Guidelines for Writing Web Applications Compatible with the IVE's Content Intermediation Engine

Introduction

This document provides guidelines for web application developers and UI designers on creating web content that will work seamlessly through the Instant Virtual Extranet (IVE). One of the core technologies offered by the IVE is the Content Intermediation Engine (CIE), a highly advanced parser and rewriter. The CIE retrieves web-based content from internal web servers and changes URL references and Java socket calls so that all network references point to the IVE. An example of how it works is when an authenticated user clicks on a link, the request goes to the IVE. The IVE parses the incoming link to determine the internal destination server and forwards the request to that internal server on behalf of the end user. The end result is what we call "intermediation" where the IVE acts as the internal server to the end user and as an end-user to the internal server. This provides protection and clear separation between end-users and internal resources.

A web application that works through the IVE is one where the CIE was able to locate all the links in the pages and rewrite them accurately. This document provides guidelines for creating such web applications. It provides general recommendations, lists the content-types that are supported, the level of support provided for each of the content types, and the language constructs to avoid.

The IVE fully supports web applications written in standard HTML, Javascript, VBscript, and Java. There are corner cases where some content is sensitive to intermediation and parsing; these are outlined by content type below. If a content type is not listed then the IVE probably does not "officially" supports it but it may still work.

Content-Types

HTML

Native HTML 4.0 is fully supported.

Use Well-formed HTML

We recommend that you run your HTML through an HTML syntax checker to ensure that it is well-formed. This will eliminate the possibility of poorly formed HTML with missing end tags, right brackets, etc. While the IVE's content intermediation engine is powerful enough to successfully intermediate invalid HTML, it is safer to write valid and well-structured HTML.

Use Standard HTML

It is recommended that standard HTML be used in the web pages. For example,

use the standard format

<A href="www.servername.com:portNo"> Click Here </A>

instead of the more rare format

<A href="www.servername.com" port="portNo"> Click Here </A>

Specify the Correct Content Type

The Content-Type header should match the actual content of the document. For example, do not send a content type of text/html if the content is XML.

URLs in the HTML Pages

Follow the URL specification, http://www.faqs.org/rfcs/rfc1738.html when constructing URLs in HTML pages. Avoid using HTML escape codes in the URLs. Use forward slashes ('/') in URLs instead of backward slashes ('\').

HTTP Headers

The IVE supports HTTP/1.1 and 1.0 when communicating to the browser whereas only HTTP/1.0 is supported to the backend server. Make sure that all HTTP headers adhere to the HTTP specification for the version being employed.

The IVE does not pass all headers from the internal web servers to the browser. Avoid using custom headers because the IVE may not pass those headers back to the browsers. Instead use the standard headers defined by HTTP.

Character Encodings

Specify the charset in the META tag to avoid problems relating to character encoding. For example, to specify that the character encoding of a document to "EUC-JP", the document should include the following META declaration:

```
<META http-equiv="Content-Type" content="text/html; charset=EUC-JP">
```

Browser-Specific Code

Avoid writing HTML that is browser-specific. If a construct is supported by most commercially-available browsers, it is likely that the IVE will support it too.

For example, the following code snippet that uses layers is supported by the IVE:

```
<style type="text/css">
<!--
#Layer1 {left: 55px; top: 120px;}
#Layer2 {left: 300px; top: 120px;}
-->
</style>
</head>
<body>
      <div id="Layer1"><a href="http://www.google.com">Google</a> </div>
      <div id="Layer2"><a href="http://www.yahoo.com">Yahoo</a> </div>
</body>
```

Microsoft Word and Microsoft PowerPoint can generate Internet Explorer-specific HTML for embedded drawings and figures. These tags are embedded within comments, so

they are not rendered in non-IE browsers. These conditional comments may not be rewritten appropriately.

Miscellaneous:

- Avoid complex nesting and escaping of quotes.
- In HTML tags, do not use null src attributes.
- Avoid using inline server-side script tags, typically marked by <% … %>. These tags are usually processed by the server before they reach the client. Occasionally, when a server does not process the server-side tags, the scripts remain on the client page which can cause problems.
- When writing <OBJECT> and <APPLET> tags, make sure codebase and cabbase, are present.
- For best performance, we recommended keeping all text between the angled brackets, "<" and ">", of a tag be less than 10,000 characters. For example: <tagName name="A very large string here could be broken up for better performance"></tagname>.

JavaScript

The IVE content intermediation engine handles complex uses of JavaScript, including menu animation, field validation, pop-up windows, frame manipulation, and calendar functions. In addition, standard and advanced JavaScript functions such as setTimeout, setInterval, and insertAdjacentHTML are also supported.

JavaScript has a feature that also allows dynamic generation of code on the browser. For example, the eval() function allows for dynamic generation and execution on the client-side. It is not possible for the server to accurately intermediate this code. For those situations, the IVE inserts into the rewritten page a client-side parser written in JavaScript whose purpose is to parse and rewrite this dynamically generated code.

Even though the rewriter is sophisticated enough to handle complex constructs in Javascript, it is possible to write code that contains multiple levels of indirection causing the intent of the code to be obscure. We recommend that the code be written in a straightforward fashion so as to enable the rewriter to capture all the URL references.

