

Implementing Specialized Data Capture Applications with InVision Development Tools (Part 3)

[This is the third of a series of white papers on implementing applications with special requirements for data capture such as barcodes, signatures, and magnetic stripes. This white paper is focuses on signature capture.]

In Part 1 and 2 of this series, we developed a basic bar-coded data capture application for a handheld device that captures data to an MS-Access inventory database for items received by a receiving clerk. In this paper, we will explore capturing signatures directly from web page.

Signature Capture

Adding signature capture opens a new world of security, accountability, and history to an application supporting data capture. Many industries require user identification at the point of transaction.

In the healthcare industry, a handheld application that also collects electronic signatures from practitioner and patient makes it possible to offer patient confirmation, practitioner accountability, and accurate history in a secure package. Where every action must be recorded with proper documentation and traceability, the availability of electronic signature capture offers security and reassurance to patient and professional alike.

Patient Safety

InVision has completed a patient safety application that can be deployed in hospitals to assure the right medication is delivered to the right patient, at the right dosage, at the right time, and by the right route (the "5" rights). This system takes orders from a central computer and allows hospital personnel to record the details of every medication administration event. It compares patient wrist bands, personnel badges, and medication barcodes with the order and displays and logs errors that may occur.

This system is dependent on approved orders received by the central system from a separate order-entry application. "Stat" orders are an impediment to timely delivery of medication especially when they occur at night, when physicians are less likely to be available at the hospital to approve an order. In this situation, staff usually phone the physician for a voice approval. However, with HIPAA regulations bearing down heavily on healthcare systems, a system that documents the approval of such orders greatly improves on security and accountability.

"Approve Medication Order"

We're going to write a simple application that assumes that a request for a medication order has been taken by the nurse and is waiting in the order entry system's database to be approved by the patient's physician. Perhaps the physician can change some of the details of the order, but we will keep it simple. We will use InVision's **iSignIt™** control to capture the physician's signature and store it back in the database.

Furthermore, we are going to host the control in a web page and use some of its specialized properties and methods:

- **SIGData**. This is a special string representation of the captured signature that may be played back in the control. The data is check summed, compressed, encrypted, and converted to an ASCII string.
- **LockSig()**. Locks the current signature and prevents any additional signing.
- **FitToWindow()**. When set to True, signatures that do not completely fit within the iSignIt control window are resized horizontally and/or vertically when signing is completed via a call to the LockSig method or when a signature is loaded from a file.

If the stylus is dragged outside the bounds of the iSignIt control while signing, iSignIt will continue to capture. When the LockSig method is subsequently called to end the signature capture session, any drawing that was done outside the bounds of the control are preserved and can be viewed by setting FitToWindow to True.

- **Clear()**. Clears the current information in the signature box.
- **ShowSignTime**. Sets or returns a value that determines whether or not the time stamp for a signature is displayed.
- **ISigWidth, ISigHeight**. Returns the width and height of the current signature in pixels.

Scan-Enabled Web Applications

One of the most exciting features of iSignIt is its ability to take advantage of allowing itself to be used in a web page. Using the standard OBJECT tag, we can set up all the parameters and actually get the control to render on a web page. The user then navigates with Pocket Explorer to the web page and can sign directly onto the iScanIt control window on a form.

We will use Visual Studio 2003 and .NET to create the web application and host the form and some other data entry fields.

Database

We're using MS SQL Server for our database server. In our table, "tblRXPendingOrders" we will display and/or update following fields:

PatientID	varchar	50
DisplayPatientFullName	varchar	100
DisplayDoctorFullName	varchar	100
OrderDate	varchar	50
Medication	varchar	50
Dosage	varchar	50
Route	varchar	50
ApprovalSignature	text	16

Notice that the Approval Signature is in "text" format. SIG-format data can be as large as 500 bytes, but still compares favorably to bitmaps, which on a 240 x 100

pixel capture control can easily start at 2000 bytes. The "text" SQL data type allows us to store data larger than 256 bytes.

