



Effective ArcPad Customization

Bryce Smith - MapTel
Craig Greenwald - ESRI



Agenda

- The basics
- What's new for developers in ArcPad 7.0?
- Customizing the UI
- Controlling ArcPad Preferences
- Utility Extensions
- Inter-process communication
- Related tables and data-linked forms
- Integrating serial and Bluetooth devices
- Internet communication
- ArcPad OEM Tool



The Basics

- Not ArcObjects for Windows CE
- No embeddable components
- Active Scripting Host
 - VBS and JS runtimes provided by system
- File based (ASCII)
 - XML
 - VBScript / JScript



Types of Customizations

- User interface (toolbars, dialogs)
- Additional tools/functionality
- Enforce data integrity
- Create templates for data consistency
- Combine common tasks
- New data formats, protocols, projections, datum transformations, and devices (via Extensions)



ArcPad customization files

- Default configurations (ArcPad.apx)
- Layer definitions (shapefile.apl)
- Applets (applet.apa)
- VBScript files (vbscript.vbs)
- JScript files (jscript.js)
- Extensions (extension.dll)
- Other supporting ArcPad XML files
 - ArcPadPrefs.apx, ArcPadBookmarks.apx
 - .apm, .aph, .apg



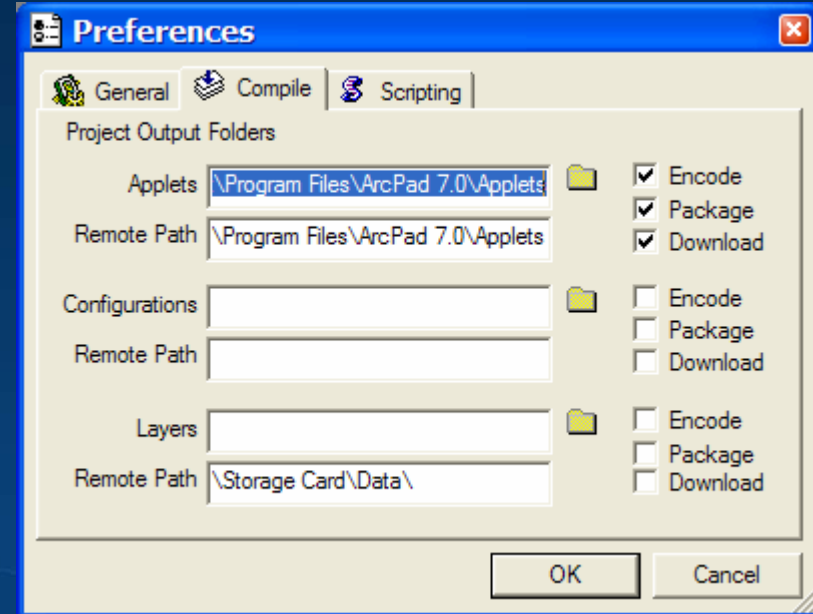
What's New for Developers in ArcPad 7.0?

- ArcPad Studio enhancements
 - Wizard for edit form creation
 - Raw XML editing
 - Min/Max validation supported without scripts
 - Auto-incrementing controls (bound, numeric)
 - Project deployment tools
 - Support for large buttons on toolbars
- JScript support (in addition to VBScript)
- Expanded Object Model
 - Rangefinder, FTP, Archive, Multimedia, Console, and Preferences objects
 - Dozens of new properties, methods, events
 - Many built-in constants
- New extension APIs
 - Cameras
 - Laser rangefinders
 - Projections and Datums
 - Edit notifications

ArcPad Studio - Preferences



- Ability to encode project file/s
- Package your project files into one file
- Download project files to your mobile device through ActiveSync
- Set your scripting language



Customizing the UI

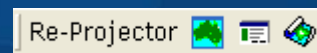
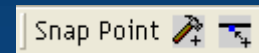
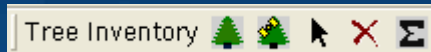
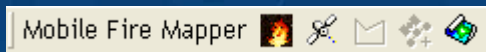


- Toolbars
 - Built-in toolbars can be removed (except Command Bar on the bottom of the screen)
 - Custom toolbars with built-in and custom tools can be added
- Forms
 - Data-linked forms (Edit, Identify)
 - General forms
- Statusbar
 - Exposed to the Object Model (read & write)

Toolbars



- Tools and sub menus
- All ArcPad commands are available
- Custom tools
- Captions and images



Scripting With Toolbars



- Programmatic access to built-in commands
 - `Application.ExecuteCommand("Command");`
- For custom tools - no distinction between a 'tool' and a 'command' like in ArcObjects. The behavior is based on which events you handle
 - `OnClick` → 'command' behavior
 - `OnPointerUp`, `OnPointerMove`, `OnPointerDown` → 'tool' behavior
 - Use `OnPointerUp` (`OnPointerDown` leaves artifacts on the screen)

