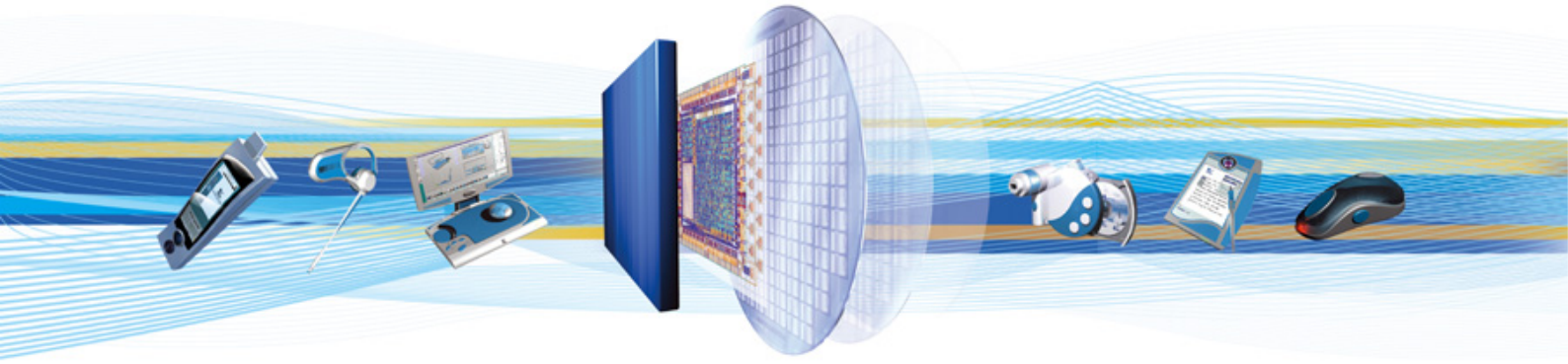
CSR Synergy Framework 3.1.0

CSR UI – User Interface

API Description

August 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

CSR_UI is a user interface toolkit which makes it possible for applications to easily interact with the user through either a text console and/or a virtual phone, consisting of a typical cellular phone keypad and a 128x160 pixel display. The application communicates with CSR_UI using a message based interface. Using a small set of signals it is possible for the application to display a consistent and functional user interface and receive user input. For the purpose of illustration this document will mainly focus on the graphical part and hence references to the text console layout will only be mentioned where appropriate. It is important to note that the API interface used by the application written on top of CSR_UI is completely independent of whether text mode and/or graphical mode is used.

2 Principles

This section describes the basic principles of CSR_UI. This section provides the foundation necessary to understand how to interact with CSR_UI from an application. It first describes the UI Emulator, which is the interface between the person using the application and CSR_UI. Then the concept of User Interface Elements (UIE) is explained, and the Display Stack (DS) is introduced. Finally, the key input handling is described.

2.1 UI Emulator

The UI Emulator of CSR_UI is shown in Figure 1.

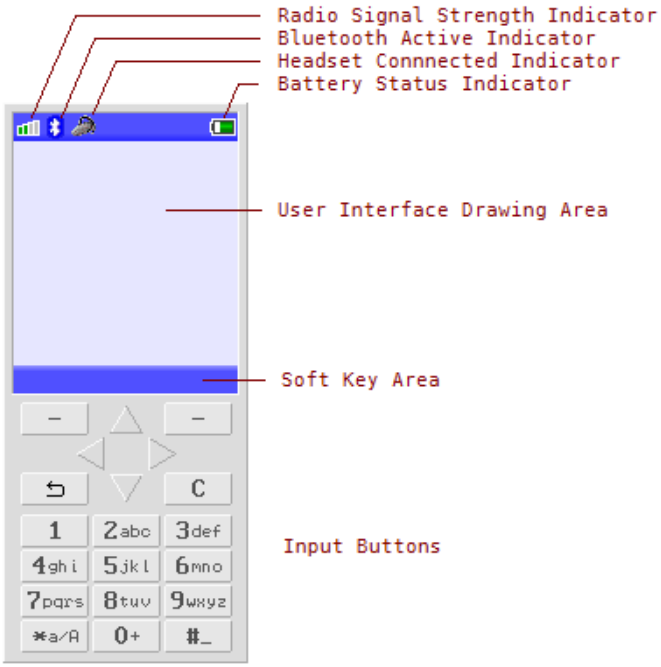


Figure 1: Layout of the virtual phone

The UI Emulator is a virtual phone consisting of a number of buttons (keypad) and a graphics display. The upper part of the graphics display is reserved for status icons, while the lower part is reserved for soft key strings and a scroll indicator. The User Interface Drawing Area (UIDA) is rendered with the particular User Interface Element (UIE) currently visible. In text console mode the input buttons will be taken directly from the platforms keyboard. Instead only the status bar in the top is printed to the console and icons are replaced by single letters. The User Interface Drawing Area and Soft Key Area is also present in text mode and just presented as strings to the user.

2.2 User Interface Elements

The User Interface Element (UIE) is the basic building block of the User Interface. The User Interface is constructed by creating UIEs and customising these to fit individual needs. UIEs are objects with properties. When they are created they are stored on the UIE Heap, which is managed automatically by CSR_UI. UIEs live on the UIE Heap from when they are created until they are explicitly destroyed. When UIEs have been constructed they can be displayed on the User Interface Drawing Area (UIDA) by signaling to CSR_UI that a particular UIE should be shown. Only one UIE can be visible on the UIDA at a time, and it is the responsibility of the applications to determine and maintain a virtual ownership of the user interface, when CSR_UI is contested by multiple applications.

All UIEs are referenced and handled by a UIE handle, which is returned when the UIE is created. There exists a fixed number of different UIE types, which are described in Section 2.5. When creating a UIE, the type shall be specified as illustrated in Figure 2.

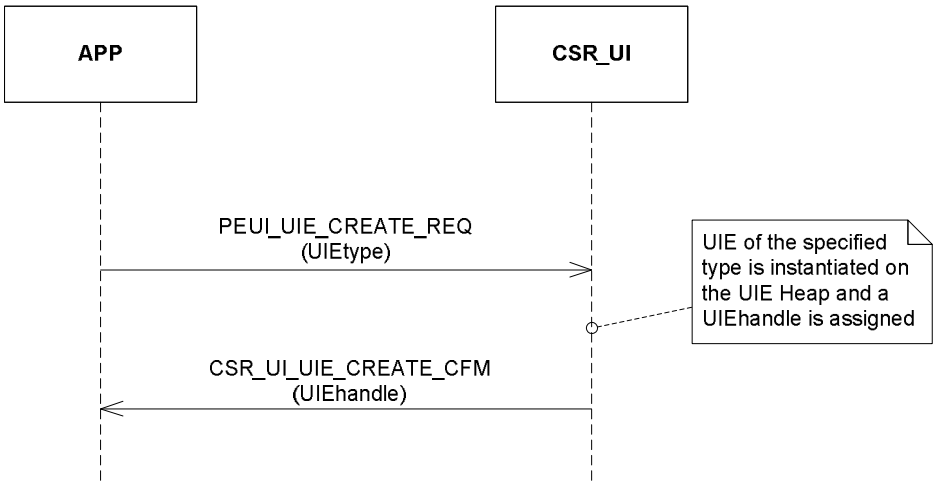


Figure 2: The process of creating a new UIE

When the UIE is created the data contained in the UIE object shall be set to define its visual and behavioural properties, such as heading, action of specific keys and text strings for the soft keys. Figure 3 illustrates how a UIE of the type Dialog can be set. For each UIE type there is a corresponding signal for setting the properties of the UIE. For certain UIE types there is also a signal for retrieving the properties.