Implementing "Approve Med Order"

We assume the reader has implemented at least one .NET ASP.NET web application with Visual Studio .NET. We are using a "Web Form" as our container for the iSignIt control. Here's the form as displayed in the Visual Studio IDE:

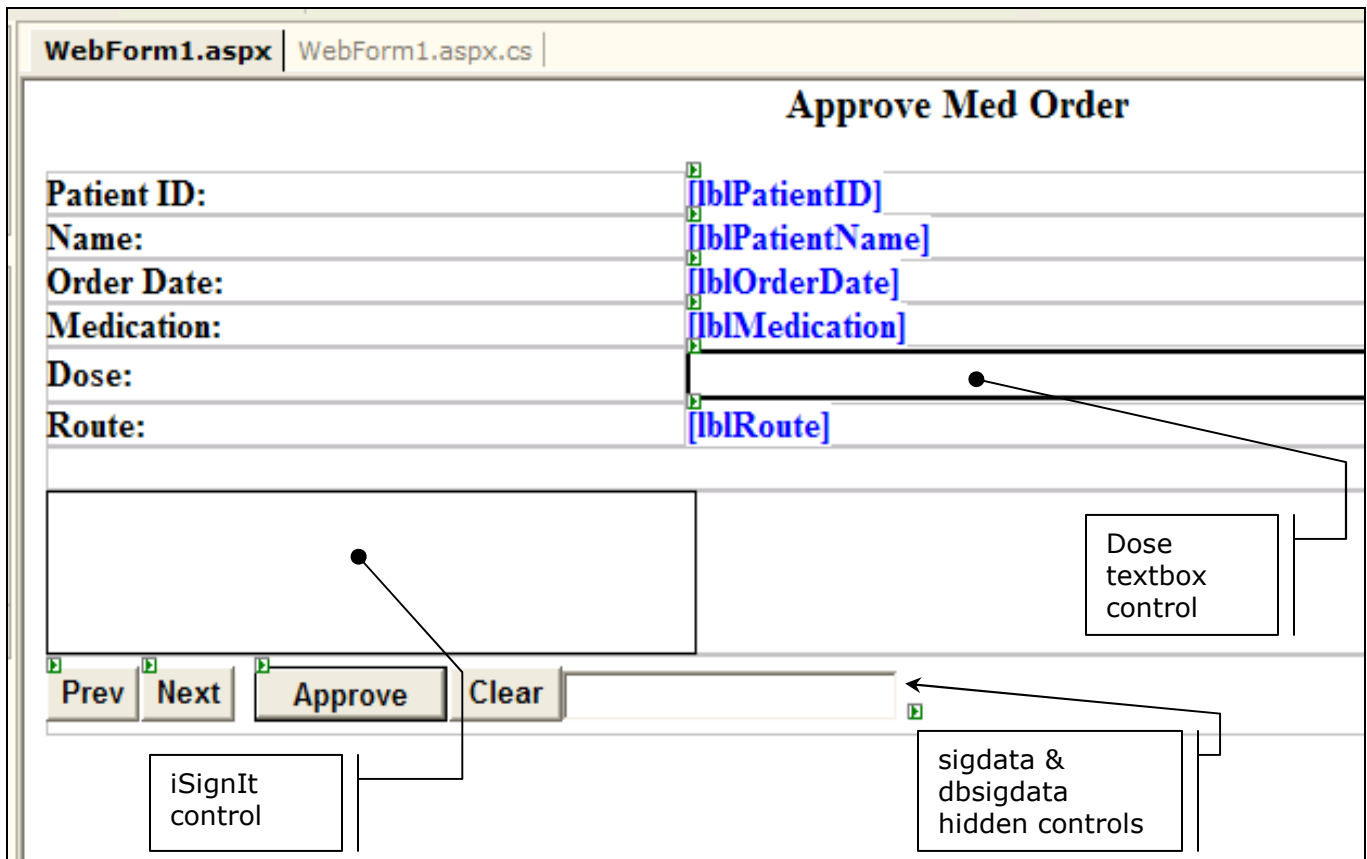


Figure 1 Visual Studio .NET Approve Med Order WebForm

The following web forms server controls exist within our form tag, all using a custom binding expression (i.e., our Patient ID label uses `DataBinder.Eval (dsRXPendingOrders, "Tables[tblRXPendingOrders].DefaultView.[0].PatientID"`):

