ESI Frequently Asked Questions

For additional information about ESI development, please refer to the *ESI Developer's Guide*, *Edge Server Handling of Edge-control and Other HTTP Headers*, *TTL Methods and Considerations*, and other documentation found on the Akamai EdgeControl Management Center at https://control.akamai.com.

Known Issues

Some Akamai-Specific Variables are Not Recognized as Strings 3
The <esi:eval> statement does not catch exceptions properly 3
The TRAFFIC_INFO Variable Must Not be Empty. 3
How ESI Knows Whether a Variable is a List, a Dictionary or a String. 3

Overview and General Features

What is ESI?. 4
How does ESI use templates and fragments? 4
How many levels of nested include statements does ESI support?. 4
Is there a limit on the number of successful includes?. 4
Is there a limit on the total size of an ESI page? 4
Does ESI provide for custom variables and custom functions? 4
Does ESI allow for any decision making when building a page? 5
Does ESI allow for iterative operations? 5
Can ESI process SSL requests? 5
Can ESI return binary output? 5
Can ESI be used with double-byte pages? 5

The Language and Syntax

What happens if the ttl attribute in an ESI include statement conflicts with a no-store? 6
What's the difference between the alt and except tag? 6
What happens if the src and alt tags cannot be retrieved? 6
Can I use a $ in an esi:assign statement when creating a custom variable? 6
Can I use the quote marks, " and ', in an esi:assign statement when creating a custom variable? 7
Can I use a $ or other special characters in an esi:include statement? 7
Is there a general method to escape special characters in ESI? 7
How Can I include HTML that contains a $ or other special characters? 7
Can I include flat text in an ESI-code page? 8
Can ESI support regular expression string pattern matching? 8
Does ESI support case insensitive substring matches? 8
How do I test to see if a variable is a Dictionary, String, or List? 8
Does ESI support generating a random number? 8
How can I convert a number to a string? 8
How can I debug or test my ESI code? 9
Can I turn on the debugger conditionally or all-the-time. 9
What operations can I use in qualifying a conditional include? 9

VARIABLES AND HTTP INTEGRATION
How can I perform geographic-based content targeting with ESI? 10
What HTTP environment variables does ESI support? 10
How Can I Test a Variable to See if It's an Integer. 10
Can ESI pages set Cookies? 10
Can ESI remove Cookies or other HTTP headers from the response? 11
Can ESI set the HTTP Content-Type Response header? 11
Does ESI support POST requests? POST responses? 11
How does ESI support HTTP redirects? 11
How can I set up different caching properties for the src and alt objects? 11
How can downstream cache-ability be set for ESI pages? 12

INTEGRATION AND IMPLEMENTATION
What should I know about CNAME'ing the site when implementing ESI? 13
Can I specify edge server configuration settings within ESI? 13
Can a default object be ESI processed? 13
Do you have any integration options for converting my JSP/ASP/SSI/CSS pages to ESI?. 13
Does ESI work with my environment (ATG, Allaire, SSI, etc)? 13
What is the performance overhead in ESI processing? 14
Can I turn on ESI processing for an entire site? 14
What if I want to use ESI, but not have to worry about re-coding my site if I turn off ESI? 14

INDEX. 15

Known Issues

Some Akamai-Specific Variables are Not Recognized as Strings

Some Akamai specific variables are not recognized as strings, .including HTTP_ACCEPT, HTTP_ACCEPT_LANGUAGE, HTTP_CONNECTION, HTTP_ACCEPT_ENCODING, and HTTP_USER_AGENT. Currently, you can't use string operations such as **`$lower()`** on these.

The workaround is to use **`$str()`** to convert the object to a string. For example, you can convert HTTP_ACCEPT_LANGUAGE to a string and then use $lower() on it.

The <esi:eval> statement does not catch exceptions properly

There is a bug with the **`<esi:eval>`** statement that causes it not to properly return HTTP response codes for errors. Until this bug is fixed in a future release, you should not use **`<esi:eval>`** in an **`<esi:try>`** block where you would expect to be able to take action based on receiving the correct error response code. To work around this, place the fail-action in the configuration file rather than depend on the **`<esi:try>`** block processing or other exception processing when using **`<esi:eval>`**.