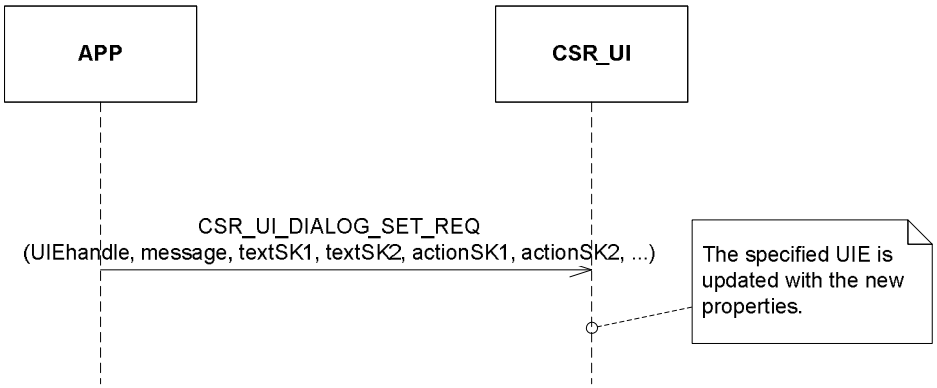


Figure 3: The process of setting the properties of a UIE

UIEs on the UIE heap can be shown or not shown, depending on whether or not they are referenced by the Display Stack which stores the current navigational state of the user interface. To make a UIE visible to the user, it shall be explicitly ordered shown by an appropriate message, which in turn will put it onto the Display Stack.

2.3 Display Stack

The Display Stack stores the current navigational state of the user interface. The Display Stack is a conventional stack of Display Stack Items (DSI). Each DSI contains a UIE handle to a UIE as well as other information related to the behaviour and appearance of CSR_UI when this UIE is displayed.

Initially the Display Stack is empty, which means that User Interface Drawing Area (UIDA) will be empty. When an application orders a UIE to be shown, a new DSI with a UIE handle to this UIE is pushed onto the Display Stack. This means that the UIE will be drawn on the UIDA and become visible to the user and subject to user input. If an application orders an additional UIE shown it will be pushed onto the Display Stack, and the topmost DSI will determine which UIE becomes visible on the UIDA. When a UIE is ordered hidden the corresponding DSI is removed from the stack (and destroyed). If the DSI at the top of the display stack is removed, the UIE referenced by the next DSI in the stack will become the currently displayed UIE.

In addition to a reference to a UIE, each DSI also contains the following fields:

- Priority: an integer in the range CSR_UI_LOWESTPRIORITY to CSR_UI_HIGHESTPRIORITY. Priority determines the relative priority of the DSI, which is respected when pushing a new DSI onto the Display Stack. If a new DSI is ordered on the Display Stack with a lower priority than the DSI at the top of the Display Stack, the new DSI is inserted at a lower position in the stack after any higher priority DSIs.
- Input Mode: determines how the UIE reacts to user input. This is described further in Section 2.4.
- Listener: The scheduler queue of the application that receives notifications when the user interacts with the UIE.

The UIE handle, priority, input mode and listener shall be given when a UIE is ordered shown. Note that these are not properties of the UIE, but properties attached to the DSI. The UIE referenced by the topmost DSI on the Display Stack is known as 'the currently displayed UIE'. The input mode specified in the topmost DSI is known as 'the current input mode'. The listener specified in the topmost DSI is known as 'the current listener'.

2.4 Keyboard Handling

The keyboard is controlled by CSR_UI. When the user clicks one of the buttons of the UI Emulator, the key press is processed by CSR_UI, taking into consideration the currently displayed UIE type and the current input mode.

There are four input modes defined:

- AUTO – Some keys are intercepted and all other keys are discarded.
- AUTOPASS – Some keys are intercepted and all other keys are sent to the current listener.
- BLOCK – All key presses are discarded.
- PASS – All key presses are sent to the current listener.

In AUTO and AUTOPASS input mode the user interface will intercept certain key presses depending on the type of the currently displayed UIE. If a key is not intercepted it is sent to the current listener in the AUTOPASS mode in a CSR_UI_KEYDOWN_IND signal and discarded in the AUTO mode. The keys that will be intercepted in AUTO and AUTOPASS mode is described in Section 2.5 for the individual UIE types. These are known as 'key filters'. If a key is filtered it will result in some internal action defined by the UIE type.

In BLOCK input mode all key presses will be discarded, and in PASS input mode all key presses will be directed to the current listener as CSR_UI_KEYDOWN_IND signals. In BLOCK and PASS input mode key filters will be ignored.

Figure 4 illustrates graphically how CSR_UI processes key input.

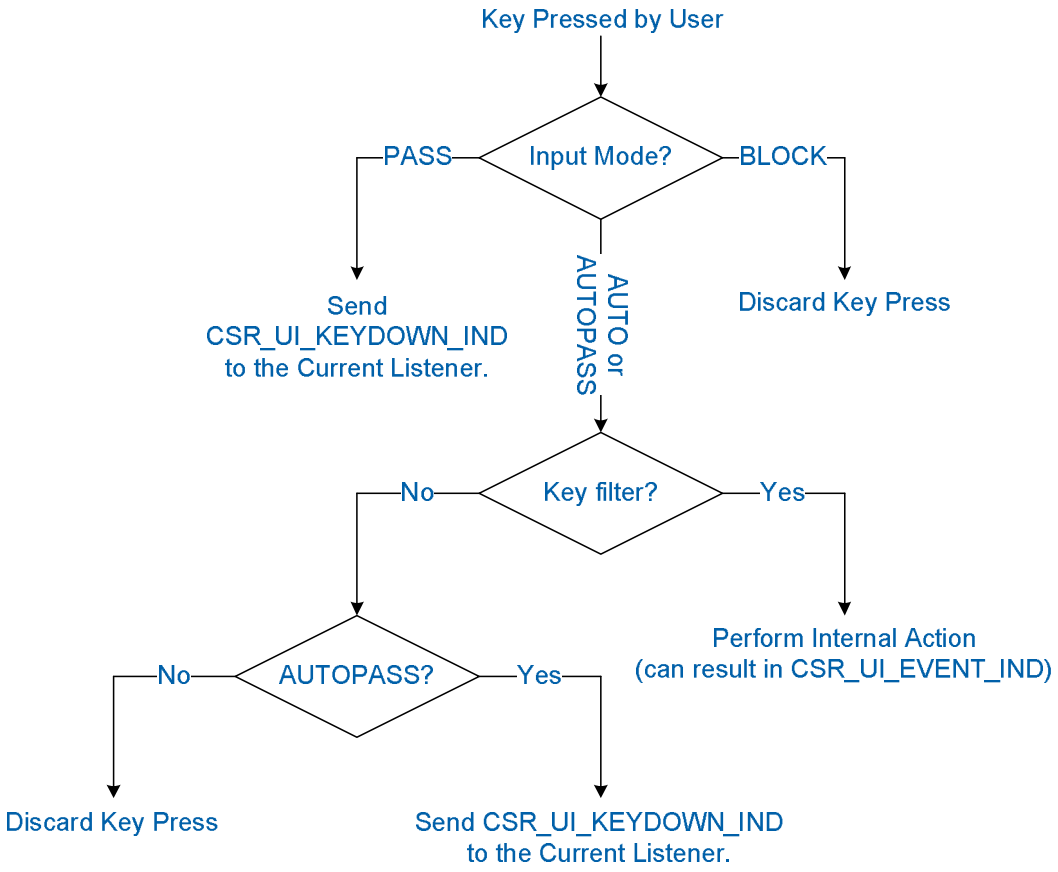


Figure 4: Key input processing

The regular key pad buttons are denoted N0 to N9. The lower left and right keys are denoted STAR and HASH respectively. This group of 12 keys is known as the input keys.

The left and right soft keys are denoted SK1 and SK2 respectively. The button below SK1 is denoted BACK, and the button below SK2 is denoted DEL. Finally the directional navigation keys are denoted UP, DOWN, LEFT and RIGHT. This group of 8 keys is known as the control keys.

In addition there is an ASCII key group.

All input keys and control keys have dedicated key codes defined by CSR_UI, and these can be accessed by pre-pending CSR_UI_KEY_ to their name. For example soft key 1 is defined as CSR_UI_KEY_SK1.

