Edge Side Includes (ESI) Technical Q&A

What is Edge Side Includes (ESI)?
Edge Side Includes is a simple markup language used to define Web page fragments for dynamic assembly at the network edge. It also contains a content invalidation specification for transparent content management across Content Delivery Networks, Application Servers and Content Management Systems. By using ESI, e-businesses can reduce the complexity and infrastructure requirements of developing, deploying and maintaining highly scalable and reliable Web sites and Web-based applications.

What problems does ESI solve?
Most Web pages today are dynamically generated before delivery to the browser. Yet the computational overhead associated with building personalized pages on-the-fly for thousands of concurrent users can result in increasing delays and failures in data delivery. Caching products and content delivery network (CDN) services are often touted as solutions to the problem of slow Web sites. However, dynamically generated Web pages typically contain a mix of personalized and non-personalized content, making them non-cacheable by legacy caching products and content delivery services.

Edge Side Includes (ESI) enable dynamic assembly of Web pages composed of both cacheable and non-cacheable page fragments. As a result,

- e-businesses can now develop highly dynamic Web-based applications that are assembled at the edge of the data center, or at the edge of the Internet, for *improved performance*;
- the aggregation and assembly of content in edge servers dramatically *reduces the cost* of infrastructure required to deliver fast, scalable and fault-tolerant applications; and finally,
- the adoption of an *open specification* guarantees interoperability of ESI-compliant applications across multiple vendor implementations.

Why is it more cost-effective to assemble and deliver content in an edge server tier than in traditional business logic tiers?
Edge servers are highly efficient content delivery mechanisms. Unlike a JVM, an edge server does not execute business logic by instantiating Java classes. No computation, formatting or byte copies are required to deliver a page out of the cache. Thanks in large part to a short code path -- hash on the request, assemble the fragments and serve the document -- an edge server need not run on expensive hardware to provide excellent performance.

Does ESI require modifications to existing applications?
ESI does not require extensive modifications to existing applications. Instead, it provides Web developers with an easy way to modularize and tag their Web pages for assembly on the edge.

Does ESI depend on a particular programming model?
ESI does not depend on any particular programming model. It can be used with HTML, XML, and any Web programming environment (JSP, ASP, PSP, PHP, ColdFusion, etc.).

What is Edge Side Includes for Java (JESI)?
Edge Side Includes for Java provides extensions to Java that make it easy to program JSPs using ESI. JSPs are server-side software modules that produce final user interface by linking dynamic content and

static HTML through tags. JESI is a specification and custom JSP tag library that developers can use to automatically generate ESI code. Even though JSP developers can always use ESI, JESI provides an even easier way for JSP developers to express the modularity of pages and the cacheability of those modules, without requiring developers to learn a new programming syntax.

How do Web developers manage the consistency of dynamic content using ESI?
ESI provides a number of ways to manage the consistency (freshness) of cached objects, including the ability to define expiration policies. Due to the unpredictable nature of frequently changing content, ESI also includes a content invalidation specification. The invalidation specification is used by syndication servers, content management systems, databases, custom scripts, application logic, etc. to send HTTP-based invalidation messages to a local cache and/or delivery network. The invalidation message tells the cache or delivery network to overwrite the metadata associated with particular ESI objects residing on edge servers. A combination of the two approaches (expiration, invalidation) and some simple fragment naming conventions can be used to ensure that the content delivered from the edge is always consistent.

How does the ESI invalidation specification compare to Web Cache Invalidation Protocol (WCIP)?
WCIP and similar solutions try to solve a different problem than ESI invalidation. Although both are a means of communicating invalidations (and possibly other information) from a Web site to caches and CDNs, they have different scopes and applications.

WCIP allows a large number of clients to subscribe to a channel, on which they'll get invalidations and other kinds of updates. The primary problems that WCIP addresses are a) scaling to a very large number of clients and b) making the service to all of those clients reliable.

ESI invalidation, on the other hand, has a one-to-one relationship; there is no channel mechanism. Instead, an invalidation message passes from the origin server directly to the local cache or CDN. This is more appropriate for sending invalidations to a small number of servers, or when the origin server doesn't have responsibility for propagating the message (such as when a CDN is used).

Both WCIP and ESI invalidation might share a common payload (that is, the invalidation message), but WCIP attempts to address much more difficult problems, and is still experimental. ESI invalidation is simpler, more robust, and more appropriate for sending invalidation messages to a known party.

Can ESI be used with highly dynamic and personalized Web sites?
ESI can be used with a Web site regardless of how dynamic it is. Even if the template page is unique to each user and has to be generated every time, most data on the page can often still be cached and assembled on the edge. Therefore, ESI will improve site performance even in this case.

Does ESI enable personalization on the edge?
ESI includes a capability to conditionally include page fragments depending on cookie values and such environment variables as HTTP_USER_AGENT, HTTP_REFERER, QUERY_STRING, etc. This can be a powerful way to assemble personalized pages without incurring round trips back to the origin Web server.

How does ESI improve fault tolerance?
ESI has several features designed to improve site fault tolerance. It is possible to associate a default object with every fragment required to build the Web page. It is also possible to provide programmatic ways to handle exceptions arising from a particular origin Web server being unavailable.

Can a Web page be marked up in such a way that it can be rendered both with and without ESI processing?
Yes. ESI enables Web developers to create Web pages that render correctly both with and without ESI processing. ESI directives can be hidden from the browser, and appropriate default content can be displayed if ESI processing is unavailable. These features are especially useful while Web developers are making the transition to using ESI.

Can HTTP proxy servers interpret ESI?
ESI is well suited for interpretation by proxy servers.

Does ESI address any security and access control issues?
A future version of ESI will include a specification for doing fragment-level access control at the edge.

Does ESI address XSLT?
XSLT support will be added to a future version of ESI. When it becomes available, Web developers will be able to utilize server-side caches and content delivery networks to cache the XML and XSLT files and perform content transformation on the edge.

***