The TRAFFIC_INFO Variable Must Not be Empty.

If you use the TRAFFIC_INFO variable, you should make sure it isn't empty—as it may well be on the first day of the reporting period when there has been no previous traffic. A client could be denied service. This is scheduled to be fixed in a future release. In the meantime, check to see if the variable exists and if so, whether it is empty. For example:

```
<esi:choose>
    <esi:when test="$exists($(TRAFFIC_INFO{TotalTraficInMonth}))"
    <esi:choose>
        <esi:when test="!$is_empty($(TRAFFIC_INFO{TotalTraficInMonth}))"
            [Take some action, for example, assign it a minimum value]
        </esi:when>
    </esi:choose>
    </esi:when>
</esi:choose>
```

How ESI Knows Whether a Variable is a List, a Dictionary or a String.

Upon assignment, ESI creates a variable of the appropriate type depending on the input. Examples:

Example Value =	Type
"foo"	string
"5"	number
"'5'"	string
"'''5'''"	string
"5.6"	string (no reals in ESI)
"4 + $(x)"	number if x is a number, otherwise a string
['1', '2', '3']	list
['a' : '3', 'foo', 'bar']	dictionary

Overview and General Features

What is ESI?

Edge Side Includes (ESI) provides for dynamically generating HTML pages at the edge of the Internet, near the end user. The ESI language is an XML-based markup language that provides the tools to assemble the content dynamically on Akamai's network.

How does ESI use templates and fragments?

A template is a base page of information that consists of common elements such as logo, navigation bars, framework, and other "look and feel" elements of the page. The fragments, which can be HTML or XML, represent dynamic subsections of the page. The template includes ESI code to assemble these fragments. Both the template and the fragments can be cached according to their individual requirements, and assembled dynamically at the edge of the Internet.

How many levels of nested include statements does ESI support?

ESI supports up to five levels of nested include statements: that is, fragments can contain ESI markup that includes other fragments.

The five levels starts with the first one in the template. Example:

- Level 1: template includes fragment1
- Level 2: fragment1 includes fragment2
- Level 3: fragment2 includes fragment3
- Level 4: fragment3 includes fragment4
- Level 5: fragment4 includes fragment5

Is there a limit on the number of successful includes?

ESI provides for up to sixty-five attempted includes per transaction. The transaction begins with the first include statement on the template and encompasses all subsequent includes associated with that template, including the attempts generated in nested pages.

ESI provides the ability to include an alternate object (`alt`) if the primary object (`src`) is not available. If the primary object is fetched and there is no need to attempt the alternate, that is one attempt; if the primary include fails and the alternate is fetched, that is two attempts. When it processes a page, the edge server processes all primary objects before it processes any alternate objects.

Is there a limit on the total size of an ESI page?

The total size of all included objects, including nested objects, for a page, cannot exceed 1 megabyte.

Does ESI provide for custom variables and custom functions?

Yes. The `<esi:assign>` statement allows you to create variables, and the `<esi:function>` statement provides for defining functions. These are called in the way as standard variables and functions.

Does ESI allow for any decision making when building a page?

ESI currently supports conditional processing (`choose/when/otherwise`), Boolean expressions, explicit exception handling (`try/attempt/except`). And it provides mechanisms to define variables, a range of functions, and the ability to use HTTP environment variables and the Akamai GEO data.

Does ESI allow for iterative operations?

ESI provides the statement, `<esi:foreach>`, with a number of keys (number of iterations, the first iteration, etc.), to compose iterations. You can iterate over a collection of items or a range.

Can ESI process SSL requests?

Yes, ESI can be used for pages that are served using the SSL protocol. If the ESI template page is retrieved using SSL, the ESI fragments for the page can only be retrieved using SSL. If the ESI template is not secure, the fragments can be either secure or unsecured.

Can ESI return binary output?

