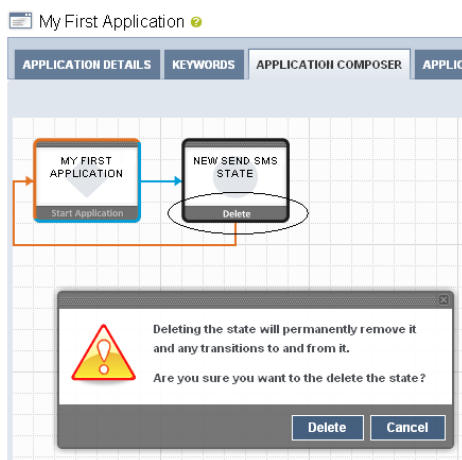


Figure 27. Delete State and Confirmation

- States that transitioned INTO the deleted state will have effected transitions.

Note: Deleting a state that had its own follow-up transitions may also then detach that portion of the application from the application flow itself.

A portion of the application that is detached is not shown in the Application Composer but still exists in the system. Detached portions of the application are never reachable by the customer through the application flow. To re-attach that portion into the application you can add a new state or add a transition that flows into a state in the detached portion.

Removing a Follow-up Transition

Removing a transition will permanently delete that transition, but will not remove the follow-up state that it was associated with. To re-attach the follow-up state into the application add a new transition using the follow-up selector.

Click on the remove follow-up icon next to the follow-up state to remove the transition, as shown in the figure below.

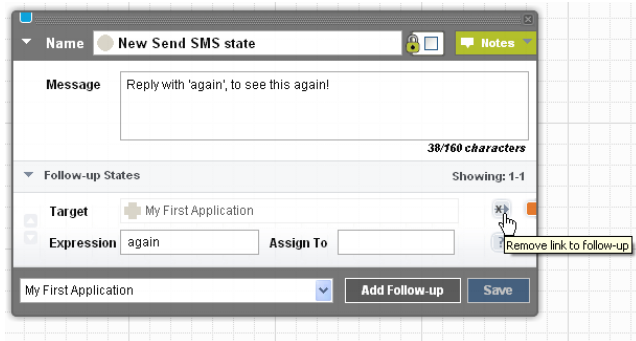


Figure 28. Remove Follow-Up Transition



4 Example Application

Sybase365 Brand Mobiliser offers a unique and highly personalised, cashless way to manage financial services more efficiently. It allows customers to conveniently redeem vouchers on any phone, remit money domestically, pay bills automatically, manage their accounts remotely, and enjoy financial benefits and other incentives as reward for their continued usage.

An example of one such financial service would be a Cash Out service. This service is provided through a specialised mBanking and mPayment service that enables mobile subscribers to pay bills or services through any mobile phone.

The Cash Out service transfers money from the customer's Money Mobiliser account to a Money Mobiliser representative's account. The Money Mobiliser representative then pays this money, as cash, to the customer.

4.1 Cash Out Process

The Cash Out process is similar to many financial oriented processes that can be implemented easily using SMS (or in USSD, a kind of session-based SMS).

Brand Mobiliser manages a unique user session throughout the process that maintains the context of the conversation the user is having with the application.

This Cash Out process would be one sub-process provided through a number of interactive applications. The applications are linked by either 'Goto Application' states, where the flow-of-control moves to the referenced interactive application, or a 'Mobiliser Method Call' state, where the flow-of-control moves temporarily to the referenced interactive application, before returning back to the application that called it.

A complete mobile service is formed from a number of interactive applications that are normally fronted by a PIN or password entry stage, so that we can somehow identify or validate the MSISDN of the customer. Although there is no internal customer list held in Brand Mobiliser, it is expected the back-end systems, such as Money Mobiliser will do that validation of customer through the Mobiliser or other 3rd party states. This Cash Out process assumes we already have a validated customer session on entry to this application.

After successful customer validation it is normal to offer a series of menus through SMS, where the user can choose from the options available to them. In this case, Cash Out would be one menu option related to the mobile financial services that may be offered to them. Again, this Cash Out process assumes we enter the process from an option on a menu, and if there are problems, or if the user wants to finish the process, or at the end after a successful transaction, we will revert back to the menu applications.

Brand Mobiliser enables you to configure the workflow of any application, like the Cash Out process, by creating a sequence of successive states. You can create these states with just a few clicks and without any knowledge of programming. However, a knowledge of both the business process, potential outcomes and how the states work to provide their individual functions, is required.

The Cash Out process is a series of simple steps:

1. Customer chooses an account to cash out from.
2. Enter the agent code who the customer wants to perform the transaction with.
3. Enter the transaction amount.
4. Validate and pre-authorise the transaction by checking for sufficient funds in customer account, the amount is between limits and the agent's SVA is correct.
5. Customer enters their account PIN, if required, or confirms the transaction.
6. Perform the send money transaction from the customer to the agent.
7. Respond to other validation problems if the transaction then failed.

What follows in this section is;

- an overview of the service using generalised workflow notation to visual the process, and
- how the Application Composer allows visualising of this service directly in the web user interface (after the application has been created, or during the process of creating it).



4.2 Workflow

The figure below shows the schematic workflow of the Cash Out process.

This workflow is typical of a single service, or application, that Brand Mobiliser can provide, with many validations, entries and actions to perform.

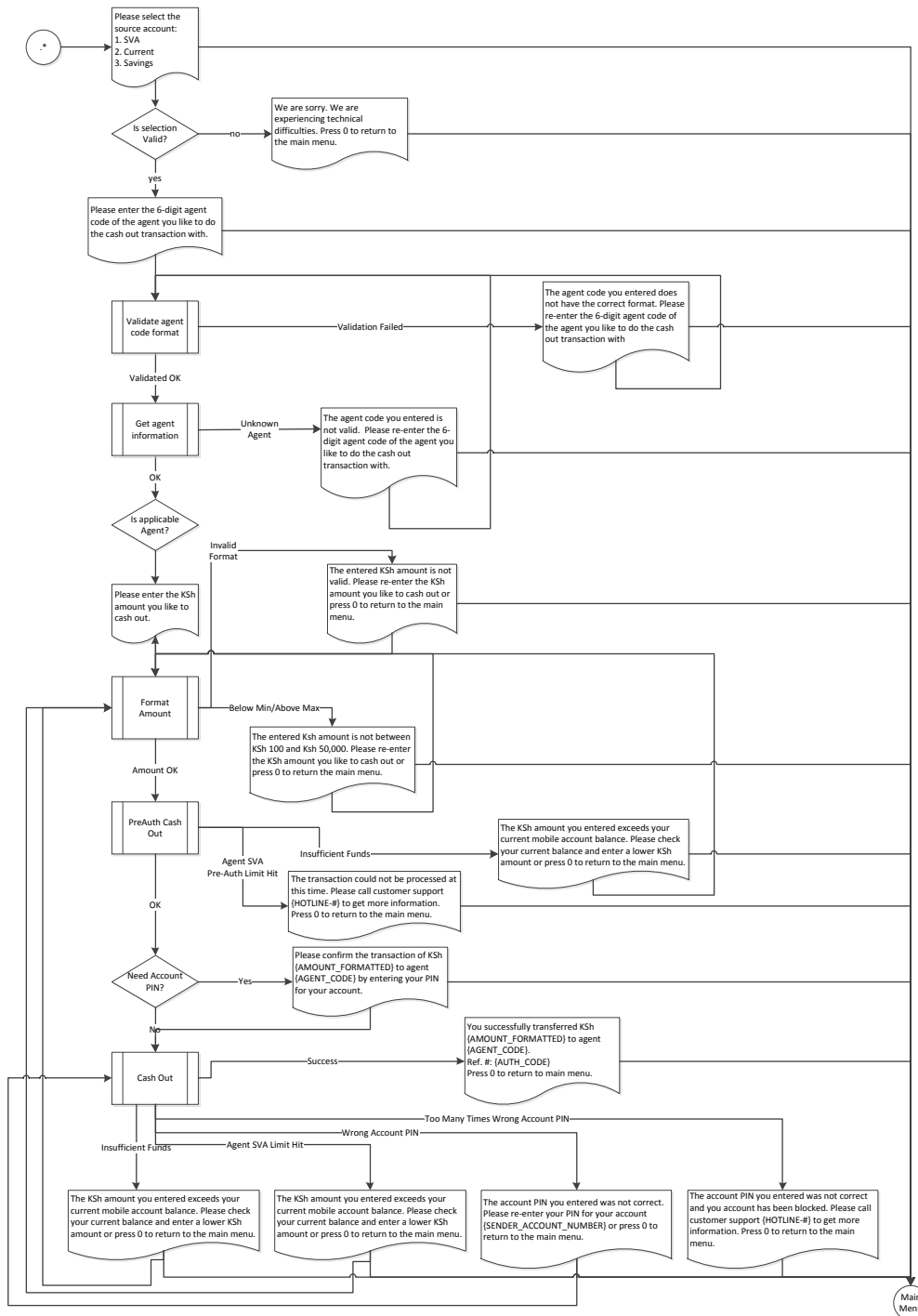


Figure 29. Workflow Schematic for Cash Out Process

4.4 State Editor

The figure below shows the first portion of the Application Composer view of the Cash Out application.

The screenshot displays the 'Cash Out Process' in the Application Composer. The main workspace shows a state transition diagram with four states: 'CASH OUT PROCESS' (Start Application), 'GET WALLET MENU' (Delete), 'ENTER AGENT CODE' (Send SMS), and 'ERROR - BACK TO MENU' (Send SMS). Arrows indicate the flow from 'CASH OUT PROCESS' to 'GET WALLET MENU', then to 'ENTER AGENT CODE', and finally to 'ERROR - BACK TO MENU'. A right-hand panel is open for the 'Get Wallet Menu' state, showing its configuration:

- Name:** Get Wallet Menu
- Message:** Please select the source account: (33/160 characters)
- Input Variables:**
 - Select an entry -- (Identification type)
 - [] Identification of the customer (overrides session).
 - 0,41,42 List of PI types. Example: 12,123,45
 - [] List of PI classes. Example: 1,2,3
 - [] Max count of PIs
- Output Variables:** (None listed)
- Follow-up States:** (Showing: 1-3 | 4-4 | All)
 - OK:** Enter agent code
 - Fail:** Error - back to menu
 - Target:** Enter agent code
 - Expression:** .* Assign To: []

Buttons at the bottom of the panel include 'Add Follow-up' and 'Save'. The interface also includes tabs for 'APPLICATION DETAILS', 'KEYWORDS', 'APPLICATION COMPOSER', and 'APPLICATION EDITOR', along with 'Application Actions' and 'Gridlines (All)' options.

Figure 31. Application Composer View of Cash Out Process

5 How to Test Applications

It is possible to test applications you have constructed in Brand Mobiliser in a number of different ways.

- Using the built-in application simulator that is part of the Brand Mobiliser Web UI.
- Using an SMPP test harness or a JMS test harness. These methods are more suited for state plugin developers or advanced system administrators, and as such are not described in this document.

5.1 Using the Application Simulator

This section describes how to test your applications using the built-in application simulator.

To access the Application Simulator page, choose the item on the drop-down list titled 'Actions' on the right hand side of the page header. Alternatively, when viewing the details of an application, choose the item on the drop-down list titled 'Application Actions'.

On entry to the application simulator you must enter a test Customer MSISDN and choose an already defined client MSISDN or short-code which identifies the workspace under which your application should be created, as shown in the figure below. (If you haven't created a client or default MSISDN refer to the Brand Mobiliser User Manual for more information.) The Customer MSISDN must be entered because Brand Mobiliser uses it to find the session. If the application being tested has states that interface with back-end systems, such as Money Mobiliser for example, it may be necessary to use an MSISDN that identifies a customer in those systems.

Simulate Incoming/Outgoing Message

Customer MSISDN:

Client MSISDN: 123456

Send SMS

Message Channel: SMS

Message Encoding: normal

Message Text:

Send to Brand Mobiliser

Send to CUSTOMER

Send with Acknowledge

Send 100 Messages at max rate to CUSTOMER

(Simulate incoming Message from customer)

(Simulate outgoing Message TO customer - You better be aware what you do)

Refresh message log

Timestamp Message Receiver Client MSISDN Dir Msg

Figure 32. Application Simulator

To test the application, you will only use button to 'Send to Brand Mobiliser' and can ignore the 'Message Channel' and 'Message Encoding' drop-downs and the 'Send to CUSTOMER' button.