- Patient ID label
- Patient Name label
- Order Date label
- Medication label
- Route Label
- Dose Textbox

- `dbsigdata` – this Textbox control is assigned a style with a “DISPLAY:NONE” attribute. *DO NOT USE* `visible=none`; it becomes invisible to client side scripting. It is bound to the “ApprovalSignature” field in the database so that it can provide the signature from a record just signed to the DHTML javascript.

Next and **Previous** buttons provide navigation backward and forward through the dataset.

The **Clear** button allows the physician to clear his signature on an form that hasn’t been approved yet.

The **Approval** button is a special case. Its “onclick” event is attached in code-behind (the element does not allow a tag in client code) so that we can test various properties of the `iSignIt` control in client script, but still continue with a post back.

iSignIt Control

As with `iScanIt`, the control is essentially a client-side control and must be installed and executed on the device. It is beyond the scope of this paper to detail methods of automatic downloading of controls, but this can be done.

Because the control executes within the context of the browser as an activeX control, we must provide techniques such as hidden form fields, so that its signature data can be transferred to and from server-side code-behind processing. This is why we employ the `dbsigdata` and `sigdata` hidden form fields.

We assume that `iSignIt` component has been installed or downloaded on the handheld device.

Data Access

We configured a `SQLDataAdapter` like so:

- a `SELECT` command to retrieve patient, medication data, and SIG data from the `ApprovalSignature` field. This command selects all the patients that do NOT have a signature in the `ApprovalSignature` field
- an `UPDATE` command to update the dosage and the SIG data

Rather than generate a typed dataset, we used an untyped dataset.

Code-Behind Page Processing

When the page loads, in the code-behind we instantiate and fill the dataset, add the onclick event to our Approve button, handle the control bindings, initialize our record pointer, and manage the button and dose enabled property.

```
private void Page_Load(object sender, System.EventArgs e)
{
    dsRXPendingOrders = new DataSet();
    sqlDataAdapter.Fill(dsRXPendingOrders);
    btnApprove.Attributes.Add("onclick", "Approve_Click()");

    if (!IsPostBack)
    {
        Session.Add("RecordPos", 0);
        lblPatientID.DataBind();
        lblPatientName.DataBind();
        lblMedication.DataBind();
        lblOrderDate.DataBind();
        lblRoute.DataBind();
        txtDose.DataBind();
        dbsigdata.DataBind();
    }

    int RecordPos = Convert.ToInt32(Session["RecordPos"]);

    if
(dsRXPendingOrders.Tables["tblRXPendingOrders"].Rows[RecordPos]["ApprovalSign
ature"].ToString() !=
        "")
    {
        txtDose.Enabled = false;
        btnApprove.Enabled = false;
    }
    else
    {
        txtDose.Enabled = true;
        btnApprove.Enabled = true;
    }
}
}
```