The ESI page must be an ASCII file and can contain HTML or any other form of text encoding as the final output. Dynamic insertion of raw binary data as the final output sent as the response to the client can be tricky, with the following limitations:

- HTTP Byte range GET requests are not supported
- Binary outputting PDF files are not supported
- Only binary outputting gifs, jpgs, jpegs file types are supported as long as the correct HTTP Content-Type header is set.
- The ESI code must be contained all on one line with no spaces between ESI tags. Please take great care to ensure that ESI does not add any new lines, tabs or white space when binary outputting an image.
- In the edge server configuration, the processor type for the `<esi:include>` of the binary object must be set to “none”.

Please consult with your Akamai representative before binary output in an ESI page.

Can ESI be used with double-byte pages?

Yes, ESI supports double- and multibyte character sets. See the *ESI Developers Guide* chapter on Internationalization for more information.

The Language and Syntax

What happens if the ttl attribute in an ESI include statement conflicts with a no-store?

The **no-store** wins, as would a **bypass-cache** attribute. You cannot currently override the **no-store** set in your edge server configuration with a **`ttl`** attribute within an **`esi:include`** tag, but this override function will be included in a future version.

The settings from the **`ttl`** attribute in an **`esi:include`** tag override any other ttl settings from other sources such as the edge server configuration file, the Edge-control header, or HTTP headers.

What's the difference between the alt and except tag?

The **`alt`** attribute of the **`esi:include`** tag may be used to specify an alternate fragment if the primary object is not available. The **`esi:except`** tag performs similar function—that is, it provides alternative processing if the primary objects are not available—but it provides more explicit exception handling in a **`try | attempt | except`** block. For example, you can set different attributes on every **`esi:include`** object in the **`try`** block, whereas the **`alt`** by default uses the same attributes set on the **`src`**, if you placed caching or other attributes on the **`esi:include`**. Also, the **`alt`** always specifies a URI, whereas the except statement is not necessarily a ESI request to include an object.

What happens if the src and alt tags cannot be retrieved?

If the edge server can fetch neither the **`src`** object nor the **`alt`** object, it returns a 404 HTTP error with a simple error message—unless the **`onerror`** attribute is present. The **`onerror`** attribute can be used with an **`src`** only or with both an **`src`** and **`alt`** attempt. If **`onerror="continue"`** is specified and the **`src`** and **`alt`** fail to fetch the object, ESI deletes the include tag and serves the page without any object replacing the include statement.

When **`onerror="continue"`** is set and the fetch fails, the edge server does not serve a default object. Without the **`onerror`** attribute, the edge server attempts to fetch a default object if one is specified in the configuration file. The default object can be processed by ESI. However, if anything goes wrong during ESI processing, the result is that the edge server will send an error to the client. If you choose to use ESI in default objects, the ESI code should be very simple and well-tested.

For more information on error handling, see "Exception and Error Handling" in the *ESI Developer's Guide*. For information on using **`onerror`** inside ESI's explicit exception handling method, see the **`try`** block documentation in the same document.

Can I use a $ in an esi:assign statement when creating a custom variable?

The $ is a reserved character in ESI and it is used to indicate the beginning of a function call or variable reference. It can't be used in assigning the name of a user-defined variable, and it can only be used in the value when: (a) the value is a string, or (b) it is specifically used for its intended purpose in a function or variable reference.

The backslash (\) can be used to escape the $ or other special characters; also, the ESI **$dollar**() function can be used to return a literal $ within an **esi:assign** tag, as can the triple quote, described in the next answer.

Can I use the quote marks, " and ', in an esi:assign statement when creating a custom variable?

The single quote and the double quote are reserved characters in ESI, and normally, they cannot be used in either the name or the value. One exception is that the double quote can be used within the value of a long form **esi:assign** tag.

However, if you enclose a string in three single quotes ('''), a triple quote, you can include any character inside the string, including double quotes, single quotes, and the dollar sign ($). The triple single quote tells ESI to see every character as a literal character. For example:

```
<esi:assign name="var">'''You're up.'''</esi:assign>
```

Note that when you use the three single quotes, do not place them immediately before or after a single quote—that is, don't use four single quotes (''''), since this may cause an error. Adding a single space suffices to correct the situation (' ''').

Can I use a $ or other special characters in an esi:include statement?

No. You can use the **$url_encode**() function to accept and encode a string that may contain certain illegal characters as specified in RFC 1738.