Forms



- Edit, Identify, and General
- Edit and Identify forms only available for Layer Definition files
 - Controls can be linked to shapefile attributes
 - Provides a visual, “intelligent” representation of a particular feature’s attributes
- Always modal

A screenshot of a software dialog box titled "Fire Perimeter Form". The dialog has a blue title bar with a close button (X) on the right. Below the title bar is a tabbed interface with a tab labeled "Collection Method". To the right of the tab are navigation arrows and a "Cc" label. The main area contains several controls: a "Collection Method" section with radio buttons for "GPS" and "Digitized" (selected); a "Source" dropdown menu showing "Aerial Hand Drawn"; a "Differential Correction" dropdown menu showing "N/A"; a "Travel Method" dropdown menu showing "N/A"; and a "Map scale of source" dropdown menu showing "1:24,000". At the bottom are "OK" and "Cancel" buttons.

Fire Perimeter Form

Collection Method Cc

Collection Method

GPS Digitized

Source

Aerial Hand Drawn

Differential Correction

N/A

Travel Method

N/A

Map scale of source

1:24,000

OK Cancel

Form Controls



- Label
- Button
- Radio Button
- Combo Box
- Slider
- DateTime
- Text Box
- Check Box
- Image Box
- List Box
- UpDown
- Sub Table



Key Form Events



- Initialization
 - Use Form's OnLoad event
- Closing
 - Use Form's OnOK / OnCancel events
- Validation
 - Use each Control's OnValidate event
 - Use Page's OnValidate event
- Control interaction
 - ComboBox / ListBox → OnSelChange event
 - CheckBox → OnClick event
 - Button → OnClick event

Events — *An Applet Form Example*



Form & Control Events

Country Names

Sample of how to make one control change/effect another control using event/s programming model.

First letter in Country Name

[] above

[]

A
N
C
U

[]

ok x

This screenshot shows the initial state of the applet. It features a title bar "Form & Control Events" and a tab "Country Names". Below the tab is a text area with a sample description. A label "First letter in Country Name" is positioned above a dropdown menu. To the right of the dropdown is the word "above". Below the dropdown is a list box containing the letters "A", "N", "C", and "U". Another dropdown menu is located below the list box. At the bottom left, there are "ok" and "x" buttons.

```
<TOOLBUTTON> FormEvents  
abc name=FormEvents  
shortcut=  
image=$question  
onclick=ShowAppletForm("Form Events", "Form1");  
abc prompt=Form & Control Events
```

Form & Control Events

Country Names

Sample of how to make one control change/effect another control using event/s programming model.

First letter in Country Name

A []

List of Countries based on above control selection

[]

Australia
Austria
Antarctic
Angola
Argentina

ok x

This screenshot shows the applet after an event. The dropdown menu now contains the letter "A". The list box below it is updated to show a list of countries: "Australia", "Austria", "Antarctic", "Angola", and "Argentina". The "Antarctic" option is currently selected. The "ok" and "x" buttons are still present at the bottom left.

- Associate an event with a control
- Use one control to change/effect another control using events

Form Validation Example

```
function ValidateInput()
{
// input value must be 5 to 8 characters and the first character cannot be a number
    var blnValid;
    var strValue = ThisEvent.Object.Value;
    if (!isNaN(parseInt(strValue.charAt(0))))
        blnValid = false;
    else
    {
        if ((strValue.length >= 5) && (strValue.length <= 8))
            blnValid = true;
        else
            blnValid = false;

        if (!blnValid)
        {
            ThisEvent.Result = false;
            ThisEvent.MessageText = "Invalid input";
            ThisEvent.MessageType = apExclamation;
        }
    }
}
```

Other Important Form Properties



- .Fields property used to access fields of current feature in an EDITFORM during Form events
- .Mode property tells you how the form is being used
 - Editing existing feature
 - Capturing new feature
 - Identifying a feature
 - General (unbound) form
 - Not currently displayed

Fields Property Example

```
// OnLoad event of an Edit Form  
var pForm = ThisEvent.Object;  
pForm.Fields("AREA").Value = pForm.Fields.Shape.Area;  
pForm.Fields("GPS").Value = g_blnGPS;  
pForm.Fields("USER").Value = g_strUser;  
pForm = null;
```

Controlling ArcPad Preferences



- Preferences object limited, but adequate to get the job done
 - Path to ArcPadPrefs.apx and other key paths
 - Ability to re-load preferences from ArcPadPrefs.apx
- ArcPadPrefs.apx structure well documented
- XMLDOM ActiveX object available on most systems (Microsoft.XMLDOM)
- A few lines of script do the trick

