How-to Guide

Mobile Client Technology

How To… Add Tablet PC specific UI Controls to Tiles

Version 1.00 – August 2007

Applicable Release: CRM Mobile Technology 4.0 and 5.0

1 Introduction

Controls are user interface elements added to a tile to enable a user to interact with a Mobile Client Application during runtime. Controls can be bound to data sources or can be unbound.

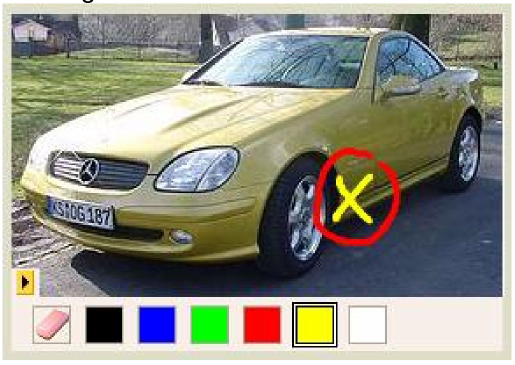
- **Bound control**: A Bound control is associated with the property of a business object or business query and displays the value of that property in the tile.
  For example, the *Name field* control is associated with the *Name property* of the *Customer* business object. At runtime, the field displays the customer name.
- **UnBound control**: The UnBound control is not associated with a data source. It is used to manipulate the user interface independently of the underlying data logic. UnBound controls are associated with pushbuttons and the pushbutton label indicates the action performed when chosen.
  For example, button controls are always UnBound controls.

Controls include checkboxes, radio buttons, and text input fields that allow the user to enter a search term, start a search, or read data displayed on a screen. You write event handlers to define the behavior of a control when it is activated.
This document describes how to add/modify various controls for use with a Tablet PC. You will learn how to:

- Enable *Image* control with Ink collection
- Use *Image* control as *Signature* control
- Add an *InkPopup* control
- Add a *StickyNote*

2 Implementing InkImage

An *Image* control is used to add images or pictures in a tile. *InkImage* can be used to enable *Image* control with Ink collection. You can click on the icon at the bottom of the image to display the toolbar. You can then select a color or eraser tool and make drawings on the picture. The drawings can be saved in the database and displayed at a later stage. See the figure below.



The image file must be one of the HTML <img> element supported files:

- Graphic Interchange Format (GIF)
- JPEG File Interchange Format (JPG)
- Windows Bitmap Format (BMP)

The format used to store ink is fortified GIF. This format is supported by the Microsoft ink controls. It contains the GIF image of the ink and the ink description at the end of the GIF file.

For example, the ink information contains the size of the initial image. If this image is displayed in a different size in another tile, the ink is stretched according to the new size.

2.1 Procedure

MSA 4.0 introduced some new runtime properties for *Image* control that allows using it as *InkImage*. If you set mode property to *InkImage*, *Image* control is substituted by *InkImage* control at runtime.
Application developers should additionally modal saving/loading *InkImage*.

Normally you would do the following steps:

1. At design time, put an *UnBound Image* control on a tile
2. At runtime, set mode property to *InkImage* in *mCore_onLoad*
3. At runtime, set *displayMode* property (fixedSize, originalSize, aspectRatio)
4. Load existing ink from binary attribute of BO

```
Private Sub mCore_onLoad() Handles mCore.onLoad

        'Other code here.

        'use image as signature
         ctrlNorImage.mode = inkImage

        'set displayMode. Apsect ratio is recommended. For
compatibility, fixedSize mode is default.
         ctrlNorImage.displayMode = aspectRatio

        'get ink from database
         Dim ink() As Byte = boAttachment.GetAttribute ("MyAttr")

        'assign to control
         ctrlNorImage.inkContent = ink

        'set image
         ctrlNorImage.value = 'C:\img.gif'
end sub
```

Ink in array is displayed in control. Control is enabled for additional inking.