$url_decode() decodes special characters that were encoded from transmitting a URL. Mainly, this is used to decode the value of members of a query string.

Note that the edge server process automatically stores the decoded version of query string values in the QUERY_STRING variable, when referenced by name ($(QUERY_STRING{'foo'})).

Is there a general method to escape special characters in ESI?

Yes, ESI provides the backslash (\) as a general escape function. For example, a \$ yields a literal dollar sign "$".

How Can I include HTML that contains a $ or other special characters?

You can include plain text or HTML that includes the $ or other characters that might otherwise be "illegal" in ESI in a couple of different ways:

- You can include the text as a fragment. That is, put the text into its own HTML file and include it with an **esi:include**.

- You can place the text into a variable and then include the variable. For example:

```
<esi:assign name="html_text">
'''The cars are valued at $12,000, $15,000, and $20,000 respectively.'''
</esi:assign>
```

You could then recall the text with **$(html_text)**.

Can I include flat text in an ESI-code page?

Yes, you can use the **esi:text** statement to include flat text that will not be processed.

Can ESI support regular expression string pattern matching?

The **matches** operators checks to see if an expression contains an *extended regular expression* (ERE); **matches_i** performs a case insensitive evaluation. ESI bases it regular expression processing on the Perl Compatible Regular Expressions (PCRE), described in **http://www.pcre.org/pcre.txt**.

Does ESI support case insensitive substring matches?

Yes. The **has** and **has_i** operators check to see if an expression contains a particular string; **has_i** performs a case insensitive evaluation. These operators are useful in checking for strings in the environment variables. The comparison can be used on a variable or on a part of a variable—for example, a key, a value, or part of a key or value.

How do I test to see if a variable is a Dictionary, String, or List?

You can using string pattern matching see if the variable is in dictionary or list format. Lists in ESI are surrounded by square brackets [ ] and dictionaries are surrounded by curly brackets { }, so that if the strings in your variable do not contain square or curly brackets, you can test to see whether the variable contains the appropriate bracket.

An example snippet from such a test, using the regex substring pattern matching to look for surrounding brackets:

```
<esi:choose>
    <esi:when test="$(value) matches '''^\[\'.*?\'\]'''">
```

You could also check to see if the specific separators used in the list or dictionary, and not used in other strings, exist in the variable.

Does ESI support generating a random number?

Yes. The **$rand**() function generates and embeds a random positive integer between 0 and 99999999, or between 0 and *(n-1)* inclusive when you use the construction, **$rand(n).** The generated number is passed from template pages to fragments, but not from fragments to templates. That is, if you use **$rand**() to generate a number in a parent page, you can use **$last_rand**() to recall the number in the child. You can also use the **esi:assign** tag to define a variable with a value generated by $rand(), and then recall that value.

How can I convert a number to a string?

ESI contains two functions: **$str()** and $int(), to convert numbers to strings and vice versa.

How can I debug or test my ESI code?

The ESI Development Tool provides a way for you to test, view, and debug your web pages containing ESI code. You can debug your production pages, or you can set up a test origin site—a Test Origin Server— and debug the pages on the test site before deployment or modification.

When it processes pages containing ESI code, the edge server will send back debugging information in its reply if a debugging cookie is sent to it. There are different ways to send the debugging cookie. For testing from a browser, Akamai provides a web application, which is available on https://control.akamai.com that will allow you to set the appropriate cookie. You can then simply turn the debugger on and go to the page to debug it.

Can I turn on the debugger conditionally or all-the-time.

For example, could I use the **<esi:debug>** tag inside an **<esi:choose>** statement to only run the debugger when certain conditions are met? This is ***not*** recommended. The recommended way for pages in production is to send the debug cookie if you want to debug.

ESI doesn't know while processing the page if an **<esi:debug>** tag is going to be processed. It is forced to generate and store all the debug messages in a buffer. This generation and storing eats up CPU and memory. Just enabling the tag, even in a conditional statement, will slow down your page. It is much better not to use it at all unless you need to debug the page.

What operations can I use in qualifying a conditional include?