The button labeled 'Refresh message log' allows all messages into and out of Brand Mobiliser *for the entered Customer MSISDN* to be displayed in the list below it. You have to manually click this button after sending a message into Brand Mobiliser to see any response message returned.

The figure below shows a sequence of interactions using a series of applications that demonstrate integration with Money Mobiliser. The message log shows messages in, using a green arrows, and messages

out, using red arrows. In this case, the customer has logged into the test system by entering their credentials and is then presented with a series of menus.

Simulate Incoming/Outgoing Message

Customer MSISDN:

Client MSISDN:

Send SMS

Message Channel:

Message Encoding:

Message Text:

Send with Acknowledge

(Simulate incoming Message from customer) (Simulate outgoing Message TO customer - You better be aware what you do)

Refresh message log

Timestamp	Message Receiver	Client MSISDN	Dir Msg
21.12.2010 11:22:46	smapp Mobiliser Menu - Top	123456	↑ Test Ident +447977257100 1. Check Identification 2. Check Credential 3 Change Credential 4. Counter 5. Copy Var 6. Get Languages 0. Back
21.12.2010 11:22:46	smapp Mobiliser Menu - Top	123456	↓ 5
21.12.2010 11:18:45	smapp Mobiliser Tester Login	123456	↑ Test Ident +447977257100 1. Get Cust. Values 2. Set Cust. Values 3. Wallet Fns 4. Money Fns 5. Misc Fns 6. Set alt test MSISDN 0. End
21.12.2010 11:18:41	smapp Mobiliser Tester Login	123456	↓ 1234
21.12.2010 11:18:36	smapp Mobiliser Tester Login	123456	↑ Mobiliser Tester Login - Enter your credentials.
21.12.2010 11:18:36	smapp Mobiliser Tester Login	123456	↓ mobiliser
21.12.2010 11:18:31	menupage Startpage	123456	↑ Startpage Welcome to Mobiliser Tester - send 'mobiliser' to continue
21.12.2010 11:18:31	menupage Startpage	123456	↓ hello

Figure 33. Application Simulator With Interaction

One of the menu options is the 'Mobiliser Counter' application as described in a previous section and is visualised in the figure below.

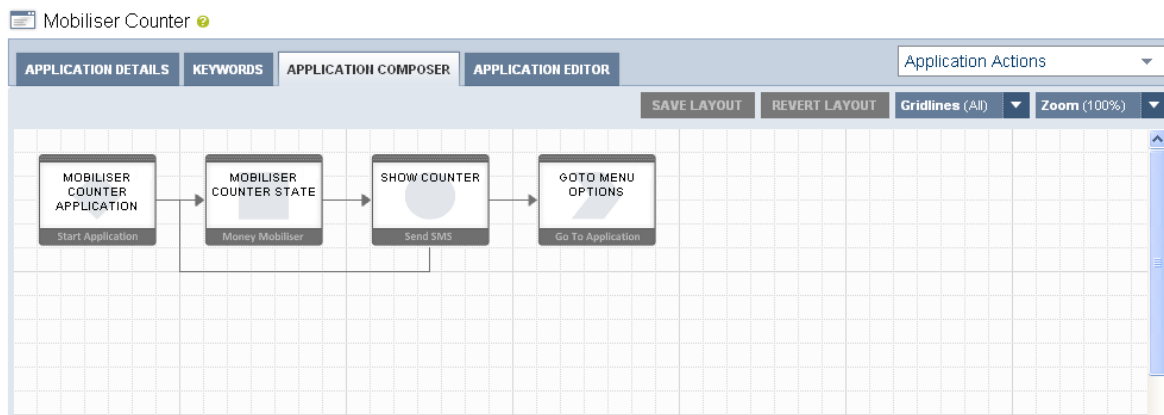


Figure 34. Mobiliser Counter Application Overview

The main logic of the counter application is to use a state to increment a session variable acting as the counter, display the value to the user then to either loop back to the increment or to exit back to the menu. The figure below outlines the first part of this application.

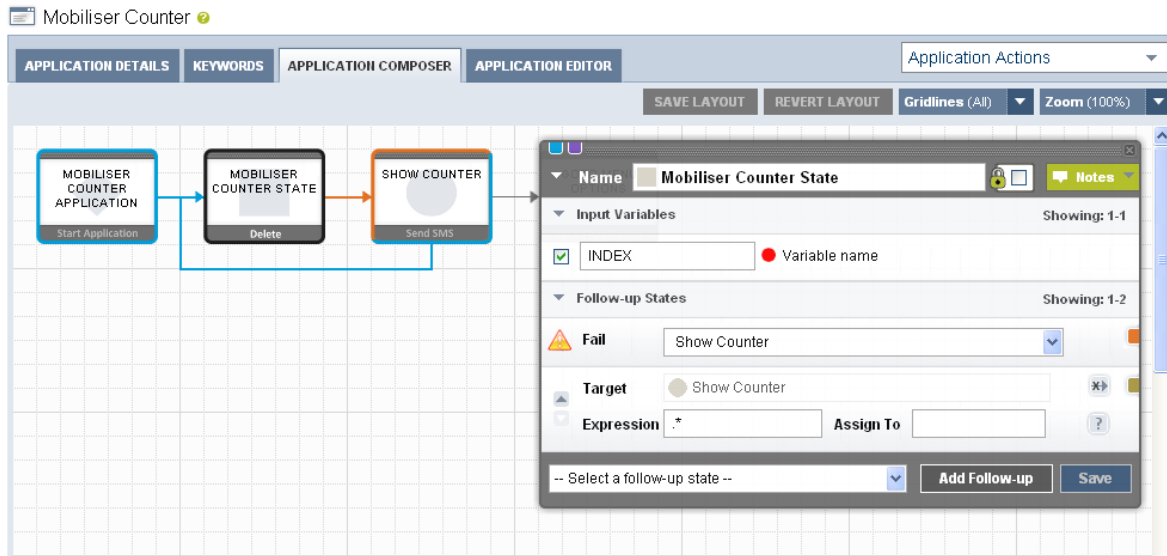


Figure 35. Mobiliser Counter State - Detail

The session variable named 'INDEX' is used as the counter variable. This variable is dynamically substituted into the text sent back to the user in the next step.

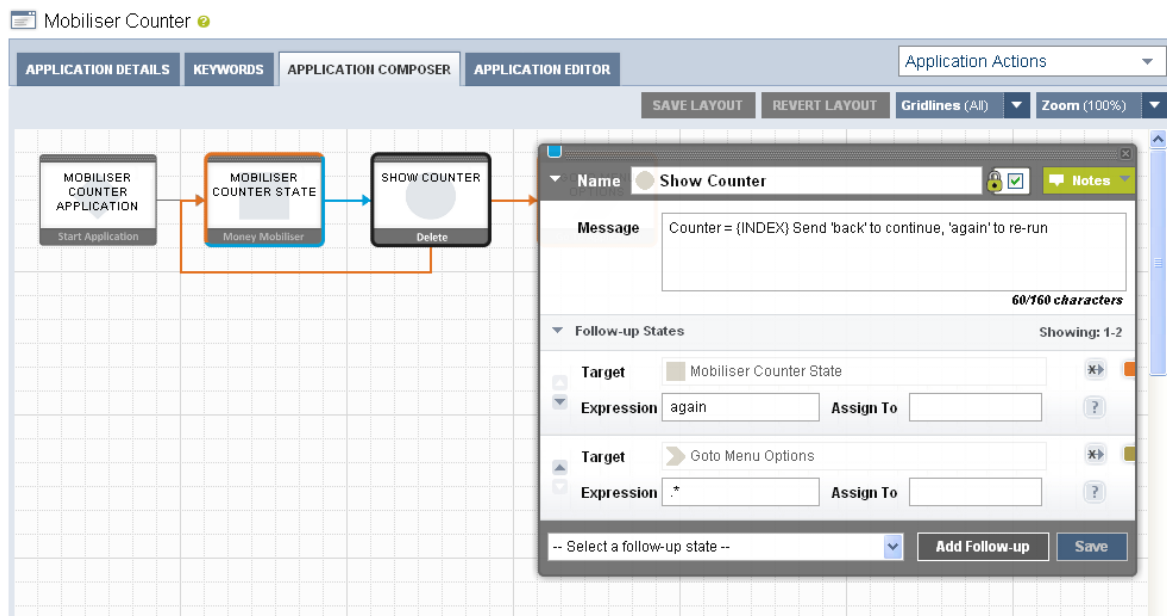


Figure 36. Show Counter State Detail

If the user responds with the keyword 'again' to the message sent from 'Show Counter', the application will loop. Any other input will cause this application to end and revert to the menu.

The interactions to get to the counter application are shown in the figure below.



On this first entry to the application, the counter value is displayed as '1' alongside the text to indicate the expected response.

Refresh message log				
Timestamp	Message Receiver	Client MSISDN	Dir Msg	
21.12.2010 11:44:21	smapp	Mobiliser Menu - Misc Functions	123456	↑ 1 Send 'back' to continue, 'again' to re-run
21.12.2010 11:44:20	smapp	Mobiliser Menu - Misc Functions	123456	↓ 4
21.12.2010 11:44:17	smapp	Mobiliser Menu - Top	123456	↑ Test Ident +447977257100 1. Check Identification 2. Check Credential 3 Change Credential 4. Counter 5. Copy Var 6. Get Languages 0. Back
21.12.2010 11:44:17	smapp	Mobiliser Menu - Top	123456	↓ 5
21.12.2010 11:44:12	smapp	Mobiliser Tester Login	123456	↑ Test Ident +447977257100 1. Get Cust. Values 2. Set Cust. Values 3. Wallet Fns 4. Money Fns 5. Misc Fns 6. Set alt test MSISDN 0. End
21.12.2010 11:44:10	smapp	Mobiliser Tester Login	123456	↓ 1234
21.12.2010 11:44:08	smapp	Mobiliser Tester Login	123456	↑ Mobiliser Tester Login - Enter your credentials.
21.12.2010 11:44:08	smapp	Mobiliser Tester Login	123456	↓ mobiliser
21.12.2010 11:44:03	menupage	Startpage	123456	↑ Startpage Welcome to Mobiliser Tester - send 'mobiliser' to continue
21.12.2010 11:44:03	menupage	Startpage	123456	↓ hello

Figure 37. Mobiliser Counter Simulation #1

To test the keyword match for the loop back to the 'Mobiliser Counter State', we enter the keyword 'again' and refresh the message log to show the following additional interactions;

Refresh message log				
Timestamp	Message Receiver	Client MSISDN	Dir Msg	
21.12.2010 11:49:22	smapp	Mobiliser Counter	123456	↑ 2 Send 'back' to continue, 'again' to re-run
21.12.2010 11:49:21	smapp	Mobiliser Counter	123456	↓ again
21.12.2010 11:44:21	smapp	Mobiliser Menu - Misc Functions	123456	↑ 1 Send 'back' to continue, 'again' to re-run
21.12.2010 11:44:20	smapp	Mobiliser Menu - Misc Functions	123456	↓ 4
21.12.2010 11:44:17	smapp	Mobiliser Menu - Top	123456	↑ Test Ident +447977257100 1. Check Identification 2. Check Credential 3 Change Credential 4. Counter 5. Copy Var 6. Get Languages 0. Back
21.12.2010 11:44:17	smapp	Mobiliser Menu - Top	123456	↓ 5
21.12.2010 11:44:12	smapp	Mobiliser Tester Login	123456	↑ Test Ident +447977257100 1. Get Cust. Values 2. Set Cust. Values 3. Wallet Fns 4. Money Fns 5. Misc Fns 6. Set alt test MSISDN 0. End
21.12.2010 11:44:10	smapp	Mobiliser Tester Login	123456	↓ 1234
21.12.2010 11:44:08	smapp	Mobiliser Tester Login	123456	↑ Mobiliser Tester Login - Enter your credentials.
21.12.2010 11:44:08	smapp	Mobiliser Tester Login	123456	↓ mobiliser
21.12.2010 11:44:03	menupage	Startpage	123456	↑ Startpage Welcome to Mobiliser Tester - send 'mobiliser' to continue
21.12.2010 11:44:03	menupage	Startpage	123456	↓ hello

Figure 38. Mobiliser Counter Simulation #2

In this case, the keyword match was confirmed as successful and the counter is incremented and re-displayed to customer. From this point any other input will return to the menu, from which the customer can continue with other functions or choose to end their session. The figure below shows the remainder of this simulation session; returning the menu and then ending the session.