5. Implement saving the ink in *onValueChanged* event. Use *inkContent* property to retrieve Ink.

```
Private Sub ctrlNorImage_onValueChanged2(boundAttributeChanged As
Boolean)
        'retrieve ink from Signature control
         Dim ink() As Byte = ctrlNorImage.inkContent
         If UBound(ink) >0 Then
        'Write ink to datbase.
         boAttachment.setAttribute("MyAttr", ink)
    End If
End Sub
```

6. You can implement clearing the *InkImage* (Optional).

```
        'Clears the inkImage.
         Dim uni As New System.text.UnicodeEncoding
         ctrlNorImage.inkContent = = uni.GetBytes("")
```

Note:
Because we store fortified GIF, ink will be visible to the user, but it will not be editable. And in case the image size changes, the quality will not be as good as on Tablet platform, because in this case we will stretch GIF, not ink.

2.2 Related CoreCtrlImage API

Public Property Let mode (ByVal mode As ImageBehaviorMode)

- Mode property defines behavior of *Image* control.
- Default value is `ImageBehaviorMode.normalImage`.
- In default mode *Image* control works like the *Image* control (the same as in 30).
- If you set mode = `ImageBehaviorMode.signature`, then at runtime Image is substituted with *Signature* control.
- If you set mode = `ImageBehaviorMode.inkImage` , then at runtime image is substituted with *InkImage* control.

Public Property Let inkContent (ByRef ink() As Byte)
Using this property you can pass array of bytes with ink to *Signature* control. Control will display it. Inking on *InkImage* control is always enabled.

Public Property Get inkContent() As Byte()
Using this property you can extract content of *InkImage* control and store it in the database.

Public Event onValueChanged2 (ByVal boundAttributeChanged As Boolean)
This event is raised when stroke is made. You should implement handler for this event. In the event handler you should write ink in BO attribute. This method is called on every stroke. If this operation is expensive, application developer can write in BO attribute only when the first time event is raised (to enable Save button) and to write finally in *BeforeSave* event for rest of the strokes.

Public Property Let value (theValue As Variant)
This property is used to set picture for *InkImage* control.

Public Property Let displayMode (ByVal mode As UFCore.ImageDisplayMode)

- *DisplayMode* property defines the way *Image* control is displayed.
- Default value is `ImageBehaviorMode.fixedSize` to keep compatibility with pervious versions.
- In *fixedSize* mode image will be displayed in position modeled in workbench, this is the image will be stretched to fit it.
- You can set *displaySize* to `ImageBehaviorMode.aspectRatio` (recommended). In this case image will be displayed inside bounds modeled by workbench, but keeping aspect ratio.
- You can set *displaySize* to `ImageBehaviorMode.originalSize` and image will be displayed in original size.

2.3 Miscellaneous Information

Image displayed in different size in a different tile

Ink information contains the size of the initial image. So when an image is displayed in a different size in a different tile ink will be stretched accordingly to the new size.

Format used to store ink

The format used to store ink is fortified GIF. This format is supported by Microsoft ink controls. It contains Gid image of ink and additionally ink description at the end of GIF file.

Ink displayed on non-Tablet platform

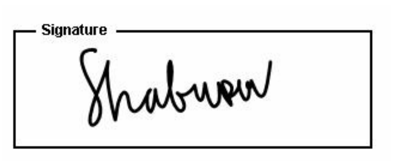
Ink will be visible to the user, but it will not be editable because we store fortified GIF. And in case the image size is changed, the quality will not be as good as on Tablet platform, because in this case we will stretch GIF and not the link.

Supported formats for Background file

GIF, JPEG, BMP

3 Using Signature Control

Signature control can be used to enter signature and for visual comparison of signature with sample ones. This is a very simple implementation of the signature. We just collect ink on a control area and raise event to application developers that related sales document was signed and the fields present should be locked. No programmatic verification is implemented.



3.1 Procedure

In MSA 4.0 release we introduced some new runtime properties for *Image* control that allows using it as a *Signature*. If you set mode property to *Signature*, *Image* control is substituted by *Signature* control at runtime.
Application developers should additionally modal saving/loading signature and locking field.

The following steps are followed:

1. Put an *UnBound Image* control on a tile.
2. Create border around it () and add label *Signature* to improve the appearance.
3. Set mode property to *Signature* in `mCore_onLoad`
4. Load existing ink signature from binary attribute of BO (currently *BOATTACHMENT* can be used to store ink)

```
Private Sub mCore_onLoad()

         ‘Other code here.

         ‘use image as signature e
          ctrlNorImage.mode = signature

         ‘get ink from database
          Dim ink() As Byte
          ink = boAttachment.GetAttribute ("PackedFile")

         ‘assign to control
          ctrlNorImage.inkContent = ink

         ‘optional, set background: ctrlNorImage.value =
                                 "c:\signature_background.jpg"
End sub
```

If ink array has signature, it is displayed in control. Sales document fields are locked this case. If ink array is empty, then signature is enabled for signing.