2.5 User Interface Element Descriptions

This section describes the User Interface Elements (UIE), their properties and appearance and how they are used. A common feature among UIEs is the ability to attach actions to certain elements of the UIE. These actions are triggered by the user when pressing certain keys, depending on the particular UIE type displayed. An action can be assigned three different values:

1. CSR_UI_DEFAULTACTION
2. UIE handle of another UIE.
3. UIE handle of an Event.

In the first case the action will do nothing when it is triggered. In the second case the UIE will be shown, as if the application has ordered it shown. The new DSI will inherit the properties of the current DSI item at the top of the

Display Stack. In the third case the Event will be triggered. The UIE type Event is a non-displayable UIE used for signalling to applications when actions are triggered in a more convenient way than using the AUTOPASS or PASS input modes. The Event is further described in Section 2.5.4.

In the following, the individual UIEs are described.

2.5.1 Menu

A standard single dimensional menu. Figure 5 shows an example of this UIE. Menu UIEs are typically used for creating the basic menu structure of the user interface. However, they can also be used for more specialised cases, such as showing search results, files in a directory, trusted device list and similar.

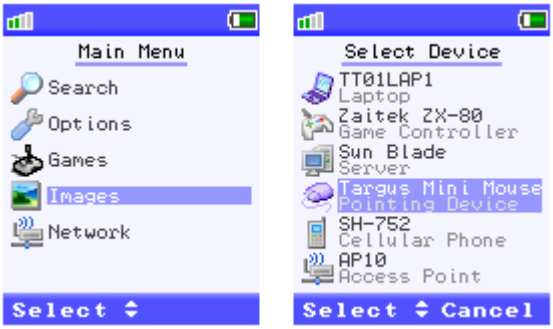


Figure 5: Visual Appearance of a Menu UIE. Left side example shows a typical main menu with icons for each item. Right side example shows the use of sub labels and icons for showing the class of device

Menus are initially empty when created. The only properties that shall be given in the CSR_UI_MENU_SET_REQ signal are the heading and the strings for the soft keys. When the Menu has been created and it's properties set, items can be added, modified and removed, by using the CSR_UI_MENU_ADDITEM_REQ, CSR_UI_MENU_SETITEM_REQ and CSR_UI_MENU_REMOVEITEM_REQ signals. It is possible to dynamically add and remove items when the Menu is displayed, which makes the Menu useful for live searching while the user is allowed to interact with the already found entries. Each menu item is assigned a numeric key when the item is added. These are assigned by the application, and need not be unique, but it is recommended to use unique numbers as the key is also the handle used for accessing the menu item later. If several menu items with the same key exists, the topmost item will be used if trying to access the key.

The behaviour of SK1, SK2, BACK and DEL depends on the action properties of the currently selected menu item. When a menu item is added to a menu an appropriate action shall be specified for each of these four keys. If the action is an Event, the Event will be triggered with the properties defined in the particular Event. If the action is another displayable UIE, the user interface will automatically navigate into this UIE. The latter is used for defining sub menu relationships. If the key is unused the action can be assigned the CSR_UI_DEFAULTACTION value which results in nothing. A special feature of the Menu is that the BACK key is filtered if the priority specified in the referencing DSI is CSR_UI_LOWESTPRIORITY. This means that if CSR_UI_LOWESTPRIORITY is used for a Menu it is possible to create a static menu structure of several UIEs, which can be traversed by CSR_UI without interacting with the application. The UP and DOWN keys are used for navigating up and down in the Menu.

All other key presses are sent to the default listener as CSR_UI_KEYDOWN_IND messages in the AUTOPASS input mode and discarded in the AUTO input mode.

Key filters (for priority equal to CSR_UI_LOWESTPRIORITY):

Key	Internal Action
UP, DOWN	Move cursor up/down.
SK1, SK2, DEL	CSR_UI_DEFAULTACTION → Discard/Pass. UIE → Show the UIE. Event → Trigger Event.
BACK	Hide the UIE (unless the Display Stack only contains one item)

Key filters (for priority > CSR_UI_LOWESTPRIORITY):

Key	Internal Action
UP, DOWN	Move cursor up/down

Key	Internal Action
SK1, SK2, BACK, DEL	CSR_UI_DEFAULTACTION → Discard/Pass. UIE → Show the UIE. Event → Trigger Event.

Note that the behaviour of the BACK button depends on the priority. It is only caught by the key filter if the priority is CSR_UI_LOWESTPRIORITY. If priority is higher the associated action is triggered, which can be CSR_UI_DEFAULTACTION, an Event or another UIE.

2.5.2 Dialog

A simple full screen dialog box with a heading and a centred text message. This can be used for conveying a message to the user and receive a response to the message. The text message can have any length, and will be automatically word wrapped to fit the screen. If the message is too large to fit into one screen, the user can use the UP and DOWN buttons to scroll up and down in the message. Text and actions for the two soft keys SK1 and SK2 can be configured, and actions can be attached to the BACK and DEL button.

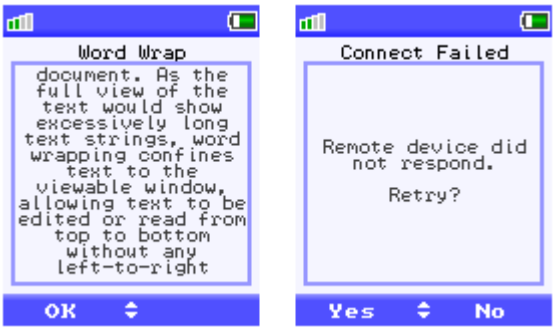


Figure 6: Visual appearance of a Dialog UIE

When a Dialog is created, it is initially empty, and shall be set using the CSR_UI_DIALOG_SET_REQ signal. Using this signal, the heading, text message, soft key strings and actions can be assigned. It is possible to attain dynamic behaviour by using this set signal while the UIE is displayed.

Key filters:

Key	Internal Action
UP, DOWN	Scroll up/down
SK1, SK2, BACK, DEL	CSR_UI_DEFAULTACTION → Discard/Pass. UIE → Show the UIE. Event → Trigger Event.

2.5.3 Input Dialog

A general purpose input dialog that can be used for receiving a string that the user can input. The Input Dialog can be configured to use different key maps for the key pad to facilitate input of different kind of strings. The NUMERIC key map is used for inputting the digits 0 to 9. The CONTROLNUMERIC key map extends this to include the special signs # (hash), * (star) and + (plus). Finally the ALPHANUMERIC key map allows for input of all alphabetical characters (lower and upper case) as well as a number of special signs and characters, including space and the new line character. 0contains additional information about the characters contained in these key maps.

Figure 7: Visual appearance of an Input Dialog UIE

When an Input Dialog is created, it is initially empty, and shall be set using the CSR_UI_INPUTDIALOG_SET_REQ signal. A heading can be specified, and optionally an icon and a message, which will be displayed above the input box. If no icon or no message is provided neither will be displayed and the input box will extend all the way to the underline of the heading. A maximum string length shall also be supplied, which will limit the input. In addition, it is possible to fill in the input box with a text that can then be edited by the user. If the supplied text is longer than the specified maximum string length, the text will be truncated to the specified length.

When the user has entered a string, the application is notified by a message triggered by an Event attached to SK1, SK2 or BACK. The application can then read the entered string by sending a CSR_UI_INPUTDIALOG_GET_REQ message.

Key filters:

Key	Internal Action
LEFT, RIGHT	Complete input in multi key state.
0...9, #, *, DEL	Enter/modify/delete characters
SK1, SK2, BACK	CSR_UI_DEFAULTACTION → Discard/Pass. UIE → Show the UIE. Event → Trigger Event.
ASCII	The supplied character is inserted unmodified into the input box.

2.5.4 Idle Screen

Idle screen with logo, heading and an input box for entering a telephone number or control code. The heading is usually used for displaying the name of the cellular phone network operator or cell information. The input box is similar to the input box of the **Input Dialog** with the CONTROLNUMERIC key map.