Use of document.write

The use of document.write is supported through the rewriter. However, its use is discouraged because parsing of document.write is done on the client-side, and this could impact performance. If you still need to use document.write, please adhere to the following guidelines:

- Do not use base href's in document.write.
- Avoid writing script tags in document.write.
- Do not write partial tags in the document.write string. For example, avoid the following:
  <script>
  document.write("("<aTag> Tag contents") ;
  </script>
  </aTag>

The eval() Function

The eval() JavaScript function is supported through our client-side parser and rewriter. However, the client-side parser is not as sophisticated as the server-side intermediation engine. As a result, there are cases where we accurately rewrite code inside a <script> tag but may not handle the same strings passed in an eval(). Therefore complicated constructs within an eval function may not work as expected. For example, `window.open()` within the `eval()` function will work, but accessing the DOM in an eval() function may not.

DOM Usage

Pages that use the Document Object Model are supported through the IVE. However, when traversing the DOM, do not assume the number of elements or the position of the elements. Instead use criteria such as the id field of the element to access specific DOM elements. The reason is that the IVE may add content to the intermediated page invalidating the original number of DOM elements. Therefore web pages that assume the number of elements or position of an element may trample upon or use content added by the IVE. For example, if there are five elements on a page, the IVE may add a sixth element to the DOM. If the web application then attempts to access and display the last element of the page, the element inserted by the IVE will be displayed, which was not the desired intent. This is especially relevant when using the IVE toolbar.

Use One Scripting Language per Page

Use only one scripting language in one page. Do not mix JavaScript and VBScript in the same page, and between the two, use JavaScript over VBScript since VBScript has no published standard that we can recommend at this time.

In relation to scripting languages, keep in mind that the use of an empty type attribute in a script tag will not work.
For example,

```
<script language="javaScript" type=''>
      Some Javascript Code
</script>
```

The "`with`" command

Limit the use of the "`with`" command. This could lead to incorrectly rewritten pages. For example, instead of using,

```
with (doc){
          location=http://...;
   }
```

use

```
   doc.location=http://...
```

Even though the '`with`' statement above is supported, it is recommended that such statements be avoided and simpler constructs be used. More complicated statements such as nested `with` statements may not be rewritten properly. The reason for problems with the "with" statement is that it is difficult to distinguish local variable references from property references on an object.

Example:

```
foo = 1;
```

is a local variable but:

```
with (obj) {
        foo = 1;
}
```

In the example above, it is difficult to determine if 'foo' is a local variable or a property of 'obj'. We use heuristics to trap the common combinations of objects and properties but as is obvious that will not translate to a general solution. For that reason, we recommend that you avoid the use of "with".

Use of document.cookie to set cookies on the client side.

Most browsers have an upper bound on the number of cookies that can be set on the client-side through the use of document.cookie. You cannot use the maximum allowable by the browser since the IVE too sets cookies. The IVE may set up to four cookies for managing configuration information (this number is determine on the configuration options chosen by the IVE administrator). Therefore the total number of cookies that can be set by the web application is limited to the maximum less four.

Miscellaneous

- Avoid using variables that indirectly assign URL references to native javascript objects using the array format rather than the regular dot format.

  For example:

  ```
  document["location"] = "http://www.yahoo.com";
  ```

  and

  ```
  var d = document;
  var l = "location";
  d[l] = "http://www.yahoo.com";
  ```

  Use document.location = `"http://www.yahoo.com"; instead.`

- Do not use HTML and Javascript reserved words and built-in functions as object names, function names or variable names in your code. For example, do not define and use variables such as *top*, *location*, and *domain*.
- The IVE occasionally generates its own Javascript functions that start with the string "Dana". To avoid conflicts with the IVE's Javascript functions, avoid using DanaXXX as function and variable names.
- Avoid embedding Javascript in the src attributes of tags.
  For example,

  ```
  <frame name="f1" src="javascript:func();">
  ```

- The use of port in window.location is not supported. For example, the javascript code, window.location.port = portNo, is not supported.

CSS

Cascading style sheets are supported. Make sure that the content type is set to text/css for cascading stylesheets. If the content type is incorrect then errors could occur through the rewriter. JavaScript in a cascading style sheet is not supported.

Java

The IVE instruments Java bytecode so that all HTTP(s) and socket based network communication is redirected to an intermediate proxy server via secure HTTPS tunneling. This approach provides a secure and portable proxy mechanism for Web-based client/server applications that utilize client Java applets. The Java rewriting technology is available on the Sun JVM (version 1.4.1+) and MS JVM platforms. For support of applets written for other JVMs, such as Oracle JVM and IBM JVM, please see the Pass Through Proxy section below.

Most network related classes and methods are supported through the Java rewriting engine. As long as the Java applet uses TCP and the network traffic is initiated from the client, the applet is supported. If the applet contains server-initiated connections through the use of the ServerSocket class then the applet will not work through the IVE.