Refresh message log			
Timestamp	Message Receiver	Client MSISDN	Dir Msg
21.12.2010 11:52:50	smapp Mobiliser Menu - Top	123456	↑ TTFN!
21.12.2010 11:52:50	smapp Mobiliser Menu - Top	123456	↓ 0
21.12.2010 11:52:42	smapp Mobiliser Menu - Misc Functions	123456	↑ Test Ident +447977257100 1. Get Cust. Values 2. Set Cust. Values 3. Wallet Fns 4. Money Fns 5. Misc Fns 6. Set alt test MSISDN 0. End
21.12.2010 11:52:42	smapp Mobiliser Menu - Misc Functions	123456	↓ 0
21.12.2010 11:52:35	smapp Mobiliser Counter	123456	↑ Test Ident +447977257100 1. Check Identification 2. Check Credential 3 Change Credential 4. Counter 5. Copy Var 6. Get Languages 0. Back
21.12.2010 11:52:35	smapp Mobiliser Counter	123456	↓ back
21.12.2010 11:49:22	smapp Mobiliser Counter	123456	↑ 2 Send 'back' to continue, 'again' to re-run
21.12.2010 11:49:21	smapp Mobiliser Counter	123456	↓ again
21.12.2010 11:44:21	smapp Mobiliser Menu - Misc Functions	123456	↑ 1 Send 'back' to continue, 'again' to re-run
21.12.2010 11:44:20	smapp Mobiliser Menu - Misc Functions	123456	↓ 4
21.12.2010 11:44:17	smapp Mobiliser Menu - Top	123456	↑ Test Ident +447977257100 1. Check Identification 2. Check Credential 3 Change Credential 4. Counter 5. Copy Var 6. Get Languages 0. Back
21.12.2010 11:44:17	smapp Mobiliser Menu - Top	123456	↓ 5
21.12.2010 11:44:12	smapp Mobiliser Tester Login	123456	↑ Test Ident +447977257100 1. Get Cust. Values 2. Set Cust. Values 3. Wallet Fns 4. Money Fns 5. Misc Fns 6. Set alt test MSISDN 0. End
21.12.2010 11:44:10	smapp Mobiliser Tester Login	123456	↓ 1234
21.12.2010 11:44:08	smapp Mobiliser Tester Login	123456	↑ Mobiliser Tester Login - Enter your credentials.
21.12.2010 11:44:08	smapp Mobiliser Tester Login	123456	↓ mobiliser
21.12.2010 11:44:03	menupage Startpage	123456	↑ Startpage Welcome to Mobiliser Tester - send 'mobiliser' to continue
21.12.2010 11:44:03	menupage Startpage	123456	↓ hello

Figure 39. Mobiliser Counter Simulation #3

Note: This sample simulation shows the testing of an application through a menu system, which is itself an application. It is possible to invoke an application logic directly, depending on the keywords specified for an application.



6 Importing/Exporting Applications

Brand Mobiliser provides the ability to create an exported copy of an application's configuration. This is useful in order to;

- Move an application between instances of Brand Mobiliser installations.
- Make a back-up copy of an individual application.

6.1 Export

To export an application, use the 'Export' button on the Application Details page.

The screenshot shows the 'Cash Out Process' application details page. The page has a header with 'Cash Out Process' and a dropdown menu for 'Application Actions'. Below the header are tabs for 'APPLICATION DETAILS', 'KEYWORDS', 'APPLICATION COMPOSER', and 'APPLICATION EDITOR'. The main content area is titled 'General Application Information' and contains the following fields:

- Name*: Cash Out Process
- Category: test
- Active From*: 12/16/10
- Active To*: 12/17/12

Below this is a section for 'Session Settings (Optional)' which includes:

- Timeout (secs): 450
- Session Limit Response: (empty text area)

At the bottom of the page are two buttons: 'Save' and 'Export'.

Figure 40. Export Application

This produces an XML file that represents the configuration of the application. A sample of the exported file is shown below;

appFlow.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root xmlns="http://ipsquare.at/xsd/mwiz">
  <smapp serviceMenuActionTypesId="3100" smappApplicationTypesId="102825"
  activeTo="2012-12-17T00:00:00.000Z" activeFrom="2010-12-16T00:00:00.000Z"
  initialStatesId="1268" shortName="1292521229969_Cash Out Process_1_1" name="Cash Out
  Process" versionId="1">
    <states>
      <state id="1595" smappStateTypeName="Send SMS" encrypted="false">
```