Figure 8: Visual appearance of a Idle Screen UIE. The soft key string for SK2 will only be shown if the user has entered a number in the input field

When an Idle Screen is created, it is initially empty, and shall be set using the CSR_UI_IDLESCREEN_SET_REQ signal. The heading and soft key strings and action shall be specified. A maximum string length shall also be supplied, which will limit the input. In addition, it is possible to fill in the input box with a text that can then be edited by the user. If the supplied text is longer than the specified maximum string length, the text will be truncated to the specified length.

Note that the string for SK2 is only shown if the user has entered a number or control code in the input box. In the example of Figure 8 this is used for showing the user that the number can be dialled. Note, however, that the absence of user input only hides the string, it does not inhibit the soft key.

When the user has entered a number, the application is notified by a message triggered by an Event attached to SK1, SK2 or BACK. The application can then read the entered string by sending a CSR_UI_IDLESCREEN_GET_REQ message.

Key filters:

Key	Internal Action
LEFT, RIGHT	Complete input in multi key state.
0...9, #, *, DEL	Enter/modify/delete characters
SK1, SK2, BACK	CSR_UI_DEFAULTACTION → Discard/Pass. UIE → Show the UIE. Event → Trigger Event.
ASCII	The supplied character is inserted unmodified into the input box.

2.5.5 Event

An Event cannot be displayed, but acts as an independent entity that can be triggered by other UIEs by attaching Events to the action properties. In turn this generates and sends a message to the listener defined in the Event when the action is activated by a key press. The Event also contains a property that determines the input mode that should be set for the currently displayed UIE when the Event is triggered.

This makes it possible to send messages to other listeners than the default listener for the currently displayed UIE, and also makes it possible to suppress auto behaviour (due to key filters) on an UIE until the application resets the input mode back to AUTO or AUTOPASS.

Events are created like any other UIE, and are initially empty. Their properties shall be set using the CSR_UI_EVENT_SET_REQ signal. The properties of the Event are:

- Input Mode: the input mode to set when the Event is triggered.
- Listener: the queue ID of the application that will receive the CSR_UI_EVENT_IND signal when the Event is triggered.

When an Event has been created and it's properties set, it can be attached to the action properties of any other UIE by setting the action property to the UIE handle of the Event. When the user clicks the corresponding key when the UIE is visible, it will result in the following actions taking place:

1. The input mode will instantly be set to the input mode specified by the Event.
2. A CSR_UI_EVENT_IND will be sent to the listener specified by the Event. The indication will contain the UIE handle of the Event as well as the UIE handle of the currently displayed UIE. In addition it will contain a key, whose semantics depends on the UIE type.

3 CSR_UI Primitives

This section gives an overview of the primitives and parameters in the interface. Almost all communication with CSR_UI is message based. To send messages to CSR_UI it is possible to use the library defined in csr_ui_lib.h to easily construct the messages and dispatch them. For each signal that can be sent to CSR_UI there is a corresponding function that performs this operation. The functions name is the name of the corresponding signal in camel notation with the underscores removed and 'Send' appended. Example:

`CSR_UI_UIE_CREATE_REQ` → `CsrUiUieCreateReqSend()`

All strings are Unicode in UCS2 format with a null character termination.

In the following the individual signals used in the interface are described.

3.1 CSR_UI_UIE

Primitives \ Parameters	type	phandle	elementType	element/handle	listener	inputMode	priority
CSR_UI_UIE_CREATE_REQ	✓	✓	✓				
CSR_UI_UIE_CREATE_CFM	✓		✓	✓			
CSR_UI_UIE_SHOW_REQ	✓			✓	✓	✓	✓
CSR_UI_UIE_HIDE_REQ	✓			✓			
CSR_UI_UIE_DESTROY_REQ	✓			✓			

Table 1: CSR_UI_UIE Primitives

Description

These signals are used for creating and destroy UIEs and to show and hide them on the user interface. The create signal will result in a confirm to the application with the queue ID specified in the phandle parameter, which contains the handle of the UIE. This handle is subsequently used as a reference to the UIE, when showing, hiding and destroying it, as well as when using signals from other groups to access the UIE. Creation of an UIE will not affect the Display Stack. That is, the new UIE will not be shown until explicitly ordered to by the CSR_UI_UIE_SHOW_REQ message. The properties of the UIE shall be set using appropriate messages before showing it. If CSR_UI_UIE_SHOW_REQ is used with a UIE that is already referenced by the Display Stack, it will act as a re-show, in the sense that the existing DSI will be removed from the stack, and a new will be inserted using the parameters given in the signal.

Parameters

type	Signal type.
phandle	Queue ID of the application. The confirm is sent to this queue.
elementType	The type of the UIE. Can be one of: CSR_UI_UIETYPE_MENU, CSR_UI_UIETYPE_DIALOG, CSR_UI_UIETYPE_INPUTDIALOG and CSR_UI_UIETYPE_EVENT. These definitions corresponds to each of the UIE types described in Section 2.5.
element/handle	UIE handle of the created element, or the element to which access is requested.
listener	The Queue ID of the application that receives CSR_UI_KEYDOWN_IND signals when the UIE is visible and the user presses a key in PASS mode or an unfiltered key in the AUTOPASS mode.
inputMode	The initial input mode to use when showing the UIE. Can be one of: CSR_UI_INPUTMODE_AUTO, CSR_UI_INPUTMODE_AUTOPASS, CSR_UI_INPUTMODE_BLOCK or CSR_UI_INPUTMODE_PASS. See Section 2.4 for additional information.
priority	The priority to use when showing the UIE. If the Display Stack contains items with higher priority, the UIE will not be visible to the user before these are hidden. The priority can be any value between CSR_UI_LOWESTPRIORITY and CSR_UI_HIGHESTPRIORITY.

3.2 CSR_UI_MENU

Primitives \ Parameters	type	phandle	menu/handle	heading	textSK1	textSK2	position	key	icon	label	sublabel	actionSK1	actionSK2	actionBACK	actionBACK
CSR_UI_MENU_SET_REQ	✓		✓	✓	✓	✓									
CSR_UI_MENU_ADDITEM_REQ	✓		✓				✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_UI_MENU_REMOVEITEM_REQ	✓		✓					✓							
CSR_UI_MENU_REMOVEALLITEMS_REQ	✓		✓												
CSR_UI_MENU_SETITEM_REQ	✓		✓					✓	✓	✓	✓	✓	✓	✓	✓
CSR_UI_MENU_GETITEM_REQ	✓	✓	✓					✓							
CSR_UI_MENU_GETITEM_CFM	✓		✓					✓	✓	✓	✓	✓	✓	✓	✓
CSR_UI_MENU_SETCURSOR_REQ	✓		✓					✓							

Table 2: CSR_UI_MENU Primitives

Description

This group of signals is used for manipulating the properties of a UIE of type Menu. These signals can safely be used whether or not the particular Menu is referenced by the Display Stack. This enables dynamic behaviour.

Parameters

type	Signal type.
phandle	Queue ID of the application. The confirm is sent to this queue.
menu/handle	UIE handle of the particular Menu.
heading	Heading of the Menu.
textSK1/textSK2	Text appearing in the soft key area of the UI Emulator for the two soft keys.
position	The position to insert the item at. CSR_UI_FIRST for top of the Menu. CSR_UI_LAST for bottom of the Menu. Can also be an arbitrary integer specifying the position, i.e. 2 for second, 3 for third etc. If the integer is greater than number of items in the Menu the item will be inserted last.
key	A key shall be specified when adding an item. This key will then be used for subsequently accessing the item. Unlike UIE handles the key is specified by the application when adding the item, and CSR_UI does not enforce uniqueness on the keys. If multiple items have the same key, subsequent signals will access the topmost item. The key will also be attached to the CSR_UI_EVENT_IND signals sent when an Event is triggered while the menu item is selected.
icon	The icon to display for this item. See 0for a list of valid icon names. If an icon other than

	CSR_UI_ICON_NONE is specified the item will be double size to accommodate for the space needed by the icon. CSR_UI_ICON_EMPTY is an icon that will not draw anything, but still make the item double size.
label	The text of the menu item
sublabel	An optional sublabel appearing under the label. If sublabel is specified the item will be double size to accommodate for the space needed by the sublabel.
actionSK1	Action to execute when the user presses SK1 in AUTO or AUTOPASS mode.
actionSK2	Action to execute when the user presses SK2 in AUTO or AUTOPASS mode.
actionBACK	Action to execute when the user presses BACK in AUTO or AUTOPASS mode.
actionDEL	Action to execute when the user presses DEL in AUTO or AUTOPASS mode.