Since the IVE intermediates the byte code, the final applet delivered to the end-user is different from the original applet. Do not include a checksum validation in the applet to verify that the applet has not been altered because in this case, the applet has been altered! Use the standard code signing procedures to secure the applet. Please read the section on Code Signing Certificates for more details on how the end-user can be assured that the applet is safe.

To prevent Java intermediation problems with the IVE, ensure that all network-related classes conform to the Sun Java specification. Class files written in a proprietary format may not be supported. If the class files do not contain standard byte code then the IVE cannot intermediate the content.

Code Signing Certificates

Most commercial Java applets that are intermediated through the IVE perform privileged tasks. To perform these tasks, the user must accept the certificate that is used to sign the applet. However, since the IVE modifies the bytecode, the original signature is invalid and the IVE must re-sign the applet. The IVE re-signs the applet with a self-signed certificate whose CA is not a trusted root. Due to the use of the self-signed certificate, the browser displays a warning that must be accepted for every launch of the applet. To avoid the frequent security warnings, you will need to import a code signing certificate. The instructions on how to import and configure code signing certificates for Java applets are included in the Administrator's Guide.

VBScript

The IVE supports rewriting of VBScript. Since VBScript does not conform to a published standard, it is more difficult to provide comprehensive guidelines. We recommend that you use JavaScript whenever possible.

ActiveX

Active X programs that do not make network calls (for example, through TCP or HTTP) are supported through the IVE. Active X programs that do make network connections may or may not work through the IVE. Since a standard is not available that states where URLs, port numbers, or hostnames can be defined, the content intermediation engine may not locate these items and modify them. For instance, an Active X program could choose to define the first parameter as a URL and the second parameter as the username while another Active X program could reverse the order of parameters. The CIE does not have the necessary knowledge to consistently rewrite the connections in all cases due to the lack of standards inherent to ActiveX. Active X programs that just contain relative links and do not make direct socket connections can be supported through the use of an option available on the rewriter called **Pass Through Proxy**.

Flash

Since the Flash object is a proprietary binary format, the links embedded in Flash are not parsed and rewritten. Therefore Flash content that contains links are not typically supported (see exception below). If the Flash objects do not contain any links then the content is displayed accurately. It is possible for a Flash object to be composed of other Flash objects. In such situations, the included Flash object may not display because the location of the included Flash content is not intermediated.

If the Flash pages contain just relative links then they can be supported through the use of an option available on the CIE called Pass Through Proxy. See section on Pass Through Proxy below.

XML/Adobe SVG

XML-based content will be displayed accurately on a browser but links within the XML content will not be rewritten (See exception below). The XML language does not define where URLs, port numbers, and hostnames can be defined within the XML content which prevents the IVE from locating and replacing these strings. However, the following links in an XML document will be intermediated properly:

- META tag
- xsl:import
- xsl:include

If the XML pages contain just relative links, they can be supported through the use of an option available on the CIE called Pass Through Proxy. See section on Pass Through Proxy below.

PDF

Since PDF is a proprietary format, the links embedded in the PDF file are not rewritten. If PDF file does not contain any links then the content displays accurately on a browser. If the PDF files contain only relative links then they can be supported through the use of an option available on the CIE called Pass Through Proxy. See section on Pass Through Proxy below**.**

Streaming Media and Video Content

Streaming media content often contains direct network connections without the use of HTTP and therefore cannot be supported through the CIE. If the streaming content is delivered through an <OBJECT> tag and one of the attributes of the tag is a URL to which an HTTP connection is made then the content may work through the IVE.

Pass Through Proxy

Pass Through Proxy is a key component of the Content Intermediation Engine designed to support various intermediation-sensitive content types such as:

Active X with relative links

Flash with relative links

XML with relative links

PDF with relative links

Other JVMs such as IBM and Oracle JVM with relative links

Summary

Our final recommendation is to test the web-based content through the IVE Content Intermediation Engine. If a page does not display accurately then this document can provide suggestions on how the code can be altered to ensure compatibility with the IVE.

To summarize, web applications written in HTML, JavaScript, VBScript, or Java that use the guidelines listed above should work seamlessly with the IVE Content Intermediation Engine (CIE). Using the Pass Thru Proxy (PTP) option of the CIE, some other content such as Flash, Active X, PDF, XML, etc. which contain only relative links are supported.

WEB APPLICATION	SUPPORT VIA IVE Content Intermediation Engine
HTML	√
JavaScript	√
VBScript	√
Java	√

Flash	√*
Active X	√*
PDF	√*
XML	√*
Streaming	✕

√ - fully supported. Only refer to this doc if you have issues

√* - partially supported. May require a PTP option on the CIE. Refer to this doc for details. In most cases, these content can be intermediated. In the few situations where they are not, the content can easily be modified to be so. If this is not possible, please contact your account team about SAM as an alternative option.

✕ - very limited support. If your application does not work, please contact your account team about Network Connect as an alternative option.

While the IVE does offer a suite of other options, the Content Intermediation Engine is the most robust rewriting engine in the Secure Access (SSL-VPN) market. By adhering to the above guidelines, you will ensure support through the CIE, which is a "pure" clientless solution.