Modify ArcPadPrefs.apx Example

```
function SetAveraging {
    var ARCPAD_APX_FILE = System.Properties("PersonalFolder") +
        "\\ArcPadPrefs.apx";

    Preferences.Write();

    var pArcPadXML = new ActiveXObject("Microsoft.XMLDOM");
    pArcPadXML.load(ARCPAD_APX_FILE);

    var pAVGElement = pArcPadXML.documentElement.
        selectSingleNode("PREFERENCES/GPS/AVERAGING");
    pAVGElement.setAttribute("enabled", "true");
    pAVGElement.setAttribute("point", "30");
    pAVGElement.setAttribute("vertex", "5");
    pArcPadXML.save(ARCPAD_APX_FILE);

    Preferences.Read();

    pArcPadXML = null;
    pAVGElement = null;
}
```

Utility Extensions



- Expose “advanced” functionality to your scripts
- Written in C/C++ and compiled as DLLs
 - Mostly boiler plate code from samples
- Add new options to the Tools menu
- .Escape method provides 3 variant arguments
 - 1 identifier, 2 for data
- .ExProperties property provides named read/write properties
- Send an event to ArcPad via PostMessage
- Send any script to ArcPad via SendMessage
 - SendScript wrapper provided in sample code
- Evaluate any script expression
 - Helper macro provided in Extension API

Create a utility extension demo



- GOAL: Keep system time synchronized with GPS time
 - Get GPS time – `GPS.Properties("UTC");`
 - Set System time – `SetSystemTime WinAPI`
 - Write utility extension to receive GPS time from an applet and set the system time
- Extension logic
 - Receive GPS time from applet via `APEXEscape`
 - Convert variant time to `SYSTEMTIME`
 - Call `SetSystemTime WinAPI`
- DEMO

Inter-process Communication



- Several ways to communicate between ArcPad and another application (VB, eVC++/C++, .NET ...)
 - Windows APIs (FindWindow, ShowWindow, SendMessage, PostMessage) exposed to scripting via the System object
 - ArcPad can receive a script via a WM_COPYDATA message and an event via a custom (WM_ARCPAD_EXEVENT) message
 - Exchange data by reading/writing to a text/binary file
 - Write a “bridge” COM DLL
 - Make your app scriptable and expose it to ArcPad

Send scripts to ArcPad from another app



- Free examples with source code on ArcScripts
 - Send script to ArcPad from external C/C++ program
 - Send script to ArcPad from external VB application
 - Send scripts to ArcPad from external .NET Compact Framework applications (C# / VB.NET)
 - Sending Scripts to Arcpad from external eVB-application
- DEMO

Related Tables and Data-linked Forms



- Rudimentary display-only related table supported provided by SUBTABLE control
- Edit/Identify forms support bound controls for shapefile attributes
- General forms can be linked to external tables via scripts
 - Applets and Default Configurations as well as Layers
 - Related table support requires managing a unique key in your scripts

Linking Forms to External Databases



- General Forms can be programmatically linked to external databases to create Edit Form behavior
- Use ArcPad's Data Access Objects with DBF tables
 - RecordSet, Fields, Field
- Use Microsoft's ADOCE for databases other than DBF
 - Creatable ActiveX objects
 - Similar to ArcPad's Data Access Objects
 - Needs to be installed on most mobile devices (free install)
 - ** No longer supported at CE 5 / WM 5
- <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adoce31/html/adowlcm.asp>

Data Forms with External Databases: Programming Pattern



- Form OnLoad event
 - Open the table
 - Get the desired record
 - Write the record's field values or default values to form controls (edit, combobox, datetime, etc.)
- Form OnOK event
 - Write form control values to the record or create a new record
 - Close the table
- OnValidate events
 - Implement OnValidate event handlers at the page and/or control level to enforce data integrity rules

Open Table Example

```
function OpenDBFTable(p_strDBFName, p_intMode)
{
    var pDBFTable;
    try
    {
        pDBFTable = Application.CreateAppObject("RecordSet");
        pDBFTable.Open(p_strDBFName, p_intMode);
    }
    catch (e)
    {
        pDBFTable = null;
    }
    finally
    {
        return pDBFTable;
    }
}
```

Related Table Form OnLoad Example

```
function FormLoad()
{
    // reference the form controls (1 page form)
    var pControls = ThisEvent.Object.Pages(1).Controls;
    // open the data table
    var pTableRS = OpenDBFTable (DBF_TABLE, 1);
    if (pTableRS == null)
    {
        Application.MessageBox("Error opening table.", apExclamation, "Error");
        pControls = null;
        return;
    }
    // populate form controls
    pTableRS.Bookmark = g_ITableBookmark;
    pControls("txtName").Value = pTableRS("NAME").Value;
    pControls("txtAge").Value = pTableRS("AGE").Value;
    // clean up
    pTableRS.Close();
    pControls = null;
    pTableRS = null;
}
```