When using the esi:when statement in the **esi:choose** block, you can use any legal ESI expression or compound expression, which includes the use of variables, match operations, Booleans, and so forth. There is a chapter on legal operations and expressions in the *ESI Developer's Guide*. For example, you could construct a choose statement as follows:

```
<esi:choose>
    <esi:when test="!$exists($(HTTP_COOKIE{'UserInfo'})) | !($(HTTP_COOKIE{'UserInfo'})
        matches '''UserId=[0-9]''')">
        [include some file]
  </esi:when>
  <esi:otherwise>
        [include some other file]
  </esi:otherwise>
</esi:choose>
```

The **<esi:when>** statement shows the use of a compound expression using functions and variables and also combining several different operators. It uses the Booleans **!** (not) and Or (**|**), and regular expression (regex) match (**matches**). It uses the variable and variable substructure **$(HTTP_COOKIE{UserInfo})**. And it uses the function **$exists** to check to see if the cookie exists.

You can read the statement as follows: IF the cookie does NOT exist, OR if the cookie exists but does NOT contain the regex match, then include "some file." OTHERWISE, include "some other file."

Variables and HTTP Integration

How can I perform geographic-based content targeting with ESI?

In ESI, the GEO variable can be made available to ESI code that represents an end user's geographic location. This information can be included within conditional statements to react to a user's location in addition to other criteria when building a page.

Akamai's EdgeScape and EdgeScape Pro provide the GEO variable information to ESI. Accessing GEO key values requires enabling an option in your edge server configuration, in addition to purchasing ESI. If you are unsure whether or not you have access to GEO data from ESI, contact your Account Manager. For a complete explanation of GEO, its keys and potential results, see the *EdgeScape and EdgeScape Pro Users Guide*, the chapter, "Implementing the EdgeScape Pro Service APIs," and the appendices. Note that ESI currently does not provide the "company," "device-type," or "domain" data.

What HTTP environment variables does ESI support?

With some limitations and with requirements discussed here, HTTP headers are honored in requests from clients and in responses from the origin. HTTP headers can be used as variables with the following conversion structure:

- The term "HTTP_" is prefixed to the name.
- The variable must be in upper case.
- Dashes (-) are replaced with underscores (_).

For example, Accept-encoding becomes HTTP_ACCEPT_ENCODING, Cookie becomes HTTP_COOKIE, and Host becomes HTTP_HOST.

How Can I Test a Variable to See if It's an Integer

A quick way to check to see if a variable represents an integer in ESI would be to see if **`$int()`** and **`$str()`** are inverse functions on that variable. That is, if:

```
$(var)==$str($int($(var)))
```

Then **`$(var)`** represents an integer.

Can ESI pages set Cookies?

Yes, the ESI **`$add_header()`** function can be used to set a Set-Cookie header or other HTTP headers. ESI will not do any conflict verification, so you'll need to make sure you've set the proper cookies in the fragments and template page. Set-Cookie headers set on fragments or templates are collected and sent to the client browser by default *if* the fragment or template with the Set-Cookie header is configured as a **no-store** object. See the *ESI Developers Guide* for more information.

Can ESI remove Cookies or other HTTP headers from the response?

Currently, ESI cannot delete a Set-Cookie header or any other HTTP response header. However, deleting HTTP client response headers is supported within your edge server configuration file. Contact your Akamai representative for more details.

Can ESI set the HTTP Content-Type Response header?

By default, ESI uses the HTTP Content-Type response header returned by the origin server of the ESI template page in the final response sent to the client. The ESI **`$add_header()`** function can be used to set HTTP Content-Type response header. For example:

```
$add_header('Content-Type', 'text/plain')
```

Note, the **`$add_header()`** function replaces the existing Content-Type header if one exists and does not add an additional Content-Type header since only one Content-Type header is valid.

Does ESI support POST requests? POST responses?

Requests: You can set POST requests with the **`post`** attribute of the **`esi:include`** statement, and you set the data for the POST with the **`entity`** attribute. Variables can be used to create the value of these attributes.

Responses: Origin servers can respond to client POST requests through ESI, providing that ability is enabled in your edge server configuration file. POST processing is enabled by default but can be disabled should you choose to do so.

