A-Z tutorial, Session 1
Requires FAA & FAS V2.0+
Session 1
Understanding FoxInCloud

This session targets users who wish to see the big picture before starting hands-on experience… it describes how FoxInCloud Application Server (FAS) operates your application on a Web server and interacts with the client browser.

Reproducing steps of this session requires setting break points at various locations of FAS source code (licensed users only).

Though you may follow the steps by browsing the coming slides and applying the steps on your own machine, you may get a better understanding by attending a FoxInCloud workshop or training session; please feel free to mail contact@foxincloud.com for workshops and training sessions to come.

FAS installs the files used in this session into <VFP9>\Tools\AB\AW\Samples\FIC\FICtuto\
## Understanding FoxInCloud, Agenda

<table>
<thead>
<tr>
<th>Episode</th>
<th>Subject</th>
<th>Slides</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FoxInCloud Architecture</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Connecting Web Server and FAS</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Displaying app’s main form in Browser</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Interacting with a form in Browser</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
</tr>
</tbody>
</table>
Web server (IIS, Apache, ...): Hosts Web site (i.e. www.mysite.com)

FIC App. Server (FAS): VFP/wConnect-based FoxInCloud Application Server

App Host: Private session where adapted application runs

HTML/CSS/JS gen: clones your app’s forms into HTML/CSS/JS which becomes the designated website (i.e. www.mysite.com)

Process: Processes HTTP requests received from the site

AJAX: Processes user actions on forms thru AJAX requests

Session 1, episode 1
Connecting Web server to FAS

Web Server → FIC App. Server → App Host

- Process
- AJAX

HTML / CSS / JS on Web site

End user on Web site
IIS comes with most Windows editions

Note: the free FoxInCloud Adaptation Assistant (FAA) automates all IIS settings to come;

Session 3 demonstrates this automated installation.

Grabbing requests from localhost web server

An “IIS application” is a virtual directory where an application can run.

Adding an IIS application

IIS allows URLs with a given extension to be processed by a given module (.exe or .dll)

Mapping an URL extension to your FAS

Target module is wConnect’s “wc.dll”

FAS relies on “West-Wind web connect” (wConnect)

wc.dll is the IIS extension that wConnect provides.

Main setting is the folder where `wc.dll` and FAS trade request and response files.

When getting an HTTP request from IIS, `wc.dll` writes request details into a file in this folder.

Open ficTuto.pjx and run main program tutoTest.prg

Starting FoxInCloud Application Server (FAS)...

awAppHost is app’s session object. Both desktop and Web versions of your app share the xxxSets Settings class.

FAS enters wait state

FAS waits for timer events

FAS timer fires

In dev. mode, FAS waits for files written by the Web server, containing HTTP request details.

FAS: ready to serve requests!

Session 1, episode 2
Displaying main form in browser

Web Server → Fic App. Server → App Host

→ Process

→ AJAX

→ HTML / CSS / JS on Web site

→ End user on Web site
index.scx is a regular VFP form with several command buttons and a background picture ...

The form we want to display in the browser

We redirect IIS's default document index.htm to an URL with the extension mapped to our FAS: index.tuto

Typing application URL

Due to IIS redirection of
`index.htm` to `index.tuto`, typing application root URL
`http://localhost/tutotest/` yields application home page

As we mapped *.tuto to wc.dll, IIS transmits the .../index.tuto request to wc.dll

wc.dll gets the request from IIS and dumps it into a file

As soon as FAS timer fires and finds a request file in the folder shared with wc.dll, FAS timer reads its content.

(Default timer interval is 200 ms.)

Timer fires, FAS reads request from file

Inside FAS, the `.tuto` extension maps to the `tutoProcess` class.

This class processes the request.

FAS routes request to application’s Process object

Application executes process class method

The process class executes the method named after justStem(url)

("index.tuto" executes the method ‘index’ of process class mapped to ‘.tuto’)

All FoxInCloud member names begin with ‘w’

oAppHost is FoxInCloud application host object.

Application host object keeps a pool of references to the various forms in the application.

Process asks application host for a reference to the form.

App. Host does not hold any reference to the form requested:

This form needs to be instantiated

First time this form is used: App Host must instantiate

App. Host instantiates the form using a regular `DO FORM...NAME` command.

This form instance lives inside App Host as long as server runs.

All users share this instance.

Form instantiates and fires events just like in desktop:
- `.Load()`,
- `.Init()`,
- `.Show()`, ...

As any Web application, FoxInCloud needs to save application and forms state for each user, including dataSession;

Initial state is restored whenever a user opens a form.

FAS saves form’s initial state into a VFP table

FoxInCloud’s awHTMLgen class walks through the form’s containers hierarchy to generate HTML CSS JS replicating the layout and behavior of original VFP form.

FAS send back a full HTML page including the form

FoxInCloud's .wFormStandardPage() method builds a full HTML page including <head>, FAS standard and application-specific CSS & JS, <body>, etc.

Any question? Post with screenshot in the 'FoxInCloud' section of http://west-wind.com/wwThreads/ or http://www.universalthread.com/
The whole process, from FAS receiving the request to response delivered takes less than a second ...

index.tuto -
Launching form #1 'index.scx'...
Instantiated in .124 secs.
Setup in .006 secs.
State saved in .134 secs.
HTML/CSS/JS generated in .334 secs.
Finished with success in .914 secs

Session 1, episode 3
Interacting with a form

Web Server → Fic App. Server → App Host → Process → AJAX → HTML / CSS / JS on Web site → End user on Web site
In the desktop application, clicking the `.rowSourceType` command button opens the `rowSourceType.scx` form.

Let’s click the same button in web browser ...

FAS has generated the same button in HTML

FoxInCloud has generated a JS `.click()` event handler

Because the `.click()` event of this button is implemented in the original VFP form, FAS has generated a similar event handler in form's JavaScript named `index_scx.js`.

All user interaction events are processed by standard FoxInCloud.js.DOMEvent method, resulting in sending an AJAX request to the FAS server; URL details the event occurred.

Clicking this button triggers an AJAX request to FAS.

FAS executes button’s .click() method

.DOMEvent() goes through several FAS methods to end up executing the application button’s .Click() method.

.wForm() opens RowSourceType.scx as a child form.

Similar to what we saw in the previous episode, `wForm()` asks Application Host for a reference to the form; if none exist, AppHost launches the form, saves its state and generate HTML/CSS/JS.
.wForm() adds child form dialog into AJAX response

Knowing that form’s HTML/CSS/JS is available, .wForm() asks the FAS AJAX response object to add the form’s HTML/CSS/JS to the AJAX response.

FAS response to AJAX requests is encoded as an XML stream.

FoxInCloud.js parses this XML response into JavaScript instructions to update HTML page.

Web dev. Tools such as Firebug for Firefox provide a very handy debugging environment; here we can see the full XML response received from FAS server, original request, and much more.

FAS builds an XML response to AJAX requests (mainly user events).

XML response embeds JS instructions for the HTML display to reflect VFP form's state after user action, just like in desktop.

FoxInCloud.js executes orders read in XML response.

Child form shows up in the browser

By comparing the state of each form before and after user action \[\text{.Click()}\] in this case, FAS could send all the necessary orders for \text{FoxInCloud.js} to update the client HTML page accordingly.