```

mtChargingItemsFkId="0" moChargingItemsFkId="0" chargingItemsFkId="0"
smappAuctionTypesId="0" subscribeAFriend="false" forwardMessageReceiversId="0"
sendIntermediateResult="false" last="0" mmsSmilMessagesId="0" smappStateTypesId="1901"
name="Wrong account PIN" versionId="1">
    <smappStateLang smsText="The account PIN you entered was not correct.
Please re-enter your PIN for your account {SENDER_ACCOUNT_NUMBER} or press 0 to return to
main menu." versionId="1">
        <language unicode="false" default2="true" description="englisch"
name="EN"/>
    </smappStateLang>
    <smappStateAttribute key="gridposx" versionId="1">
        <intValue>13</intValue>
    </smappStateAttribute>
    <smappStateAttribute key="gridposy" versionId="1">
        <intValue>1</intValue>
    </smappStateAttribute>
</state>
...

```

Note: This file is not expected to be manually edited or created.

6.2 Import

To import an application, using the 'Upload' button on the New application/Add an asset page.

Note: The import process can produce duplicate named applications.

▼ New Application

Create New Application:	New
OR	
Upload from External File	
Application Name:	<input type="text"/>
	<input type="button" value="Choose File"/> No file chosen <input type="button" value="Upload"/>
	> Import Success. Application name is 'Cash Out Process'. <input type="button" value="View Application Details"/>

Figure 41. Import Application

Note: Importing applications that have states that link to other applications require the applications linked to be created prior to the import in order for the import process to resolve the links. For example, a 'Go To Application' state will require that the application being linked to exists prior to the import.

If there are circular references, which are common in menu-based systems, the application being imported will still be imported, but a manual process of re-linking applications is required before the application can be used.



A. Appendix – Base States

This appendix section describes the base states provided in Brand Mobiliser. The base states are expected to be the basis of any interactive application.

A.1 Goto Application

The user can move to a new application from this state by selecting any one of the applications displayed in the drop down box. In the example shown in the figure below, the “Mobiliser Menu – Misc Functions” application has been selected in the “Goto Application”.

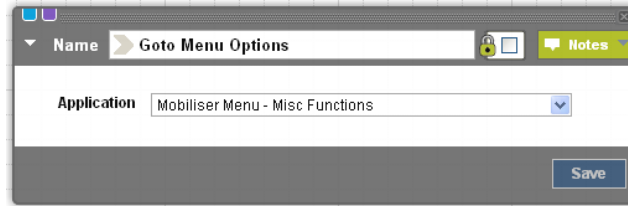


Figure 42. Example Goto Application State

Inputs

Application - Brand Mobiliser Application to be called.

Note: you can only go to the application within the same workspace.

A.2 Send SMS

This state is used whenever text, e.g. an instruction is to be sent to the customer via SMS.

To embed the value of a session variable into the text enter the name of the variable surrounded by { and }. For example, the text “The value of the counter is {INDEX}”, would replace the text {INDEX} with the value of the session variable ‘INDEX’. If no variable with that name exists, the text is sent as-is, including the text ‘{INDEX}’.

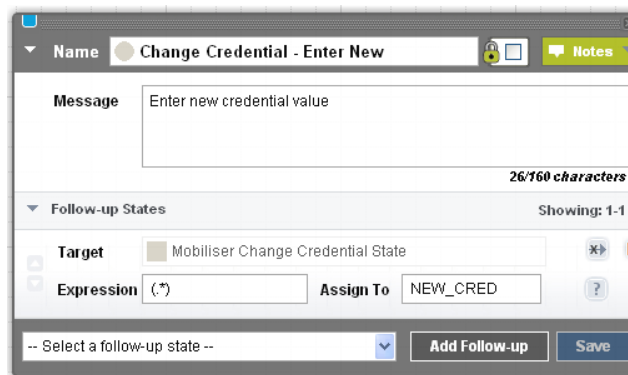


Figure 43. Send SMS State

Inputs

SMS Text - The Text to be sent via SMS.

If the length of the text exceeds 160 characters, the text is truncated and multiple messages are sent.

Note:

- It may not be possible to determine the exact number of characters in the message prior to run-time, if session variable values are embedded into the message.
- During runtime, the Send SMS state results in a temporary suspension of the application flow to wait for the user's response. By default, the wait (also known as session timeout) lasts for 7.5 minutes (450 seconds). Once the session timeout, the user's response to continue the application flow will not be honored. Depending on the setup, the user may be presented with a guidance message or main menu. The session timeout can be adjusted for each application using the Application Details screen.

A.3 Set Variable

This state is used to set a variable with a specified value. With this state, it is only possible to define a variable with a given string, integer, etc. It is not possible to copy a variable.

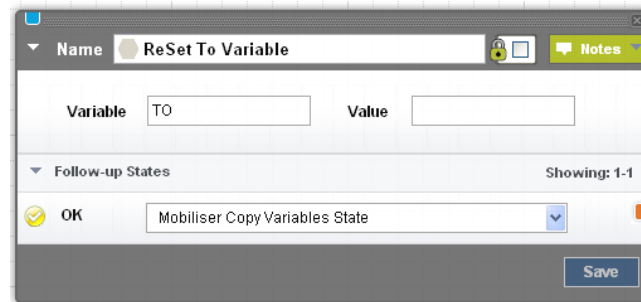


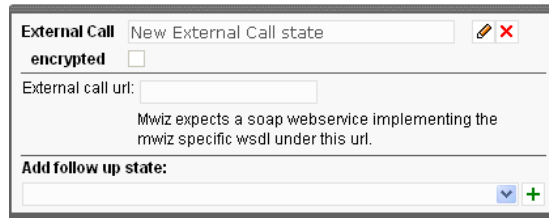
Figure 44. Set Variable State

Inputs

Variable - The name of the variable must be entered into the "Variable" field on the left side of the box and the value into the field on the right.

A.4 External Call

This state is used to forward messages to an external system via a SOAP Request.



The screenshot shows a configuration window titled "External Call" with a subtitle "New External Call state". It includes an "encrypted" checkbox, an "External call url:" text box, a note stating "Mwiz expects a soap webservice implementing the mwiz specific wsdl under this url.", and an "Add follow up state:" section with a dropdown menu and a plus sign button.

Figure 45. External Call State

Inputs

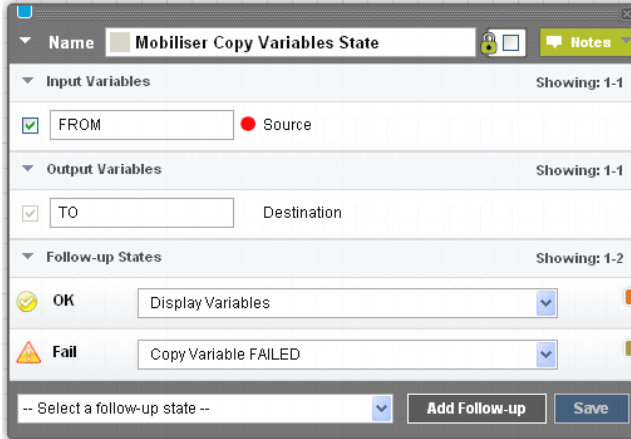
External call url - The URL to which the SOAP Request (processMessage) should be sent.

B. Appendix - Mobiliser Utility States

This appendix section describes those states provided in the Mobiliser Core states plugin that are general purpose based. These base states are expected to be used for utility functions in a majority of interactive applications.

B.1 Mobiliser Copy Variables

This state copies the value of a source variable to a destination variable.



The screenshot shows a configuration window for the 'Mobiliser Copy Variables State'. The window has a title bar with a lock icon and a 'Notes' button. Below the title bar, there is a 'Name' field containing 'Mobiliser Copy Variables State'. The main area is divided into three sections: 'Input Variables' (Showing: 1-1) with a checked checkbox and a text field containing 'FROM' and a red dot labeled 'Source'; 'Output Variables' (Showing: 1-1) with a checked checkbox and a text field containing 'TO' and the label 'Destination'; and 'Follow-up States' (Showing: 1-2) with two entries: 'OK' with a checkmark icon and a dropdown menu set to 'Display Variables', and 'Fail' with a warning triangle icon and a dropdown menu set to 'Copy Variable FAILED'. At the bottom, there is a dropdown menu with the text '-- Select a follow-up state --', an 'Add Follow-up' button, and a 'Save' button.

Figure 46. Mobiliser Copy Variables State

Inputs

Source - The name of the source variable whose value was copied.

Output

Destination - The name of the destination variable to which the value is copied.

B.2 Mobiliser Counter

This state creates a variable that is increased by one every time the state is called.

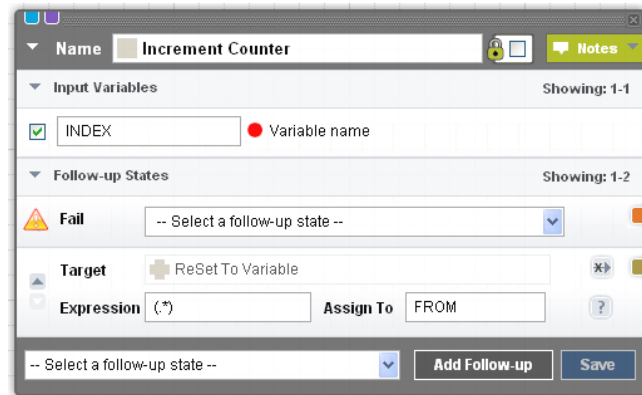


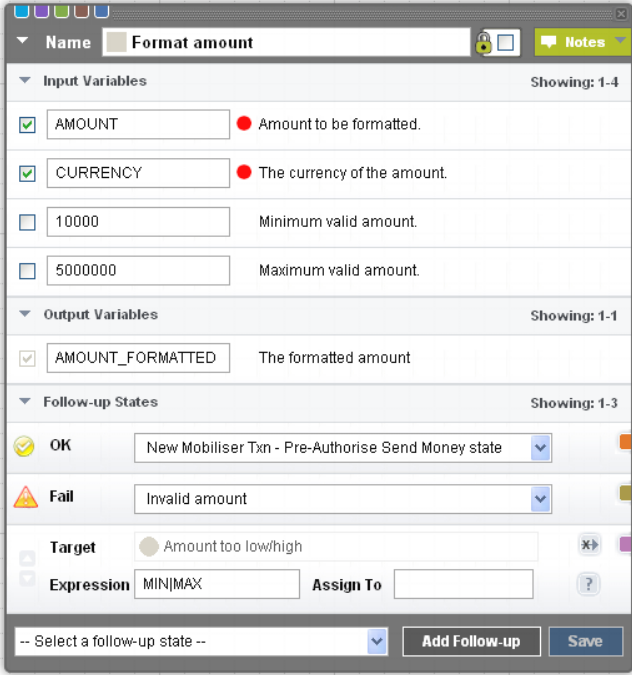
Figure 47. Mobiliser Counter State

Inputs

Variable name - The name of the variable that is increased.

B.3 Mobiliser Format Amount

This state formats the entered amount of a given currency. It checks the validity of the entered amount and returns it in formatted form. The amount is reformatted to comply with the number of decimal places specified for the respective currency and rounded off if necessary.



The screenshot shows a configuration window for a state named "Format amount". It is divided into several sections:

- Input Variables (Showing: 1-4):**
 - AMOUNT: Amount to be formatted.
 - CURRENCY: The currency of the amount.
 - 10000: Minimum valid amount.
 - 5000000: Maximum valid amount.
- Output Variables (Showing: 1-1):**
 - AMOUNT_FORMATTED: The formatted amount.
- Follow-up States (Showing: 1-3):**
 - OK:** New Mobiliser Txn - Pre-Authorise Send Money state
 - Fail:** Invalid amount
 - Target:** Amount too low/high
 - Expression:** MIN|MAX
 - Assign To:** (empty field)

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 48. Mobiliser Format Amount State

Inputs

Amount to be formatted - The value of the amount to be formatted.

The currency of the amount - The currency of the amount to be formatted.

Minimum valid amount - The minimum valid value of the amount.

Maximum valid amount - The maximum valid value of the amount.

Outputs

The formatted amount - The reformatted amount is returned here.



B.4 Mobiliser Get Languages Menu

This state loads the languages available in the current Brand Mobiliser installation and wraps it into a menu.

The following follow-up states are defined:

- FAIL: If an error occurs.
- OK: If the user selected a menu item.

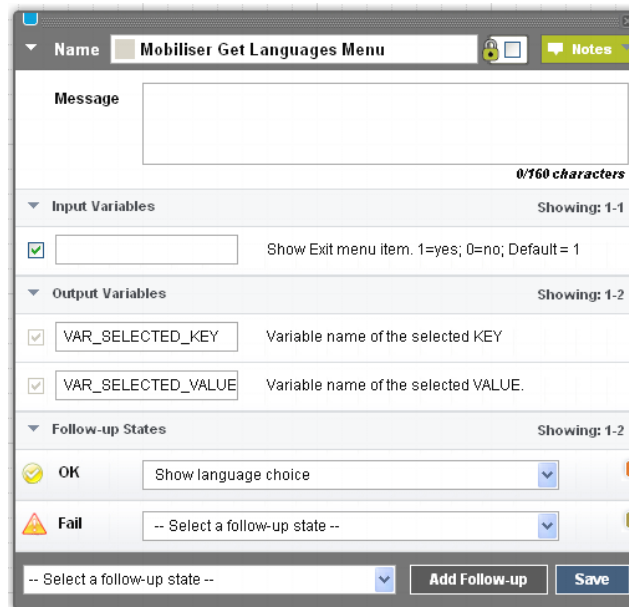


Figure 49. Mobiliser Get Languages Menu State

Inputs

Show Exit menu item - This defines whether the 'Exit' menu item (text '0. Exit') should be displayed. 0=no, 1=yes. If not set, the default is to display the 'Exit' menu item.

Outputs

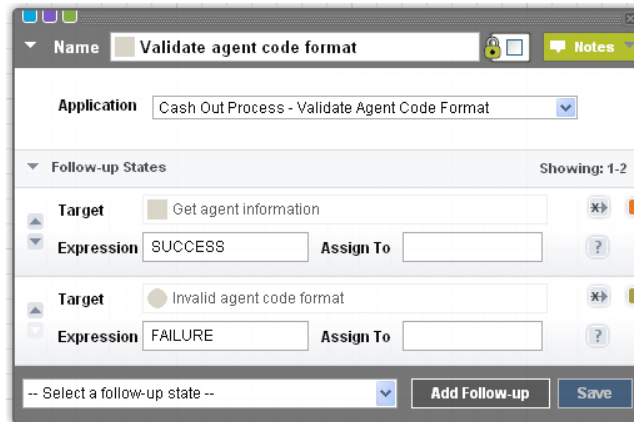
Variable of the selected KEY - This is the number assigned to the menu item.

Variable of the selected VALUE - This is the text of the menu.

B.5 Mobiliser Method Call

Like the “Goto Application”, this state also accesses a new application. However, in this case, the user does not completely leave the current state-flow. The “Mobiliser Method Call” enables you to access another application and use its return values.

For example, consider the case when the user must enter a 6-digit code that identifies an agent. This code must be validated. Because this check is carried out frequently in various applications, the validation procedure may be written as a separate application. It returns SUCCESS if the code is successfully validated, or FAILURE if validation is unsuccessful. Using multiple follow up states we can match to the returned value for the appropriate next state.



The screenshot shows a configuration window for a state named "Validate agent code format". The "Application" dropdown is set to "Cash Out Process - Validate Agent Code Format". Under "Follow-up States", there are two entries: one with "Target" "Get agent information" and "Expression" "SUCCESS", and another with "Target" "Invalid agent code format" and "Expression" "FAILURE". Each entry has an "Assign To" field. At the bottom, there is a dropdown menu for selecting a follow-up state, and "Add Follow-up" and "Save" buttons.

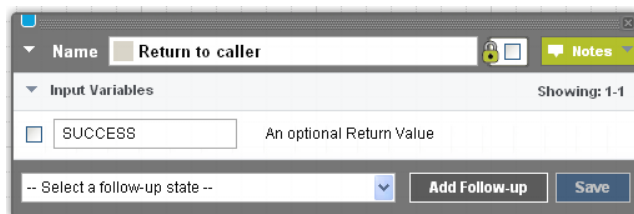
Figure 50. Mobiliser Method Call State

Inputs

Application - Brand Mobiliser Application to be called.

B.6 Mobiliser Method Call Return

If you define an application that is to be called by another application, the application can return a value. This state assures that a value is returned.