5. Implement saving the signature in *onValueChanged* event. Use *inkContent* property to retrieve Ink.

```
 Private Sub ctrlNorImage_onValueChanged2(boundAttributeChanged As
Boolean)

        ‘retrieve ink from Signature control
         Dim ink() As Byte
         ink = ctrlNorImage.inkContent

         If UBound(ink) >0 Then
        ‘Write ink to databse.
         boAttachment.setAttribute(“PackedFile”, ink)
   End If

        ‘Lock Sales Document and UI fields. (You will write the appropriate
code here)
         mCore.setReadOnly True
End Sub
```

6. You can clear the Signature by (optional):

```
       ‘Clears the signature and allows new signing.
        ctrlNorImage.inkContent = ""

       ‘Unlock Sales document and Ui fields.(You will write the appropriate
code here)
        mCore.setReadOnly False
```

3.2 Related CoreCtrlImage API

Public Property Let mode(ByVal mode As ImageBehaviorMode)

- Mode property defines behavior of *Image* control.
- Default value is `ImageBehaviorMode.normalImage`.
- In default mode *Image* control works like the *Image* control (the same as in 30).
- If you set mode = `ImageBehaviorMode.signature`, then at runtime image is substituted with *Signature* control.
- If you set mode = `ImageBehaviorMode.inkImage` , then at runtime image is substituted with *InkImage* control.

Public Property Let inkContent(ByRef ink() As Byte)
You can use this property to pass array of bytes with signature to *Signature* control. The control will display it. If ink is non-empty, then inking is disabled (you can not write on the control, which already displays signature). If ink is empty, inking on *Signature* control is enabled.

Public Property Get inkContent() As Byte()
Using this property you can extract content of *Signature* control and store it in database.

Public Property Let inkSignatureContent (ByRef ink() As Byte)
See inkContent. The only difference is that it assigns array, that is BMP picture.

Public Property Get inkSignatureContent () As Byte()
See inkContent. The only difference is that it retrieves array, that is BMP picture.

Public Event onValueChanged2(ByVal boundAttributeChanged As Boolean)
This event is raised when signature is made. You should implement handler for this event. In the event handler you should store signature in database, lock Sales document.

Public Property Let value(theValue As Variant)
This property optionally can be used to set background picture for *Signature* control. (if you want to have it on different background than the tile background).

3.3 Miscellaneous Information

What happens if the user signs document?

On the control level: After time from last made stroke exceeds 2 seconds, control gets locked and raises event *OnValueChanged2*.

On application level: Application receives event *onValueChanged* and in event handler retrieves ink and stores it to *bo.attribute*. It also locks sales document and all related fields in UI part. Application developer can add encryption on received ink. Application developer can also take sales document fields and store it together with signature.