3.3 CSR_UI_EVENT

Primitives \ Parameters	type	event	inputMode	listener	displayElemnt	key
CSR_UI_EVENT_SET_REQ	✓	✓	✓	✓		
CSR_UI_EVENT_IND	✓	✓	✓		✓	✓

Table 3: CSR_UI_EVENT Primitives

Description

Used for setting the properties of an Event. CSR_UI_EVENT_IND is received by the listener when the user activates a key whose action is attached to the Event.

Parameters

type	Signal type.
event	UIE handle of the particular Event.
inputMode	The initial input mode to set for the currently displayed UIE when the Event is triggered. Can be one of: CSR_UI_INPUTMODE_AUTO, CSR_UI_INPUTMODE_AUTOPASS, CSR_UI_INPUTMODE_BLOCK or CSR_UI_INPUTMODE_PASS. See Section 2.4 for additional information.
listener	The queue ID of the application that receives the CSR_UI_EVENT_IND signal when the Event is triggered
displayElement	The currently displayed UIE when the Event is triggered.
key	The key code of the pressed key or the key of the selected menu item, if the Event was triggered by a menu item.

3.4 CSR_UI_DIALOG

Primitives \ Parameters	type	dialog	heading	message	textSK1	textSK2	actionSK1	actionSK2	actionBACK	actionDEL
CSR_UI_DIALOG_SET_REQ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 4: CSR_UI_DIALOG Primitives

Description

Used for setting the properties of a Dialog. This can be used when the UIE is visible to the user to attain dynamic behaviour.

Parameters

type	Signal type.
dialog	UIE handle of the particular Dialog.
heading	Heading to use for the Dialog.
message	Message to display in the Dialog. The message will automatically be word wrapped to fit to the screen. It is possible to insert new line characters (0x000A) to force a line break.
textSK1/textSK2	Text appearing in the soft key area of the UI Emulator for the two soft keys.
actionSK1	Action to execute when the user presses SK1 in AUTO or AUTOPASS mode.
actionSK2	Action to execute when the user presses SK2 in AUTO or AUTOPASS mode.
actionBACK	Action to execute when the user presses BACK in AUTO or AUTOPASS mode.
actionDEL	Action to execute when the user presses DEL in AUTO or AUTOPASS mode.

3.5 CSR_UI_INPUTDIALOG

Primitives \ Parameters	type	phandle	inputDialog/handle	heading	message	icon	text	textLength	keyMap	textSK1	textSK2	actionSK1	actionSK2	actionBACK	actionDEL
CSR_UI_INPUTDIALOG_SET_REQ	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_UI_INPUTDIALOG_GET_REQ	✓	✓	✓												
CSR_UI_INPUTDIALOG_GET_CFM	✓		✓				✓								

Table 5: CSR_UI_INPUTDIALOG Primitives

Description

These signals are used for setting the properties of an Input Dialog, as well as retrieving the string entered by the user.

Parameters

type	Signal type.
phandle	Queue ID of the application. The confirm is sent to this queue.
inputDialog/handle	UIE handle of the particular Input Dialog.
heading	Heading for the Input Dialog.
message	Optional message to display above the input box. Will only be shown if an icon is specified also.
icon	Optional icon to display above the input box. Will only be shown if a message is specified also.
text	The text in the input box. When specifying this text in the CSR_UI_INPUTDIALOG_SET_REQ signal it will be truncated to the value of textLength.
textLength	The maximum text length the user is allowed to enter into the input box.
keyMap	Key map to use for input. Can be one of: CSR_UI_KEYMAP_NUMERIC, CSR_UI_KEYMAP_CONTROL_NUMERIC or CSR_UI_KEYMAP_ALPHANUMERIC. See 0 for further details.
textSK1/textSK2	Text appearing in the soft key area of the UI Emulator for the two soft keys.
actionSK1	Action to execute when the user presses SK1 in AUTO or AUTOPASS mode.
actionSK2	Action to execute when the user presses SK2 in AUTO or AUTOPASS mode.
actionBACK	Action to execute when the user presses BACK in AUTO or AUTOPASS mode.
actionDEL	Action to execute when the user presses DEL in AUTO or AUTOPASS mode. This action will always be ignored, because the DEL key is used for editing the text in the input box.

3.6 CSR_UI_IDLESCREEN

Parameters / Primitives	type	phandle	idleScreen/handle	heading	text	textLength	textSK1	textSK2	actionSK1	actionSK2	actionBACK	actionDEL
CSR_UI_IDLESCREEN_SET_REQ	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_UI_IDLESCREEN_GET_REQ	✓	✓	✓									
CSR_UI_IDLESCREEN_GET_CFM	✓		✓		✓							

Table 6: CSR_UI_IDLESCREEN Primitives.

Description

These signals are used for setting the properties of an Idle Screen, as well as retrieving the string entered by the user.

Parameters

type	Signal type.
phandle	Queue ID of the application. The confirm is sent to this queue.
inputDialog/handle	UIE handle of the particular Idle Screen.
heading	Heading for the Idle Screen.
text	The text in the input box. When specifying this text in the CSR_UI_IDLESCREEN_SET_REQ signal it will be truncated to the value of textLength.
textLength	The maximum text length the user is allowed to enter into the input box.
textSK1/textSK2	Text appearing in the soft key area of the UI Emulator for the two soft keys.
actionSK1	Action to execute when the user presses SK1 in AUTO or AUTOPASS mode.
actionSK2	Action to execute when the user presses SK2 in AUTO or AUTOPASS mode.
actionBACK	Action to execute when the user presses BACK in AUTO or AUTOPASS mode.
actionDEL	Action to execute when the user presses DEL in AUTO or AUTOPASS mode. This action will always be ignored, because the DEL key is used for editing the text in the input box.

3.7 CSR_UI_DISPLAY

Primitives \ Parameters	type	phandle	element/handle	inputMode
CSR_UI_DISPLAY_SETINPUTMODE_REQ	✓		✓	✓
CSR_UI_DISPLAY_GETHANDLE_REQ	✓	✓		
CSR_UI_DISPLAY_GETHANDLE_CFM	✓		✓	

Table 7: CSR_UI_DISPLAY Primitives

Description

Signals to access and modify the Display Stack. CSR_UI_DISPLAY_SETINPUTMODE_REQ can be used for modifying the input mode of a DSI. Note that the DSI is identified by the UIE it references. If the same UIE is shown several times (not recommended) the signal will access the topmost DSI on the Display Stack only. CSR_UI_DISPLAY_GETHANDLE_REQ can be used for retrieving the UIE handle of the topmost DSI on the Display Stack (which is the currently displayed UIE).

Parameters

type	Signal type.
phandle	Queue ID of the application. The confirm is sent to this queue.
element/handle	UIE handle of the UIE referenced by the DSI item to set the input mode for.
inputMode	The input mode to apply to the DSI. Can be one of: CSR_UI_INPUTMODE_AUTO, CSR_UI_INPUTMODE_AUTOPASS, CSR_UI_INPUTMODE_BLOCK or CSR_UI_INPUTMODE_PASS. See Section 2.4 for additional information.