The values contained in the POST are available in the ESI document as a dictionary under the name POST, thus making the POST data available under the variable call, $(POST{'sub1'}), where sub1 is the name key in a name:value pair. See the *ESI Developers Guide* for more information.

How does ESI support HTTP redirects?

ESI contains a function, **set_redirect**(), that provides for generating HTTP redirect responses to the client. ESI also supports chasing a redirect returned by an **`esi:include src`** or **`alt`** URL, if this option is enabled in your edge server configuration file. By default, chase redirect is turned off. ESI does not support serving a redirect returned by an **`esi:include`** if chase redirect is not setup in your edge server configuration file. In the latter situation ESI generates an error.

How can I set up different caching properties for the src and alt objects?

The best way to do this is to use the **`esi:try`** block instead of the **`alt`** attribute, since the **`esi:try`** allows you to do everything you can do with the **`alt`**, and more.

However, if you need to use the **`alt`**, you can set different caching properties by *not* using either the **`ttl`** or the **`no-store`** attributes in the **`esi:include`** statement, and instead specifying the caching properties for the objects in the edge server configuration file.

How can downstream cache-ability be set for ESI pages?

By default, the root response header sent downstream by ESI will be set to no more than the least **ttl** (time-to-live) value of all objects associated with the template. In other words, the resulting ESI page will be cached downstream for a period defined by the shortest **ttl** of all the objects that compose the page. If any object is no-stored or bypass-cached, the downstream page will be no-stored or bypass-cached.

Also, downstream cache-ability of your ESI pages can be specified in your edge server configuration file, or by Edge-control response header. Contact your Akamai representative for further details.

Third, the ESI language provides a function that explicitly provides for purging and busting downstream caches—the **`$add_cachebusting_header()`** function. There is a known bug, to be fixed in the next release, that can prevent this function from working properly. Currently the bug can be worked around by turning on the "Honor Cache Control" attribute in your edge server configuration.

Finally, downstream cache-ability can be specified in an ESI page using the **`$add_header()`** function. For example:

```
$add_header('Expires', $http_time($time()))
$add_header('Cache-Control', 'max-age=0, no-cache, no-store')
$add_header('Pragma', 'no-cache')
```

Integration and Implementation

What should I know about CNAME'ing the site when implementing ESI?

As you switching to Akamai and ESI, local DNS servers might not refresh their records in a timely way; some client requests may get resolved to Akamai servers while others are resolved to the origin server. One solution is to setup your edge server configuration for a origin server different from the existing origin. Another solution would be to setup a virtual host on the existing origin server, then use a host header such as origin.xyz.com that only edge servers will send for the ESI enabled site.

Can I specify edge server configuration settings within ESI?

The following configuration settings can be specified as attributes to the **`esi:include`** tag:

- The optional maxwait specifies a time-out period, in milliseconds, for the edge server to wait for the **`src`** or **`alt`** to complete the fetch successfully.
- The optional **`ttl`** specifies a maximum time interval, in seconds, minutes, hours, or days, for the fetched object to reside in cache before the edge server revalidates that the object has not changed.
- The optional **`no-store="on|off"`** turns on or off the instruction to the edge server not to cache the object.
- The type of dynamic content assembly (**`dca`**) processing to use—ESI, Akamaizer, or none.

Examples:

```
<esi:include src="frag.html" maxwait="2000" ttl="2h"/>
<esi:include src="frag.html" maxwait="2000" no-store="on"/>
```

Can a default object be ESI processed?

Yes—*if* you have specified a default object in your edge server configuration file, and if you have specified that this object be ESI-processed. However, if anything goes wrong during ESI processing, the result is that the edge server will send an error to the client. If you choose to use ESI in default object, the ESI code should be very simple and well-tested.

If you use both a fail-action default object and an ESI **`alt`** attribute or the **`esi:attempt`** statement (in a **`try`** block), ESI will normally use the fail-action object, using the **`alt`** object only if the fail-action itself fails. However, you can set your edge server configuration to allow ESI to use the **`alt`** or **`attempt`** processing in preference to the default object. If a default object is configured on **alt** or **except** objects, it will be invoke should those fail.