The screenshot shows a configuration window for a state named "Return to caller". Under "Input Variables", there is one entry with a checkbox, the text "SUCCESS", and a description "An optional Return Value". At the bottom, there is a dropdown menu for selecting a follow-up state, and "Add Follow-up" and "Save" buttons.

Figure 51. Mobiliser Method Call Return State

Inputs

An optional Return Value - The value that is returned to the caller.

B.7 Mobiliser Regex

This state matches a given variable against a regular expression.

Figure 52. Mobiliser Regex State

Inputs

Variable to match - The variable that should be checked against a regular expression

Regular Expression - The regular expression used to check the variable.

B.8 Mobiliser Set Language (Session Only)

This state sets the language for the current Brand Mobiliser session. The language must exist in Brand Mobiliser database.

The following follow-up states are defined:

- FAIL: If an error occurs.
- OK: If everything went well.

Figure 53. Mobiliser Set Language (Session only) State

Inputs

The selected Language - The language to be set for the session as a locale (e.g. en, de, etc.).

C. Appendix - Mobiliser Web Service States

This appendix section describes those states provided in the Mobiliser Core states plugin that interface with an external Money Mobiliser installation. These states generally require a validated customer session in the external Money Mobiliser system (as provided by the Mobiliser Login state). These states are expected to be used for any interactive applications that require interaction with Money Mobiliser.

C.1 Mobiliser Login

This state performs a login. If the identification input is not set, the current customer will be used.

The following follow-up states are defined:

- FAIL: Not in use.
- OK: If everything went well.
- Dyn: [Error CODE]



Figure 54. Mobiliser Login State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

PIN - The PIN of the customer.

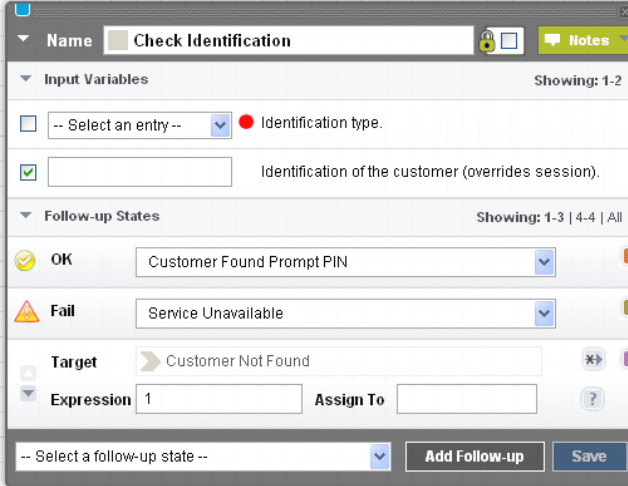
C.2 Mobiliser Check Identification

This state can be used to check the status of a customer. The status can be one of the following:

- UNKNOWN – returns value '1'
- BLACKLISTED - returns value '2'

If the state does not return 1 or 2, it can be assumed the customer is registered in the system and can make further interactions.

This state also sets the information: NAME, PASSPORT_ID, the database index for the LANGUAGE and the two- letter language code LANGUAGE_A2 (e.g. EN).



The screenshot shows a configuration window for a state named "Check Identification". The window has a title bar with a close button and a "Notes" icon. Below the title bar, there is a section for "Input Variables" with a "Showing: 1-2" indicator. It contains two entries: one with a dropdown menu labeled "-- Select an entry --" and a red dot icon, and another with a checked checkbox and a text input field. Below this is a section for "Follow-up States" with a "Showing: 1-3 | 4-4 | All" indicator. It lists three states: "OK" with a green checkmark icon and a dropdown menu set to "Customer Found Prompt PIN"; "Fail" with a yellow warning triangle icon and a dropdown menu set to "Service Unavailable"; and "Target" with a right-pointing arrow icon and a dropdown menu set to "Customer Not Found". Below the "Target" state, there is an "Expression" field containing the number "1" and an "Assign To" field. At the bottom of the window, there is a dropdown menu labeled "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 55. Mobiliser Check Identification State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

C.3 Mobiliser Check Credential

This state checks a customer's credential.

The customer is identified using the specific identification type (the current session is used if not set).

The customer's credential may be a PIN or a password and its value is mandatory.

Use the following follow-up states:

- OK State: If credential check was OK.
- Fail State: If an error occurs

The screenshot displays the configuration for the 'Check MPIN Credential' state. It includes sections for 'Input Variables' and 'Follow-up States'. Under 'Input Variables', there are four items: 'By mobile-no' (unchecked), 'Identification of the customer (overrides session)' (checked), 'Credential is a PIN' (unchecked), and 'PIN' (checked). Under 'Follow-up States', there are two items: 'OK' (Send Money state) and 'Fail' (ERROR Check MPIN). The 'Target' field is set to 'Wrong MPIN Try again' and the 'Expression' field is set to '1113'. The interface also has 'Add Follow-up' and 'Save' buttons.

Figure 56. Mobiliser Check Credential State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Credential Type – Indicator of the type of credential that is being provided.

The customer's credential value – The value of the customer's credential.

C.4 Mobiliser Change Credential

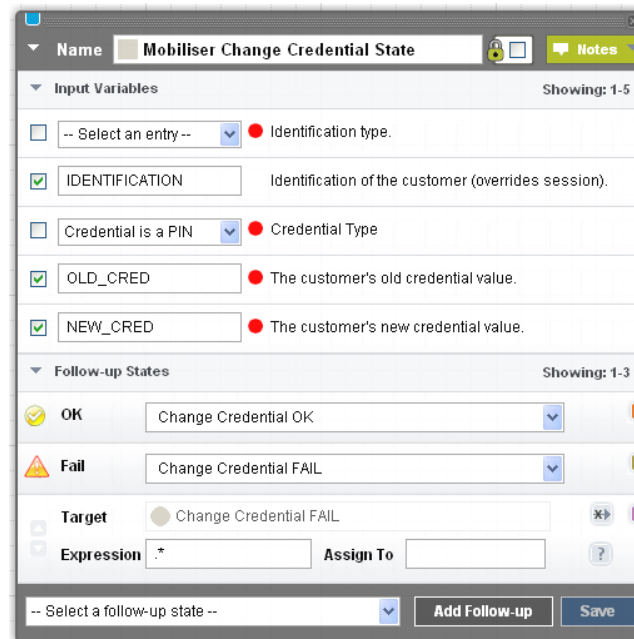
This state is used to change the credential value for a customer.

The customer is identified using the specific identification type (the current session is user is used if not set).

The customer's credential may be a PIN or a password and the old and new values are mandatory.

Use the following follow-up states:

- OK State: If credential check was OK.
- Fail State: If an error occurs
- Dyn: [Mobiliser Error Code]



The screenshot shows a configuration window for the 'Mobiliser Change Credential State'. The window has a title bar with 'Name' set to 'Mobiliser Change Credential State' and a 'Notes' button. Below the title bar, there are two main sections: 'Input Variables' and 'Follow-up States'. The 'Input Variables' section shows five variables: 'Identification type' (unchecked), 'IDENTIFICATION' (checked), 'Credential Type' (unchecked), 'OLD_CRED' (checked), and 'NEW_CRED' (checked). Each variable has a description and a red dot indicating it is required. The 'Follow-up States' section shows three states: 'OK' (checked), 'Fail' (unchecked), and 'Target' (unchecked). Each state has a name and a dropdown menu. At the bottom, there is an 'Expression' field with the value '.*' and an 'Assign To' field. There are also 'Add Follow-up' and 'Save' buttons.

Figure 57. Mobiliser Change Credential State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Credential Type – Indicator of the type of credential that is being provided.

The customer's old credential value – The value of the customer's old credential.

The customer's new credential value – The value of the customer's new credential.

C.5 Mobiliser Get Customer Information

Loads a set customer and returns selected user information.

If the identification values are not set, the current customer MSISDN is used.

Figure 58. Mobiliser Get Customer information State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

MAX length of display name - This parameter limits the length of the name that is displayed.

Outputs

Customer ID - The customer identification.

Blacklist Reason (ID) - If the customer is blacklisted, a blacklist reason is set and can be stored in a value.

Language - The customer's preferred language.

Country - The country the customer comes from.

Display Name - The customer name to be displayed



Orgunit - The OrgUnit defines an organisational unit. This is a grouping of customers. For example, a grouping of regions that includes the region the customer comes from.

Security Answer - The correct answer to the security question.

Security Question - The security question.

Is Active (true/false) - Indicates whether the customer is active or not.

Is Test (true/false) - Defines whether the customer is a test user or a real user.

Is blocked until (milliseconds) - If the customer is blocked, this method returns the remaining time for which the customer is blocked in milliseconds.

MSISDN - The customer's telephone number.

Product (ID) - The product defines the customer type. This ID is used to differentiate between a Street Agent, Mini Agent, Super Agent and a normal customer.

User Right (ID) - The User Right ID can be ignored here because in this context it is the same ID as the Product ID

Consumer Blacklist Reason (ID) - If the consumer is blacklisted, the ID of the reason for the blacklisting is stored in the database.

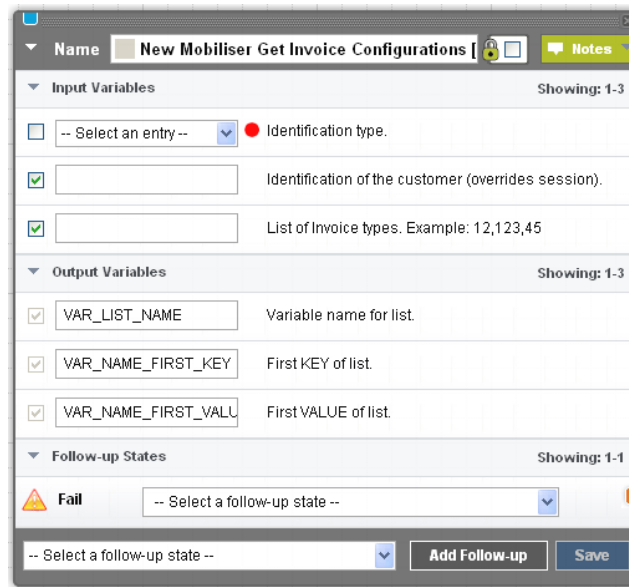
Consumer Receipt Mode (ID) - This mode defines whether the receipt should be sent by Email, SMS, both or not at all.

Consumer Cancellation Reason (ID) - The reason why a customer was cancelled. In this state, this can be ignored.

Is Consumer Active (true/false) - Indicates whether the customer is active or not. Usually, an inactive customer cannot use the Cash Out menu.

C.6 Mobiliser Get Invoice Configurations [List]

This state returns a list of all invoices that are bound to an invoice configuration of a given invoice type. With an invoice configuration, you can describe for example the period of the invoice.



The screenshot shows a configuration window titled "New Mobiliser Get Invoice Configurations [List]". It is divided into three main sections: "Input Variables", "Output Variables", and "Follow-up States".

- Input Variables (Showing: 1-3):**
 - Row 1: A dropdown menu with "-- Select an entry --" and a red dot icon. Description: "Identification type."
 - Row 2: A checked checkbox and an empty text input field. Description: "Identification of the customer (overrides session)."
 - Row 3: A checked checkbox and an empty text input field. Description: "List of Invoice types. Example: 12,123,45"
- Output Variables (Showing: 1-3):**
 - Row 1: A checked checkbox and a text input field containing "VAR_LIST_NAME". Description: "Variable name for list."
 - Row 2: A checked checkbox and a text input field containing "VAR_NAME_FIRST_KEY". Description: "First KEY of list."
 - Row 3: A checked checkbox and a text input field containing "VAR_NAME_FIRST_VALU". Description: "First VALUE of list."
- Follow-up States (Showing: 1-1):**
 - Row 1: A yellow warning triangle icon, the text "Fail", and a dropdown menu with "-- Select a follow-up state --".

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 59. Mobiliser Get Invoice Configurations [List] State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

List of Invoice types - A list of the invoice types you want to retrieve.

Outputs

Variable name - The name of the variable the retrieved list should be stored to.

First KEY of list - The first key of the list.

First VALUE of list - The first value of the list.



C.7 Mobiliser Get Invoices [List]

This state retrieves a list of all invoices with the given status.