Supported formats for Background file

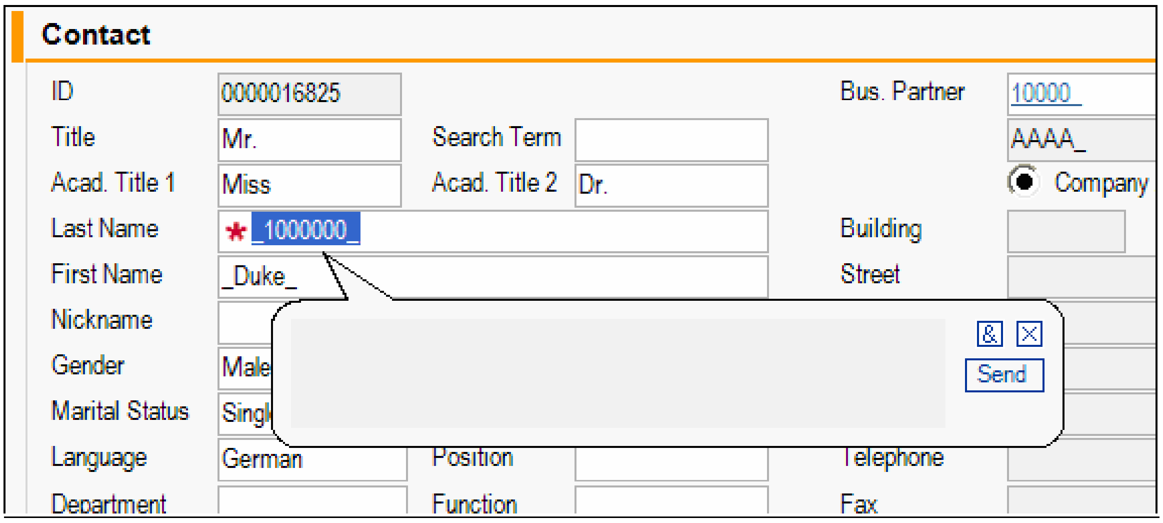
GIF, JPEG, BMP

Running on Tablet PC

The application can find out if the system is a Tablet PC key or not via the Tablet PC key in gServices. If `gServices.settings("TabletPC")=1`, you are running on a Tablet PC device and the *Image* control can be changed into the signature mode. If not the change to the signature mode will do nothing.

4 Adding an InkPopup Control

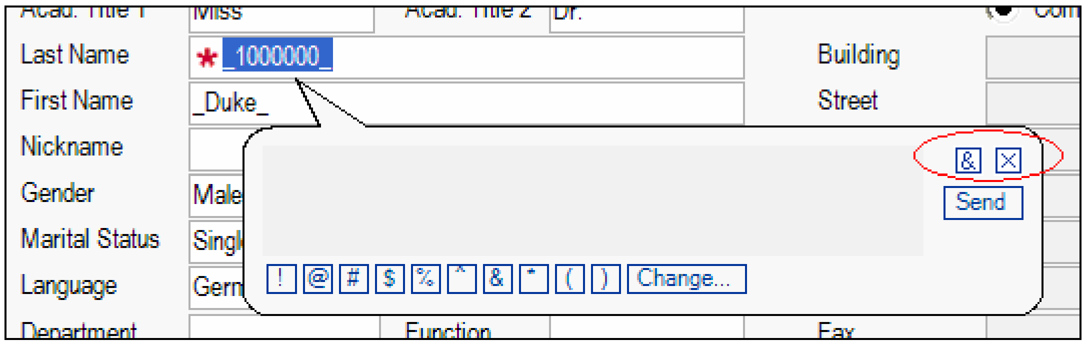
InkPopup is a control which enables users to enter data into the Mobile Client Application using a Tablet PC pen. The data will be entered as a handwritten text, which will be converted into usual text data and inserted in the field from which the popup control was opened. The *InkPopup* control itself looks like a popup window having a nice balloon style:



Contact

ID	0000016825			Bus. Partner	10000
Title	Mr.	Search Term			AAAA_
Acad. Title 1	Miss	Acad. Title 2	Dr.		● Company
Last Name	* _1000000_			Building	
First Name	_Duke_			Street	
Nickname					
Gender	Male				
Marital Status	Singl				
Language	German	Position		Telephone	
Department		Function		Fax	

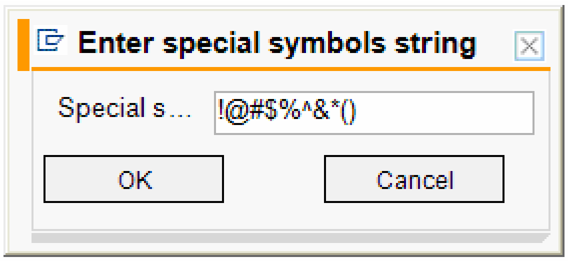
The *InkPopup* control automatically appears when the user clicks in the field where he can enter text data.
The user can enter handwritten text in the highlighted field in the center of the *InkPopup*. Then he can press *Send* button to cause this input to be converted into the text data and inserted into the corresponding field. He can close the popup with the *close* button, or he can press *special symbols* button (Text: '&': on the left side of close button) to get an area where he can select to enter some special symbols, which are normally hard to enter with a Tablet PC pen.