Related Table Form OnOK Example

```
function FormOK
{
    // reference the form controls (1 page form)
    var pControls = ThisEvent.Object.Pages(1).Controls;
    // open the data table
    var pTableRS = OpenDBFTable (DBF_TABLE, 2);
    if (pTableRS == null)
    {
        Application.MessageBox("Error opening table.", apExclamation, "Error");
        pControls = null;
        return;
    }
    // populate record
    pTableRS.Bookmark = g_ITableBookmark;
    pTableRS("NAME").Value = pControls("txtName").Value;
    pTableRS("AGE").Value = pControls("txtAge").Value;
    pTableRS.Update();
    // clean up
    pTableRS.Close();
    pControls = null;
    pTableRS = null;
}
```

Serial Device Communication



- ArcPad supports serial communications
- Bluetooth devices are supported via the Serial Port Protocol (SPP)
- GPS, Rangefinder, and AUX communications are exposed via the corresponding objects
- Serial parameters are specified in ArcPadPrefs.apx

AUX Object



- Two way communication with attached serial device
- Supports ASCII and binary communication
- Uses include control or monitoring of other instruments or devices
 - Water quality probes
 - Soil quality probes
 - Temperature sensors

Auxiliary Communication Port

AUX

- Break : Boolean
- CommEvent: Integer
- InBufferCount: Long
- IsOpen: Boolean
- OutBufferCount: Long
- ← Close
- ← Open: Boolean
- ← Read(NumChars: Integer, Optional Wait: Boolean): String
- ← ReadByte(Optional Wait: Boolean): Integer
- ← ReadLine(Optional Wait: Boolean): String
- ← Write(Text: String): Long
- ← WriteByte(Byte: Integer): Boolean
- ← WriteLine(Optional Text: String): Long
- ← WriteSentence(Text: String): Long
- OnClose
- OnComm
- OnOpen

Read AUX Port Example

```
function ReadAUXForTemp()
{
// called in OnComm event to read data from AUX port

// read from the AUX port
var strRawData = AUX.ReadLine();

// parse the comma-delimited data
var arrParsedData = strRawData.split(",");

// if data is valid, return the temperature
if (arrParsedData[0] == "$PTEMP")
    return (parseFloat(arrParsedData[1]));
else
    return (NaN);
}
```

Internet Communication



- Move some application logic to the server side
 - You must be connected to the internet (wired or wireless)
- INET object provides connectivity with generic web servers (ASP, JSP, ASP .NET, etc.)
- FTP object provides access to FTP operations
- ArcIMS object provides connectivity with ArcIMS
 - You must construct requests and parse responses

Internet Objects



ArcIMS

creatable

- RequestAXL: String
- ResponseAXL: String
- ServerName: String
- ServerPort: Long

- ← SendAXLRequest
(ServiceName: String,
AXL: String): String

INET

creatable

- HostName: String

- ← OpenURL(URL: String): Variant

FTP

creatable

- ErrorCode: Long
- ErrorDescription: Long
- FailOnError: Variant
- Port: Long

- ← Close
- ← CreateDirectory(Directory: String): Boolean
- ← DeleteFile(FileName: String): Boolean
- ← FindFiles(SearchFile: String, Optional AttributeFilter: Long):
Variant
- ← GetCurrentDirectory: String
- ← GetFile(RemoteFileName: String, LocalFileName: String,
Optional FailIfExists: Boolean): Boolean
- ← Open(HostName: String, Optional UserName: String, Optional
Password: String, Passive: Boolean): Boolean
- ← PutFile(LocalFileName: String, RemoteFileName: String):
Boolean
- ← RemoveDirectory(Directory: String): Variant
- ← RenameFile(ExistingFileName: String, NewFileName: String):
Boolean
- ← SetCurrentDirectory(Directory: String): Boolean

INET Example

```
function SendToServer (p_pPoint)
{
    // create the INET object
    var pINET = Application.CreateAppObject("INET");

    // construct the ASP querystring
    var strQueryString = "http://myserver.com/Process/ReceivePoint.asp?";
    strQueryString = strQueryString + "XCoord=" + p_pPoint.X;
    strQueryString = strQueryString + "&YCoord=" + p_pPoint.Y;

    // send the querystring to the server
    pINET.OpenURL (strQueryString);

    // clean up
    pINET = null;
}
```



ArcPad OEM Tool

- Allows you to re-brand an ArcPad-based solution
 - All references to 'ArcPad' are replaced with your product name
 - You are responsible for documentation and support
- Available only to ESRI Business Partners

Session Evaluations Reminder



Session Attendees:
Please turn in your session evaluations.

... Thank you