The screenshot shows a configuration window for a state named "New Mobiliser Get Invoices [List] state". It features a "Notes" button and a "Name" field. The configuration is organized into three sections: "Input Variables" (Showing: 1-4), "Output Variables" (Showing: 1-3), and "Follow-up States" (Showing: 1-1). Under "Input Variables", there are four entries: a dropdown menu for "Identification type", a checkbox for "Identification of the customer (overrides session)", a checkbox for "Status Filter (one number only). Example: 42", and a checkbox for "ID_INVOICE" with a red dot indicating a required field. Under "Output Variables", there are three entries: a checkbox for "VAR_LIST_NAME" (Variable name for list), a checkbox for "VAR_NAME_FIRST_KEY" (First KEY of list), and a checkbox for "VAR_NAME_FIRST_VALU" (First VALUE of list). Under "Follow-up States", there is a "Fail" state with a dropdown menu for "-- Select a follow-up state --". At the bottom, there are two more dropdown menus for "-- Select a follow-up state --" and two buttons: "Add Follow-up" and "Save".

Figure 60. Mobiliser Get Invoices [List] State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Status Filter - The status of the invoices to be searched for.

Variable name for the ID of the invoice - The ID of the invoice.

Outputs

Variable name for list - The name of the variable to which the retrieved invoices should be stored.

First KEY of list - The first key of the retrieved list.

First VALUE of list - The first value of the retrieved list.

C.8 Mobiliser Get PI Balance

This state loads the current balance and currency of a payment instrument. The following follow up states are in use:

- FAIL: If an error occurs.
- OK: If the balance was loaded.
- Dyn: [Mobiliser Error Code]

The screenshot shows a configuration window for the state 'Mobiliser Get PI Balance State'. It is divided into several sections:

- Name:** Mobiliser Get PI Balance State
- Input Variables (Showing: 1-2):**
 - PI_ID: Name of the PI ID.
 - [Empty]: Optional PIN for Bank Accounts or other PIs.
- Output Variables (Showing: 1-2):**
 - BALANCE: Balance of the PI.
 - CURRENCY: Currency of PI.
- Follow-up States (Showing: 1-3):**
 - OK: Mobiliser Get PI Balance - show balance
 - Fail: Mobiliser Get PI Balance - get balance FAILED
 - Target:** New Send SMS state
 - Expression:** (*) **Assign To:** MOB_ERROR

At the bottom, there is a dropdown menu with the text '-- Select a follow-up state --', an 'Add Follow-up' button, and a 'Save' button.

Figure 61. Mobiliser Get PI Balance State

Inputs

Name of the PI ID - The name of the payment instrument ID.

Optional PIN for Bank Accounts or other PIs - Optional PIN for Bank Accounts or other PIs.

Outputs

Balance of the PI - The balance of the PI.

Currency of PI - The currency of the PI.

C.9 Mobiliser Get Transaction Details

This state retrieves all details of a transaction given to the state via the transaction ID.

The screenshot shows a configuration window titled "Mobiliser Get Transaction Details State". It is divided into several sections:

- Input Variables (Showing: 1-3):**
 - A dropdown menu with "-- Select an entry --" and a red dot icon, labeled "Identification type".
 - A checked checkbox next to "IDENTIFICATION" with the description "Identification of the customer (overrides session)".
 - A checked checkbox next to "TXN_ID" with the description "TransactionId".
- Output Variables (Showing: 1-5 | 6-10 | 11-13 | All):**
 - Checked checkboxes next to "TXN_AMOUNT" (Amount), "TXN_CURRENCY" (Currency), "TXN_AUTHCODE" (Authcode), "ERROR_CODE" (Error Code), and "STATUS_CODE" (Status Code).
- Follow-up States (Showing: 1-2):**
 - A green checkmark icon next to "OK" with a dropdown menu set to "Get Transaction Detail - OK".
 - A yellow warning triangle icon next to "Fail" with a dropdown menu set to "Get Transaction Detail - FAILED".

At the bottom, there is a dropdown menu "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 62. Mobiliser Get Transaction Details State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

TransactionId - The ID of the transaction.

Outputs

Amount - The amount of the transaction.

Currency - The currency of the transaction.

Authcode - The authorisation code of the transaction.

Error Code - The error code of the transaction.

Status Code - The status code of the transaction.

Type - The type of the transaction.

Payee Name - The name of the payee of the transaction.

Payer Name - The name of the payer of the transaction.

Payee ID - The ID of the payee of the transaction.

Payer ID - The ID of the payer of the transaction.

Text - The text of the transaction. (E.g. reference)

Payer Fee [Amount] - The fee that is defined for the payer of the transaction.

Payee Fee [Amount] - The fee that is defined for the payee of the transaction.

C.10 Mobiliser Get Transactions

This state retrieves all transactions that belong to a given Use Case specified by the use case id. To limit the results, the number of transactions can be specified.

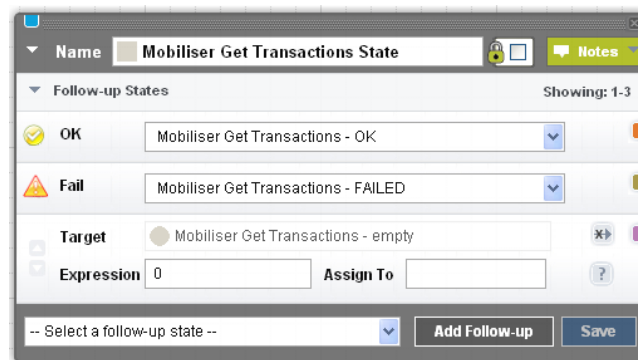


Figure 63. Mobiliser Get Transactions State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Number of Transactions - Number of the retrieved transactions.

Use Case ID of fee to be applied (opt.) - Filter for the transaction type.

C.11 Mobiliser Get Wallet Menu

This state displays a menu to the user showing all the filtered payment instruments. The SMS text defines the title of the menu.

The screenshot shows a configuration window for the state 'Mobiliser Get Wallet Account Menu'. It includes a 'Message' field (0/760 characters), 'Input Variables' (Identification type, IDENTIFICATION, List of PI types, List of PI classes, Max count of PIs), 'Output Variables' (VAR_CURRENCY, PI_TYPE, PI_CLASS, PI_STATUS, VAR_SELECTED_KEY), and 'Follow-up States' (OK, Fail, Target, Expression). The 'Follow-up States' section is expanded to show 'OK' with the message 'Wallet Menu - selected item', 'Fail' with 'Wallet Menu - failed', and 'Target' with 'Wallet Menu - no entries'. The 'Expression' field is set to '0' and 'Assign To' is empty. At the bottom, there is a dropdown for 'Follow-up state' and buttons for 'Add Follow-up' and 'Save'.

Figure 64. Mobiliser Get Wallet Menu State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

List of PI types - The PI types to be filtered for.

List of PI classes - The PI classes to be filtered for.

Max count of PIs - The maximum number of retrieved PIs

Show Exit menu item - This defines whether the Exit menu item (0. Exit) should be displayed. 0=no, 1=yes.

Outputs

Currency of the selected PI - The currency of the selected PI.

PI Type - The type of the selected PI.

PI Class - The class of the selected PI.

PI Status - The status of the selected PI.

Variable of the selected KEY - This parameter returns the identification (ID) of the PI, that the user selected.

Variable of the selected VALUE - This parameter returns the name of the PI that the user selected.

C.12 Mobiliser Block Until

This state is used to block a specific customer. Optionally, you can define a MSISDN that should be blocked. If no MSISDN is specified, the current customer using the menu of this state gets blocked. You can also define the time for which the customer should be blocked. If you wish to inform the customer of the block, you can provide this information using the “Contact Notification” textbox.

Figure 65. Mobiliser Block Until State

Inputs

Identification type - The type of customer identification to use; by ‘mobile-no’ (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Time to block customer in minutes - The term for which the customer should be blocked in minutes.

Contact Notification - The text that is sent to the customer.

C.13 Mobiliser Delete Network Entry

This state is used to delete a network entry.

The state expects that the variable named NETWORK_ENTRY_ID is already set in the current session.

Figure 66. Mobiliser Delete Network Entry State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

The ID of the network entry to delete - The id of the entry to delete.

C.14 Mobiliser Set Language (Session & Customer)

This state sets the language for the current Brand Mobiliser session and the Money Mobiliser customer. The language must exist in Brand Mobiliser database.

The following follow-up states are defined:

- FAIL: If an error occurs.
- OK: If everything went well.
- Dyn: Not in use.

The screenshot shows a configuration window titled "New Mobiliser Set Language (Session & Customer)". It is divided into two main sections: "Input Variables" and "Follow-up States".

Input Variables (Showing: 1-3):

- Row 1: A dropdown menu with "-- Select an entry --" and a red dot icon. Label: "Identification type."
- Row 2: A text input field with a green checkmark icon. Label: "Identification of the customer (overrides session)."
- Row 3: A text input field with "LANGUAGE" and a red dot icon. Label: "The selected Language. Expects a locale: (en, ..."

Follow-up States (Showing: 1-2):

- Row 1: A green checkmark icon, the text "OK", and a dropdown menu with "-- Select a follow-up state --".
- Row 2: A yellow warning triangle icon, the text "Fail", and a dropdown menu with "-- Select a follow-up state --".

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 67. Mobiliser Set Language (Session & Customer) State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Selected Language - The language to be set for the session.

C.15 Mobiliser Set Blacklist Reason

This state sets the blacklist reason for a given MSISDN. If the MSISDN is not set, the current user is being blacklisted. The following follow up states are defined:

- FAIL: If an error occurs.
- OK: If everything is OK.
- Dyn: Not in use.

Figure 68. Mobiliser Set Blacklist Reason State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

ID of the blacklistreason - The ID of the blacklist reason. This value must reference a value as listed in the section 'Appendix – Mobiliser Web Service Lookup Values' for the table 'Blacklist Reason'

PIN from the MSISDN - The PIN of the MSISDN that is blacklisted.

C.16 Mobiliser Set Identity

This state sets a new identity value of the specified type.

The customer whose identity is changed is identified by the current user session's MSISDN.

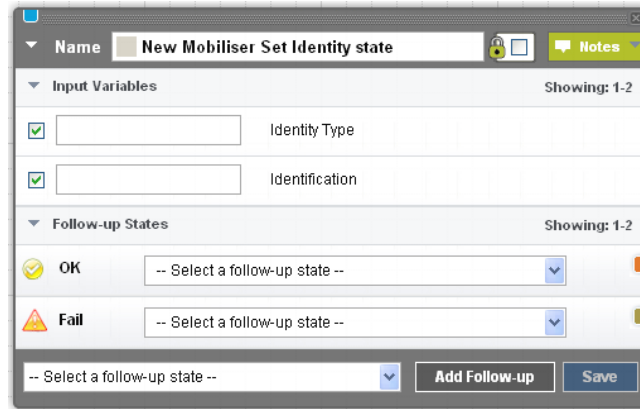


Figure 69. Mobiliser Set Identity State

Inputs

Identity type - The ID of the customer identity to set – this must match one of the ID values listed in the section 'Appendix – Mobiliser Web Service Lookup Values' and the table 'Identity Values'.

Identification – The value of the identity type to set.

C.17 Mobiliser Set Info Mode