Acad. Title 1	Miss	Acad. Title 2	Dr.		Com
Last Name	* _1000000_			Building	
First Name	_Duke_			Street	
Nickname					
Gender	Male				
Marital Status	Singl				
Language	Germ				
Department		Function		Fax	

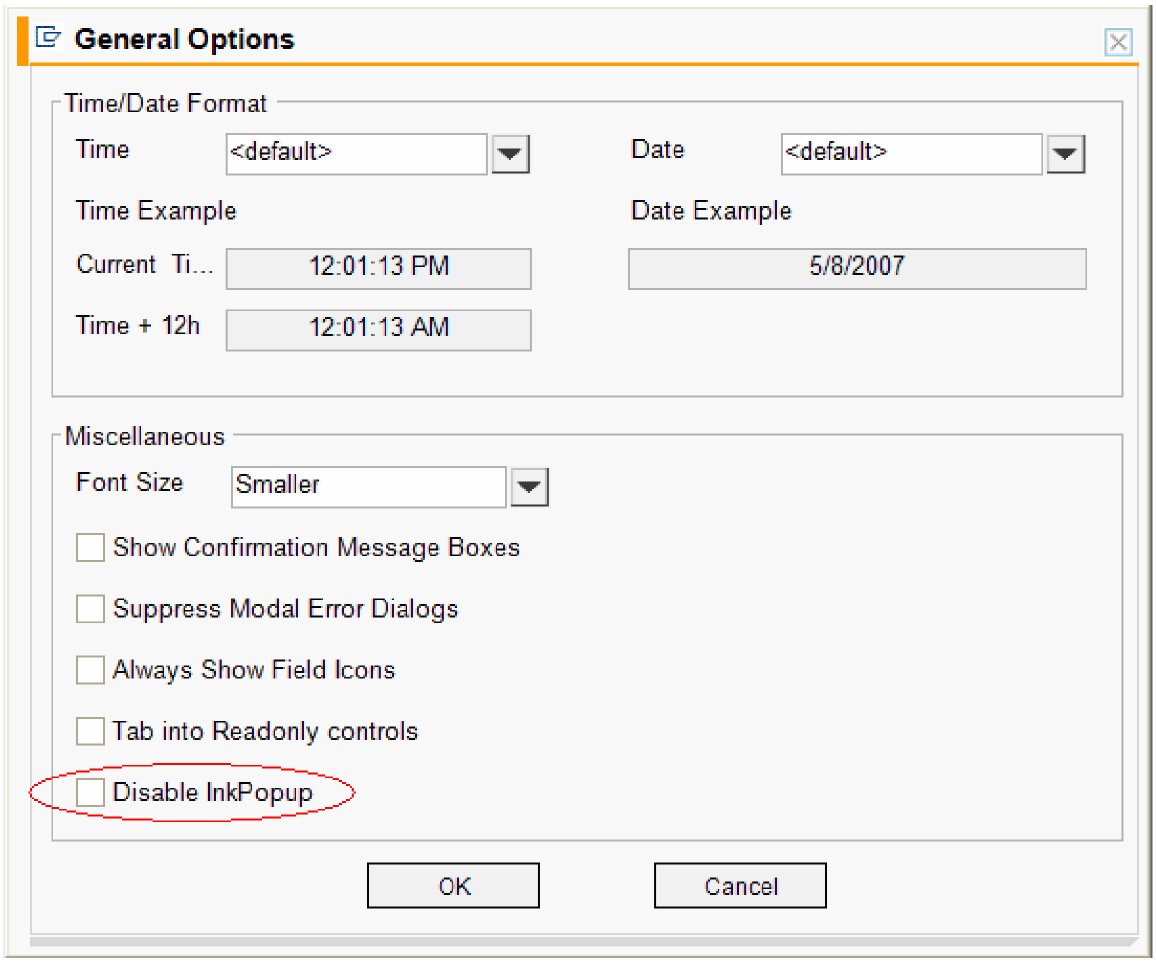
The user choose which special symbols to display by clicking on the button *Change.* This will open a new dialog with the current special symbols. Users can then enter any special character.

Note:
These changes will be specific to each mobile user.



Special s... !@#$%^&*()

If the user writes something in the input field and then waits for a couple of seconds, the handwritten text will be automatically converted and inserted.
The user can completely disable showing of *InkPopup* every time he clicks in a field choosing the *Disable InkPopup* option in General Options dialog.