3.8 CSR_UI_STATUS

Primitives \ Parameters	type	percentage	headset1	headset2	active
CSR_UI_STATUS_BATTERY_SET_REQ	✓	✓			
CSR_UI_STATUS_RADIOMETER_SET_REQ	✓	✓			
CSR_UI_STATUS_HEADSET_SET_REQ	✓		✓	✓	
CSR_UI_STATUS_BLUETOOTH_SET_REQ	✓				✓
CSR_UI_STATUS_WIFI_SET_REQ	✓				✓

Table 8: CSR_UI_STATUS Primitives

Description

Signals used for modifying the icons of the status bar of CSR_UI.

Parameters

type	Signal type.
percentage	An integer between 0 and 100 specifying the status of the battery or the status of the radio signal strength.
headset1	Specify CSR_UI_STATUS_HEADSET_PRESERVE to leave the primary headset state unmodified, CSR_UI_STATUS_HEADSET_NOT_PRESENT to set it as not present, CSR_UI_STATUS_HEADSET_PRESENT to set it as present without a battery indicator and CSR_UI_STATUS_HEADSET_PRESENT_INDICATOR(value) to set it as present with a battery indicator level as specified by the value given to the macro in the range 0-100.
headset2	Same as headset1, but targets the secondary headset.
active	A boolean value specifying whether or not Bluetooth/WiFi is active.

3.9 CSR_UI_KEYDOWN

Parameters / Primitives	type	displayElement	key
CSR_UI_KEYDOWN_IND	✓	✓	✓

Table 9: CSR_UI_KEYDOWN primitives

Description

This signal is sent to the current listener (property of the topmost DSI of the Display Stack) when the user presses a key in the PASS input mode or presses an unfiltered key in the AUTOPASS mode.

Parameters

type	Signal type.
displayElement	UIE handle of the currently displayed UIE.
key	The key code of the key pressed by the user. See 0for further details.

4 Examples

Figure 9 illustrates a typical usage scenario. This example creates two UIEs of the types Dialog and Event. The Event is attached to the action pertaining to soft key 1 (left), so that the application is notified when the user presses this button.

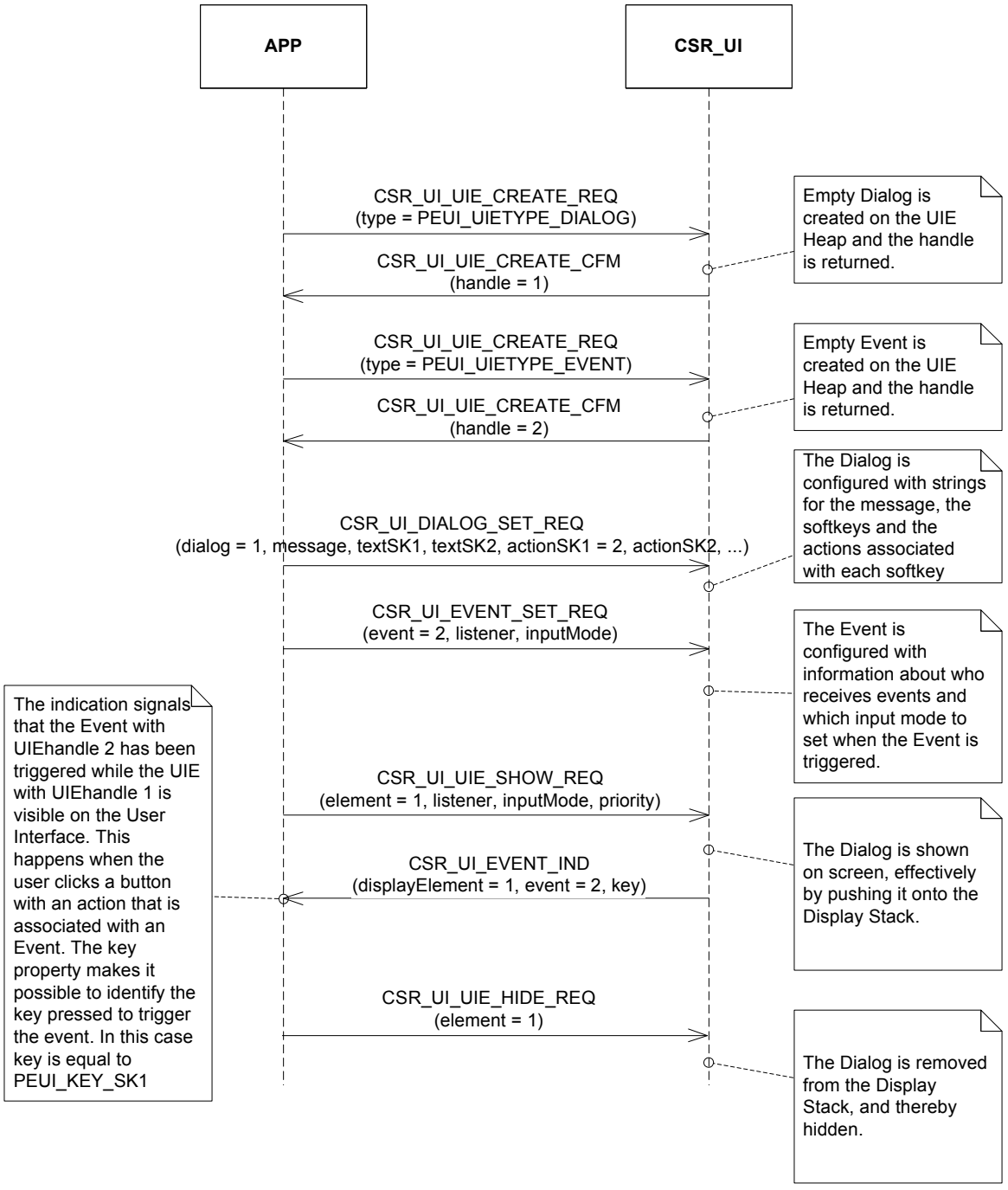


Figure 9: Typical usage example with a Dialog

After creating the UIEs and assigning their properties the Dialog is requested shown, and the application waits until the Event is triggered (i.e. the user clicks the soft key), and then hides the UIE.

The example of Figure 10 demonstrates how a menu is created, items added and events received:

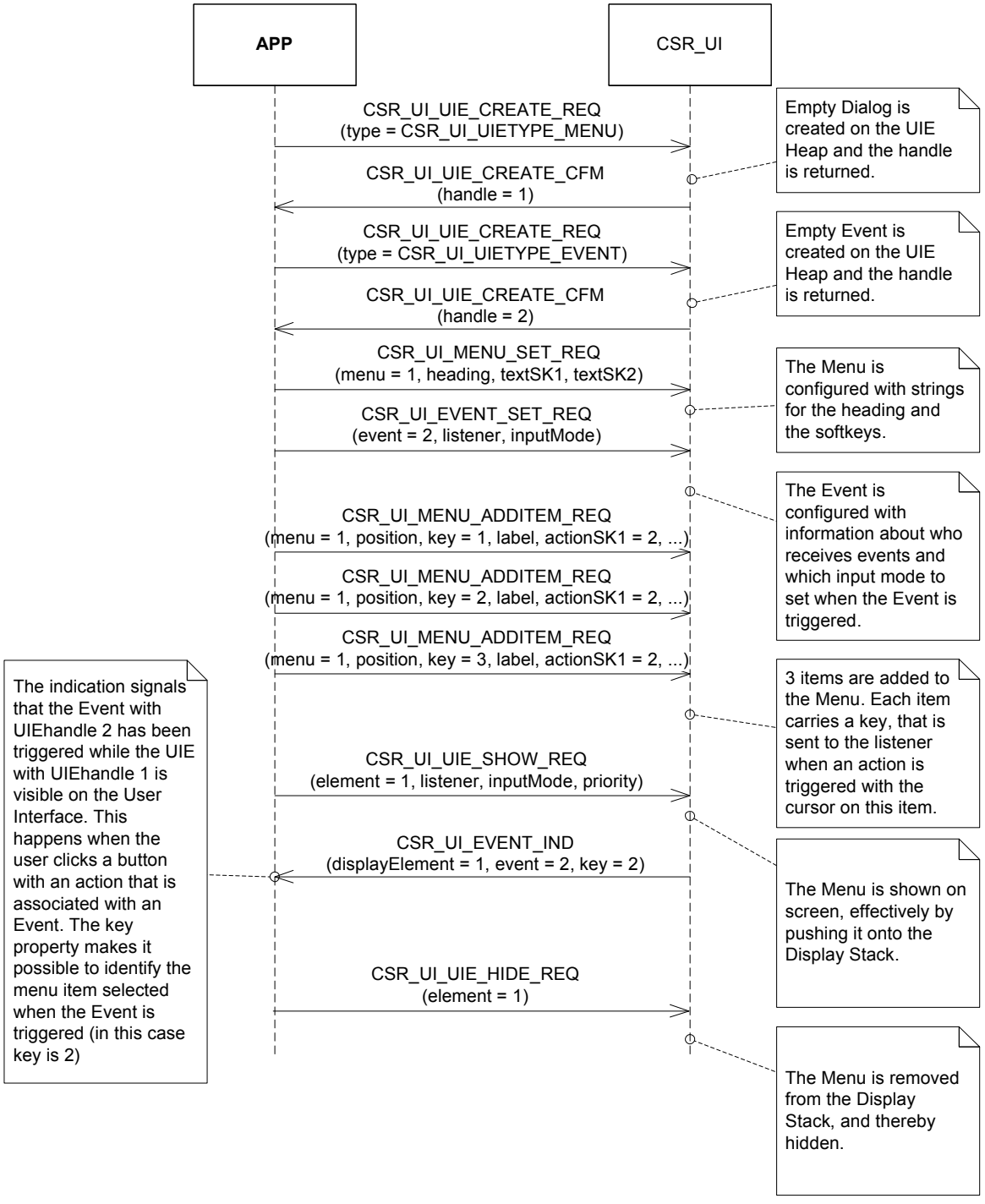


Figure 10: Typical usage example with a Menu

5 CSR_UI lower interface

Nearly all communication with CSR_UI is message based, but there exists a few functions, which can be called directly and are not message based. This section describes these functions.

void CsrUiTouchEvent(int16_t x, int16_t y);

Description:

This function can be called to signal a touch on a particular pixel within the extends of the display.

Parameters:

x	The x coordinate of the pixel
y	The y coordinate of the pixel

void CsrUiKeyEvent(uint16_t key);

Description:

This function can be called to signal activation of a particular button.

Parameters:

key — The type of button which has been activated. Can be one of: (defined in csr_ui_keycode.h)

CSR_UI_KEY_N0
CSR_UI_KEY_N1
CSR_UI_KEY_N2
CSR_UI_KEY_N3
CSR_UI_KEY_N4
CSR_UI_KEY_N5
CSR_UI_KEY_N6
CSR_UI_KEY_N7
CSR_UI_KEY_N8
CSR_UI_KEY_N9
CSR_UI_KEY_STAR
CSR_UI_KEY_HASH
CSR_UI_KEY_DEL
CSR_UI_KEY_BACK
CSR_UI_KEY_SK1
CSR_UI_KEY_SK2
CSR_UI_KEY_LEFT
CSR_UI_KEY_RIGHT
CSR_UI_KEY_UP
CSR_UI_KEY_DOWN

void CsrUiRegisterKey(int16_t left, int16_t top, int16_t right, int16_t bottom, uint16_t key);

Description:

Call to register a key. The key is registered as a rectangle with corners (left, top), (right, top), (left, bottom), and (right, bottom). If the CSR_UI is compiled with the preprocessor macro CSR_UI_DRAW_RECTANGLE_AROUND_KEY then a black rectangle is drawn around the key on the phone.

Parameters:

left	The left coordinate of the rectangle
top	The top coordinate of the rectangle
right	The right coordinate of the rectangle
bottom	The bottom coordinate of the rectangle
key	The type of the key registered. See the function **CsrUiKeyEvent**.

```
void CsrUiGraphicsRendererConfigurationGet(CsrUiGraphicsRenderer *renderer);
```

Description:

Called by CSR_UI during initialisation to retrieve information about the Graphics Renderer interface. This function is not implemented by CSR_UI, and shall be provided by some external entity. If graphics rendering is desired, the fields of the CsrUiGraphicsRenderer struct described below must be filled with appropriate values. If no graphics rendering is desired, the active field is set to FALSE, and the remaining fields can be left unassigned.

```
typedef struct
{
    /* Set to TRUE to activate the Graphics Renderer */
    CsrBool active;
    /* Top edge of screen (in pixel coordinates) */
    CsrUint16 top;
    /* Left edge of screen (in pixel coordinates) */
    CsrUint16 left;
    /* Called by CSR_UI to paint a particular pixel with a specified colour */
    void (*setPixel)(CsrInt16 x, CsrInt16 y, CsrUint32 colour);
    /* Called by CSR_UI to paint a range of pixels with the specified colours */
    void (*drawBox)(CsrInt16 left, CsrInt16 top, CsrInt16 right, CsrInt16 bottom,
CsrUint32 outlineColor, CsrUint32 fillColor, CsrBool fill);
    /* Called by CSR_UI before a redraw of the display is started */
    void (*lock)(void);
    /* Called by CSR_UI after a redraw of the display completes */
    void (*unlock)(void);
} CsrUiGraphicsRenderer;
```

void CsrUiTextRendererConfigurationGet(CsrUiTextRenderer *renderer);

Description:

Called by CSR_UI during initialisation to retrieve information about the Text Renderer interface. This function is not implemented by CSR_UI, and shall be provided by some external entity. If text rendering is desired, the fields of the CsrUiTextRenderer struct described below must be filled with appropriate values. If no text rendering is desired, the active field is set to FALSE, and the remaining fields can be left unassigned.

```
typedef struct
{
    /* Set to TRUE to activate the Graphics Renderer */
    CsrBool active;
    /* Called by CSR_UI to update the text display with new strings */
    void (*setText)(CsrUint16 *status, CsrUint16 *canvas, CsrUint16 *softkey1,
CsrUint16 *softkey2);
} CsrUiTextRenderer;
```