This state sets the mode of how the customer should receive information.

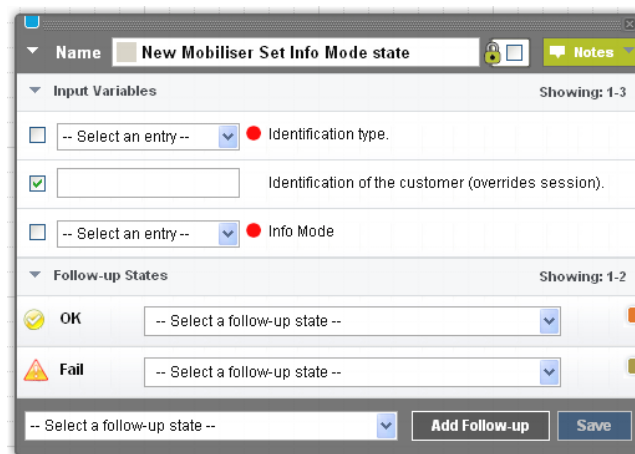


Figure 70. Mobiliser Set Info Mode State



Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Info Mode - The customer's info mode. EMAIL, SMS or BOTH.

C.18 Mobiliser Set Customer Attribute

This state sets a specific attribute value for a specific customer.

Figure 71. Mobiliser Set Customer Attribute State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Attribute ID – The ID value of the attribute to be changed.

Attribute Value – The new value to assign to the attribute.

C.19 Mobiliser Set Customer Name

This state sets the display name for the specific customer.

Figure 72. Mobiliser Set Customer Name State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

The new name of the customer – The value of the customer name to be changed. The name is split into separate items using a space as an item separator in order to extract a first, middle and last name. If the split results in only one entry, it is used as the last name. If the split results in more than three entries, the middle name is set to the first middle name found.

C.20 Mobiliser Set Primary PI

This state enables you to set a payment instrument as primary debit and/or credit.

Input Variables		Showing: 1-5
<input type="checkbox"/>	-- Select an entry --	● Identification type.
<input checked="" type="checkbox"/>		Identification of the customer (overrides session).
<input checked="" type="checkbox"/>	PI_ID	● Payment Instrument ID
<input type="checkbox"/>	-- Select an entry --	● Is Primary Debit
<input type="checkbox"/>	-- Select an entry --	● Is Primary Credit

Follow-up States		Showing: 1-2
<input checked="" type="checkbox"/>	OK	-- Select a follow-up state --
<input type="checkbox"/>	Fail	-- Select a follow-up state --

-- Select a follow-up state -- Add Follow-up Save

Figure 73. Mobiliser Set Primary PI State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Payment Instrument ID - The ID of the payment instrument.

Is Primary Debit - Defines whether the payment instrument should be used for primary debit.

Is Primary Credit - Defines whether the payment instrument should be used for primary credit.

C.21 Mobiliser Set Receipt Mode

This state sets the receipt mode for a given MSISDN. The receipt mode can be EMAIL, SMS, BOTH.

The screenshot shows a configuration window for a new state. The title bar reads "New Mobiliser Set Receipt Mode state". Below the title bar, there is a "Name" field containing "New Mobiliser Set Receipt Mode state" and a "Notes" button. The main area is divided into two sections: "Input Variables" and "Follow-up States".

Input Variables (Showing: 1-3):

- Item 1: A checkbox is unchecked, followed by a dropdown menu with "-- Select an entry --" and a red dot. The label is "Identification type".
- Item 2: A checkbox is checked, followed by a text input field. The label is "Identification of the customer (overrides session)".
- Item 3: A checkbox is unchecked, followed by a dropdown menu with "-- Select an entry --" and a red dot. The label is "Receipt Mode".

Follow-up States (Showing: 1-2):

- Item 1: A checkbox is checked, followed by a dropdown menu with "-- Select a follow-up state --". The label is "OK".
- Item 2: A checkbox is unchecked, followed by a dropdown menu with "-- Select a follow-up state --". The label is "Fail".

At the bottom of the window, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 74. Mobiliser Set Receipt Mode State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

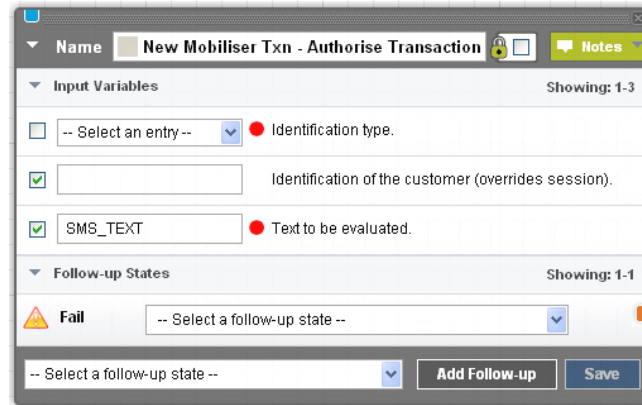
Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Receipt Mode – None, SMS, E-Mail or SMS and E-Mail.



C.22 Mobiliser Txn – Authorise Transaction

This state authorizes a transaction by changing its status. The authorisation step of a transaction first checks whether enough money is available for the transaction. If it is, this money is reserved and cannot be used for other transactions. The authorisation also considers the call fees.



The screenshot shows a configuration window for a state named "New Mobiliser Txn - Authorise Transaction". It is divided into two main sections: "Input Variables" and "Follow-up States".

- Input Variables:** Shows three variables:
 - Variable 1: A dropdown menu with "-- Select an entry --" and a red dot icon. Label: "Identification type."
 - Variable 2: A text input field with a green checkmark icon. Label: "Identification of the customer (overrides session)."
 - Variable 3: A text input field with "SMS_TEXT" and a red dot icon. Label: "Text to be evaluated."
- Follow-up States:** Shows one state:
 - State 1: A dropdown menu with "-- Select a follow-up state --" and a yellow warning triangle icon. Label: "Fail".

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 75. Mobiliser TXN – Authorise Transaction State

Inputs

Identification type - The type of customer identification to use; by 'mobile-no' (MSISDN), by customer id or by username.

Identification of the customer – The value or session variable to use as the identification value. If not set the current customer session value is used.

Text to be evaluated - This text is used for reference.

C.23 Mobiliser Txn – Charge Fee

The Charge Fee state performs a transaction that is not a financial transaction, e.g. request account balance. This state specifies the fee charged for the transaction.

The screenshot shows a configuration window titled "New Mobiliser Txn - Charge Fee state". It has a "Name" field with the text "New Mobiliser Txn - Charge Fee state" and a "Notes" button. Below the name are three sections:

- Input Variables (Showing: 1-3):**
 - USE_CASE_ID Use Case ID.
 - CURRENCY Currency.
 - PaymentInstrument ID
- Output Variables (Showing: 1-1):**
 - TXN_ID Transaction ID.
- Follow-up States (Showing: 1-1):**
 - OK -- Select a follow-up state --

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", an "Add Follow-up" button, and a "Save" button.

Figure 76. Mobiliser TXN – Charge Fee State

Inputs

Use Case ID - The ID of the use case, in which the fee is defined.

Currency - The currency of the transaction.

PaymentInstrument ID – Identifies the payment instrument that has the fee charged to it.

Outputs

Transaction ID - The ID of the transaction.

C.24 Mobiliser Txn – Pay Invoice

This state pays an invoice.

Section	Variable Name	Description
Input Variables	<input checked="" type="checkbox"/>	Amount you want to pay.
	<input checked="" type="checkbox"/>	Currency. If Amount is set, this field must be set
	<input checked="" type="checkbox"/>	Payer PaymentInstrumen ID.
	<input checked="" type="checkbox"/>	Payee PaymentInstrumen ID.
	<input checked="" type="checkbox"/>	INVOICE_ID ● Invoice ID you want to pay.
Output Variables	<input checked="" type="checkbox"/>	AUTH_CODE Auth. Code.
	<input checked="" type="checkbox"/>	TRANSACTION_ID Transaction ID.
Follow-up States	OK	-- Select a follow-up state --
	Fail	-- Select a follow-up state --

Figure 77. Mobiliser TXN – Pay Invoice State

Inputs

Amount - The amount to pay.

Currency - The currency of the transaction.

Payer Payment Instrument ID - The ID of the payer's payment instrument.

Payee Payment Instrument ID - The ID of the payee's payment instrument.

Invoice ID - The ID of the invoice.

Outputs

Auth Code - The authorization code for the transaction.

Transaction ID - The ID of the transaction.

C.25 Mobiliser Txn – Pre-Authorise Send Money

This state simulates a transaction. The regular steps are:

-> Authorisation (Auth)

The authorisation step of a transaction first checks whether enough money is available for the transaction. If there is, this money is reserved and cannot be used for other transactions. The authorisation also considers the call fees.

-> Capture:

The capture step of a transaction performs the real transaction. The money and fees are debited here.

The PreAuth Send Money state simulates the authorisation procedure. This checks whether enough money is available for the transaction. Then the customer is asked to confirm the transaction.

The screenshot shows a configuration window for the state "Pre-Auth Send Money". It is divided into three main sections: Input Variables, Output Variables, and Follow-up States. Each section has a "Showing:" indicator and a "Notes" button.

- Input Variables (Showing: 1-5 | 6-10 | 11-11 | All):**
 - EUR: Currency of transaction.
 - 193: Use Case ID.
 - AMOUNT: Amount to transfer.
 - By mobile-no: Payer Identification Type.
 - [Empty field]: Payer Identification.
- Output Variables (Showing: 1-3):**
 - PAYER_FEE: The fee for the payer.
 - PAYEE_FEE: The fee for the payee.
 - PAYEE_NAME: The name of the payee.
- Follow-up States (Showing: 1-2):**
 - OK: MPIN
 - Fail: ERROR Preauthorized Send Money

At the bottom, there is a dropdown menu with "-- Select a follow-up state --", and two buttons: "Add Follow-up" and "Save".

Figure 78. Mobiliser Txn – PreAuth Send Money State

Inputs

Currency of transaction - The currency for the transaction

Use Case ID - The Use Case ID identifies the type of transaction. For a CashOut transaction, the Use Case ID is 1005.

Amount - The amount of the transaction

Payer Identification Type - The identification type of the payer. This can be either “by mobile No.”, “by customer-id” or “by user-name”.



Payer Identification - The payer identification. Normally this is retrieved through the mobile No. so this field can be left blank.

Payer PI ID - The payment instrument used by the payer is identified here.

Payer PI class - The class of the payment instrument used by the payer. This could be “Stored Value Account”, “Bank Account”, “Credit Card” or “External Account”.

Payee Identification Type - The identification type of the payee. This can be “by mobile No.”, “by customer ID” and “by user name”.

Payee Identification - The identification of the payee.

Payee PI ID - The identification of the payment instrument for the payee.

Payee PI class - The class of the payment instrument used for the payee. This could be “Stored Value Account”, “Bank Account”, “Credit Card” or “External Account”.

Outputs

The fee for the payer - This fee is debited to the payer.

The fee for the payee - This fee is the commission paid to the agent for the Cash Out service.

The name of the payee - The name of the payee.

C.26 Mobiliser Txn – Send Money

This state performs a transaction. The regular steps are:

-> Authorisation (Auth)

The authorisation step of a transaction first checks whether there is enough money available. If there is, this money is reserved and cannot be used by another transaction. The authorisation also considers all fees.