Time <default>

Date <default>

Current Ti... 12:01:13 PM

5/8/2007

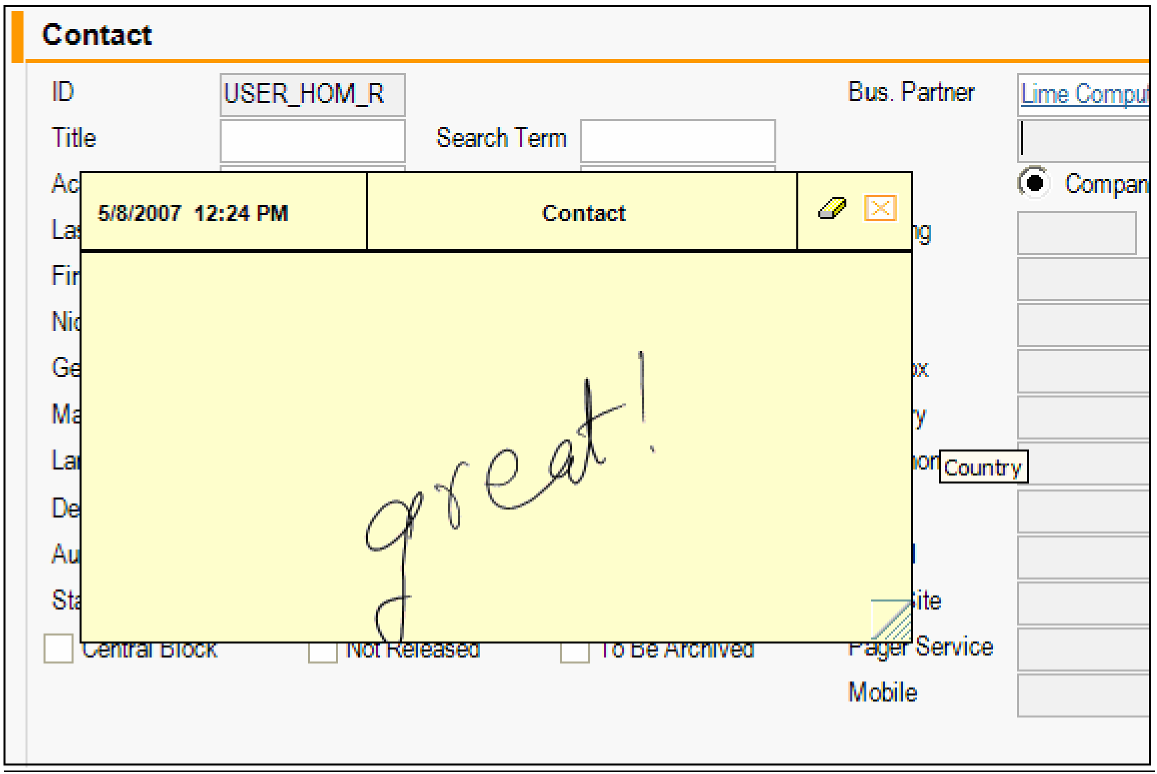
Time + 12h 12:01:13 AM

Font Size Smaller

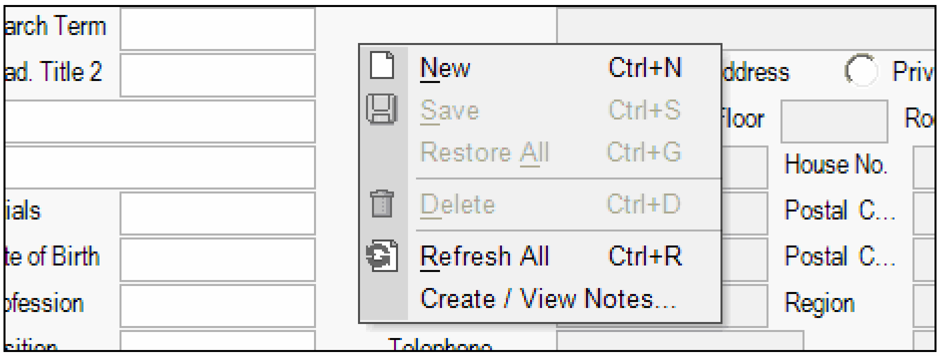
- [ ] Show Confirmation Message Boxes
- [ ] Suppress Modal Error Dialogs
- [ ] Always Show Field Icons
- [ ] Tab into Readonly controls
- [ ] Disable InkPopup

5 Adding a StickyNote

StickyNote (further will be referred as Note) is a control which is very similar to MS Outlook notes, which provides a possibility for the user working on a Tablet PC to enter hand-written text or drawings. The typical look-and-feel is shown on the figure below.