The Approve button code-behind stores the signature if one has been entered, otherwise it exits and stores any edits the physician may have done to the dosage. This is done via hidden form fields:

```
private void btnApprove_Click(object sender,
System.EventArgs e)
{
    string strSigData = Request.Form["sigdata"].ToString();

    if (strSigData == "")
        return;
}
```

```
        string strDose = Request.Form["txtDose"].ToString();

        int RecordPos = Convert.ToInt32(Session["RecordPos"]);

        dsRXPendingOrders.Tables["tblRXPendingOrders"].Rows[RecordPos]["ApprovalSignature"] =
            strSigData;

        dsRXPendingOrders.Tables["tblRXPendingOrders"].Rows[RecordPos]["Dosage"] =
            strDose;

        sqlDataAdapter.Update(dsRXPendingOrders);

        lblPatientID.DataBind();
        lblPatientName.DataBind();
        lblMedication.DataBind();
        lblOrderDate.DataBind();
        lblRoute.DataBind();
        txtDose.DataBind();
        dbsigdata.DataBind();

        if
(dsRXPendingOrders.Tables["tblRXPendingOrders"].Rows[RecordPos]["ApprovalSignature"].ToString() !=
        "")
        {
            txtDose.Enabled = false;
            btnApprove.Enabled = false;
        }
        else
        {
            txtDose.Enabled = true;
            btnApprove.Enabled = true;
        }
    }
}
```

Most of the rest of the code-behind handles the record positioning and control binding.

DHTML Page Processing

We need to write a few lines of JavaScript (Approve_Click(), init(), ClearSig()) to initialize the iSignIt control and to pre-process the signature before calling the code-behind.

The Approve_Click() function deserves some special mention. Originally, I was testing for a null string ("") in the SigData field to distinguish between a signed and an unsigned control. This is not feasible, as data in the SigData field is not valid until:

- after the LockSig method is called
- a previously captured signature is loaded via the Load method
- setting the SigData property with valid signature data.

It is a much better practice to use the ISigHeight and ISigWidth properties to determine if the user has ever touched the control's window. Whether the user has actually penned a signature or just accidentally touched the control is best left up to the application to determine. The point is that a minimum height and width could be established to filter out accidental stylus contact. I elected to test for 0 height and width.

Here's the HTML and scripting for the entire page:

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs" AutoEventWireup="false"
Inherits="WebiSignIt.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
  <HEAD>
    <title>WebForm1</title>
    <meta content="Microsoft Visual Studio .NET 7.1"
name="GENERATOR">
    <meta content="C#" name="CODE_LANGUAGE">
    <meta content="JavaScript" name="vs_defaultClientScript">
    <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
    <script language="javascript">
<!--
function Approve_Click()
{
  with (document.Form1)
  {
    if (iSignIt.lSigWidth == 0 && iSignIt.lSigHeight == 0)
    {
      alert ("You have not signed the approval");
      return;
    }

    iSignIt.LockSig();
    sigdata.value = iSignIt.SigData;
  }
}

function init()
```

```

{
    if (document.Form1.dbsigdata.value == "")
        document.Form1.iSignIt.Clear();

    else
        document.Form1.iSignIt.SigData = document.Form1.dbsigdata.value;
}

function ClearSig()
{
    document.Form1.iSignIt.Clear();
}

//-->
    </script>
</HEAD>
<body bottomMargin="0" topMargin="0" onload="init()">
    <form id="Form1" method="post" runat="server">
        <h3 style="FONT-WEIGHT: bold; TEXT-ALIGN:
center">Approve Med Order</h3>
        <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="100%" border="0">
            <TR>
                <TD align="left" width="35%" colSpan="1"
rowSpan="1"><STRONG>Patient ID: &nbsp;&nbsp;</STRONG></TD>
                <TD noWrap align="left" width="65%"><asp:label
id=lblPatientID runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].PatientID") %>' ForeColor="Blue">
                    </asp:label></TD>
            </TR>
            <TR>
                <TD align="left"
width="35%"><STRONG>Name:</STRONG>&nbsp;&nbsp;&nbsp;</TD>
                <TD align="left" width="65%"><asp:label
id=lblPatientName runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].DisplayPatientFullName") %>'
ForeColor="Blue">
                    </asp:label></TD>
            </TR>
            <TR>
                <TD align="left" width="35%"><STRONG>Order
Date:</STRONG>&nbsp;&nbsp;&nbsp;</TD>
                <TD noWrap align="left" width="65%"><asp:label
id=lblOrderDate runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].OrderDateinDateTime", "{0:g}")
%>' ForeColor="Blue">
                    </asp:label></TD>
            </TR>
            <TR>
                <TD align="left"
width="35%"><STRONG>Medication:</STRONG>&nbsp;&nbsp;&nbsp;</TD>

```