Do you have any integration options for converting my JSP/ASP/SSI/CSS pages to ESI?

Currently, an Akamai representative who is very familiar with these environments will work with you to determine the best approach and integration path to convert your site to take full advantage of ESI. Every ESI user has unique needs and requirements, and each implementation is different, so that a default integration approach is not appropriate.

Does ESI work with my environment (ATG, Allaire, SSI, etc)?

ESI will work in any environment, as ESI is merely additional XML tags embedded within the HTML delivered to Akamai. The environment that generates the HTML may vary, but ESI will work with any such environment.

What is the performance overhead in ESI processing?

ESI can improve site performance by caching the objects that comprise dynamically generated HTML pages at the edge of the Internet, close to the end user. ESI allows for dynamic content assembly at the edge.

There is no significant overhead in processing ESI pages. The only performance issues occur when the origin site is performing poorly, and non-cacheable elements or items that are not yet in cache must be inserted. Essentially, the slowest piece of the page is the bottleneck—ESI processing itself does not degrade performance in this case.

Can I turn on ESI processing for an entire site?

It is not recommended to turn on ESI processing for all objects on the web site. Instead, ESI processing should be turned on for only the objects on the site that will contain ESI code. Turning on ESI processing for all content on a site will cause errors if the site contains binary objects that cannot be processed by ESI.

What if I want to use ESI, but not have to worry about re-coding my site if I turn off ESI?

You can easily code pages to use ESI without the need to re-code the site if you turn off ESI processing. ESI supports functionality that allows the ESI code to be hidden (as commented text) to a user's browser if Akamai is not involved. ESI also allows you to insert HTML that will be processed by a user's browser if ESI is off, but removed by Akamai if ESI is on.

Index

Symbols

$ 6
$int() 8
$str() 8
“ 7
‘ 7

A

add_cachebusting_header 12
add_header() 10, 11
Allaire 14
alt 6
ASCII 5
ASP 13
assign statement 4
assign tag 7
ATG 14

B

backslash 7
binary includes 5
Booleans 9
bugs 3
bytes total 4

C

Cache-Control 12
caching 6, 12, 13
CGI variables 10
choose 5, 9
CNAME 13
composing ESI pages 4
conditional include 9
conditional processing 5
configuration 13
Content-Type 11
Cookies 10, 11
CSS 13
custom functions, variables 4

D

debug 9
decision-making 5
default object 13
delete Set-cookie header 11
dictionaries 3
dictionary 8
DNS 13
double-byte 5
downstream cache-ability 12

E

Edge Side Includes 4
EdgeScape 10
edge server configuration 13
environments 14
error handling 6, 13
escaping special characters 7
ESI 4
ESI Development Tool (ESID) 9
ESI fragments and fail-action 3
eval 3
except tag 6
Expires 12
extended regular expression 8

F

fail-action 3
foreach 5
fragments 4
function statement 4
functions, user-defined 4

G

GEO variable 10
gifs 5

H

has 8
HTTP environment variables 10
HTTP headers 11
https 5

I

implementation 14
include limits 4
include statement 6
include statement encoding 7
includes 5
integers, testing for 10
internationalization 5
issues 3
iteration 5

J

jpegs 5
JSP 13

K

known issues 3

L

limits, includes 4
limits, size of page 4
lists 8
lists and strings 3

M

matches 8
max-age=0 12

multibyte 5

N

nested include statements 4
no-store 6
no-store="on|off" 13

O

onerror 6
otherwise 5
overhead 14

P

page size 4
performance overhead 14
POST 11
Pragma 12

Q

quote marks 7

R

random numbers 8
recursed includes 4
redirects 11
regular expression 8, 9
remove headers 11

S

secure pages 5
Set-Cookie 10
src 6
SSI 13, 14
SSL 5
string pattern matching 8
strings and lists 3
substring matches 8

T

templates 4
testing for dictionaries/lists 8
total bytes 4
TRAFFIC_INFO 3
try block 6
ttl 6

U

URL encoding 7
url_encode() 7
user-defined 4

V

variables, testing for lists or dictionaries 8
variables, user-defined 4

W

when 5, 9

X

XML 4