The user can create one note for the particular BO. To do that, the user can select *Create Note* option from the tile context menu. This option is available only on detail tiles, and only if there is a BO on the tile.



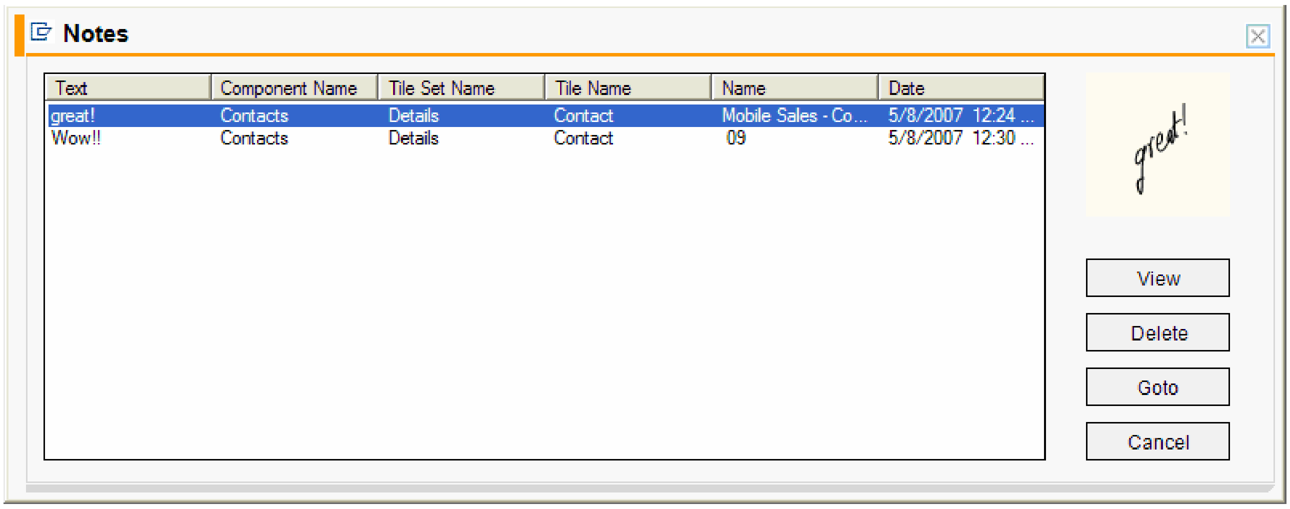
After the note window is opened, the user can enter text or draw, he can also move and resize the note window. The content and the position of the note will be saved if the user closes the note or if the user navigates to different BO or tileset.

If a note already exists for a particular BO, then the note button is visible in the tile caption at the right, near by the maximize button (as shown on the picture). Using this button the user can open and close the note.

If the note is opened and the user navigates to another BO, the note window remains opened only if the other BO also has a note. In this case the content and the position of the note will be updated according to the note of the current BO.

5.1 Managing the list of all notes in the application

The user can open a dialog with the list of all entered notes in the application, selecting *Notes…* option from the *Window* menu.



Notes

Text	Component Name	Tile Set Name	Tile Name	Name	Date
great!	Contacts	Details	Contact	Mobile Sales - Co...	5/8/2007 12:24 ...
Wow!!	Contacts	Details	Contact	09	5/8/2007 12:30 ...

The columns in this dialog contain the following information:

- recognized text of the note
- name of the component, where the note is created
- name of the tileset, where the note is created
- name of the tile, where the note is created
- name of the BO, with which the note is associated
- date of last update of the note

The dialog also contains a small preview window.

The following actions are possible in this dialog:

- *View* – Upon pressing this button, the user gets note window opened and can view and modify the content and the position of the note
- *Delete* – Deletes the selected note
- *Goto* – Using this button the user can navigate directly to the Component/Tileset/Tile/BO, where it has been created to open the note.