```

                <TD align="left" width="65%"><asp:label
id=lblMedication runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].Medication)" %>'
ForeColor="Blue">
                </asp:label></TD>
        </TR>
        <TR>
                <TD align="left"
width="35%"><STRONG>Dose:</STRONG>&nbsp;&nbsp;&nbsp;</TD>
                <td align="left" width="65%"><asp:textbox
id=txtDose runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].Dosage)" %>'
ForeColor="Transparent" Width="100%" BorderStyle="Solid">
                </asp:textbox></td>
        </TR>
        <TR>
                <TD align="left"
width="35%"><STRONG>Route:</STRONG>&nbsp;&nbsp;&nbsp;</TD>
                <TD align="left" width="65%"><asp:label
id=lblRoute runat="server" Font-Bold="True" Text='<%#
DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].Route)" %>' ForeColor="Blue">
                </asp:label></TD>
        </TR>
</TABLE>
<TABLE id="Table2" cellSpacing="0" cellPadding="0"
width="100%" align="left" border="0">
        <TR>
                <TD height="5"></TD>
        </TR>
        <TR>
                <TD valign="middle">
                        <OBJECT id="iSignIt" height="71"
width="300" classid="clsid:AFA576D0-E313-11D4-9337-00E029519C8D"
                                name="iSignIt" VIEWASTEXT>
                                <PARAM NAME="_cx" VALUE="7938">
                                <PARAM NAME="_cy" VALUE="1879">
                                <PARAM NAME="ForeColor"
VALUE="2147483656">
                                <PARAM NAME="BackColor"
VALUE="2147483653">
                                <PARAM NAME="BorderColor"
VALUE="2147483654">
                                <PARAM NAME="BorderWidth"
VALUE="1">
                                <PARAM NAME="DrawWidth" VALUE="1">
                                <PARAM NAME="Enabled" VALUE="-1">
                                <PARAM NAME="SaveToFolder"
VALUE=".\">
                                <PARAM NAME="FitToWindow" VALUE="-
1">
                                <PARAM NAME="Rotation" VALUE="0">
                                <PARAM NAME="SigData" VALUE="">

```