Icons

The following icons are defined in the csr_ui_icon_index.h file:

```
#define CSR_UI_ICON_NONE 0
#define CSR_UI_ICON_EMPTY 1
#define CSR_UI_ICON_BELL 2
#define CSR_UI_ICON_BLUETOOTH 3
#define CSR_UI_ICON_BOOK 4
#define CSR_UI_ICON_BOX 5
#define CSR_UI_ICON_BRICK 6
#define CSR_UI_ICON_CALCULATOR 7
#define CSR_UI_ICON_CALENDAR 8
#define CSR_UI_ICON_CAMCORDER 9
#define CSR_UI_ICON_CAMERA 10
#define CSR_UI_ICON_CAR 11
#define CSR_UI_ICON_CD 12
#define CSR_UI_ICON_CD_MUSIC 13
#define CSR_UI_ICON_CHECKBOX_CLEAR 14
#define CSR_UI_ICON_CHECKBOX_SET 15
#define CSR_UI_ICON_CLOCK 16
#define CSR_UI_ICON_COMPUTER 17
#define CSR_UI_ICON_COMPUTER_LAPTOP 18
#define CSR_UI_ICON_CONNECT 19
#define CSR_UI_ICON_CROSS 20
#define CSR_UI_ICON_DEVICE 21
#define CSR_UI_ICON_DISCONNECT 22
#define CSR_UI_ICON_DRIVE 23
#define CSR_UI_ICON_DRIVE_CD 24
#define CSR_UI_ICON_DUN_GATEWAY 25
#define CSR_UI_ICON_ERROR 26
#define CSR_UI_ICON_EYE 27
#define CSR_UI_ICON_FILMSTRIP 28
#define CSR_UI_ICON_FOLDER 29
#define CSR_UI_ICON_FOLDER_EXPLORE 30
#define CSR_UI_ICON_FOLDER_OPEN 31
#define CSR_UI_ICON_FOLDER_TRANSFER 32
#define CSR_UI_ICON_GAMEPAD 33
#define CSR_UI_ICON_GEAR 34
#define CSR_UI_ICON_HEADPHONES 35
#define CSR_UI_ICON_HEADSET 36
#define CSR_UI_ICON_HEADSET_EARPLUG 37
#define CSR_UI_ICON_HOURGLASS 38
#define CSR_UI_ICON_IPOD 39
#define CSR_UI_ICON_JOYSTICK 40
#define CSR_UI_ICON_KEY 41
#define CSR_UI_ICON_KEYBOARD 42
#define CSR_UI_ICON_LETTER 43
#define CSR_UI_ICON_LOCK 44
#define CSR_UI_ICON_LOCK_OPEN 45
#define CSR_UI_ICON_LOUDSPEAKER 46
#define CSR_UI_ICON_MAGNIFIER 47
#define CSR_UI_ICON_MARK_ACCEPT 48
#define CSR_UI_ICON_MARK_ADD 49
#define CSR_UI_ICON_MARK_CANCEL 50
#define CSR_UI_ICON_MARK_DELETE 51
#define CSR_UI_ICON_MARK_EXCLAMATION 52
#define CSR_UI_ICON_MARK_HELP 53
#define CSR_UI_ICON_MARK_INFORMATION 54
#define CSR_UI_ICON_MARK_NUMBER1 55
#define CSR_UI_ICON_MARK_NUMBER2 56
#define CSR_UI_ICON_MARK_SHUTDOWN 57
#define CSR_UI_ICON_MICROPHONE 58
#define CSR_UI_ICON_MONITOR 59
```

```
#define CSR_UI_ICON_MOUSE 60
#define CSR_UI_ICON_MUSIC 61
#define CSR_UI_ICON_NETWORK_AP 62
#define CSR_UI_ICON_PDA 63
#define CSR_UI_ICON_PHONE 64
#define CSR_UI_ICON_PICTURE 65
#define CSR_UI_ICON_PRINTER 66
#define CSR_UI_ICON_RADIOBUTTON_CLEAR 67
#define CSR_UI_ICON_RADIOBUTTON_SET 68
#define CSR_UI_ICON_REMOTECONTROL 69
#define CSR_UI_ICON_SCANNER 70
#define CSR_UI_ICON_SERVER 71
#define CSR_UI_ICON_SERVER_CONNECT 72
#define CSR_UI_ICON_SOUND 73
#define CSR_UI_ICON_SPYGLASS 74
#define CSR_UI_ICON_TAG 75
#define CSR_UI_ICON_TELEPHONE 76
#define CSR_UI_ICON_TELEVISION 77
#define CSR_UI_ICON_TEXT_SIGNATURE 78
#define CSR_UI_ICON_TEXT_WRITE 79
#define CSR_UI_ICON_THUMB_DOWN 80
#define CSR_UI_ICON_THUMB_UP 81
#define CSR_UI_ICON_TOOLS 82
#define CSR_UI_ICON_VCARD 83
#define CSR_UI_ICON_WEBCAM 84
#define CSR_UI_ICON_WORLD 85
#define CSR_UI_ICON_WORLD_RING 86
#define CSR_UI_ICON_WORLD_SEARCH 87
```

Key Codes

The key codes for each key on the UI Emulator are defined in csr_ui_keycode.h and are as follows:

Input Keys:

```
#define CSR_UI_KEY_N0          0x8000
#define CSR_UI_KEY_N1          0x8001
#define CSR_UI_KEY_N2          0x8002
#define CSR_UI_KEY_N3          0x8003
#define CSR_UI_KEY_N4          0x8004
#define CSR_UI_KEY_N5          0x8005
#define CSR_UI_KEY_N6          0x8006
#define CSR_UI_KEY_N7          0x8007
#define CSR_UI_KEY_N8          0x8008
#define CSR_UI_KEY_N9          0x8009
#define CSR_UI_KEY_STAR        0x800A
#define CSR_UI_KEY_HASH        0x800B
```

Control Keys:

```
#define CSR_UI_KEY_DEL         0x800C
#define CSR_UI_KEY_BACK        0x800D
#define CSR_UI_KEY_SK1         0x800E
#define CSR_UI_KEY_SK2         0x800F
#define CSR_UI_KEY_LEFT        0x8010
#define CSR_UI_KEY_RIGHT       0x8020
#define CSR_UI_KEY_UP          0x8040
#define CSR_UI_KEY_DOWN        0x8080
```

ASCII Keys:

```
#define CSR_UI_KEY_ASCII(code) ((CsrUint16) (0x8100 | (code) & 0x7F))
```

The valid range for the code parameter of the CSR_UI_KEY_ASCII definition is 0x20-0x7E.

For UIEs that allow user input, the input keys (N0 to N9, STAR and HASH) are mapped to characters according to the specified key map. The Idle Screen always uses the CONTROLNUMERIC key mapping.

NUMERIC key mapping

CSR_UI_KEY_N0	'0'
CSR_UI_KEY_N1	'1'
CSR_UI_KEY_N2	'2'
CSR_UI_KEY_N3	'3'
CSR_UI_KEY_N4	'4'
CSR_UI_KEY_N5	'5'
CSR_UI_KEY_N6	'6'
CSR_UI_KEY_N7	'7'
CSR_UI_KEY_N8	'8'
CSR_UI_KEY_N9	'9'
CSR_UI_KEY_STAR	NULL
CSR_UI_KEY_HASH	NULL

CONTROLNUMERIC key mapping

CSR_UI_KEY_N0	'0' '+'
CSR_UI_KEY_N1	'1'
CSR_UI_KEY_N2	'2'
CSR_UI_KEY_N3	'3'
CSR_UI_KEY_N4	'4'
CSR_UI_KEY_N5	'5'
CSR_UI_KEY_N6	'6'
CSR_UI_KEY_N7	'7'
CSR_UI_KEY_N8	'8'
CSR_UI_KEY_N9	'9'
CSR_UI_KEY_STAR	'*'
CSR_UI_KEY_HASH	'#'

ALPHANUMERIC key mapping

CSR_UI_KEY_N0	'+' '0'
CSR_UI_KEY_N1	'.' ',' '-' '?' '!' '1'
CSR_UI_KEY_N2	'a' 'b' 'c' '2'
CSR_UI_KEY_N3	'd' 'e' 'f' '3'
CSR_UI_KEY_N4	'g' 'h' 'i' '4'
CSR_UI_KEY_N5	'j' 'k' 'l' '5'
CSR_UI_KEY_N6	'm' 'n' 'o' '6'
CSR_UI_KEY_N7	'p' 'q' 'r' 's' '7'
CSR_UI_KEY_N8	't' 'u' 'v' '8'
CSR_UI_KEY_N9	'w' 'x' 'y' 'z' '9'
CSR_UI_KEY_STAR	Special Function[1]
CSR_UI_KEY_HASH	' ' '\n' '#' '*'

[1] This key will act as a Caps Lock that switches between lower case and upper case.

6 Document References

Terms and Definitions

BlueCore™	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
Bluetooth SIG	Bluetooth Special Interest Group
CSR	Cambridge Silicon Radio
DSI	Display Stack Item
CSR_UI	UI Emulator
UIDA	User Interface Drawing Area
UIE	User Interface Element

Document History

Revision	Date	History
1	09 JUN 09	Ready for release.
2	30 NOV 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	OCT 2010	Ready for release 2.2.0
5	DEC 2010	Ready for release 3.0.0
6	Aug 2011	Ready for release 3.1.0

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

No statements or representations in this document are to be construed as advertising, marketing, or offering for sale in the United States imported covered products subject to the Cease and Desist Order issued by the U.S. International Trade Commission in its Investigation No. 337-TA-602. Such products include SiRFstarIII™ chips that operate with SiRF software that supports SiRFInstantFix™, and/or SiRFLoc® servers, or contains SyncFreeNav functionality.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.