-> Capture:

The capture step of a transaction performs the real transaction. The money and fees are debited here.

In this state, an Authorisation is carried out and captured automatically through the AutoCapture flag.

The screenshot shows a configuration window for the 'Send Money state'. It is divided into three main sections: 'Input Variables', 'Output Variables', and 'Follow-up States'.
- **Input Variables:** Contains five fields. 'EUR' is selected with a checkbox and labeled 'Currency of transaction.' '193' is selected and labeled 'Use Case ID'. 'AMOUNT' is selected and labeled 'Amount.' There is an empty field with a checkbox labeled 'Optional PIN for Bank Accounts or other Pls.'. 'By mobile-no' is selected in a dropdown menu and labeled 'Payer Identification Type.'
- **Output Variables:** Contains two fields. 'TXN_ID' is selected and labeled 'Transaction ID.'. 'AUTHCODE' is selected and labeled 'Authorisation code.'
- **Follow-up States:** Contains two states. 'OK' is selected with a checkmark icon and a dropdown menu showing 'Get Latest SVA Balance'. 'Fail' is selected with a warning icon and a dropdown menu showing 'ERROR Send Money'.
At the bottom, there is a dropdown menu with '-- Select a follow-up state --', an 'Add Follow-up' button, and a 'Save' button.

Figure 79. Mobiliser TXN – Send Money State

Inputs

Currency of transaction - The currency for the transaction

Use Case ID - The Use Case ID identifies the type of transaction. The Use Case for a CashOut transaction is 1005.

Amount - The Amount for the transaction

Optional PIN for Bank Accounts or other Pls - If money is debited from a bank account, the customer must enter the pin for the respective bank account.

Payer Identification Type - The identification type of the payer. This can be “by mobile No.”, “by customer ID” or “by user name”.

Payer Identification - The payer is identified here. Normally, this ID is retrieved through the mobile number. so this field can be left blank.



Payer PI ID - The identification of the payment instrument used by the payer.

Payer PI class - The class of the payment instrument used by the payer. This could be “Stored Value Account”, “Bank Account”, “Credit Card” or “External Account”.

Payee Identification Type - The identification type of the payee. This could be “by mobile No.”, “by customer ID” or “by user name”.

Payee Identification - The identification of the payee.

Payee PI ID - The identification of the payment instrument for the payee.

Payee PI class - The class of the payment instrument used for the payee. This could be “Stored Value Account”, “Bank Account”, “Credit Card” or “External Account”.

Outputs

Transaction ID - Internal system reference number that identifies the transaction.

Authorisation code - This code identifies the transaction. It is the customer’s reference to the transaction in the system and appears on the receipt. The customer can use this code to track the transaction and request assistance from customer support if there are any problems.

D. Appendix – Using the Legacy Application Editor View

The Application Editor view provides a more expanded view of all the states in an application and is the secondary way of adding new states, modifying the configuration and deleting states or transitions.

Although fully functional, the Application Editor has restrictions of what can be shown on one web page and for large applications it is difficult to visualise the processes and can take time to load the complete details of the application. That is why the Application Composer is the preferred mechanism for application and state configuration.

However, issues of visualising the whole process using the Application Editor can be mitigated by breaking applications down into smaller units and relying on the ‘Go To Application’ or ‘Method Call’ States, described later.

D.1 Adding States

After creation of a new application, there will always be the ‘Start Application’ as the initial state.

New states are always added from the configuration of existing states, therefore the initial state acts as the seed point of the application.

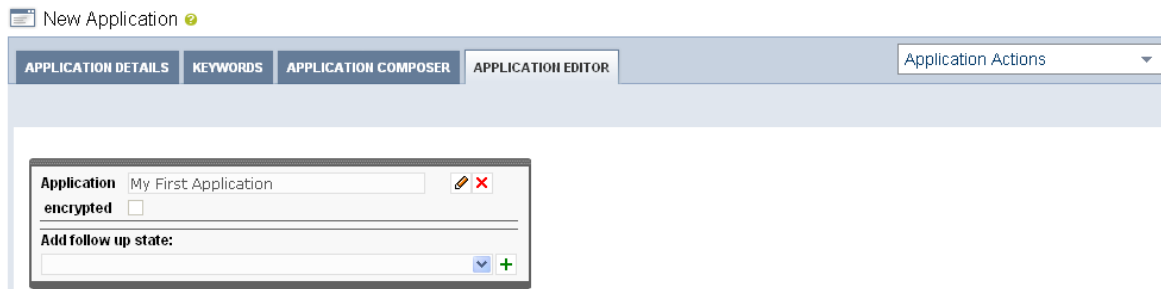






Figure 80. Start Application State

This state allows for three main actions, identified by the icons on the state;

	Edit the configuration of the state. When clicked, the edit icon will change to save icon  indicating an edit mode. Clicking the save icon will toggle back to the edit icon indicating a view mode.
	Remove this state.
	Add a follow-up state, as selected from the drop-down list on the left of the icon.

To add a transition from the “Start Application” state (i.e., My First Application), choose a state from the “Add follow up state” drop-down list, and then click the **Add** icon. The figure below shows all the available follow up states.

Note: the “Add follow up state” dropdown list is disabled while in the edit mode.

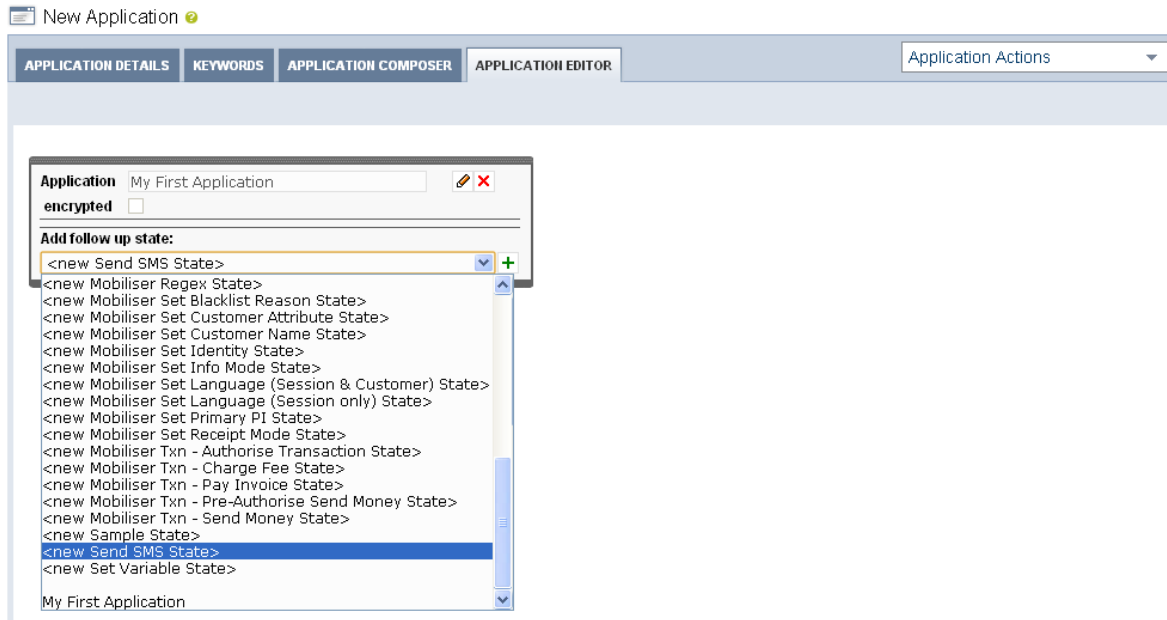


Figure 81. Choose Next State

Note: For a new application, you can add new instances of states. When you have created some states, you can instead add a follow-up to an existing state. This will result in the application’s control-of-flow going to the point of that existing state. In the figure, you can select the “My First Application” state that essentially loop-back to itself, although in this case it does not really make sense to do so.

The following figure shows the view of the Application Editor, after adding a new ‘Send SMS’ state.

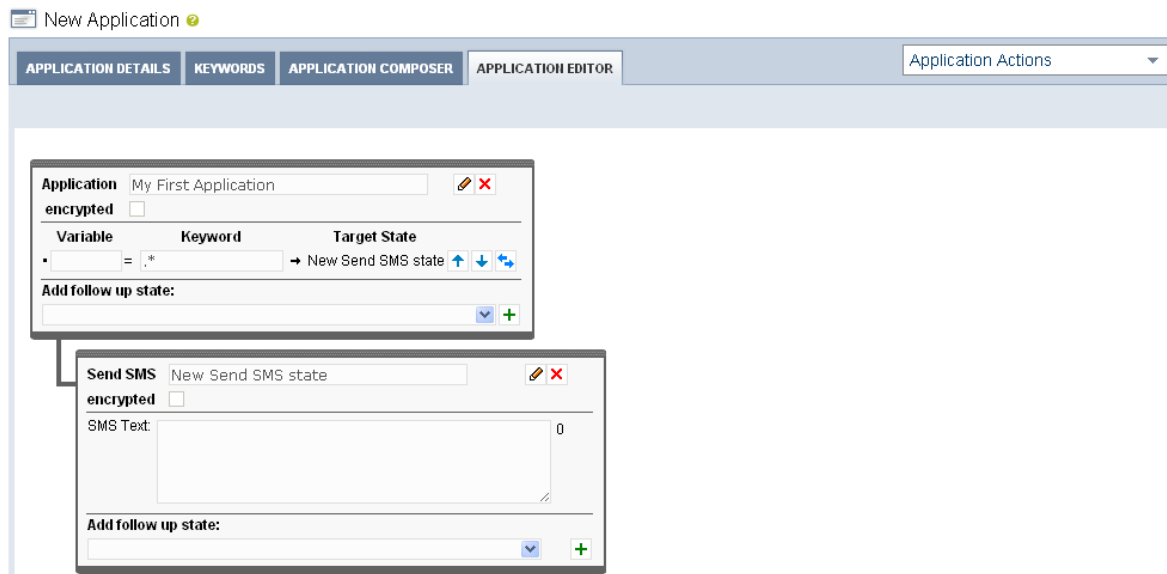


Figure 82. New State Added

Note: After a new state is added, it is automatically assigned the name ‘New <type of state> state’. This name is shown in the new state property window and the state window from which it was created.

Note: After a new state is added, it is also automatically assigned the keyword pattern of ‘.*’ (match to any and all characters).

Use the Edit button to change to edit mode, and modify the default value to something more relevant. Click the save button when finish editing.

The figure below shows what the application looks like after a follow-up state has been added which is an existing state (the initial state). This application has been modified so that if the user replies to the message with the response ‘again’, they will see the message again, otherwise the application ends.

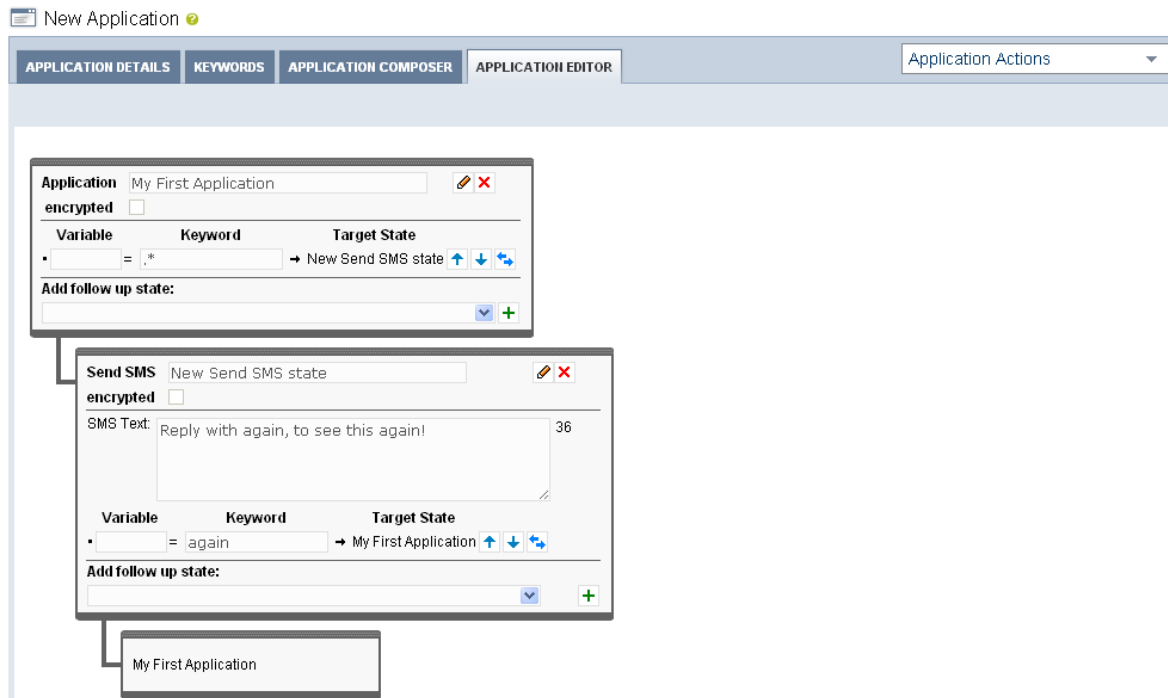




Figure 83. New Link to Existing State Added

D.2 Modifying State Configuration

To change any state configuration you click on the **Edit** icon to change into an edit mode. That icon then changes into;

	Save changes.
	Revert changes made and go back to last saved values.

All editable fields on the state property window are then enabled. The figure below shows the state property window, while modifying the ‘New Send SMS state’ configuration. Click the Save icon to save and return to the view mode.

Note: the “Add follow up state” dropdown list is disabled while in this edit mode. Also, only one state can be in edit mode at any one time.



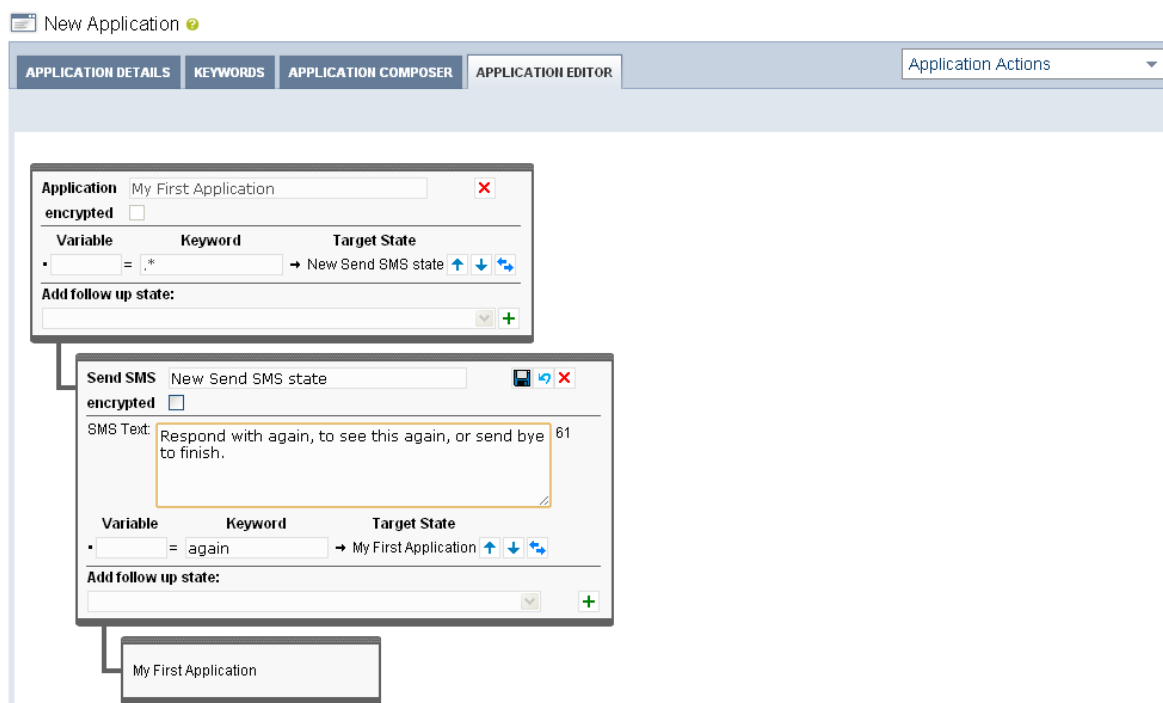




Figure 84. Modifying state configuration

D.3 Removing States and Transitions

Applications may have states and transitions removed from them. It is important to understand the different implications of removing either a state or a transition, and the resulting changes made to application.

- To remove a state, use the **Remove** icon .
- To remove a transition, use the double horizontal arrow icon next to the target state name .

Removing a State

Removing a state will permanently delete that state and any transitions that were associated with it.

- States that transitioned INTO the deleted state will have effected transitions.

Note: Deleting a state that had its own follow-up transitions may also then detach that portion of the application from the application flow itself.

A portion of the application that is detached is not shown in the Application Editor (nor the Application Composer), but still exists in the system. Detached portions of the application are never reachable by the customer through the application flow. To re-attach that portion into the application you can add a new state or add a transition that flows into a state in the detached portion.

Removing a Transition

Removing a transition will permanently delete that transition, but will not remove the follow-up state that it was associated with. To re-attach the follow-up state into the application add a new transition.

Note: Deleting a transition may also detach the follow-up state from the application flow.

E. Appendix – Mobiliser Web Service Lookup Values

This appendix section references lists of static values that are returned by the Mobiliser Web Services and the Mobiliser Web Service States.

E.1 *Blacklist Reason*

ID	Description
0	OK
1	Wrong PIN or password entered 3 times
2	Device lost
3	Payment dispute
4	Fraud suspicion
5	Registration pending
6	Rejected by third party scoring
10	Third party scoring failed
99	OFAC

E.2 *Card Types*

ID	Description
1	Visa
2	MasterCard
3	American Express

E.3 *Customer Cancellation Reasons*

ID	Description
0	Not cancelled
1	Double customer



E.4 Identification Types

ID	Description
0	MSISDN, in international format, e.g. +14155551234, +491701234567
1	Customer ID
2	IMEI
4	Foreign customer ID
5	Username
7	Email

E.5 Identity Types

ID	Description
0	Generic, see Issuer of identity
1	Citizenship Card
2	Identity Card
3	Foreigner Card
4	Personal Tax ID Number

E.6 Info Mode & Receipt Mode

ID	Description
0	None
1	SMS
2	Email
3	Email & SMS

E.7 User Rights/Product ID

ID	Description
2002	Mini Agent
2003	Super Agent

E.8 Errors

ID	Description
106	Wrong FI account PIN entered too many times (3 times)
116	Insufficient funds in FI account
117	Wrong FI account PIN
126	Wrong FI account PIN
127	Wrong FI account PIN
128	Wrong FI account PIN
206	Too many times wrong FI account PIN
1010	Entity not found
2611	Insufficient funds
2711	Agent SVA limit hit



SYBASE, INC.
WORLDWIDE HEADQUARTERS
ONE SYBASE DRIVE
DUBLIN, CA 94568-7902 USA
Tel: 1 800 8 SYBASE

www.sybase.com

DOCUMENT ID: DC01690-01-0110-01
LAST REVISED: March 2011

Copyright © 2010 Sybase, an SAP company. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase, and the Sybase logo, are trademarks of Sybase, Inc. or its subsidiaries. ® indicates registration in the United States of America. SAP and the SAP logo, are trademarks or registered trademarks of SAP AG in Germany and in several other countries. All other trademarks are the property of their respective owners.