```

VALUE="2">
VALUE="0">
VALUE="0">
1">
VALUE="1">
VALUE="1">
VALUE="0">
<PARAM NAME="ShowSigTime"
<PARAM NAME="bgFileName" VALUE="">
<PARAM NAME="bgOriginX" VALUE="0">
<PARAM NAME="bgOriginY" VALUE="0">
<PARAM NAME="bgRectWidth"
<PARAM NAME="bgRectHeight"
<PARAM NAME="bgFitToRect" VALUE="-
1">
<PARAM NAME="bgAlignHorz"
<PARAM NAME="bgAlignVert"
<PARAM NAME="bgRotation" VALUE="0">
<PARAM NAME="bmpBitCount"
<PARAM NAME="SigTimeLoc" VALUE="4">
</OBJECT>
</TD>
</TR>
<TR>
<TD vAlign="middle" noWrap colSpan="1"
height="40" rowspan="1">
<asp:button id="Button2" runat="server"
Font-Bold="True" Text="<Prev"></asp:button>
<asp:button id="btnNext" runat="server"
Font-Bold="True" Text="Next">></asp:button>&nbsp;
<asp:button id="btnApprove"
runat="server" Font-Bold="True" Text="Approve" Width="89px"
Height="24px"></asp:button>
<INPUT onclick="ClearSig()" style="FONT-
WEIGHT: bold" type="button" value="Clear"><input id="sigdata" style="DISPLAY:
none" type="hidden" name="sigdata">
<asp:textbox id=dbsigdata style="DISPLAY:
none" runat="server" Text='<%# DataBinder.Eval(dsRXPendingOrders,
"Tables[tblRXPendingOrders].DefaultView.[0].ApprovalSignature") %>'
Width="30px" Height="16px">
</asp:textbox></TD>
</TR>
</TABLE>
</form>
</body>
</HTML>

```

Try it!

After permissioning the web application and SQL Server permissions for your particular environment, visit the web page. You should see the following:

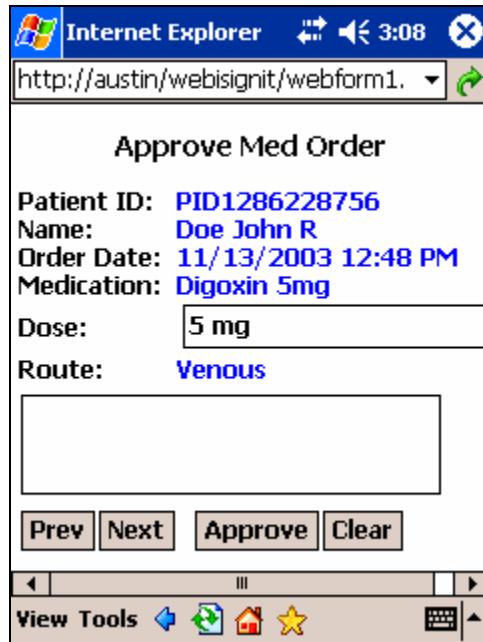


Figure 2 Webform Page in Pocket IE

Write your signature in the signature box, change the dosage if you wish, and hit Approve:

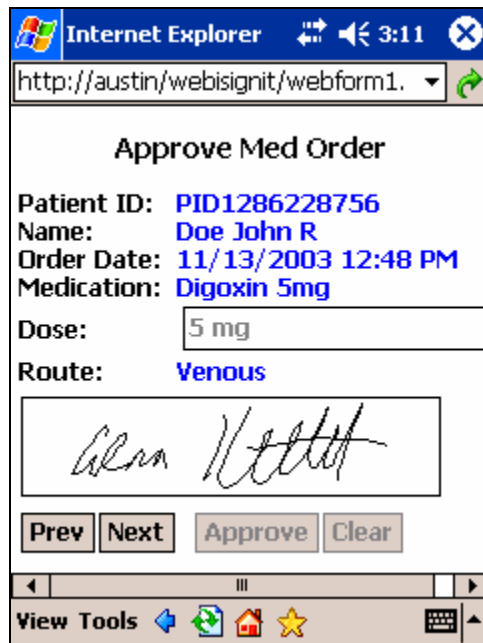


Figure 3 Approved Medication Record

The page will refresh with your signature so that you can verify you signed this record. Notice the Approve and Clear buttons become disabled. If you then tap

anywhere on the iSignIt window, it will tell you the date and time the signature was captured:

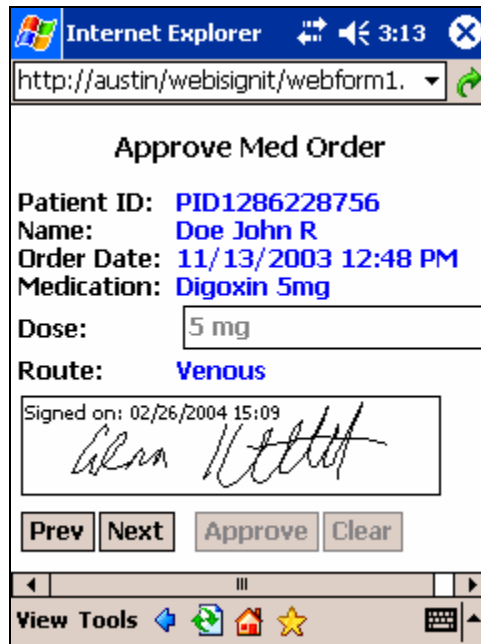


Figure 4 Approved Medication Record With Timestamp

That's it for now. Enjoy "i-Signing" all your applications!!!